Optimal Trajectory Planning For Parking Maneuver of Autonomous Ground Vehicles Using Multi-Stage Optimization

Sundara Subramanian Nagappan
Mechanical and Aerospace Engineering
NC State University
Raleigh, USA
snagapp2@ncsu.edu

Kritika Nijhawan
Electrical and Computer Engineering
NC State University
Raleigh, USA
knijhaw@ncsu.edu

***Abstract*—The main goal of this paper is to generate an optimal trajectory for parking maneuver of autonomous ground vehicles in the shortest time possible. The design composes of two optimization stages. The multi-stage optimization framework is adopted to promote robustness and improved convergence ability. The first stage generates a near-optimal parking trajectory using Particle Swarm Optimization (PSO) technique. This trajectory then serves as the input to the second stage which further uses a gradient-based optimization technique to obtain the final optimal parking maneuver trajectory. We test the proposed framework and generate simulation results for different parking scenarios. We further perform a comparative study to exhibit the strength of our framework.**

***Index Terms*—Optimal Parking Trajectory, Particle Swarm Optimization (PSO), Gradient-based Optimization, Autonomous Ground Vehicles.**

I. INTRODUCTION

Optimal trajectory planning can be considered as a non-linear, constrained optimization problem. Trajectory optimization focuses on generating a trajectory that minimizes or maximizes the performance attributes while satisfying a set of constraints. It finds a wide variety of applications in aerospace, robotics and autonomous industries. It is primarily used for path planning of unmanned autonomous vehicles [5]. It is also used in robotics for performing manipulation task like pick and place.

We propose a two-stage optimization framework that generates an optimal trajectory for performing safe parking maneuver in the shortest time possible. The motivation for designing two-stage optimization framework lies in the exploitation of two powerful yet contrasting optimization techniques (a) Sequential Quadratic Programming (SQP) – a traditional gradient-based method; (b) Particle Swarm Optimization (PSO) – a stochastic, population-based method. Gradient-based techniques exhibit powerful convergence properties but are very sensitive to initial guess and also increase the computational time. PSO exhibits exploration and exploitation capabilities and thus, is suitable for finding the global optimum [3]. It is not sensitive to initial guess and significantly reduces the computational time. Both the techniques complement each other and thus, influence our design.

In the first optimization stage, we use PSO to generate a near-optimal trajectory for parking of unmanned autonomous vehicle. This serves as the control input for the second optimization stage. In the second stage, we use gradient-based method (SQP) for generating the final optimal trajectory. This framework is then compared with single-stage optimization framework to prove its efficiency and robustness.

The main contributions made in this paper are:

- Set up a well-posed optimization problem subject to constraints.
- State the mathematical modelling of a front-steering vehicle.
- List all the constraints taken into account for optimal trajectory planning.
- Transcribe the optimization problem from continuous time to discrete time
- Design the two-stage optimization framework.
- Perform a simulation study to test the designed framework for parking maneuver of an autonomous ground vehicle.
- Compare the two-stage optimization framework with the single-stage optimization framework to showcase its convergence abilities and reduced computational cost.

II. FORMULATION OF TRAJECTORY OPTIMIZATION PROBLEM

In this section, we will formulate the parking trajectory optimization problem subject to constraints. For formulating the overall optimization problem, we first need to understand the vehicle kinematics. We also need to specify the constraints that the framework must take into consideration when generating the optimal parking trajectory. The mathematical modelling of the vehicle and the constraints acting on the vehicle are referred from [1]. Fig. 1 shows the parking layout.

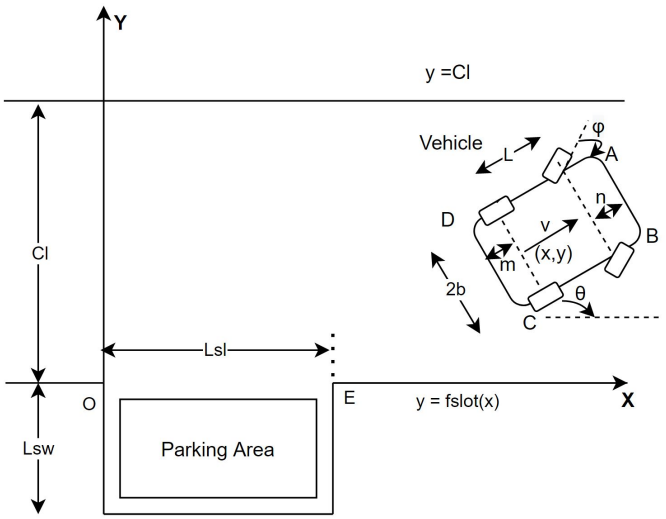

Fig. 1. Parking Layout

A. Mathematical Modelling of the Autonomous Ground Vehicle

We have made some assumptions about the front-steering vehicle:

- Ground vehicle is a rigid body.
- We have considered the kinematic bicycle model for deriving the equations of motion of the vehicle.

