MOTION PLANNING AND CONTROL FOR AUTONOMOUS VEHICLES IN URBAN PARKING SPACES

Gireesh Suresh
A.James Clark School of Engineering
University of Maryland
College Park, USA
gireesh@umd.edu

Manohar Anumolu
A.James Clark School of Engineering
University of Maryland
College Park, USA
manoharanumolu.96@gmail.com

***Abstract*— Automatic Parking is an autonomous car maneuvering system that moves a vehicle from a traffic lane into a parking spot to perform parallel parking. The automatic parking system aims to enhance the comfort and safety of driving in constrained environments where much attention and experience is required to steer the car. The parking maneuver is achieved by means of coordinated control of the steering angle and speed which considers the actual situation i.e., the free spaces and the obstacle spaces in the environment to ensure collision-free motion within the available space. The path shape required for a parking maneuver is evaluated from the environmental model, generating a fifth-order polynomial, the corresponding control commands are selected and parameterized to provide motion within the available space. In real-time application, the commands are executed by the car servo-systems which drive the vehicle into the parking place.**

Keywords—collision-free; parking maneuver; path planning; automatic vehicle parking; advanced driver assistive system[ADAS].

I. Introduction

Parking is considered to be one the most challenging tasks where the vehicle must be moved backwards in narrow spaces, with high probabilities of collision. The complexity lies in the generation of a suitable parking motion, the simultaneous control of vehicles longitudinal and lateral motion and the avoidance of obstacles in the environment. Due to Modern vehicle design considerations, safety requirements and aerodynamics, the visibility of a driver is highly impacted, leading to collateral damage while engaging in parking maneuvers. This problem is addressed using AVM (Automated Valet Maneuvering) by providing fully automated parking functions for parallel and perpendicular parking. To be commercialized, the automatic parking must fulfill the expectations of the drivers with a fast, predictable and cost-effective solution to the vehicle maneuvering problem. Nevertheless, one requirement is to be affordable, which implies low computational cost and few sensors. In this paper, we consider the parallel and perpendicular parking problem for ease of vehicle maneuvering. [2]

II. RELATED WORK

The objective of this paper is to provide automotive assistance/planning assistance while engaging in parking maneuvers. Before engaging in the vehicular dynamics and control, it is of primary concern to generate a feasible and cost-effective path using available environment data to proceed in the parking maneuvers. The path is a sequence of vehicle configurations containing the positional and orientation parameters of the vehicle at discrete intervals of time during the motion sequence. It is important that the path has to be feasible enough for a car due to the car's dynamic constraints. For decades, many researchers have been working on automatic parking path planning problems and have come up with all kinds of path geometries, depending on different mathematical models.

Some common proposals for path-computation between the initial and final states are path curves generated with circular arcs and straight lines [3], clothoid curves [4], arc tangent curves and cubic polynomial curves, Bezier curves, composite functions containing one or more of the above mentioned path geometries, triangular functions [5],etc. Research results have proved that the path with straight line segments combined with circular arcs of maximum curvature or minimum radius has the shortest distance between two configurations [6]. Some others include using a combination of multiple path primitives including straight lines, circular arcs, and continuous curvature curves (CC Turns) [7], [8] which were used for path planning. However, the main drawbacks in such algorithms include the discontinuities at the critical transition point which lead to sub-optimal vehicle configurations at some defined, given intervals of time (these discontinuities force the car to stop and reorient itself for the next computed arc/curve trajectory before proceeding forward). These kinds of paths lack 'smoothness' and brings about extra effort and unnecessary stop-points during the computed vehicular motion. In this project, the primary method of path generation is by using a fifth-degree polynomial curve to compute the path. One major advantage is the lack of discontinuities during path generation. Since the initial and final conditions are provided with regards to the parking maneuvers, it is in our best interest to take a higher-degree polynomial for our implementation, with a polynomial of order *n* =5, as the bare minimum [9]. Higher-order polynomials can be used, at the cost of increased computational analysis and possible extraneous computations. The motion planning with algorithms remains an open problem with multiple solutions. In this paper we present the implementation of one adaptive method i.e., to plan the expected motion and path first using a 5th order polynomial, and then send the command to the cars servo-controller. In this method, a new parking curve is generated for each successive step depending on the new initial conditions, leading to the determination of the next move. An important point to be noted is that this project deals with the computation of smooth continuous curved paths and does not involve any cost optimization, reactive collision detection, machine learning paradigms to achieve our end-objective, i.e., generation of a smooth continuous path to facilitate automated parking actions in a given vehicle.

