An Efficient and Smooth Path Planner Based on Hybrid A* Search and Frenet Frames

Pin-Wen Wang, Yi-Chi Tseng, and Chung-Wei Lin(✉)

National Taiwan University, Taipei City, Taiwan
{b09902048,b09902028}@ntu.edu.tw, cwlin@csie.ntu.edu.tw

Abstract. As the technology of autonomous vehicles advances, the importance of automatic path planning also grows significantly. This leads to the exploration of diverse algorithms and learning-based techniques. While most methods safely and efficiently navigate vehicles to their destinations, the comfort of a journey is often overlooked. To address the issue, this paper focuses on a path planning algorithm that integrates the hybrid A* path planner [2] and the Frenet Frame trajectory generator [8]. We evaluate the performance of the proposed algorithm in terms of travel efficiency and passenger comfort. The experimental results demonstrate that the proposed algorithm better trades off travel efficiency and passenger comfort, compared with the pure Frenet Frame trajectory generator. The results also provide an insight that input preprocessing, even if it is a simple one, can affect Frenet Frame trajectory generator significantly, and it is worth future exploration.

Keywords: Autonomous Vehicles · Path Planning

1 Introduction

With the advance of technology, the design of autonomous vehicles has grown increasingly sophisticated. One critical challenge in autonomous driving is to ensure safe and timely arrival at the destination. It requires vehicles to plan paths from their current positions to the destinations.

Path planning in the context of autonomous driving has gathered considerable attention from researchers. Early efforts focused on searching algorithms, which have evolved to incorporate techniques such as path planning using reinforcement learning [3]. These algorithms aim to generate paths that not only lead to the destination safely without hitting obstacles but also adhere to the vehicle dynamics for optimal performance in terms of time efficiency. Reinforcement learning has emerged as a popular approach for path planning, as evidenced by existing studies [4,9]. These learning models can simultaneously optimize different objectives. The results obtained from applying reinforcement learning to path planning are promising. However, one limitation is the relatively high training time required for these learning models.

Y. Chen et al. (Eds.): SmartSP 2023, LNICST 552, pp. 75–82, 2024.
https://doi.org/10.1007/978-3-031-51630-6_5

Besides safety and efficiency which are usually the main objectives of existing research, passenger comfort is also a crucial objective for path planning. Human comfort is usually described as driving smoothly, which implies less changes in acceleration [6]. For example, existing path planning algorithms may generate paths that efficiently navigate twisted or tortuous roads and fit the road geometries perfectly. However, these paths may not prioritize passenger comfort, where passengers prefer a slightly longer but smoother and more stable path.

To address the issue, many studies have been conducted to smooth planned paths. A post-processing approach which is based on the Pythagorean-Hodograph cubic curve has been proposed to smooth the path generated from a hybrid A* search algorithm [1]. A hybrid A* based motion planning method is also proposed to improve a hybrid A* search algorithm with nonlinear optimization and Catmull-Rom interpolation on post-processing the path [7]. In our paper, we also aim to optimize the time efficiency and the passenger comfort through a functional optimization approach. We explore a path planning algorithm that integrates the hybrid A* path planner [2] and the Frenet Frame trajectory generator [8]. By considering passenger comfort in the path planning process, we can manage the trade-off between travel efficiency and passenger comfort and achieve a good balance between them. The experimental results demonstrate that the proposed algorithm better trades off travel efficiency and passenger comfort, compared with the pure Frenet Frame trajectory generator. The results also provide an insight that input preprocessing, even if it is a simple one, can affect Frenet Frame trajectory generator significantly, and it is worth future exploration.

The rest of the paper is organized as follows. Section 2 defines the problem. Section 3 describes the proposed algorithm. Section 4 shows the experimental results. Section 5 concludes this paper.

2 Problem Definition

Given a scenario which includes the starting point of a vehicle, the destination of the vehicle, and the road structure and boundary, the path planning problem is to compute a path from the starting point to the destination for the vehicle and minimize two objectives:

- The *time cost* is defined as the total time for the vehicle to move from the starting point to the destination.
- The *comfort cost* is defined as the average jerk, *i.e.*, derivative of acceleration, of each time step.

Although we do not consider safety in this paper, it can be modeled as constraints like road boundaries. Given a scenario, an algorithm outperforms another algorithm only if its time and comfort costs are both smaller than those of the other one. Some example scenarios and path planning results are shown in Fig. 1.

