Perception and Planning in Autonomous Car

Technical Report · December 2020

DOI: 10.13140/RG.2.2.31312.38401

5 authors, including:

Shreyas More
Indian Institute of Technology Kharagpur
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Akash Singh
Indian Institute of Technology Kharagpur
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Prithwish Jana
Georgia Institute of Technology
20 PUBLICATIONS **90** CITATIONS
SEE PROFILE

Bithika Pal
Indian Institute of Technology Kharagpur
25 PUBLICATIONS **87** CITATIONS
SEE PROFILE

Perception and Planning in Autonomous Car

Manasvi Sagarkar Roll: 20CS60R28
Shreyas Damodar More Roll: 20CS60R34
Akash Singh Sant Roll: 20CS60R40
Prithwish Jana Roll: 20CS60R66
Bithika Pal Roll: 19CS92R05

{manasvi, shreyas.more, akshshu, jprithwish}@kgpian.iitkgp.ac.in, bithikapal@iitkgp.ac.in
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur

Abstract

Autonomous cars (also known as, self-driving/driver-less cars) are vehicles that flaunts a fully-automated driving system and are believed to be the urban mode-of-transport for the future. Ideally, such cars should be capable of comprehending its environment and navigate amidst dynamic on-road situations, to exactly mimic an efficient human driver. The major sub-systems involved in self-driving cars for their efficient operation are *Perception*, *Planning*, *Control* and *Coordination*. In this report, we elaborate on the Perception and Planning subsystems, and provide salient observations regarding some of the recent developments in these fields. Such autonomous vehicles have a network of sensors and cameras that gather bulks of information about its surrounding. Perception involves intelligently interpreting such raw sensor data into meaningful internal representations so as to gain a clear picture of its environment. A perception subsystem can be further categorized into two main functionalities viz., detection and segmentation. While the detection block is responsible for identifying objects like pedestrians, nearby cars etc. and classifying traffic signals, the segmentation block identifies chunks of semantically meaningful segments e.g. asphalt roads, footpaths. Together, they enable the vehicle to delineate drivable areas alongside avoiding potential obstacles. Planning subsystem involves route, behaviour and motion planning. While route planning is responsible for computing an optimal global source-terminal route, behaviour planning takes into account smoothness of roads, velocity profiles and real-time traffic information to choose a safe executable behaviour of the car. Motion planning decides on the sequence of actions needed to reach from start to goal without collisions. In combinatorial motion planning, the continuous planning problem is discretized without any approximations. In sampling-based motion planning, the path from start to goal is built by generating random samples in the configuration space. Based on the computed trajectory, the controllers steer the car to the next configuration and pass the feedback to the planner. In this report, we study the broader view of perception and planning in autonomous vehicles (AV) and elaborate upon the technologies in use and the recent breakthroughs.

Index Terms

Autonomous Vehicle (AV), Self-Drive car, Perception, LiDAR, R-CNN, Route Planning, Behaviour Planning, Motion Planning, Control

Publicly-accessible links of presentation video:
https://youtu.be/4d_Y27HOWEA
https://drive.google.com/file/d/1U5FbuTdXsAKe7fhEl6Fq0qzqLxUaZ243/view?usp=sharing

I. Introduction and Background

AUTONOMOUS cars, also known as driver-less cars, are vehicles that can sense its environment and operate without human input. In autonomous driving, there is no need of a human driver. It can navigate similarly as any a normal car would do, but without a human at the wheel. Thus, such vehicles are expected to perform all the functionalities of an experienced driver, and thereby mimic them. In contrary to today's traditional cars, self-drive cars offer great range of advantages viz., (i) potential for additional safety, (ii) better productivity, (iii) better road efficiency (iv) minimal negative impact on environment. Thus, these are believed to play a major role in determining how urban transportation systems will become in the future.

Now, for a self-driving systems to be considered as a successful innovation deployable in real-time operating systems, it should be able to take correct decisions in dynamic environments i.e. it requires artificial intelligence. If the concept of self-driving and its on-road implementation proves to be successful, there will be many positive implications for society and environment. In the recent years, as more and more research are being conducted in this field, design and manufacture sectors of autonomous cars are experiencing some major developments. Alongside, research and certain studies are also being conducted to model and anticipate the social impact of implementing Autonomous Vehicles. Such studies have shown that autonomous vehicle system would make mobility more secure, and thereby access to mobility will become more affordable and convenient compared to traditional vehicle systems. If we talk about autonomous driving on urban roads, it has seen significant progress in last couple of years. Few of them are enlisted below:

- BMW announced a partnership with German automotive giant, Daimler (famously known as, Mercedes-Benz). The alliance is ambitious to create an open-standards based platform for bringing self-driving cars to the market. They have plans to put its first vehicle, the BMW iNEXT, on the road by 2021
- Daimler and Karlsruhe Institute of Technology (KIT), recreated the first cross-country automobile journey [1] in history, in an autonomous way
- Volvo Cars has reached an agreement with Luminar (whose plug-ins are famous for use in Apple and Adobe products) to enable their LiDAR and perception technology to be fitted onto Volvo cars from 2022. These are expected to include sensing technology integrated into the roof of the vehicle
- Uber plans to launch its self-driving cars in pockets of cities where weather, demand and other conditions are most favorable
- Sophisticated GPR (Ground Penetrating Radar) sensing systems, which has been proven to be capable of analysing road subsurface conditions in all types of weather, are currently being used in military and being evaluated

The typical autonomous driving system can be decomposed [2] into roughly six levels: *Sensing*, *Perception*, *Prediction*, *Planning*, *Control* and *Coordination*. In the following paragraphs, we provide an introductory explanation on three of these technologies and further, elaborate upon them in the subsequent sections.

