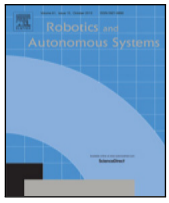
Robotics and Autonomous Systems 174 (2024) 104630

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Path planning algorithms in the autonomous driving system: A comprehensive review

Mohamed Reda [a,b,*,1], Ahmed Onsy [a,2], Ali Ghanbari [a], Amira Y. Haikal [b,3]

[a] School of Engineering, University of Central Lancashire, Preston, PR1 2HE, UK
[b] Computers and Control Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, 35516, Egypt

ARTICLE INFO

Keywords:
Autonomous driving system
Path planning
Driverless cars
Local planning
Global planning

ABSTRACT

This comprehensive review focuses on the Autonomous Driving System (ADS), which aims to reduce human errors that are the reason for about 95% of car accidents. The ADS consists of six stages: sensors, perception, localization, assessment, path planning, and control. We explain the main state-of-the-art techniques used in each stage, analyzing 275 papers, with 162 specifically on path planning due to its complexity, NP-hard optimization nature, and pivotal role in ADS. This paper categorizes path planning techniques into three primary groups: traditional (graph-based, sampling-based, gradient-based, optimization-based, interpolation curve algorithms), machine and deep learning, and meta-heuristic optimization, detailing their advantages and drawbacks. Findings show that meta-heuristic optimization methods, representing 23% of our study, are preferred for being general problem solvers capable of handling complex problems. In addition, they have faster convergence and reduced risk of local minima. Machine and deep learning techniques, accounting for 25%, are favored for their learning capabilities and fast responses to known scenarios. The trend towards hybrid algorithms (27%) combines various methods, merging each algorithm's benefits and overcoming the other's drawbacks. Moreover, adaptive parameter tuning is crucial to enhance efficiency, applicability, and balancing the search capability. This review sheds light on the future of path planning in autonomous driving systems, helping to tackle current challenges and unlock the full capabilities of autonomous vehicles.

1. Introduction

1.1. Background of ADS

The history of autonomous driving systems (ADS) dates back nearly a hundred years. The early concepts of self-driving cars have evolved into today's sophisticated ADS due to technological advancements. Significant developments in computing, artificial intelligence, and sensor technology have marked this evolution. Initially, the progress in ADS could have been faster, constrained by the limitations of early technology. However, the last few decades have seen a rapid transformation in this field, bringing us closer to the goal of fully autonomous vehicles [1].

One of the critical turning points in the history of ADS was the DARPA Challenges in the mid-2000s. These challenges encouraged the collaboration of experts from diverse backgrounds to address complex problems in vehicle automation. This event significantly boosted interest and investment in autonomous vehicle technology. Since then, there has been a surge in research and development in ADS, with many companies and academic institutions dedicating resources to this area [2].

Modern autonomous vehicles use a combination of technologies like radar, GPS, cameras, and lidar to navigate their surroundings safely. These technologies allow vehicles to perceive and interact with their environment, making real-time decisions essential for autonomous operation. Over the years, the improvements in these technologies have made autonomous cars more reliable and safer [3].

Autonomous vehicles are now seen as a transformative innovation in transportation, potentially greatly enhancing road safety and efficiency. While fully autonomous vehicles are still being developed, the advancements made thus far point to a future where vehicles can operate independently, shaping the future of transportation [4].

Around 95% of car accidents are because of human errors based on a recent report by Kent County Council in the UK [5]. Therefore, intensive research is behind the improvement of the Automated Driving

* Corresponding author at: School of Engineering, University of Central Lancashire, Preston, PR1 2HE, UK.
E-mail addresses: mohamed.reda.mu@gmail.com, mohamed.reda@mans.edu.eg, mramohamed@uclan.ac.uk (M. Reda).
[1] https://www.scopus.com/authid/detail.uri?authorId=57220204540
[2] https://www.scopus.com/authid/detail.uri?authorId=36110045700
[3] https://www.scopus.com/authid/detail.uri?authorId=37022897600

https://doi.org/10.1016/j.robot.2024.104630
Received 13 March 2023; Received in revised form 7 December 2023; Accepted 2 January 2024
Available online 12 January 2024

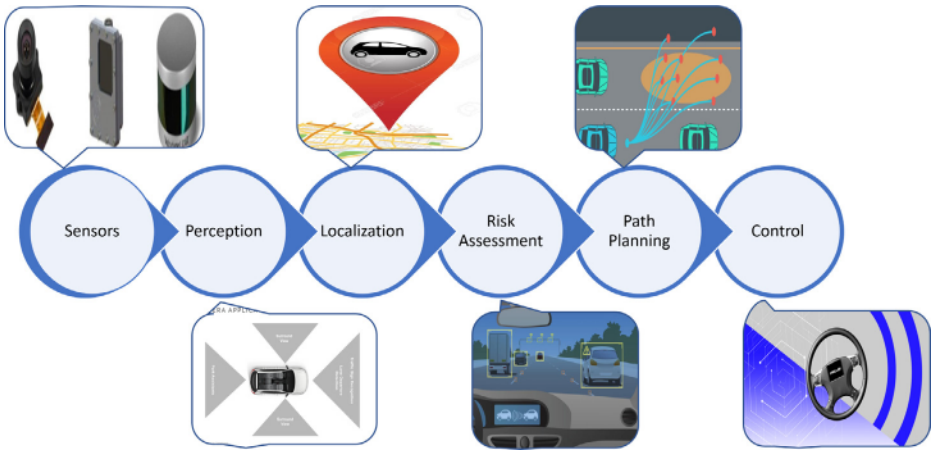


Fig. 1. Phases AD System.

System (ADS) to reduce accidents and harmful emissions by reducing the path length of trip [6].

Recently, the improvements in the artificial intelligence research and availability of sensors greatly participated in the improvement in the ADS [7]. Before going through the challenges of the current algorithms, the phases of the ADS will be explained in the following section.

1.2. Stages of AD systems

The AD system has six stages, as shown in Fig. 1. It starts with the sensors' hardware layer, which gathers the data from the environment. The second stage is the perception stage, where object tracking, object detection, and lane detection tasks are performed. The third stage is called the localization and mapping stage, followed by the assessment stage. The fifth stage is concerned with planning and decision-making. The final stage is the hardware control layer in which the control actions, such as steering angles, are considered [7].

The ADS starts with the hardware sensors layer. The sensors are the main factors in interacting with the environment in the ADS. The information about the surroundings, including static and dynamic objects, is gathered to be fed to the next perception layer. The primary purpose of the perception stage is to process the data gathered from the sensors and extract information that will be useful in the following stages for the ADS [8].

The third stage is the localization and mapping layer. The main objective of the mapping and localization is to get the vehicle's position described in the reference frame in the environment [9]. The fourth layer is the assessment layer. Assessment is concerned with the overall risk estimation and predicting the intentions of the surrounding human drivers to avoid accidents.

The fifth layer is the path planning and decision-making layer. It is concerned with getting the shortest path, free of collisions, in real-time for the vehicle between the start point and the endpoint [8]. The last layer concerns the vehicle control actions used to perform the path in real such as torque, acceleration, steering wheel angle, etc. [10].

1.3. Challenges and levels of AD systems

There are five levels of the ADS, as shown in Fig. 2. Level zero is a fully manual driving system where human drivers make all decisions. Level one uses basic systems to assist the human driver, such as the Anti-lock Braking System (ABS). Level two is a hands-off system, called a partially automated system, in which advanced assistance is provided to help the human driver, such as collision avoidance and emergency braking systems.

The real challenges start from level three, where the ADS stages should be considered a whole system rather than sub-systems. In level three, the car can drive itself, but the human driver can only interfere to respond to emergency alerts. It can only work in limited situations, such as motorways. Audi claims to be the first manufacturer to design a level 3 autonomous car on the highways. However, the transition between manual and autonomous driving can increase the risk of collision [11].

Level four can only work in limited environments in which detailed maps exist but without the need for human driver attention. The car can start the trip, drive safely to the destination, stop, and park without interference from a human driver. Level five is a fully automated driving system that can operate under any weather conditions and in any area without a human driver. According to Toyota research, no company is close to the level five autonomous driving system [7].

Levels four and five are open-challenging problems that still need to be achieved. Moreover, levels two and three have not been safely achieved as well. From July 2021 to May 2022, around 400 level 2 ADS car accidents have been reported by the National Highway Traffic Safety Administration (NHTSA), involving 273 T cars. Therefore, optimal performance is essential to achieve these levels, beginning with the perception layer and continuing to the path planning layer. Failure in one layer can lead to an accident. The rain was the reason for the Hyundai ADS crash. The ADS of Google crashed into a bus because of a failure to measure the bus speed. A driver was killed in a Tesla car accident because the car could not recognize a white truck.

The research is still working on the challenges in each stage of the ADS. Most stages in the AD system are well-researched. However, the most challenging problem in the AD system is the path planning stage with avoiding obstacles [7]. Therefore, the focus of this study will be on path planning.

The contributions of this review paper can be listed as follows:

1. New and comprehensive taxonomy of the path planning algorithms.
2. The stages of the ADS will be discussed in detail, starting from sensors to the decision-making layer.
3. The recent state-of-the-art techniques to address the path planning problem in the last five years have been discussed intensively.
4. The main concept of each path planning algorithm is explained, including the pros and cons of recent research; they are also compared in a tabular manner.
5. Statistics are conducted on the references used in the path planning algorithms, which can help to give insight for future research.

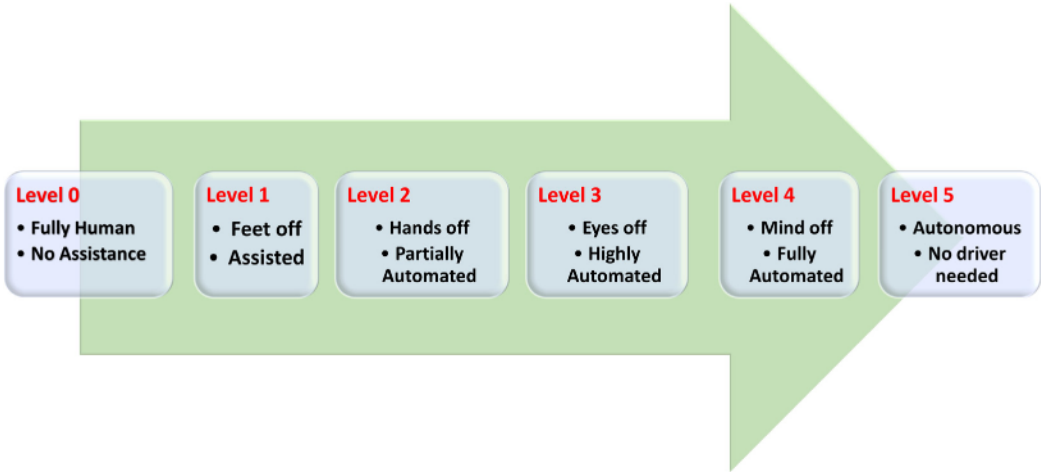


Fig. 2. Levels of Automated driving.

The paper is organized as follows: Section 2 discusses related review papers. Section 3 covers sensor technologies in autonomous driving systems. Section 4 explores perception techniques, followed by Section 5, which addresses localization and mapping technologies. Section 6 discusses assessment methods. The path planning phase, a crucial component in ADS, is divided into three main categories: Traditional Techniques, Machine Learning Techniques, and Meta-heuristic Optimization Techniques. These categories are detailed, starting with traditional techniques in Section 7, which includes graph-based methods, sampling-based methods, gradient-based methods, optimization-based methods, and interpolation curve methods. Section 8 discusses machine learning techniques for path planning, and Section 9 delves into meta-heuristic optimization methods. Section 10 analyzes the path planning literature from a statistical point of view. The paper concludes with a summary of the pros and cons of these algorithms. Appendix (Table 11) lists all the abbreviations used in the paper.

2. Related work

The literature has multiple review papers on path planning and the autonomous driving system. Gonzalez et al. introduced a review paper on the motion planning algorithm in the ADS in 2015, but it did not cover the machine learning and meta-heuristic optimization categories. Moreover, it did not explore the other phases in the ADS [12]. Claussmann et al. proposed a review of the path-planning algorithms in the motorway driving scenario until 2018. Furthermore, They did not cover the other phases of the ADS [13].

Aradi proposed a review paper that focuses only on the deep reinforcement learning algorithms that deal with path planning in the ADS until the year 2019. This paper did not explore the other stages of the ADS [14]. Yoganandhan et al. introduced a paper that explored ADS's fundamentals and stages. This paper focused more on the method than the related work behind each method. They cited only 42 references until 2019 [15]. Hadi et al. proposed a review paper that presents the recent path-planning techniques for underwater vehicles, not autonomous cars [16].

Zhou et al. proposed a review paper exploring the ADS's path planning algorithms until 2020. However, this review did not cover the meta-heuristic optimization algorithms, and they did not explore the other stages of the ADS [17]. Puente et al. introduced a review paper on the algorithms applied to address the motion planning problem in Unmanned Aerial Vehicles (UAVs), not autonomous cars [18]. Huang and Chen introduced a review paper that explores the path planning algorithms in the ADS until 2020, but it did not cover machine learning and meta-heuristic optimization algorithms. Moreover, the other stages of the ADS were not explained [19].

Tyagi and Aswathy proposed a paper that explores the open challenges in the smart autonomous vehicle. They did not focus on a specific stage or specific category of algorithms. However, they described the tasks and challenges in the ADS [20]. Vagale et al. introduced a review paper that explores the path planning problem for surface vehicles, not autonomous cars [21]. Ye et al. proposed a review paper about deep reinforcement learning algorithms that deal with path planning in the ADS until 2020. It did not explore the other categories of the algorithm, and it did not present the other phases of the ADS [22].

Xiao et al. presented a review paper exploring machine learning techniques for the path-planning problem in the ADS. They focused only on the machine learning category [23]. Ignatious et al. proposed a review of the sensors in the ADS. It did not focus on the other stages of the ADS or the path planning phase [24]. This review paper discusses all the stages of ADS. Moreover, a detailed taxonomy of the recent techniques that solve the path planning problem in ADS until 2022 is explored.

3. Layer 1: Sensors

The sensors are the first layer of the ADS, which is responsible for interacting with the environment in the ADS. The information about the surroundings, including static and dynamic objects, is gathered to be fed to the next perception layer. Multiple types of sensors have recently been used in the ADS [25].

3.1. Monocular cameras

Monocular Cameras are used to sense color. They are passive sensors; no signals are emitted to obtain the measurements. The cameras have different characteristics, such as resolution, frame rate, and lens size. One of the challenges in the cameras is night visibility and capturing dark shadow spots [26]. This problem is addressed via multiple solutions such as infrared cameras [27]. High Dynamic Range (HDR) is another challenging problem in cameras. This problem happens when dark and highly illuminated regions exist in the same frame. Recently, the cameras have HDR mode that can deal with this problem [28]. The main drawback of Monocular Cameras is the difficulty of getting the depth information. 3D cameras and event cameras can solve this problem.

3.2. 3D cameras

The 3D camera can obtain depth information by three principles. The first principle is stereo vision. In the stereo cameras, the image is taken by two monocular cameras separated by baseline distance and

pointing in the same direction [29]. The stereo cameras suffer from low-textured pattern problems.

The second principle of the 3D camera is structured light cameras. A monocular camera is integrated into a structured light camera with an infrared emitting device. It overcomes the low-textured and has fast computational time. However, they need calibration with high accuracy restricted with a small operative range (less than 20 m) [30].

The third principle of the 3D camera is the time-of-flight camera. A set of infrared LEDs emits modulated light; then, the sensor captures the reflected light. The round-trip time is estimated for each pixel. Then, the distance can be obtained. These cameras have a high refresh rate (50 Hz) with depth map creation. However, they have a short operative range (10–20 m) [31].

3.3. Event camera

In the event cameras, each pixel in the frame is triggered independently and asynchronously when there is a light density change (the event). A stream of events is grouped to construct a frame-like image. This camera has a high dynamic range (up to 120 dB) and fast response time (within microseconds). This type of camera is suitable for dynamic object detection. Nevertheless, the event cameras have limited image resolution and pixel size [32].

3.4. Omnidirectional cameras

These cameras generate a 360-degree panorama vision. This vision can be done through camera arrays; each has wide-angle lenses (fish-eye). This array of cameras requires precise calibration between them to construct the image. This type of camera is suitable for 3-dimensional vision [33] and is used in SLAM applications [34]. However, the complex calibration process can generate highly distorted spherical images. There are other emerging types of cameras, such as flash cameras [35] and thermal cameras [36].

3.5. Radar

The main concept of Radar sensors relies on the Frequency-Modulated Continuous Wave (FMCW) technique. The radar emits high-frequency electromagnetic radio waves and receives the reflected waves from the object. The time of the go and return trip of the waves is called round-trip time. The distance is calculated based on the frequency shift between the reflected and emitted waves. The main advantage of radar is good object detection in dark, rainy, foggy, and dusty weather conditions. Furthermore, it has a high operative range (up to 250 m) [37]. Moreover, it considers depth.