The state space equations of the vehicle model are defined as:

$$\begin{aligned} \dot{x}(t) &= v(t)\cos(\theta(t)) \\ \dot{y}(t) &= v(t)\sin(\theta(t)) \\ \dot{\theta}(t) &= \frac{v(t)\tan(\phi(t))}{l} \\ \dot{v}(t) &= a(t) \\ \dot{\phi}(t) &= w(t) \end{aligned} \tag{1}$$

where (x, y) is the center of the rear axle; v and a are the velocity and acceleration of the vehicle respectively; θ is the angle of orientation; ϕ is the steering angle with respect to the steering wheel; l is the length between the front and rear wheel; $t \in [0, t_f]$ denotes the time.

The state variables are defined as $X = [x, y, v, \theta, \phi]^T$ and the control inputs are defined as $U = [a, w]^T$. In Fig. 1, n is the front overhang length, m is the rear overhang length, $2b$ is the width of the vehicle, Cl is the width of the road and L_{sl} and L_{sw} are the length and width of the parking area.

B. Constraints acting on the vehicle

1) State constraints: The state constraints are defined as:

$$\begin{aligned} x(t) &\in [x_{min}, x_{max}] \\ y(t) &\in [y_{min}, y_{max}] \\ v(t) &\in [v_{min}, v_{max}] \\ \phi(t) &\in [\phi_{min}, \phi_{max}] \end{aligned} \tag{2}$$

These constraints are evident in the real-world scenarios and thus, have been taken into account. Constraints on velocity are applied to ensure that the vehicle is parked safely without colliding with the obstacles and without causing discomfort to the passengers. We have also applied constraints on the steering angle to restrict the vehicle's degree of turn.

2) Control constraints: Control constraints are defined as:

$$\begin{aligned} |a(t)| &\le a_{max} \\ |\dot{k}(t)| &\le \dot{k}_{max} \end{aligned} \tag{3}$$

In Eq.(3), $\dot{k} = \frac{w(t)}{l\cos^2\phi}$ is the derivative of the instantaneous curvature.

Constraints are imposed on $\dot{k}$ to ensure that the passengers experience a comfortable ride. Thus, a bumpy ride can be avoided by imposing these constraints.

3) Path constraints: To impose parking constraints, we first define the corner points of the vehicle. The corner points of the vehicle are given below:

$$\begin{aligned} (A_x, A_y) &= (x + \cos(\theta)(l+n) - b\sin(\theta), \\ &\qquad y + \sin(\theta)(l+n) - b\cos(\theta)) \\ (B_x, B_y) &= (x + \cos(\theta)(l+n) + b\sin(\theta), \\ &\qquad y + \sin(\theta)(l+n) - b\cos(\theta)) \\ (C_x, C_y) &= (x - m\cos(\theta) + b\sin(\theta), \\ &\qquad y - m\sin(\theta) - b\cos(\theta)) \\ (D_x, D_y) &= (x - m\cos(\theta) - b\sin(\theta), \\ &\qquad y - m\sin(\theta) + b\cos(\theta)) \end{aligned} \tag{4}$$

Once we have defined the corner points, we need to specify some inequality constraints. These constraints must be satisfied by the vehicle in order to achieve the desired parking maneuver.

The inequality constraints are defined as:

$$\begin{aligned} f_{slot}(A_x) \le A_y \le Cl \\ f_{slot}(B_x) \le B_y \le Cl \\ f_{slot}(C_x) \le C_y \le Cl \\ f_{slot}(D_x) \le D_y \le Cl \end{aligned} \tag{5}$$

$$f_{slot}(x) = -(H(x) + H(x - L_{sw}))L_{sl} \tag{6}$$

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \ge 0 \end{cases} \tag{7}$$

$f_{slot}(x)$ uses the Heavy side-step function $H(x)$. It makes sure that, when any of the vehicle's corner points has its x coordinate within the desired parking space, the boundary conditions are changed from $y = Cl$ and $y = 0$ to $y = Cl$ and $y = L_{sw}$ respectively.

4) Terminal conditions:

$$\begin{aligned} v(t_f) &= 0 \\ a(t_f) &= 0 \\ \theta(t_f) &= 0 \end{aligned} \tag{8}$$

The terminal conditions ensure that the velocity, acceleration and orientation angle of the vehicle go to zero and thus, the vehicle halts indicating successful parking of the autonomous ground vehicle.

C. Optimization Problem Setup

The main aim is to minimize the time taken by the vehicle to achieve parking maneuver. Thus, time is the objective function J which we want to minimize and it is subject to the set of constraints mentioned previously.

The overall optimization problem setup is given below:

$$\begin{aligned} \min J = & \int_{t_0}^{t_f} dt \\ \text{subject to } & \text{Eq.(1) - Dynamic constraints} \\ & \text{Eq.(2) - State constraints} \\ & \text{Eq.(3) - Control constraints} \\ & \text{Eq.(5) - Path constraints} \\ & \text{Eq.(8) - Terminal conditions} \end{aligned} \tag{9}$$

D. Discretized Mathematical Model

To solve a time optimal control problem, we have to make sure that we have finite number of points to be optimized. There are many approaches to transcribe a system from continuous time to discrete time. In this paper, we have adopted Forward Euler Approximation due to its simplicity. Thus, we discretize the continuous time mathematical model as follows:

$$\begin{aligned} x(k+1) &= x(k) + v(k)\cos(\theta(k))dt \\ y(k+1) &= y(k) + v(k)\sin(\theta(k))dt \\ \theta(k+1) &= \theta(k) + \frac{v(k)\tan(\phi(k))dt}{l} \\ v(k+1) &= v(k) + a(k)dt \\ \phi(k+1) &= \phi(k) + w(k)dt \end{aligned} \tag{10}$$

The state variables are defined as $X = [x(k), y(k), v(k), \theta(k), \phi(k)]^T$ and the control inputs are defined as $U = [a(k), w(k)]^T$.