A. Perception

The subsystem of Perception caters to understanding the world surrounding the vehicle. By virtue of perception, autonomous vehicles collect information and thereby, extract and exploit required knowledge from the environment. It is oftentimes regarded as the first stage in the computational pipeline for the safe functioning of a self-driving car. The objective of perception subsystem is similar to that of Cyber-Physical Systems (CPS) [3]. They take inputs from their surroundings through various capturing devices (sensors, LiDAR, RGB-D cameras, GPS, etc.) and extract useful information from these bulks of data through efficient back-end systems (e.g. object detection, semantic segmentation, location, etc.).

B. Planning

Planning is the process of making decisions in order to achieve the robot's higher order goals,In autonomous cars this goal normally refers to bring the car from one location to another location in an optimal and secure manner, Planning can be categorised into three levels:

a) Route Planning:	It is concerned with finding the optimal means of travelling from one location to another. For Route Planning, algorithms like A* and Dijkstra are used.
b) Behavior Planning:	A behavior planning system aims to safely achieve the given driving mission under various driving situations using the set of high level driving actions.
c) Motion Planning:	Motion planning is a computational problem that aims to find a sequence of valid configurations that moves the object from one location to another.

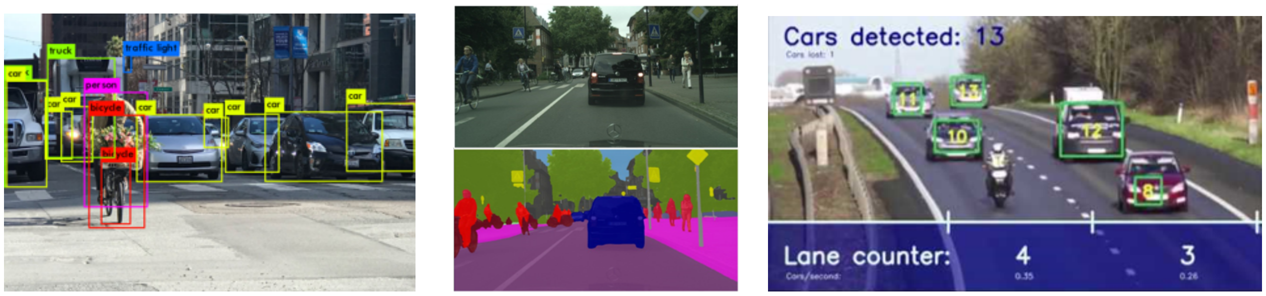
C. Control

A control component's main purpose is to execute the actions that were planned by giving required inputs to the hardware level that in turn will be responsible for generating the required motion. A control module actuates throttle, braking and steering to track the trajectory produced by planning. Controllers map the interaction with the real world in terms of scientific terms like forces, and energy understandable by the vehicle. The simplest and most commonly used controller is the proportional integral derivative (PID) controller, which is used to track a reference velocity in the case of cruise control. One of the downsides is that it requires extensive tuning, as the controller parameters configurations depends on velocity, surface and steering conditions. However it is effective and commercially used in many systems, including autonomous driving systems.

II. Perception in Autonomous Driving

We, humans, can easily analyze and perceive our surrounding, owing to the priceless gifts of sensation, vision and cognition. These help a human to perform a variety of activities ranging from the simplest tasks like reading a book, to complex multi-tasking like flying an aeroplane. Driving requires such similar multi-tasking skills where the driver has to continuously perceive bulks of information around him and simultaneously have to steer through a street without meeting an accident or committing a traffic penalty. Even few decades back, expecting such skills from a computer (here, an autonomous car) was a far-thought idea, but is now on the way of becoming a reality through the advent of autonomous cars. Similar to how the 'ability to perceive' enables humans to react to external stimuli instantaneously, the power of accurate perception can help autonomous cars to continuously track movements of surrounding objects and thereby, avoid collision.

Detection of forthcoming obstacles and object-tracking are two important aspects [4] of perception in autonomous vehicles. These form the basis for the subsequent path-planning and decision making steps of autonomous vehicle. There are quite a few approaches that enables an autonomous car to perceive objects and obstacles surrounding it. As for example, to get a real-time perception of the circumambient environment, autonomous cars use LiDAR, Global Positioning System (GPS), RGB+Depth (RGB+D) cameras, night-vision cameras and various other sensors. In the subsequent paragraphs, we explain one of the most widely-used technology i.e. LiDAR. Figure 1 illustrates three of the important tasks involved in a perception system for autonomous cars viz., object detection, semantic segmentation and object tracking. We elaborate on these in the subsequent paragraphs.



