2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Informed Hybrid A Star-based Path Planning Algorithm in Unstructured Environments

Antonio Acernese, Giulio Borrello, Luca Lorusso and Michele Basso

Abstract— **The continuous improvements of sensors' accuracy, communication frameworks, and computing technologies, have pushed the horizon of autonomous driving to a new branch of mobility, which enhances safety, efficiency, and convenience of automotive transportation. A fundamental step to the success of these systems is the design of a robust, safe, and sample-efficient decision-making module. However, real-world applications to semi-structured or even unstructured environments, such as home zones, parking valets, and narrow passages, are very limited. This paper proposes Informed Hybrid A Star (InHAS), a new computationally lightweight path planning algorithm which efficiently provides optimal paths while taking into account vehicle dimensions and satisfying non-holonomic constraints. We validate the effectiveness of the proposed method both in simulation and in real-world application.**

I. Introduction

In the last decades, research community has put considerable effort into autonomous driving (AD). AD cars, also known as self driving (SD) cars and driverless cars, lay the foundation for a new mobility, which prioritizes performance and safety of transportation systems, leading to reduce road fatalities, enhance traffic efficiency, and empower new services such as mobility on demand [1].

To execute complex tasks in an autonomous way, AD technology inherits some ideas from robotics theory. Particularly, AD task can be roughly divided into two main categories: the perception and the decision-making systems. Similarly to human driving, the perception module aims at observing the surrounding of the AD vehicle, i.e., the structure of the road, the relevant obstacles, traffic rules, etc [2]. The output of this phase is an estimate of the state of the SD vehicle, in terms of position, heading angle, and velocity, as well as an estimate of the states of the dynamic objects in the surrounding of the vehicle, and static road and traffic rules information. The second system, i.e., the decision-making, continuously exploits data coming from perception and generates control actions to move the vehicle; it is commonly partitioned into a hierarchical structure comprised of four main modules [3]: route (or mission) planner [4], behavioural planner [5], [6], local (or motion) planner, and motion (or low-level) control (see Fig. 1).

Particularly to the motion planner (MP), it can be further divided into path planning (PP) and trajectory planning (TP) modules. The former decides how to proceed from the current state of the vehicle to a goal state, while guaranteeing vehicle feasibility (black line in Fig. 1). The output is a static path that does not take into account the time. The latter computes the commands, i.e., steering wheel angle and acceleration, required to follow such a trajectory, including the time constraint in terms of reference velocity (green line in Fig. 1).

In the last years the AD community has put considerable effort into PP. This task has been addressed through sampling-based, interpolation-based, and optimization-based methods [3]. Sampling-based methods are based on incrementally growing a graph rooted in the current state of the vehicle by sampling the surrounding state space. The major challenge is how to construct a reliable discretization of the environment that approximates well the reality. Different solutions have been proposed to solve this problem, and promising results come from Dijkstra-based algorithms [7], [8], and A-star-based ones [9], [10]. Moreover, the authors in [11] proposed the anytime Dstar to compute a path for the SD car, and [12] proposed the Hybrid A star (HAS) to generate smooth paths in unknown environments.
Interpolation-based methods use a given set of waypoints of a route (e.g., generated with sampling-based methods) to generate a new set of points forming a smoother path that respects the vehicle and environment constraints. These techniques suffer some major problems, such as significant tracking error in position and orientation of the AD vehicle, and the lack of a closed form expression [13], [14]. Recently, optimization-based PP has gained attention from the AD community, due to the continuously increasing computing power and wide choice of numerical optimization tools. These approaches are attractive due to the possibility of explicitly capturing the vehicle model, in the form of kinematic, dynamic and actuator constraints [15]. However, when dealing with non-linear and non-convex frameworks, guaranteeing real-time performance and convergence becomes a big challenge, and the application of those methods is mainly limited to simulated environments.

