The Decision Making of Autonomous Vehicles: A Survey

Wenxing Lan
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China.
12032882@mail.sustech.edu.cn

***Abstract*—**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Autonomous vehicles (also known as self-driving vehicles and driverless vehicles) have been researched by many universities, research institutes, internet companies, vehicle companies and companies of other industries around the world since the middle 1980s [1]. In the last two decades, several crucial of autonomous vehicle research platforms are as follows: the Navlab's mobile platform [2], University of Pavia's and Parma's car, ARGO [3], and UBM's vehicles, VaMoRs and VaMP [4].

In the last decade, the Defence Advanced Research Projects Agency (DARPA) organized three competitions to accelerate the development of correlation technique about autonomous vehicles [5]. In 2004, the first competition named as DARPA Grand Challenge was held in the Mojave Desert, USA [1]. The goal is to get autonomous vehicles to travel 140 miles of off-road routes as fast as possible [5]. Unfortunately, there is no autonomous vehicle which was able to complete the whole journey [1].

In 2005, the DARPA Grand Challenge was held again and required autonomous vehicles to travel 132 miles of difficult desert roads across Nevada which contain a mixture of featureless terrain, dust, global positioning system drop-outs, sharp turns, narrow openings, bridges, railroad overpasses, long tunnels, obstacles and a narrow winding mountains road with a 200-foot drop-off [6]. This competition had 23 finalists and 4 cars finished the course within 10-hour limit. Remarkably, the Stanford University's vehicles, Stanley, won the first-place prize, and the Carnegie Mellon University's cars, Sandstorm and H1ghlander, came in second and third place, respectively [6].

The third competition which is named as the DARPA Urban Challenge was held at the now-closed George Air Force Base, California, USA, in 2007. The objective was for a autonomous vehicles to complete the 60 mile course in less than 6 hours. Additionally, the autonomous vehicles were asked to obey all traffic regulations which avoiding other vehicles including driverless and humandriven vehicles [7]. There are 11 finalists in this competition and 6 vehicles completed the route within the allotted time limit. Specifically, Boss, the vehicle of Carnegie Mellon University, won the first-place prize, the Stanford University's car, Junior, claimed second place, and the Virginia Tech's car, Odin, finished in third [7]. Although the challenges presented by these competitions cannot cover all the challenges encountered in everyday traffic, they have been hailed as milestones in the development of autonomous vehicles [5].

After the DARPA Challenge, there was a flood of driverless events. Relevant examples include: Intelligent Vehicle Future Challenges [8], from 2009 to 2013; Hyundai Autonomous Challenge [9], which was held in 2010; VisLab Intercontinental Autonomous Challenge [10], in 2010; the Grand Cooperative Driving Challenge (GCDC) [11], in 2011 and 2016 and Public Road Urban Driverless-Car Test [12], which was held in 2013. At the same time, many industrial and academic teams have invested a lot of research and development energy in the field of autonomous driving. Among them, the industry is represented by the OEMs of Ford, Toyota, Hyundai and other car companies, as well as IT and emerging companies such as Waymo, Tesla, Uber, Intel, Baidu, Pony.ai, etc., and have developed various autonomous vehicle platforms based on commercialization goals; Academia, including CMU, Stanford, UC Berkeley, Tsinghua University, Tongji University, Southern University of Science and Technology and other major domestic and foreign universities have carried out a series of researches around the key technical fields of autonomous driving.

In order to regulate the application of unmanned driving technology, the National Highway Transportation Safety Administration of the United States Department of Transportation has divided the level of automatic driving (based on the SAE International Standard J3016 [13]):

1) Level 0 (manual driving): completely controlled by a human driver;
2) Level 1 (assisted driving): The driving environment provides support for one of the steering wheel and acceleration and deceleration operations, and the rest is operated by humans. Contains basic auxiliary driving, such as adaptive cruise control (ACC), anti-lock brake system (ABS), electronic stability control system (ESC);
3) Level 2 (semi-automatic driving): The driving environ-

ment provides support for multiple operations in the steering wheel and acceleration and deceleration, and the rest is operated by humans. Contains some advanced auxiliary driving functions, such as a horizontal/vertical control system with minimal risk, emergency braking, etc.;
4) Level 3 (Highly Autonomous Driving): The unmanned driving system completes all operations, but requires the human driver to take over when leaving the operational scene of unmanned driving.
5) Level 4 (Ultra-high auto-driving): An unmanned driving system with limited roads and environmental conditions, and does not require human drivers to respond to system requests.
6) Level 5 (Fully Automated Driving): Unmanned driving system that does not limit roads and environmental conditions

The perception system and the decision-making system are two main parts of the autonomy system architecture of autonomous vehicle [5]. To limit the scope of this survey, We focus on some aspects of decision-making system, which includes route planning, path planning, behavior selection, motion planning, obstacle avoidance and control, in particular, for systems falling into the automation level of 3 and above [1]. The remainder of the paper id structured as follows:

1) Overview of the Decision-Making Hierarchy
2) Route planning
3) Path planning
4) Behavior selector
5) Motion planning
6) Obstacle Avoidance and control
7) Conclusion

II. Overview of the Decision-Making Hierarchy

In this section, the decision making system of a typical autonomous vehicles is described and the brief description of each component of decision making system is provided. The perception system of autonomous vehicles use data captured by on-board sensors, such as Light Detection and Ranging (LIDAR), Radio Detection and Ranging (RADAR), camera, Global Positioning System (GPS), Inertial Measurement Unit (IMU), odometer, etc., and prior information about the models of sensors, road network, traffic rules, vehicle dynamics, etc. to estimate the state of vehicles and create representation of the surrounding environment [1]. Then, the decision making system use the estimation and environment constructed to control the vehicle in order to reach the goal object [5].

There are six subsystems which are shown as the orange blocks in Fig 1 in the decision making system of a typical autonomous vehicles [1]. The brief description of each subsystem is shown as belows.