Radar has some drawbacks, such as false positive and false negative detection. In false positive detection, the radar can amplify the signal of small obstacles that can be ignored, such as cans, and make its size bigger than reality. In false negative detection, the radar cannot detect massive objects made from non-reflecting materials such as wood. The horizontal resolution and accuracy are considered limitations of the radar. The emitted signals are material-dependent, so it will not be easy to separate between objects with similar reflection characteristics [38].

3.6. Light detection and ranging (LiDAR)

The LiDAR sensor is based on the same working principle as the radar sensor, but the only difference is that the LiDAR uses low-power infrared laser light instead of radio waves. A 360-degree rotating mirror is operated to change the direction of the laser pulse. This mirror helps to construct a 3D point cloud of the surrounding environment. LiDAR considers the depth information and has higher accuracy than radar and can measure objects within 0.1 meters [39].

LiDAR sensor has some disadvantages. It has a low vertical resolution. LiDAR has poor dark color detection, leading to false negative detection. Black cars can be invisible to LiDAR because the black color absorbs most of the radiation with little reflection. Moreover, weather conditions like rain and fog can negatively affect LiDAR. The water drops of rain and fog can scatter the light beams of LiDAR, which can lead to false readings [40]. LiDAR performance is better than that of cameras but worse than that of radar. Furthermore, the LiDAR was bigger than radar and cameras.

From the above discussion of sensors, each sensor has advantages and disadvantages. The state-of-the-art techniques are based on sensor fusion between multiple sensors instead of using a single sensor because this helps to eliminate the single point of failure problem, [41]. The sensors' characteristics can be summarized in Table 1.

4. Layer 2: Perception

The main objective of the perception layer is to receive data from the surrounding environment and extract information that will be useful in the following stages of the ADS. The primary sensor used in the perception layer is the camera with the aid of computer vision. Detection is the main task of the perception layer. The tasks of the perception layer are discussed in the following subsections.

4.1. Image-based 2D object detection

Object detection recognizes static and dynamic objects like road signs, traffic lights, and other moving vehicles. Image-based object detection is one of the standard methods to detect objects. The main idea is to detect if specific objects exist in the captured image. Next, the size is identified using a rectangular bounding box. Cameras are the primary sensors used for the Image-based object detection process.

Deep Convolutional Neural Network (DCNN) is the most well-known object detection method. It can be done as a single stage to detect a specific object, or it can be done in two stages: one to detect general regions of interest and the second to detect a specific object in the region. The most common single-stage detector is the You Only Look Once (YOLO) CNN network [42]. It is improved to multiple versions such as YOLO9000 [43] and YOLOv3 [44]. Single Shot Detector (SSD) is another single-stage detector based on DCNN and is faster and can detect small objects better than YOLO [45]. However, the Region Proposal Networks (RPN) is a popular two-stage detector [46]. Single-stage detection is faster and consumes less memory, so that it can be used in real-time more than two-stage detection.

4.2. Image segmentation

Image segmentation is classified into two types: Semantic Segmentation and instance segmentation. In Semantic Segmentation, a class label is assigned to each pixel in the image. In instance segmentation, a class label is assigned to each instance in the image instead of classifying pixels based on box boundaries. Instance segmentation is more suitable for autonomous driving tasks to separate objects with different trajectories. However, Semantic Segmentation is better for detecting objects that are difficult to recognize by box bounties, such as sidewalks, traffic lines, and buildings. The primary sensor used in the segmentation task is the camera.

Neural networks are the most commonly used techniques in the segmentation process, such as Recurrent Neural Networks (RNN) [46], Pyramids Networks [47], and sparse convolutions networks [48]. Moreover, Transposed Convolutions Networks, based on convolution to extract features from images followed by the deconvolutions process, are used to get the labels for Pixels [49]. The main merit of segmentation networks is that they can learn as a universal feature extractor that can be generalized for all perception tasks in the ADS. However, the segmentation networks are slow and require high computational time.

Table 1
Characteristics of sensors in AD system.

Sensor	Idea	advantages	drawbacks
Monocular Cameras	Passive sensors, collect lights via the lens.	Can sense color.	No depth , dynamic range. low night visibility
3D stereo camera	2 monocular cameras separated by distance and point in the same direction.	measure depth.	low-textured patterns problem.
3D structured light cameras	monocular camera integrated with infrared emitting devices.	fast, measures depth, no low-textured problem.	need high accuracy calibration, small range (< 20 m).
3D time-of-flight camera	camera integrated with set of infrared LEDs.	high refresh rate (50 Hz), generates depth map.	short range (10–20 m).
Event Camera	each pixel is triggered independently when a light density changes (event), then, A stream of events are grouped to construct a frame-like image.	high dynamic range (up to 120 dB), fast response time (within microseconds), good for dynamic object detection.	limited image resolution, limited pixel size.
360 Omnidirectional cameras	generates 360 panorama vision via camera arrays, each camera has wide angle lenses (fish-eye), which require precise calibration.	good for 3-dimensional vision, used in SLAM applications.	the calibration process is difficult. Generate highly distorted spherical images.
Radar	Emits radio waves, measure frequency shift between reflected and emitted waves.	high range (up to 250 m), works under any weather conditions. considers depth information.	false positive and negative detection, low accuracy and horizontal resolution, material-dependent reflection.
LiDAR	the same as radar but uses low power infrared Laser light.	Depth info, higher accuracy than radar, measure objects within 0.1 meters.	low vertical resolution, big size. affected by weather conditions.

4.3. 3D object detection

In 3D object detection, depth information should be considered to convert 2D objects into 3D objects. This information can be obtained using multiple techniques like 3D stereo cameras, LiDAR, and Radar sensors.

Grouping sparse 3D data points obtained from 3D sensors into a 3D object point cloud is challenging for 3D object detection. There are multiple algorithms and neural networks that can be used in this process such as euclidean clustering [50], region-growing methods [51], VoxelNet grids based on RGB-D for colored point cloud [52,53], VeloFCN network [54]. These algorithms are tested on scenes dataset [55] and KITTI dataset [56] because they provide labeled 3D scenes.

4.4. Object tracking

The estimation of object location is not sufficient in the ADS. It is essential to detect the dynamic object trajectory and velocity to expect the object's future location. This data about moving objects can be gathered from cameras, LiDAR, and radar sensors. Using only one sensor for object tracking is insufficient, but sensor fusion is required for accurate object tracking [57]. Object tracker algorithms are based on three steps: first, generate an occupancy map for all sensors; second, data association techniques are used to find the association between objects; finally, filtering methods are used for smooth dynamic detection.

As mentioned, multiple sensors gather accurate data for the object tracking task, called sensor fusion. Therefore, occupancy maps are used as a single frame for all sensors that gather all data about surrounding objects [58]. The next step is to use the data association technique on the occupancy frame to find a relationship between objects. The most common Data association techniques that establish an association between objects are nearest neighbor methods, Image-based methods combined with gradients and KLT features [59], Point cloud-based methods integrated with point density and Hausdorff distance [60]. The third step is to use filtering methods for smooth dynamic detection such as traditional Bayes filters, Kalman filters [61], particle filters as non-parametric filter [62], Rao-Blackwellized particle filters [63].

Recently, deep neural networks have been used as a one-step object tracker. These networks need to be trained before using it in real-time applications. Convolution Neural Networks (CNN) are used for real-time object tracking using images obtained from monocular cameras [64,65].

4.5. Road and lane detection

The previously discussed object detection techniques are insufficient to detect objects with continuous surfaces such as roads and lanes. Road detection aims to obtain a drivable region for the autonomous vehicle. Semantic segmentation can be used to find the drivable area, but more is needed to understand road intersections and branches. Therefore, this task needs to be discussed separately [66].

The lane and road detection techniques are performed via several steps. The first step is the pre-processing of the raw data obtained from sensors. The most common pre-processing tasks are color correction for camera images [67], map-based filtering [60] and ground extraction [68] for LiDAR data and identifying the dynamic objects to isolate static objects such as the road lanes.

The second step is lane feature extraction. Road estimation and lane markings can be done based on filters, gradient information, and intensity information. The 3D road structure can be identified based on lidars and 3D stereo cameras, machine learning methods, and appearance-based segmentation.

The third step is to construct the continuity of the lanes and roads based on geometric interpolation parametric techniques such as splines [69], non-parametric continuous models [70].

The last step is temporal integration, in which the road tracking system is integrated with vehicle dynamics to achieve smooth results. Kalman filtering [71], or particle filtering [72] can be used to make the results smoother. The perception tasks can be summarized in Table 2.

5. Layer 3: Localization and mapping

The primary purpose of the localization and mapping is to find the position of the vehicle relative to the reference frame in the environment [9]. There are three standard techniques for localization: GPS-IMU fusion, SLAM algorithm, and priori map-based localization.

Table 2
Perception tasks.

Task	Idea	Sensors	Steps and algorithms
Image-based 2D Object detection	Detect if specific objects exit in the image, determine its location and size using a rectangular bounding box.	Cameras	Single stage: YOLO, SDD, two-stage: faster. RPN. Single stage faster.
Image segmentation	Semantic Seg. classifies pixels in images, instance seg. classifies objects based on box boundaries.	cameras	RNN, Pyramids Networks, Transposed Convolutions Networks, Slow.
3D object detection	depth information should be considered to convert the 2D objects into 3D objects.	3D stereo cameras, LiDAR, Radar	Euclidean clustering , VeloFCN, VoxelNet network.
Object tracking	Detect the dynamic object trajectory velocity to predict the future location of the object	cameras, LiDAR, Radar	1- occupancy map as a single frame for all sensors. 2- data association (nearest neighbor, Image, Point cloud based) 3- filtering for smoothing (Bayes, Kalman, particle filters).
Road and lane Detection	Find the drivable region for the autonomous vehicle.	cameras, lidar, radar	1- data pre-processing (color correction, map-based filtering). 2-lane feature extraction. 3- construct the continuity of lanes (geometric parametric and non-parametric alg.). 4- temporal integration for smoothing (Kalman, particle filters).

5.1. GPS-IMU fusion

The Inertial Measurement Unit (IMU) sensor consists of some sensors such as a gyroscope, magnetometer, and accelerometer. These sensors can identify the angle and the direction of the vehicle. However, each sensor has its errors, so using multiple sensors leads to error accumulation that can lead to failure over time. Global Positioning System (GPS) sensor provides the coordinates values: the latitude and longitude, which can be represented in the X and Y axes of the earth. The fusion between GPS and IMU sensors is used to correct the accumulated error in the IMU sensor. Therefore, it can define the location and orientation of the object. Nevertheless, the precision of the GPS-IMU is not high enough for the ADS, but it can be used as an initial estimation for the ADS [73].

5.2. Simultaneous Localization and Mapping (SLAM)

SLAM technique is based on building an online map for an unknown environment and obtaining the vehicle's location simultaneously. There are two main types of SLAMs: visual SLAM (vSLAM) and LiDAR SLAM. Visual SLAM builds the map based only on images collected by cameras (monocular, stereo, omnidirectional). It is cheap and can represent landmarks, but no depth information exists. There are two types of vSLAM algorithms: Dense algorithms, which rely on the overall image brightness such as DTAM, LSD-SLAM, DSO, and SVO; Sparse algorithms, which are based on some feature points in images such as PTAM and ORB-SLAM [74].

LiDAR SLAM collects information from the LiDAR sensor to construct point cloud maps. It has high-accuracy distance measurements. However, the matching process needs to be detailed with more information. In general, SLAM does not need any information about the environment. Therefore, it can be deployed anywhere. However, it has high computational time, especially in outdoor applications [74].

5.3. Localization using priori map

Localization is obtained by comparing online information, such as location, from a GPS sensor and a pre-built map. The exact location is determined based on the best match between them. Map-based methods consist of two main types: point cloud matching methods and landmark search.

In the landmark search, signs and road markers are distributed in the environment and used as landmarks. At the same time, the vehicle moves, sensors, and road marking detection algorithms are used to identify the landmarks and compare them to a predefined map to identify the location. This method has low computational time, but it is landmark-dependent. If the landmarks are insufficient, the ADS is prone to fail [75].

The second map-based method is point cloud matching. An online small point cloud map obtained from a sensor, such as LiDAR, will be compared to an offline large priori point cloud map. The location is determined based on the best match between two points on the two maps. The algorithms used for point cloud matching are normal distributions transform (NDT) and iterative closest point (ICP) algorithms [76]. This method can give a more accurate location than landmark search, but it has a high computational time.

The main demerit of the map-based technique is the need to build a map before using the algorithm. Moreover, any environmental change will lead to misleading localization, and the maps should be updated to adapt to the environmental change [77]. The localization and mapping techniques are summarized in Table 3.

6. Layer 4: Assessment

Assessment is concerned with the overall risk estimation and predicting the intentions of the surrounding human drivers to avoid accidents. The assessment has three main types as follows:

- uncertainty and risk assessment.
- assessment of the behavior of surrounding human driving.
- identifying driving style.

Uncertainty and risk assessment is concerned with monitoring the risk of the overall driving scene. It can detect unsafe lane change events and changes in the road. This process can be done using sensors such as cameras and detection algorithms such as neural networks.

Surrounding human driving behavior assessment is concerned with understanding the intention of the surrounding human drivers by predicting the future behavior of their vehicle to avoid accidents. Predicting the dangerous cuts and overtaking performed by surrounding drivers are examples of this task. Human traits should be taken into consideration to judge the overall driving scenes. The main challenge

Table 3
Localization methods.

Method	Idea	Sensors	Pros.	Cons.
GPS-IMU	GPS for location, IMU for direction, fusion to reduce accumulated error.	GPS sensor, IMU (gyroscope, magnetometer accelerometer)	Low computational time, small size, low cost, give quick initial position.	Low accuracy.
Visual SLAM	Build online map using camera images and determine the location at the same time.	Camera	Cheap, represents landmarks, can work anywhere.	Lack of depth, high computational time.
LiDAR SLAM SLAM	Build online map using lidar and determine location at the same time.	LiDAR	High accuracy distance, can work anywhere.	Not a fine map, large size, very slow.
Landmark mab based	Sensor detect landmark (signs, markers), compared to pre-defended map to match the location.	Marker-based i.e. camera	Moderate speed high accuracy	Landmark dependent, prone to fail, need a priori updated map.
Point cloud matching mab-based	Online small cloud map is compared to offline large priori cloud map, location is obtained by ICP or NDT.	GPS, LiDAR	Best stability best accuracy	Time consuming largest size, need a priori updated map.

in this task is the short time available to sense the surrounding vehicles in real-time, which needs to be improved to predict long-term behavior.

Driving style recognition is the most common task in the assessment phase. The driving style can be defined based on the human driver's aggressiveness or the surrounding vehicle's relative speed [78]. Clustering unsupervised learning algorithms, such as PCA algorithm and K-means, and supervised learning methods, such as neural networks and SVM, are the most common methods used to classify the driving styles [79].

7. Layer 5 - part 1: Path planning (traditional techniques)

The path planning stage aims to obtain a safe trajectory for the vehicle. This process can be done through two phases. First, the global path is generated between the start point and the destination based on GPS localization and offline map. The next step is to get an obstacle-free local path that executes the global path without collision. The challenge of finding the shortest path can be considered as an NP-hard optimization problem. Therefore, finding the optimal path exponentially grows with the number of available nodes. The research is still trying to find an optimal solution for the NP-hard problem via enhancing the performance of the current algorithms [80]. Obtaining the optimal shortest collision-free path is still challenging in the ADS [7].

Most stages in the AD system are well-researched. However, the most challenging problem in the AD system is the path planning stage with avoiding obstacles [7]. Therefore, the focus of this research is on path planning.

7.1. Graph-based methods for path planning in ADS

Dijkstra [81] and A* [82] are the most well-known techniques for the graph-based path-planning problem. These algorithms always give discontinuous paths instead of continuous ones, leading to jerky paths [12].

Yijing et al. proposed a novel A* algorithm with an Equal-Step Sampling (A*ESS) algorithm to address local path planning. They designed an enhanced reward function based on the kinematics model of the car to improve the path comfort. Results proved that the A*ESS algorithm outperformed the traditional A* algorithm with the central search node of each lattice cell. However, the statistical analysis is poor, and the algorithm is not evaluated on benchmark problems [83].

Udomsil et al. applied the A* algorithm in solving motion planning in a static environment. A* is used to generate the path. Moreover, the A* is also used to avoid obstacles by generating a new trajectory through the Unity 3D collision detection system. The algorithm managed to obtain a path in a simple environment. Nevertheless, it is not compared to any other algorithm to measure its relative performance. Furthermore, the obstacles were too simple and static [84].

The main demerit of the conventional $A*$ method is the high complexity. Nevertheless, Song et al. [85] introduced a hybrid algorithm of the $A*$ and PSO algorithms. The new algorithm managed to enhance the complexity of the paths. However, it has a high computational time.