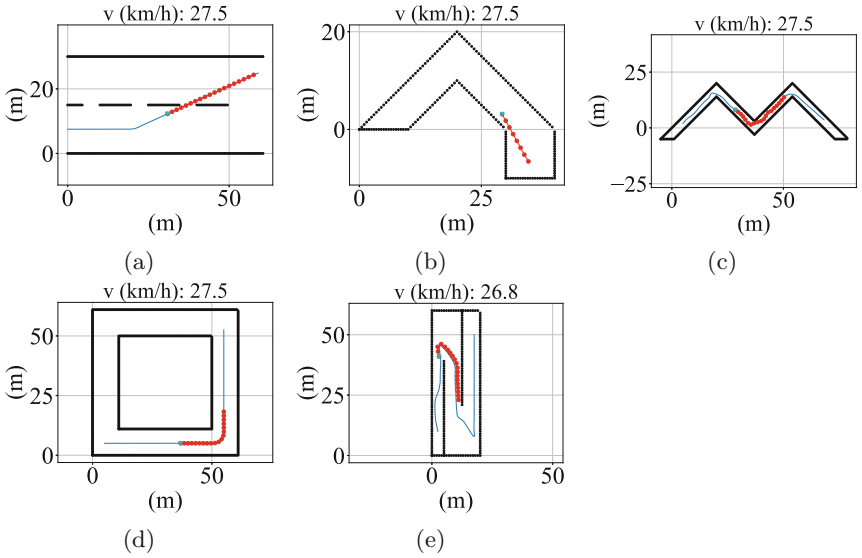

Fig. 1. Example scenarios and path planning results, where the blue solid lines are the paths computed by the hybrid A* path planner, and the red dotted lines are the paths outputted by the proposed algorithm at the blue triangles. (Color figure online)

Algorithm 1: The Proposed Algorithm

Input: starting point, destination, road structure;
Output: planned path P;
1 P' = Hybrid-A*-Path-Planning(starting point, destination, road structure);
2 P = Frenet-Frame-Trajectory-Genration(P');

3 Algorithm

The overview of algorithm is listed in Algorithm 1. The main idea is to use the hybrid A* path planner to generate a path. Then, the algorithm uses the generated path as the central line for the Frenet Frame trajectory generator to follow. The two steps are introduced in the following sections.

3.1 Hybrid A* Path Planning

Dolgov *et al.* introduced the hybrid A* algorithm [2], an extension of the traditional A* algorithm. It is designed to take into account the non-holonomic nature of vehicles. It introduces a 3D state space of the vehicle $< x, y, \theta >$ and a 4D search space $< x, y, \theta, r >$, where x and y represents the position of the vehicle, θ represents the orientation of the vehicle, and r is the current direction of the vehicle. To calculate the cost of path planning, there are two heuristics in forward searching. One is *non-holonomic-without-obstacles* which

Algorithm 2: Hybrid-A*-Path-Planning

```
Input: starting point, destination, road structure;
Output: path computed by the hybrid A* path planner P';
P' = ∅;
Maintain a priority queue Q for all the expanding nodes;
Add the starting point to Q;
while True do
    if Q ≠ ∅ then
        Pop a node n with the minimal cost from Q;
        Remove n from Q and mark n as expanded;
        if n reaches the destination then
            Return P';
        end
        for each unexpanded child m ∈ Analytic-Expansion(n) do
            Update-Cost(m);
            if m ∉ Q then
                Add m to Q;
            end
        end
        Update P';
    end
end
```

can be precomputed since it is independent of real-time sensor data. The other is *holonomic-with-obstacles* which reduces the number of expanded nodes and discovers obstacles well. As for the node expansion, the Reeds-Shepp model is used to make paths smoother and improve search speed. Since the path planning has strict timing requirements, and the hybrid A* path planner is computationally lightweight, we use it to compute the reference line (central line) for the Frenet Frame trajectory generator.

Based on the reference [2], the hybrid A* path planner is listed in Algorithm 2. Given a starting point, a destination, and a road structure, the algorithm maintains a priority queue Q based on the cost of each node. The algorithm applies Analytic-Expansion() to expand nodes either by simulating kinematic models within a short term or by generating an optimal Reeds-Shepp path to the destination. Analytic-Expansion() can improve the planning accuracy and computational efficiency. The cost is updated by Update-Cost(), which considers two heuristics, *non-holonomic-without-obstacles* and *holonomic-with-obstacles*.