Following this stream of research, this work focuses on the problem of PP for an automated vehicle (AV) operating in unknown environments. An AV is an intermediate system between traditional vehicles with no automation, and SD ones, where the driver is not expected to intervene at all [16]. Although both perception and decision-making modules have been designed, we exploit the MP module in more details, assuming that the AV has adequate sensing and localization capabilities. Particularly, we:

- propose a computationally lightweight MP algorithm, namely the Informed HAS, which provides optimal

A. Acernese, G. Borrello, L. Lorusso and M. Basso are with the Department of Automated Driving Control, CRF, Strada Torino 50, 10043 Orbassano, Italy. {antonio.acernese, giulio.borrello, luca.lorusso, michele.basso}@stellantis.com.

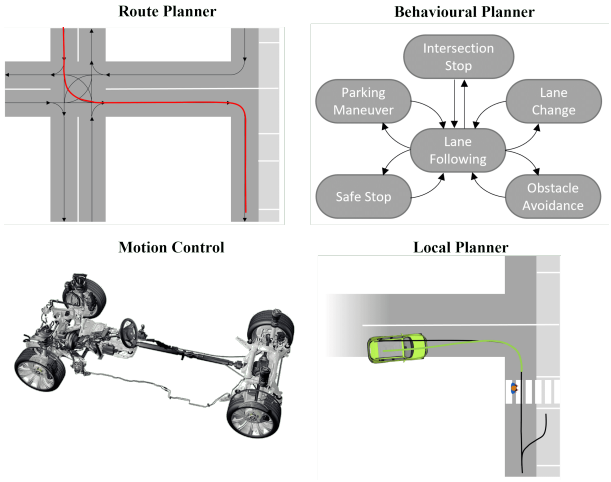

Fig. 1: AD decision making structure.

paths in unstructured environments, while guaranteeing real-time feasibility;
- design a model in the loop to simulate the proposed AV scheme using Matlab and IPG CarMaker tools;
- validate the effectiveness of the proposed approach on different real-wold scenarios, through a real proof of concept (PoC) of the AV.

II. Preliminaries

In this section we recall the vehicle model used to create feasible paths in the PP module, and we provide formal problem definitions of the PP and TP, respectively.

A. Vehicle Model

For the application of AVs to low speed scenarios, a good vehicle approximation that can be used in the PP module to take into account vehicle's dynamics and generate feasible trajectories is the kinematic bicycle model. With the term low speed we refer to all the applications where the longitudinal velocity is limited to the range $v_x \in [0, 20]\ km/h$, e.g., parking valets, dwelling surroundings, and narrow passages.

The 4 degrees of freedom (DoF) kinematic bicycle model is one of the most popular models in MP [17]. Using this approximation, the vehicle is represented as a single track model, where both left and right wheels are fused together into two wheels positioned at the centers of the axles (see Fig. 2), front and rear, respectively. Consequently, the tire force difference of the two wheels, as well as the side-slip angles are neglected.

Under these assumptions, let $x \in \mathbb{R}$, $y \in \mathbb{R}$ be the $X-Y$ Cartesian coordinates in the world frame (WF), and $\psi \in \Psi := [-\pi, \pi]$ describes the orientation (or yaw angle) of the vehicle in the WF. Moreover, let $v \in V := [v_{min}, v_{max}]$ and $a \in A := [a_{min}, a_{max}]$ denote the velocity and acceleration of the vehicle, respectively, within acceptable ranges. Note that in the sequel we refer to the longitudinal (along the X axis) and lateral (along the Y axis) components of a measurement with the subscripts $(\cdot)_x$ and $(\cdot)_y$, and the x and y Cartesian coordinates are always referred to the center of the rear axle of the vehicle.

The steering angle of the front wheels of the vehicle can be represented as $\delta \in [\delta_{min}, \delta_{max}]$, assuming that the rear wheels cannot be steered, where the limitation on maneuverability δ_{min} and δ_{max} is referred to as a non-holonomic constraint [18].