Yeong et al. proposed a hybrid algorithm between the Predictive-Dynamic Window Approach (P-DWA) algorithm and the A* algorithm, called A*-PDWA. The A* algorithm is used to obtain an approximate path. Then, a cardinal spline interpolation technique is applied to smooth the obtained path. After that, the P-DWA algorithm is used to avoid obstacles in the local path and follow the points obtained from the global trajectory. The algorithm managed to find an acceptable path without collision. However, the algorithm is not compared to any other algorithm [86].

Zhong et al. designed a novel hybrid algorithm between the A* and Adaptive Window Approach (A*-DWA) algorithms. The A* algorithm generates the rough path. Next, the DWA algorithm is deployed to achieve real-time trajectory planning with obstacle avoidance. The practical and simulation results indicated that the A*-DWA algorithm can achieve a feasible path in complex dynamic environments. Nevertheless, the algorithm's parameters must be optimized as they were assumed by experience. Moreover, the algorithm is not compared to other techniques [87].

Maw et al. introduced the improved Anytime Dynamic A*(iADA*) algorithm for a dynamic environment. The concept of the iADA* algorithm is to find an initial path to allow the vehicle to begin movement. Then, the path is optimized for a short path during the vehicle's movement. If the vehicle faces an obstacle, the algorithm updates the path to get a new collision-free path. The results proved that the iADA* algorithm is faster than ADA*, D* Lite, MPGAA*, and D* Extra Lite algorithms. However, the algorithm gave longer paths in terms of distance, and more scenarios need to be validated [88].

Thoresen et al. proposed a Traversability Hybrid A* (THA*) algorithm, which uses estimated traversability to optimize the distance of the path for the car. The THA* algorithm outperformed the original hybrid A* algorithm over distances of up to 270 m in the experiment. Nevertheless, the computational time of the algorithm is high, which is sometimes too short for a planning horizon [89].

Zhu et al. developed a reverse labeling Dijkstra algorithm (RLDA), which relies on the basic Dijkstra algorithm but with reverse labeling. The RLDA algorithm is compared with ACO, GA, PSO, NNa, and OPABRL algorithms. The results indicated that the RLDA algorithm has a faster convergence rate than ACO, NNA, and GA. The main drawback is that the algorithm has a high computational time when the problem nodes are more significant than 350 [90].

Liu et al. applied a two-level algorithm to solve the path-planning problem in a smart car. The DWA algorithm is deployed for local path planning, whereas the Dijkstra algorithm is used for global path planning. This algorithm managed successfully to avoid obstacles from

Table 4
Literature Review Summary for graph-based techniques.

Algorithm	Idea	Pros	Cons
A*ESS [83]	enhanced cost function based on vehicle kinematics model, Equal-Step Sampling.	outperformed traditional A* algorithm with search node.	poor statistics, not tested on benchmark functions.
A*-based [84]	A* finds path, and avoids obstacles in Unity 3D collision detection system.	managed to find path in simple static environment.	No comparison, too simple obstacles.
A*-PSO [85]	Hybrid A* with PSO	better than A*	high computational time
A*-PDWA [86]	Hybrid; A* gets global path, cardinal spline smooths path, P-DWA avoids obstacles.	managed to find safe path. managed to find a safe path.	algorithm is not compared with any other algorithm.
A*-DWA [87]	A* finds the global path, DWA avoids the obstacle.	achieved feasible path in complex dynamic environments, applied in real-time.	parameters are not optimized, no comparison.
iADA* [88]	finds initial path to start movement, path is optimized to be shorter during movement.	faster than ADA*, D* Lite, MPGAA*, D* Extra Lite.	longer paths, needs more scenarios.
THA* [89]	Hybrid A* that estimates terrain traversability.	outperformed the original hybrid A* algorithm, 270 m real experiment distance.	High computational time that leads sometimes to a short planning horizon.
RLDA [90]	Dijkstra algorithm with reverse labeling	Fast solutions for simple search space	Slow in complex search space
D-DWA [91]	Dijkstra algorithm with dynamic window approach	Successful implementation	No comparative and statistical analysis
IA*FC [92]	A* evaluation function includes fuel consumption in idle state.	outperformed the traditional A* by 16.949%.	High complexity and computational time, not accurate in actual traffic.
LTSTP [93]	Long-term: A* finds path, JPS utilizes path, Long-term: PSO optimize path.	obtained safe path, avg. time 31s, worst-case time 94s.	no comparison, needs more complex scenarios.
IA*-DWA [94]	hybrid; A* finds path, Cubic Bezier curves smooths path, DWA avoids obstacles.	outperformed A*, DWA algorithms.	needs testing on more complex environment and benchmark functions.

the starting position to reach the final position. However, this study focused on hardware implementation, but no statistical analysis is used to compare the performance of the algorithm with the current literature [91]

Liu and Zhang proposed an improved A* based on the fuel consumption (IA*FC) algorithm for the idle states in the ADS. The objective function of the A* algorithm includes fuel consumption, especially in idle states in red traffic. Experimental results showed that the IA*FC algorithm outperformed the traditional A* algorithm by 16.949% during red light traffic. However, the complexity and computational time need to be improved. Moreover, the algorithm needs to be more accurate in the actual traffic environment [92].

Kim et al. designed a Hierarchical Long-Term and Short-Term Planner (LTSTP) algorithm for highway driving scenarios. This algorithm has two stages: short-term and long-term planning. In the long-term planning, the A* algorithm finds a rough 2D path, and then the Jump Point Search (JPS) algorithm is applied to optimize the path obtained from the A* algorithm. In short-term planning, the PSO algorithm is deployed to enhance the path and generate parameters, such as steering angle. The simulation results proved that the LTSTP algorithm obtained a collision-free path with 31 s average computation time and 94 s worst-case computation time. However, the algorithm is not evaluated against any other state-of-the-art algorithm to judge the relative performance. More complex scenarios need to be considered [93].

Li et al. developed a hybrid algorithm between the DWA algorithm, the Improved A* algorithm, and Bezier Curves, named IA*-DWA. The purpose of the A* algorithm is to find the path, which is improved through the Cubic Bezier interpolation curves to make the path smoother. The use of the DWA algorithm avoids moving obstacles. The results showed that the IA*-DWA algorithm outperformed the A* and DWA algorithms. Nevertheless, the algorithm needs to be tested on a more complex environment and benchmark functions to test the performance [94]. Graph-based methods are summarized in Table 4.

Graph-based methods are mainly used for discrete search spaces that a graph can represent. Therefore, it cannot deal with complex scenarios with continuous search spaces. Moreover, the path generated from them is jerky. Future research in graph-based algorithms for autonomous driving systems could focus on hybridizing graph search algorithms with interpolation techniques to make the path smoother. On the other hand, dealing with continuous and complex search spaces requires hybridizing them with other path-planning methods, such as the PSO algorithm, which adds the ability to deal with continuous search spaces. Another strategy is to hybridize them with sampling methods that facilitate the representation of the continuous search space for the graph-based algorithms.

7.2. Sampling-based methods

The concept of the Sampling-Based Planning (SBP) algorithms is to construct connections in the C-space by generating random paths inside it [95]. Rapidly-exploring Random Tree (RRT) [96] is the most common SBP algorithm. The main idea of the RRT algorithm is to incrementally build the path between the start and endpoints with random tree-like branches. The paths between two points are guaranteed to be found if given enough run-time. Therefore, it is called a probabilistically complete algorithm [96]. The main drawback of the SBP are the jerky solutions like the Graph-based algorithms [12].

Lim et al. introduced a hybrid algorithm between the Sequential Quadratic Programming (SQP) numerical optimization algorithm and a sampling-based algorithm named SB-SQB. The sampling-based algorithm is used to get the path using environmental models. The SQP numerical optimization technique is deployed to obtain the trajectory based on the path obtained from the sampling-based planner. The hybrid SB-SQB algorithm managed to find reasonable paths within 50 ms. Nevertheless, the SB-SQB algorithm is not compared to other algorithms. Furthermore, it must be tested on more complex scenarios [97].

Wang et al. proposed a hybrid algorithm between the Rapidly-exploring Random Tree algorithm and the DWA algorithm (RRT-DWA). The RRT algorithm finds the global path, while the DWA algorithm estimates rotational and translational velocity. Results indicated that the RRT-DWA method is faster and smoother than the Dijkstra-DWA algorithm and A*-DWA algorithm. However, the dynamic objects were

not taken into consideration. Furthermore, the statistical analysis was insufficient, and more scenarios must be tested [98].

Varghese and Jisha proposed an Improved Rapidly-exploring Random Tree associated with a PI controller (IRRT-PI). The IRRT algorithm is integrated with the PI controller. Using the control theory, the PI controller eliminates the position and acceleration errors between the actual and generated paths. The algorithm managed to find a path with acceptable error. However, the algorithm is not compared to any other algorithm to estimate its performance [99].

Lim et al. designed a hybrid algorithm between the sampling and numerical optimization algorithms (S-NO). The sampling-based technique in [100] is used to find the lateral movement for a dynamic environment. Quadratic programming (QP) is deployed to address the convex optimization problem to generate longitudinal movements that ensure diversity with no restrictions. The algorithm managed to find a comfortable and safe path practically. However, the algorithm is not compared to other algorithms in literature [101].

Li et al. introduced an Adaptive Sampling-based Motion Planning with a Non-Conservatively Defensive Strategy (ASMP-NCSS) algorithm. The sampling method in the algorithm is based on time-varying distribution combined with a non-conservatively defensive strategy to generate safe paths. Results proved that the ASMP-NCSS algorithm is better than the traditional sampling-based Motion Planning (SMP) algorithm with uniform sampling in a dynamic environment. Nevertheless, more scenarios need to be tested. Furthermore, all the statistics were based on the graphical methods, not the numerical ones [102].

Feraco et al. implemented the basic RRT algorithm to solve the local trajectory planning for racing ADS. This algorithm managed to find the paths in an unknown environment that contains obstacles, such as traffic cones. A LIDAR sensor is used in the perception layer to sense the surrounding environment. The algorithm is implemented successfully. However, this algorithm is not compared to any other algorithm [103].

Chen et al. developed a two-level algorithm to address complicated environments with multiple obstacles. The improved Bi-RRT algorithm implements the high-level part to give an approximate initial path, achieving the non-holonomic constraints of the car. The low-level part of the algorithm is based on the polynomial Vector Field Histogram-(VFH-) algorithm to improve the trajectory generated from the Bi-RRT algorithm. The algorithm managed to find collision-free paths in different driving scenarios. However, it took a long time. Moreover, It is not compared to any other algorithm [104].

Wang et al. designed a hybrid algorithm between the convolutional neural network (CNN), the A* algorithm, and the RRT algorithm called the neural RRT* (NRRT*). A* algorithm generates the training dataset that forms the map information. The CNN network estimates the probability distribution of the optimal trajectory obtained by the A* algorithm. Then, this distribution is used to generate the path using the RRT algorithm via the sampling process. The results showed that the NRRT* algorithm could find better solutions than the ones obtained by the other RRT variants. However, there is no online interaction with the environment [105].

Zhang et al. implemented an improved Sampling-Based Motion Planning algorithm. A novel bias sampling technique speeds up the traditional SBMP algorithm. This technique selects only the most strategically necessary sample points to construct a smooth and safe path. The algorithm gave smoother and faster solutions than the ones from traditional sampling-based algorithms. Moreover, the solutions are even better than the ones from human drivers. However, this algorithm is compared only to sampling-based algorithms (uniform sampling and bias Gaussian sampling) without considering other path planning types [106].

Rong et al. introduced an improved RRT* algorithm based on machine learning named Attention-RRT*. The sampling distribution in the Attention-RRT* algorithm is generated using a Conditional Variational Encoder based on 3D CNN. The results showed that the Attention-RRT* outperformed the traditional RRT* algorithm with uniform sampling. However, the construction of the 3D-CNN needs to be enhanced. Moreover, inaccuracy was not considered; the statistical analysis was insufficient [107].

Huang et al. proposed an Improved RRT (i-RRT) algorithm in which the i-RRT algorithm finds the path, and the B-spline interpolation algorithm makes the path smoother. Results proved that the i-RRT algorithm can find a personalized collision-free trajectory. Nevertheless, the algorithm is not compared to any other algorithm in literature [108].

Jin et al. proposed a Modified RRT* (MRRT*) algorithm, in which the RRT* algorithm generates the path, and the cubic B-spline interpolation algorithm makes the path smoother. The MRRT* algorithm succeeded in finding safe paths in real. However, no statistical analysis is performed with no comparison to recent state-of-the-art algorithms [109].

Rakita et al. introduced the SPRINT algorithm, a novel sampling-based algorithm. This algorithm is based on minimizing the number of collision checks using three heuristics. The first heuristic is giving high priority to the promising search regions. The second heuristic is getting samples from regions with a local minimum. The third heuristic is to direct the search away from states with previous collisions. The results showed that the algorithm outperformed RRT, RRT*, RRT-Connect, and other sampling-based algorithms. Nevertheless, the heuristics were constructed by observation. Moreover, the algorithm does not guarantee optimal solutions. Furthermore, results need more improvements [110].

Wang et al. proposed the Kinematic Constrained Bi-directional Rapidly-exploring Random Tree with Efficient Branch Pruning (KB-RRT*) algorithm. The concept of the algorithm is to avoid unnecessary tree growth by integrating the kinematic constraints. Furthermore, the branch pruning technique assists the obtained state in getting a better parent state and removing edges at high costs. The simulation results showed that the KB-RRT* algorithm performs better than the conventional K-RRT and K-RRT* algorithms. Nevertheless, only graphical statistical analysis had been demonstrated to judge the performance. Moreover, the algorithm is model-driven and cannot predict the promising sampling region [111].

Zhang and Wang designed an innovative Path Planning with a Line Segment algorithm (LSPP). The LSPP consists of three parts. The first part is the modified RRT algorithm to find the path. Second, the Artificial Potential Field with Azimuth and Distance (ADAPF) algorithm finds the distance to the obstacle and can fully utilize all information about the obstacles. Third, the Modified Dogleg technique integrated with the Symmetric Rank-1 algorithm guarantees that the car is far from obstacles. Real-world test results proved that the LSPP algorithm outperformed RRT, APF, and PRM algorithms. Nevertheless, the results were only based on graphical statistical analysis without quantities criteria [112].

Ganesan et al. proposed a Directional RRT* (D-RRT*) algorithm to decrease the sampling space by forming a simple elliptical heuristic, focusing on the direction between the start and endpoints. The results showed that the D-RRT* algorithm outperforms the traditional RRT* algorithm. Moreover, the TurtleBot3 robot validated the algorithm in real-time. However, the heuristic technique should be enhanced to minimize the number of visited nodes in a narrow environment. Furthermore, the algorithm is model-driven, not data-driven [113].

Flores et al. developed an improved RRT algorithm with a Custom Probability Density Function (RRT-CPDF) algorithm. A Custom Probability Density Function (C-PDF) is used in the RRT algorithm instead of the Uniform Probability Density Function (U-PDF). The simulation results indicated that the RRT-CPDF technique outperformed the traditional RRT algorithm with a Uniform Probability Density Function (U-PDF). However, the dynamic obstacles should be taken into consideration. Moreover, more complex scenarios should be tested [114].

Huang et al. addressed the path planning problem by proposing an improved RRT (IRRT) algorithm, in which random points are obtained based on the circular sampling technique to ensure randomness and enhance sampling efficiency. Moreover, they designed an extended

random point rule to filter the points based on the cost function. The B-spline curve interpolation method has been applied to make the path smoother. Simulation results proved that the IRRT algorithm outperformed the traditional RRT and Bi-RRT algorithms. However, the Improved RRT algorithm has some limitations in its model-driven scheme that need improvements. Furthermore, more scenarios and statistical analysis must be done [115].

Yang and Yao introduced a hybrid algorithm between the Pruning Rapidly-exploring Random Tree (PRRT) and the B-Spline Curve, named the PRRT-BSC algorithm. The RRT algorithm finds the path using a pruning method based on obstacle distribution. Then, the B-Spline interpolation technique is applied to make the path smoother. The PRRT-BSC algorithm found acceptable paths but is not compared to any algorithm in literature [116].

Zhang et al. introduced an Adaptive Improved RRT algorithm, where an adaptive directional sampling technique has been applied to reduce the random sampling points. A node selection technique has been used for further smoothness of the trajectory. The results proved that the path quality had been improved compared to the other RRT variants. However, this algorithm has some disadvantages. The algorithm takes a long time, especially on curved roads. The algorithm is not applied in real situations. The algorithm is only compared to RRT variants.[117]

Lu proposed an Enhanced Rapidly-exploring Random Tree (ERRT) algorithm by expanding the random tree based on the target random point. The results showed that ERRT is better than traditional RRT. Nevertheless, the statistical analysis needs to be more comprehensive. Moreover, the dynamic obstacles are not considered [118].