III. PROBLEM FORMULATION

The main objective of this work is to test and implement a motion planning algorithm and the motion control that will facilitate parking in urban Spaces. This includes, but not limited to perform parallel, angular and perpendicular parking maneuvers for an autonomous vehicle (the secondary target being a car with differential drive steering dynamics utilized in our model, but the priority objective will be the actual path generation with a solid having dimensional-constraints engaging in a parallel parking maneuver) in a simulated environment and further extend this to perpendicular and angular parking upon completion of the priority task.

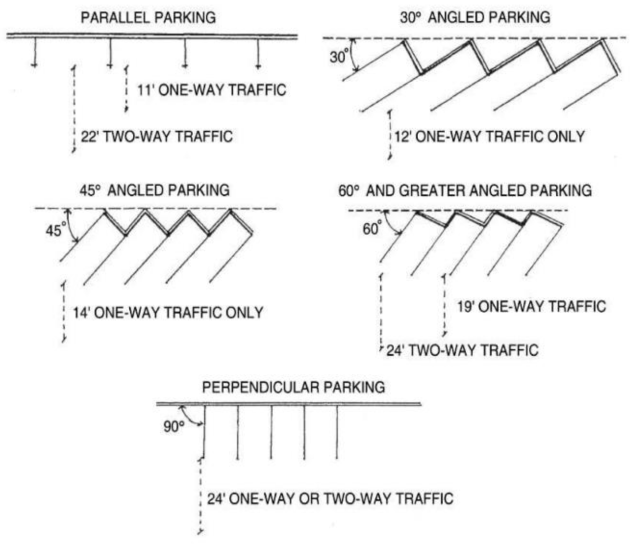

Fig.1 Types of Parking found in Urban Spaces [10]

IV. VEHICULAR KINEMATICS

Due to the nature of the object engaging in these maneuvers, there are certain kinematic and dynamic constraints imposed on the vehicle. In this project, the test run would include a vehicle with no such constraints and merely a solid rectangular block, but after testing the planning algorithm on it, the actual vehicle being modeled is a simple differential drive model, whose parameters are given below.

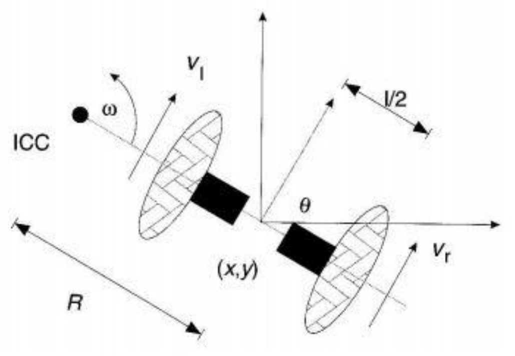

Fig.2 Differential Drive Mechanism [13]

Many mobile robots use a drive mechanism known as differential drive. It consists of 2 drive wheels mounted on a common axis, and each wheel can independently being driven either forward or backward. While we can vary the velocity of each wheel, for the robot to perform rolling motion, the robot must rotate about a point that lies along their common left and right wheel axis. The point that the robot rotates about is known as the ICC - Instantaneous Center of Curvature. [13]

