2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)
June 27-28, 2017, Vienna, Austria

Flexible Unit A-star Trajectory Planning for Autonomous Vehicles on Structured Road Maps

Zahra Boroujeni, Daniel Goehring, Fritz Ulbrich, Daniel Neumann, Raul Rojas[1]

***Abstract*— In this paper we propose a trajectory planning approach for autonomous vehicles on structured road maps. Thererfore we are using the well-known A* optimal path planning algorithm. We generate a safe optimal trajectory through a three-dimensional graph, considering the two-dimensional position and time. (1) The graph is generated dynamically with fixed time differences and flexible distances between nodes, based on the vehicle's velocity, using a structured road map. (2) Furthermore the position of dynamic obstacles is predicted over time along the road lanes. The proposed Flexible Unit A* (FU-A*) algorithm was tested for real-time applications with execution times of less than 50 ms on the car's main computer. The feasibility and reliability of FU-A* is validated by implementing on simulated autonomous car of Freie university "MadeInGermany" using the roadmap of Tempelhof, Berlin.**

I. Introduction

Autonomous cars are the emerging future of the automotive industry. In the way of the upcoming ubiquity of automotive industry, two factors are of utmost importance: the safety of the passengers and efficiency (*i.e.*, pollution prevention). Autonomous cars perfectly satisfy the second condition. However, the current technology of commercial batteries for electric vehicles imposes a limited drive time per charge.

And, since the main role of this new technology is to provide comfort to human drivers, a very recent trend in academic and industrial research centers is to prepare the grounds to avoid possible side effects of the driver-less cars. Beyond this, there are strong hopes for the researchers in this field that the new invention will make the current situation of human driving much better in terms of safety and organization, as it happened for the airplanes thirty years ago. As computer programs do not sleep, do not get anxious, do not break the known rules, and do not lose concentration, they will probably cause fewer fatal and non-fatal accidents while driving a car.

The research area of trajectory planning tries to provide solutions for autonomous cars, most important in order to avoid crashing and thus to guarantee the safety of the passengers and to use as few energy as possible. To achieve the main two objectives of autonomous cars, the trajectory planning problem could be formalized to improve the convenience and comfort of driving by avoiding unnecessary braking, and to reduce fuel consumption by finding the shortest path.

Fig. 1. Our test autonomous vehicle "MadeInGermany".

On the other side, nowadays the map of the cities become more and more precise. Topological maps [1] like Route Network Definition File (RNDF) structured map [2], [3] contain roadmap nodes and arcs. Roadmap nodes represent important features, such as entries and exits, and arcs indicate drive lanes and street borders between neighbor nodes. The drive lanes are defined as cubic splines. Although driving in a structured map has restrictions (*e.g.* keep driving within lanes, overtaking only on the left side, e.g., in countries like Germany), finding the optimal trajectory while using constraints will be faster since the search space is limited.

In this paper, we use the well-known A* path planning algorithm while considering time as an extra dimension of the nodes to find an optimal trajectory for an autonomous car in a structured map. The grid unit of the search area changes, depending on the speed of the nodes. Decreasing or increasing the speed makes the grids shorter or longer, in the other words makes grid units flexible. The structured road map in which the autonomous car moves, is not obstacle free. E.g. there exist other cars in the road, we consider them as dynamic obstacles. We propose an approach to predict the position of the obstacles on the structured map, to evaluate which nodes are obstacle free (in the future) during the the FU-A* search algorithm. The rest of the paper is organized as follows: The next subsection briefly reviews the related works. Section II describes the utilization of the A* algorithm to solve the shortest path problem. In Section III the practical issues are highlighted. Then, in Section IV numerical simulation results are provided to show the effectiveness and efficiency of the proposed approach. Finally, concluding remarks are outlined in Section V.

[1] The authors are all affiliated with Dahlem Center for Machine Learning and Robotics, Computer Science Institute, Freie Universität Berlin, Germany {zahra.boroujeni, daniel.goehring, fritz.ulbrich, daniel.neumann, raul.rojas}@fu-berlin.de

The research leading to these results has received funding from the project KLEE (FKZ: 16EMO0159) is funded by the German Federal Ministry of Education and Research within the program IKT 2020. The project executing organization is VDI/VDE-IT.