Spanogiannopoulos et al. applied the traditional RRT method to the path planning of self-driving cars in real-time—the proposed algorithm used only the point cloud data generated from a local sensor. The algorithm is applied and validated in a well-known benchmark. Moreover, the algorithm generated safe real-time trajectories of non-holonomic autonomous cars in unknown static environments. The paths are practically generated in roundabouts with obstacles for cars at speeds of up to 45 km. However, it is not compared to any other algorithm to test its relative speed and accuracy [119]. Sampling-based methods are summarized in Table 5.

Future directions for improving Sampling-Based Planning (SBP) methods in autonomous driving should focus on enhancing efficiency, smoothness, and adaptability. Developing hybrids that blend SBP with interpolation approaches, like spline methods, eradicates the sharpness of the jerky paths, making them smoother and more feasible, especially in complex driving scenarios. Regarding complex search spaces, applying bias sampling techniques and machine learning models within SBP algorithms can accelerate path planning and refine trajectories for complex and dynamic environments.

7.3. Gradient-based (Artificial Potential Field) methods for path planning in ADS

The Artificial Potential Field (APF) algorithm is a gradient-based technique in which the vehicle is displayed as a point in a potential field. This point is attracted to the destination endpoint and diverges from the obstacles. The resultant trajectory in the potential field represents the final path. The main merit of this technique is that it can produce collision-free trajectories within a small computation time. Nevertheless, the main demerit of the algorithm is that it could get stuck into a local minimum, which means there is a better solution, but the algorithm cannot reach it. The reason behind this is the gradient behavior of the algorithm. The APF algorithm relies on making the gradient reach a zero slope. When the zero-slope state is reached, the algorithm terminates. Therefore, the algorithm cannot explore new search space [120].

Hongyu proposed an Improved Artificial Potential Field (IAPF) algorithm that consists of five potential components: road potential, lane potential, target potential, velocity potential, and vehicle potential. The road and lane potential maps the road structure. Velocity potential prevents unnecessary change in the lane. Vehicle potential ensures a safe distance between obstacle and vehicle. The path is obtained through the integration of potential components. The results showed that the IAPF algorithm has a stable and robust performance. However, the algorithm is not validated in real life. Moreover, the algorithm is not compared to any other algorithm [121].

Lu et al. introduced the Adaptive Potential Field for Path Planning (ADPF-PP) algorithm. The dynamic characteristics of the ADPF algorithm are used to map the surrounding environment. Then, the ADPF-PP produces a trajectory free of collisions by minimizing the Potential Field that considers the surrounding obstacles. The results proved that the ADPF-PP outperformed the traditional CPF-PP and Improved Potential Field-based Path Planning algorithms. Nevertheless, the algorithm falls in local optima in a complex unknown environment [122].

Lin et al. designed a Model Predictive Path Planning based on the APF algorithm, named the MPPP-APF algorithm. They proposed an innovative curve-fitting technique integrated with the APF algorithm to generate the trajectory. The results showed that the MPPP-APF algorithm outperformed the standard APF algorithm. Nevertheless, more scenarios need to be validated. They only used graphical statistical analysis [123].

Li et al. developed a hybrid algorithm between Model Predictive Control (MPC) and the APF algorithm considering Driving Style, named APF-MPC-DS. The APF algorithm is used to model the environment and the driving style. Then, the MPC is integrated with APF to optimize the path and the motion control outputs. The results demonstrated that the APF-MPC-DS algorithm is more stable than the APF-MPC algorithm with no driving style proposed in [124]. However, the significance level needs to be discussed; they only used graphical statistical analysis [125].

Huang et al. proposed an integrated algorithm between the Constrained Delaunay Triangulation (CDT) and Artificial Potential Field (APF) Algorithm, named the CDT-APF algorithm. First, the CDT method has been applied to determine a safe region from the recent location of the car to the endpoint. Then, the APF algorithm defines the boundaries of this safe region. Then, the Model Predictive Controller (MPC) is deployed to obtain the path and utilize the control parameters. The results proved that the CDT-APF algorithm outperformed the traditional APF algorithm. However, the algorithm has yet to be tested on real-time applications. Moreover, only graphical statistical analysis is performed. Error measurements were not considered [126]. As previously mentioned before, in the sampling-based algorithms, Zhang and Wang introduced the LSPP algorithm, which is a hybrid algorithm between the modified RRT algorithm and the Artificial Potential Field algorithm [112].

Zhang et al. developed a hybrid algorithm between the Improved Artificial Potential Field (IAPF) and Gradient Descent Method (GDM) algorithm, called the IAPF-GDM algorithm. The IAPF algorithm, inspired by distance optimization, is used to utilize information about obstacles. The GDM algorithm generates the path based on the data from the IAPF algorithm. The IAPF-GDM algorithm outperformed the algorithm introduced in [127]. However, the algorithm assumed an idealistic environment and did not consider future and dynamic obstacles. Moreover, the study was based only on graphical statistics, not numerical ones, to judge the significance level [128].

Li et al. proposed the Dynamic Enhanced Firework Algorithm with APF algorithm (DynEFWA-APF), where dynamic environments, such as vehicle dynamics and road structure, are considered. Then, the optimization problem is formulated considering all environmental constraints. Finally, the DynEFWA-APF optimizes the problem to obtain a safe path. The algorithm gave an acceptable performance in a static and dynamic environment. However, the algorithm did not outperform the A*, APF, and GA-APF algorithms. Moreover, the driving style is not considered [129].

Table 5
Literature Review Summary for sampling-based Path Planning techniques.

Algorithm	Idea	Pros	Cons
SB-SQB [97]	hybrid; Upper SB: finds the paths based on environment, Lower SQP optimizes path.	obtained reasonable paths within 50 ms.	no comparison to other algorithms, needs more complex scenarios.
RRT-DWA [98]	Hybrid; RRT finds global path, DWA calculates velocity.	smoother and faster than Dijkstra-DWA and A*-DWA algorithms.	no dynamic objects, , no statistical analysis, needs more scenarios.
IRRT-PI [99]	hybrid RRT with PI controller to eliminate error between the generated and actual path.	managed to find a path with acceptable error.	No comparison with any other algorithm.
S-NO [101]	hybrid; sampling alg. [100] for lateral path, QP for longitudinal path and diversity.	managed to find a comfortable, safe path in real-time.	No comparison with other algorithms.
ASMP-NCSS [102]	based on time-varying distribution with non-conservatively defensive strategy.	better than SMP with uniform sampling in dynamic.	needs more scenarios, only graphical statistics.
RRT [103]	Basic RRT	successful in unknown environment	No comparison with any other algorithm
Bi-RRT [104]	Two levels: 1-improved Bidir. RRT for Approx. Path. 2- polynomial Vector Field Histogram- planning for acc. path	Managed to find safe paths.	Took long time, , no comparison with any other algorithm
NRRT* [105]	Hybrid: A* generates training data, CNN predicts probability distribution, RRT generates path.	Better than RRT and RRT*	No relative comparison with the other state-of-the-art
ISBMP [106]	improve SBMP by selecting only the most strategic sample points.	Faster than uniform sampling and bias Gaussian sampling.	does not consider other AI techniques.
Att-RRT* [107]	generated sampling distribution using CVAE attention mechanism based on 3D CNN.	outperformed the traditional RRT* with uniform sampling.	3D-CNN needs improvement, poor statistical analysis.
i-RRT [108]	i-RRT find the path, B-spline interpolation smooths path.	managed to find personalized safe trajectory.	no comparison with the state-of-the-art algorithms.
MRRT* [109]	MRRT* incrementally finds the path, cubic B-spline smooths the path.	managed to find collision-free paths in real.	no statistical analysis, no comparison with other algorithms.
SPRINT [110]	minimize no. of collision checks using three heuristics.	outperformed the other sampling-based algorithms.	heuristics based on observation, no guarantee optimal paths, results need improvements.
KB-RRT* [111]	kinematic constraints avoids tree overgrowth, branch pruning replaces bad edges with better ones.	better than K-RRT and K-RRT* algorithms.	model-driven, cannot predict good sampling area, only graph statistics.
LSPP [112]	Hybrid; MRRT finds path, ADAPF utilizes obstacles data, MDL-SR1 guarantees far obstacles distance.	outperformed RRT, oAPF, and PRM algorithms in real.	only graphical statistics.
D-RRT* [113]	reduces sampling space by elliptical heuristic from start to end.	outperforms traditional RRT*, real tested.	model-driven, high number of visited nodes in a narrow environment.
RRT-CPDF [114]	Custom Probability Density Function is used as a sampling method.	outperformed RRT with Uniform sampling distribution.	no dynamic obstacles, needs more complex scenarios.
IRRT [115]	based on circular sampling, random point filter, B-spline smooths path.	outperformed RRT and B-RRT algorithms.	model-driven with limitations, needs more scenarios and statistics.
PRRT-BSC [116]	hybrid; RRT finds path by pruning fn. based on obstacles, B-Spline smooths the path.	managed to find acceptable paths.	not compared to any other state-of-the-art algorithm.
AIRRT [117]	adaptive improved RRT with adaptive adaptive directional sampling and node node selection to make the path smoother.	Best performance compared with RRT variants.	Limitation in planning, slow performance, no on-site data, Only compared with RRT variants
ERRT [118]	RRT is enhanced by using the target point as a random point for random tree expansion.	better than traditional RRT.	statistical analysis is not sufficient, no dynamic obstacles.
RRT [119]	basic RRT with only point cloud data of nonholonomic cars, static unknown environments.	generate safe trajectories	No relative comparison with state-of-the-art algorithms.

Lin et al. introduced a hybrid algorithm between Potential Field and Sigmoid-based Safe Passage in the Model Predictive Controller algorithm, named the PF-SPMPC algorithm. The PF algorithm is used to model the environment. Then, the MPC controller is used to optimize the path with sigmoid-based safe passage constraints. The results proved that the algorithm outperformed the PF-MPC algorithm. However, more complex scenarios need to be tested. Moreover, Only graphical statistical analysis is performed [130].

Szczepanski et al. introduced the predictive APF (PAPF), where a novel local minimum avoidance method, called the top quark-based mechanism, is used to predict and bypass obstacles in advance. The Husarion ROSbot 2.0 PRO robot has been used to validate the algorithm. The results proved that the PAPF algorithm can provide a shorter path by up to 8.73% than the ones obtained from the traditional APF algorithm. However, the statistical analysis should be more

Table 6
Literature Review Summary for the gradient-based Path Planning techniques.

Algorithm	Idea	Pros	Cons
IAPF [121]	5 potential parts(road, lane, target, velocity, vehicle), the gradient method finds a path.	achieved stable and robust performance.	not validated in real-life, no comparison with other algorithms.
ADPF-PP [122]	PF maps the environment, then path is generated by minimizing the PF.	outperformed IPF-PP and CPF-PP algorithms.	fall in local optima in complex unknown environment.
MPPP-APF [123]	novel curve-fitting integrated with the APF and MPC controller to generate a path.	outperformed the standard APF algorithm.	more scenarios need to to be validated, only graphical statistics.
APF-MPC-DS [125]	hybrid; APF maps environment and driving style, MPC optimizes the path.	more stable than APF-MPC with no driving style [124].	only graphical statistical analysis.
CDT-APF [126]	hybrid; CDT finds safe path region, APF defines path boundaries, MPC finds a path.	outperformed traditional APF	only graphical statistics, Error was not considered, not tested in real-time.
IAPF-GDM [128]	Hybrid; IAPF utilizes obstacles data, GDM finds path based on IAFP data.	outperformed algorithm in [127].	no future dynamic obstacles, only graphical statistics, idealistic environment.
DynEFWA-APF [129]	consider dynamic constraints in the optimization problem solved by APF.	acceptable path in static and dynamic environment.	did not outperform A*, APF, GA-APF, driving style not considered.
PF-SPMPC [130]	Hybrid; PF maps the environment, MPC optimizes path with sigmoid passage constraints.	outperformed PF-MPC algorithm.	need more complex scenarios, only graphical statistics.
LSPP [112]	Hybrid; MRRT finds path, ADAPF utilizes obstacles data, MDL-SR1 guarantees far obstacles distance.	outperformed RRT, APF, APF, and PRM algorithms in real.	only graphical statistics.
PAPF [131]	top quark-based mechanism enables APF to bypass future obstacles.	Real-time test, outperformed traditional APF by 8.73%.	insufficient statistical analysis, no dynamic obstacles.
PCAPF [132]	the problem is a quintic polynomial curve, AFP implements constraints, solved by fmincon MATLAB toolbox.	gave acceptable paths. other path planning	no comparison with any algorithms.

comprehensive to judge the significance level. Furthermore, the dynamic obstacles are not considered [131].

Wang et al. designed a hybrid algorithm between the APF algorithm and the Polynomial Curve, named (PCAPF) algorithm. In the PCAPG algorithm, the path planning problem is formulated as a quintic polynomial curve optimization problem. Then, the APF algorithm is applied to formulate the objective function considering the obstacles. Finally, the Fmincon MATLAB toolbox is used to solve the optimization problem. The results showed that the algorithm gave acceptable paths, but the algorithm is not compared with any other path planning algorithm [132]. Gradient-based methods are summarized in Table 6.

The future direction for Gradient-Based (Artificial Potential Field, APF) methods in autonomous driving systems should focus on overcoming the challenge of local minima problems due to the gradient behavior. Integrating APF with global search and optimization techniques is a suitable way to provide more diversity to the search space and reduce the local minimum problem. Developing hybrid systems that combine APF with other path planning methods, like Rapidly-exploring Random Trees (RRT), can offer a more comprehensive solution where the other algorithm finds the global path. At the same time, the APF can avoid the dynamic obstacles in the path.

7.4. Optimization-based methods for path planning in ADS

Optimization algorithms, or mathematical programming, are methods to find numerical solutions for optimization problems. Optimization problems can be formulated as the maximization or minimization of the objective function. It is called convex programming, where the objective function should be minimized, while it is called concave programming, where the objective function should be maximized. Quadratic programming is convex programming in which the objective function contains a quadratic term. Quadratic programming has been applied in the literature to address the path planning problem in the ADS as a convex (minimization) optimization problem that needs to minimize the path length [133].

The stochastic optimization problem is an optimization problem that has a random term in the objective function or has a random input in the search process. Model Predictive Control (MPC) model is an example of a model that deals with random measurements and inputs and is formulated as a stochastic optimization problem. Dynamic programming is an approach that is used to address the stochastic optimization problem with unknown model parameters. The path planning problem in the literature can be formulated as a Model Predictive Control (MPC) stochastic optimization problem, which can be solved by dynamic programming [134]. Generally, optimization methods can deal with complex search spaces and give high-quality solutions, but they have high computational times.

Viana et al. proposed a Distributed Model Predictive Control (MPC) that implements the Human Driver Model integrated with Mixed-integer quadratic programming (MIQP), named the HDM-MIQP algorithm. The concept of the HDM-MIQP algorithm is to integrate the Human Driver Model (HDM) into the MPC model. Then, the problem can be assumed to be a mixed-integer quadratic programming (MIQP), which can be easily solved by the optimization tool of CPLEX IBM. The algorithm managed to find acceptable paths in multiple scenarios. However, the algorithm is not compared to any other algorithm. In addition, the HDM errors and uncertainties were assumed to be constant values [135].

Kanchwala applied the HDM-MIQP algorithm introduced in [135] under the same project supported by Innovate UK. He applied different vehicle dynamics to the bicycle model, which is widely used. He implemented the Human Driver Model (HDM) using a CarSim driving simulator. He obtained acceptable results, but HDM uncertainties and errors were assumed to be fixed values. Moreover, the algorithm is not compared to any other algorithm [136].

There are some hybrid algorithms between the sampling-based algorithms and the quadratic programming algorithms mentioned before in the sections of the sampling-based algorithm. The SB-SQB algorithm is a

hybrid algorithm between the sampling-based method and the Sequential Quadratic Programming (SQP) [97]. Furthermore, the S-NO algorithm is based on the sampling-based algorithm in [100] and quadratic programming to solve the convex optimization problem [101].

Oliveira et al. applied the Sequential Quadratic Programming (SQP) optimization algorithm, where the motion planning is implemented as Sequential Quadratic Programming (SQP) optimization solved by the CVX package. The SQP problem is based on the road-aligned car model that takes the distortion of the car and obstacles. Moreover, a new approximation method is used to capture this distortion. The simulation results proved that the SQP-based algorithm performs better than the one obtained from the algorithm proposed in [137]. However, the approximation error was not taken into consideration. The statistical analysis is insufficient to judge the performance [138].

Zhu et al. proposed a Parameterized Curvature Control (PCC) algorithm where the cubic spline interpolation technique is used as a parameterization method. The cubic spline is better than a single polynomial because the cubic spline can represent all possible curves with curvature. Then, the splined optimization problem is solved by the SQP algorithm. The results proved that the PCC algorithm obtained suitable safe paths in a dynamic environment. However, more complex scenarios need to be validated, and the whole algorithm is not compared to any other algorithm in literature [139].