A. Route Planning Subsystem

Given a requested destination defined in the road network by the user, the route planning subsystem aims to compute a route, R, through the road network from its current position to

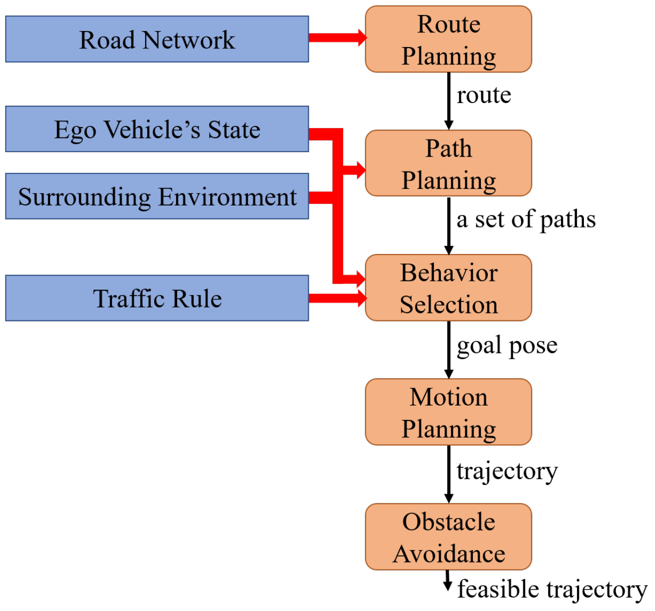

Fig. 1. The five subsystems of decision making system of a typical autonomous vehicle.

the requested destination [5]. A route, $R = \{r_1, r_2, ..., r_{|R|}\}$, is a sequence of way point, where each way point, r_i where $i \in \{1, 2, ..., |R|\}$, is a coordinate pair. i.e. $r_i = (x_i, y_i)$, in the road network [1]. Actually, route palnning is a kind of global route planning since it needs to find the complete route from ego vehicle's current position to the final goal, such as the route planned by the BAIDU maps[1]. The related methods for route planning is shown is Section III.

B. Path Planning Subsystem

Given a route computed by the route planning subsystem, the path planning subsystem computes a set of paths, $P = \{P_1, P_2, ..., P_{|P|}\}$ by considering the ego vehicle's current state and the traffic rules as well as the surrounding environment [1]. A path is a sequence of poses, i.e. $P_j = \{p_1, p_2, ..., p_{|P_j|}\}$ where $j \in \{1, 2, ..., |P|\}$ [1]. Additionally, each pose, p_i, is a coordinate tuple in the static map which only has static object, such as static obstacles, buildings and roads, aroung the ego vehicle and the desired vehicle's orientation at the position defined by this tuple, i.e., $p_i = (x_i, y_i, \theta_i)$ [1]. The relevant methods used in path planning subsystem will be given in Section IV.

C. Behavior Selection Subsystem

The behavior selection subsystem is responsible to select the current reasonable driving behavior for the vehicle according to the behavior, traffic rules and road conditions of the current surrounding traffic participants [5]. The behaviors output here are specific driving maneuvers, such as acceleration and deceleration, lane change, overtaking, and following [5]. Additionally, the behavior selection subsystem need to select a

[1]https://map.baidu.com/

pose in a selecting path among a set of paths generated by path palnning subsystem a few seconds ahead of the current ego vehicle's state [1]. The relevant methods used in Path Behavior Selector is shown in Section V.

D. Motion Planning Subsystem

When the behavior is selected by the behavior selection subsystem (for example, it may be lane changing or left turning), motion planning subsystem needs to find a trajectory T which is form the current ego vehicle's state to the curent goal state [5]. More importantly, the trajectory needs to satisfy the kinematic and dynamic constraints and be confortable for passengers [1]. The trajectory $T = \{c_1, c_2, ..., c_{|T|}\}$ can be defined as a sequence of commands, $c_k = (v_k, \phi_k, \Delta_{t_k})$, where v_k denotes the desired velocity at time k, ϕ_k represents the desired steering angle at time k, and Δ_{t_k} is the duration of c_k [1]. In Section VI, the detail of methods used in motion planning subsystem will be given.

E. Obstacle Avoidance Subsystem

The obstacle avoidance subsytem needs to change the trajectory (usually decreasing the speed) which is the output of motion planning subsystem in order to avoid collision with other obstacle [1]. There is little literature on how to perform the function of this subsystem. Some of the literature related to this subsystem will be discussed in Section VII.

III. Related Work of Route Planning Subsystem

In this section, the related techniques reported in the literature for the Route Planning subsystem will be surveyed.

The road network, which is the input of the route planning subsystem, is usually represented as a weighted directed graph $G = (V, E)$ [1]. As a assumption, the weight of each edge is always a nonnegative value. In weighted directed graph G, each vertex v_i where $v_i \in V = \{v_1, v_2, ..., v_{|V|}\}$ is road junction, such as crossroads, three forks, etc., and each directed edge e_{ij} where $e_{i,j} \in E, i \neq j$ connects pairs of road junctions v_i and v_j. Specifically, edge weight w_{ij} denotes the cost of traversing edge e_{ij}. Computing a route, the task of route planning subsystem, can be reduced to find a path in a weighted directed graph. Normally, the path should be shorest since most users are willing to reach destination as soon as possible [1]. Unfortunately, classical shortest path algorithms, such as Dijkstra [14] and A^* [15] are impractical because of their high time complexity to large road networks, such as city road network or even nation road network [1].

The performance of route planning algorithms in road networks has improved significantly in the past decade [1]. The driving direction can be calculated by some newly developed algorithm in milliseconds or less, even at national scale [16]. Various technologies supply different balance between preprocessing effort, query time and space usage [16]. The queries can be responded in less than a microsecond by some alforihtms, while the real-time traffic can be effectively handled by other algorithm [16]. The techniques used in route planning in a road network can be divided into six classes [16]:

1) Basic Techniques
2) Goal-directed Techniques
3) Separator-based Techniques
4) Hierachical Techniques

A. Basic Techniques

Dijkstra's algorithm provides a standard solution to the one-to-all shortest path problem in a weighted directed graph, and pesudo code of Dijkstra's algorithm is shown in Algorithm 1 [14]. *label-setting*, one of the property of Dijkstra's algorithm, means that the traversing cost from start site v_s to destination site v_e is shortest once the destination site v_e has been scanned [14]. Consequently, the algorithm can be stopped as soon as it has been scanned the destination site for one-to-one queries [16]. The time complexity of Dijkstra's algorithm is influenced by the using priority queue. For example, the time complexity is $O((|V| + |E|)log|V|)$ if the priority queue is binary heaps [17].

