A Technical Review of Autonomous Driving

Chenyi Song, Siddhartha Palit, Wenhao Zhang, Alexander Dennington, Haochen Jin
cs3n21@soton.ac.uk, sp4u21@soton.ac.uk, wz8u20@soton.ac.uk, ad2n18@soton.ac.uk, hj1u21@soton.ac.uk

Abstract: This paper reviews and summarizes the technologies in autonomous driving. First, a overview of the General architecture of autonomous vehicles is given. Then the solutions applied in each part are discussed. Finally, prospects for the future are presented.

1. Introduction

Human interest in autonomous driving can be traced back to 1925. In recent years, the research boom in autonomous driving began with the three challenges held by the Defense Advanced Research Projects Agency (DARPA) in 2004-2007.

The autonomous vehicle is a kind of intelligent car that free the people from driving. Although the division of the structure of autonomous driving system is inconsistent in different literature, the architecture consists of several parts, environment perception, decision-making and control execution.[1]

This review gives a general look at autonomous vehicles and discusses solutions that are applied in environment perception, decision-making and control execution.

2. General architecture of autonomous vehicles

The autonomous vehicle is the focal point of a number of different machine learning disciplines, ranging from image recognition and object detection to reinforcement learning. These disciplines contribute to the individual systems that undertake environment perception, decision-making and control execution.[1]

2.1. Environment perception

This is the system involved with determining the surroundings of the vehicle, using a range of cameras and lasers mounted around the vehicle. Using pre-trained models, it can recognise different elements of the scene, detect potential obstacles and determine the car's position on the road. The environment perception system does this using deep learning methods which have become a staple of image processing, namely convolutional neural networks and recurrent neural networks. These have been pre-trained on a large dataset of labelled images to give them the prior knowledge needed to detect different objects faced by the vehicle, separate them from their surrounding environment, and categorise them with a high degree of accuracy. The convolutional and recurrent neural networks are generally used in obstacle detection and scene classification. Recent improvements in these areas have led to binocular convolutional neural networks which have a three-dimensional interpretation of the images, resulting in improved estimates of depth, and therefore improving obstacle detection and avoidance. Other types of deep learning techniques used in environmental perception include auto-encoders.[1][2]

2.2. Decision-making

The environmental output from the environment perception system is passed on to the Decision-making system, which then uses the information obtained about positional awareness, potential obstacles, route recognition - as well as taking into account traffic rules and passenger safety - to determine the vehicle's next course of action. An important decision to be made is path planning, determining where to move in order to reach the final destination. This decision can be broken down into two problems: local path and global path planning. The local path determines the short term movements of the vehicle in order to maneuver in the surrounding environment safely, while global path planning determines the bigger picture, specifically on making sure the car is making progress towards its final destination. These two problems must be balanced, in order for the car to have enough autonomy to react to environmental changes in the immediate vicinity, whilst also staying on-route towards its overall target. [1][2][3]

2.3. Control Execution

This is the final part of the process, where the sensory data is mapped into steering commands and speed-changes. The best-known method to perform the former task is a composite neural network, made up of a CNN and an LSTM RNN. The features from the first of these are fed into the

LSTM, which outputs a steering angle for the automated steering system to use.[1]

3. Environment Perception

3.1. Moving objects detection

Moving object detection is one of the most important in terms of avoiding any accidents or causalities and enabling smooth maneuvering and braking of car. Various methods are being used for object detection these days. We will discuss some of them.

3.1.1. Multi Sensor Fusion

In Multi-Sensor fusion, multiple sensors are used for detection of pedestrians, vehicles, and bicyclists. This model uses a point model (Mp) and a box model (Mb). Each object of interest such as vehicles, pedestrians and bicyclists have their own kinematics i.e., pedestrians can move in any direction they want but bicyclists and vehicles are constrained to non-holonomic motion.[4] Three different observation models are used:

1. Radar Observation Model
2. Lidar Observation Model
3. Camera Observation Model

The radar observation model is used for detection of point objects, Lidar Observation model are used for box target and Camera observation model is used for bounding box measurements in the image plane. This model was successful with 93.7% detection rate with 5.7% false positives per minute.