Zhang et al. designed a novel Two-Step Quadratic Programming (TSQP) optimization algorithm to improve the path planning in the ADS. This method consists of two stages: first, a smooth line is obtained as a driving guide by a smoothing procedure; second, the optimal trajectory is obtained by a path optimizer based on piecewise-jerk formulation. The TSQP problem is resolved by the Operator Splitting Quadratic Program (OSQP) tool. The algorithm has been validated in the Baidu Apollo Platform and is used for a road test on hardware. Nevertheless, more scenarios need to be tested. The algorithm is not compared to any other algorithm in literature [140].

Changhao et al. developed a Dynamic Programming-based algorithm integrated with Clothoid Curve (DPCC) algorithm. The concept of the algorithm is to obtain future T samples and then fit them into a clothoid curve path by solving it as a dynamic programming problem. This algorithm managed to find an acceptable path with acceptable performance. However, more scenarios need to be tested. Moreover, the parameters of the algorithm were not optimized. Furthermore, the algorithm is not compared to any other algorithm [141].

Li et al. innovated a hybrid algorithm between the Quadratic Programming Optimization-based algorithm and the Search-based algorithm, named the SQPO algorithm. First, Search-based algorithms are used to find a rough path based on two cases: If the environment is known, the Dijkstra algorithm generates the path. If the environment is unknown, the Space Exploration Guided Heuristic Search (SEHS) algorithm will be applied to find the path. Second, path planning is represented as a Quadratic Programming Optimization (QPO) problem, which can be resolved by the OSQP tool. The QP algorithm is applied to obtain a trajectory with high quality. The results showed that the SQPO algorithm performs better than the algorithm proposed in [142]. Nevertheless, the interaction uncertainty is not taken into consideration. Moreover, only graphical statistical analysis was used to judge the performance [143].

Hu et al. introduced the Event-Triggered Model Predictive Adaptive Dynamic Programming (ET-MPADP) algorithm for road intersection driving scenarios. The ET-MPADP algorithm is based on the integration between the control policy generation of the MPC controller and the mismatch of cost function approximation using the critic–actor scheme. The ET-MPADP algorithm outperformed the time-triggered MPADP (TT-MPADP) algorithm. However, the structure of regression needs to be considered. Moreover, the unknown motion of the obstacles is not considered as well [144].

Wang et al. designed a hybrid algorithm among sampling-based algorithm, Dynamic Programming, and Quadratic Programming (SBDP-QP) algorithms. The sampling-based algorithm generates the graph. Then, the Dynamic Programming technique is deployed to generate an initial path. Finally, the quadratic programming optimization problem is solved to get the optimal trajectory considering all the driving constraints. The results proved that the SBDP-QP algorithm can generate a smooth collision-free path. However, the algorithm is not compared with other algorithms in the literature, and more scenarios need to be tested [145].

Typaldos et al. used the Feasible Direction Algorithm integrated with the Dynamic Programming algorithm (FDA-DP). The DP algorithm is used to give an initial guess path. Then, the FDA algorithm is used to find a numerical solution to the path planning problem. Moreover, connected vehicles can exchange information about the last obtained trajectory so far in real-time. The FDA-DP algorithm is used within the MPC controller framework. The FDA-DP algorithm gave better results than the one obtained from manually driven vehicles and the Forward Dynamic Programming (FDP) algorithm. Nevertheless, synergistic effects are not considered, and more scenarios need to be validated [146].

Jiang et al. introduced a quadratic programming-based path planner, the QP-SQP algorithm, used in a static, cluttered environment. The algorithm has two main stages: first, a collision-free guideline is generated by Quadratic Programming (QP); second, the optimal path is generated with the help of the guideline using the SQP technique. The QP and SQP optimization problems were solved by the Open Source Quadratic Programming (OSQP) solver. The simulation results indicated that the QP-SQP algorithm is better than Piecewise Jerk Method (PJM) and Minimum Snap Method (MSM). However, more scenarios should be tested with deep statistical analysis [147]. Optimization-based methods are summarized in Table 7.

Optimization-based methods are mainly time-consuming, with high computation time methods based on using optimizers to solve the optimization problem. Therefore, they are problem dependents, as one optimizer can work with a driving scenario while it fails with another one. Therefore, the future direction for utilizing this method is via hybridization with other techniques to reduce computational complexity while improving real-world adaptability. In practice, developing hybrids that combine optimization techniques like Quadratic Programming and Model Predictive Control with sampling-based methods can offer more efficient and practical solutions.

7.5. Interpolation curve methods for path planning in ADS

The interpolating curve methods are techniques used to generate a new path from an existing one to avoid collisions [12]. Initially, we start with a path that already exists from a global planner or another local planner. This path can lead to a collision with an object. Then, the interpolating curve methods modify the path to avoid this object. A new sub-path will be created by fitting a new set of collision-free points between a re-entry point and an exit point on the original path. The obtained path is collision-free and smooth. Nevertheless, the computational time is high in comparison with the other methods. The most commonly-used curve families are Bezier curves [148], clothoid [149], splines [150], and polynomials [151].

Wang et al. developed a hybrid algorithm between Stackelberg Differential Game Theory and Polynomial Curve (SDGT-PC) algorithm. The Stackelberg Differential Game Theory algorithm is deployed to get the global path. The polynomial curve method is applied to remove the errors in the global trajectory and make the path smoother. The experiment and simulation results proved that the SDGT-PC algorithm could obtain feasible paths. Nevertheless, it is not compared to the other algorithms in the literature; only the control algorithms were compared [152].

Hu et al. designed a dynamic path planning algorithm using the cubic spline (DPP-CS) interpolation method. The algorithm considered the dynamic obstacles. However, it considered only one obstacle, and

Table 7
Literature Review Summary for the optimization-based (quadratic/ dynamic programming) Path Planning techniques.

Algorithm	Idea	Pros	Cons
HDM-MIQP [135]	Human Driver Model is integrated with MPC, MIQP is solved by CPLEX IBM.	managed to find acceptable paths in multiple scenarios.	HDM errors and uncertainties are not considered, no comparison.
SB-SQB [97]	hybrid; Upper SB: finds the paths based on environment, Lower SQP optimizes path.	obtained reasonable paths within 50 ms.	no comparison to other algorithms, needs more complex scenarios.
S-NO [101]	hybrid; sampling alg. [100] for lateral path, QP for longitudinal path and diversity.	managed to find a comfortable, safe path in real-time.	No comparison with other algorithms.
HDM-MIQP [136]	same alg. in [135], HDM simulated by CarSim, vehicle dynamics is bicycle model.	obtained acceptable results.	HDM uncertainties and errors were constants, no comparison.
SQP-based [138]	SQP solved by CVX, road-aligned vehicle model with approximator for vehicle–obstacle distortion.	better than the algorithm in [137].	approx. error not considered, poor statistics.
PCC [139]	cubic spline is used as parameterization optimization method, then solved by SQP.	managed to find suitable paths in a dynamic environment.	more complex scenarios, no comparison.
TSQP [140]	1-smooth line is generated as driving guide, 2- generates optimal path, solved by OSQP.	released in Baidu Apollo Open Platform.	more scenarios need to be tested, no comparison with other algorithms.
DPCC [141]	get future samples, fit them into clothoid path by solving a dynamic programming problem.	managed to find acceptable path.	needs more scenarios, parameters were not optimized, no comparison.
SQPO [143]	Dijkstra and SEHS are used to find rough path, QP solved by OSQP optimizes the path.	better than the algorithm in [142].	interaction uncertainty not considered, only graph statistics.
ET-MPADP [144]	hybrid; control policy generation (MPC) and cost function approx. using a critic–actor scheme of ADP.	better than TT-MPADP and SPSO2011-GM algorithms.	regression structure and obstacles unknown movements are not considered.
SBDP-QP [145]	Sampling-based gets search graph, DP finds initial path, QP finds the optimal constrained path.	generate smooth collision-free path.	not compared with state-of-the-art algorithms, need more scenarios.
FDA-DP [146]	DP generates initial path, FDA finds numerical optimal path, data exchange among cars.	better results than manually driven cars and FDP algorithm.	no synergistic effect, more scenarios need to be validated.
QP-SQP [147]	guide line is generated by QP, then path is generated by SQP, solved by OSQP solver.	faster, better than PJM and MSM.	more scenarios need to be tested with deep statistics.

the algorithm needs to be validated against other algorithms in the literature. Moreover, the algorithm is not validated in real life [153].

Klanvcar et al. introduced a hybrid algorithm between the continuous Bernstein–Bézier (BB) interpolation technique and the discrete grid-based E* search algorithm, named the HE* algorithm. The HE* algorithm has two stages: first, the E* algorithm is used to calculate the direction-guiding heuristics; second, the quality of the search is improved in the second stage by a continuous expansion of the fifth-order Bernstein–Bézier (BB) curve. Finally, the Complete Node Mechanism (CNM) guarantees the complete bounded path. The HE* algorithm outperformed the HA* algorithm and SST* algorithm. However, there is no path of re-planning in case of a dynamic environment [154].

You et al. introduced two algorithms based on the Bézier curves algorithm: the first one is called the Joint Quadratic Bézier Curves (JQBC) algorithm; the second one is called the Fourth-Order Bézier Curves (FOBC) algorithm. The JQBC algorithm uses a smaller search space to find the path, but it is not easy to track because the path is C^1 continuous. The FOBC algorithm makes it easier to track its paths because it is C^2 continuous and can generate smoother paths. The dynamic cell (DC) concept is introduced to modify the traffic state dynamically. The results showed that the algorithms can find smooth, acceptable collision-free paths. However, the JQBC algorithm is not compared to the other algorithms in the literature, and more scenarios need to be validated [155].

Sedighi et al. proposed the Direct Visibility Diagram Algorithm with Clothoid Curves (DVD-CC) algorithm as a path planning algorithm. The Direct Visibility Diagram (DVD) algorithm is deployed to obtain the optimal holonomic path. The Clothoid Curves technique has been applied to get a smoother non-holonomic path. The DVD-CC algorithm is faster than standard Visibility Diagram (VD), RRT, and Hybrid A* (HA*) algorithms by (60%). Nevertheless, more scenarios should be validated, and the dynamic obstacles were not considered [156].

Piscini et al. proposed a Clothoid-Based Algorithm (CBA) based on finding the trajectory given two directions and two points as a G^1 interpolation problem solved by the clothoid curves. The results indicated that the CBA algorithm outperformed the Preview-based Algorithm and Pursuit-based algorithm. However, More scenarios should be tested, and dynamic obstacles are not considered [157].

Moreau et al. proposed the Bézier Curve Optimization (BCO) algorithm as an interpolation approach for the path planning problem. The BCO algorithm considers the traffic and obstacles constraints as a constraint Bézier curves optimization problem. The Newton–Raphson method and the Lagrangian algorithm have been applied to resolve this optimization problem. The BCO algorithm managed to obtain a safe path free of collision. However, the BCO algorithm has not been compared to the state-of-the-art algorithms, and more scenarios need to be tested [158].

Zambon et al. introduced the B-Spline Curve Optimization (BSCO) algorithm to deal with trajectory planning in the ADS. The concept of the BSCO algorithm is to obtain smooth paths that a combination of B-spline linear functions can describe. Furthermore, based on the minimum distance to obstacles, a smooth penalty function is used to convert a constrained optimization problem to an unconstrained one. The algorithm managed to find smooth paths. Nevertheless, the algorithm is not compared to other algorithms, and the sub-optimal solution depends on the tuning parameter [159].

Mu et al. developed an improved PSO (IPSO) algorithm to address the three-dimensional route planning problem. A penalty term has been added to the fitness function, and They made the paths smoother using cubic spline interpolation. Results showed that the IPSO algorithm gave shorter paths by 90% of those obtained from the PSO algorithm. However, the statistical analysis needed to be more comprehensive to judge the performance in a complex environment. [160].

Kamil et al. designed the Adaptive Dimension Limit-Artificial Bee Colony (ADL-ABC) Algorithm. The concept of the ADL-ABC algorithm

is to apply an adaptive limit parameter instead of a fixed, stable limit. The cubic polynomial interpolation technique has been applied to make the path smoother. The ADL-ABC algorithm outperforms the traditional ABC algorithm. However, more complex scenarios should be tested to judge the significance level [161].

Luo et al. innovated the Gradient Descent and Bézier Curve (GDBC) algorithm as a post-optimization method. The GDBC Algorithm consists of two steps: first, the gradient descent algorithm (GDA) is deployed to utilize the path away from obstacles; second, the Bézier Curve interpolation technique has been applied to make the path smoother. The results showed that the GDBC algorithm outperformed the traditional Gradient Descent Algorithm (GDA). However, more scenarios need to be validated, and only graphical statistical analysis is used, which is insufficient to judge the significance level. Moreover, this is not an independent path planning algorithm, but it can be applied to enhance the path obtained from a traditional path planning algorithm [162].

Jin et al. proposed a coupled Longitudinal and Lateral planning with Clothoid interpolation (LLC) algorithm. The algorithm starts by designing the speed profile of the longitudinal horizon. Then, the Clothoid interpolation algorithm has been deployed to obtain a set of local paths. After that, the best path is selected based on the objective function, but if all candidate paths fail, a re-planning algorithm is used for higher planning success. The results showed that the LLC algorithm outperformed the traditional discrete methods. Nevertheless, the path-tracking method is simple, with no adaptive planning. Moreover, only graphical statistical analysis was used [163].

Dai et al. applied the cubic spline interpolation technique to generate a smooth path in the ADS. This algorithm gave a smoother path than the one obtained from the cubic polynomial fitting algorithm. Moreover, the Frenet coordinate frame in the cubic spline algorithm simplified the solution compared to the Cartesian coordinate frame in the cubic polynomial algorithm. However, it is slower than the algorithms in the state-of-the-art [164].

Feher et al. created a hybrid algorithm called HRL-MPC, a hierarchical RL algorithm, to obtain the optimal path. The twin Delayed DDPG reinforcement learning algorithm has been applied to generate the path. Next, the clothoid curves interpolation technique is deployed to make the path smoother. The CARLA results showed that the HRL-MPCC algorithm outperformed human drivers. However, no real-life tests are performed, and no comparisons are held against other state-of-the-art algorithms [165]

Lambert et al. implemented a Clothoid Curves Optimization (CCO) algorithm to generate an optimal path. The concept of the CCO algorithm is to formulate the path planning problem as a clothoid optimization problem with G^2 continuity constrained within the convex region. Then, a generic constrained non-linear solver has been applied to resolve the optimization problem and get a smooth path. Moreover, a weighting parameter b controls the peak's sharpness. The results proved that the CCO gave smoother paths than the cubic spline interpolation method. However, the CCO takes longer time than the cubic spline method. When b is more than 100, the convergence becomes poor. Furthermore, more scenarios should be tested in a broader range of environments [166].

Bulut developed the Quintic Trigonometric Bézier Curve (QTBC) algorithm, which is used to construct a predetermined trajectory based on the Quintic Trigonometric Bézier Curve with C_3 continuity. The two shape parameters adjust the predetermined path without altering any obstacle. The QTBC algorithm is compared with itself by changing the Bézier shape parameters: cubic Bézier, quintic Bézier, cubic trigonometric Bézier, and the proposed quintic trigonometric Bézier. The QTBC algorithm gave the best results. However, the algorithm is not compared to the other algorithms in the literature, and more scenarios need to be validated [167].

Li et al. created a hybrid algorithm between the Tentacle and the B-Spline Curve algorithms, named (TCSC) algorithm. This algorithm uses the Tentacle algorithm to find pre-calculated paths using virtual tentacles. Then, the B-spline algorithm generates a path based on the sampling area of the best tentacle path. Finally, the final trajectory is formed by segments of the best tentacle and the B-spline curve. The results proved that the TCSC algorithm outperformed the sampling-based path algorithm introduced in [168]. Nevertheless, the algorithm does not consider the uncertainty of the environment [169].

Horvath and Pozna introduced the Trajectory Following Approach (TFA) algorithm as an interpolation approach, in which the path is generated by a set of segments joined by clothoid curves. The results showed that the TFA algorithm can provide realistic approximation paths. Nevertheless, obstacles were not taken into consideration at all. Moreover, the algorithm was not compared with other algorithms in the literature [170].

Shentu et al. applied the Bezier Curve (BC-based) algorithm in a hybrid navigation ADS. The concept of using the Bezier curve is that the Bezier curve does not bypass all the points that define the path. The BC-based algorithm managed to find reasonable paths. However, the algorithm is not compared to any other algorithm, and more complex scenarios need to be tested [171].

Some hybrid algorithms based on interpolation methods are mentioned in previous sections. The DPCC algorithm is a hybrid algorithm that combines dynamic programming as an optimization method with Clothoid Curve as an interpolation method [141]. The PCC algorithm combines the Sequential Quadratic Programming with the cubic spline curve for a smooth path [139]. The A*-PDWA algorithm is a hybrid algorithm that integrates the A* algorithm as a graph-based algorithm and the cardinal spline interpolation method [86].