E. Discretized Optimization Model Setup

The discretized optimization problem setup is given below:

$$\begin{aligned} \min J = & \sum_{i=0}^{N-1} \triangle T_i \\ \text{subject to } & \text{Eq.(10) - Discretized system dynamics} \\ & \text{and all other above-mentioned constraints} \\ & \text{in their discretized form} \end{aligned} \tag{11}$$

We say that an optimization problem is well-posed if there exists an optimal solution that satisfies all constraints and that solution is logical in the real world application [2]. Our current optimization problem is poorly-posed because with just the constraints imposed so far, we will end up getting an optimal solution that makes the vehicle move nowhere so as to reduce the total time of travel. So, in order to ensure a well-posed optimization problem, we include a terminal constraint on the time and the state of the vehicle. This ensures that the vehicle reaches the desired parking space.

III. Two-stage optimization framework

The objective of our paper is to utilize a multi-stage optimization framework to improve the convergence speed and robustness. We have used Particle Swarm Optimization (PSO) algorithm as the first-stage optimization technique. The solution from the first-stage has been used as the initial guess for the Sequential Quadratic Programming (SQP), second-stage optimization technique.

A. First stage Trajectory Planner

After setting up the optimization problem, it is desired to choose an algorithm for finding the optimal solution. Traditional gradient-based optimization methods are very sensitive to initial guesses and are likely to get stuck at an infeasible point or local optimizer. To deal with this issue, we have used the first-stage to improve the initial guess for the second-stage which is gradient-based. This makes sure that our search for the optimal point in the second-stage starts very close to the actual optimal point. Newton's method performs extremely well when the search starts very close to the optimizer, thus drastically improving the convergence. A brief explanation of the PSO algorithm has been given below.

Particle Swarm Optimization (PSO) is a Derivative-free Optimization technique to find the global optimizer [4]. It is a population-based algorithm that was inspired by social interaction principles. To make sure we don't get stuck in a local optima, we start with many candidate solution points called particles. These set of particles form the population. The population is sometimes called a swarm.

The general idea of the PSO algorithm is to start with an initial randomly generated population where each point is viewed as a particle moving with an associated velocity. The cost function/objective function is evaluated at each point of the swarm and based on these values, we create a new swarm of particles with new set of associated velocities. Each particle keeps track of its best position so far. This is called the personal best. We also keep track of the population's best position so far. This is called the global best. The particles interact with each other by updating their velocities taking into account their individual personal best and the global best. The following equations are used for updating the position and velocity in each iteration.

$$\begin{aligned} V_i^{k+1} = & wV_i^k + c_1 {r_1}_i^k.(P_{best}{}_i^k - X_i^k) \\ & + c_2 {r_2}_i^k.(g_{best}^k - X_i^k) \end{aligned} \tag{12}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{13}$$

'.' denotes the Schur product (Element wise multiplication) where w is the inertia term; c_1 is the cognitive coefficient; c_2 is the social coefficient.

The quality of the particle is evaluated using the fitness value calculated from the objective function. The objective of our PSO algorithm is to find a near-optimal solution that can serve as an initial guess for the second stage. An objective function with a heuristic approach is used for tackling the optimal time problem. Using the heuristic approach, we have indirectly minimized the time. We minimized the difference between shortest distance and the current distance between the initial position and the final position. In addition to this, we have penalized the control inputs and states at the final time step to impose the terminal constraints in the optimization problem.

The fitness value is computed via,

$$\begin{aligned} J =& k_1(d_{ref} - d)^2 + k_2(x - x_f)^2 + k_3(y - y_f)^2 + \\ & k_4(w_f)^2 + k_5(a_f)^2 \end{aligned} \tag{14}$$

where $k1, k2, k3, k4, k5$ are the weights; d_{ref} is the euclidean distance from the starting position to the desired final position; d is the euclidean distance from the current position to the desired final position; x_f is the final x position; y_f is the final y position; w_f and a_f are the final steering rate and the final acceleration of the vehicle respectively.

The algorithm keeps iterating for predetermined number of times and we terminate it when i equals $MaxIter$. The current overall best particle, G_{best} is the near-optimal solution that will serve as the initial guess for the second-stage optimization. Fig. 2 shows the flowchart of the PSO algorithm.

B. Second-Stage Optimization

The optimization technique adopted in the second-stage is the traditional gradient-based method. In this paper, we have used Sequential Quadratic Programming (SQP). Since the SQP uses Newton's method, starting at a near-optimal solution helps in quick convergence with less number of iterations than any other single-stage optimization technique.

