Report for assignment 1: Agent Models and Path Planning in the course DD2380 at KTH

GROUP19

Ilias Lachiri 2001-03-10 lachiri@kth.se

Peiheng Li 2001-03-15 peiheng@kth.se

Abstract

This study addresses the challenge of optimizing autonomous vehicle navigation in simulated environments, a critical area in computer science with wide-ranging applications. We introduce an innovative approach combining the A* algorithm with advanced path smoothing techniques and a tailored Proportional-Derivative controller, significantly enhancing path efficiency and vehicle control. Our contributions include a braking system adaptive to vehicle orientation. This work not only advances autonomous navigation strategies which yields good performances in all the situations but also sets a foundation for future research in dynamic and complex environments.

1 Introduction

The interplay of artificial intelligence (AI) with autonomous navigation has not only piqued the interest of researchers but has also become integral to the advancement of self-guiding vehicles. This project unfolds at the frontier of this evolving domain, investigating the sophisticated coordination required between ground-based automobiles and their aerial counterparts, drones. Our study is predicated upon a robust AI system capable of proficiently navigating an array of maps, each replete with its own set of challenges. These maps, while initially five in number, serve only as a starting point for a system designed to tackle any number of maps, each presenting a microcosm of the real world filled with both natural and constructed obstacles.

What distinguishes our AI system is its agility and adaptability, hallmarks of an advanced navigational assistant that can seamlessly transition between environments, irrespective of the number or nature of barriers. It is an exploration into the essence of dynamic pathfinding, wherein the AI is not merely a tool but an intelligent entity that learns, adapts, and optimizes its path in real-time. Our AI does not simply find a path; it contemplates the optimal path, considering factors such as the shortest route, the least energy-intensive, and the safest for both cars and drones.

The technical prowess of our AI lies in its dual capability. It is adept at controlling cars that must navigate the complexities of ground travel, accounting for obstacles and the physics of terrestrial movement. Simultaneously, it is skilled at managing drones which operate in a multidimensional space, where considerations of altitude and aerial maneuverability are paramount. This AI system does not force uniformity but rather celebrates the unique modalities of each vehicle type, ensuring that each follows a path that is most suited to its design and capabilities.

In crafting this AI, we have amalgamated a suite of AI techniques, from the precision of machine learning models that learn from each journey to the strategic foresight of heuristic algorithms that map out potential futures. This project goes beyond the traditional confines of algorithmic navigation; it is a testament to the symbiotic relationship between advanced computation and intuitive design. By pushing the boundaries of what AI can achieve in path planning, we pave the way for a future where the integration of car and drone mobility is not just viable but is a cornerstone of intelligent transportation systems and urban planning.

1.1 Contribution

Our project makes contribution to AI-based navigation in simulation by integrating several innovative techniques. Firstly, we utilize the map information within Unity to generate a traversable grid-based graph with high resolution, which forms the foundation of our pathfinding framework. We then implement the A* algorithm to find the optimal path through this graph. Due to the physical based vehicle system, we apply a smoothing process using Line-Of-Sight algorithm and interpolation. This refinement allows for more efficient and realistic navigation paths. Additionally, we further enhanced the PD controller for precise movement adjustments. A novel aspect of our approach is the introduction of a braking system that dynamically adjusts based on the vehicle's angle to the path, significantly minimized the tracking error. These contributions collectively advance the performance on the test by providing a comprehensive solution that balances path efficiency with vehicle control in complex terrains.

1.2 Outline

Our report begins with an introduction to the AI's capabilities in diverse path planning situations, followed by an exploration of related work, highlighting the selection and application of the A* algorithm and PD controllers. The proposed method section elaborates on creating a traversable path, path improvement techniques, and the specifics of car and drone PD controllers, addressing implementation challenges. Experimental results are then presented, comparing pathfinding efficiency and heuristic strategies, with a summary concluding the findings and implications.

