Multi-Goal Trajectory Planning for a Car in an Obstacle Course

Emanuel Iwanow, Sébastien Bohn & Sophia Moyen
Supervisors: Andreas Höhl & Lukas Theiner

***Abstract*— This paper presents an approach for time-optimal trajectory planning of a car in an environment with multiple goals and obstacles. We formulate the problem as an Optimal Control (OC) framework, incorporating both trajectory optimization and goal ordering within a unified structure. To achieve this, we leverage the Traveling Salesman Problem (TSP) solution as an initial guess for the global solution within the OC framework. This approach enables joint optimization of individual trajectories and the goal visiting sequence. Our primary goal is to develop a method for the car to efficiently navigate through designated goals while respecting a set of equality and inequality constraints arising from the car's dynamic model, the environment, and the task itself. The evaluation is performed in simulation using the CasADI library [1].**

I. Introduction

Motion planning, particularly for autonomous vehicles navigating complex environments, has been an active area of research with significant practical applications. One crucial challenge lies in efficiently planning trajectories for a vehicle to visit multiple designated goals while avoiding obstacles and adhering to specific operational constraints. This paper addresses the problem of Multi-Target Trajectory Planning (MGTP) for a car in an obstacle course. Existing research on MGTP, such as Janos *et al.* [2] and Ishida *et al.* [3], often separates the problem into two distinct parts:

- **Time-optimal trajectory generation**: Optimal trajectory generation: This focuses on finding collision-free paths between goals, while minimizing the cost to do so. Often utilizing sampling based planners, like Rapidly-exploring Random Trees (RRTs) (LaValle *et al.* [4]), or other motion planning algorithms. These algorithms excel at navigating complex environments with obstacles, but require separate approaches to determine the order of visiting goals.
- **Target ordering**: It is a combinatorial problem that involves determining the most efficient sequence for visiting the designated goals using the already found collision-free paths between each goal, often framed as as the Traveling Salesman Problem (Robinson, J. [5]).

In this paper, we propose a well integrated approach of these two parts, that combines the strengths of both Optimal Control (OC) and TSP for MGTP. We formulate the problem as an OC framework, encompassing both trajectory optimization and goal ordering within a unified framework. To achieve this, we use a simplified version of our actual OC Problem as input for the TSP. Then we leverage the TSP solution as an initial guess for the global solution within the OC framework. This approach allows for joint optimization of both the individual trajectories and the sequence in which the car visits the goals.

Our primary objective is to develop a method that enables the car to efficiently navigate through a defined set of goals, adhering to operational constraints, and ultimately returning to its initial position in the shortest time possible. This optimal trajectory search challenge is characterized by a two-dimensional problem, where the vehicle must not only respect the constraints of door positioning, but also take into account the complete physical dynamics of travel. These constraints can encompass various aspects, including the car's dynamic model, the limitations of its environment, and the specific requirements of the task, such as speed limits, acceleration profiles, or safety considerations.

The remainder of this paper will delve deeper into the details of our proposed methodology, including the problem formulation, solution approach, and performance evaluation. We will also discuss the limitations of our work and potential future directions.

II. Problem Formulation

We have two-level problem formulation, that includes modelling the environment in which the car must act and the car dynamics itself. A typical environment includes a start/end position, obstacles and goals, as seen in figure 1.

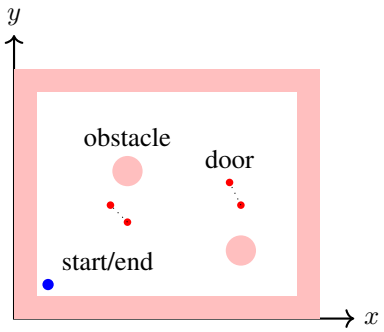

Fig. 1: Typical 2D scenario which the method aims to tackle. The car starts (doors) without collision with obstacles.

The task of multi goal trajectory planning can be modeled as an Optimal Control Problem (OCP). The cost function J in equation 1 summarizes the purpose of passing through all the goals while minimizing the overall time to do so. The car starts at an initial pose (x_0, y_0, θ_0), passes through all n goals and goes to its final pose (x_N, y_N, θ_N). There are, therefore, $n + 1 = N$ stagewise trajectories that must be optimized. The timestep at which the car passes through a specific goal is represented by t_i with $i = 1, ..., N$. The

problem is subject to equality and inequality constraints, that include: the **dynamic model** of the car (eq. 2); **boundary conditions** that constraints the initial and final position to be $(x_0, y_0) = (x_N, y_N) = (0, 0)$; **path constraints**, that are directly related to the intermediate timesteps t_i; and **stagewise path constraints**, that relate to the timesteps $t \in [t_{i-1}, t_i]$ of each of the N subtrajectories.

$$\min_{x(\cdot),u(\cdot),t_1,\ldots,t_N} J = \int_0^{t_N} dt \tag{1}$$

subject to

Dynamic Model: $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)),\ t \in [t_0, t_N]$
Boundary Conditions:
$x(0) = y(0) = x(t_N) = y(t_N) = 0$,
$\sin(\theta(0)) = \sin(\theta_0), \sin(\theta(N)) = \sin(\theta_N)$,
$\cos(\theta(0)) = \cos(\theta_0), \cos(\theta(N)) = \cos(\theta_N)$
Path Constraints: $h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0,\ t \in [t_0, t_N]$
Stagewise Path Constraints $h_i(\mathbf{x}(t)) \leq 0,\quad t \in [t_{i-1}, t_i],\ i = 1, ..., N$

A. Dynamic Model

The dynamic model chosen to represent the car was an altered Kinematic Bicycle Model, as described in Freire *et al.* [6]. The assumptions include front wheel steering, front and rear tires represented as one single tire in each axle and no slip condition. As input the model would have the acceleration (a) and the steering angle (δ), and as output it would have the vehilcle speed (v), the heading angle (θ) and the rear axle center position (x, y). The representation of the model can be seen on figure 2.