In this paper, FMINCON from MATLAB was used to implement the Sequential Quadratic Programming. FMINCON requires two functions, one is the objective function and the other is the constraint set. In this stage, unlike PSO algorithm we directly use time in the objective function. Since the total time taken for the parking maneuver in discrete time depends on the length of each time step, we include each time step as a decision variable to be minimized.

Unlike the PSO algorithm, where just terminal constraints were considered, in SQP, in addition to those constraints, we also include state constraints, control constraints and path constraints at every time step. The constraints are imposed at every time step to make sure that the vehicle not only reaches the desired final position but also does it without violating

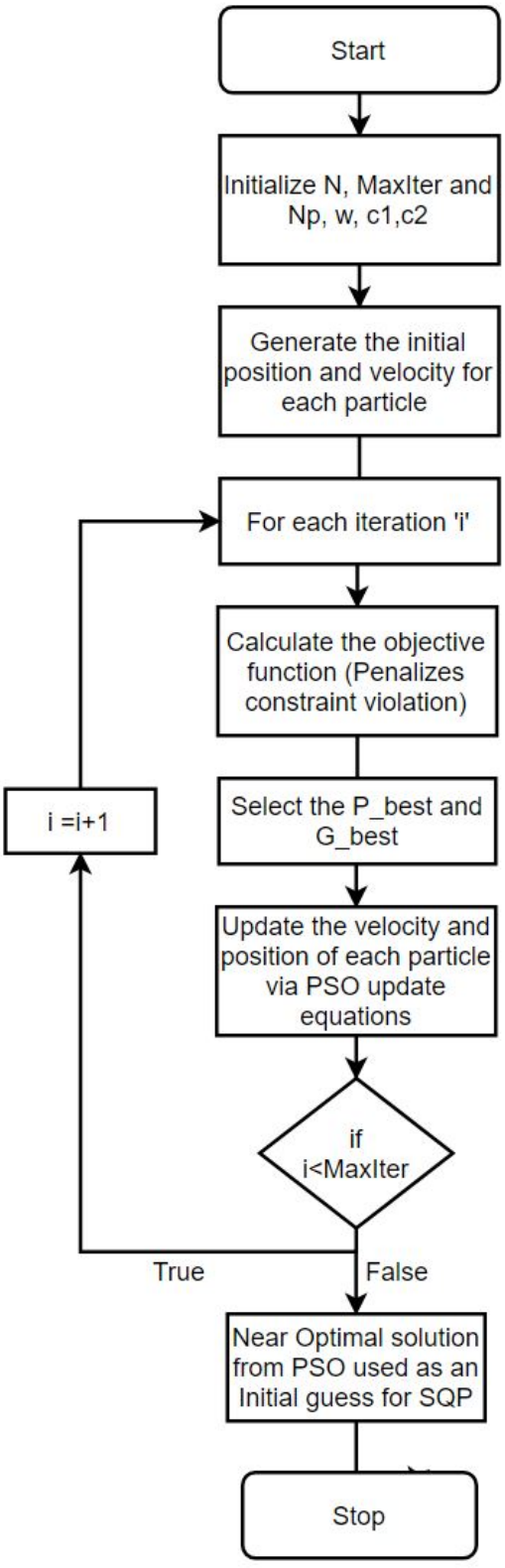

Fig. 2. Flowchart for PSO Algorithm

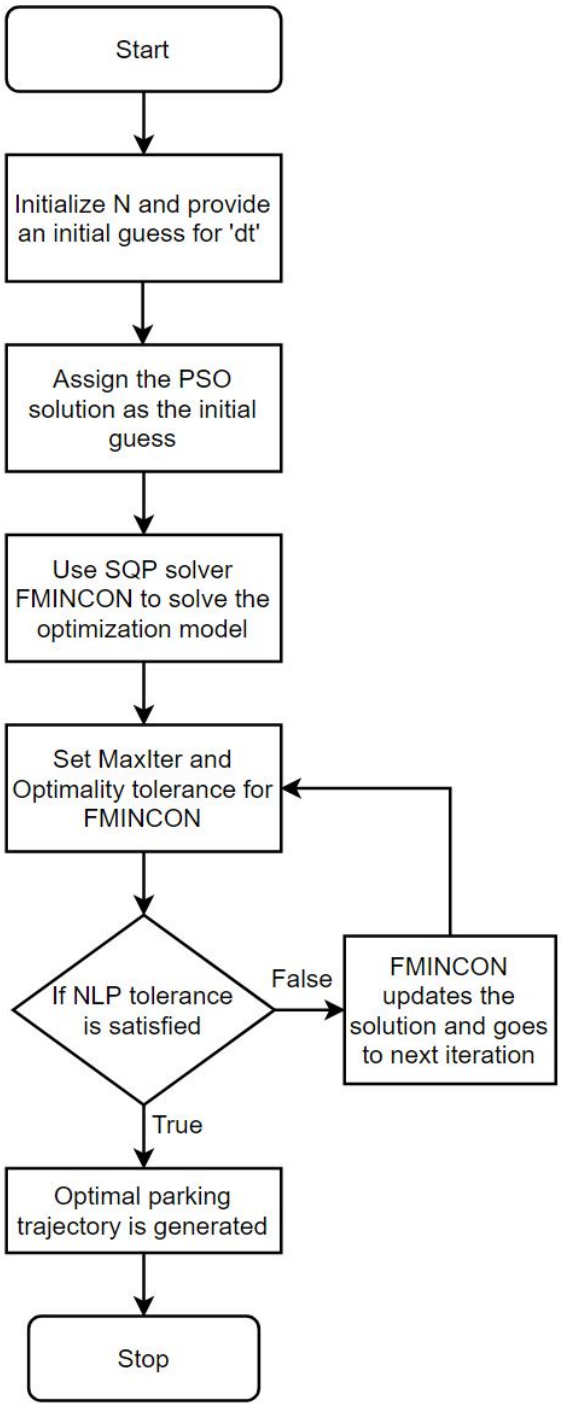

Fig. 3. Flowchart for SQP Algorithm

the upper and lower bounds on states, control inputs and path boundaries. Fig. 3 shows the flowchart of the SQP algorithm.

IV. Simulation Results