By varying the velocities of the two wheels, we can vary the trajectories that the robot takes. Because the rate of rotation ω about the ICC must be the same for both wheels, we can write the following equations (1*A)*:

$$\begin{aligned}\omega\,(R + l/2) &= V_r\\ \omega\,(R - l/2) &= V_l\end{aligned}$$

where *l* is the distance between the centers of the two wheels, V_r , V_l are the right and left wheel velocities along the ground , and *R* is the signed distance from the ICC to the midpoint between the wheels. At any instance in time we can solve for *R* and ω (1*B*) :-

$$R = \frac{l}{2}\frac{V_l + V_r}{V_r - V_l}\,;\ \omega = \frac{V_r - V_l}{l};$$

Assume the robot is at some position (*x*, *y*), headed in a direction making an angle θ with the X-axis. We assume the robot is centered at a point midway along the wheel axle. By manipulating the control parameters V_l , Vr, we can get the robot to move to different positions and orientations. The forward kinematics equations are as follows (1*C*):

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}$$

where the *ICC* is given by (1*D*) :-

$$ICC = [x - R\,sin(\theta)\,,\, y + R\,cos(\theta)]$$

In general, we can describe the position of a differential drive robot capable of moving in a particular direction Θ(t) at a given velocity V(t) as (1*E*):

$$x(t) = \frac{1}{2}\int_0^t [v_r(t) + v_l(t)]cos[\theta(t)]dt$$

$$y(t) = \frac{1}{2}\int_0^t [v_r(t) + v_l(t)]sin[\theta(t)]dt$$

$$\Theta(t) = \frac{1}{l}\int_0^t [v_r(t) - v_l(t)])dt$$

A differential drive robot imposes what are called non-holonomic constraints on establishing its position. For example, the robot cannot move laterally along its axle. A similar nonholonomic constraint is a car that can only turn its front wheels. It cannot move directly sidewise, as parallel parking a car requires a more complicated set of steering maneuvers. So we cannot simply specify an arbitrary robot pose (x, y, θ) and find the velocities that will get us there.[13].For the special cases of $V_l = V_r = v$ (robot moving in a straight line) the motion equations become (1*F*):

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + v\,cos(\theta)\delta t \\ y + v\,sin(\theta)\delta t \\ \theta \end{bmatrix}$$

If $V_r = -V_l = v$, then the robot rotates in place and the equations become (1*G*):

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + 2v\delta t/l \end{bmatrix}$$

The additional parameters mentioned are the state derivatives, which are the linear velocities in the (*x*,*y*)-directions, '*v*' is speed at front axle '*L*' is defined as the wheelbase [13]

A differential drive robot cannot move in the direction along the axis - this is a singularity. Differential drive vehicles are very sensitive to slight changes in velocity in each of the wheels.[13]

V. PATH PLANNING

A. Generating a Reference Curve

The goal is to compute the path for an effective parking maneuver in a 2-dimensional coordinate system. The mid-point of the rear-wheel axle is taken as the 'center' of the car and the official point of interest with respect to our calculations, i.e., each (*x, y*) dot on the curve represents the midpoints of the car's rear-axle. The cars velocity and orientation are assumed to be parallel to the parking lot and approaches to zero when it is near to the goal point, the curve's slope *y' and* curvature *y''* at the goal point (which is assumed to be the origin (*0,0*) are both zero. The above initial conditions are taken and tabulated below for further perusal. A fifth-order polynomial and their derivatives are setup for the path curve [12]: -

$$y(x) = ax^3(x-x_0)^2 + bx^3(x-x_0) + cx^3 + dx^2 + ex + f \quad (2)$$

$$\begin{aligned} y'(x) &= 3ax^2(x-x_0)^2 + 2ax^3(x-x_0) \\ &\quad +3bx^2(x-x_0) + bx^3 + 3cx^2 + 2dx + e \end{aligned} \quad (3)$$

$$\begin{aligned} y''(x) &= 6ax(x-x_0)^2 + 12ax^2(x-x_0) + 6bx(x-x_0) \\ &\quad +2ax^3 + 6bx^2 + 6cx + 2d \end{aligned} \quad (4)$$