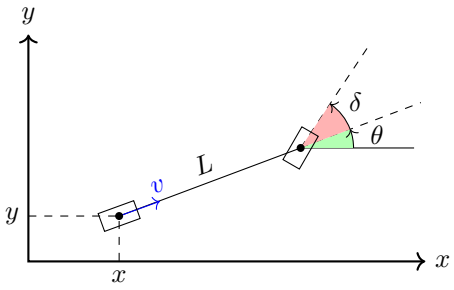

Fig. 2: Representation of the Kinematic Bicycle Model on the xy-plane.

The state equation of the dynamic model $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ is given by equation 2 with $\mathbf{x} = (x, y, \theta, v)^T$, $\mathbf{u} = (\delta, a)$ and L being the length between wheel axes.

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \end{pmatrix} = \begin{pmatrix} v(t) \cdot cos\theta(t) \\ v(t) \cdot sin\theta(t) \\ \frac{v(t)}{L} \cdot tan\delta(t) \\ a \end{pmatrix} \tag{2}$$

x, y: Rear axle center position
θ: Heading angle
v: Vehicle speed
δ: Steering angle
a: Acceleration
L: Length of the car

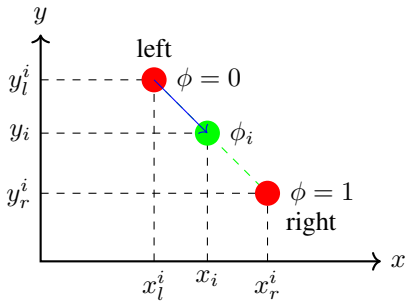

Fig. 3: Calculation of optimization variable ϕ_i that constraints the car to pass through each of the goals, situated between the left and the right extremity.

B. Path Constraints

The path constraint is related to guaranteeing the car passes through all the goals. To mathematically model that, we create another optimization variable $\phi_i \in [0, 1]$ for each of the n doors. The position of the car x_i and y_i in the moment it crosses the goals needs to be contained in the imaginary segment connecting the two extremities, as seen in figure 3.

The solution for $x_i = x(t_i)$ and $y_i = y(t_i)$ can be seen in equations 3 and 4, respectively.

$$x(t_i) = x_l^i + \phi_i \cdot (x_r^i - x_l^i) \tag{3}$$

$$y(t_i) = y_l^i + \phi_i \cdot (y_r^i - y_l^i) \tag{4}$$

The optimization variable ϕ_i represents how close x_i and y_i are from each of the two extremities of the door. If it is closer to the left extremity, then ϕ is closer to 0. If the car is closer to the right extremity, then ϕ is closer to 1.

C. Stagewise Path Constraints

These constraints are applied at each timestep t of the entire trajectory and include safety limitations and also constraints related to the dynamic model.

- Limitations on steering angle δ, acceleration a, velocity v and centripetal acceleration $a_{c_{max}}$:

$$\begin{cases} \delta_{min} \leq \delta < \delta_{max} \\ a_{min} \leq a < a_{max} \\ 0 \leq v \leq v_{max} \\ \frac{v^2 \cdot tan(\delta)}{L} \leq a_{c_{max}} \end{cases}$$

- To guarantee the car doesn't leave a delimited areas, we limit the allowed values for x and y:

$$\begin{cases} x_{min} < x < x_{max} \\ y_{min} < y < y_{max} \end{cases}$$

- For any obstacle, a safety circumference around it can be drawn, which the car is not allowed to enter. The obstacle's area is defined as being a circle on the xy-plane, as seen in figure 5.