In this section, results were obtained by applying the two-stage optimization framework to the trajectory planning problem. The various parameters used in the optimization problem are shown in Table I and Table II. The mission cases have also been specified in Table III.

TABLE I
VEHICLE PARAMETERS AND BOUNDS ON STATES

Parameters	Values	Variables	Range
L_{sl}, m	7.3	x, m	[-2,15]
L_{sw}, m	2	y, m	[-2,3.5]
Cl, m	3.5	v, m/s	[-2,2]
n, m	0.8	a, m/s^2	[-0.75,0.25]
m, m	0.7	ϕ, deg	[-30°, 30°]
l, m	2.5		
$2b$, m	1.5		

TABLE II
ALGORITHM PARAMETERS

Parameters	Values/ranges	Parameters	Values/Range
w	0.9	Np	50
c_1	2	$MaxIter$	1000
c_2	2	N	50
r_1	[0,1]	ϵ	10^{-6}
r_2	[0,1]		

TABLE III
MISSION CASES

Case No.	Initial conditions
1	$x(0) = 10.70$ $y(0) = 1.95$ $\theta(0) = 0$ $x(t_f) = 1$ $x(t_f) = -1$
2	$x(0) = 10.70$ $y(0) = 1.95$ $\theta(0) = \pi/15$ $x(t_f) = 1$ $x(t_f) = -1$
3	$x(0) = 10.70$ $y(0) = 1.95$ $\theta(0) = 0$ $x(t_f) = 1$ $x(t_f) = 1$
4	$x(0) = 10.70$ $y(0) = 1.95$ $\theta(0) = -\pi/9$ $x(t_f) = 1$ $x(t_f) = 1$

A. Optimal trajectories for various Parking scenarios

In order to validate the two-stage optimization framework, many mission cases were tested. Testing for mission cases is very important because the parking trajectory optimization algorithm should be able to produce an optimal path that allows the vehicle to reach the destination in the shortest time.

