# Multi-lane Motion Planning in Autonomous Driving via Laminated A-star Algorithm

DNTC

## Abstract

In order to drive safely and smoothly on roads, autonomous vehicles should make the motion plan in a dynamic environment. On a multi-lane road, both longitudinal and lateral motions need to be decided simultaneously in the time domain. In this work, each lane is projected into an s-t map. A laminated A-star algorithm is proposed on a set of s-t map layers corresponding to all lanes. The algorithm can take options including cruise, overtake and lane change in different scenarios. The trajectory is generated along the time axis to achieve the balance between safety, speediness, comfort and smoothness.

## Keyword

Multi-lane, Motion planning, Laminated A-star algorithm.

## Introduction

Autonomous driving is one of the most focused topics in both artificial intelligence and automobile fields, attracting tremendous attention from researchers. To find the reasonable behavior in dynamic environment, which is called motion planning, is the most challenging problem. The motion planning works as the autonomous brain and builds up a bridge between the environment perceptron and driving behavior control.

The driving control of an autonomous vehicle is a combination of lateral control and longitudinal control, corresponding to the steering wheel and the acceleration/brake pedal, respectively. In most previous works, the lateral and longitudinal motions are decided separately, known as the path planning and velocity controlling [[[1]](#endnote-1)]. First, the ego vehicle generates a trajectory based on the environment with MPC [[[2]](#endnote-2)], artificial potential fields [[[3]](#endnote-3)], virtual force fields [[[4]](#endnote-4)], and so on. Classical Dijkstra or A-star searching algorithms are also popularly used in finding the shortest path across a static map. Then, velocity planning methods such as collision cone are employed to avoid obstacle on the path [[[5]](#endnote-5)].

However, in a real-time driving environment, the mobile obstacles including other vehicles, bicycles, and pedestrians are possibly moving all the time. If the ego vehicle arrives to a path node at a different time as the velocity changes, the environment obstacle distribution may no longer remain the same. It demands that the trajectory planning and velocity planning must be coupled. Some literatures provided end-to-end solutions to calculate the longitudinal and lateral controlling signals directly from the raw sensor information using CNN (convolutional neural network) based deep learning [[[6]](#endnote-6),[[7]](#endnote-7)]. It relies on well labeled training dataset and strong calculation processors. Another drawback of end-to-end methods is that it appears as a black box to both developers and customers [[[8]](#endnote-8)]. The behavior is hard to estimate or coach. The uncertainty limits the reliability in autonomous driving fields.

As discussed before, the path and velocity need to be considered simultaneously to form an entire motion planning. In another word, the planning system should calculate the trajectory along the time axis. In this work, a simple motion planning method is promoted using heuristic search algorithm.

## Related work

### Map define

1. Coordinate conversion

The map is converted from absolute world coordinates into a curvilinear coordinate, so as to be regularized along the lane center line ignoring the curvature change of lane curves [[[9]](#endnote-9)]. The map is then simplified as a rectangular Cartesian coordinate, as indicated in Figure 1. As a result, the longitudinal and lateral controls are mutual independent. To avoid misunderstanding, the regularized coordinates are defined as shift and offset.