A. Related Work

To make sure that a trajectory is optimal w.r.t. safety, passenger comfort, time and energy saving constraints and to be able to calculate this trajectory under real-time conditions is a challenging problem. The proposed approaches in this field can be categorized mainly in two classes. On the one hand trajectory planning is defined as an optimization problem and numerical methods like Newton-Raphson algorithm [4], MPC [5], and time elastic band [6] are utilized to solve the problem. The main drawback of this class is that the resulting trajectories are not stable, i.e., for different runs under similar conditions one usually gets different trajectory results. Another class of trajectory planners are represented by search based algorithms which are largely deployed for complicated static environments [7] [8] [9].

In this paper, we focus on the applicability of search based planning algorithms for autonomous vehicles. The kinematics and dynamics of an autonomous car are similiar to non-holonomic wheeled mobile robots, therefore the related work regarding them will be reviewed here. In [9] a variant A* combined with ReedShepp algorithm is used for free environments (unstructured or semi-structured environment) while just considering static obstacles. Re-planning using this algorithm took on similar hardware 300 ms, while in our algorithm - with dynamic obstacles are also considered - the approach took less than 50 ms.
A well-known variant of A*, the so called dynamic A* or D* [10] updates edge costs incrementally instead of recalculating all over again when some of the edges changed. Further, D* computes the plan from goal to start. For large graphs this variant saves a lot of computation time. In our case we do not use this approach and instead create a small graph again from scratch while the car is moving.

Randomized search algorithms, such as RRT, create a path by using random samples from the search space [11]. For an unstructured environment they provide good results which converge to an optimal solution with an increasingly large number of samples. Since we are using a structured environment it would be possible to shrink the search space based on the map, and then to choose samples randomly from there. However, this approach would not have any advantages compared to our grid sampling scheme, since our approach is fast enough. In [12] trajectory planning algorithm is shown that aims to avoid obstacles while following the reference trajectory formulated as a Markov Decision Process. However, the definition of a reference trajectory (local or global) is not specified. And, if we consider the reference trajectory as one of the street lanes, may cause a lot of unnecessary lane changes in case of traffic. For example in order to avoid obstacles, the car changes the lane and then comes back to the previous lane, while the second lane change is unnecessary.

Prediction of dynamic obstacle behaviour is a challenging part of the trajectory planning which has been studied utilizing machine learning techniques [13], and probabilistic models [14] [15]. In [14] a dynamic obstacle is modeled as a box, and for the prediction, it is assumed that the car drives on the road while following the traffic rules. In [15] an obstacle behaviour prediction is modeled as a quintic polynomial based on the deviation of the obstacle's heading from the street center, under the assumption of small road curvature. Differently from these methods, in our approach the target lane of the obstacle is determined based on the minimum distance of vehicle from the lane's center, and then the predicted trajectory is modeled as a cubic polynomial along the road (could be a curvy road) under the assumption of a slow time varying velocity which is the most probable prediction.

II. Trajectory Planning

While in common A* algorithm the environment map is gridded in fix units, in our approach flexible grids are defined in 3 dimensions: x, y, and time. It means the car can plan to the next sequence points (on a 2D manifold) with a flexible distance from previous points in fixed time steps. Table I shows the steps of Flexible Unit A* (FU-A*) algorithm.

TABLE I
THE FLOW CHART OF FLEXIBLE UNITS A* ALGORITHM

FU A* algorithm

g: Cost of reaching node
h: Heuristic function
f: g+h
node(n): x,y,v,parent,f
Input: start(n), goal(n)
Output: path

```
1- if reachAroundGoal(start) = true then return makePath(start)
2- open ← closestPoint(start)
3- closed ← 0
4- while open ≠ 0 do
5-     sort(open)
6-     n ← open.pop()
7-     if reachAroundGoal(n)=true then return makePath(n)
8-     neighbors ← expandFlexibleUnits(n)
9-     for all the neighbors do
10-        if neighbor ∉ Obstacles
11-           neighbor.f ← (n.g + g) + (n.p + p) + h
12-           if neighbor ∩ closed = 0 then open←neighbor
13-       else
14-           closed←neighbor
15-   closed←n
16- return 0
```

In the first step an open and a closed list are created, and the closest point of the structured map is put to the current position of the car as the first node. Then we determine the neighbor points, in the same lane and adjacent lanes with different speed in the next T seconds. We assume that a car could do a lane change in T seconds. We could find a good practical approximation of T for a specified speed range of a car. Dynamic obstacle avoidance for each neighbor points is checked, and free neighbor points are placed into the open list. The rest is placed into the closed list. The cost function for each point is calculated, and the open list is sorted. We will continue the structure the car reaches around

the goal. A goal point for each planning would be N meters ahead of the car on the desired offline path which is given by the structured map. In the sequel, each step of the FU-A* algorithm is clarified.