(a) Object Detection (b) Semantic Segmentation (c) Object Tracking

Fig. 1: Major Tasks involved for Perception in Autonomous Cars (Source: Redmon et al. [5], Kundu et al. [6] and Sturdevant [7] respectively)

A. Light Detection and Ranging (LiDAR): A Technology used for Perception

LiDAR is considered to be the 'eyes' of an autonomous car. Previously it was placed as a rotating object (as in Figure 2a) on the roof of a car. With advancement of technology, its size has shrunken manifold to the size of a small chip (as in Figure 2b) and is mounted on the dashboard or bonnet of modern-day autonomous cars. Effectively what this device does, is to provide an all-around vision of the surrounding thus helping the controller of the car to pursue instantaneous and long-term actions.

A autonomous car has to dynamically detect objects thus giving it enough information about its size, shape and position. As an effect, its control-algorithm can effectively plot the safest path through them. A LiDAR is a special kind of laser-based probe. A similar technology is used in RADAR antennas that emits regular pulses of radio or microwave at aeroplanes. As such, they can perceive their location based on the time taken by the wave to bounce back from the body of the aeroplane and reach back. On contrary to the larger wavelength signals used in RADAR, LiDAR of autonomous cars used narrow beam of infrared signals that can image features corresponding to even the smallest objects on the street. With the help of these short successive laser beams, the autonomous car perceives the depth resolution of objects around it.

The outputs created by LiDAR capturing devices, that catches reflected LiDAR waves, are typically in the form of *point clouds*. Example of point-cloud image generated at a cross-road is provided in Figure 2c. In fact, most of the methods that employ some kind of remote-sensing or photogrammetry, produce similar point clouds. These point clouds are a combination of images perceived at multiple angles and at multiple time-instances to form point clouds. Along with these point-cloud images, the x/y/z coordinates of an obstacle (that reflected wave) is computed using the position and angular orientation of the emitter, the angle of the receiver, and the distance amongst emitter, reflector and receiver. It may be noted here that, although emitter and receiver are mounted on the same car but they have positional differences as in the time a wave was emitted and captured, the car had moved forward by a distance.

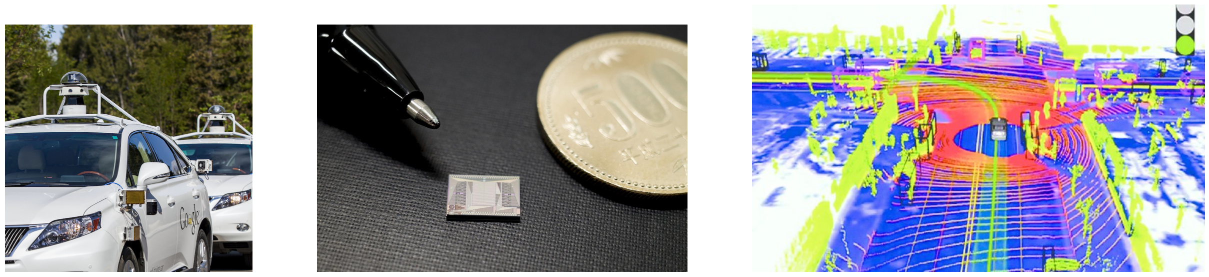

(a) LiDAR atop car (b) LiDAR chip & sensor (c) Point-cloud Image

Fig. 2: (a-b) Comparison of LiDAR devices used previously and the chips used nowadays (c) Point cloud image of a car approaching a cross-road (Source: Port [8], Tsumura [9] and Seif et al. [10] respectively)

B. Object Detection & Recognition

The aim of *Object detection* is to localize an object of a certain class in a scene. Such tasks also include detection of faces and thereby, pedestrians. Algorithms of face detection typically use either Haar-like features [11] or AdaBoost, or a combination of these two. More specialized pedestrian detection system involves Histogram-of-Gradients (HoG) features [12] and Support Vector Machine (SVM) to subsequently classify them. Also, with the rise and popularity of deep machine learning, nowadays object detection is put forward as a binary classification problem, that involves scanning a visual image through successions of convolutions and pooling. Sophisticated systems require objects to be classified not only into two class but in multiple categories. This is because, there may also arise specific difficult scenarios, where a self-drive car requires an in-depth grasp on the surrounding environment. This is usually achieved through a single deep neural network or or multimodality-fused networks. It may not always suffice to just detect the presence of an obstacle, recognition is also important. For example, a pre-informed idea about the position of pedestrians and bicyclists may help the autonomous car to make some intelligent precedence decisions and also predict their future trajectories.