2 Related work

The pursuit of an optimal path planning algorithm for our autonomous navigation system was a meticulous process, informed by a comparative analysis across various applications. The study titled "Comparative Analysis of Pathfinding Algorithms A*, Dijkstra, and BFS on Maze Runner Game"[6] played a pivotal role in this selection process. Their research illuminated the A* algorithm's ability to adeptly balance computational speed with memory usage. The study revealed that while the Dijkstra algorithm was marginally faster in computation, A* algorithm's reduced memory requirements made it particularly advantageous for large-scale pathfinding problems, like those encountered in autonomous vehicle navigation. This characteristic of the A* algorithm is crucial for our project as it aligns with the need for real-time

path planning and execution within the limited computational resources of our car and drone models.

Upon validating the A* algorithm's efficacy, we explored its application within vehicle tracking systems. The influential work "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing"[3] provided insights into the harmonious function of A* algorithm alongside a Proportional-Derivative (PD) controller. This combination was proven effective in maintaining precise vehicular trajectory on unpredictable off-road terrain, successfully navigating through environmental uncertainties with minimal deviation from the planned path. The research delineates the PD controller's robustness in adhering to the A*-determined trajectories, ensuring accurate steering control.

Inspired by these findings, our approach integrates the A* algorithm to process map data into a comprehensive graph. This graph delineates possible paths through a network of nodes and edges, forming the foundation upon which the most efficient route is ascertained. Subsequently, this optimal route is fed into a PD control system, which then meticulously guides the vehicle along the path. The PD controller ensures the vehicle's adherence to the designated trajectory with an impressive degree of precision, which is paramount for the smooth operation of autonomous systems.

The sophistication of our path planning approach is further enhanced by the study presented in "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot"[7] from Scientific Reports. This research introduced a refined heuristic function within the A* algorithm that incorporates obstacle information directly into the pathfinding process. This enhancement enables the algorithm to navigate through complex environments more effectively, taking into account not only the path efficiency but also critical operational constraints such as energy efficiency and the minimization of overturn risks. These improvements in the heuristic function are particularly pertinent to our project objectives, which emphasize not only on the efficiency but also the safety and reliability of autonomous navigation.

Although the A* algorithm completes the task of searching for an optimized path from the start to the goal, it operates under certain constraints. As detailed in "Theta*: Any-Angle Path Planning on Grids" [2], A* is optimized specifically for grid maps. This limitation means that A* does not consider non-grid paths, leading to a globally sub-optimal solution. Addressing this issue, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles" [4] introduces the concept of a visibility map. This map includes the start and goal vertices and the corners of all obstructed cells, with paths represented as straight lines from the start vertex until an

obstacle is encountered. This visibility map also serves as a heuristic for A* in our post-processing algorithm. Another perspective on the constraints of A* is its reliance on paths with fixed expansion angles. Theta* overcomes this limitation by allowing expansion from a vertex in any direction.

The PD controller is essential for guiding a vehicle along the path to its destination. As "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles" [5] states, the primary challenges with PD controllers are path stabilization and trajectory stabilization. Path stabilization focuses on minimizing path errors, while trajectory stabilization aims to reduce velocity errors. A major issue arises with path discontinuities, which challenge the basic principles of PD control theory. A potential solution is discussed in "Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty" [1], where the authors propose a backstepping control design that moves beyond conventional feedback laws.

3 Proposed method

3.1 Traversable path

To expand on the detailed methodological approach for constructing a navigable graph, essential for autonomous path planning. Initially,we meticulously analyze the environmental map to create a grid-based representation. This initial step involves a thorough examination of each node within the grid to assess its accessibility, categorizing them based on their traversability status—free, partially obstructed, or completely blocked. The crux of our methodology lies in the strategic selection and integration of navigable nodes into the graph, ensuring the inclusion of crucial waypoints such as starting and ending positions to facilitate comprehensive pathfinding capabilities.