A. Neighbors

By considering time as a dimension, we define each grid with specified speed and different acceleration actions. As a result, not only the distance between the grids is not fix, but also they have deterministic overlaps and do not have a continuous pattern. In the structured environment, the number of lanes, an their positions are well defined. Therefore, we define at most nine actions which are possible for each grid cell. As shown in Fig. 2, the actions are:

- following the same lane,
- go to the left lane (if existing),
- go to the right lane (if existing),

while

- decelerating,
- continuing with the same speed,
- accelerating.

The destination of the nine actions after the specified time (T) are called children nodes of a parent node.

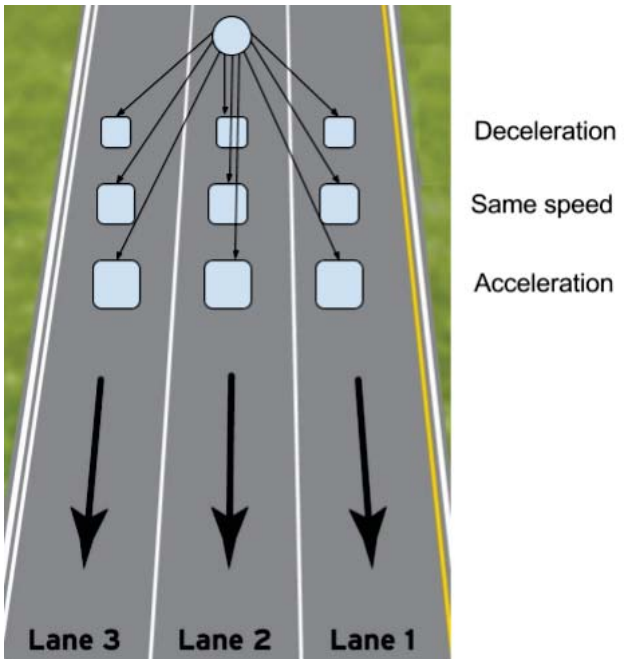

Fig. 2. Graphical representation of the proposed search algorithm: each parent node has nine neighbors, which are defined as decelerating, or continuing with the same speed, or accelerating in the same lane or left/right lane

Although the child node may reach the same positions from different parent nodes, it usually will not have the same time stamp (considering the time stamp that each node is augmented with).

B. Obstacle Position Prediction

The 3D laser scanner and stereo camera provide us sensory data that, by combining them, help us to reliably detect obstacles[1]. The classified obstacle detection system provide us the width and length of the obstacles, as well as their current speed. Imparting the future behaviour of the obstacles based on their type and direction is a very challenging part of the urban driving. Many existing approaches assume dynamic obstacles as a quasi-static or assume that they linearly continue their path along their current heading and with their present velocity. In this paper, we assume that the car will remain in the same lane of the street which may cause a change of the heading. For example if the street is curvy the car would follow the street. Therefore, we have more realistic predictions when structured maps are given. Signals from the car ahead about a lane change could be also considered, however since the replanning time (around 50 ms) is negligible compared to the lane change time (3 to 8 s), this complex prediction seems unnecessary. But, in the intersection area all possible actions (going straight, turning to left or right) are considered.

Each obstacle (the other cars around) is modeled as a band, warped along the street lanes. This is to consider the position uncertainty of an obstacle within a lane. Indeed, the band presents the area which may be occupied by the obstacle. To predict the position of the obstacles over time, the travel distance at time step i is evolved from the current velocity of the obstacle according to

$$d_{\Delta t} = v * \Delta t + w \tag{1}$$

where $d_{\Delta t} \in \mathbb{R}$ is the travel distance calculated in the time step i, $v \in \mathbb{R}$ is the current linear longitudinal velocity of the obstacle, and $w \in N(0, \sigma^2)$ is the process noise, which is assumed to be drawn from a zero mean Gaussian distribution with variance σ^2.