$$(x - x_p)^2 + (y - y_p)^2 \geq R^2 \tag{5}$$

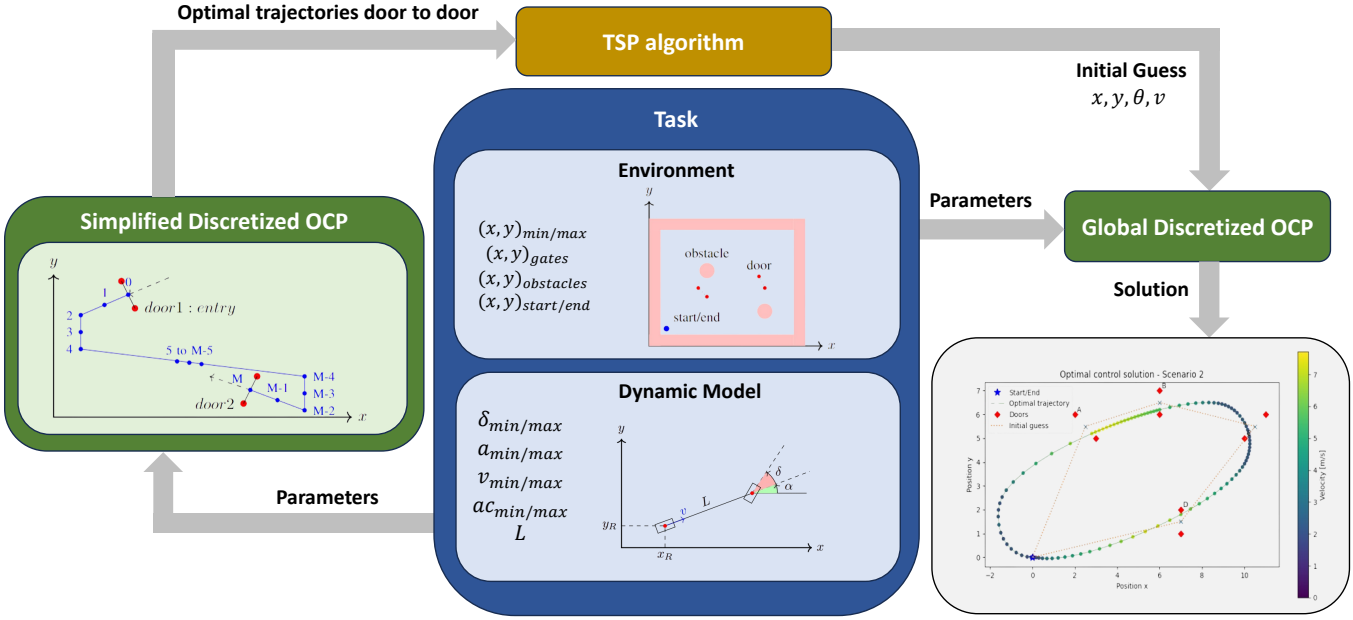

Fig. 4: Overview of pipeline used for solving the multi goals trajectory planning. The parameters of the task are used to formulate a simplified discretized Optimal Control Problem. Based on that formulation, the optimal trajectory between doors and initial/final position are calculated. The TSP algorithm the does a combinatorial analysis to provide an initial guess for the global discretized Optimal Control framework.

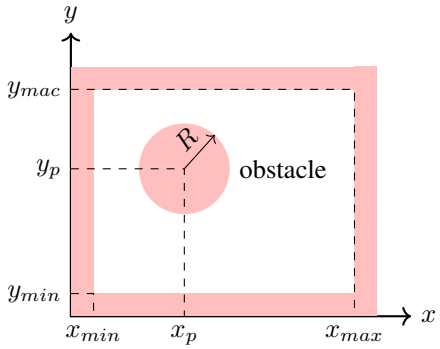

Fig. 5: The colored areas represent the position constraint for the car's trajectories. At any given point x and y must not exceed the maximum and minimum values. The circle representing the obstacles provide also forbidden values for x and y.

III. Method

Given the OCP presented, the first step of the method is to discretize the problem so that it can be numerically solved. The discretization is done via Direct Multiple Shooting with the help of Runge-Kutta-4 method. Using the discretized OCP with simpler problem formulations, we calculate the optimal trajectories from door to door. Then, with Traveling Salesman Problem, we define the order in which to visit the doors as well as an initial guess for the control variables x,y,θ and v. This initial guess is then finally fed to the global OCP problem for a better initialization and to avoid local minimas and infeasibilities. The overview of the entire pipeline can be seen in figure 4.

A. Discretized Optimal Control Problem

In order solve the optimal control problem numerically, it must be discretized and converted to a nonlinear programming problem (NLP). This is done using direct multiple shooting, as described in Bock *et al.* [7] and Erhard *et al.* [8]. With this method, the time horizon is divided in subintervals, whose ODE's can then solved by an ODE solver, such as CasADi. For our case, the integration steps are done using one iteration of the Runge Kutta 4 (RK4).

Considering the problem that the car starts at the origin, passes through n goals and returns to the origin, we will have n+1=N piecewise trajectories from door to door. The whole time to complete the trajectory is divided in N time intervals T_i, that are not necessarily of the same duration and must be individually optimized. The time intervals T_i are then divided in M control intervals. An ilustration of the discretization grid can be seen in figure 6.

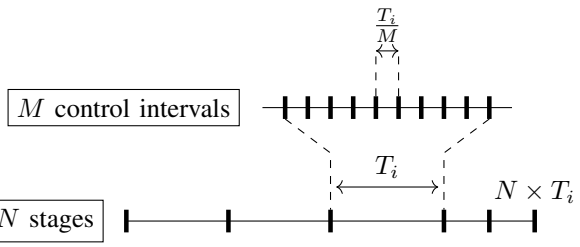

Fig. 6: Setup of the discretization grid. T_i represents the time to complete one stage, which means going from on goal to another. M is user input and determines the number of control intervals for the multiple shooting for each stage.

For each subinterval $T_{ij} = \frac{T_i}{M}$, we apply a single shooting method using the RK4, so that we have a multiple shooting method at the end for the global problem. For each T_{ij}, there is a control vector $\mathbf{u_{ij}} = [\delta_{ij}, a_{ij}]$ linked to it. The integration for each T_{ij} is performed from an initial state $\mathbf{s_{ij}}$ to obtain the final state $\mathbf{x_{ij}}$, using the controls $\mathbf{u_{ij}}$. An illustration of the integration steps can be seen in figure 7.