3.1.2. MODNeT: Motion and Appearance based object detection of Autonomous Vehicles

This Model detects and classifies an object simultaneously as moving or static. It uses a VGG-16 encoder to combine motion and appearance features together. This is followed by two decoders for motion detection and segmentation. The advantage of this model is it uses motion cues which none of the other model uses together with the detection and segmentation. The joint training improves the motion segmentation by 8.2% in F-score. This model outperforms MPNet by 21.5% in mAP and it is because the model provides a better representation for motion than MPNet by using KITTI data and not synthetic data. Although there are many advantages to this model, the main issue is it suffers from lack of large datasets for motion segmentation and is far from being deployed into real-world.[5]

3.1.3. FuseMODNet: For low-light Autonomous Driving

This proposes a model which fuses colored images with motion signals to generate motion masks. The encoder used is responsible for extracting features before the up-sampling phase which reduces cost of computation. The decoder is composed of three deconvolution layers because of which it has low-complexity and a light architecture. This model uses a three-stream mid-Fusion network which consists of three encoders for rgbFlow, RGB and lidarFlow respectively. An improvement in accuracy is observed. The lidarFlow Augmentation shows an improved accuracy of 2% in IoU compared to camera only solution. The results prove that motion segmentation task is beneficial regardless of illumination.[6]

3.1.4. LIDAR based Vehicle Detection

Traditionally, LIDAR based detection involves four steps:

1. Background Filtering
2. Object clustering
3. Feature extraction and
4. Object Classification

Classification requires expert knowledge. The density-based clustering does not work well with the point cloud data with non-uniform density. One disadvantage involves selection of parameters of the clustering algorithms.[7]

3.1.5. Frame Difference Model

In this model, the moving objects are detected by threshold of time difference in adjacent frame's pixels. The advantages of this model are good background update and adaptive performance but this model couldn't detect objects with comfortable inter color.

3.1.6. Detection Based on Invariant Moments

This model assumes that there is only one car at the front and can move freely in any direction. Object detection is done by separating background and moving objects based on colour, intensity, location in frame. Region of Interest is defined by the model using background image by the subtraction process. The proposed technique performs better in Recall, Precision and FoM (0.87,0.97,0.91) respectively compared to previous models.[8]

3.1.7. Enhanced Object Detection

This model uses an enhanced 2-Dobject detector based on faster R-CNN that improves on detection accuracy. The main aim of this model was to improve on default anchor generation and performance drop-in minority class. Since

performance improved with optimized parameters, it proves that default anchors of faster R-CNN are not suitable for object detection. The results also showed that assigning weights to class distribution was more effective. In future, they plan to improve the model on inference rates and achieving real time speed.[9]

3.2. Moving objects tracking

The Moving Object Tracker (MOT) subsystem gives instructions to the car to avoid collisions with dynamic objects in the self-driving car's surroundings by monitoring their path. For some time in the past, the location of moving obstacles has been estimated based on sensors that test distances and do their job by capturing data, like monocular cameras and LIDAR. Based on different technical bases, MOTs are classified into many categories, and in this paper we will focus on several approaches for MOTs, namely traditional, model-based and grid diagram-based MOTs.

3.2.1. Traditional MOT

The traditional MOT based approach is very straightforward and is divided into three phases, which are data segmentation, association and filtering. In the first stage, the data collected by the sensor is segmented by pattern recognition techniques. And in the second stage, the data is correlated with the target, i.e. moving obstacles. In the final stage, the geometric mean of the data is calculated and this is used to estimate the position of the obstacle, which is usually updated by a particle filter. In 2015, Amaral et al. proposed a method that uses a 3D laser sensor to track a moving obstacle: first the Euclidean distance of the points in the data map is calculated, how it is associated with the obstacle using a nearest neighbor algorithm, and finally a particle filter to estimate the state of the obstacle, set a threshold, and mark those whose speed exceeds the threshold as moving vehicles.[2]

3.2.2. Model-based MOT

Model-based MOT uses non-parametric filters that do not require segmentation and correlation of the data, but infer directly from the sensor usage data, based on the geometric model of the moving object built out by the sensor. In 2009, Thrun and Petrovskaya proposed to update the state of a moving object by combining a Kalman filter and a Rao-Blackwellized particle filter (RBPF) hybrid formulation to achieve state update of moving objects, using laser sensors to capture and update data of moving objects to infer hypotheses about the moving objects.