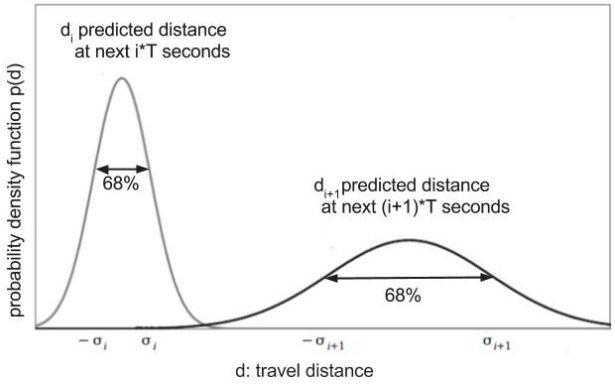

Fig. 3. Travel distance at time sample i vs time sample $i+1$: The horizontal axis represents travel distance of the obstacle center, and the vertical axis represents the probability density function of travel distance.

Fig. 3 compares the prediction of the obstacles travel distance at time sample i with time sample $i+1$. The horizontal axis represents travel distance of the obstacle center, and the vertical axis represents the probability density

[1]Recently, commercial products like Mobileye© and Ibeo© made the classification of the obstacles easier

function of travel distance. The left Gaussian function shows the predicted position distribution of an obstacle at a certain time sample i. By increasing the uncertainty (variance) over time, the distribution of the probability density function becomes wider at the next time sample (the right Gaussian function). One can define the band length b_l as

$$b_l = N\sigma + l_o \tag{2}$$

where N determine the confidence interval, that is for example 1 for 68%, and l_o is the obstacle length.

By finding the closest point to the obstacle on the lane splines (from the map) we can make an assumption in which lane the obstacle is driving. The band position will be calculated along the drive spline of the street for the given predicted travel distance as shown in Fig. 4. In the case of static obstacles the band length is the same as the obstacle length over the time. However, for dynamic obstacles the band becomes longer in each step as the probability density function for the obstacle's position become wider over time.

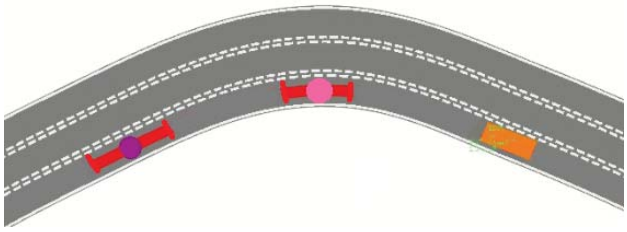

Fig. 4. Schematic representation of the proposed obstacle prediction: orange box is a dynamic obstacle, the red bands show the predicted obstacle position in next T and $2T$ seconds.

C. Obstacle Avoidance

For obstacle avoidance whole way from parent node to the child node should be free of the predicted position of the obstacles. Two different conditions, that together cover all the possible conditions, are checked. The first one is if child and parent nodes are at the same lane, and the second one is if the child is at the adjacent lane of the parent node. At the first one, we consider just the obstacles in the same lane of child and parent nodes. The predicted obstacle band should not be between the parent and the child nodes. In the second condition, not only the obstacles in the child node lane should be checked, but also the obstacles ahead in the parent node lane should be checked. For the obstacles in the child node lane, the predicted obstacle band should not be between the child node and the image of parent node (on the child node lane). We cannot perform a lane change if the obstacle ahead is very near to us. Therefore, for the obstacles ahead and in the parent node lane, the obstacle band should not be in the specified distance V ahead of the parent node, where V is equal to the parent node velocity.

In both conditions, it is obvious that if the child node of a lane with decreasing speed is blocked, the next two children of the same lane (with the same speed and accelerating speed) will be blocked as well, and we must put them in the closed list in our A* planner. Also if the child node with the same speed of a lane is blocked, the next child of the same lane (with accelerating speed) should be blocked and we do not need to check them again. In this way can avoid unnecessary calculations for obstacle avoidance.

TABLE II
THE FLOW CHART OF OBSTACLE AVOIDANCE

if (obstacles in child node lane)
  if ($child\_node \prec obstacle\_band \prec Image\_parent\_node$)
    child_node is invalid.
else if (obstacles in parent node lane)
  if ($parent\_node \prec obstacle\_band \prec parent\_node + Vmeter$)
    child_node is invalid.