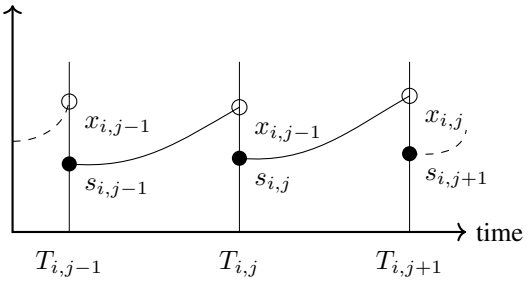

Fig. 7: Multiple shooting integration steps using RK4.

The global optimization vector $w = [w_1, .., w_N]$ contains the optimization subvectors w_i for each time interval T_i:

$$w_i = [s_{i,0}, u_{i,0}, ..., s_{i,M-1}, u_{i,M-1}, s_{i,M}, T_i] \quad (6)$$

The global discretized OCP is realized by minimizing the time intervals T_i needed to complete each stage. Through continuity constraints, we ensure that the result from the integration steps are connected, such as the state values at the end of one subinterval $x_{i,j-1}$ are seamlessly compatible with the state values $s_{i,j}$ at the beginning of the next:

$$\min_{w} \sum_{i=1}^{N} T_i \quad (7)$$

subject to

Interval Continuity:
$\mathbf{x}_{ij} = \mathbf{s}_{i,j-1},\ (i,j) = (1,1), ...(N, M-1)$
Stage Continuity:
$\mathbf{s}_{i,M} = \mathbf{s}_{i+1,0},\ i = 1, ...N-1$
Boundary Conditions: $s_{1,0} = s_{N,M} = 0$
Path Constraints:
$\tilde{h}(\mathbf{s}_{ij}, \mathbf{u}_{ij} \leq 0, (i,j) = (1,1), ...(N, M-1)$
Stagewise Path Constraints:
$\tilde{h}_i(\mathbf{s}_{ij}) \leq 0, (i,j) = (1,1), ...(N, M-1)$

B. Creation of an initial guess

In order to solve the global optimization problem with all the doors, the solver needs an initial guess from which it starts and try to optimize the trajectory. The choice of this initial guess is therefore very important.

The algorithm 1 to find the initial guess can be described as follows: We're solving an optimization problem between only two doors at the time; We perform the previous task for all possible door combinations, so we get a table containing all the times needed to travel from a goal to another one; The TSP algorithm is then used to find the time optimized path and thus the order in which to pass through the doors; The set of optimal paths found is put end-to-end in order to generate the initial guess.

A few simplifications need to be taken to reduce the number of cases to be dealt with:

- Doors are considered as fixed points, to guarantee continuity of paths when they are placed end-to-end. This implies that the robot can only pass through the center of each door.
- Initial and final points are therefore also considered as doors.
- Doors can only be passed through in two directions, with two possible angles corresponding to the perpendicular to the segment between the two extremities of the door. The figure 8 shows these two possibilities.

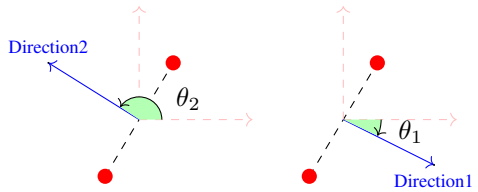

Fig. 8: Two possible angles when crossing a target

Although in reality we only have N doors in total to cross (one of which corresponds to the initial and final points), our problem theoretically comprises $G = 2 * N$ goals .

Algorithm 1 Initial guess creation

$G \leftarrow 2 * N$
$doors \leftarrow doors\ information$ ▷ position, angle, speed
$time \leftarrow empty\ table\ of\ size\ N * N$
$i,\ j,\ k \leftarrow 0$
$x,\ y,\ angle,\ speed \leftarrow [\ ]$
while $i \neq N$ **do**
  while $j \neq N$ **do**
    $time[i][j] \leftarrow OPC\_Solve(gates[i], gates[j])$
    $j \leftarrow j + 1$
  end while
  $j \leftarrow 0$
  $i \leftarrow i + 1$
end while
$Path \leftarrow Traveling\_Salesman\_Problem(time)$
while $k \neq size(best\_path) - 1$ **do**
  $x1, y1, \theta1, v1 \leftarrow OPC\_Solve(Path[k], Path[k+1])$
  $Concatenate[x, x1]$
  $Concatenate[y, y1]$
  $Concatenate[angle, \theta1]$
  $Concatenate[speed, v1]$
  $k \leftarrow k + 1$
end while
return $x, y, angle, speed$

1) OPC problem resolution between two goals: To solve the door-to-door optimization problem, we use a simplification of the general problem with only two goals. The cost function J in equation 8 summarizes the purpose of passing from initial position $(x_0, y_0, \theta_0, v_0)$ to the end position $(x_T, y_T, \theta_T, v_T)$ while minimizing the time to do so. The problem is subject to the same equality and inequality

constraints mentioned in the global problem, that include the dynamic model, boundary conditions and path constraints:

$$\min_{x(\cdot),u(\cdot),T} J = \int_0^T dt \tag{8}$$

subject to

Dynamic Model: $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)),\ t \in [0, T]$
Boundary Conditions: $x(0) = x_0,\ y(0) = y_0,\ \theta(0) = \theta_0,\ v(0) = v_0,\ x(T) = x_T,\ y(T) = y_T,\ \theta(T) = \theta_T,\ v(T) = v_T$
Path Constraints: $h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0,\ t \in [0, T]$