The i-RRT algorithm is another hybrid algorithm between the RRT algorithm as a sampling-based method and the B-spline interpolation algorithm [108]. MRRT* algorithm is a hybrid algorithm between the RRT* and the cubic B-spline interpolation algorithm [109]. IA*-DWA is a hybrid algorithm that integrates the DWA algorithm, the Improved A* algorithm as a graph-based algorithm, and the Bezier Curves as an interpolation algorithm [94].

Moreover, the IRRT algorithm is a hybrid between the RRT algorithm and the B-spline curve interpolation method [115]. The PCAPF algorithm is another hybrid algorithm between the APF algorithm and the polynomial curve interpolation method [132]. The PRRT-BSC algorithm integrates the Pruning RRT algorithm with the B-Spline Curve interpolation algorithm [116]. Curve Interpolation methods are summarized in Table 8.

As future research guidance, interpolation-based algorithms are too sluggish and fail to find a path if they are used as a stand-alone algorithm. However, their great value lies in smoothing out paths generated by other methods, such as graph-based and sampling-based algorithms. Their integration can significantly enhance path smoothness, particularly in routes involving sharp turns or complex maneuvers. By combining interpolation techniques like Bezier curves, splines, and clothoids with other path-planning approaches, the resulting hybrid algorithms can offer optimally smooth paths while maintaining computational efficiency and adaptability in dynamic environments. This approach can balance practicality and path quality in autonomous driving systems.

8. Layer 5 - part 2: Path planning (machine and deep learning techniques)

Machine Learning is the science in which a computer program can learn from the experience and can make predictions and future decisions based on this experience. In machine learning, a model is assumed with unknown parameters. Then, a machine learning algorithm is used to obtain these parameters. The machine learning process has three major phases: training phases, validation phases, and application phases. A machine learning algorithm gets the unknown model parameters in the training phase. In the validation phases, the model is tested to make sure that the model can obtain optimal results with acceptable

Table 8
Literature Review Summary for the Curve interpolation Path Planning techniques.

Algorithm	Idea	Pros	Cons
SDGT-PC [152]	Game Theory finds global path, polynomial curve removes errors and smooths the path.	can find feasible and effective paths.	not compared to the other state-of-the-art path planning algorithms.
DPP-CS [153]	dynamic path planning method based on the cubic spline interpolation	considered dynamic obstacles.	only one obstacle, no comparison, no real-life validation.
HE* [154]	hybrid; E* computes heuristics, Bézier curves improves path, CNM finds the final path.	outperformed HA* and SST* algorithms.	no path re-planning in the dynamic environment.
JQBC/FOBC [155]	2 alg. based on Bézier curves, FOBC is smoother, DC modifies traffic state.	can find smooth acceptable paths.	need more scenarios, not compared to other algorithms.
DVD-CC [156]	DVD finds holonomic path, Clothoid finds a smooth non-holonomic path.	faster than VD by 60%, RRT, and HA* algorithms.	more scenarios needed, no dynamic obstacles.
CBA [157]	clothoid solves G^1 interpolation problem given 2 directions and 2 points.	outperformed Preview-based and Pursuit-based algorithm.	needs more scenarios, no dynamic obstacles.
BCO [158]	constraint Bézier curves optimization is solved by Newton–Raphson with Lagrangian Method.	managed to find collision-free paths.	no comparison to other algorithms, needs more scenarios.
BSCO [159]	linear combination of B-spline basis functions, smooth penalty fn. for the unconstrained problem.	managed to find smooth paths.	sub-optimal path depends on tuning parameter, no comparison.
IBSO [160]	Improved BSO; added penalty term to fitness fun., cubic spline interpolation for a smooth path.	better than PSO algorithm by 90%	poor statistical analysis.
ADL-ABC [161]	improve ABC, used adaptive limit parameter, Cubic polynomial smooths path.	outperform the traditional ABC algorithm.	more complex scenarios need to be tested.
DPCC [141]	get future samples, fit them into clothoid path by solving a dynamic programming problem.	managed to find acceptable path.	needs more scenarios, parameters were not optimized, no comparison.
GDBC [162]	GDA optimizes the path, Bézier Curve smooths the path.	outperformed traditional GDA Algorithm.	more scenarios, only graphical statistics, only post-algorithm to improve paths.
LLC [163]	Clothoid generates multiple paths, cost fn. chooses the best path, re-plan if it fails.	outperformed traditional discrete methods.	simple path tracking with no adaptive planning, only graphical statistics.
Cubic spline [164]	apply the basic cubic spline interpolation algorithm	smoother and simpler than cubic polynomial algorithm	slower than the other state-of-the-art algorithms.
PCC [139]	cubic spline is used as parameterization optimization method, then solved by SQP.	managed to find suitable paths in a dynamic environment.	more complex scenarios, no comparison.
HRL-MPC [165]	DDPG algorithm generates the path, clothoid algorithm makes the path smoother.	outperformed human drivers.	no real tests, no comparison against other algorithms.
A*-PDWA [86]	Hybrid; A* gets global path, cardinal spline smooths path, P-DWA avoids obstacles.	managed to find safe path. managed to find a safe path.	algorithm is not compared with any other algorithm.
i-RRT [108]	i-RRT find the path, B-spline interpolation smooths path.	managed to find personalized safe trajectory.	no comparison with the state-of-the-art algorithms.
MRRT* [109]	MRRT* incrementally finds the path, cubic B-spline smooths the path.	managed to find collision-free paths in real.	no statistical analysis, no comparison with other algorithms.
CCO [166]	clothoid with G^2 continuity, sharpness control, obstacles constraints, solved by a generic non-linear solver.	smoother than cubic spline.	slower, poor convergence when $b > 100$, more scenarios.
QTBC [167]	gets path based on 2 shape Bézier parameters that modify path without obstacle change.	compared by changing shape parameters, QTBC is better than QB, CB, and CTB.	no comparison, need more scenarios.
TCSC [169]	Tentacle Alg. finds pre-calculated paths, B-spline finds path based on best Tentacle path.	better performance than the algorithm in [168].	environment uncertainty not considered.
TFA [170]	generated path by set of straight segments connected by clothoid curves.	can provide realistic approximated paths.	obstacles are not considered, no comparison to other algorithms.
PCAPF [132]	the problem is a quintic polynomial curve, AFP implements constraints, solved by fmincon MATLAB toolbox.	gave acceptable paths.	no comparison with any other path planning algorithm.
BC-based [171]	applied Bezier curve to find the optimal smooth path.	managed to find reasonable paths, real application.	need more complex scenarios, no comparison to other algorithms.
IA*-DWA [94]	hybrid; A* finds path, Cubic Bezier curves smooths path, DWA avoids obstacles.	outperformed A*, DWA algorithms.	needs testing on more complex environment and benchmark functions.
IRRT [115]	based on circular sampling, random point filter, B-spline smooths path.	outperformed RRT and B-RRT algorithms.	model-driven with limitations, needs more scenarios and statistics.
PRRT-BSC [116]	hybrid; RRT finds path by pruning fn. based on obstacles, B-Spline smooths the path.	managed to find acceptable paths.	not compared to any other state-of-the-art algorithm.

accuracy. In the application phase, the real-life model is used to make decisions.

Machine Learning algorithms are categorized into three main classes: supervised learning, unsupervised learning, and reinforcement

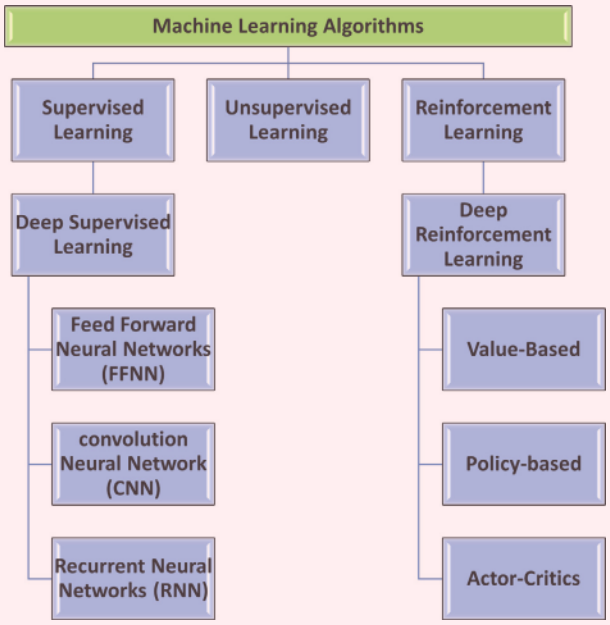


Fig. 3. Machine Learning algorithms classification.

learning. In supervised learning, labeled training data containing the input and the corresponding output is used to train the model and obtain the model parameters. It is called supervised because there is an output for each input used during the training phase. In unsupervised learning, unlabeled training data is used. The primary purpose of unsupervised learning is to cluster the input data or reduce its dimensions as a pre-processing stage for the data. In reinforcement learning, no training data is used at all. The agent model reacts to the environment via sensors to learn by trial-an-error technique.

Deep learning is a sub-branch of machine learning in which the assumed model is a neural network that consists of multiple layers between the input and the output. The powerful merit of neural networks is that they can act as a model approximator for a complex model that deals with massive data with multiple dimensions, such as images. The deep learning and neural network techniques can be applied in supervised learning as a model and can be applied in reinforcement learning as a Q-function approximator.

As shown in Fig. 3, machine learning techniques can be classified into three main categories: supervised, unsupervised, and reinforcement learning. This section focuses on deep-supervised, reinforcement, and deep-reinforcement learning because they are the most commonly used techniques in autonomous driving systems.

8.1. Deep supervised learning techniques

Deep learning techniques are mainly based on deep neural networks consisting of layers, each containing a set of neurons. There are three main types of neural networks: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Forward Neural Networks (FFNN). There are other neural network structures, but we mainly speak of the basic types. FFNN networks are the primary architecture of multi-layer neural networks [172]. They are usually deployed for classification and recognition problems as supervised learning applications so that they can be applied in lane detection and traffic road sign recognition in the ADS.

CNN networks are usually used to deal with images and recognize details inside images such that they can play an important role in the perception layer in the ADS. RNN networks usually deal with sequential data such as voice and time series. Deep supervised learning techniques can be used in two forms in the ADS:

- Deep neural networks such as CNN and RNN can be used in the perception layer to deal with sensory data such as images.
- Deep neural networks could be deployed as an End-to-End driving scheme starting with perception and ending with path planning.

Deep supervised learning techniques can give fast and accurate solutions in familiar scenarios. However, they require a considerable training dataset to build reliable models. Therefore, the main demerit of NN is the need for offline training. Moreover, the need to re-train the whole model to make any updates to the model is too time-consuming.

Song et al. developed a deep cascaded neural network as an End-to-End planning scheme. The proposed architecture combines an improved VGG neural network, a type of CNN network, and long short-term memory (LSTM). The proposed network is called IVGG-LSTM. In the IVGG-LSTM network, the IVGG layer has been used to obtain spatial features from the input images, whereas the LSTM layer is deployed to get temporary features of the input images. Then, the IVGG-LSTM model uses this data to map between the output parameters for the motion and the input images. The practical results proved that the IVGG-LSTM scheme outperformed the network introduced in [173] and the traditional VGG-16 neural network. Moreover, the IVGG-LSTM network can learn from the human driver and adapt to different roads. Nevertheless, this network is not applied to real roads, and the algorithm's speed needs further improvements [174].

Kiki et al. designed a novel technique based on a neural network to address path planning in the ADS. This method can predict feasible paths based on a gradient-based self-supervised learning algorithm. The algorithm can exploit the experience gathered in the past via the learning algorithm. It achieved solutions that are 14 times faster than the ones obtained from RRT* and State Lattices. However, the algorithm achieved only 74% precision on the benchmark validation [175].

Moraes et al. developed a real-time image-based method using the CNN neural network, named DeepPath, as an End-to-End driving scheme for the IARA self-driving car. The DeepPath receives the pose of the current car and an image as input and obtains the trajectory as an output. A CNN is used to infer the path model. Results proved that DeepPath can generate paths for IARA that are slightly different from the desired path. However, No comparison had been made to any other algorithm made [176].

Markolf et al. introduced a supervised learning approach based on Nonlinear Optimal Control (NOC) theory to get approximate solutions for the path planning problem, named NN-NOC. This approach solves nonlinear optimal control problems to generate training data. Then, these data have been deployed to train a neural network. Simulation results showed that this model performed well. Nevertheless, the NN-NOC algorithm is not feasible for real-time applications and has high computational time. Furthermore, no comparison is made to judge the relative performance [177].

Guo et al. designed an LSTM neural network as a supervised learning approach to find the shortest path. The LSTM neural network is trained by training data from a fuzzy control algorithm. The LSTM-based architecture outperformed the BP neural network and the fuzzy control algorithm. However, the dynamic environment is not considered, and the network needs further improvements to be adaptive [178].

Sakurai et al. innovated a Spiking Neural Network (SNN) algorithm consisting of two agents; one is to approach the goal, and the other is to avoid obstacles. The agents can learn and deal with dynamic objects through dynamic graphs using SNN properties. The SNN algorithm outperformed the threshold-based agent algorithm. Nevertheless, the algorithm needs more validation using more complex obstacles [179].

Wang et al. created a trajectory learning algorithm using deep neural networks named CNN-Raw-RNN. This scheme is similar to End-to-End driving, but the difference is that the output of the proposed scheme is the trajectory instead of vehicle control actions. The system comprises two main parts; the first part is a hybrid neural network

between CNN and RNN, named the pilot, which receives the frames of the video input and constructs the future path. The second part is called the Copilot, which receives the pilot's future path, verifies the path's safety, and generates the results of the semantic segmentation perception to update the pilot mechanism. Then, the trajectory can be easily mapped to the control actions of the car. The algorithm outperformed the CNN and CNN-LSTM architectures. However, the algorithm needs to be verified on more scenarios and compared with more AI algorithms [180].

Kalathi et al. deployed neural networks to address the path-planning problem of road sign recognition. They implemented an autonomous car prototype that can drive on different tracks. This paper claimed that the neural network can determine the path. However, the neural network in this proposed structure is only used for perception and recognition. Therefore, more work is needed on the path planning algorithm [181].

Lee et al. proposed a Depth-wise Separable UNet for Path Prediction (DSUNet-PP) algorithm for End-to-End learning in the ADS. In this algorithm, a CNN network, UNet, is combined with a path prediction algorithm based on a quadratic polynomial to maintain lane centering. Results showed that the DSUNet-PP outperforms UNet-PP in a dynamic environment. However, the DSUNet-PP takes about five images to generate an acceptable path within a range of 10 m. Therefore, image processing has high computational time [182].

8.2. Reinforcement learning techniques

The recent reinforcement learning techniques in the path planning of the AD system are discussed in this section. In reinforcement learning, no training data is used at all. The agent model communicates with the environment via sensors to learn by trial-and-error technique [183]. The agent's performance is evaluated during the training process based on the reward function. In each state, the agent takes an action that will make the agent move to another state. If the new state makes the agent move towards the goal, the agent receives a good reward. Otherwise, the agent will not receive a good reward. The objective of the RL process is to maximize the cumulative reward as an optimization problem [184].

The main merit of the RL methods is that they can deal with complex problems and obtain reasonable solutions in the long term. The model can be learned from error and modified during run time. However, the main challenge in the RL is the compromise between exploitation and exploration. In exploitation, the agent needs to take the best action from a set of known actions to exploit and choose the best action with the best reward. On the other hand, the agent sometimes needs to take new actions, which can lead to better reward than the one obtained from the known actions, which is called exploration [185]. Therefore, parameter tuning is a challenging problem. Moreover, generating a state–action table will be too massive in complex problems, which require more memory and high computational time to search for suitable action.

The reinforcement learning process can be formulated as Markov decision processes (MDPs). The MDP agent is composed of a set of actions A, a set of states S, and a reward function R. $R(s,a)$ indicates the value of the reward when the agent commits an action $a \in A$ at a specific state $s \in S$. A policy π is a set of concussive actions. The objective is to obtain a policy that maximizes cumulative rewards for its actions. The Q-function is an action-value function representing the cumulative rewards for actions taken in specific states. The target is to maximize the Q-function to obtain the best policy. The concept of the RL is shown in Fig. 4. In RL, the $Q(a,s)$ indicates the reward for a specific action a taken at a specific state s. The Q values for specific actions taken at specific states are stored in a Q table. This table represents the experience gained by the agent. The RL methods have been applied and enhanced to improve the path planning solutions.

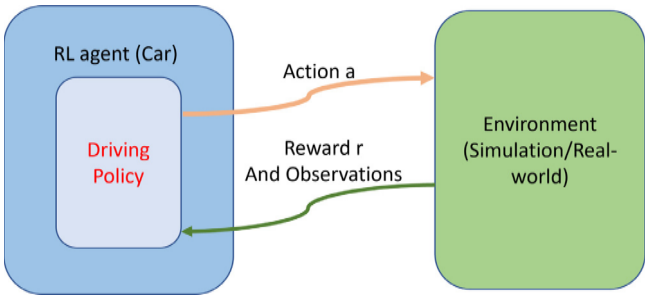


Fig. 4. Concept of Reinforcement Learning.