Algorithm 1 Dijkstra's Algorithm [14].

```
Input: weighted directed graph $G = (V, E)$; start vertex $v_s \in V$
1: Initial a priority queue $Q = \{v_s\}$ of vertices which is ordered by total traversing cost from $v_s$.
2: Initial all traversing cost $cost(v_s, v_i) = +\infty$ where $v_i \in V$.
3: Set $cost(v_s, v_s) = 0$.
4: while not all vertex $v \in V$ has been traversed do
5:     Exact a vertex $u$ with minimum traversing cost from $Q$.
6:     Find all arcs $E' = \{e|e = (u, v) \in E\}$ incident to $u$.
7:     for each $e = (u, v) \in E'$ do
8:         if $cost(v_s, u) + w(u, v) < cost(v_s, v)$ then
9:             Set $cost(v_s, v) = cost(v_s, u) + w(u, v)$.
10:        end if
11:        Add $u$ into $Q$.
12:    end for
13: end while
14: Return all traversing cost.
```

Bellman-Ford algorithm is an alternative method which can compute shortest path in a weighted directed graph [18], [19]. At each iteration, it scans all vertices whose traversing cost have improve [18], [19]. It often uses a simple FIFO queue to keep track of vertices to scan next [16]. Additionally, Bellman-Ford algorithm has the *label-correcting* property: each vertex may be scanned many times, and the time complexity of this algorithm is $O(|V||E|)$. Finally, the Floyd-Warshall algorithm calculates traversing cost between all pairs of vertices in $O(|V|^3)$ time [20]. It is more suitable for sufficiently dense graphs than Dijkstra's algorithm [16].

In summary, these algorithm metioned above are the basic algorithm to find shortest path in a weighted directed graph. Although they all have high time complexity, they always can find one shortest path in a given nonegative weighted directed graph.

B. Goal-directed Techniques

By avoiding scanning the vertices that are not in the direction of the target vertex, the goal-directed technology guides the search from the source vertex to the target vertex [16].

A^* algorithm, a classic goal-directed shortest path algorithm, uses a heuristic function $f(u,t)$ to estimate the traversing cost from vertex u to t (t is usually the destination site.) [15]. Normally, the value of $f(u,t)$ is the lower bound of the traversing cost from vertex u to t. Then, it runs a modified version of Dijkstra's algorithm which regards the value computed by the formula $cost(v_s,u)+f(u,t)$ as the prioirty of a vertex u [16]. Therefore, the vertices which are much closer to the destiniation site t will be scanned earier during the implementation of the algorithm [16]. ALT (A^*, landmarks, and triangle inequality) algorithm uses inequality property among landmarks to automatically obtain a heuristic function with better lower bounds in order to replace the user-defined heuric function in A^* algorithm [21].

Arc flags, an another goal-directed method, needs to divide the grah into C cells which are roughly balanced (have similar number of vertices) and have a few boundary vertices during preprocessing [22], [23]. A vertoc of C bits, named as arc flags, is maintained by each arc [22], [23]. Additionally, if the arc lies on a shortest path to some vertex of cell i, the i-th bit is set [22], [23]. The arc marking of unit i is calculated by growing the shortest path tree backward from each boundary vertex and setting the i-th flag for all arcs (i.e. edges) of the tree [22], [23]. In the query phase, the algorithm prunes the arcs that are not labeled for the pixels that contain the target vertices [16]. The preprocessing time of arc flags algorithm is high, but the query time of arc flags algorithm is the fastest among goal-directed technology [1].

In summary, these algorithms mentioned above use destination site as information to guide search process. Therefore, the search process is more targeted. Additionally, all of them use heuristic function which is user-defined or obtained by structure of the graph to estimate the cost between experted exploring vertex and destination vertex.

C. Separator-based Techniques

Either vertex or arc is regarded as separator of graph among the methods using separator-based techniques [1]. The vertex (or arc) separator is a small part of a vertex (or arc), and its removal breaks the graph into several balanced components [16]. The algorithm based on vertex separator uses vertex separator to calculate overlay [16]. Quick arcs are added to the overlay to preserve the distance from any pair of vertices in the entire graph. The overlay graph is much smaller than the complete graph and is used to speed up the query algorithm [16]. High performance multi-level routing (HPML) algorithm is a variant of this method [24]. It significantly reduces query time by adding more shortcuts to the graph, but spans different levels at the cost of increasing space usage and preprocessing time [24].

The algorithm based on arc separator reduces the number of cutting arcs connecting the boundary vertices of different elements as much as possible [16]. In order to preserve the distance between boundary vertices in each pixel, a shortcut is added to the overlay [16]. The customized route planning (CRP) algorithm is a variant of this method, which is mainly used to meet the needs of real-world road network, such as dealing with turning costs and performing fast updating of cost functions [25]. Its preprocessing is divided into two stages [25]. In the first stage, the superimposed multi-layer partition and topology are calculated [25]. In the second stage, the cost of group arc is calculated by bottom-up and parallel processing units [25].

D. Hierachical Techniques

The hierarchical method mainly uses the inherent level of road network [16]. In the road network, important roads (such as highways) constitute a small arterial subnet. When the query algorithm is far away from the source and target, the algorithm only scans the vertices of the above subnets [16]. In the preprocessing stage, the importance of the vertex or arc is calculated according to the actual shortest path structure. The contraction hierarchy (CH) algorithm is a hierarchical technique [26]. The main idea is to create a shortcut to skip the less important vertices [26]. If the shortest path between them is unique and contains vertices to delete (i.e., shrink), it repeats the vertex shrink operation, which removes the least important vertices from the drawing and creates a shortcut between each pair of adjacent vertices [26]. CH can be used as the basis for other point-to-point algorithms and extended queries because it is universal [26].

E. Summary

1) think about the using time, traffic

IV. Related Work of Path Planning Subsystem

In this section, the related techniques reported in the literature for the Route Planning subsystem will be surveyed.

A set of paths, $P=\{P_1,P_2,...,P_{|P|}\}$, are computed by the path planning subsytem by thinking about the route planned by route planning subsystem, the ego vehicle's state, traffic rules and the surrounding environment [1]. A sequence of pose $p_i=(x_i,y_i,\theta_i)$ makes up a path $P_j=\{p_1,p_2,...,p_{|P_j|}\}$, and the pose includes the position of vehicle and its orientation in the map [1]. Algorithms for path planning susbsytem mainly includes two classes: graph serach based and interpolating curve based [1].

A. Graph Search Based Algorithms

In graph search based algorithms, all possible ego vehicle's state make up a state space which is denoted as a graph [27]. The best path between ego vehicle's current state and a goal state is searched by this kind of algorithms. Additionally, the goal state is a pose of a way point w_i of the route W obtained by the route planning sussystem [27]. For path planning subsystem, Dijkstra, A^* and the variants of A^* are three kinds of most common grap search based algorithms [1].

The detail of Dijkstra algorithm can be refered in Section III-A. Odin, an autonomous vehicel, uses Dijkstra algorithm to compute a path which is entering or away park [28].

Additionally, Dijkstra algorithm is employed to construct a path in simulation environment [29]. [30] computes a path for the autonomous vehicle Verdino bu using Dijkstra algorithm.