3.2 Frenet Frame Trajectory Generation

There are many works on the path planning of autonomous vehicles, but there are relatively less works considering travel efficiency and passenger comfort at the same time. Werling *et al.* utilized the middle of the road as the central line for the Frenet Frame [8], and the goal is to balance travel efficiency and passenger

Algorithm 3: Frenet-Frame-Trajectory-Generation

Input: path computed by the hybrid A* path planner P';
Output: planned path P;
Initialize a state $S = P'$.starting-point
for each step **do**
  Π = Generate-Path-Set(S, P')
  P'' = Select-Minimum-Cost(Π)
  if P'' == P'.destination **then**
    Return P;
  end
  $S = P''$
end

comfort which can be quantified by the jerk, the derivative of the acceleration. To describe the characteristic of a vehicle on the road with the state and the environment, the traditional Cartesian Frame is replaced by Frenet Frame:

$$\boldsymbol{x}(s(t), d(t)) = \boldsymbol{r}(s(t)) + d(s(t)) \cdot \boldsymbol{n}(s(t)), \tag{1}$$

where $\boldsymbol{x}$ is the Cartesian Coordinates, s is the central line of the Frenet Frame, d is the perpendicular offset, $\boldsymbol{r}$ is the current position of the vehicle, and $\boldsymbol{n}$ is the normal vector for the trajectory. The vehicle then generates lateral and longitudinal movements, calculates jerk, and chooses the trajectory with the minimum cost:

$$C = W_T \cdot T + W_J \cdot J + W_H \cdot H, \tag{2}$$

while C is the total cost, T is the time cost, J is the jerk (comfort cost), H is the heuristic cost of next-step selection, and W_T, W_J, and W_H are the constant weights for time cost, jerk cost, and heuristic cost, respectively. It is mentioned that a pre-calculated path can serve as an alternative central line. Based on this insight, we use the path computed by the hybrid A* path planner as the central line for the Frenet Frame. Also, we set $(W_T, W_J, W_H) = (2 - W, W, 1)$ in our setting, where $0.5 \leq W \leq 1.5$.

Based on the reference [8], the Frenet Frame trajectory generator is listed in Algorithm 3. Given the path computed by the hybrid A* path planner, the algorithm regards it as the central line of the road. The algorithm initializes the state of the vehicle at the starting point, including the position, the speed, and the acceleration of the vehicle. Before the vehicle reaches the destination, the algorithm iteratively generates a set (Π) of possible paths along the central line and selects the one (P'') with the minimum cost based on Eq. 2.

4 Experimental Results

We test our algorithm with 5 different scenarios as shown in Fig. 1. We record the time cost (s) and the comfort cost (m/s^3). The implementation is based on

a Python code collection on path planning [5]. In Fig. 1, the blue solid lines are the paths computed by the hybrid A* path planner, and the red dotted lines are the paths outputted by the proposed algorithm at the blue triangles. The proposed algorithm avoids sharp turns and lowers speeds, if needed, to lower comfort cost. For twisted and narrow scenarios (Fig. 1 (c) and (e)), the Frenet Frame trajectory generator modifies the paths more significantly.

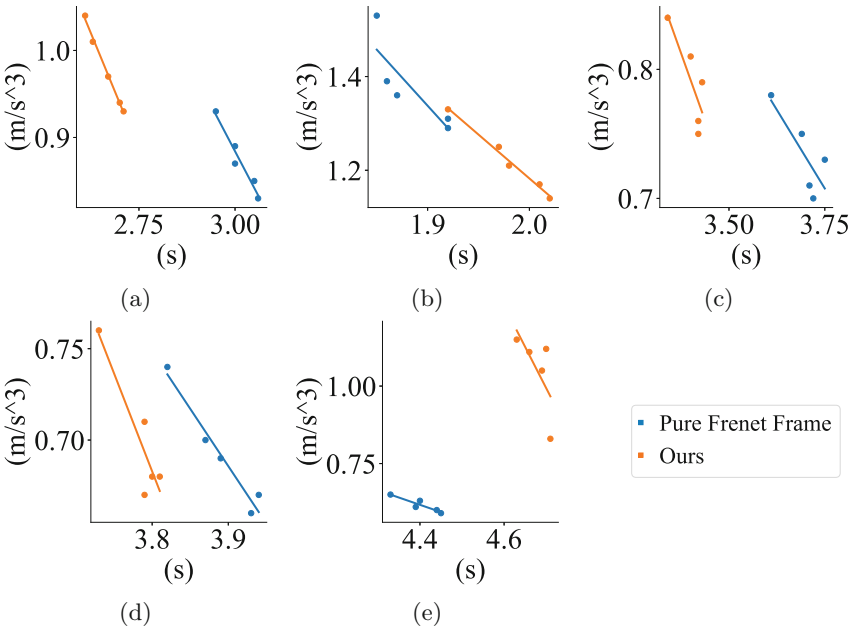

Fig. 2. The experimental results. An x-axis represents the time cost (s), and a y-axis represents the comfort cost (m/s^3). A blue line shows the linear regression result of blue dots, which are the results of the pure Frenet Frame trajectory generator (without the hybrid A* path planner) with the weight $W \in \{0.5, 0.75, 1, 1.25, 1.5\}$. An orange line shows the linear regression result of orange dots, which are the results of the proposed algorithm with the weight $W \in \{0.5, 0.75, 1, 1.25, 1.5\}$. (Color figure online)

We compare the proposed algorithm with the pure Frenet Frame trajectory generator (without the hybrid A* path planner). The experimental results are shown in Fig. 2. An x-axis represents the time cost (s), and a y-axis represents the comfort cost (m/s^3). A blue line shows the linear regression result of blue dots, which are the results of the pure Frenet Frame trajectory generator with the weight $W \in \{0.5, 0.75, 1, 1.25, 1.5\}$. An orange line shows the linear regression result of orange dots, which are the results of the proposed algorithm with the weight $W \in \{0.5, 0.75, 1, 1.25, 1.5\}$. Each dot is the average of 10 runs.