Applying the initial and final condition information we have generated/assumed in Table 1 to (2), (3), (4) and solving for the '*f*', '*e*' and '*d*' parameters of the given polynomial, we see that they are all zero. With this, (1) can be simplified to:

$$y(x) = ax^3(x-x_0)^2 + bx^3(x-x_0) + cx^3 \quad (5)$$

	Initial Condition	Final Condition
Path Curve y(x)	$y(x_0) = y_0$	$y(0) = 0$
Slope of the Path Curve y'	$y'(x_0) = m$	$y'(0) = 0$
Curvature of the Path Curve y''	$y''(x_0) = n$	$y''(0) = 0$

Table.1 Initial and Final Conditions for Path Planning [12]

Now, applying the initial conditions again on (3), (4), (5), we get the following values for '*a*', '*b*' and '*c*':

$$a = \frac{n}{2x_0^3} + \frac{3m}{x_0^4} + \frac{6y_0}{x_0^5} \quad (6)$$

$$b = \frac{m}{x_0^3} - \frac{3y_0}{x_0^4} \quad (7)$$

$$c = \frac{y_0}{x_0^3} \quad (8)$$

To visualize the path equation, consider Fig.4, which shows the parking reference path based on the fifth-degree polynomial equation *y*(*x*) when $x_0 = 2$, $y_0 = 1$, *m*=0, *n*=0.

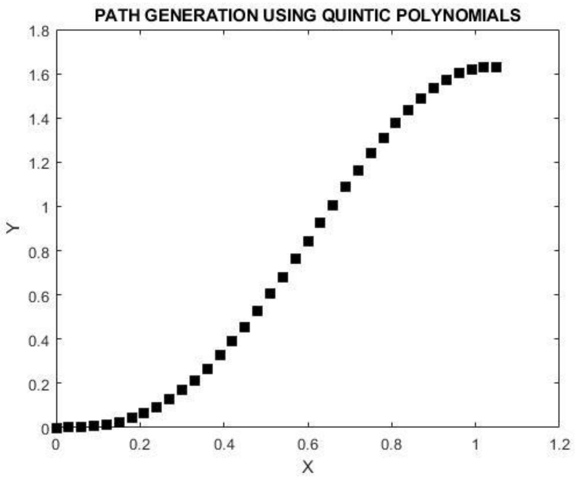

Fig. 4. A Sample Plot at $x_0 = 2$, $y_0 = 1$, *m*=0, *n*=0.The markers represent the path to be taken at that state (x_0 y_0 *m n).*

B. Automatic Parking Algorithm

Using the above-mentioned information and data, an adaptive algorithm can be proposed, which recalculates the path curve, slope and curvature of the path curve over multiple iterations. At each new iteration, new initial conditions are observed, which leads to a path curve to determine the next motion command. As every iteration comprises of fresh initial conditions, accumulations of errors such as 'radical imprecision', 'angular uncertainty' and other forms of error is minimized [12]. This computation, however, requires that the precise location of the car can be observed at any given interval of time, (for example, using motion sensors, video surveillance etc.) and also the value of the slope of the path curve. In this project, we will be using motion sensors attached to the car to provide us with the above information. Here in this algorithm, we simulate the results from the sensors by using the 'normrnd' function, to incorporate errors generated from the sensors' detection. [12]

As we can see, the path curve is computed adaptively, with increasingly larger number iterations (increased 'resolution' of sorts) giving greater accuracy during path generation, as the fifth-order polynomial for the path is computed every single instant of time. At the beginning, the value of '*n*', which is the initial curvature is set to be zero, since the car is assumed to be moving in a straight line at the initial point of the parking path. The rest of the main algorithm is then contained in a loop, containing six separately defined functions [12], the functions being the following:

1. Sensor_Turn: Sensor used after turn motion to get x_0, y_0, m.
2. Calc_abc: Recalculation of a, b, c in (6), (7), (8).
3. Straight: Get new x, y from x_0, y_0.
4. Sensor_Straight: Sensor used after straight motion.
5. Calc_n: Calculating new curvature.
6. Turn: Calculate new slope.