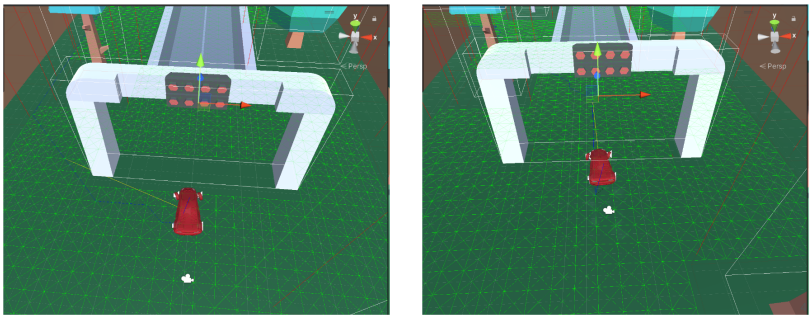

Figure 1: Adding some important nodes specially in the start point

The establishment of inter-node connections is a critical phase, where we ensuring that each edge maintains a safe distance from obstacles, preserving clear navigation pathways. This involves intricate checks for line-of-sight clearances and, in the case of diagonal connections, the verification of intermediate node presence.

After initializing the navigable graph, we use A* to perform efficient pathfinding. The algorithm's core lies in its heuristic function, which estimates the most cost-effective path to the target by evaluating the Euclidean distance between nodes. This process iteratively expands the path from the starting node towards the goal, considering each node's total cost - the sum of the actual distance traveled and the estimated distance to the goal. The final path is then constructed by tracing back from the goal node to the starting node, ensuring an optimal route that avoids obstacles and minimizes travel distance. This method effectively combines the theoretical efficiency of A* with practical navigation needs, ensuring autonomous agents can navigate complex environments with precision.

3.2 Improving the path

The path drawn from A* is executable, however, glitches exist for driving and it is hard for the car and drone to follow the exact path. The major problem is that A* algorithm restricts the path to grid steps, meaning that from a given vertex, the next step can only be to one of its eight neighboring vertices. This approach often results in redundant and complex paths. To improve this, we integrate the Line-Of-Sight algorithm, which draws inspiration from the visibility graph described in [4]. This enhancement to A* makes pathfinding more efficient.

The Line-Of-Sight algorithm begins at a fixed starting point and progresses through successive vertices. At each step, it connects the starting point to the current vertex and checks if this direct line is unobstructed, using vector-boundary overlap verification. If the line is clear, the algorithm proceeds to the next vertex. Otherwise, the current path is recorded as the "post-smoothed" path for that segment of the A* route. The process then restarts from the last post-smoothed vertex added. The result of Line-Of-Sight comparing with A* is shown in Figure 2.a.

The Line-Of-Sight algorithm successfully optimizes the path from the starting point to the goal, but it introduces a new challenge: the resultant path contains too few nodes, as shown in Figure2.b, leading to limited information for navigation. To address this, interpolation is employed. Interpolation involves adding additional nodes along the straight paths smoothed by Line-Of-Sight, thereby enriching the path data for better vehicle tracking

and adherence, as in Figure2.c.

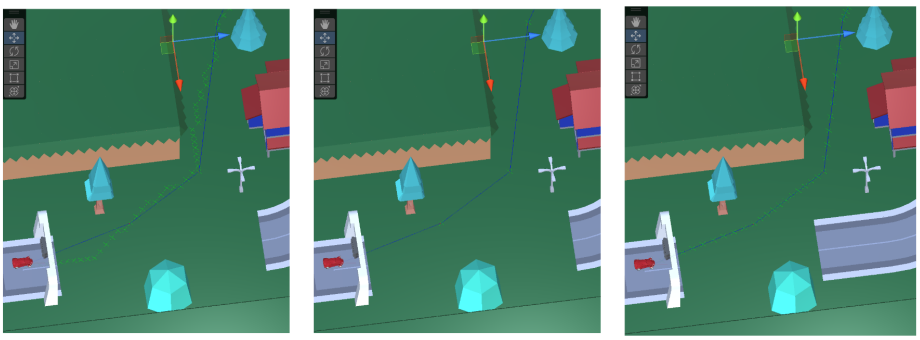

Figure 2: a.Post-Smooth A*; b.Line-Of-Sight vertices; c.Interpolation

Moreover, the PD controllers used for the vehicles, including both cars and drones, take into account position and velocity errors. It's crucial that these two types of errors are comparable in scale to ensure the controller parameters are effective and meaningful. This similarity in scale also guides the resolution of interpolation. We base the interpolation resolution on the length of the post-smoothed path, ensuring that the position and velocity errors remain within a similar range, thus optimizing controller responsiveness and accuracy.