For the three scenarios in Fig. 1 (a), (c), and (d), the proposed algorithm outperforms the pure Frenet Frame trajectory generator. The trends in Fig. 2

(a), (c), and (d) show that, with the same time cost, the proposed algorithm has a lower comfort cost, or, with the same comfort cost, the proposed algorithm has a lower time cost. For the scenario in Fig. 1 (b), the trend in Fig. 2 (b) shows that the proposed algorithm has similar but slightly worse results than the pure Frenet Frame trajectory generator. For the scenario in Fig. 1 (e) which is an extremely special case, the trend in Fig. 2 (e) shows that the proposed algorithm has worse results than the pure Frenet Frame trajectory generator. We infer that more obstacles bring challenges to Analytic-Expansion() in the hybrid A* path planner.

The results indicate that the use of the hybrid A* path planner can improve the objectives of path planning in most cases. Besides, we also observe that the hybrid A* path planner performs better when there are more curves in the scenario. The results also provide an insight that an alternative central line can affect the Frenet Frame trajectory generator significantly, and it is worth exploration.

5 Conclusion

In this paper, we focused on a path planning algorithm that integrates the hybrid A* path planner and the Frenet Frame trajectory generator. We evaluated the performance of the proposed algorithm in terms of travel efficiency and passenger comfort. The experimental results demonstrated that the proposed algorithm better trades off travel efficiency and passenger comfort, compared with the pure Frenet Frame trajectory generator. The results also provided an insight that input preprocessing, even if it is a simple one, can affect Frenet Frame trajectory generator significantly, and it is worth future exploration.

Last but not least, the proposed algorithm provides a smoothing technique, which can improve the robustness or even the security of path planning. For example, if the position of a vehicle is faulty at a certain time, no matter the source is malicious or not, the vehicle may deviate from its original path for a short period. The the proposed algorithm can smooth the path and project the vehicle against the fault. This usage of the proposed algorithm is also worth more exploration.

Acknowledgement. This work is partially supported by Ministry of Education (MOE) in Taiwan under Grant Number NTU-112V2003-1 and National Science and Technology Council (NSTC) in Taiwan under Grant Numbers NSTC-112-2636-E-002-010 and NSTC-112-2221-E-002-168-MY3.

References

1. Chu, K., Kim, J., Jo, K., Sunwoo, M.: Real-time path planning of autonomous vehicles for unstructured road navigation. Int. J. Automot. Technol. **16**, 653–668 (2015)
2. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Path planning for autonomous vehicles in unknown semi-structured environments. Int. J. Robot. Res. **29**(5), 485–501 (2010)
3. González, D., Pérez, J., Milanés, V., Nashashibi, F.: A review of motion planning techniques for automated vehicles. IEEE Trans. Intell. Transp. Syst. **17**(4), 1135–1145 (2015)
4. Kuderer, M., Gulati, S., Burgard, W.: Learning driving styles for autonomous vehicles from demonstration. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2641–2646 (2015)
5. Sakai, A., Ingram, D., Dinius, J., Chawla, K., Raffin, A., Paques, A.: PythonRobotics: a Python code collection of robotics algorithms. arXiv preprint arXiv:1808.10703 (2018)
6. Takahashi, A., Hongo, T., Ninomiya, Y., Sugimoto, G.: Local path planning and motion control for Agv in positioning. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems. The Autonomous Mobile Robots and Its Applications, pp. 392–397. IEEE (1989)
7. Tu, K., Yang, S., Zhang, H., Wang, Z.: Hybrid A* based motion planning for autonomous vehicles in unstructured environment. In: IEEE International Symposium on Circuits and Systems (2019)
8. Werling, M., Ziegler, J., Kammel, S., Thrun, S.: Optimal trajectory generation for dynamic street scenarios in a Frenet Frame. In: IEEE International Conference on Robotics and Automation, pp. 987–993. IEEE (2010)
9. Zhu, M., Wang, Y., Pu, Z., Hu, J., Wang, X., Ke, R.: Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. Transp. Res. Part C Emerg. Technol. **117**, 102662 (2020)