First, sensors will detect the current data for x_0, y_0 and m. a, b and c are calculated afterwards by using equations (6), (7) and (8). With this, a new parking curve is calculated at the current point and simultaneously the car moves forward. Next, sensors are going to detect and collect the new values for x_0, y_0, m after the previous small straight movement. The new value for n is calculated by using equation (1). After that, the car gets a new turn command according to n and makes a turn motion. Therefore, a single loop contains two motions, with one turn and one straight movement. For successful vehicular locomotion, it is of paramount importance to ensure that the path and state information is computed correctly and with respect to the most recent position, slope and curvature parameters.

C. Minimum Horizontal Distance Problem

When the initial vertical distance y_0 is fixed (for practical purposes, when shifting from a traffic lane to a parking lane, it usually is a constant distance), there exists a lower limit on the horizontal distance x_0 for the algorithm to perform successfully. The problem also depends on the dimensional properties of the vehicle, conditions of the road etc. As before, we assume that the car's initial velocity is parallel to the parking area. We assume the initial conditions as mentioned before, (Refer Table.1), with the addition of one additional constraint, i.e., $|y''| \leq$ M, where M is the upper limit of the curvature of the quantic polynomial used for path generation. We need to find x_1 such that $y''(x_1)$ =M, for which we set y'''= 0.

$$10ax^2 - (8ax_0 - 4b)\,x + ax_0^2 - bx_0 + c = 0 \qquad (9)$$

where, the values of a, b, c are given in (6), (7), (8) when m = 0, n = 0. Substituting those values: -

$$6x^2 - 6x_0x + x_0^2 = 0 \qquad (10)$$

Solving, and finding the smaller solution: -

$$x_1 = \frac{3-\sqrt{3}}{6}x_0 \qquad (11)$$

Substituting (11) in (4), d = 0 (Initial Conditions) and equating it to 'M', i.e., $y''(x_1)$ =M, we get the minimum horizontal distance as:-

$$x_0 = \sqrt{\frac{12\sqrt{3}y_0}{\mathrm{M}}} \qquad (12)$$

when m = 0, n = 0.

From the result shown above, we can see that when the upper limit of the curvature M becomes larger, the minimum distance turns out to be smaller. If the upper limit of curvature tends to infinity, the minimum horizontal distance approaches to zero. But M is also bounded by the physical constraints imposed by the vehicle, as its radius or curvature cannot exceed a certain limit. Therefore, x_0 exists well above zero.

VI. SIMULATION

Using the above information, a local planner for parallel parking was implemented. For simulation we have first used MATLAB in order to plot the path and get the path data as follows: The Initial Point was given as (15,10) and the goal point is obviously O (0,0), for the algorithm requirements. The path was simulated three times, in order to display the path distortion for variance in sensor detection of the real-time location of the car. This shows that the algorithm is highly robust to sensor noise, due to iterative calculation of the initial and final conditions.