In the same year, Vu et al. also gave an MOT method, again based on a model. The principle of this method is to infer the possible trajectory of a moving obstacle within a window for a given time period, based on its laser measurements. This trajectory will take many forms, possibly I-shaped, L-shaped as well as prime, which all satisfy the constraints of the measurement and motion model. The complexity of this algorithm is quite high compared to other algorithms, most notably in the use of Markov chain Monte Carlo techniques (DD-MCMC).[2] This technique is data-driven and it is able to find the optimal solution in the spatial problem very efficiently by traversal. It generates observations over a period of time by sampling the probability distribution of the trajectory ground and performing iterations. After each iteration, the new state can be sampled again according to the generated judgment decision, and the new decision state is then accepted with a certain probability. The initial values of the iterations are fitted by predefining the object model ground, and the laser sensor measures the dynamic segment so as to generate hypotheses about possible moving obstacles.

3.2.3. Grid map-based MOT

The grid map-based MOT is somewhat similar to the traditional approach in that it also requires data segmentation, association, and filtering steps first to generate an object-level representation of the scene. Of course, before that it needs to construct a dynamic environment grid map.[10] In 2014, Azim et al. divided the environment into occupied, idle and unknown voxels by means of a 3D local occupancy grid map based on an octree. They first constructed the local grid map and subsequently monitored the moving targets based on the inconsistency of the idle and occupied spaces in it. The moving targets are subjected to cluster analysis and are classified into different classes based on motion and geometric features, like vehicles and pedestrians.

In 2017, Ge et al. completed modeling for static environments using 2.5D occupancy grid maps as a way to complete monitoring of moving targets. They stored the average height of 3D points of 2D projections falling into the cell space domain into grid cells, compared the grid to the environment model, and generated hypotheses based on the differences.

3.2.4. Other MOTs

There are other MOTs that are similar in idea to the previous methods, but each has some differences in the technical basis. Stereo vision-based MOT, for example, analyzes moving targets based on factors such as color and depth of stereo images. Sensor fusion-based MOT, on the other hand, combines data generated from different devices, such as cameras and LIDAR, and uses this data to analyze the individual characteristics of the obstacle.

Deep learning-based MOT, on the other hand, analyzes the location, geometry, etc. of moving targets by building deep neural networks to make predictions about their future behavior from the data.

4. Decision-Making

The decision-making module aims to drive the vehicle from its current location to the destination, considering the vehicle's current state and the environmental information obtained from the perception module.[2]

A typical decision planning module can be divided into three parts. Route Planning gives a long-term goal, that is the global path. After that, the Behavior Layer makes specific short-term behaviour decisions following the long-term goal combined with the environmental information. At last, the Motion Planning generates trajectories that satisfy constraints based on specific behavioral decisions.[11]

4.1. Route Planning

Route planning is similar to the navigation function that we often use in our lives. When the user gives the destination, the route planning finds an optimal path based on a high-precision map. The 'optimal' here means the shortest route or the fastest arrival time. This section will introduce two classic global path searching algorithms, Dijkstra's algorithm and the A* search algorithm. They and their variants are the most common algorithms used in route planning.

Dijkstra's algorithm is to find the shortest path between nodes in a graph. Basically, Dijkstra's algorithm does two things. First, it continuously runs the breadth-first algorithm to find visible points, and calculate the distance from the visible point to the start point. Second, it selects the shortest length from the currently known paths. Therefore, the result is guaranteed to be optimal if there is one. However, Dijkstra's algorithm is not efficient because each time it searches for the next visible points, it expands in every direction, including those far from the final goal.[12]

To solve the efficiency problem of Dijkstra's algorithm, A* search algorithm introduces a heuristic function, which is given by

$$f(n) = g(n) + h(n)$$

in which

- $g(n)$ is the actual cost from the start point to node n .
- $h(n)$ is the heuristic function, which means the estimated total cost from node n to goal.
- $f(n)$ is estimated total cost through node n to the goal.

The heuristic function is the core of A* search algorithm. By defining different heuristic functions, search conditions like the shortest route or the fastest arrival time can be easily satisfied. Through carefully designed heuristic functions, the algorithm is guaranteed to efficiently expand the search path along the least cost direction towards the goal.[13]

4.2. Behavior Layer

After receiving the route, the Behavior Layer makes specific behavioral decisions such as choosing to change lanes to overtake or follow.