Yao et al. introduced a hybrid algorithm called the Reinforcement Learning Multi-Objective Hyper-Heuristic algorithm (RL-MOHH) to obtain the safest best trajectory in a smart city. A parallel version of the algorithm is also proposed called (RL-PMOHH). The Heuristic algorithm finds a set of new routes, and then the Reinforcement Learning (RL) algorithm selects the best route. Results showed that the RL-PMOHH algorithm is faster than the NSGA-II and the PNSGAII algorithms. However, both the RL-MOHH/RL-PMOHH algorithms need more online training because they could only get a maximum of 80% optimal solutions [186].

Liu et al. proposed a new hybrid algorithm between Reinforcement Learning (RL) and A* algorithms, called OPABRL, specially designed for congestion and accident scenarios. The RL algorithm is used to choose the best trajectory with length priority. At the same time, the A* algorithm has been applied to improve the trajectory generated by the RL algorithm. Results proved that the algorithm can give the shortest path in a short run time. However, the algorithm is not tested on a significant complex problem where the maximum number of nodes is only 30. Therefore, the performance of complex problems needs to be tested [187].

Chen et al. designed a hybrid algorithm between the greedy selection algorithm and the Q-learning algorithm to find the optimal path in the ADS in a static environment and reduce fuel consumption. The algorithm has been compared with the K-shortest and Dijkstra algorithms, outperforming them in terms of fuel consumption and shortest path. However, they did not deal with the situation of an unknown environment [188].

Wang et al. implemented a hybrid technique, named (QLOR), between the Q-learning algorithm and the Linear Output Regulation algorithm (LOR) in which a Q-learning algorithm has been applied to obtain the optimal trajectory. At the same time, the LOR algorithm has been deployed to design the tracking controller. The algorithm succeeded in two scenarios, but the dynamic obstacles were not considered. Moreover, no comparison had been made to other algorithms [189].

Wang et al. developed the Augmented Adversarial Inverse Reinforcement Learning (AAIRL) algorithm, where the traditional AIRL algorithm is improved by augmenting it with semantic rewards in an interactive reinforcement learning framework. The AAIRL algorithm is compared with four baseline RL algorithms. Simulation results showed that the AAIRL algorithm outperformed all other methods. However, the AAIRL algorithm has yet to be compared to other powerful algorithms in state-of-the-art [190].

Liu et al. innovated a Hybrid algorithm between Reinforcement Learning and the A* Algorithm (HARL) algorithm. The A* algorithm generates a reference path optimized based on the Markov process in the reinforcement learning algorithm using prior knowledge. Results proved that the HARL algorithm outperformed the PSO, ACO, GA, and Artificial Neural Networks (ANN) algorithms. Nevertheless, the results of the number of turns obtained from the HARL algorithm could be better. Moreover, the storage space of the prior knowledge for reinforcement learning is massive [191].

Liu et al. introduced a hybrid algorithm between the PSO and Reinforcement Learning (RL) algorithms. The PSO algorithm has been

used to optimize the hyper-parameters of the RL algorithm to make the algorithm faster. Results proved that the hybrid algorithm can provide better solutions. The main drawback of this algorithm is that the parameters of the PSO optimizer are static. Moreover, the statistical analysis is based only on graphical methods, so more numerical analysis is required to judge the confidence level [192].

Kim et al. used a real-time Q-learning (RTQL) technique to address ADS's real-time path planning problem. The simulation results showed no difficulty in reaching the destination point in various scenarios. However, no comparison is made to judge the algorithm's performance among the recent state-of-the-art algorithms [193].

Chang et al. created an Improved Dynamic Window Approach using the Q-Learning (IDWAQ) algorithm. The Q-learning algorithm is used for motion planning. Then, the Q-learning algorithm learns the DWA parameters to adapt to the dynamic environment. Simulation results showed that the improved DWAQ is better than the original DWA. However, dynamic obstacles are not considered, and more complex obstacles need to be tested [194].

Low et al. developed an Improved Q-learning (IQL) to address the trajectory planning in the ADS. The original Q-learning algorithm is improved by adding a distance metric, a virtual target concept, and a Q-function that overcomes dead-ends. Results showed that IQL outperformed QL in all maps, which is better than RRT and VG in most cases. The drawback of IQL is that it is time-consuming during escape mode. Moreover, dynamic obstacles are not considered [195].

Rousseas et al. presented a hybrid algorithm between the Integral Reinforcement Learning (IRL) algorithm and the Artificial Harmonic Potential Fields (AHPFs) algorithm. The AHPFs method generates the path, while the IRL algorithm enhances the path via interaction with the environment. The algorithm outperformed the RRT* algorithm. However, it neglected the 3D and dynamic environment [196].

8.3. Deep reinforcement learning techniques

As described in traditional reinforcement learning, the Q-table represents the reward corresponding to the action taken at a specific state. This table represents the (state, action) space and its corresponding reward (Q-value). However, getting the optimal policy will be complicated when the (state, action) space is vast. Moreover, the Q-table will consume more memory space. Therefore, the benefit of the neural network will arise. The neural network can act as a Q-function approximator to predict the (state, action) pair, called the Q-value. Due to the flexibility and the multi-input multi-output characteristics of the neural network, it can fit any complicated state–action space by adding more hidden layers as required, called Deep Reinforcement Learning (DRL). Nevertheless, this adds more parameter tuning complexity to train the neural network and balance the exploration and exploitation process. DRL comprises three sub-categories: policy-based DRL, value-based DRL, and actor–critics DRL.

8.3.1. Value-based/policy-based methods

Reinforcement learning is concerned with obtaining the optimal policy with the best reward. To understand the difference between policy-based and value-based methods, the difference between value and policy should be clear. The policy is the set of actions that should be taken to reach the goal. The value represents how good the policy is. The conversion between the policy and the value is a two-way relationship. Given the policy, we can get its value using the Bellman operator (Q-function). On the other hand, given the value, we can get the policy using the optimality Bellman operator. This process is explained in Fig. 5.

In policy-based algorithms, the algorithm keeps an eye on improving the policy itself and then evaluates its value. It starts with a random policy and then finds its corresponding value using the Bellman operator. During iterations, the policy will be improved by using the previous policy, then the value of the improved policy will be evaluated using the Bellman operator, and so on. The new policy is guaranteed to be a strict enhancement of the previous one [197].

In value-based algorithms, the algorithm keeps an eye on the value function. It starts with a random value function (Q-function). Then, the algorithm gradually maximizes the value function until it reaches the optimal value of the Q-function. Then, the optimal policy can be generated from the obtained optimal value function using the optimality Bellman operator, which contains a max nonlinear operator. Therefore, value-based and policy-based algorithms have the same working principle to get the optimal policy with the optimal value. The policy-based technique can obtain the optimal policy and then evaluate its value, while the value-based technique can obtain the optimal value and then reduce the optimal policy. The value-based algorithms are the most commonly used in literature to address the path planning in the ADS using the Q-learning algorithm [198].

Bernhard et al. designed the Experience-Based Heuristic-Search (EBHS) algorithm, a hybrid heuristic search, and the Deep Q-Network. The DQN represents experiences and is used as a heuristic function in a heuristic search algorithm. The results proved that the EBHS algorithm is better than the DQN and Hybrid A* algorithms. However, the algorithm lacks generalization, and it is not used in dynamic environment [199]

You et al. introduced the Maximum Entropy Deep Inverse Reinforcement Learning algorithm (MEP-DIRL). The algorithm relies on a stochastic Markov decision process (MDP). The reward function is designed using a Deep Neural Network (DNN) trained via the Maximum Entropy Principle (MEP). Simulated results demonstrated desired driving behaviors. However, it is not applied in real life and is not compared to other algorithms to compare its performance [200].

Liao et al. applied the Double Deep Q-network (DDQN) algorithm to find the best path when overtaking vehicles on the highways. The DDQN algorithm is compared to the traditional Deep Q-Network (DQN) Algorithm. Simulation experiments showed the merits of the DDQN algorithm in terms of the convergence rate and safe overtaking. However, online hardware-in-loop (HIL) experiments are not considered. Furthermore, no comparisons are made to the other state-of-the-art optimizers [201].

Zhao et al. just applied the Double Deep Q-learning Network (DDQN) to train a path planning policy under an interactive highway environment. The SUMO simulator is used to simulate an environment in which the traffic distribution and the speed can be manipulated easily. The DDQN algorithm achieved good paths without collisions. However, the algorithm is not compared to any other algorithm [202].

Liao et al. introduced a hybrid technique between the Potential Field (PF) algorithm and the Deep Reinforcement Q-Learning Algorithm deployed in unknown environments. This algorithm can give better solutions than the ones obtained from the original RL algorithm. However, the best success rate of the hybrid algorithm reached only 88.62% for 10000 comparisons [203].

Chen et al. innovated a hybrid algorithm between the Fuzzy control concept and the Deep Q-network (DQN) Algorithm, named the Conditional Deep Q-network (CDQN) algorithm. This algorithm has been used to resolve the directional path planning problem in an End-to-End driving scheme of the ADS. The global path is generated from the DQN algorithm. Then, a defuzzification output layer in the DQN is used to improve the motion commands. The algorithm is simulated via the CARLA simulator and compared to DDQN and A3C algorithms. Results demonstrated that the CDQN algorithm outperformed the others. However, the obstacle avoidance task needs to be addressed [204].

Wen et al. presented a hybrid algorithm called Dueling Deep Q-Network with a Fast active SLAM (DDQN-FSLAM) algorithm operated in an unknown environment. The DDQN algorithm is used to construct a path using the DRL technique. At the same time, the FSLAM method has been applied to construct a 2D map of the surrounding environment and recognize the obstacles. The natural and virtual results showed the algorithm's success in a different environment. Nevertheless, the

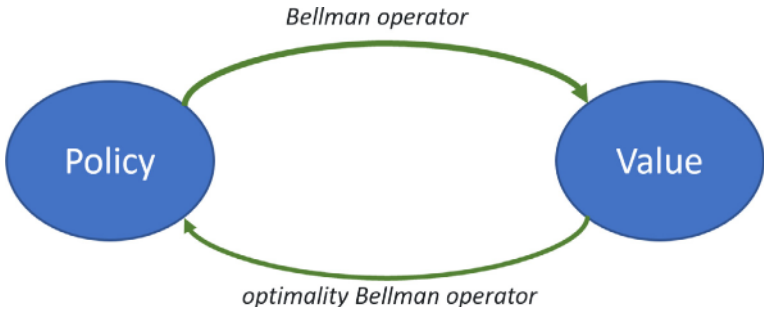


Fig. 5. Relationship between policy and value.

algorithm is not compared to any other algorithm. More improvements are required for deep reinforcement learning to reduce the time to finish the task [205].

Li et al. implemented a Deep Q-Network algorithm to get a suitable path through intersections. In this study, an End-to-End AD scheme is assumed. The first stage of the neural network is a CNN to collect images and implement the perception process. Then, a Deep Q-Network is followed to obtain the optimal driving policy via environmental interaction. This scheme is implemented successfully. However, this algorithm is not statistically compared with the well-known algorithms to judge its performance [206].

Sainath et al. implemented a hybrid neural network architecture to simulate the ADS system as an End-to-End driving scheme. The part of the neural network responsible for path planning is the DQN network, which is learned by the Deep Reinforcement Learning algorithm. This study implemented deep learning and AI techniques for developing autonomous driving systems. However, statistical analysis has yet to be performed to judge the algorithm's performance [207].

Naveed et al. designed a Robust Hierarchical Reinforcement Learning (RHRL) architecture. The traditional Hierarchical Double Deep Q-learning (hDDQN) is improved by deploying an LSTM network layer in the proposed network to interact with the environment and discover noise. Moreover, Hybrid Reward Mechanism and Reward-Driven Exploration are used to improve efficiency. CARLA simulation results showed that the RHRL algorithm outperformed the Vanilla DQN (VDQN) algorithm and Hierarchical Double Deep Q-Learning (hDDQN) algorithm. However, more scenarios need to be tested [208].

Li et al. applied Deep Reinforcement Learning in an End-To-End driving scheme (IDQNPER-ETE). The End-to-End architecture begins with a hybrid deep neural network consisting of CNN and LSTM and is used to gather and process multi-sensor data. Next, the DQN network is applied to generate the path using the IDQNPER algorithm. The algorithm managed to achieve acceptable performance in a static environment. Nevertheless, the algorithm is not compared to any other algorithm. Moreover, dynamic environments are not considered [209].

Peng et al. proposed the DRL-GAT-SA algorithm to find an optimal and safe path in autonomous driving systems. This algorithm combined deep reinforcement learning with a graph attention network into a controller called Graph Attention Reinforcement Learning (GARL), which relies on the safety field model. The algorithm outperformed the DDQN algorithm and DGN algorithm under two different scenarios. However, no real-life tests are performed, and more complex scenarios are required [210].

Perez et al. implemented and compared two Deep Reinforcement Learning algorithms as a machine learning approach to find the shortest path for the car. The two DRL methods are the traditional DQN network and the Deep Deterministic Policy Gradient (DDPG) algorithm. CARLA Simulator is used to simulate the training and validation processes. Both the DDPG and the DQN algorithms can safely reach the destination, but DDPG performs better and is more similar to human driver performance. The main drawback is that the algorithms were not tested on real-life maps in CARLA [211].

8.3.2. Actor-critic methods

The Actor-critic algorithms are a hybrid architecture between the value-based and policy-based schemes. There are two networks connected in series: the first network is called the actor, which generates the policy (action) based on the current state and observations; the second network is called critics, which receives the actions and optimizes the reward value based on the policy obtained from the actor.

The HRL-MPC algorithm is a hybrid algorithm mentioned in the interpolation-based algorithms section. It combines the clothoid curves interpolation algorithm with the hierarchical actor–critics RL algorithm [165]. Wang et al. designed a combined reinforcement learning algorithm called Monte Carlo Search and actor-critic (MCS/AC) algorithm to solve path planning in the ADS. The simulation results from the Simulation of Urban Mobility (SUMO) simulator and the SMARTS simulator proved that they showed that the MCS/AC algorithm outperformed the Soft actor–critics (SAC) algorithm and Monte Carlo Q-learning (MCQ) algorithm. Nevertheless, the algorithm should be tested on more complex scenarios and interactions with the environment [212].

Zhang et al. proposed the Lyapunov-based Soft actor–critics with Collision Probability Prediction (LSAC-CPP) algorithm to find safe and stable paths for the ADS. They integrated two concepts into the well-known SAC algorithm; the first is the Collision Probability Prediction to identify the risk of collisions, and the second is the Lyapunov control theory to generate stable paths. The results indicated that the LSAC-CPP algorithm achieved a higher success rate than the SAC algorithm. The algorithm's limitations are as follows: no real-world test is performed, and dynamic environments are not considered [213].

Xu et al. developed a novel Actor-Critic Reinforcement Learning (ACRL) algorithm for multi-vehicle complex traffic scenarios. The policy network in the ACRL algorithm is divided into two sub-networks: the lane-keeping and the lane-changing networks. The ACRL updates its policy based on the gradient of the current advantage of the policy. The reward function is designed for efficiency, safety, and driving comfort. The real-time results showed that the ACRL had smoother paths than the rule-based algorithm. However, the algorithm is based on the pre-trained network, and it is not able to deal with new situations [214]

Choi et al. proposed the mobile robot collision avoidance learning with path (MCAL-P) algorithm based on the soft Actor-Critic (SAC) as the reinforcement learning algorithm. The virtual and real results showed that the (MCAL-P) algorithm outperformed the Timed Elastic Band (TEB) algorithm and the Dynamic Window Approach (DWA). The disadvantage of the algorithm is that it cannot drive in a straight line [215].

Tang et al. deployed a Soft Actor-Critic (SAC) scheme as a deep reinforcement learning technique to address highway driving scenarios with continuous action space. The simulation results showed that the SAC algorithm can balance safety, efficiency, and comfort more than DQN and DDPG algorithms. However, no real-world experiments are executed, and the kinematic model of the vehicle is not feasible [216]. Machine learning algorithms are summarized in Table 9.

Table 9
Literature Review Summary for deep supervised learning Path Planning techniques.