The detail of A^* algorithm can be refered in Section III-A. Rocky, an autonomous vehicle, computes path with A^* algorithm in the 2005 DARPA Grand Challenge [31]. A^* with two different heuristic cost function is applied to build a path and was tested in an autonomous vehicle, AnnieWAY [32]. One heuristic cost function, named as Rotation Translation Rotation (RTR), thinks about the vehicle's kinematics constraints of the car, and the other uses the information of positions and shapes of all obstacles including static obstacles and dynamic obstacles [32].

Other literatures use A^* variants in path planning susbsytem. The anytime D^* is proposed to compute a path for the autonomous vehicles Boss [33]. For the autonomous vehicle Junior, the hybrid-state-A^* is used to construct a path [34]. A kind of A^* variant is introduced to generate a smooth path by thinking about the kinematic constraints of vehicles [35].

B. Interpolating Curve Based Algorithms

Interpolating curve based algorithms insert some new points into a already known set of points (for example, the way points generated by route planning subsystem) to generate a smooth path [1]. Spline curves are the most used in interpolating curve based algorithms in path planning subsystem [1]. Cubic spline curve is employed in [36] and [37] in path planning subsystem. Both of them build a centerline from the route extracted from the road network [36], [37]. They use arc length and offset to the centerline to generate a series of parametric cubic splines that represent possible candidate paths [36], [37]. The best path is selected based on the function of safety and comfort [36], [37].

V. Related Work of Behavior Selection Subsystem

In this section, the related techniques reported in the literature for the behavior selection subsystem will be surveyed.

Given a path planned by the path planning subsystem, the behavior selection subsystem needs to select an suitable behavior at any time according to real-time traffic, the behaviors of other traffic participants, traffic rule and traffic signals [5]. For example, when a vehicle is approaching an intersection, the behavior layer will command the vehicle when to go straight, turn left or turn right. Methods for behavior selection subsystem can be mainly divided into three classes: finite state machine (FSM) based methods, ontology-based methods and Markov decision processes based method [1].

A. FSM-based Methods

In FSM-based methods, next action of the ego vehicle is selected according to a rule-based decision process in different traffic sence [1]. Normally, state denotes the behavior and transition conditions are the information extracted from perception system [1]. The main disadvantage of this method is that it is difficult to model all the uncertain and complex urban traffic scenes [1]. Junior team used FSM to record several simple urban traffic sences in DARPA Urban Challenge [7]. Additionally, the autonomous vehicle A1 used a FSM that had states for more complex traffic scenario to select a suitable driving behavior [38].

Each traffic sence uses a FSM in the IARA's behavior selection subsystem [39]. Some state transition rules are checked in order to find next suitable driving behavior in FSM for each scene according to the information provided by perception system [39]. A network which mixes decision tree and deterministic automate is a kind of model of the behavior selection subsystem [40]. It is a good idea to reduce the model complexity that dividing the driving task into a finite set of logitudinal and laterral guidance states [40]. Cruise controls states and a single critical control state are two components of the longitudinal states. The lateral guidance state mainly includes the state about lane keeping or lane chnaging [40]. In roundabouts situations, Okumura et al. modeled a high-level behavior selection subsystem as a classifier of hybrid a support vector machine (SVM) and FSM [41]. There are two levels in this behavior selection subsystem [41]. Firstly, the SVM classifier maps the ego vehicle's current state and information provided by perception system to next action of the vehicle [41]. Secondly, the action is tranfer to high-level commands by the FSM [41]. A hierarchical concurrent state machine is used in behavior selection subsystem to due with more complex traffic scenes [42]. A set of constraints are genrated by this kind of behavior selection subsystem by thinking about information such as lane characteristics [42].

B. Ontology-based Methods

Ontology is a framework of knowledge representation, which can be used to model concepts and their relationships. Ontology based knowledge base to model traffic regulations and sensor data is used to help autopilot understand the world [43]. In order to build the behavior selection subsystem, ontology based knowledge base is build by human. What happens at intersections and on narrow roads is focused on [43]. The system makes decisions considering rules such as right of way rules, and sends decisions such as ”left”, ”stop” or ”give way” to the path planning system to change the route or stop to avoid collision [43]. The disadvantage of this approach is the need to design an accurate world model, which consists of items such as lanes and traffic rules drawn at each location, which are usually done manually [43]. In recent work, others improved their previous work to use only a small part of the knowledge base, making it 1/20 to 1/10 smaller than the previous work [44].

C. Markov Decision Processes Based Methods

The uncertainty in perception and actions transition between states can be solved by the partially observable Markov decision process (POMDP) [1]. Online POMDP is employed in behavior selection subsystem to change lane in city environments [45]. The proposed method is divided into two parts in order to do real-time decision making and reduce the complexity of the POMDP [45]. At first, a signal processing network is used to evaluated the situation [45]. The network is a graph that thinks about velocities, relative distance and time to collisions with obstacle near the ego vehicle, and output if changing lane or not [45]. The POMDP only plans online for the curretn state by using the network output [45].

In order to reson about observation uncertainty and potentially hidden objects, a continuous POMDP method is used by thinking about the interactions of other traffic participants [46]. The planning is speeded up by assuming finite number of policies [46]. As a first step, the reward function aims to optimize comfort and efficiency by returning the acceleration and deceleration costs to reach the target area [46]. This step depends only on the state of the vehicle and the previously defined objectives [46]. The second step is to consider other traffic participants by increasing the cost of collision with other road users [46]. The reward function is the summation of two steps cost, and the scalar value of it denotes the driving objection [46].

Galceran et al. proposed an integrated reasoning and behavior selection method, which modeled vehicle behavior and nearby vehicles as a group of discrete strategies [47]. In this work, they used a set of manual design strategies to deal with in lane and intersection driving situations [47]. In the history of other vehicles, they used Bayesian change point detection to infer possible future actions, thus estimating the distribution of potential strategies that each vehicle in the vicinity might be executing [47]. Then, the behavior selection algorithm performs the strategy with the maximum reward value by approaching the POMDP solution that evaluates the strategy of the predicted vehicle through forward simulation [47]. However, they assume that most driving participants follow traffic rules and act in a regular, predictable way. The experiment was carried out on an autopilot vehicle platform [47].

Wray proposes an intermodal behavior selection subsystem for automated driving vehicles using interactive multiple online decision components (MODIA) instead of a set of policies [48]. Modia models the traffic participants as independent Markov decision process (MDP) [48]. Each MDP maintains its own beliefs and proposed actions at each time step, resulting in a set of estimated actions [48]. The dictionary performer action function (leaf) performs only the best (in terms of preference) actions in this set (for example, stop actions have preferences) [48]. The action can simply stop, edge or walk, and encode the motion by assigning the required velocity and target point along the AV trajectory [48]. Modia is still easy to handle because the number of instantiated decision processes increases linearly [48]. The method is tested in the automatic driving vehicle under the scene of intersection, and compared with the ignorance and naive baseline algorithm, successfully solving the problem of automatic driving vehicle interaction in the intersection scene [48].