Now, object detection is a harder feat to achieve that typical classification problems. This is because, the former requires to draw a bounding box around the specific objects of interest to locate it within a scene. Moreover, it may not be sufficient to draw only a single bounding-box because there may be multiple objects of interest within a scene. What is more problematic is that, the mutual position and counts of such objects are not known beforehand. And this is why, standard Convolutional Neural Networks (CNN) cannot be used to achieve object detection. The number of objects is always variable and it cannot be tackled by the fixed-size output of a fully-connected layer that typically follows a CNN. To solve this, Girshick et al. [13] proposed Region-based Convolutional Neural Networks (R-CNN) that performs a selective search to obtain region proposals from an image. After being warped into square shape, such region proposals were fed into a CNN that serves as a feature extractor. Thereby, the last fully-connected dense layer of the CNN now contains features of there region proposals. Further, they were fed to a SVM that classified the object in each region proposal into one of several categories. Also, there is an option for certain offset that permits correction in the region proposals when certain portions of objects have been detached from other parts of it. But, R-CNN has a major disadvantage in being not suitable for real-time systems for autonomous cars. To solve this issue, Girshick et al. [14] proposed Fast R-CNN that relied upon the use of convolutional feature maps. So now, instead of feeding each and every region proposal to a CNN, the whole image was fed along with the corresponding feature maps for the various region proposals. Another significant breakthrough in this domain was the work on 'You Only Look Once' (YOLO) model by Redmon et al. [5]. In this framework, an input image is split into several square grids and bounding-boxes are considered within each such grid. Thereafter, the YOLO framework predicts a class label and offset value for each such bounding box and the topmost choice of bounding-box is selected from a grid. Although YOLO suffers from detection of minute objects, it is significantly faster than R-CNN and Fast R-CNN.

C. *Segmentation & Object Tracking*

By virtue of *Semantic Segmentation*, a class is assigned to each and every pixel of an image and the latter is thereby associated with a semantically relevant object label. From the perspective of a self-driving vehicle, these classes includes but are not limited to "vehicles", "pedestrians", "animals", "trees", "sky", etc. Delineating such areas within an image helps in pin-pointing the driveable areas and *mapping* visual scenes.

Object tracking involves the task of identifying moving objects and track them through their navigation trajectory. For such task of localization and mapping of objects, Simultaneous Localization And Mapping (SLAM) [15] empowers autonomous cars to construct map of unknown surroundings and localize itself in the surrounding, all at once. One of the disadvantages of SLAM is that it assumes surrounding to be static. Here, Detection and Tracking of Moving Objects (DATMO) [16] served as one of the major breakthrough in putting up with the deficiencies of SLAM. A combination of SLAM and DATMO serves as an excellent pair to help mimic humanoid vision in autonomous cars.

III. Planning in Autonomous Driving

The planning is an important component responsible for considering the current state and all other possible states in the future and based upon this information, chart out a safe, secure and feasible path. This activity can be considered to be done in three stages namely the mission planner or high level route planner, behavioural planner or decision maker that interacts with other agents to ensure that rules are followed and finally motion planner that performs a set of actions to fulfil the local objectives. Each of the planner will be explained in detail in the sections that follow.

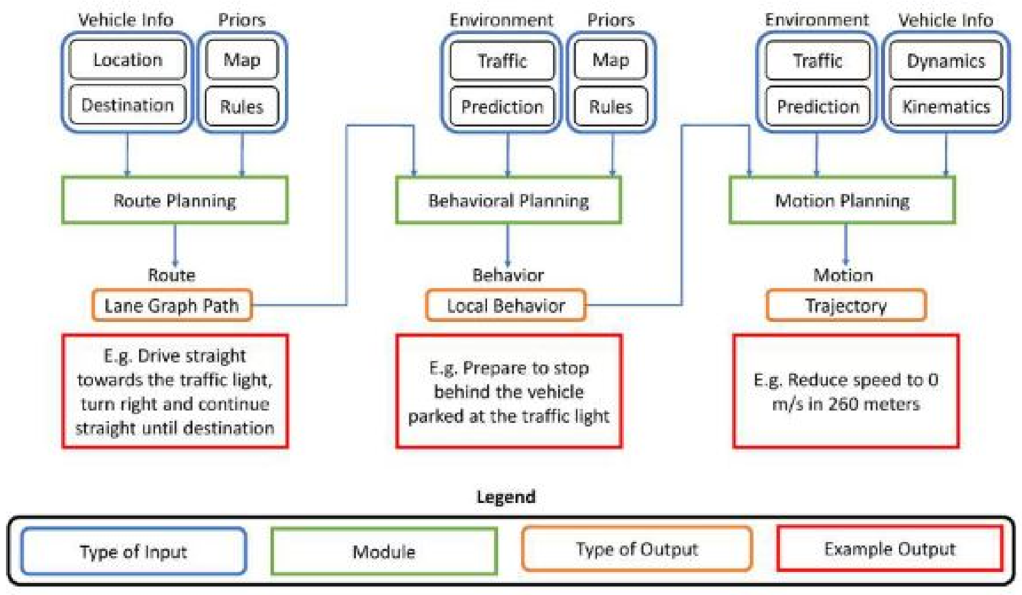


Fig. 3: Hierarchical structure of planners. The rounded boxes at the top represent various modalities of input, while the rounded boxes in the middle represent different outputs. The boxes at the bottom serve as example outputs that feed into downstream layers. (Source: Fang et al. [17])

A. Route Planning

Route planning is performed on a directed graph in which the start and finish vertices are the starting and destination locations and the road/paths between the vertices are given by the directed edges. This forms a weighted graph and thus graph search algorithms can be employed for finding minimum cost paths. Some of the popular algorithms for route planning include A* [18], Dijkstra's [19] and bidirectional search. These algorithms work really well for small graphs however they cannot be scaled for graphs with millions of nodes which is required in real life applications. This lead to research [20] in improving the pre-processing step to facilitate faster search time later on.