The continuous problem is then discretized and the initial guess for this simplified problem is defined to distribute the M control intervals as follows in figure 9. The first two steps (1 and 2) move orthogonally to the starting door, in the direction in which the robot is to start (depending on the theta angle). The next two steps (3 and 4) involve moving vertically upwards (respectively downwards) if the door to be reached is higher (resp. lower) than the initial door: $Y2 - Y1 > 0$ (resp. $Y2 - Y1 < 0$). The last two steps (M-1 and M) also move orthogonally to the starting goal, in the opposite direction to where the robot should stop (depending on the theta angle). The two forward steps (M-3 and M-2) also concur to move vertically downwards (respectively upwards) if the door to be reached is higher (resp. lower) than the initial door: $Y2 - Y1 > 0$ (resp. $Y2 - Y1 < 0$). Finally, the other points are evenly distributed on a straight line between steps 4 and M-3.

This initial guess avoids many infeasible problems, for example when two doors are parallel (same value of x or same value of y). The problem between the two doors is then solved using the method explained in section III-A.

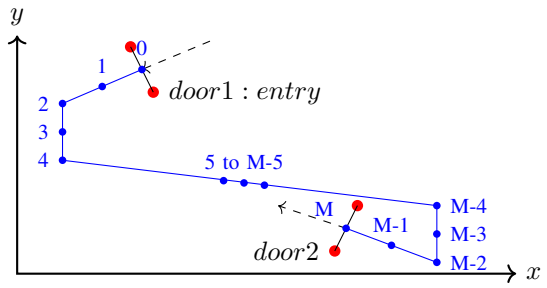

Fig. 9: First guess for the door to door resolution

2) Traveling Salesman Problem: Let's consider a number N of goals to be reached (it is assumed that the start and end points are also goals). Once all the combinations between all the doors have been tested, we have a table listing all the times taken to get from one goal to another, of size 2*N (as explained in the previous point).

We want to calculate the travel time for each possible path, in order to find the optimal one. To do this, we start by generating the set of possible path permutations; this corresponds to a combination of N doors from the 2*N available at the start. So, in total, there are $C(2N, N) = \binom{2N}{N} = \frac{(2N)!}{N!(2N-N)!} = \frac{(2N)!}{N!^2}$ possible permutations. This number can be significantly reduced by applying the following conditions:

- Each permutation must start with door 1, which is the starting point.
- Each permutation must end at target 2N, which is the end point.
- It is not possible to pass through the same real door twice, even if the angle is different. This last constraint drastically reduces the number of possible permutations.

For the remaining permutations, we add up the time needed to pass successively between each door. We then keep the minimum time, and the corresponding permutation gives the order and angle of the doors to be passed through. The following algorithm 2 summarizes this:

Algorithm 2 Traveling Salesman Problem

```
min_time ← ∞
best_path ← None
i ← 0
all_permutations = permutations(2N, N)
keeped = filter_permutations(all_permutations)
while i ≠ length(keeped) do
    time = global_time(keeped[i])
    if time ≤ min_time then
        min_time ← time
        best_path ← keeped[i]
    end if
    i ← i + 1
end while
return min_time, best_path
```

C. OCP global solution

The OCP global solution uses the values generated in the TSP algorithm as initial guess for the global discretized optimal control problem. That way, the number of infeasible solutions should decay, the processing for solution should be faster and the solution would represent the best sequence of doors possible.

IV. Evaluation

In this section, we evaluate the performance of the proposed approach for solving optimal control problems across three distinct scenarios. The evaluation is performed in simulation using CasADi, which is an open-source tool for numerical nonlinear optimization and algorithmic differentiation. The NLP solver used was the Interior Point Optimizer (Ipopt). The code is available in GitHub in the following repository: `https://github.com/emanueliwanow/pamcar`.

Thes scenarios are 2D environments, represented in Figure 10, in which the model always starts and ends at the position $x_0, y_0 = 0$. Each scenario provides different location for the doors and is designed to be progressively more challenging, which can show the range of applicability of our method as well as its limitations. We aim to verify the effectiveness of

the proposed method in achieving optimal trajectories across different situations and investigate the impact of varying problem characteristics on the solution.

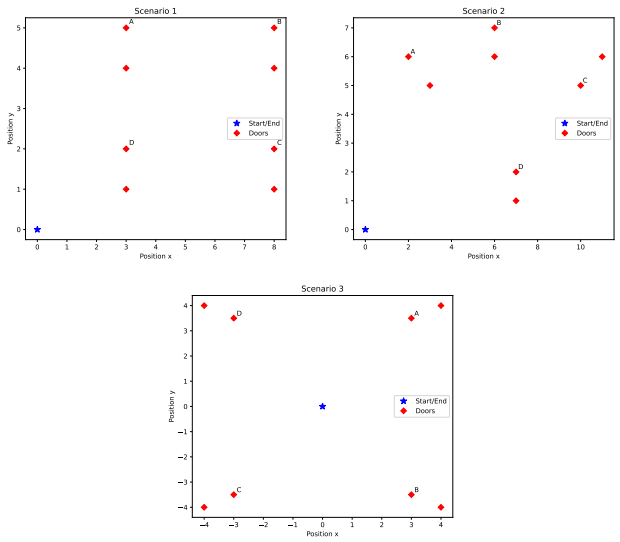

Fig. 10: Scenarios chosen for the evaluation of the method. The first scenario (upper left) has 4 doors parallel to one another. The second scenario (upper right) constitutes an oval-shaped track. The third scenario (lower) requires the car to start in the center and go through 4 doors around it.