The corresponding kinematic bicycle model of the vehicle can be represented as

$$\begin{cases} \dot{x} = v\ cos(\psi) \\ \dot{y} = v\ sin(\psi) \\ \dot{\psi} = \frac{v}{L}\ tan(\delta) \\ \dot{v} = a \end{cases}, \tag{1}$$

where L is the wheelbase of the vehicle. Thus, the state and input vectors of such a model can be defined as $\mathbf{X} := \{x, y, \psi, v\}$ and $\mathbf{U} := \{\delta, a\}$, respectively.

B. Path Planning

Oftentimes, PP can be formulated as an optimization problem that takes into account the travel time, comfort, action efforts, for example. The 4 DoF model (1) forms a 4-dimensional state space $\mathcal{C} \subseteq \mathbb{R}^2 \times \Psi \times V$, commonly called configuration space [19]. Specifically to the PP framework, one typically neglects the velocity profile, as it is demanded to the lower level TP module, i.e., $v = \bar{v}$ is assumed to be a constant in the PP. Under this assumption, the configuration space becomes $\mathcal{P} := \mathbb{R}^2 \times \Psi \times 1 = \mathbb{R}^2 \times \Psi \subseteq \mathcal{C}$. An N-length path is a sequence of configurations, i.e., $P := \{p_1, p_2, \ldots, p_N\} \subseteq \mathcal{P}$, where $p_i = \{x_i, y_i, \psi_i, \bar{v}\}$. The optimal PP problem can be then formulated as follows.

Proposition 2.1: Let $\mathcal{P}$ be the configuration space of the vehicle, and let its initial configuration be $p_{start} \in \mathcal{P}$. The path is required to end in a goal region $\mathcal{P}_{end} \subseteq \mathcal{P}$, given that the set of all allowed free configurations is $\mathcal{P}_{feas} \subseteq \mathcal{P}$. Then, the optimal PP problem is to find a path

$$\begin{aligned} P^* = \underset{\{p_1,\ldots,p_N\}\in\mathcal{P}}{\arg\min}\ & \mathcal{J} \\ \text{subject to}\quad & p_1 = p_{start} \\ & p_N \in \mathcal{P}_{end} \\ & p_i \in \mathcal{P}_{feas} \qquad i = 1, \ldots, N \\ & \text{system (1)} \\ & v = \bar{v}, \delta \in [\delta_{min}, \delta_{max}] \end{aligned}, \tag{2}$$

where $\mathcal{J}$ is the cost functional.

C. Trajectory Planning

The second MP module is the TP, that, given a generated path $P^* := \{p_1, p_2, \ldots, p_N\}$, post-processes it to (i) enhance comfort, and (ii) explicitly combine the restricted configuration space $\mathcal{P}$ with the execution time of the path. The time can be included in terms of velocity profile to be followed. The TP can be formulated as a tracking problem, and several methods coming from different application domains have been proposed from research community, e.g., model predictive control (MPC). One of the biggest challenges when dealing with optimization methods is the computational load

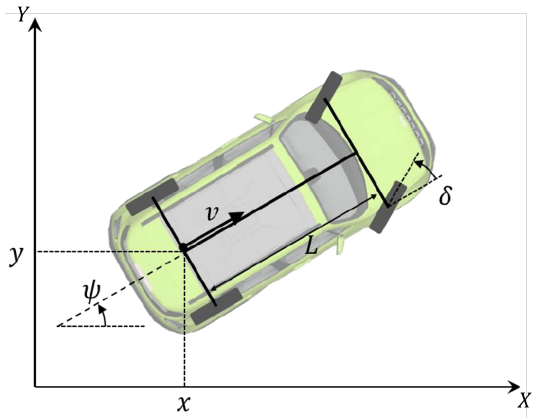

Fig. 2: 4 DoF kinematic bicycle model.