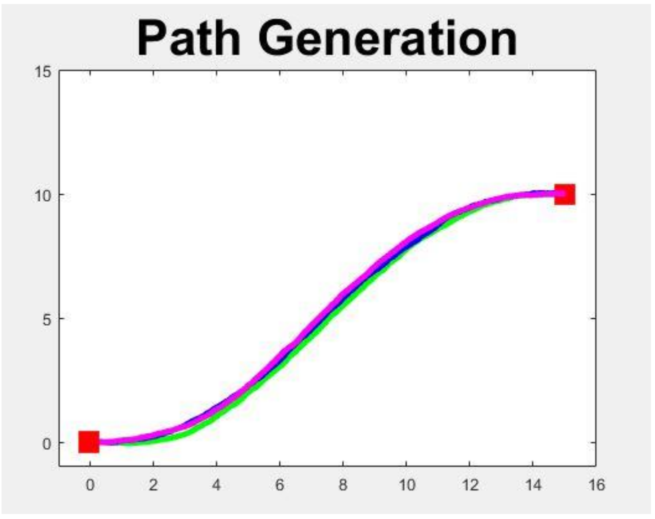

Fig.5 Path Generation for Parking Maneuver

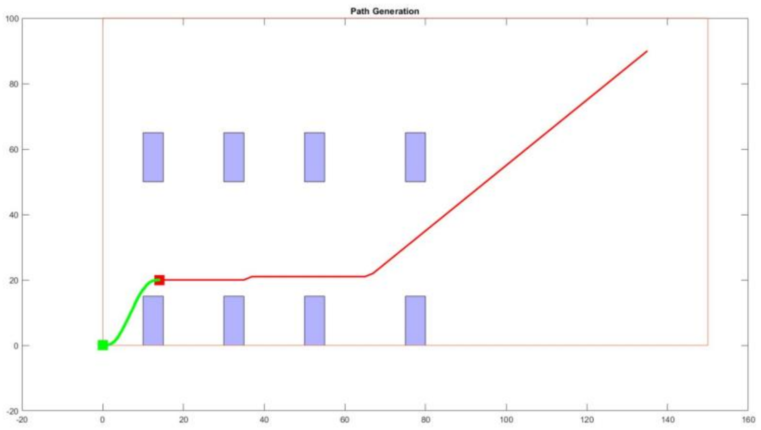

Fig.6 Complete Path from simulated parking lot to the goal point.

This planner, which is the 'local' planner, is used to engage the parking maneuver. However, in order to get the robot to the initial position, i.e., a good initial position which satisfies the minimum horizontal distance requirements shown in section *C*, we use another planner for navigation in the parking lot. We have used the A* Algorithm to move the robot to the required position, from where the local planner kicks in.

Simulation was also done in Gazebo after creating a parking lot world, using an online floor plan as reference [14]. The building editor program in Gazebo was found to be very useful in this endeavor, however, it is only used for model building. We then take the resulting model data and use it to realize a world with the parking lot.