D. Cost Function

At each iteration of the FU-A* algorithm, the free nodes in the open list are sorted based on minimizing a cost function which is defined as follows:

$$f(n) = g(n) + p(n) + h(n) \tag{3}$$

The cost function contains three terms. The first term ($g(n)$) is the travel time of reaching a node, which is defined by increasing the step from start point.
The second term ($p(n)$) penalizes hazardous motions, such as going to the adjacent lane which costs k_1. Aborting a lane change maneuver and going back to the previous lane can cause other drivers to be confused and passenger discomfort. Therefore, if in the last trajectory the car decided to do a lane change in the first T seconds of trajectory, changing this decision is penalized by k_2, which means the planner shall not change its decision until a lane change saves more than k_2 seconds to reach the goal. Another discomfort action is unnecessary braking, therefore decreasing speed costs k_3, which means till braking does not provide us more than k_3 seconds time saving, it will not be chosen.
The third term ($h(n)$) is the distance to the goal point which leads to the preferrence of search solutions closer to the goal.

E. Reaching the Goal

The search algorithm must stop when the car reaches the goal point. The goal point is not necessarily an integer multiple of flexible units, therefore if the goal point is between parent and child nodes, the parent node will be chosen as the end node of the graph.
In the case of a blocked street, the search algorithm cannot reach the goal. Therefore, based on the obstacle distance, a "smooth brake" or "emergency brake" maneuver will be chosen as the desired trajectory.

III. PRACTICAL ISSUES

The FU-A* path gives us a sequence of the set points which their distance are $d = VT$, being V the former speed of the car. Thus, the distance is proportional to the speed. The long distance between set points causes two issues:

- the car may not stay on the street lane;
- a big difference between the points of the resulting trajectory results in a large error for control input which causes uncomfortable steering or gas changes.

To deal with these issues, the gaps between the points of the solution trajectory are filled with subsampling points (for every meter) w.r.t. the drive lane spline or a predefined lane changing spline described in the following sub-section. If the parent and child nodes are at the same lane then the drive lane spline is used for the sampling points. Otherwise the lane change spline as described below is sampled.

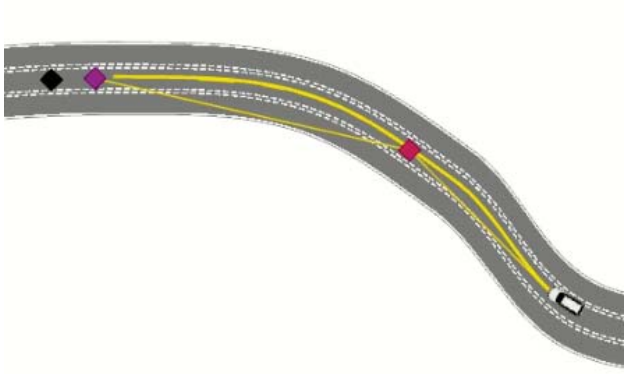

Fig. 5. Resampling along the road: parent and child nodes are at the same lane, therefore we subsample for every meter along the drive spline from parent node toward the child node.

A. *Predefined Lane Changing Spline*

To have a smooth and convenient lane change, a cubic polynomial is defined between parent node and child node in the adjacent lane. The time distance between the parent and child nodes is T seconds. This time should be practically sufficient for a lane change. To find the parameters of a cubic polynomial four assumptions are needed. The first and end points of the spline are equal to the parent and child nodes' position. The first derivative of the start point and end point must be the same as the first derivative of the drive lane splines at the same positions.

In order to avoid set points jumping during a lane change, and to allow the car to follow the same trajectory until it finishes the lane change, it is important not to update the predefined spline during a lane change maneuver until the corresponding child node stays at the same lane. But, if during a lane change the car decides to go back to the previous lane, the new spline between the current position of the car and child node will be defined and sampled.

IV. SIMULATION

A comprehensive simulation study, is performed to validate the proposed algorithm. In this section two common scenarios are simulated to show the safety and efficiency of the proposed algorithm.