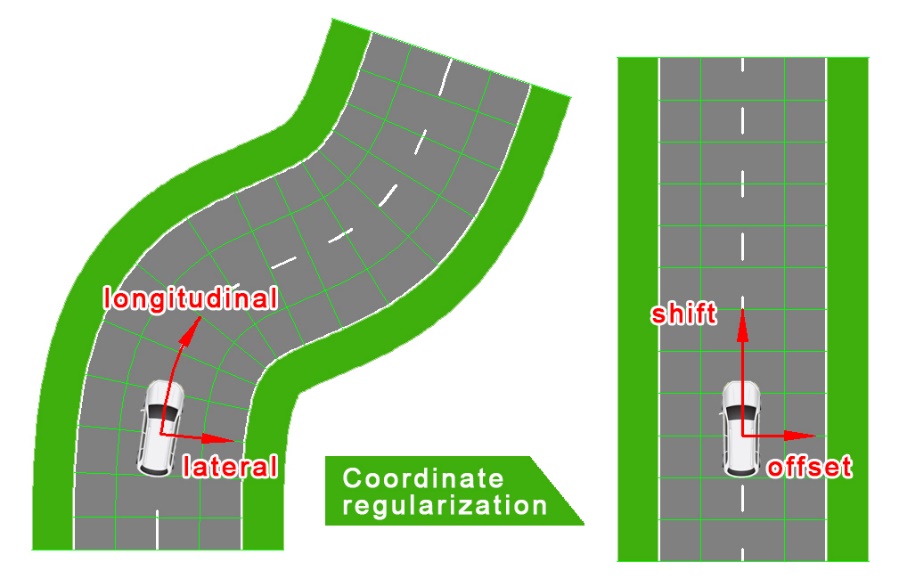


Figure 1 Illustration of coordinate conversion.

In the left, the coordinate of each position on the road is marked in longitudinal and lateral directions, which are along and vertical to the center line, respectively. Then the coordinate is regularized in a rectangular Cartesian coordinate, as shown in the right.

1. Grid maps

To make the coordinate discretized, we divided the map into grids. The surrounding environment has finite number of elements [[[10]](#endnote-10)]. Each object would occupy some lattices according to its position and size.

### S-t mapping

A typical grid map has only two dimensions, indicating the spatial state at a fixed time node. It is widely used in static path planning. For a continuous time sequence, there should be a cluster of grid maps corresponding to all time nodes, respectively. Notice that during driving in a structured road, the trajectory is constrained in lanes. The main available movement is in the longitudinal direction in each lane. The longitudinal shift distance in a single lane can be mapped versus time to compose an s-t map. Hence we can get the distribution of path node in time domain.

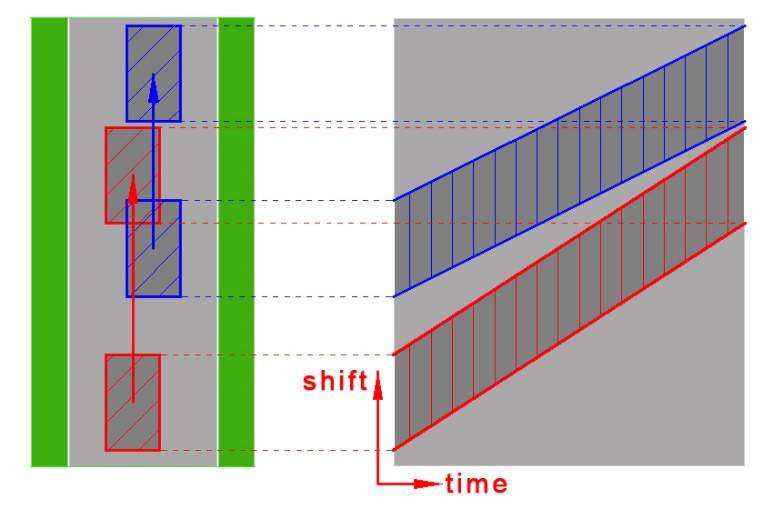


Figure 2 An example of s-t map in a single lane. The abscissa axis is defined as time, while the vertical axis is longitudinal shift in the lane.

The motion of a moving vehicle can be projected in an s-t map. A vehicle occupies a band which can indicate the position at each time. The width of the band is corresponding to the vehicle length, while the slope represents the vehicle velocity.

### A-star algorithm

It is generally to make use of path search algorithm to find an optimized way on a two dimension map. In comparison of Dijkstra method which traverses almost all the free nodes, A-star algorithm is much more computationally efficient and fast due to a heuristic guidance [[[11]](#endnote-11)]. The search domains of the two algorithms are shown in Figure 3.