B. Behavioral Planning

Behavioral planning uses the route plan and produces a high-level behavior for the vehicle to follow. It makes ad hoc decisions to properly interact with other agents and follow rules restrictions, and thereby generates local objectives. For example, if the vehicle is in the left-most lane and needs to make a right turn, an appropriate behavior would be: prepare to change lane right. This may be realized through a combination of goal setting, virtual obstacle placement, adjustment of driveable region bounds, and/or regional heuristic cost adjustment. Decisions were made onboard most DUC vehicles through Finite State Machines (FSMs) of varying complexity to dictate actions in response to specific perceived driving contexts [21–24]. The terms *precedence observer* and *clearance observer* were coined to categorize functions which checked certain logical conditions required for state transitions, where precedence observers were to check whether the rules pertaining to the vehicle's current location would allow for it to progress, and clearance observers would check "time to collision", the shortest time by which a detected obstacle would enter a designated region of interest to ensure safe clearance to other traffic participants.

Finite state machines of this nature are limited in that they are manually designed for a set number of specific situations. The vehicle may then perform unexpectedly in a situation that was not explicitly accounted for in the FSM structure, perhaps finding itself in a livelock, or even a deadlock state if there aren't sufficient deadlock protections. Recent research works have sought to improve organization in large decision making structures to thus manage larger rules sets [25–27].

C. Motion Planning

Motion Planning refers to deciding on the sequence of actions needed to reach a specified goal without colliding with obstacles. Motion planners are evaluated based on two criteria. Firstly, they are evaluated based on their run time and how well they scale with higher dimensional data. Secondly, they are evaluated on their *completeness*. The planner is considered *complete* if it always terminates in finite time and always returns a solution when one exists. If no solution exists, it indicates that no solution exists. The search space for motion planning is very large which leads to high computational complexity of the planner. Further, we may need an exhaustive search of all possible paths in order to guarantee *completeness*. One solution to this complexity is to transform our continuous model of space into a discrete model. We will discuss two broad approaches to carrying out this transformation - Combinatorial planning and Sampling based planning.

1) Combinatorial Planning: Combinatorial planners find paths through the continuous state space without making any approximations. They are also called exact algorithms due to this. In Combinatorial planners, we build discrete models that exactly represent the original problem. These planners are complete and have a better performance compared to the sampling based planners we will discuss below. Since they do not approximate,they are mainly useful for special, constrained classes of planning problems. For instance, we can generate geometric solutions efficiently for low dimensional spaces with discrete convex obstacle spaces by constructing visibility graphs. However as the number of obstacles and the dimensionality of the space increases, combinatorial methods become less feasible.They are also useful for providing theoretical upperbounds on time needed to solve motion planning algorithms.

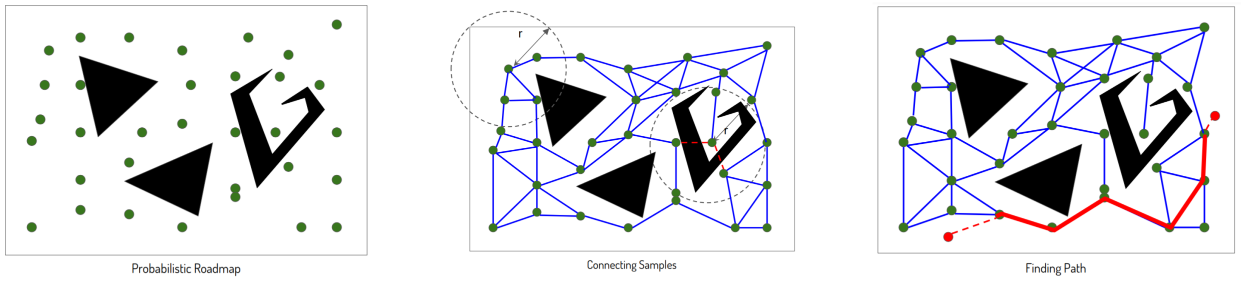
Almost all combinatorial planning algorithms construct roadmaps to solve queries.A roadmap is a graph that represents the free space. A roadmap provides a discrete representation of the continuous motion planning problem while retaining the original connectivity information. The planner then finds the path using the roadmap and the problem reduces to graph search. Some common methods for creating roadmaps are Visibility graphs and Voronoi

diagrams. Other than creating roadmaps, some combinatorial planners also use the cell decomposition method. In cell decomposition, the space is decomposed into cells and the outcome of the search is a sequence of cells from start to finish. However cell decomposition methods may generate many infeasible solutions and generally suffer from combinatorial explosion which is the a rapid increase in the complexity of the problem. These drawbacks are absent from roadmap based methods.
By working in a discretized space the complexity of exhaustive search is greatly reduced. We can no longer guarantee completeness but as long as we consider a fine enough ”resolution”, completeness can be guaranteed. Sufficiently fine resolutions are however hard to achieve in high dimensional spaces. Despite this discretization of space is still widely used.