3.3 Car PD Controller

To delve deeper into the PD controller's application in our autonomous car system, we harness its capabilities to intelligently navigate through predetermined paths with precision. By analyzing the upcoming path nodes, specifically the next three, the car calculates the angles between them. This angle calculation is pivotal; a non-180-degree angle indicates a turn, prompting the PD controller to adjust the car's speed and braking to match the turn's sharpness. This dynamic adjustment ensures that the vehicle can handle both sharp and gentle turns efficiently without veering off course.

In straight segments, the objective shifts towards maximizing speed while maintaining path adherence. Here, the PD controller's role expands to fine-tuning the vehicle's steering and acceleration, employing adaptive values for k_p (proportional gain) and k_d (derivative gain) to optimize performance. These adaptive gains are crucial for adjusting the car's response to varying path dynamics, enhancing its ability to stay on course at higher speeds or during abrupt direction changes.

Furthermore, the integration of braking dynamics into the PD controller framework illustrates a nuanced approach to vehicle control. Depending on the vehicle's velocity and the impending turn's angle, braking intensity is modulated to decelerate the car appropriately before a turn. This preemptive deceleration, coupled with strategic steering adjustments, allows for smooth cornering and minimizes the risk of overshooting the path. This detailed implementation of the PD controller, from angle-based steering adjustments to adaptive gain tuning and braking modulation, exemplifies a comprehensive strategy for autonomous vehicle navigation in diverse driving conditions.

3.4 Drone PD Controller

The PD controller for the drone shares similarities with that of the car, but with notable distinctions due to the drone's dynamic point model. The drone's motion is governed by controlling its horizontal and vertical accelerations (along the x-axis and z-axis, respectively), eliminating the need for steering and reversing mechanisms typical in a car. These accelerations are derived from the desired acceleration, calculated based on both position and velocity errors.

Position error is straightforwardly determined by the vector difference between the drone's current position and the target's position. Velocity error, however, is derived from the discrepancy between the drone's current velocity and the target's 'pseudo' velocity. This pseudo velocity represents not the actual movement of the target but the desired velocity of the drone at the target point. For instance, on a straight path, the drone is expected to achieve maximum speed.

Challenges arise at turning points where the possibility of assigning multiple velocities at or near the same point can cause abrupt changes, potentially leading to unattainably high desired accelerations due to the drone's limited deceleration capability. This could result in the drone failing to adhere to the planned path. To mitigate this, the target velocity at turning points is set to a lower value, allowing the PD controller to calculate appropriate accelerations for the drone to slow down smoothly. After the turn, the drone resumes normal operation.

Additionally, to enhance control, the drone's system includes a mechanism to detect turning angles. When a turn is detected, the PD controller places greater emphasis on target velocity to ensure a smooth turn. Conversely, in straight paths, the focus shifts more towards minimizing position error, allowing for quick and accurate responses to any path deviations.

4 Experimental results

Following the development of our navigation method, we proceed to evaluate its performance against other group solutions. This comparison will highlight the efficiency and effectiveness of our approach in real-world scenarios.

4.1 Experimental setup

For the experimental setup in our study, we utilized Unity as the primary environment, with all pathfinding algorithms and navigation logic implemented in C#. The project code, hosted on GitHub, was configured with a grid size of 0.1 units for x, y, and z dimensions to optimize pathfinding results. We conducted tests across three distinct maps (labeled terrainA, terrainB, and terrainC) to evaluate the algorithm's performance. The assessment was twofold: visually inspecting the path to ensure it was logically sound and obstacle-free, followed by measuring the time it took for the autonomous car to travel from start to finish. This approach aimed to identify the fastest trajectory on each map, thereby ensuring replicability and comprehensiveness of our experimental methodology.