For abalation studies, we analysed the final solution with and without our TSP method. Firstly, we run the global optimal control algorithm only with a random goal order and an interpolated initial guess. Then, we run the algorithm with the goal order provided by the TSP and a simple interpolated initial guess. Then we will compare the results from the OCP problem using the developed TPS algorithm as initial guess, including the goal ordering. On the final test, we insert obstacles on the environment to test the algorithm with TSP goal ordering and the simple interpolated initial guess.

The task parameters have a big influence on the solution. The greater the length of the car L, for example, the more open the curves must be. The parameters chosen are presented in table I and were selected after being considered compatible with the scenarios proposed.

TABLE I: Task Parameters

Symbol	Value	Unit	Description
L	1.6	m	Length of the car
δ_{min}	$-\pi/4$	rad	Minimum steering angle
δ_{max}	$\pi/4$	rad	Maximum steering angle
a_{min}	-5	m/s^2	Minimum acceleration
a_{max}	5	m/s^2	Maximum acceleration
ac_{min}	-2	m/s^2	Minimum centripetal acceleration
ac_{max}	2	m/s^2	Maximum centripetal acceleration
v_{max}	18	m/s	Maximum velocity
$(x,y)_{max}$	20	m	Maximum x,y position
$(x,y)_{min}$	-20	m	Minimum x,y position
θ_0	π	rad	Initial heading angle
θ_N	$-\pi$	rad	Final heading angle
v_0	0.3	m/s	Initial speed door to door TSP
v_T	0.3	m/s	Final speed door to door TSP
M	20	-	Number of control intervals

A. OCP global with interpolated initial guess and goal order from TSP

For this part of the evaluation, the initial guess used for the solver was a linear interpolation between the middle points of each door, and the sequence of doors in which the model visits was obtained from the TSP algorithm. In the first scenario, the solver was able to find a optimal solution and it can be seen Figure 11. The interpolated initial guess given is clearly not feasible, since it doesn't respect the car dynamics. The long curved trajectory reflects the influence of task parameters chosen on the constraints for the trajectory planning, specially the length of the car L and the maximum centripetal acceleration ac_{max}. It is interesting to note how the point at which the car crosses each door varies along the doors's extremities according to the ideal global trajectory as a consequence of the inequality constraint applied to the optimization variables ϕ_i.

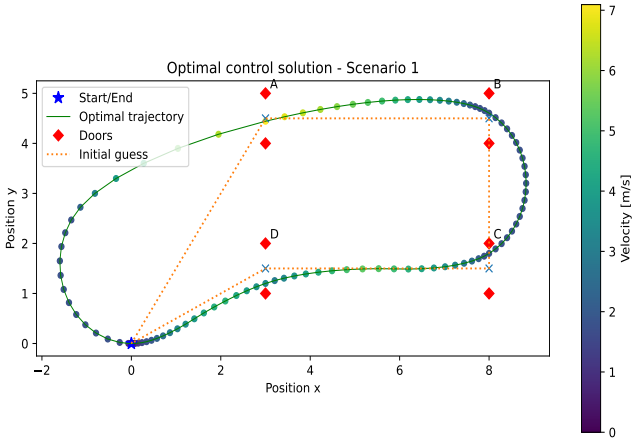

Fig. 11: Solution on scenario 1

For the second scenario, as seen in Figure 12, the solver was also able to find a solution. We can see how the velocity between door A and door B was very fast and the time steps T_{ij} are very closed to each other, which reflects the discretization method that allows different values of T_i, while keeping the same number of control intervals M for all intervals.

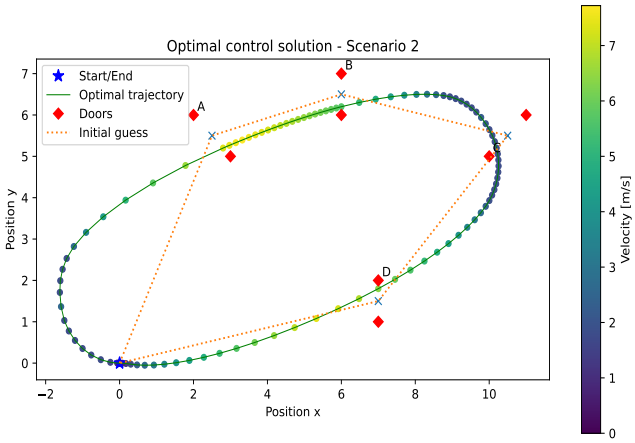

Fig. 12: Solution on scenario 2

For the third scenario, however, the solver did not find a solution as it converged to a point of local infeasibility.