The main problem in this process is the real-time nature and the uncertainty of the actual driving scene. The real driving scene is a multi-factor environment, including the autonomous vehicle itself, other vehicles on the road, and pedestrians. The behavior of each participant will have an impact on other participants in the environment. This problem is one of the core bottlenecks to truly realize high-level autonomous driving technology.

The traditional approaches to implement a behaviour selector include Finite State Machine Model and Decision Tree Model.

The Finite State Machine Model constructs a graph. Each node in the graph represents a driving state such as accelerating and braking. And each connection describes the transition relationship between driving states. It selects different driving actions based on the current driving state.

The Decision Tree Model defines the driving state and control logic into a tree structure. It searches for driving strategy through a top-down mechanism.

Both models are simple and easy to implement. But they ignore the dynamics and uncertainties of the environment. They are appropriate for simple scenarios. It's hard to be competent for behavioral decision-making tasks in urban road environments with rich structural features.

4.3. Motion planning

Motion Planning takes behaviour as input and transforms them into a specific driving trajectory. Then, it can produce control signals for the vehicles to make the move.

In most cases, if we regardless of altitude changes, the trajectory planning problem of autonomous driving is a three-dimensional constrained optimization problem (a 2D space and time). Therefore, we can spill the original problem into several low-dimensional problems.

The most common practice is Frenet coordinates. Since roads in the real world are curved, Frenet coordinates represent positions on a road more intuitively. As shown

in figure 1 on the right, Frenet coordinates take the vehicle itself as the origin. The coordinate axes are perpendicular. One axe is along the road representing the distance on the road, named Longitudinal, denoted by s. The other is the normal direction to the road representing the distance off the road, named Lateral, denoted by d. Then the driving state is expressed by two variables.

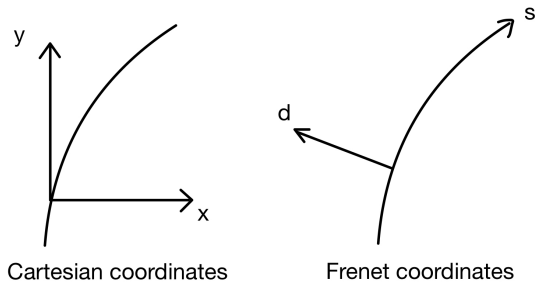

Figure 1: The Cartesian Coordinates and Frenet coordinates

5. Control Execution

The control execution module is to select the optimizing trajectory. It gets the parameter of trajectory created by the Motion Planner subsystem and Obstacle Avoider subsystem, then calculates and sends the effort command to the throttle, steering wheel and brakes to control trajectory.[2]

5.1. Two definitions of trajectory

5.1.1. sequence of commands

$T_c = \{c_1, c_2, \ldots, c_{|T^c|}\}$,each $c_i = (\upsilon_i, \varphi_i, \Delta t_i)$, υ_i is the exception speed, φ_i is the exception angle of steering at time i, Δt_i is the duration of c_i.

5.1.2. sequence of states

$T_s = \{s_1, s_2, \ldots, s_{|T^s|}\}$, each state $s_i = (p_i, t_i)$, p_i is a pose of the car,t_i is the time of the pose.

T_c trajectory control by accepting controller subsystem is always used in direct hardware control method, while accepting a T_s trajectory can be considered as a path tracking method.

5.2. Two kinds of control systems

5.2.1. Direct equipment actuation control methods

Direct equipment actuation control methods are by receiving data from the motion planner subsystem, calculating the force input of the vehicle's steering, throttle, and brake actuators, constantly adjusting the speed υ and steering wheel angle φ to correct errors.

1) Feedback control method
The feedback control method applied efforts to observe the speed υ and the angle of steering φ then adjusted future effort to correct errors. Figure 1 shows how the systems work.The controller block is combined with many controllers, and the system block is combined with many controlled system equipment.

Funke et al. put the feedback control method into the process of creating controller subsystem of Audi. This high-speed hardware can make its controller subsystem to drive the car at speeds of up to 160 km/h.