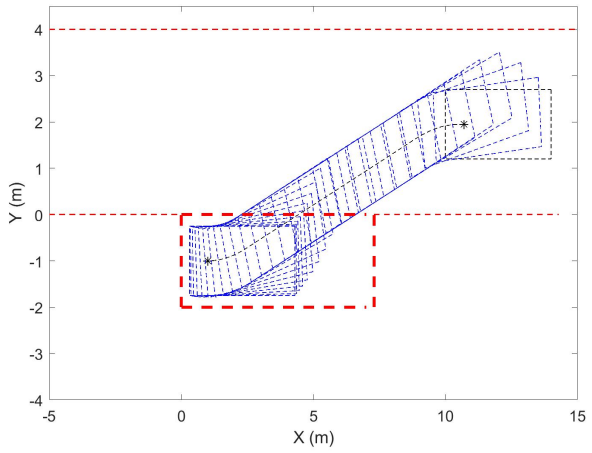

Fig. 4. Parking Scenario 1: Optimal Parking Trajectory

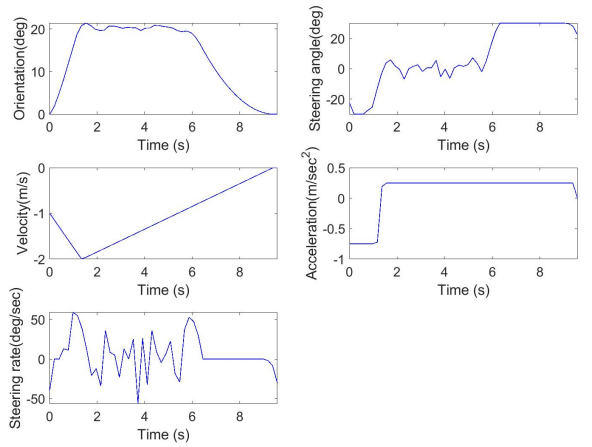

Fig. 5. Parking Scenario 1: Plots for optimal states and optimal control inputs

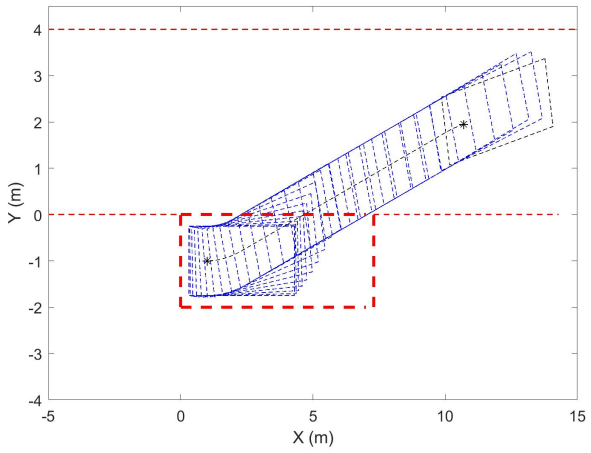

Fig. 6. Parking Scenario 2: Optimal Parking Trajectory

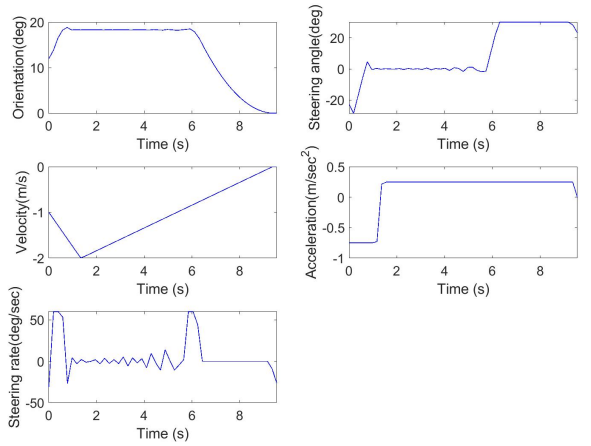

Fig. 7. Parking Scenario 2: Plots for optimal states and optimal control inputs

1) Parking Scenario 1: In the first mission case, we perform parallel parking. Autonomous industries believe that it is one of the most challenging parking scenarios. This scenario can be dealt with the proposed two-stage optimization framework. From Fig. 4 and Fig. 5, it is clear that the optimal control input steers the vehicle to a particular angle and then maintains that angle for a certain period of time. Towards the end, the vehicle turns in a certain direction to achieve the desired final orientation of zero degrees with respect to the global reference frame's x axis. Since we have also imposed a constraint on the rate of curvature, the optimal control input makes sure that the trajectory profile generated is smooth. In some cases, we might not be able to negotiate a curve if it's too sharp due to the constraints imposed by the vehicle's dynamics. Thus, we have incorporated such constraints in our algorithm to mimic real-world scenarios.

2) Parking Scenario 2: The second mission case is very similar to the first mission case. The only difference is the initial orientation of the vehicle. Since the vehicle's orientation is already in the direction of the parking lot, it doesn't steer initially but as it gets close to the desired final position, it turns appropriately and the final orientation goes to zero degrees. Thus, a perfect parking is achieved. This is evident from Fig. 6 and Fig. 7.

3) Parking Scenario 3: From Fig. 8 and Fig. 9, it is evident that the third mission case is the easiest of all the cases since the vehicle has to travel straight for most part of its trajectory and make a slight turn towards the end to align itself parallel to the road boundary ensuring a final orientation angle of zero degrees. Thus, a perfect parking is achieved.

4) Parking Scenario 4: In the fourth mission case, we see that the initial orientation is $-\pi/9$. So, the control input for steering rate is high initially. Once it aligns itself to a positive orientation, the vehicle travels straight for certain period of time. Then, it steers again to get the orientation to zero degrees.