4.2 Analysis of Outcome

4.2.1 Pathfinding Efficiency: A* vs BFS

Comparing the A* algorithm and Breadth-First Search (BFS) for pathfinding, the initial choice to implement BFS was driven by its simplicity and the team's strategic focus on developing a perfect traversable path and optimizing the PD controller.BFS, known for its straightforward implementation and a solid foundation , allowed the researchers to allocate more time and resources to enhancing the vehicle's ability to navigate complex environments efficiently and with stability.

However, as the experiments progressed, it became evident that the A* algorithm offered superior performance in terms of efficiency and time-saving, particularly in environments more complex than the relatively simple Map C. Unlike BFS, which explores all possible paths in a breadthward motion without considering the goal until it is found, A* employs a heuristic to estimate the cost of the cheapest path from the current point to the goal. This heuristic approach enables A* to prioritize paths that are more likely to lead to the goal, significantly reducing the Car/Drone execution time required to achieve the goal. A* provides a more efficient and time-effective solution for navigating through more challenging terrains, highlighting the critical role

of algorithm selection in the development of autonomous navigation systems.

4.2.2 Comparison Between A* Heuristics

After implementing A*, the study delves into the performance of the A* algorithm using different heuristic approaches for pathfinding, specifically contrasting a basic heuristic based on Euclidean distance with an innovative heuristic used by another team (Group 4). The fundamental distinction between these approaches lies in their prioritization criteria: while the basic heuristic focuses on minimizing the Euclidean distance from the current position to the goal, Group 4's heuristic emphasizes the reduction of turns along the path, favoring routes that allow for more extended straight-line travel, irrespective of the total distance covered.

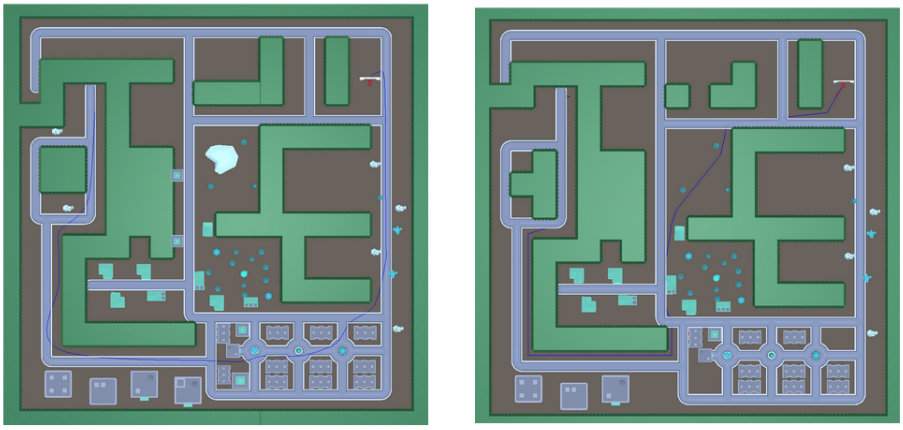

Figure 3: Map A : a.Group4's path; b.Group19's path

This strategic preference for straighter paths over shorter distances stems from the specific dynamics of car navigation, where the ability to maintain higher speeds on straight segments significantly reduces overall travel time. This is particularly advantageous in automotive applications, where the cost of deceleration and acceleration due to frequent turns can outweigh the benefits of a shorter path. Consequently, Group 4's heuristic yielded superior results in car navigation scenarios, as it facilitated a path that, although not the shortest in terms of distance, resulted in faster completion times due to fewer required turns and braking events.