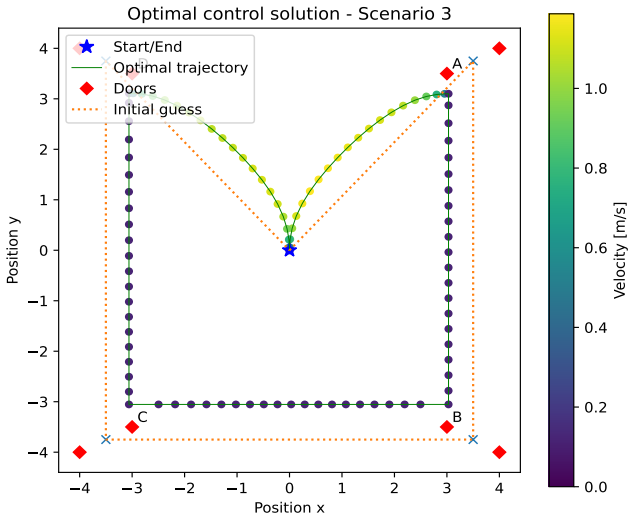

Fig. 13: Infeasible solution on scenario 3

B. OCP global with randomly ordered doors and interpolated initial guess

In this experiment, we don't inform the algorithm the optimal order in which to visit the goals obtained by the TSP. As we can see from the following figure 14, if the order of the doors is randomized for interpolation, we obtain an unfeasible problem with no solution. This is not what happens in the previous example of figure 11, where the initial interpolated guess is already relatively optimizes at the start. Thus, the importance of the TSP algorithm for goal ordering.

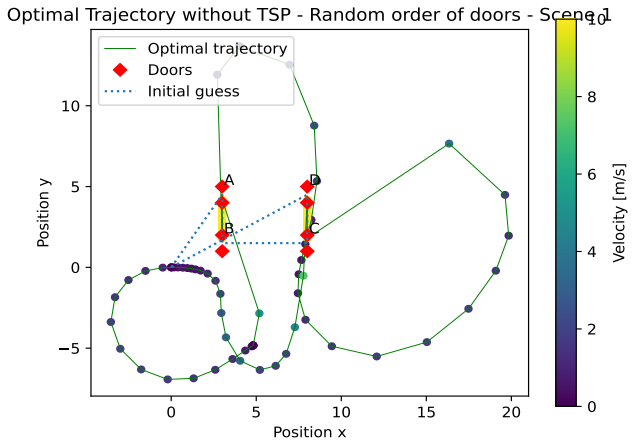

Fig. 14: Infeasible solution on scenario 1 with random ordered doors

C. OCP global with TPS for goal ordering and initial guess

Now using the OCP global solution with the TSP algorithm output not only for the goal ordering, but also for a feasible initial guess (x, y, θ, v), we have the following results for scenario 1, 2 and 3 in figure 15,16 and 17 respectively.

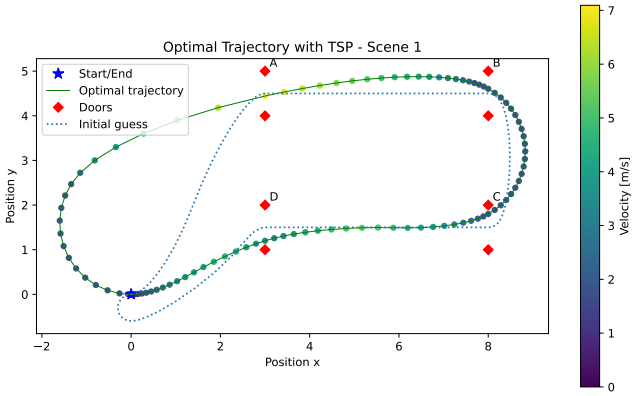

Fig. 15: Solution with TSP on scenario 1

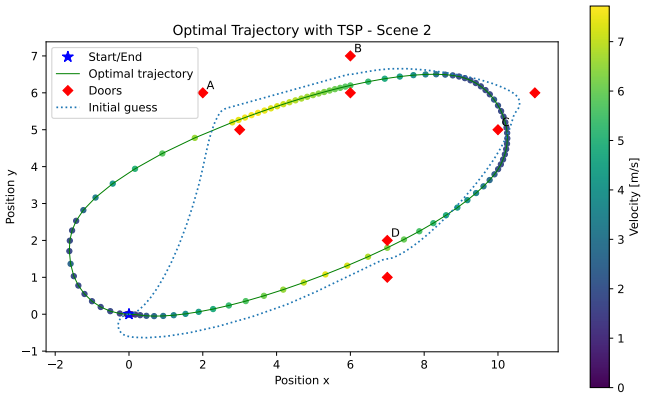

Fig. 16: Solution with TSP on scenario 2

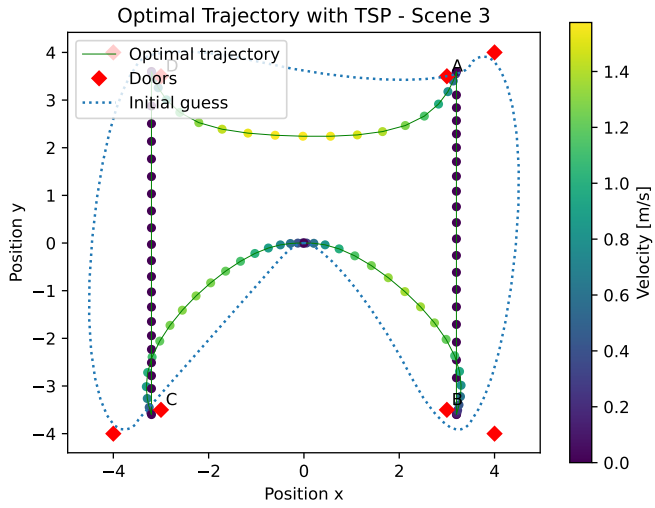

Fig. 17: Infeasible solution with TSP on scenario 3

It can be seen that again for scenerio 3 the whole algorithm wasn't able to find a solution given that it only found a local infeasibility.