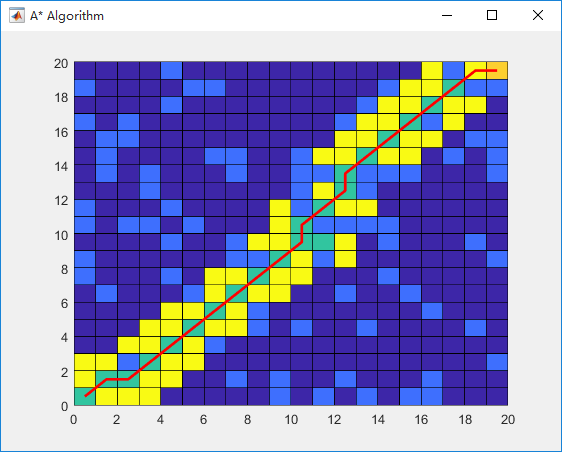
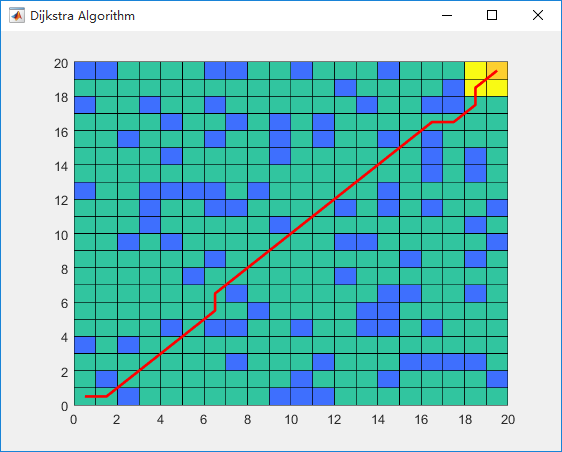


Figure 3 A sample for Dijkstra and A-star searching algorithm in path generation.

Color indicators: dark blue for free nodes; light blue for obstacles; green node for visited nodes; and yellow for nodes to be considered. The path starts from point (0,0) and ends at (20,20).

The Dijkstra algorithm traversed almost all the free nodes to find the shortest path across a random map. The A-star algorithm expands the path along a specific direction leading to the end node.

## Methodology

In this work, we translate the trajectory generation mission into a path planning task on s-t maps. The structured road in divided by lanes. For each lane, there is an s-t map composed of a longitudinal shift axis and a time axis. Multi-lane scenario can be remapped in laminated s-t map architecture. On each s-t map, the A-star algorithm is employed to find the optimized path, which presents the longitudinal motion in time domain. The lateral movement can be achieved by the migration between adjacent maps, known as the lane change.

## Implementation

### Problem statement

The input of the motion planning mission is an occupancy map of the surrounding environment, indicating the position and velocity of the ego vehicle and other vehicles. The perceptron range is 400 meters forward and 50 meters backward in all lanes, as shown in Figure 4 (a). The planning time limit is 20 seconds.

To make the motion planning problem resolvable, several reasonable hypotheses is put forward on the environment scenario. Firstly, only vehicles on the roads are under consideration, while the bicycles and pedestrians are omitted. Secondly, the velocity of surrounding vehicles are supposed to be constant and without lane change. The motion planning process is executed repeatedly during driving, so the motion could be updated in time once the other vehicles’ trajectories take changes.

### Multi-lane in laminated s-t map

As mentioned before, each lane is projected into an s-t map. At one time node, the surrounding vehicle occupies several nodes as an obstacle. Occupied grids are labelled in yellow, as shown in Figure 4(b). As the time passes, the occupied grids move along the longitudinal direction. The gradient is responding to the vehicle velocity. The rest grids in blue indicate the free nodes that the ego vehicle could take. In this sample, obstacle vehicles in three lanes are set with different start positions and velocities.