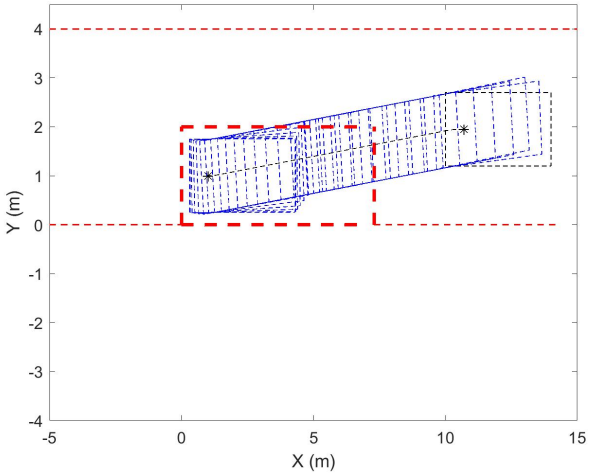

Fig. 8. Parking Scenario 3: Optimal Parking Trajectory

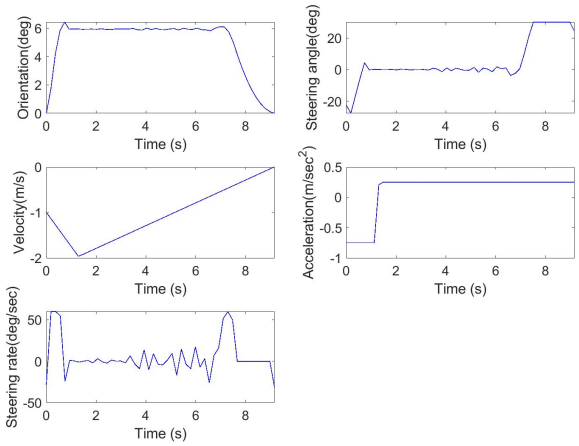

Fig. 9. Parking Scenario 3: Plots for optimal states and optimal control inputs

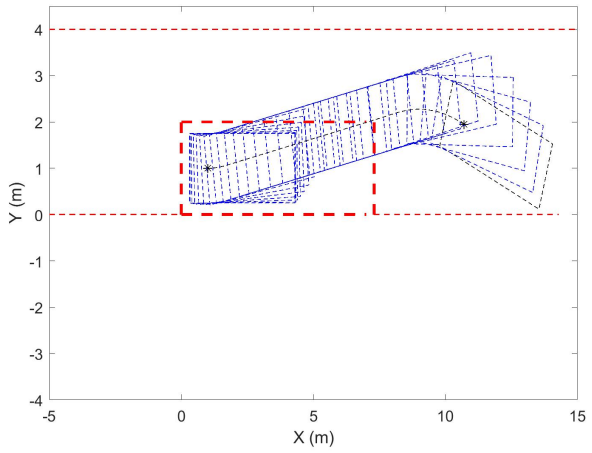

Fig. 10. Parking Scenario 4: Optimal Parking Trajectory

This is evident from the Fig. 10 and Fig. 11.

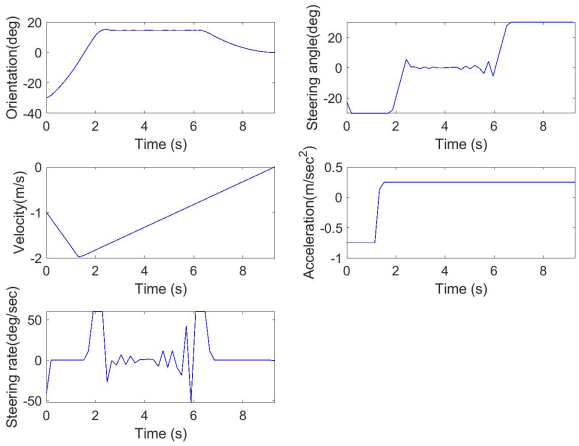

Fig. 11. Parking Scenario 4: Plots for optimal states and optimal control inputs

The acceleration profile and velocity profile look almost the same for all the mission cases. This is because, the initial state of the vehicle in all the mission cases is such that the vehicle is facing in the direction opposite to where the desired parking space is. To reach the desired final point in the shortest time possible, the vehicle initially accelerates backwards. Since we have terminal constraints on velocity and acceleration (to bring the vehicle to stop), the acceleration further increases gradually till the vehicle halts. From the acceleration vs time plots in Fig. 5, 7, 9 and 11, it is clear that the acceleration is initially negative maximum to speed up the parking maneuver and then at some point along the path the acceleration becomes positive maximum to bring the vehicle to complete stop.

B. Comparison of performance between two-stage and single-stage Optimization

Iter	Func-count	Fval	Feasibility	Step Length	Norm of step	First-order optimality
270	27946	9.574320e+00	2.526e-09	1.000e+00	1.614e-02	1.156e-05
271	28049	9.574320e+00	3.603e-08	1.000e+00	5.851e-02	3.416e-05
272	28152	9.574320e+00	4.817e-08	1.000e+00	3.464e-02	4.671e-06

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

Fig. 12. Computational Cost in Single-Stage Framework

83	8700	9.575326e+00	7.474e-07	1.000e+00	1.169e-02	1.771e-02
84	8804	9.575305e+00	1.089e-07	1.000e+00	5.372e-02	2.608e-02
85	8908	9.575278e+00	7.327e-08	1.000e+00	6.594e-02	2.968e-02
86	9012	9.575251e+00	2.921e-08	1.000e+00	6.482e-02	3.045e-02
87	9121	9.575250e+00	2.825e-08	1.681e-01	2.969e-03	3.014e-02
88	9152	9.575250e+00	2.825e-08	2.254e-05	1.464e-06	3.014e-02

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

Fig. 13. Computational Cost in Two-Stage Framework

As mentioned previously, the single-stage optimization has more computational cost compared to the multi-stage approach. Multi-stage optimization has clearly improved the convergence ability and has proved to be robust by performing well for many parking scenarios. From the Fig. 12 and Fig. 13 it is evident that it takes less number of iterations for the solution to converge in the Multi-stage approach compared to the single-stage approach.

V. Future Scope

We have considered the kinematic bicycle model for deriving the equations of motion of the autonomous ground vehicle and thus, have ignored the lateral forces acting on the vehicle. For future work, we can consider a dynamic bicycle model for deriving the equations of motion of the vehicle. This will take the side-slip problem into consideration and thus, will be more plausible with the real-world scenarios.

In our optimization problem formulation, we have not considered the obstacle avoidance constraints. To further optimize the parking trajectory, we can take these constraints into account. In reality, many environmental constraints will directly or indirectly affect the vehicle's performance. Thus, to ensure that the passengers have a safe and comfortable parking maneuver experience, we can add some environmental constraints as well.

In the first-stage, we generate a near-optimal parking trajectory using the PSO algorithm. While designing the first stage optimization framework, we only applied the terminal constraints. For future work, we can penalise the objective function based on all the constraints mentioned in the paper. This will drastically improve the performance of the PSO algorithm and we will get a better initial guess for the second-stage of the optimization framework.