and the corresponding time to obtain an effective solution. To solve this issue and allow valuable trajectories in real-time, we formulate the TP problem as a combination of two distinct linear MPC problems: one related to the lateral dynamics, and the other referred to the longitudinal ones. Specifically, given the non-linear model (1), we apply a time-state control form (T-SCF) transformation to describe the lateral dynamics as a system of linear, differential equations, namely

$$\begin{bmatrix} \frac{\partial z_2}{\partial z_1} \\ \frac{\partial z_3}{\partial z_1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mu_2, \tag{3}$$

where $z_1 = x$, $z_2 = y$, $z_3 = tan(\psi)$, and $\mu_2 = \frac{tan(\delta)}{L\ cos^3(\psi)}$. Furthermore, the longitudinal dynamics can be well-approximated with a double integrator model, whose states are the travelled distance $\xi \subseteq \mathbb{R}$, and the vehicle speed $v \in V$:

$$\begin{bmatrix} \frac{\partial \xi}{\partial t} \\ \frac{\partial v}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \xi \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a. \tag{4}$$

This formulation allows to express the original non-linear model (1) as a combination of two, independent, linear models. Then, one can discretize those two models, e.g., using the Tustin discretization, and apply two independent linear MPCs to efficiently solve the TP task. For further details on the adopted MPCs, refer to [17], where a detailed model transformation can be found, together with the control design explanation and implementation.

III. Methodology: Informed Hybrid A Star

In this section we introduce our PP procedure to use in unstructured environments, and we propose the novel Informed HAS (InHAS) algorithm to speed-up path exploration. To this end, assume to have from the perception module static information about the scenario under test, e.g., the occupancy grid map. Moreover, assume to be in an unstructured environment, such as a parking valet, and to have a maneuver $\bar{P} \subseteq \mathcal{P}_{feas}$ that the driver recorded and stored on the car devices. The goal is to optimize such a maneuver (w.r.t. some criteria such as the travelled distance or the number of direction changes) through the PP, and replicate it whenever the driver falls again in the same scenario.

The proposed InHAS algorithm is essentially based on the combination of two main methods: the area optimization (AO) and the HAS. The AO module is responsible to "inform" the HAS about the most likely regions of the occupancy grid map to explore, thus strongly speeding-up the PP process. Then, the HAS algorithm computes a drivable path in such areas. The resulting path guarantees the following AD metrics:

- Computationally efficient, due to the AO
- Feasible and drivable, that is inherent in the HAS algorithm
- Safe and robust, assured by a continuous collision check over the perimeter of the vehicle.

A. Hybrid A star

The traditional A star algorithm is a well-known method in PP [9] that uses an occupancy grid map to discretize the configuration space $\mathcal{P}$ and represent it through a finite set of nodes. Given an initial configuration p_{start}, A star assigns a cost $F_i = G_i + H_i$ for each surrounding node i, where G_i represents the cost to reach the i-th node from p_{start}, and H_i is an estimate representing an heuristic cost to reach the target configuration p_{end} from node i. The main limitation of A star is that it works only for holonomic robots. The HAS algorithm extends the traditional A star to non-holonomic vehicles [12], leveraging the vehicle's kinematics (1) to estimate its exact motion. It provides two main improvements w.r.t. A star; first, the term hybrid refers to a modified state-update rule that maps continuous state data (x, y, ψ) into discrete search space of normal A star. Second, the HAS extends the configuration space $\mathcal{P} \subseteq \mathbb{R}^2 \times \Psi$ to include vehicle's direction $r := \{0, 1\}$, for reverse and forward motions, respectively[1].