2) Sampling Based Planning: In contrast to combinatorial planning, sampling based planning algorithms works in *continuous space* and guarantees *Probabilistic Completeness*, i.e., planner will eventually find the solution if exists, using random sampling (e.g. Monte Carlo sampling). Here, the generation of a *feasible trajectory graph* is performed through collision checking of nodes and edges. Based on different ways of the search tree generation, two most influential categories of sampling-based methods are Probabilistic RoadMaps (PRM) [28–30] and Rapidly-exploring Random Trees (RRT) [31], [32].

- *Probabilistic RoadMaps (PRM):* The basic motion planning problem asks for computing a collision-free feasible path for an object (or kinematic device); here it is a car, from a given start to a given goal placement in a workspace with obstacles. In general PRM works with static obstacles [33], [34]. Recent developments of PRM extend it for dynamic environments with moving obstacles, as well [35]. PRM constructs a graph-structure (roadmap) of the free configuration space. It can be used to find multiple paths/queries from start to goal. The biggest advantage of using PRM is that it retains the graph structure and reuses it every time you select two points. Sometimes, it is also called multi-query planner. PRM planner works in two concurrent phases, namely *RoadMap Construction* and *Path Creation*.
  - *RoadMap Construction Phase:* The nodes in the roadmap are collision-free configurations and the edges are collision-free paths. The roadmap is built by repeating the following two steps:
    - A configuration is selected randomly from the configuration space of the autonomous car. Then, it is checked for collision and repeated this step until the selected configuration is collision-free. Here, sampling the free configuration space can be done either deterministically using grids, or purely at random.
    - The generated sample is connected to the former configuration in the roadmap by a fast local planner, if the edge is not passes through any obstacles. The connection is made if the nodes are within some distance r, or by connecting k-nearest neighbors for every node.
  - *Path Creation Phase:* To find a path between an initial and goal configuration, this step attempts to connect these configurations to the roadmap and searches in the roadmap for a sequence of local paths linking them. Graph search algorithm using Dijkstra or A*, we obtain a feasible path for the vehicle. The path for self-driving cars actually consists of path segments and to reach from one configuration to other, the car needs to follow the possible templates of path segments. For example, in Dubin’s path, there are three path segments: sharpest-possible turn left (L), right (R) or straight (S), which results in total six templates:$\{LRL, RLR, LSL, LSR, RSL, RSR\}$ [36]. In this way the car moves forward. In autonomous driving, Sampling-based methods rely on *steering function*. Steer(x,y) returns a feasible path segment between configuration x and y. Steering function respects kinematic and dynamic constraints, but does not consider obstacles. These are often obtained by simulating a dynamic model of the vehicle. Among all the paths, in the roadmap, connecting start to goal configuration, the graph algorihm returns the shortest path. However it is not the optimal one, for this case it will be needed an optimizing planner.

  One example of PRM planning is shown in Figure 4. Now, in PRM planner the concern point is how to choose the connecting radius r. If the r is small the roadmap will be disconnected. Again for the large value of r, many paths will be created, which will become computationally expensive. To resolve this, Sertac Karaman and Emilio Frazzoli proposes *PRM**, which are provably asymptotically optimal, i.e., such that the cost of the returned solution converges almost to the actual optimum. It also chooses the connection radius as a function of the number of samples of graph as $r = \gamma^d \sqrt{\log n/n}$ [37]. It attempts to create $O(\log n)$ connections at each iteration, and maintains asymptotic optimality with $O(n \log n)$ complexity.