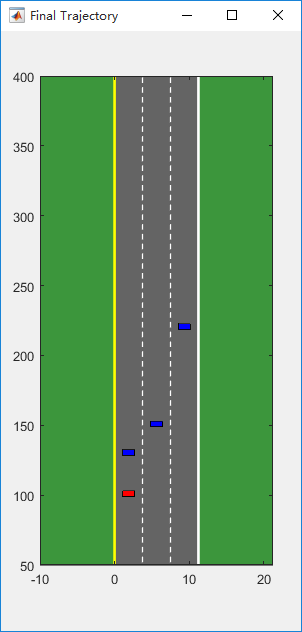
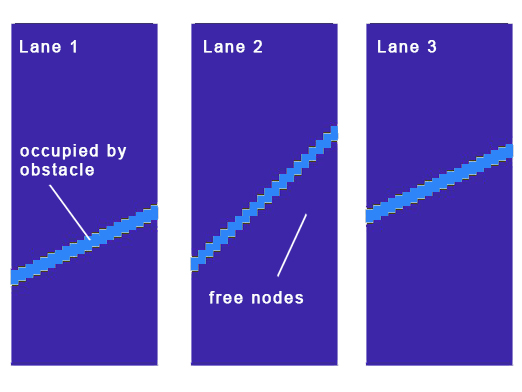
 

Figure 4 Multi-lane in laminated s-t maps. (a) A sample scenario of a three-lane road with one vehicle (blue) in each lane. The ego vehicle is labelled in red. (b) Three s-t maps for all lanes, respectively. The nodes occupied by vehicles are indicated in light blue, while the free nodes in dark blue.

### A-star algorithm

For an A-star algorithm, there are three most important configurations such as the searching set, the G-cost, and the H-cost.

1. Search set

At each time step, A-star algorithm searches a possible node set from the current position. In each s-t map, the possible positions for the ego vehicle to arrive at the next time step are limited by the current velocity, acceleration limitation and velocity limitation. In multi-lane roads, possible positions in adjacent lanes should also be visited. In this sample, reverse driving is forbidden.

1. G-cost function

The A-star algorithm uses a recorded G-cost and a heuristic H-cost in path planning. The G-cost presents the actual cost to reach a position from the start point. In this work, the G-cost is composed of time cost, acceleration cost, and lane change cost. If the grid is occupied by an obstacle vehicle, there should be a to punish the possible crack. A slight parasitical cost is attached in the left overtaking lane and the right slow lane so as to lead the ego vehicle to the center lane preferentially.

The motion planning mission is to find an optimized motion to the end point with the minimum total cost with dynamic limitation, achieving the balance between safety, speediness, comfort, and smoothness.

1. H-cost function

The H-cost is the key component of A-star algorithm. It estimates a future cost from a possible point to the end. The point with lower sum of G-cost and H-cost would be prior visited. It makes the A-star algorithm a heuristic searching method with high computational efficiency.

It should be distinguished that the records the total spent time to get the current position, while the estimates the remaining time to get the target considering the obstacle vehicle in the lane.

### Pseudo codes

The pseudo codes are given to explain the main concept of A-star algorithm.

main()

initialization

g\_cost[all] = 0;

f\_cost[all] = Inf;

g\_cost[obstacle] = Inf;

OPEN = [];

CLOSE = [];

set start to OPEN;

while OPEN != empty

find min cost(cpoint) in OPEN

move cpoint from OPEN to CLOSE

for each npoint in search\_set(cpoint)

temp\_g = g\_cost(cpoint);

temp\_f = g\_cost(cpoint) + h\_cost(npoint);

if npoint == goal

predecessor(npoint) = cpoint;

reverse PATH from npoint to start;

returen PATH;

elseif npoint in CLOSE

continue;

elseif npoint in OPEN

update predecessor(npoint) with min g\_cost;

else set npoint to OPEN;

end if

end for

end while

end

## Results

Figure 5 exhibits the calculation result of the motion planning on s-t maps of three lanes. The light blue grids indicate the environment vehicles in each lane, while the yellow and green grids present the open and close set in the A-star algorithm, respectively. The red circle marks the ego vehicle positions at different time nodes. The ego vehicle started from lane\_1, and then changed to lane\_2 to make an overtaking. After that, it returned to lane\_1 to pass away the obstacle vehicle in lane\_2. At last, the ego vehicle was encouraged to drive at the center lane until it got to the goal distance. Lane\_3 was not chosen because a slow obstacle vehicle was detected ahead. The final trajectory is given in Figure 6.