VI. Related Work of Motion Planning Subsystem

In the autonomous driving system, motion planning is an extremely important module, responsible for generating a safe, effective, and collision-free motion trajectory from the starting point to the destination for the autonomous vehicle [27]. In the industry, motion planning is often divided into path planning (global planning) and trajectory planning (local planning) [27]. The sequence of points or curves connecting the starting point and the end point is called a path, and the strategy of forming a path is called path planning [27]. The goal of path planning is to make the distance between the path and obstacles as far as possible and the length of the path as short as possible; the purpose of trajectory planning is to make the running time of the car as short as possible or the energy as small as possible [27]. Trajectory planning adds time series information on the basis of path planning to plan the speed and acceleration of the car when performing tasks to meet the requirements of smoothness and speed controllability [27].

Considering that sometimes the algorithms used in global planning and local planning can be used interchangeably, we will explain the advantages and disadvantages of the methods used in global planning and local planning [5]. At present, the more widely used motion planning methods mainly include the following categories: motion planning methods based on graph search, motion planning methods based on random sampling, methods based on numerical optimization, methods based on curve interpolation, methods based on probability, planning methods based on machine learning, etc [5].

A. Graph Search Based Methods

The basic idea of the method based on graph search is to traverse the space from point A to point B. This method usually represents the state space as an occupancy grid or grid, which describes objects in the environment. From a planning point of view, a path can be set to implement the graph search algorithm to access different states in the grid and provide solutions. Some of these algorithms have been applied to the development of autonomous vehicles.

Dijkstra algorithm is a graph search algorithm to find the single source shortest path in the graph [14]. The configuration space is approximately a discrete unit grid space, lattice, such as [59] and [60]. The concept and implementation of the algorithm are described in [61] and [62]. In autonomous driving, it has been implemented by [29] in a multi-cabin simulation, with the Ben Franklin Racing Team entering the DARPA Urban Challenge [63] and Victor Tango Team [28].

A^* algorithm is a heuristic implementation based on graph search algorithm, enabling fast node search (it is an extension of Dijkstra graph search algorithm) [15]. Its most important design aspect is the determination of the cost function, which defines the weight of the node, which is suitable for searching the space mainly known in advance by the vehicle [64], but its cost is higher in terms of memory and speed [15].

Some path planning algorithms for mobile robots have also been used as improvements, and are used in autonomous driving systems, such as Field D star [65], Theta star [66], Anytime repairing A star (ARA star) and anytime D star (AD star) [67] and so on. Ziegler et al. applied A star algorithm and

TABLE I
PROS AND CONS OF ALL KINDS OF MOTION PLANNING METHODS

Algorithm	Pros	Cons
Dijkstra [14]	1. Suitable for structured or unstructured roads. 2. The shortest path can be found in the grid map.	1. Due to the need to traverse the nodes, the computational cost is large. 2. The path obtained is not continuous and is not suitable for automatic driving scenarios that require real-time calculation.
A^* series [15]	1. Extension of Dijkstra's algorithm. 2. A heuristic algorithm. 3. Can reduce calculation time.	1. The searched path is not continuous. 2. The heuristic rule is not applicable in some scenarios.
Status grid [49]	1. Can handle multi-dimensional issues (such as position, velocity, acceleration, time, etc.). 2. Suitable for local planning in a dynamic environment.	1. Computationally expensive. 2. Only suitable for more discrete state space.
RRT series [50]	1. Can provide a fast search algorithm in complex space. 2. Both global planning and local planning can be used.	1. The resulting trajectory is not continuous. 2. The optimization degree of the trajectory is more dependent on the time frame sequence of the RRT.
Curve interpolation [27]	1. Suitable for curve smoothness optimization. 2. Can generate a comfortable and continuous trajectory in a dynamic environment. 3. Simple calculation.	1. Rely on the path points generated by the global plan. 2. Large computational expense when dealing with obstacle scenes.
Clothoid [51]	1. Smooth transition between straight line and curve. 2. Suitable for local planning.	1. Due to the definition of curve integral, the calculation consumes a lot. 2. The curve is continuous but not smooth
Polynomial curve [52]–[54]	1. Low computational cost. 2. Can generate continuous smooth motion trajectory	1. It is often a polynomial of order 4 or higher, resulting in a large computational cost.
Bezier curve [55], [56]	1. Low computational cost. 2. Curve shape can be determined by control points. 3. Continuous smooth motion trajectory can be generated	1. Rely on the path points generated by the global plan. 2. When the curve dimension is increased, the calculation consumes more.
Spline curve [57], [58]	1. Low computational cost. 2. Continuous curve is controlled by different nodes	1. The resulting path may not be optimal.

Voronoi cost function to plan unstructured space application scenarios [68].

State grid algorithm uses the discrete of the planned area of the state grid to represent the actual road environment [1]. This grid is called the status grid [5]. The motion planning search is applied in [49]. The path search in this algorithm is based on a set of grids containing all feasible features, allowing the vehicle to travel from the initial state to several other states [49]. The cost function determines the pre-calculated optimal path grid [49].

B. Random Sampling Based Methods

This type of planning method attempts to solve the problem of time constraints, that is, a type of problem that cannot be satisfied by using deterministic methods in high-dimensional spaces [5]. This broad category of methods includes random sampling configuration space or state space, among which the most commonly used methods are the Probabilistic Road Map Method (PRM) [69] and the Rapid Exploration Random Tree (RRT) [70]. The latter has a wide range of applications in the field of autonomous driving.

Rapid Search Random Tree (RRT): It is a sampling-based algorithm used in path planning [50]. It can perform rapid planning in a semi-structured space [70], and then perform random search through the navigation area. So as to plan a suitable path [50]. This method can also consider some incomplete constraints, such as the maximum turning radius and momentum of the vehicle [50].

These algorithms are reviewed in [71] and [50]. In the autonomous driving system, this type of algorithm has been used by the MIT team in the DARPA Urban Challenge [72]. However, the resulting path is not optimal. In [73], an algorithm named RRT star was developed, which can achieve convergence to the optimal solution.

C. Numerical Optimization Based Methods

Numerical optimization based methods includes various methods based on numerical optimization and nonlinear optimization. These methods consider the influence of different constraint variables, and then generate the optimal path by minimizing or maximizing the function [5]. In the actual path planning method, it usually optimizes the previously rough trajectory to generate a smooth and feasible trajectory [74]. In addition, the optimal trajectory is also calculated from the kinematics-based constraints [75].