The general HAS algorithm is reported in Algorithm 1, where n, m are the number of rows and columns of the occupancy map, with a cell resolution xy_grid_res, and a yaw resolution yaw_grid_res. Moreover, the algorithm takes in input the vehicle dimensions, and the motion resolution (mot_res), that corresponds to the minimum distance that the vehicle can travel at each time-step. The function $popNode$ extracts the current node based on the cost functional. Then, the function $RS\_expansion$ computes Reeds-Shepp (RS) curves, trying to link the current node with the target node through the combination of curves and straight segments [20]. The function $updateSets$ updates the costs associated to the current node, and generates n_steer new nodes based on mot_res and using vehicle dynamics.

In the algorithm, $SB\_C, B\_C, S\_CH\_C, S\_C, H\_C$ represent the weights that form the cost function, which we defined as:

[1]With a slight abuse of notation, to lighten the symbols we represent the new configuration space as $\mathcal{P}$, assuming that when we talk about HAS it is a 4-dimensional space.

$$
\begin{aligned}
&F_i = G_i + H_i \\
&H_i = H\_C\sqrt{(x_i - x_{end})^2 + (y_i - y_{end})^2} \\
&G_i = G_{i-1} + G_{res} + G_{SB} + S\_C \cdot |\delta_i| \\
&\qquad + S\_CH\_C \cdot |\delta_i - \delta_{i-1}| \\
&\text{s.t.} \quad G_{res} = \begin{cases} mot\_res, & \text{if } r_i = 1 \\ B\_C \cdot mot\_res, & \text{if } r_i = 0 \end{cases} \\
&\qquad G_{SB} = \begin{cases} SB\_C \cdot mot\_res, & \text{if } r_i \neq r_{i-1} \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \quad . \tag{5}
$$

Intuitively, we keep the traditional contributions on the travelled distance G_i and the heuristic distance H_i, but we add some additional terms to: (i) smooth the trajectory (last two terms proportional to the steering control), and (ii) prefer forward motion (G_{res} and G_{SB} are functions of the travelling direction).

Furthermore, we introduce a procedure in the $RS\_expansion$ function to effectively use the HAS in real-world applications. RS curve theory states that it is always possible to connect two configuration points with a combination of straight segments and minimum radius curves, and the generated sequence always terminates with a curve, eventually with near-zero length if it is not necessary. However, RS curves themselves do not take into account vehicle dimensions. It is clear that, when dealing with narrow passages or tight parking valets, traditional RS curves fail to enter without collisions, and generate non-optimal paths. For the sake of an example, consider one of the scenarios we have covered in the experimental results, representing a real-world home-parking, in Fig. 4. The results of traditional RS curves is reported in red in the figure. To allow safe and robust PP, we perform a collision check between the vehicle and static objects during the RS paths generation, with a resolution of mot_res. Moreover, we modify RS curves generation; at each iteration of the algorithm, we compute both normal RS curves and additional RS paths constrained by a straight segment at the end of the curve. This choice, while violating the minimization of the travelled distance, assures a collision-free path in such scenarios (see the green path in Fig. 4), and helps the PP to speed-up the learning process. Indeed, in the example shown in Fig. 4, the algorithm finds a valid path after only 3 iterations, while in the the original HAS with the AO phase (described below) it increases to 253 iterations. Surprisingly, if one considers the original HAS without any additional improvements, the number of iterations before finding a valid solution explodes to 1751, even though it is not a comfortable path.

B. Pre-processing: Area Optimization

The HAS algorithm alone is essentially based on a deep and myopic exploration phase. It turns out that there could be some scenarios in which basic HAS fails to easily find a path. For example, assume the case of Fig. 3; here, p_{start} and p_{end}

Algorithm 1 Hybrid A^*

Input: Grid map $GM : \mathbb{R}^{n+m} \rightarrow [0, 1]$, p_{start}, $\mathcal{P}_{end}$
$cfg \leftarrow \{veh\_dims, mot\_res, n\_steer, xy\_gm\_res, yaw\_gm\_res, gm\_dims, SB\_C, B\_C, S\_CH\_C, S\_C, H\_C\}$
Output: Optimal planned path P^*