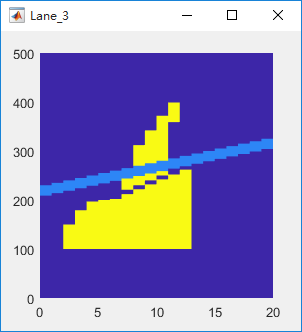
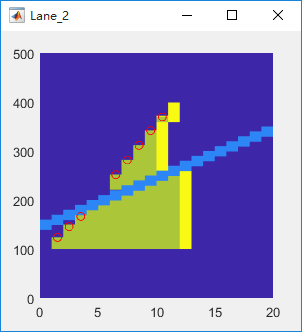
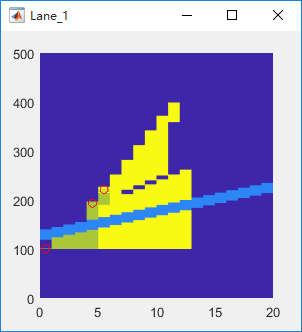


Figure 5 Motion planning on laminated s-t maps.

Color labels: dark blue for free nodes, light blue for occupied nodes, yellow for possible nodes in open list, and green for visited nodes in closed list, respectively.

The chosen position of the ego vehicle at each time step is marked with the red circle.

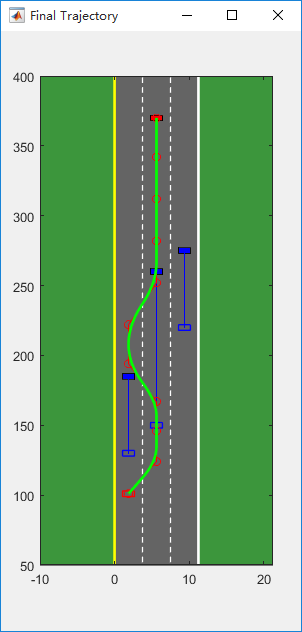


Figure Optimized trajectory in the sample scenario. The blue frames and blocks indicate the start and end position obstacle vehicles, respectively. The ego vehicle is marked in red. The green line presents the final trajectory generated by the laminated A-star algorithm, with red dot for the key position at each time step.

More demonstrations in different driving situations are exhibited in Figure 7. In each test scenario, a safe and smooth trajectory can be generated via the motion planning algorithm.