(a) Probabilistic RoadMap (b) Connecting Samples (c) Finiding Path

Fig. 4: One Example of Sampling-based Roadmap Construction

- *Rapidly-exploring Random Trees (RRT):* Compared to Combinatorial planning and PRM, RRT works in incremental fashion, and also able to work in high-dimensional spaces. A typical RRT algorithm starts with an initial configuration as root of the tree, and grows the tree rooted by exploring the reachable region of the configuration space. Once first branch reaches goal region, return the branch as the first solution and Keep reporting the shortest branch found so far. For nonholonomic moving objects like car and to handle more obstacles, RRTs are very useful motion planning algorithms, as it does need exact steering. Though RRTs are single query algorithm, they can be considered as a technique for generating open-loop trajectories for non-linear systems which involve more system specific constraints, due to it's less expensive path search than PRM. [32].
The ground of RRT is simple and straight forward. Starting from an initial configuration, the samples are randomly generated from the configuration space and checked if it is in the obstacle free space or not. Then, the node is connected to the closest available node if the connecting edge avoids any collision. The algorithm ends when a node is generated within the goal region. Once the first branch reaches to the goal, then step is repeated again to improve the solution further, and get a shorter path. Additionally, the method of chaining the randomly generated node can be customized. One example method can be calculating the vector that forms the shortest distance between the new vertex and the closest edge. At the point of intersection, a new node is added to the edge and connected to the randomly generated vertex. The graph generated by RRT is like *cubic graphs*, as the nodes are attached to their nearest neighbors. Due the structural nature of these graphs, it reduces the probability of finding an optimal path. This problem of RRT is addressed in RRT*, with the cost of computational burden.
Optimal Rapidly-exploring Random Trees (RRT)* is an optimized version of RRT, which is asymptotically optimal [37]. The basic principle of RRT* is similar to RRT. Starting from the initial node, random samples are generated in the obstacle-free space and connected to the neighbors within distance r avoiding collision, in a tree fashion. However, there are two key additions in comparison to RRT. First, RRT* records the distance, which each node has traveled relative to its parent node as a `cost()` of the node. After the closest node is found in the graph, a neighborhood of nodes in a fixed radius from the new node are examined for the cost updating. If a node with a smaller `cost()` than the proximal node is found, the new edge created to the node with smaller travel cost. Due to this reason, instead of cubic struture of RRT-generated graph, *fan shaped twigs* are added in the tree structure of RRT*. The second difference in RRT* is the rewiring of the tree. After a node has been connected to the neighbor with smaller cost, the neighbors are again examined to remove the existing edge if smaller cost path is captured. This is called the rewiring of the graph, which reduces the cost of the tree and directs towards optimality.
In comparison ofRRT, RRT* creates incredibly straight paths. Additionally, graphs of RRT* are characteristically different from those of RRT (Shown in Figure 5). For finding an optimal path, especially in a dense field of obstacles, the structure of RRT* is incredibly useful. The graph crawls around objects or obstacles, finding shorter paths in comparison to RRT. Due to examining neighboring nodes and rewiring the graph, RRT* suffers from a reduction in performance. Compared to PRM, both RRT and RRT* are single query algorithms.

All the sampling based methods have major dependence how the sampling of a random configuration is generated,

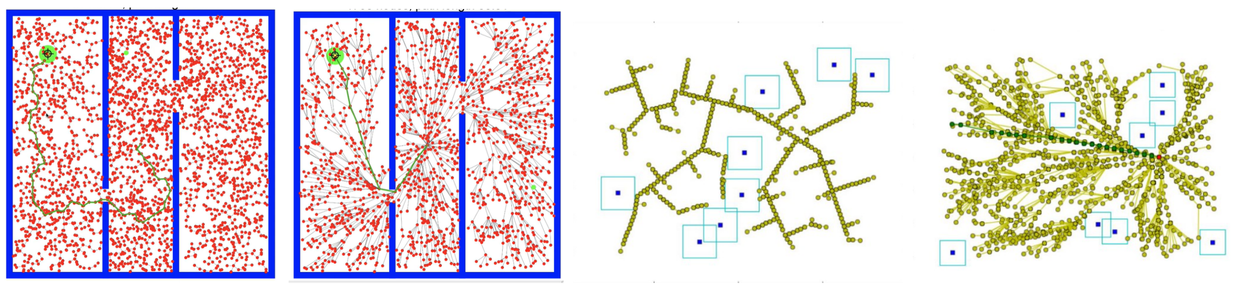

(a) (b) (c) (d)

Fig. 5: Examples showing graph construction using RRT (a,c) and RRT* (b,d) for different type of obstacles

based on different types of obstacles [33]. Other than, random and grid, which have been discussed earlier, there are other popular methods like, random halton points (good coverage of a region with points), cell-based, Gaussian (to add more samples near obstacles), obstacle-based.

Other than combinatorial and sampling based planning, there are optimizing planners which optimize different goals in autonomous driving, like minimizing shortest path, maximizing the curvature, keeping safest distance from the obstacles, etc. Planner also handles the dynamic behaviour of environment. Based on the trajectory as it gets in every time step, the planner instructs the controllers to follow the rule of kinematics and direct the handlers like brake, accelerator, to reach into the new state in the work-space. In this process it also communicates with the environment and other feedback controllers to decide the next action and completes the work-cycle of autonomous car.

IV. Conclusion

For the past few years, autonomous vehicles (AV) have been the common buzzword and extensive collaborative research are being carried out by academia and industry. As compared to traditional human-driven cars, AVs are expected to do away with human errors thereby proferring advantages like, improved on-road safety, less traffic congestions and lower rates of pollution. And to achieve such precision and safety standards, an autonomous decision-making backbone software is essential that makes use of artificial intelligence to interpret dynamic on-road situations. In this report, we provide some key observations in the perception and planning subsystems of such software backbones. Perception is a fundamental function that is performed first in the overall pipeline of such autonomous systems. Here, we first provide details of data-capturing technologies like LiDAR that capture bulks of sensor data. Further, we had an overview of some recent computer vision algorithms for object detection (R-CNN, YOLO) and semantic segmentation utilizing V-SLAM and DATMO that are responsible for extraction and exploitation of essential information related to the driving environment. As a result, the car identifies drivable areas and precise location of each of its surrounding obstacles which enables it to even approximate the latter's future trajectories and navigate itself in a safe path of its own. In motion planning we looked at two approaches. Combinatorial planners that do not approximate the original problem and sampling based planners that use approximations. In Combinatorial planning we looked at two approaches – roadmaps and cell decomposition. We also looked at the drawbacks of Combinatorial planners and the need for sampling-based planning. In sampling-based planning, the path from start to goal is built by generating random samples in the configuration space. Then, the planner passes the control to the physical handlers of the vehicle to steer at certain configuration. Finally, the controllers provide the feedback about the next configuration, and the environment is perceived again. In real-life, the configuration space is very high-dimensional, and planner needs to compute all the trajectory in real-time, which makes the problem more challenging.