$P^* \leftarrow \emptyset$
$openSet \leftarrow \{p_{start}\}$
$closedSet \leftarrow \emptyset$
while $\neg isempty(openSet)$ **do**
  $[curr\_node, openSet] \leftarrow popNode(openSet, cfg)$
  $closedSet \leftarrow \{closedSet \cup curr\_node\}$
  $[isGoal, RS\_path] \leftarrow RS\_expansion(curr\_node, \mathcal{P}_{end}), cfg)$
  if $isGoal$ **then**
    $P^* \leftarrow getFinalPath(RS\_path, closedSet, cfg)$
    $break$
  end if
  $[openSet, closedSet] \leftarrow updateSets(curr\_node, openSet, closedSet, cfg)$
end while

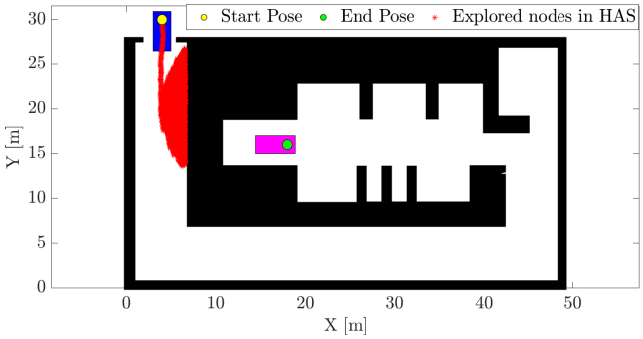

Fig. 3: HAS limits example.

are close each other, but due to the occupancy grid map, the path that connects those two points is more complex. In this situation, due to the contribution given by the heuristic cost H, the HAS algorithm will erroneously explore nodes that are in the neighborhood of p_{start} and in the direction toward p_{end}. To solve this issue and speed-up the learning process, we propose the following simple yet effective procedure, that we call area optimization (AO). Specifically, given a recorded non-optimal maneuver $\bar{P}$, instead of optimizing it through HAS from $\bar{p}_1$ to $\bar{p}_N$, one can think to optimize only specific segments, e.g., focusing on the areas in which there are maneuvers (one change in the direction of the motion), or multi-maneuvers (two or more changes). Consequently, instead of running the HAS on the whole path and the entire occupancy map, we apply *divide et conquer* policy by running the PP algorithm multiple times on reduced areas of the occupancy map, created by identifying the segments to be optimized. This procedure turns out to be very powerful in practice. Indeed, for each sub PP problem it limits the area in which the HAS can explore, thus filtering out non-sense paths. The output of the InHAS algorithm will be the combination of optimized paths segments (through modified HAS), and acceptable recorded segments from the original maneuver (such as straight lines). For example, consider the recorded maneuver represented in blue in Fig. 4. The AO method outputs a reduced occupancy map (represented as a black rectangle), and consequently the initial configuration related to InHAS (yellow point) is much nearer to the goal