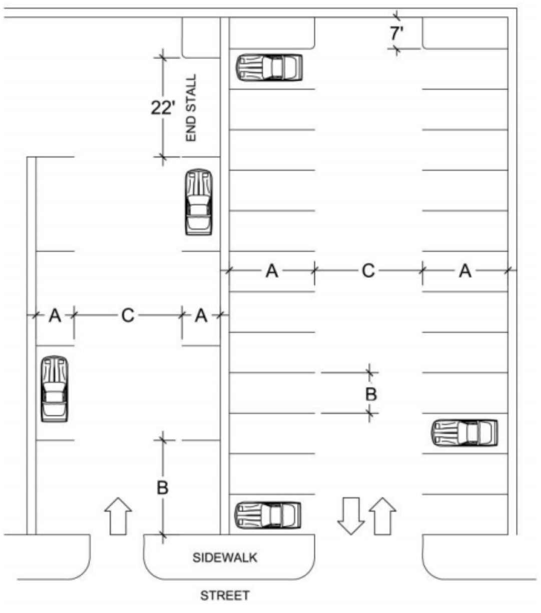

Fig.8 Parking Floor Plan

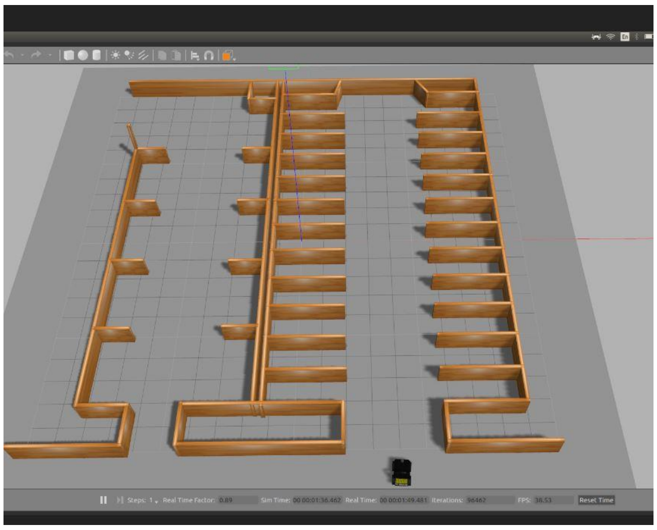

Fig. 9 Parking Lot World in Gazebo

After building the world, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

The robot we have used is the 'Husky' , a rugged, outdoor-ready unmanned ground vehicle (UGV), suitable for research and rapid prototyping applications. Husky fully supports ROS and all the packages are available online. [15]

The following figure shows the generation of the 2-D Occupancy Grid by manually moving the robot.

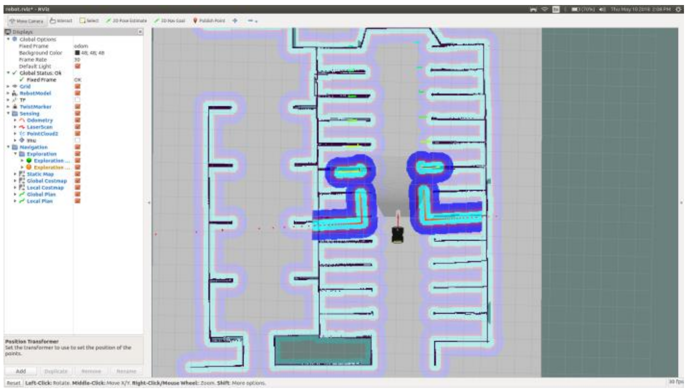

Fig.9 2-D Occupancy Grid

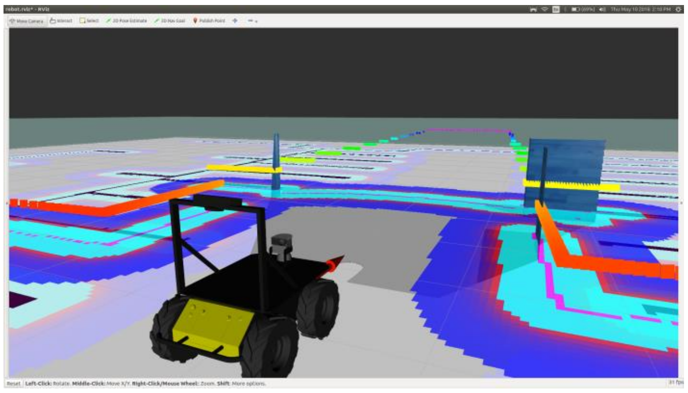

Fig.10 Husky moving around in parking lot

VII. RESULTS

This report presents a path planning method and designed an automatic parking algorithm based on a paper [12]. A fifth-degree polynomial reference path is calculated and applied to the automatic parking algorithm. The advantage of a fifth-degree polynomial is its smoothness and continuous curvature which is better for a robot to follow. Also, it is very advantageous to use a higher order polynomial as the higher-order derivatives are needed to compute the minimum horizontal distance required for parking. In the algorithm, an adaptive method is used for the motion-planning, where at each point a new parking curve is computed, and a new motion command is sent to the robot controller. Moreover, the proposed work has investigated the minimum horizontal distance problem, and the result regarding the initial parallel distance condition has been shown. We have also shown the integration of a global planner (using A*) to navigate the robot to the initial point for the parking maneuver, from where the local planner (Quintic polynomial path planner) kicks in. The resulting planning algorithm was used in simulations using ROS and Gazebo, however, due to some errors, we observed an offset between the robot in the simulator and the visualizer package ‘rviz’ after launching the simulation. Other than the offset, the planning algorithm seems to work as expected, although further work needs to be done. The results mainly demonstrate a better and safer way to develop automatic parallel parking algorithms, especially when the minimum distance problem is a major concern. The local path planner is found to be highly robust to sensor noise, as the initial conditions are being computed every iteration. Furthermore, due to recomputation of the initial conditions every iteration, errors do not accumulate. However, this is predicated on the condition that the robot can be observable at every instance of time, to find the real-time location and state. The algorithm was also found to be very cheap computationally, with the time taken to generate a path in a 2-D cartesian plane with a resolution of 0.05 taking around 250-300ms.