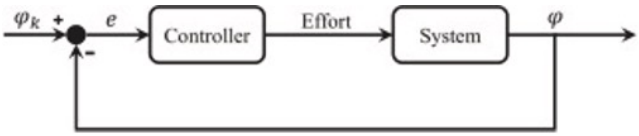

Figure 2: Process of the feedback control

2) Proportional Integral Derivative control method
To assure the trajectory's regular operation, the procedure involves calculating two parameters. One is the equipment input, and the other is the error measure, both of which are used to determine the difference between the equipment output and the needed equipment input. The PID control method uses this information to calculate the equipment's input effort, which is Ip times the error measure (according to which the error is proportional to the error), plus Ii multiplied by the integral of the error, plus Id multiplied by the derivative of the error, where (Ip, Ii, Id) are the PID control method's parameters. The same as the previous control techniques. PID control methods are also focused on the current bias, except past errors that accumulate over time.

5.2.2. Path tracking methods

Path tracking methods can be identified as simplified trajectory planning techniques which reduce the occurrence of model-induced uncertainties by trying to stabilize the execution of motion plans calculated by the motion planning subsystem. It involves finding a point in the path at a certain forward-looking distance from the current path and turning the front wheel so that the arc connects the centre of the rear axis to the point in the path, as shown in Figure 2. The pure tracing method is improved by a set of variants. Samson[14] proposed the use of the rear wheel position as hardware output to stabilise the execution of the motion plan.Teron et al. tried to take the position of the front wheel as the adjusting variable and proposed the control method of autonomous driving vehicle Stanley. Stanley's controller subsystem is capable of driving the car at speeds of up to 60 km/h.

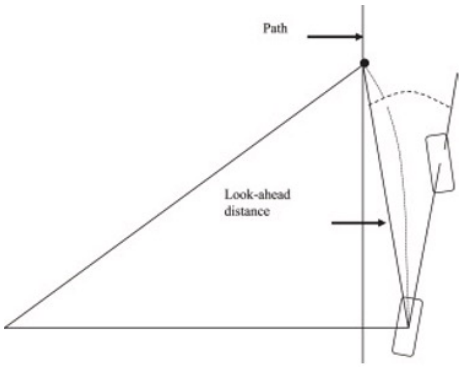

Figure 3: Pure pursuit approach geometry

6. Conclusion

Autonomous driving has great meaning in reducing traffic accidents, alleviating traffic congestion and improving logistics efficiency. People will not stop exploring autonomous driving technology.

In the future, solutions that combine autonomous driving with vehicle control theory, state parameter estimation and other multi-domain methods will become a new development trend. For instance, combined with vehicle dynamics, researchers fully consider vehicle dynamics factors and adopt Model Predictive Control theory. Besides, researchers are adopting the end-to-end model in deep learning to directly generate the control signal of the vehicle based on the vehicle state and external environment information.[15] Although this method is unexplainable as a black box, it has strong development potential due to its outstanding advantages of simplicity and efficiency. In addition, autonomous vehicles making optimal decisions in complex environments fits well with the definition of reinforcement learning. Therefore, the research teams integrate behaviour selector and motion planning modules, using reinforcement learning to learn the optimal policy directly. Here is an example from Mobilieye. In their approach, they train agents to imitate human driver behaviours and plans manoeuvres so that to learn a driving simulation[3].

References

[1] Y. C. J. Z. D. A. Jianjun Ni, Yinan Chen and W. Cao, "A survey on theories and applications for self-driving cars based on deep learning methods," *Journal of Applied Sciences*, vol. 10, no. 8, 2020.

[2] C. Badue *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[3] E. Santana and G. Hotz, "Learning a driving simulator," *arXiv preprint arXiv:1608.01230*, 2016.

[4] H. Cho *et al.*, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1836–1843.

[5] M. Siam *et al.*, "Modnet: Motion and appearance based moving object detection network for autonomous driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2859–2864.

[6] H. Rashed *et al.*, "Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[7] J. Wu *et al.*, "Vehicle detection under adverse weather from roadside lidar data," *Sensors*, vol. 20, no. 12, p. 3433, Jun 2020. [Online]. Available: http://dx.doi.org/10.3390/s20123433

[8] A. M. S. Rahma and N. B. Abd, "Detection of a moving car based on invariant moments." *J. Comput. Sci.*, vol. 14, no. 3, pp. 310–316, 2018.

[9] Z.-Q. Zhao *et al.*, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[10] A. Faisal *et al.*, "Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy," *Journal of Transport and Land Use*, vol. 12, no. 1, pp. 45–72, 2019. [Online]. Available: https://www.jstor.org/stable/26911258

[11] B. Paden *et al.*, "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.

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

[13] 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.

[14] S. Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[15] M. Bojarski *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.