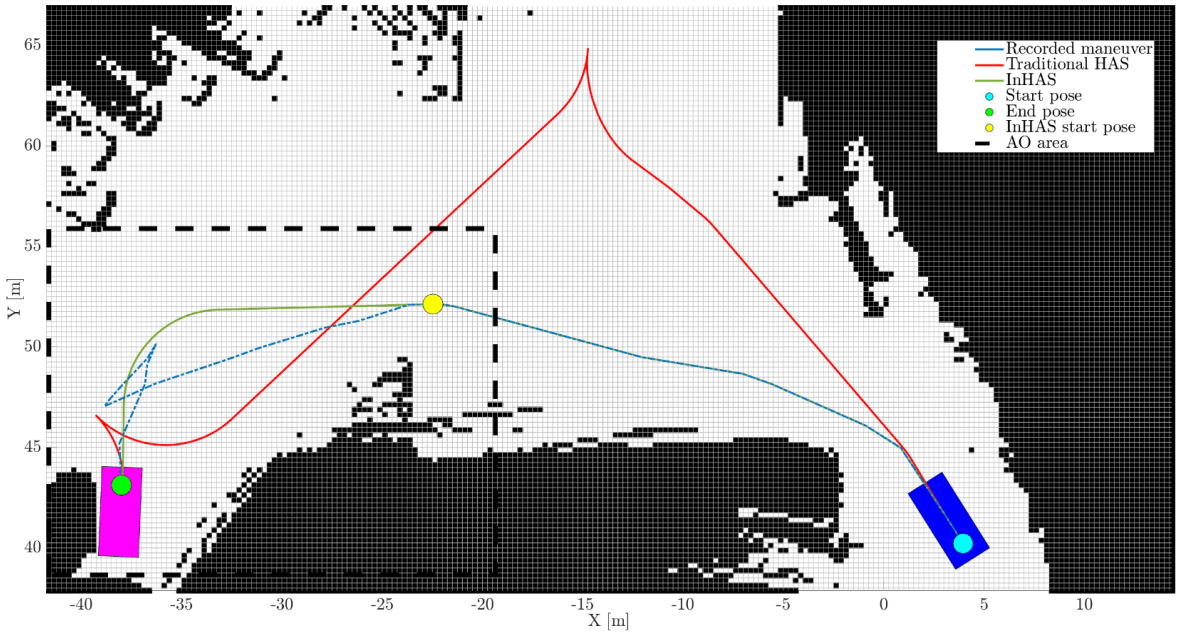

Fig. 4: Comparison between traditional HAS and InHAS. Required time to provide a path is 7.6 sec and 1.3 sec, respectively.

configuration (pink rectangle) than the initial configuration related to HAS (blue rectangle). This significantly reduces the computational time required (7.6 sec for HAS and 1.3 sec in the case of InHAS). Furthermore, in the case of unstructured environments where there is no high-level constraint on the direction to follow as well as rules of the road, traditional HAS could perform very bad. Indeed, the optimized HAS path (red line) is tremendously worse than the recorded one, in terms of travelled distance, and most importantly, length of the path travelled backward. By contrast, even if the proposed InHAS algorithm does not guarantee global optimal solutions, it provides a very acceptable optimized path (green line), obtained as a combination of the recorded maneuver and the modified HAS algorithm.

Remark 1: It is worth to note that we tried to lighten as much as possible the hardware required for the decision-making module. Thus, our solution satisfies many AD metrics, and we feel that it could be of interest for the application of MP to real-world scenarios.

IV. Experimental results

This section reports the experimental results of the proposed methodology on different real-world scenarios, and compares the performance with a model in the loop (MIL) simulation using IPG CarMaker tool and Matlab/Simulink.

A. Setup

The prototypal vehicles which has been used to design and test the proposed AD scheme is a Jeep Renegade, with dual dry clutch transmission (Fig. 5). It is equipped with a PwrPak7 Novatel dual antenna GNSS, twelve Valeo ultra sound scan sensors, six Arbe imaging radars, one RoboSense LIDAR, and four Fisheye cameras. The generated data are then fused and processed through a NUC PC and ROS environment. The MP module runs on an independent hardware, namely a dSPACE MicroAutobox II.

Fig. 5: Jeep Renegade prototypal vehicle.

B. Simulation and experimental results

The covered scenarios are represented in Fig. 6. Particularly to the scenario on the top, Fig. 4 shows the results of a perpendicular parking maneuver. As we already mentioned in the previous section, the proposed methodology outputs an optimized maneuver (green line) that performs much better than the traditional HAS, while minimizing the number of changes in the direction of the motion, ensuring a feasible and collision free trajectory.

The obtained results related to the other covered scenario (the one at the bottom of Fig. 6) are shown in Fig. 7. First, given a recorded maneuver (the blue dashed line), InHAS correctly optimizes it minimizing the travelled distance. Moreover, from the figure, one can appreciate the reliability of the MIL; indeed, the real and the simulated optimized paths (blue and red lines) are very similar each other. Fig. 8 shows the comparison between the followed trajectory on the PoC and the simulated one, along with a corridor that

Fig. 6: Selected scenarios.