The algorithm and simulation was implemented using the ROS framework. The FU-A* trajectory planning for the autonomous car ran at 20 Hz, and a path planner

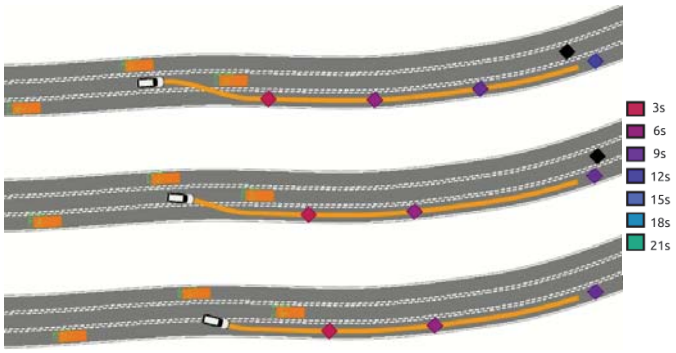

Fig. 6. Resampling along predefined cubic spline: parent and child nodes are at the different lanes, a cubic spline is defined between parent and child nodes.

(following the lane road) for the other cars ran at 100 Hz. The simulated road map is the map of the former Tempelhof airport, Berlin (Fig. 7). The FU-A* parameters used in simulation are described in table III.

TABLE III
FU-A* PARAMETERS USED IN SIMULATION

T (sec.)	k_1 (sec.)	k_2 (sec.)	k_3 (sec.)
3	3	10	20

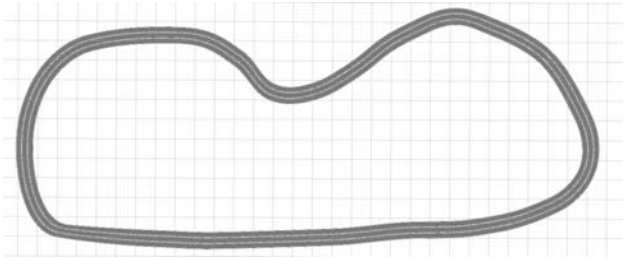

Fig. 7. Map of former Tempelhof airport (Berlin, Germany): grid cell size is 10 meter.

Fig. 8 illustrates a simple lane change maneuver. The sequence points of FU-A* are shown with diamond markers. The predicted obstacle distribution centers are shown with circle markers. The colors of the diamonds and circle markers for each sample time are the same, which are described in the legend. The color changes from pink to green over the time. The black diamond shows the goal point. The color of the lane between the markers shows speed of the action between nodes. The speed color changes from red to green when the velocity changes from 0 to 18 m/s.

In the first test the autonomous car merge to traffic speed as shown in Fig. 9. The autonomous car decreases speed from 10 m/s to 6 m/s to merge into traffic speed and to plan with the traffic speed. The sequence pictures shall illustrate the car position and behaviour overtime with 3 seconds timestamp.

In the second scenario the car ahead breaks instantaneously, therefore the autonomous car decreases its speed and then overtakes from the left side while caring about the car driving on the left lane Fig. 10. The sequence pictures illustrate

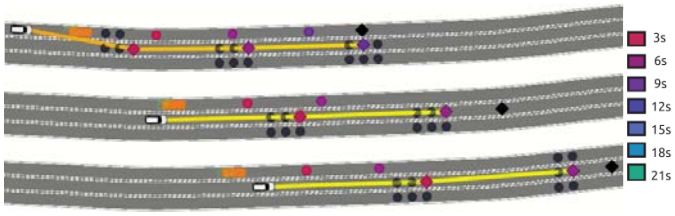

Fig. 8. Simple overtaking: the colors of diamonds and circles show the time sequences. The sequence pictures illustrate the car position and behaviour overtime with 3 seconds timestamp. The car increases the speed from 8m/s to 9m/s and overtakes.

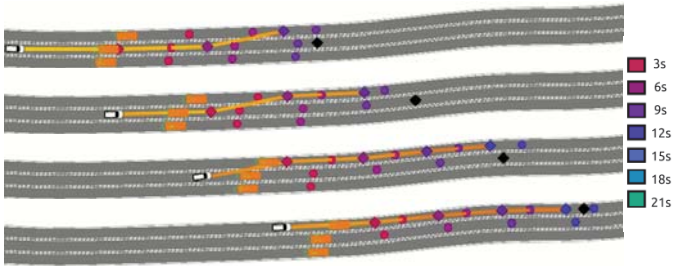

Fig. 9. Merge to traffic: the sequence pictures illustrate the car position and behaviour overtime with 3 seconds timestamp.

the car position and behaviour over time with 1 second timestamp.

Interested readers are encouraged to watch a video of the simulations at `https://youtu.be/Lw_Mk37N6G0`.