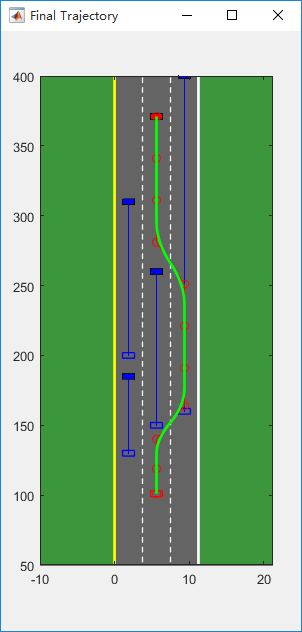
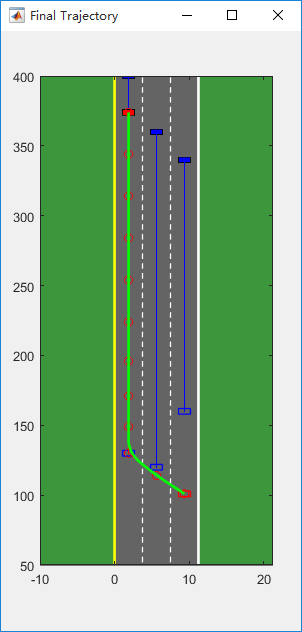
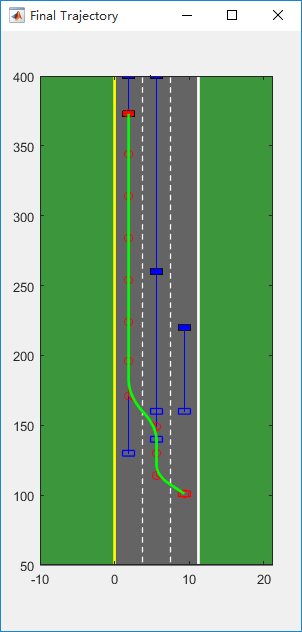
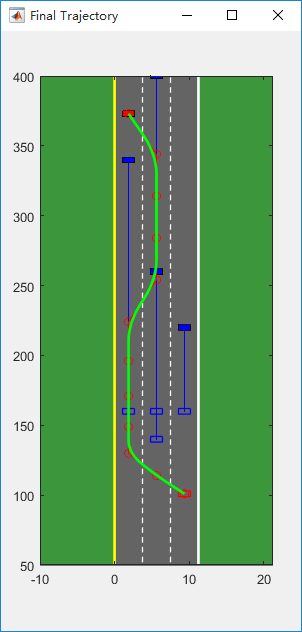
   

Figure 7 More demonstrations in different driving situations.

## Conclusions

The A-star algorithm on laminated s-t maps for multi-lane scenarios is proposed in motion planning. The motions in the time domain are generated simultaneously. The ego vehicle under control can take options including cruise, lane change, velocity change, and overtake according to the surrounding environment with the consideration of safety, speediness, comfort and smoothness. The longitudinal and lateral can be easily calculated based on the optimized motion via vehicle dynamic characteristics.

## Reference

1. [] M. Babu, Y. Oza, A.K. Singh, et al. Model Predictive Control for Autonomous Driving Based on Time Scaled Collision Cone. [↑](#endnote-ref-1)
2. [] M. Nolte, M. Rose, T. Stolte, et al. Model Predivtive Control Based Trajectory Generation for Autonomous Vehicles – An Architectural Approach. [↑](#endnote-ref-2)
3. [] H. Safadi. Local Path Planning Using Virtual Potential Field. [↑](#endnote-ref-3)
4. [] Z. Dong, Z. Chen, R. Zhou, et al. A Hybrid Approach of Virtual Force and A\*; Search Algorithm for UAV Path Replanning. [↑](#endnote-ref-4)
5. [] Z.A. Daniels, L.A. Wright, J.M. Holt, et al. Collision Avoidance of Multiple UAS Using a Collision Cone-Based Cost Function. [↑](#endnote-ref-5)
6. [] M. Bojarski, D.D. Testa, D. Dworakowski, et al. End to End Learning for Self-Driving Cars. [↑](#endnote-ref-6)
7. [] H.M. Eraqi, M.N. Moustafa, J. Honer. End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies. [↑](#endnote-ref-7)
8. [] M. Bojarski, P. Yeres, A. Choromanaska, et al. Explaining How Deep Neural Network Trained with End-to-End Learning Steers a Car. [↑](#endnote-ref-8)
9. [] T.D. Barfoot, C.M. Clark. Motion Planning for Formations of Mobile Robots. Robotics and Autonomous Systems, 2004, 45:65-78. [↑](#endnote-ref-9)
10. [] M. McNaughton. Parallel Algorithms for Real-time Motion Planning. [↑](#endnote-ref-10)
11. [] J.V. Tooren, M. Heni, A. Knoll, et al. Development of an Autonomous Avoidance Algorithm for UAVs in General Airspace. [↑](#endnote-ref-11)