VIII. FUTURE SCOPE & CONCLUSION

Automakers are starting to market self-parking cars because they sense a consumer demand. Self-parking cars can help to solve many pre-existing problems in dense-urban areas. There is a large demand for good parking solutions which can help mitigate this problem. This project attempts to regenerate and extend the results of a paper attempting to solve this problem. [12]. We have integrated two planners in order to achieve an end-to-end parking solution. However, further work can be done in the aspect of using different kinematic constraints for different robots, trying to find parking solutions for perpendicular, angular parking as well.

ACKNOWLEDGEMENT

We would like to express our gratitude to our Principal Advisor, Dr. Reza Monfaredi, Teaching Assistant, Mr. Rishabh Agarwal for their instruction and guidance. We are also thankful to the University of Maryland for the access to the research and educational resources from their online databases. Our thanks to the EECS Department in University of Berkeley for their concise notes on differential drive kinematics which we have found very useful. Special thanks to Mr. Zhang Shuwen, Mr. Mehrdad Simkani of University of Michigan and Mr. Mehrdad Hosseni Zadeh of Kettering University upon whose contributions we have used as our primary source of information for the modeling and simulation aspects of this project.

REFERENCES

[1] Paromtchik, Igor (June 2003). "Planning Control Commands to Assist in Car Maneuvers" (PDF). Proceedings of the IEEE International Conference on Advanced Robotics. Coimbra, Portugal. pp. 1308–1313.

[2] Banzhaf.H et al “The Future of Parking: A survey on automated valet Parking with an outlook on high-density parking”,2017 IEE Intelligent Vehicles Syposium (IV) , June, 11-14, 2017, Redondo Beach, CA.

[3] J. Vandorpe, Navigation techniques for the mobile robot LiAS. PhD thesis, Katholieke Universiteit Leuven-Faculteit Teogepaste Wetenschappen, 199

[4] S. Fleury et al., Primitives for smoothing mobile robot trajectories, IEEE transactions on robotics and automation, 11, No. 3 (June 1995).

[5] C. Zhu, R. Rajamani, Global positioning system-based vehicle control for automated parking, IMechE, 220 (2006) Part D: J. Automobile Engineering.

[6] A. Scheuer, T.H Fraichard, Planning continuous-curvature paths for carlike robots, IEEE Transactions on Intelligent Robots and Systems 3 (1996) 1304-1311. Asg

[7] A. Scheuer, T.H Fraichard, From reeds and shepp’s to continuouscurvarure paths, IEEE Transactions on Robotics and Automation 20 (2004) 1025-1035.

[8] E. Szádeczky-Kardoss, B. Kiss, Path planning and tracking control for an automatic parking assist system, Springer Tracts in Advanced Robotics 44 (2008) 175-184.

[9] T.-H.S. Li, S.-J. Chang, Y-X. Chen, Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robots, IEEE Transactions on Industrial Electronics 50 (2003) 867-880.

[10] Image for illustration : -http://www.permastripe.com/parking-lot-design-orlando-florida/

[11] Images: F. Lamiraux and J.P. Laumond “Smooth Motion Planning for car-like vehicles, IEEE Transactions on Robotics and Automation,Vol 17, No.4, August 2001

[12] Zhang.S et al, “Automatic Vehicle Parallel Parking Design using FIfith Degree Polynomial Path Planning” Vehicular technology Conference, IEEE 2011, San Francisco, CA, USA.

[13] https://chess.eecs.berkeley.edu/eecs149/documentation/differentialDrive.pdf

[14] https://up.codes/s/accessory-off-street-parking-and-loading-regulations-off-street-parking-regulati

[15] https://github.com/husky