Function optimization finds the root of an actual value function (minimize variable result) [5]. It can be used to improve the potential field method (PFM) and be applied to mobile robots in narrow passage environments and autonomous vehicles in special situations [5]. This type of method can realize continuous trajectory position, velocity, and acceleration as planning parameters [5].

D. Curve Interpolation Based Methods

The curve used in curve interpolation based methods includes Clothoid curve, polynomial curve, Bezier curve and spline curve method, etc [5]. Computer-aided geometric design technology is often used to solve smooth path points for a given path [51]. This allows the motion planning algorithm to consider its feasibility, comfort, and vehicle dynamics related parameters to generate the required trajectory [51].

Interpolation is defined as constructing and inserting a set of new data, known set reference points, within the previous data range [27]. This means that these algorithms can use the previous set of path points to generate a new set of data (that is, a smoother path), which is more conducive to the continuity of the trajectory and is more suitable for the constraints of vehicle driving and dynamic environment [27]. When planning, if there are obstacles, it can generate a new path, and then re-input the previously planned path to get the feasible trajectory we want [27].

The method based on the clothoid curve is defined based on the Fresnel integral [51]. The use of clothoid curves can define the trajectory of linear changes in curvature, because their curvature is equal to the arc length; and smooth transitions can be made between straight and curved segments [51]. This type of method has been applied in highway and railway driving scenarios, and it is also suitable for some wheeled mobile robots [76].

Polynomial curves are usually implemented to meet the constraints required by their interpolation points, where the constraints specifically include fitting position, angle, and curvature [27]. The expected values or constraints in the beginning and end of this type of method will determine the coefficient of the curve. For the calculation of polynomial coefficients, please refer to [52], [53] and [54].

Bezier curve is a kind of parametric curve method, which relies on control points to determine the curve shape. The core of the Bezier curve is the Bernstein polynomial. This type of method has been widely used in CAGD applications, technical drawing, aviation, and automotive design. Its advantage is low computational cost, because the shape of the curve is determined by the control points. We can place these control points correctly to meet the constraints of the beginning and end of curvature [55], [56].

A good example of Bezier modularity and ductility is in [77], which can generate interconnected circular shapes and interconnected Bezier curves with continuous curvature. These curves are usually used to approximate cyclotron curves [78], [79], or to achieve reasonable Bezier rapid planning curves, such as spline curve. Spline is a piecewise polynomial parameter in the graded interval [80]. The curve is defined as a polynomial curve [52]. In fact, B-spline [57], [58] can also be represented by Bezier curve [71] or clothoid curve [76]. The connections between each sub-segment are called knots, and they usually have a high degree of smoothness constraints [71], [76].

E. Probability Based Methods

It mainly includes probability-based planning methods such as MDP and POMDP [5]. When dealing with uncertainties in dynamic environments, a widely used model framework is the partially visible Markov decision process model [5]. Part of it can be seen that the Markov decision process model provides a framework that can deal with uncertain variables in the environment, such as the measurement error of the sensor, the uncertainty of the location of surrounding obstacles, and so on [5].

In the work of [81]–[83], they used the partially visible Markov decision process model, and used the pedestrian's target position as the state of attention and belief to predict the future operation state of the pedestrian. This method can be used to plan and calculate a safe, effective and collision-free motion trajectory in the partially visible Markov decision process model [81]–[83]. These methods are more suitable for motion planning in special scenes, such as special scenes such as entrances and exits and sidewalks [81]–[83]. However, in the actual autonomous driving environment, it is more difficult to obtain the target position of a car without the help of real-time maps [81]–[83]. This also makes the method lacks a certain degree of robustness and is difficult to apply to actual autonomous driving system [81]–[83]. In the literature [84], it plans a suitable trajectory for the autonomous car by constructing a linear motion model and then evaluating the future position of the surrounding cars [84]. However, it also assumes that the speed of the surrounding cars is a constant value. This type of motion planning method can only be applied to some simple driving scenarios [84].

Considering that some of the visible Markov decision process models have a strong ability to deal with uncertainty problems, there have been more motion planning algorithms based on this model in recent years [1]. The model uses unobservable variables from other surrounding cars as hidden in the model. Model the variable state information [5].

In the previous process of solving partially visible Markov decision process models, we usually focused on solving problems in the offline state [27]. Obviously, it can be understood literally that to solve the problem in the offline state, the optimal strategy we solve is not under the current belief state, so this type of solver in the offline state is not suitable for actual automatic driving scene. In addition, when solving this type of part, the Markov decision process model can be seen. For a large number of state spaces, the computational cost is also relatively large [27]. Even for some small model problems in the offline state, it often takes several minutes or even hours of computational consumption to estimate the optimal solution [27]. However, for dynamic and complex driving scenarios, we must update the decision-making status of our car frequently (for example, every 100 milliseconds) to ensure its driving safety [27]. Therefore, in most cases, we often simplify this type of problem, which can effectively reduce its computational complexity and is more suitable for actual autonomous driving conditions [27].

In the work of [85], it takes the attention state of other surrounding cars as a hidden state variable in the partially visible Markov decision process model, and by planning the movement trajectories of all surrounding cars on a previously known path. To simplify the model, thereby effectively reducing the state space dimension of this type of problem, thereby reducing its computational resource consumption [85].

In addition, in the work of [85], the author proposed a real-time planning method for urban driving environment, which combines multi-strategy behavior evaluation method and rolling time domain trajectory planning method, which can be used in real-time urban autonomous driving environment. The author also adopted a partially visible Markov decision process model and used it for the future trajectory evaluation of other cars in the driving scene [85]. After that, a nonlinear model predictive controller based on chance constraints was used to obtain our optimal trajectory planning strategy [85].

F. Machine Learning Based Methods

Machine learning based methods including various end-to-end motion planning methods such as reinforcement learning [86]. Bayesian reinforcement learning methods [86] and deep reinforcement learning methods [87] have also been proposed to plan safe, effective, and collision-free vehicle trajectories in dynamic environments. Unfortunately, these methods have many limitations. These methods can not only be used for a large amount of training data in a discrete state space, but also consume a lot of computing resources. Compared with these methods, our method does not require so much data for the training set, so it can save more computing resources. At the same time, NVIDIA recently proposed a vision-based end-to-end motion planning model [88], which directly inputs the video information collected by the car camera to the neural network model, and directly outputs the control value of the car after processing. The decision-making and path planning process is improved [88]. This method has been verified in his paper, but obviously it only relies on the input of visual information [88]. In bad weather, the visual information will be greatly affected, and it will greatly test the safety of the car used in driving scenarios [88].