[1] mentions an Adaptive Gradient Particle Swarm Optimization (AGPSO) technique to improve the initial guess and reduce the computational cost. We can derive a similar objective function and take degree of violation into account so as to improve our first-stage of the framework.

In the real-world scenario, the autonomous vehicles receive data from the perception node and further process it to produce a trajectory. For future work, we can first tune the data received from the perception node and accordingly set up a well-posed optimization problem. If a localized map is available, we can take into account the different parking scenarios and then test the performance of our two-stage optimization framework.

VI. Lessons Learned

Optimization problems can be classified as constrained optimization problems and unconstrained optimization problems. In reality, most of the problems are subject to tons of constraints. It is really important to consider these constraints when setting up a well-posed optimization problem. Efficiency and robustness of the optimization technique depends on how well it is able to tackle all the imposed constraints. We faced challenges while setting up the optimization problem. In our case, the objective function was time which we wanted to minimize. In the real-world scenario, many factors and constraints influence time. Thus, in the first-stage of the optimization framework, we used a heuristic approach for minimizing time. Distance traversed is directly proportional to the time taken to reach the destination. So, we minimized the difference between shortest distance and the current distance between the starting point and the final point. Such an approach indirectly minimized the original objective function, time. In the second-stage of the optimization framework, we used time step as a control input. Along with optimizing the constraints, we also optimized the time. Thus, we learned how to tackle constrained optimization problems.

VII. Conclusion

In this paper, we have studied the effect of the multi-stage approach on the convergence ability and robustness. The results from multi-stage optimization approach were compared with that of the single-stage optimization technique. It was observed that the former converged to the optimal solution quicker.

Considering the sensitivity of the Newton's method to the initial guess and its convergence ability when it starts the iterations from a near-optimal point, we have adopted PSO which is a non gradient-based algorithm to generate the initial trajectory. The solution from PSO served as the initial guess for the SQP. Since, we used a near-optimal point as the initial guess, our multi-stage approach had less computation time and exhibited robustness and better convergence ability.

The above conclusion was verified using the simulation results corresponding to different mission cases.

References

[1] G. R. Chai, A. Tsourdos, A. Savvaris, S. Chai and Y. Xia, "Two-Stage Trajectory Optimization for Autonomous Ground Vehicles Parking Maneuver," in IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 3899-3909, July 2019, doi: 10.1109/TII.2018.2883545.

[2] X. Yin and Q. Chen, "Trajectory Generation With Spatio-Temporal Templates Learned From Demonstrations," in IEEE Transactions on Industrial Electronics, vol. 64, no. 4, pp. 3442-3451, April 2017, doi: 10.1109/TIE.2016.2613843.

[3] Q. Zhao and C. Li, "Two-Stage Multi-Swarm Particle Swarm Optimizer for Unconstrained and Constrained Global Optimization," in IEEE Access, vol. 8, pp. 124905-124927, 2020, doi: 10.1109/ACCESS.2020.3007743.

[4] M. Bevilacqua, A. Tsourdos and A. Starr, "Particle swarm for path planning in a racing circuit simulation," 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin, 2017, pp. 1-6, doi: 10.1109/I2MTC.2017.7969735.

[5] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia and C. L. P. Chen, "Multiobjective Optimal Parking Maneuver Planning of Autonomous Wheeled Vehicles," in IEEE Transactions on Industrial Electronics, vol. 67, no. 12, pp. 10809-10821, Dec. 2020, doi: 10.1109/TIE.2019.2962482.

Appendix

We proposed a two-stage optimization framework for parking trajectory optimization of autonomous ground vehicles. The first-stage used a Particle Swarm Optimization (PSO) technique for generation of a near-optimal trajectory for parking maneuver. Kritika Nijhawan formulated the objective

function for PSO algorithm. A heuristic approach was adopted for designing the objective function. She implemented the PSO algorithm in MATLAB.

The second-stage used the Sequential Quadratic Programming (SQP) technique for generating the final optimal trajectory for parking maneuver. Sundara Subramanian Nagappan formulated the objective function for SQP algorithm. He took all the constraints into account when designing the problem setup. He coded the SQP algorithm in MATLAB.

A decent time was spent in integrating the two stages together and producing desired results. The two-stage framework was implemented for different parking scenarios. We constructed our own flowcharts for both the algorithms. This was primarily done to elaborate our approach in the simplest way possible.

A. Timeline

Date	Tasks	Responsibilities
1st Oct	Problem Setup	We researched different mathematical models to obtain the right objective function and the constraint set.
7th Oct	Literature review on Optimization techniques	We explored different optimization algorithms and picked the best ones for the multi-stage framework.
21th Oct	First Optimization Stage	Kritika implemented the PSO algorithm.
5th Nov	Second Optimization stage	Sundar implemented the SQP algorithm.
11th Nov	Simulation of use-cases	We simulated different use-cases for the parking maneuver.
18th Nov	Perform a comparative study	We compared our results with single-stage optimization framework.
7th Oct - 18th Nov	Final Report	We worked on it together and completed each section.