D. OCP with obstacles

For the case with obstacles, we used the interpolated initial guess as it proved to be realible for scenerio 1 and 2, and the results can be seen in figure 18 and 19. The infeasible solution for scenario 3 was omitted given that the algorithm did not work for the previous cases in this scenario.

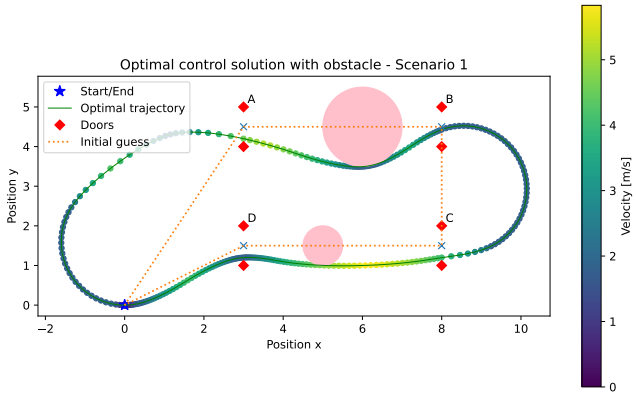

Fig. 18: Solution with obstacles on scenario 1

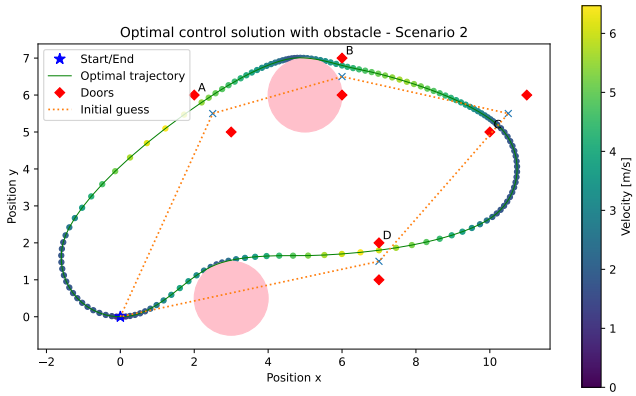

Fig. 19: Solution with obstacles on scenario 2

E. Comparing results

Finally, the table I compares the total time for the model to complete each task. In the first scenario, the algorithm with random goal ordering and interpolated initial guess, found no solution. However, when we fed the initial guess for goal ordering obtained by the TSP, for the first and second scenarios, the algorithm found a feasible solution, even though the initial guess was a unfeasible interpolation. When adding a feasible initial guess obtained from the TSP algorithm, the same solution as the previous case was found by algorithm, with exactly the same total times. This may be an indication that these cases are easy and don't require a feasible initial guess. The third scenario, however, was too difficult for both algorithms to successfully plan a trajectory. Ideally, a slightly more difficult scenario than the first and second ones should be tested, in which the interpolated initial guess fails.

TABLE II: Time to complete each task

Scenario	Total time only OCP	Total time only OCP random order	Total time OPC+TSP	Total time only OCP with obstacles
1	9.00	-	9.00	11.28
2	8.85	-	8.85	10.27
3	-	-	-	-

V. Conclusion

In this report, we have presented a method for solving a multi-goal trajectory optimization problem based on optimal control techniques for a static obstacle course. We formulated our optimal control problem to incorporate both the dynamic constraints of our car and the static constraints of the environment. By solving the TSP for the permutations every door-to-door trajectory generated by a simplified optimal control problem, we could provide a feasible initial guess for the global optimal control problem.

Here we compared the global solution with and without a feasible initial guess and optimized goal ordering. Our results show the relevance of the goal ordering provided algorithm for obtaining a solution. The relevance of the feasible initial guess for states x, y, θ and v was not attested, since the results for both cases were similar. Our conclusion was that the algorithm should be tested in more scenarios to attest the relevance of a good initial guess provided by the TSP algorithm. In light of the unexpected results, we verified that the constraints for initial and final heading angles were not being correctly included in the numerical calculations. Further work would include a better mathematical formulation for the angle constraints and a more detailed look into the CasADi Ipopt solver.

In conclusion, our report details a step-by-step way of formulating an optimal control problem for multi-goal trajectory planning. We highlight the importance of a very precise formulation and parameters for a complex problem such as ours, as poorly formulated constraints lead to infeasible results.

References

[1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[2] J. Janos, V. Vonasek, and R. Penicka, "Multi-goal path planning using multiple random trees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, p. 4201–4208, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3068679

[3] S. Ishida, M. Rigter, and N. Hawes, "Robot path planning for multiple target regions," 09 2019, pp. 1–6.

[4] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[5] J. Robinson, *On the Hamiltonian game (a traveling salesman problem)*. Rand Corporation, 1949.

[6] V. Freire and X. Xu, "Optimal control for kinematic bicycle model with continuous-time safety guarantees: A sequential second-order cone programming approach," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 681–11 688, 2022.

[7] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[8] M. Erhard, G. Horn, and M. Diehl, "A quaternion-based model for optimal control of the sky-sails airborne wind energy system. zamm–journal of applied mathematics and mechanics (2016)."