References

[1] 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.

[2] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, Planning, Control, and Coordination for Autonomous Vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[3] S. Bhaumik, P. Jana, and P. P. Mohanta, "Event and Activity Recognition in Video Surveillance for Cyber-Physical Systems," *Emergence of Cyber Physical System and IoT in Smart Automation and Robotics*, ch. 4, Springer Nature. 2020.

[4] D. Xie, Y. Xu, and R. Wang, "Obstacle Detection and Tracking Method for Autonomous Vehicle based on Three-Dimensional LiDAR," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, 2019.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[6] A. Kundu, V. Vineet, and V. Koltun, "Feature Space Optimization for Semantic Video Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3168–3175.

[7] M. Sturdevant. Object Tracking in Video with OpenCV and Deep Learning. (accessed: November, 2020). [Online]. Available: https://www.youtube.com/watch?v=19vaot75JCY

[8] J. Port. LIDAR: How Self-Driving Cars 'See' Where They're Going. (accessed: November, 2020). [Online]. Available: https://cosmosmagazine.com/technology/how-self-driving-cars-can-see-where-they-re-going/

[9] A. Tsumura, "Slow Light to Speed up LiDAR Sensors Development," *Yokohama National University*, (accessed: November, 2020). [Online]. Available: https://www.eurekalert.org/pub_releases/2020-01/ynu-slt011220.php

[10] H. G. Seif and X. Hu, "Autonomous Driving in the iCity—HD Maps as a Key Challenge of the Automotive Industry," *Engineering*, vol. 2, no. 2, pp. 159–162, 2016.

[11] G. I. Hapsari, G. A. Mutiara, and H. Tarigan, "Face Recognition Smart Cane using HAAR-like Features and Eigenfaces," *Telkomnika*, vol. 17, no. 2, 2019.

[12] K. W. Chee and S. S. Teoh, "Pedestrian Detection in Visual Images using Combination of HOG and HOM Features," in *10th International Conference on Robotics, Vision, Signal Processing and Power Applications*. Springer, 2019, pp. 591–597.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[14] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[15] D. Gálvez-López, M. Salas, J. D. Tardós, and J. Montiel, "Real-Time Monocular Object SLAM," *Robotics and Autonomous Systems*, vol. 75, pp. 435–449, 2016.

[16] K. Na, J. Byun, M. Roh, and B. Seo, "RoadPlot-DATMO: Moving Object Tracking and Track Fusion System using Multiple Sensors," in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2015, pp. 142–143.

[17] E. Fang, "Dynamic Deadlines in Motion Planning for Autonomous Driving Systems," *UC Berkeley*, 2020.

[18] N. J. Nilsson, "A Mobile Automaton: An Application of Artificial Intelligence Techniques," Sri International Menlo Park CA Artificial Intelligence Center, Tech. Rep., 1969.

[19] E. W. Dijkstra *et al.*, "A Note on Two Problems in Connetion with Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[20] 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.

[21] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[22] 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.

[23] 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," in *The DARPA Urban Challenge*. Springer, 2009, pp. 231–255.

[24] C. R. Baker and J. M. Dolan, "Traffic interaction in the urban challenge: Putting boss on its best behavior," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1752–1758.

[25] A. Broggi, S. Debattisti, M. Panciroli, and P. P. Porta, "Moving from analog to digital driving," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 1113–1118.

[26] A. Furda and L. Vlacic, "Towards increased road safety: Real-time decision making for driverless city vehicles," in *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2009, pp. 2421–2426.

[27] ——, "Enabling Safe Autonomous Driving in Real-world City Traffic using Multiple Criteria Decision Making," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 4–17, 2011.

[28] L. Kavraki and J.-C. Latombe, "Randomized Preprocessing of Configuration for Fast Path Planning," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 2138–2145.

[29] 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.

[30] J.-M. Lien and Y. Lu, "Planning Motion in Environments with Similar Obstacles," in *Robotics: Science and Systems*, 2009.

[31] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," 1998.

[32] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[33] R. Geraerts and M. H. Overmars, "A Comparative Study of Probabilistic Roadmap Planners," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 43–57.

[34] C. Martínez Alandes, "A Comparison among Different Sampling-based Planning Techniques," 2015.

[35] L. Jaillet and T. Siméon, "A PRM-based Motion Planner for Dynamically Changing Environments," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. IEEE, 2004, pp. 1606–1611.

[36] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[37] 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.