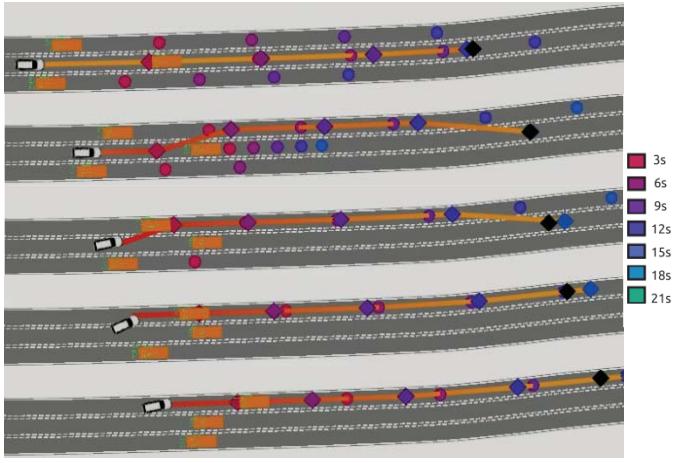

Fig. 10. Decrease the speed and overtake: the sequence pictures illustrate the car position and behaviour overtime with 1 second timestamp.

V. Conclusion

FU-A*, our proposed algorithm, is a new approach for trajectory planning in a structured urban area, considering static and dynamic obstacles. Its output trajectory is locally optimized and feasible. Dynamic obstacles in the road maps are carefully considered by utilizing a predictive approach that takes into account the velocity of the obstacles and the spline of the road. Simulation results, in which we used the simulated autonomous car "MadeInGermany and Tempelhof, Berlin road map, revealed the validity and reliablity of our proposed algorithm. The work is in progress to validate the algorithm using our real platform MadeInGermany in reality and using simulated obstacles, that will be reported in the future work.

Acknowledgment

This research was supported by the Federal Ministry of Education and Research (BMBF) as a part of the Project Kombinierte Logik fr Energieeffiziente Elektromobilitt (KLEE). We thank our colleagues from Autonomous GmbH, IAV, ZF Friedrichshafen AG, TU Dresden, Ibeo Automotive Systems GmbH who provided insight and expertise that greatly assisted the research.

References

[1] D. A. R. P. Agency, "Urban challenge route network definition file (rndf) and mission data file (mdf) formats," 2007.

[2] P. Czerwionka, "A three dimensional map format for autonomous vehicles," Master dissertation, Freie University of Berlin, 2014.

[3] M. Wang, *A Cognitive Navigation Approach for Autonomous Vehicles*. mbv, 2012. [Online]. Available: https://books.google.de/books?id=eO5nmwEACAAJ

[4] A. W. Divelbiss and J. T. Wen, "A path space approach to nonholonomic motion planning in the presence of obstacles," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 443–451, Jun 1997.

[5] M. Jalalmaab, B. Fidan, S. Jeon, and P. Falcone, "Model predictive path planning with time-varying safety constraints for highway autonomous driving," in *Advanced Robotics (ICAR), 2015 International Conference on*, July 2015, pp. 213–217.

[6] C. Rsmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *Control Conference (ECC), 2015 European*, July 2015, pp. 3352–3357.

[7] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *CoRR*, vol. abs/1604.07446, 2016. [Online]. Available: http://arxiv.org/abs/1604.07446

[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[9] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *I. J. Robotics Res.*, vol. 29, no. 5, pp. 485–501, 2010. [Online]. Available: http://dx.doi.org/10.1177/0278364909359210

[10] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, pp. 968–975 vol.1.

[11] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5041–5047.

[12] H. Mouhagir, R. Talj, V. Cherfaoui, F. Guillemard, and F. Aioun, "A markov decision process-based approach for trajectory planning with clothoid tentacles," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 1254–1259.

[13] F. Havlak and M. E. Campbell, "Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments," *CoRR*, vol. abs/1309.0766, 2013. [Online]. Available: http://arxiv.org/abs/1309.0766

[14] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *2008 IEEE Intelligent Vehicles Symposium*, June 2008, pp. 1149–1154.

[15] C. Guo, C. Sentouh, B. Soualmi, J. B. Hau, and J. C. Popieul, "Adaptive vehicle longitudinal trajectory prediction for automated highway driving," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 1279–1284.