VII. Related Work of Obstacle Avoidance Subsystem

Unmanned vehicle navigation is a classic research field in the field of robotics, and obstacle avoidance is the core technology in autonomous driving navigation systems [1]. In the process of autonomous navigation, the robot needs to reach the destination as quickly as possible while avoiding obstacles [1]. Therefore, in the field of robotics, many algorithms have been developed to solve this problem for different environments [1]. In the field of autonomous driving navigation, the environment is more complex, with moving obstacles of different shapes and static obstacles with winding and winding, therefore higher requirements for obstacle avoidance technology are put forward [1]. For autonomous vehicles, high-precision sensors are necessary to detect the surrounding environment and obstacles to help the algorithm plan the correct obstacle avoidance path [1]. The methods of abstacle avoidance subsytem can be divided into two classes: sensor-based methods and model-based methods.

A. Sensor-based Methods

Sensor-based obstacle avoidance technology refers to the recognition of the environment only through the acquired sensor data, and then the corresponding control is made to avoid obstacles [1]. Commonly used sensors include cameras, lidars, optical flow sensors, etc [1]. For the global sensor-based path planning, the environment is modeled through the data obtained by the sensors, and then the global path planning is performed [1]. On the other hand, local path planning uses sensors on the vehicle to observe the surrounding unknown environment and nearby obstacles, and continuously calculate the planned trajectory under the current observation environment [1]. This method reduces the calculation time and is easy to implement a real-time navigation system [1]. There are many obstacle avoidance methods for static obstacles in local path planning, such as dynamic window method [89], curvature speed method [90], lane curvature method [91] and so on. On the other hand, there are speed obstacle method [92], collision cone method [93] and unavoidable collision state method [94] for dynamic obstacles. These methods assume that the speed and position of the dynamic obstacle are known at each moment to plan their own obstacle avoidance path.

B. Model-based Methods

The model predictive control (MPC) is one of the method of model-based methods [5]. Model Predictive Control (MPC) is an obstacle avoidance technology widely used in vehicle autonomous navigation [95]. MPC not only includes a stable path planning algorithm but also ensures convergence [95]. When the robot runs MPC, in each time substep, a collision-free path is planned through the algorithm, and then according to the vehicle's dynamic model, the corresponding control speed of the path will be applied to the vehicle [95]. The continuous update cycle process enables vehicles to run online path planning algorithms to deal with different environments [95]. MPC mainly includes three parts, vehicle model, trajectory prediction and speed control [95]. A major advantage of the MPC method is that it has many corresponding variant algorithms for different environments to resist the interference of environmental factors [95].

VIII. Conclusion

Acknowledgment

References

[1] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741742030628X

[2] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Toward autonomous driving: The cmu navlab part ii — architecture and systems," *IEEE Expert-Intelligent Systems and their Applications*, vol. 6, no. 4, pp. 44–52, 1991, cited By 32. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0026208452&doi=10.1109%2f64.85920&partnerID=40&md5=b5f4ffce8a77610673add61ca0017da0

[3] A. Broggi, M. Bertozzi, and A. Fascioli, "Argo and the millemiglia in automatico tour," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 1, pp. 55–64, 1999, cited By 63. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0032739352&doi=10.1109%2f5254.747906&partnerID=40&md5=59ef7fb52d03255108c0daf79cf8065f

[4] R. Gregor, M. Lützeler, M. Pellkofer, K.-H. Siedersberger, and E. Dickmanns, "Ems-vision: A perceptual system for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 48–59, 2002, cited By 74. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-19044381801&doi=10.1109%2f6979.994795&partnerID=40&md5=0489977a5732f1882f41c889972bc86c

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

[6] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[7] ——, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[8] J. Xin, C. Wang, Z. Zhang, and N. Zheng, "China future challenge: Beyond the intelligent vehicle," *IEEE Intell. Transp. Syst. Soc. Newslett*, vol. 16, no. 2, pp. 8–10, 2014.

[9] P. Cerri, G. Soprani, P. Zani, J. Choi, J. Lee, D. Kim, K. Yi, and A. Broggi, "Computer vision at the hyundai autonomous challenge," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 777–783.

[10] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, "The vislab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 3, pp. 147–164, 2012.

[11] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, 2016.

[12] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti, "Proud—public road urban driverless-car test," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3508–3519, 2015.

[13] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE International: Warrendale, PA, USA*, 2018.

[14] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[16] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route planning in transportation networks," in *Algorithm engineering*. Springer, 2016, pp. 19–80.

[17] G. E. Forsythe, "Algorithms," *Communications of the ACM*, vol. 7, no. 6, pp. 347–349, 1964.

[18] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[19] L. R. Ford Jr, "Network flow theory," Rand Corp Santa Monica Ca, Tech. Rep., 1956.

[20] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[21] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A search meets graph theory." in *SODA*, vol. 5, 2005, pp. 156–165.

[22] M. Hilger, E. Köhler, R. H. Möhring, and H. Schilling, "Fast point-to-point shortest path computations with arc-flags," *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, vol. 74, pp. 41–72, 2009.

[23] U. Lauther, "An experimental evaluation of point-to-point shortest path calculation on road networks with precalculated edge-flags." in *The Shortest Path Problem*, 2006, pp. 19–39.

[24] D. Delling, M. Holzer, K. Müller, F. Schulz, and D. Wagner, "High-performance multi-level routing," *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, vol. 74, pp. 73–92, 2009.

[25] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, "Customizable route planning in road networks," *Transportation Science*, vol. 51, no. 2, pp. 566–591, 2017.

[26] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies," *Transportation Science*, vol. 46, no. 3, pp. 388–404, 2012.

[27] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[28] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[29] R. Kala and K. Warwick, "Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 3-4, pp. 559–590, 2013.

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

[31] B. M. Leedy, J. S. Putney, C. Bauman, S. Cacciola, J. Michael Webster, and C. F. Reinholtz, *Virginia Tech's Twin Contenders: A Comparative Study of Reactive and Deliberative Navigation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 155–182. [Online]. Available: https://doi.org/10.1007/978-3-540-73429-1_5

[32] J. Ziegler, M. Werling, and J. Schröder, "Navigating car-like robots in unstructured environments using an obstacle sensitive cost function," 2008, pp. 787–791, cited By 46. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-57749196757&doi=10.1109%2fIVS.2008.4621302&partnerID=40&md5=11accfcd3da5016fd0cbe4e4919fd9b7

[33] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20255

[34] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010. [Online]. Available: https://doi.org/10.1177/0278364909359210

[35] K. Chu, J. Kim, K. Jo, and M. Sunwoo, "Real-time path planning of autonomous vehicles for unstructured road navigation," *International Journal of Automotive Technology*, vol. 16, no. 4, pp. 653–668, 2015.

[36] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1599–1616, 2012.

[37] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 482 – 500, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327017303825

[38] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—part ii: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 8, pp. 5119–5132, 2015.