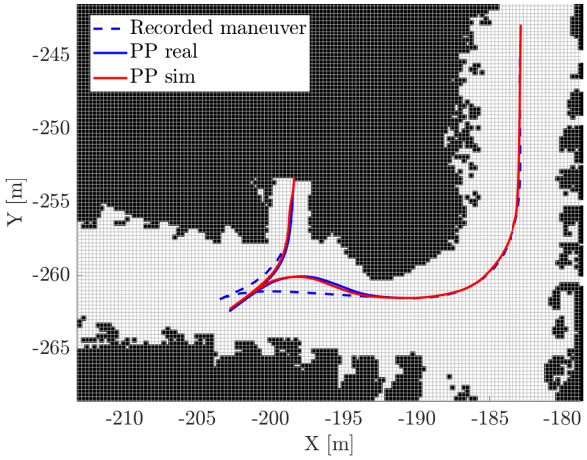

Fig. 7: PP performance, scenario 2.

determines the constraints provided to the MPC problem. It is worth noticing that such constraints are updated in real-time based on data coming from the sensor fusion, allowing the possibility to manage maneuvers such as safe stop and obstacle avoidance. Since the focus here is on the PP module, and due to space limits, we omit this situations that are closely related to the TP module.

REFERENCES

[1] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, pp. 6469–6486, 2021.

[2] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz, L. Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[4] K. Tsiakas, I. Kostavelis, A. Gasteratos, and D. Tzovaras, "Autonomous vehicle navigation in semi-structured environments based on sparse waypoints and lidar road-tracking," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1244–1250.

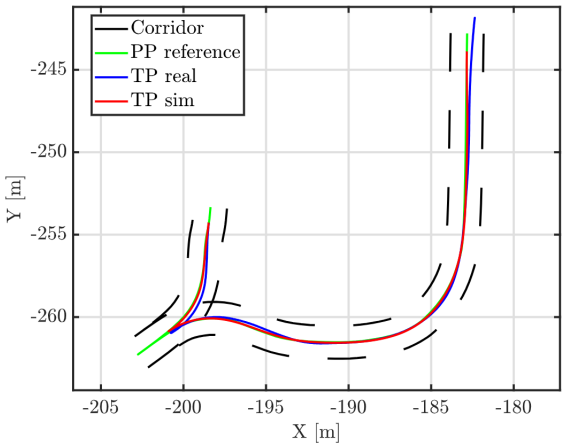

Fig. 8: TP performance, scenario 2.

[5] F. Havlak and M. Campbell, "Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 461–474, 2014.

[6] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic mdp-behavior planning for cars," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 1537–1542.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[8] R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta, "Safe and reliable path planning for the autonomous vehicle verdino," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 2, pp. 22–32, 2016.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *SIGART Bull.*, no. 37, p. 28–29, dec 1972.

[10] J. S. Leedy, Brett M.and Putney, C. Bauman, S. Cacciola, J. Michael Webster, and C. F. Reinholtz, *Virginia Tech's Twin Contenders: A Comparative Study of Reactive and Deliberative Navigation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 155–182.

[11] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, and J. Dolan et al., "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[12] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[13] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[14] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 482–500, 2018.

[15] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[16] S. International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," On-Road Automated Driving (ORAD) committee, Standard, 2021.

[17] G. Borrello, E. Raffone, C. Rei, and M. Fossanetti, "Trajectory planning and vehicle control at low speed for home zone manoeuvres," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 516–15 523, 2020, 21st IFAC World Congress.

[18] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[19] R. Murray and S. Sastry, "Nonholonomic motion planning: steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

[20] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, pp. 367–393, 1990.