	Car				
Terrian	A	D	E	B	C
Group 4	28.23	26.55	35.08	36.99	10.61
Group 19	32.5	32.63	33.82	36.11	9.1

Table 1: Car results for different A* heuristic

The impact of these heuristic choices was especially notable on Map A, D and E based on Table 1, where the shortest path is replete with turns, presenting a suboptimal route for a car aiming for speed.

However, the same heuristic that benefits car navigation does not necessarily translate to improved performance for drones. Given that drones are not subject to the same speed constraints and can navigate turns without significant deceleration, the advantage of minimizing turns is less pronounced. Hence, the choice of heuristic must be tailored to the specific dynamics and physical constraints of the vehicle in question. This is presented in Table 2.

	Drone				
Terrian	A	D	E	B	C
Group 21	41.1	40.62	46.36	56.27	12.12
Group 19	38.84	39.8	43.19	52.07	12.37

Table 2: Drone results for different A* heuristic

4.2.3 Comparison Between A* and RRT*

Our path planning primarily relies on the A* algorithm, a grid-based search method. In this approach, we discretize the space, which may bypass certain locations that could represent the true shortest path. In contrast, we are interested in comparing our method with RRT*, a sample-based path planning algorithm. RRT* uses heuristics related to the cost of vertices along the path, potentially offering optimized, smoother trajectories in a continuous space. Although RRT* is expected to yield smoother paths, the actual outcomes can vary significantly. This variation is often due to the diverse auxiliary methods implemented by different groups for post-processing, even when using the same fundamental technique.

For our comparative analysis, we will juxtapose our A* results with those of Group 21, who employed RRT* for both car and drone navigation. The comparative data is presented in Table 3.

The data presented in Table 3 reveals interesting insights into the performance of our path planning method compared to that of Group 21. While our approach yielded marginally better results for a car in Terrain E and a

	Car		Drone	
Terrain	D	E	D	E
Group 21	19.24	35.1	40.62	46.36
Group 19	32.63	33.82	39.8	43.19

Table 3: Comparison of Group 21 and Group 19 on terrain D and E

drone in both terrains, Group 21's RRT* method demonstrated a significant advantage for a car in Terrain D. This can be attributed to RRT*'s ability to consider the overall terrain and prioritize straighter paths. In Terrain D, RRT* often finds routes with fewer turns, thereby offering a considerable advantage. On the other hand, in Terrain E, the difference in path planning methods is less pronounced, leading to similar timings. In this case, the performance appears to be more influenced by the controller aspects of the system.

In summary, both efficient path planning and a stable control system are essential for optimal navigation. However, RRT* stands out for its ability to uncover 'hidden' shorter paths that are particularly advantageous for certain types of vehicles. This characteristic allows RRT* to potentially excel in scenarios where non-obvious, more efficient routes exist.

5 Summary and Conclusions

In our report, we have delved into the intricacies of autonomous vehicle navigation within simulated environments, implementing a refined A* algorithm complemented by advanced path smoothing techniques and a specifically designed PD controller with an innovative addition of an adaptive braking system based on vehicle orientation. We envisage that our contributions could spearhead further research in diverse settings, potentially forming a solid framework for the future projects. Future explorations might include adapting our methodologies to more complex environments or integrating our findings into multi-agent applications, thereby broadening the applicability of our research. This project offers a platform for speculative discussion on the AI path planning technologies, encouraging a broader contemplation of their potential advancements and applications.

References

[1] A. Pedro Aguiar and JoÃo P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[2] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39:533–579, October 2010.

[3] Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. *IEEE Xplore*, 2007.

[4] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, oct 1979.

[5] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[6] Silvester Dian Handy Permana and Ketut Bayu Yogha Bintoro. Comparative analysis of pathfinding algorithms a*, dijkstra, and bfs on maze runner game. *IJISTECH (International Journal Of Information System & Technology)*, 2018.

[7] Dan Xiang, Hanxi Lin, Jian Ouyang, and Dan Huang. Combined improved a* and greedy algorithm for path planning of multi-objective mobile robot. *Scientific Reports*, 12, 2022.