[39] R. Guidolini, L. G. Scart, L. F. R. Jesus, V. B. Cardoso, C. Badue, and T. Oliveira-Santos, "Handling pedestrians in crosswalks using deep neural networks in the iara autonomous car," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[40] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen, "Experience, results and lessons learned from automated driving on germany's highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 42–57, 2015.

[41] B. Okumura, M. R. James, Y. Kanzawa, M. Derry, K. Sakai, T. Nishi, and D. Prokhorov, "Challenges in perception and decision making for intelligent automotive vehicles: A case study," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 20–32, 2016.

[42] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[43] L. Zhao, R. Ichise, T. Yoshikawa, T. Naito, T. Kakinami, and Y. Sasaki, "Ontology-based decision making on uncontrolled intersections and

narrow roads," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 83–88.

[44] L. Zhao, R. Ichise, Z. Liu, S. Mita, and Y. Sasaki, "Ontology-based driving decision making: A feasibility study at uncontrolled intersections," *IEICE Transactions on Information and Systems*, vol. E100D, no. 7, pp. 1425–1439, 2017, cited By 10. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85021751087&doi=10.1587%2ftransinf.2016EDP7337&partnerID=40&md5=f930067d3514ba87c935ae822a11e1c5

[45] S. Ulbrich and M. Maurer, "Probabilistic online pomdp decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 2063–2067.

[46] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," 2014, pp. 392–399, cited By 110. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84937131521&doi=10.1109%2fITSC.2014.6957722&partnerID=40&md5=d969a3200443b25288d2a06cf93e18f0

[47] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, vol. 41, no. 6, pp. 1367–1382, 2017.

[48] K. H. Wray, S. J. Witwicki, and S. Zilberstein, "Online decision-making for scalable autonomous systems," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 4768–4774. [Online]. Available: https://doi.org/10.24963/ijcai.2017/664

[49] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*. Munich Germany, 2005, pp. 1–7.

[50] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[51] M. Brezak and I. Petrović, "Real-time approximation of clothoids with bounded error for path planning applications," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 507–515, 2013.

[52] A. Piazzi, C. L. Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic g/sup 2/-splines for the iterative steering of vision-based autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 27–36, 2002.

[53] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *IEEE Transactions on intelligent transportation systems*, vol. 11, no. 3, pp. 589–606, 2010.

[54] A. Simon and J. C. Becker, "Vehicle guidance for an autonomous vehicle," in *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383)*. IEEE, 1999, pp. 429–434.

[55] J. P. Rastelli, R. Lattarulo, and F. Nashashibi, "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 510–515.

[56] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. IEEE, 2008, pp. 158–166.

[57] Z. Shiller, Y.-R. Gwo *et al.*, "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, 1991.

[58] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 167–172, 2009.

[59] F. M. Marchese, "Multiple mobile robots path-planning with mca," in *International Conference on Autonomic and Autonomous Systems (ICAS'06)*. IEEE, 2006, pp. 56–56.

[60] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[61] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[62] J. Y. Hwang, J. S. Kim, S. S. Lim, and K. H. Park, "A fast path planning by path graph optimization," *IEEE Transactions on systems, man, and cybernetics-part a: systems and humans*, vol. 33, no. 1, pp. 121–129, 2003.

[63] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[64] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[65] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field d* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006.

[66] M. J. Kahana, D. Seelig, and J. R. Madsen, "Theta returns," *Current Opinion in Neurobiology*, vol. 11, no. 6, pp. 739 – 744, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959438801002781

[67] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613 – 1643, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000437020800060X

[68] J. Ziegler, M. Werling, and J. Schroder, "Navigating car-like robots in unstructured environments using an obstacle sensitive cost function," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 787–791.

[69] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[70] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[71] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[72] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[73] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

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

[75] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent transportation systems magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[76] D. J. Walton and D. S. Meek, "A controlled clothoid spline," *Computers & Graphics*, vol. 29, no. 3, pp. 353–363, 2005.

[77] K. Setsompop, B. A. Gagoski, J. R. Polimeni, T. Witzel, V. J. Wedeen, and L. L. Wald, "Blipped-controlled aliasing in parallel imaging for simultaneous multislice echo planar imaging with reduced g-factor penalty," *Magnetic resonance in medicine*, vol. 67, no. 5, pp. 1210–1224, 2012.

[78] L. Wang, K. T. Miura, E. Nakamae, T. Yamamoto, and T. J. Wang, "An approximation approach of the clothoid curve defined in the interval [0, π/2] and its offset by free-form curves," *Computer-Aided Design*, vol. 33, no. 14, pp. 1049–1058, 2001.

[79] J. Sánchez-Reyes and J. M. Chacón, "Polynomial approximation to clothoids via s-power series," *Computer-Aided Design*, vol. 35, no. 14, pp. 1305–1313, 2003.

[80] N. Montés, A. Herraez, L. Armesto, and J. Tornero, "Real-time clothoid approximation by rational bezier curves," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 2246–2251.

[81] S. Bansal, A. Cosgun, A. Nakhaei, and K. Fujimura, "Collaborative planning for mixed-autonomy lane merging," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4449–4455.

[82] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, "Porca: Modeling and planning for autonomous driving among many pedestrians," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418–3425, 2018.

[83] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee*

international conference on robotics and automation (icra). IEEE, 2015, pp. 454–460.

[84] S. Brechtel, T. Gindele, and R. Dillmann, “Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps,” in *17th international IEEE conference on intelligent transportation systems (ITSC)*. IEEE, 2014, pp. 392–399.

[85] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, “Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1671–1678.

[86] Y. Wang, K. S. Won, D. Hsu, and W. S. Lee, “Monte carlo bayesian reinforcement learning,” *arXiv preprint arXiv:1206.6449*, 2012.

[87] M. Everett, Y. F. Chen, and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[88] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[89] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[90] R. Simmons, “The curvature-velocity method for local obstacle avoidance,” in *Proceedings of IEEE international conference on robotics and automation*, vol. 4. IEEE, 1996, pp. 3375–3382.

[91] N. Y. Ko and R. G. Simmons, “The lane-curvature method for local obstacle avoidance,” in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, vol. 3. IEEE, 1998, pp. 1615–1621.

[92] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[93] A. Chakravarthy and D. Ghose, “Obstacle avoidance in a dynamic environment: A collision cone approach,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.

[94] T. Fraichard and H. Asama, “Inevitable collision states—a step towards safer robots?” *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[95] D. Q. Mayne and S. Raković, “Model predictive control of constrained piecewise affine discrete-time systems,” *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 13, no. 3-4, pp. 261–279, 2003.