Deep Supervised Learning algorithms			
Algorithm	Idea	Pros	Cons
IVGG-LSTM [174]	combines CNN (IVGG) for spatial data with LSTM for temporary data, then End-to-End planning.	outperformed VGG-16 , and NN in [173].	no real application, speed needs improvement.
NN [175]	novel NN trained by gradient-based self-supervised Algorithm.	faster than RRT* and SL algorithm	Only 74% accuracy, need more statistical analysis.
DeepPath [176]	CNN End-to-end scheme, CNN receives input image and car pose, then output the path.	can generate acceptable paths for IARA car.	No comparison was made with any other algorithm.
NN-NOC [177]	solve nonlinear optimal control problems to generate training data for NN.	good performance.	high computational time, not suitable for real-time, no comparison.
LSTM-based [178]	Fuzzy control generates data to train LSTM, LSTM generates the path.	outperformed BP neural network and fuzzy control algorithm.	no dynamic environment, needs further improvements to be adaptive.
SNN-based [179]	two agents; one to reach goal, the other to avoid obstacles. Input as a dynamic graph.	outperformed greedy and the threshold-based agent algorithms.	requires more obstacles, complex scenarios validation. and validation.
CNN-Raw-RNN [180]	Pilot: CNN-RNN receives input image and generates path, Copilot: verifies path safety.	outperformed the CNN , CNN-LSTM architectures in real.	needs to be verified on more scenarios with more AI algorithms.
NN [181]	Just used a traditional NN	Good perception accuracy	NN only limited for perception, wrong claim about path planning
DSUNet-PP [182]	integrated UNet with path prediction algorithm based on quadratic polynomial.	DSUNet-PP outperformed UNet-PP.	DSUNet-PP takes 5 images to generate path, high computational time.
Reinforcement Learning algorithms			
Algorithm	Idea	Pros	Cons
RL-MOHH [186]	Hybrid; Heuristic algorithm generates paths, RL chooses the best path.	faster than PNSGAII and NSGA-II algorithm to find safe path	needs more online training. (only 80% accuracy).
OPABRL [187]	Hybrid; RL generates path, A* improves the path.	High performance for low complexity problems.	Not tested on high dimensions problems.
DQN-GS [188]	hybrid between Q-learning algorithm and greedy selection GS.	outperformed Dijkstra and K-shortest algorithm in fuel consumption	neglected unknown dynamic environment.
QLOR [189]	hybrid Q-learning with Linear Output Regulation to design tracking controller.	successful in two scenarios.	no dynamic obstacles, no comparison to other algorithms.
AAIRL [190]	improve AIRL by augmenting it with semantic rewards.	Good results only compared with RL variants	Not compared to the other state-of-the-art algorithms.
HARL [191]	Hybrid A* with RL; A* find the path, RL optimizes path based on prior knowledge.	outperformed ACO, PSO, GA, and ANN algorithms.	High number of turns, large storage space for RL.
HPHA [192]	Optimize parameters of RL using PSO algorithm.	reliable solutions	PSO parameters are static, only graphical statistical analysis.
RTQL [193]	Used real-time Q-learning (RTQL) algorithm.	Managed to find path in various scenarios.	No comparison or statistics are made.
IDWAQ [194]	Hybrid Q-Learning with DWA; QL to find a path, DWA for unknown environment.	better than original DWA.	dynamic and complex obstacles are not considered.
IQL [195]	improved QL by adding distance metric, virtual target, dead-ends-free Q-function.	outperformed QL, RRT and VG algorithms.	too slow in escape mode, no dynamic obstacles.
AHPF-IRL [196]	Hybrid Artificial Harmonic PF with Integrated RL.	outperformed outperformed	neglect 3D and Dynamic environment.
Deep Reinforcement Learning algorithms (Value/Policy based)			
Algorithm	Idea	Pros	Cons
EBHS [199]	Hybrid; DQN is used as a heuristic function in a heuristic search algorithm.	outperformed DQN and Hybrid A* algorithms.	no generalization, no dynamic environment.
MEP-DIRL [200]	based on stochastic MDP, NN reward function trained via Maximum Entropy Principle.	obtained desired driving behaviors.	no real application, no comparison with other algorithms.
DDQN [201]	applied Double DQN	Better than DQN algorithm	No online interaction, no comparison.
DDQN [202]	Just applied Double DQN	Simulate highway driving in SUMO.	No comparison made.
DQN-PF [203]	Hybrid DRL with Potential Field algorithm	Better than traditional RL	success rate only 88.62% for 10000 trials.

(*continued on next page*)

Table 9 (*continued*).

CDQN [204]	hybrid DQN with Fuzzy control; DQN finds the path, fuzzy improves motion commands.	outperformed DDQN and A3C algorithms.	No obstacle avoidance.
DDQN-FSLAM [205]	Hybrid DDQN with FSLAM; DDQN to find path, FSLAM to map environment.	succeeded in different environments.	Slow with high computational time, No comparison with. other algorithms.
ETE-DRL [206]	Applied Deep Q-Network (DQN) in End-to-End scheme	Consider the environment	not compared with the state-of-the-art algorithms.
DRL [207]	implemented DRL in End-to-End scheme	Managed to implement DRL in ADS	No statistical analysis
RHRL [208]	improve the Hierarchical DDQN by using LSTM layer and a hybrid reward mechanism.	outperformed VDQN and hDDQN algorithms.	more scenarios needs to be tested.
IDQNPER-ETE [209]	End-to-End; CNN-LSTM processes sensor data, DQN generates path using IDQNPER algorithm.	achieved acceptable performance in a static environment.	no comparison, no dynamic environment.
DRL-GAT-SA [210]	combined graph attention network and DRL based on the safety model.	outperformed DDQN and DGN algorithms.	no real tests, more complex scenarios are required.
DDPG [211]	Implemented DQN and Deep Deterministic Policy Gradient (DDPG)	Both reach the goal, human-like performance in CARLA simulator.	Lack of real maps.
Deep Reinforcement Learning algorithms (actor–critics)			
Algorithm	Idea	Pros	Cons
HRL-MPC [165]	DDPG algorithm generates the path, clothoid algorithm makes the path smoother.	outperformed human drivers.	no real tests, no comparison against other algorithms.
MCS/AC [212]	combined RL techniques; Monte Carlo Search (MCS), and Actor-Critic (AC).	faster than SAC and MCQ algorithms.	need more complex scenarios and interactions with the environment.
LSAC-CPP [213]	improved SAC by Collision Probability Prediction, and Lyapunov theory for stability.	more success rate than traditional SAC.	no real-world test, no dynamic environment.
ACRL [214]	Actor-Critic based with two sub-networks, gradient-based policy update.	smoother paths than Rule-based Algorithm.	based on pre-trained network, cannot deal with new situations.
MCAL-P [215]	based on Soft Actor-Critic (SAC) algorithm.	outperformed DWA and Timed Elastic Band (TEB) algorithm.	cannot drive in a straight line.
SAC [216]	implemented Soft Actor-Critic (SAC)	better than DQN and DDPG algorithms.	no real-world experiments, car kinematic model is not feasible.

The future of Deep Supervised Learning Techniques in autonomous driving systems should focus on enhancing real-time processing capabilities and reducing reliance on extensive training datasets. Integrating these techniques more effectively with other path-planning strategies, like optimization or interpolation methods, could yield more efficient and adaptive solutions. In addition, using pre-trained models can increase training speed. Additionally, improving the algorithms' ability to learn from limited data and incorporating self-improving mechanisms through continuous learning will be vital in advancing these techniques for practical autonomous driving applications.

The Traditional Reinforcement Learning (RL) techniques do not need a training data set to be trained; they can interact with the environment and update the model in real time. However, storing the Q table requires a massive memory for complex driving scenarios. The research direction solved this problem by using deep reinforcement learning (DRL), which incorporates advanced neural network architectures that can replace the need to store substantial state–action tables. However, these networks generate the need for parameter tuning for the weights of the neural networks, which makes the training process too time-consuming. The best way to tackle this problem is to use pre-trained deep Q networks to accelerate the learning process for practical and reliable autonomous vehicle navigation.

9. Layer 5 - part 3: Path planning (meta-heuristic optimization techniques)

Nowadays, meta-heuristic optimization algorithms have been intensely applied to a wide range of optimization problems [217]. As long as the path planning problem in the AD system can be modeled as an optimization problem, meta-heuristic optimization algorithms can be easily applied to resolve this problem [218]. The main advantage of meta-heuristic optimization algorithms is that they are problem-independent and are considered general problem-solvers. They can give an optimal solution for any complex search space. Therefore, it can be applied to any problem with few modifications [219]. Meta-heuristic optimization algorithms are less likely to get stuck into a local minimum, compared to the gradient-based algorithms, because of the exploration phase that can help to explore different search spaces.

Nevertheless, the main challenge of the meta-heuristic optimization algorithms is the parameter tuning to achieve the balance between exploitation and exploration. In exploitation, the algorithm seeks the best solution in the current search space region. While in exploration, the algorithm explores different search spaces to ensure diversity in the candidate solutions [220]. Moreover, they can provide different paths for the same problem if run multiple times. The reason behind this problem is the random behavior in specific stages of the algorithms.

The meta-heuristic optimization techniques can be classified into four categories: evolutionary algorithms, trajectory-based algorithms, swarm-based algorithms, and nature-based algorithms. Evolutionary algorithms imitate the concept of the survival of the fittest. Genetic Algorithm (GA), Differential Evolution (DE), Harmony Search (HS) algorithm, and Genetic Programming are the most common evolutionary algorithms. Trajectory-based algorithms rely on updating solutions by moving through neighboring candidates. The most used algorithms are Tabu Search (TS), Simulated Annealing (SA), hill climbing, and Iterated Local Search (ILS) algorithm.

Swarm-based algorithms mimic the nature of animals and humans, such as animal behavior in searching for food in groups. The most famous Swarm-based algorithms are the Grey Wolf Optimization (GWO) algorithm, Artificial Bee Colony (ABC) algorithm, Cat Swarm Optimization Algorithm (CSOA), Particle Swarm Optimization (PSO) algorithm, Bacterial Foraging Optimization (BFO), Ant Colony Optimization (ACO) algorithm, and Chicken Swarm Optimization (CSO) Algorithm. The

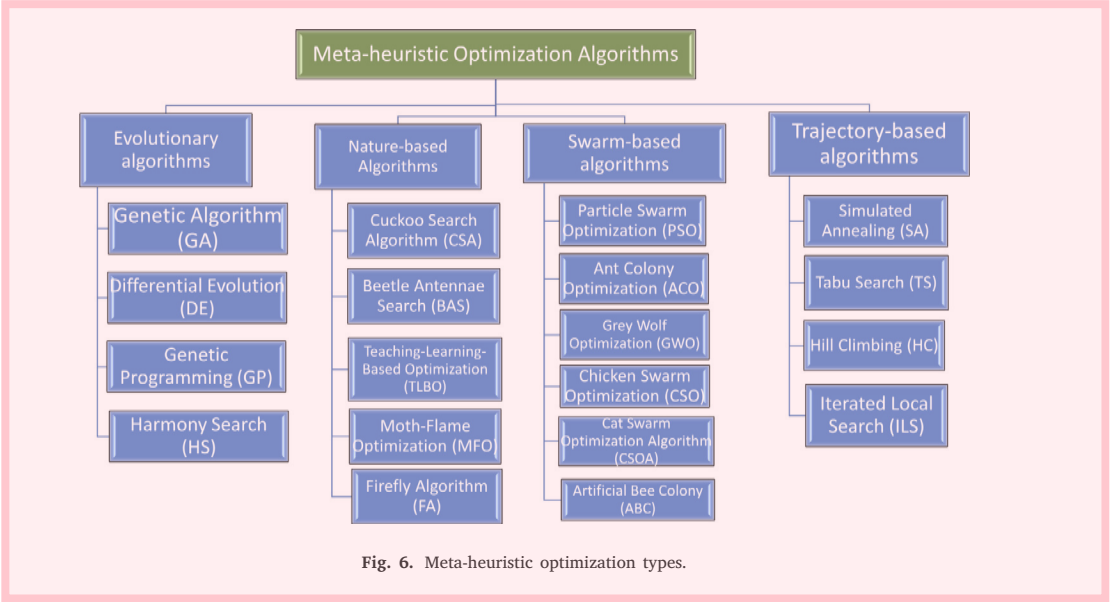


Fig. 6. Meta-heuristic optimization types.

last category is the nature-based algorithms, which are inspired by natural phenomena that exist in animals and humans, such as the Moth–Flame Optimization (MFO) Algorithm, Firefly Algorithm (FA), Teaching-Learning-Based Optimization (TLBO) Algorithm, Beetle Antennae Search (BAS) Algorithm, and Cuckoo Search Algorithm (CSA). The categories of the meta-heuristic algorithms are shown in Fig. 6.

9.1. Genetic Algorithm (GA)

The genetic Algorithm (GA) is the most popular evolutionary algorithm, which imitates the principle of survival of the fittest. This principle was first discovered in 1958 by Bremermann [221]. The GA in its recent form as an optimization algorithm was proposed by Holland in 1975 [222]. GA and its variants have recently been applied in various applications, including path planning.

GA begins with an initial randomly generated population of candidate solutions. Each candidate is called an individual and has a fitness value, which describes how good the solution is. Each solution is selected based on its fitness and passes its genetics to two newly generated solutions. The previous operation is called crossover. To ensure the diversity of the new generation, another operation, called a mutation, is applied by a low percentage. The previous steps are repeated until they are solved with the desirable accuracy.

Utami et al. proposed the Modified Crossover Genetic Algorithm (MCGA), in which the crossover operator was modified by comparing the newly generated solutions' fitness with the original solutions' fitness. The best fitness will survive, and the poor fitness will be ignored. Simulation results showed that the MCGA algorithm can generate optimal paths based on the desired conditions. Nevertheless, the algorithm is not compared to any other algorithm, and the dynamic obstacles are not considered [223].

Receveur et al. designed a hybrid algorithm called Genetic Algorithm-Potential Field (GA-PF), which combines the GA with the Potential Field Algorithm. The global path is generated and optimized by GA. Then, the PF algorithm generates the local path based on the global path obtained from the GA algorithm. This algorithm overcame the drawbacks of the PF algorithm, such as getting stuck in a local minimum. However, the final trajectory is not periodically recalculated. Moreover, the non-holonomic car model and the dynamic constraints are not considered. Therefore, the solutions are not feasible for real-life ADS [224].

9.2. Differential Evolution (DE)

Differential Evolution (DE) is another evolutionary meta-heuristic optimization algorithm, like the GA algorithm. It starts with a set of candidate solutions called agents. These agents explore the search space based on a simple mathematical rule to combine the position of agents in the population. If the new position is improved, the new position will be accepted to form part of the population. Otherwise, it will be neglected. This algorithm was first introduced by Storn and Price in 1997 [225]. It has recently been improved to solve the path planning problem.

Guo et al. applied the Differential Evolution (DE) algorithm to the Hongqi autonomous car HQ430. A model predictive control (MPC) scheme has been implemented to design the controller; then, the DE algorithm is deployed to obtain the optimal path. The experimental results showed that the implementation can achieve suitable control performance for path planning in autonomous cars. However, no theoretical statistical analysis or comparison has been made to any other algorithm [226].

9.3. Simulated Annealing (SA)

Simulated Annealing (SA) is a trajectory-based meta-heuristic algorithm recently used to solve optimization problems. The concept of the SA algorithm relies on the idea of annealing in metallurgy, which involves cooling and heating the materials to change their physical properties of the material. In the SA algorithm, the temperature smoothly decreases from a starting value until it reaches zero. The slow cooling process is interpreted in the SA algorithm as the slow decay in the probability of accepting poor solutions while exploring new solutions. During the cooling process, the thermodynamic characteristics will be changed, representing the improvement of solutions [227]. This algorithm has recently been adapted to enhance trajectory planning solutions.

Magdy et al. implemented the SA algorithm for motion planning in the Autonomous Transportation Operating Modules (ATOM) system. The SA algorithm is compared to ACO and PSO algorithms. Concerning the computational time, the results of the SA outperformed the PSO algorithm and ACO Algorithm. However, PSO gave solutions with better cost than ACO and SA. The algorithms are not tested in a real-life experiment. Moreover, it is compared only to ACO and PSO algorithms, neglecting the other promising meta-heuristic optimization algorithms [228].

Yin et al. designed a hybrid algorithm between the SA algorithm and the PSO algorithm called the SAPSO algorithm, in which the

PSO algorithm was kept as simple as possible. The hybrid algorithm showed better results than those obtained from the Simulated Annealing Teaching Learning Based Optimization (SA-TLBO) and the original PSO algorithm. However, the dynamic environment is not considered, and no numerical statistical analysis has been made [229].

Shang et al. developed an Improved Simulated Annealing (ISA) algorithm, where the original SA algorithm is improved by designing neighborhood transformation techniques and better solution acceptance rules to improve the global search. The results proved the robustness of the ISA. However, the execution time of the algorithm is relatively high, and the algorithm was not tested on high dimensions problems, more than 200 nodes [230].

9.4. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a widely used swarm-based meta-heuristic optimization algorithm. It imitates the organized movements of bird flocks to find food without a leader; the birds go with the one nearest to the food source. It was first proposed by Kennedy and Eberhart in 1995 [231]. The PSO algorithm represents the candidate solutions as a group of particles. Recently, the PSO algorithm has been modified and customized to the path planning.

Li et al. [232] introduced a hybrid algorithm called SLPSO, which combined Particle Swarm Optimization (PSO) and a Self-adaptive Learning algorithm. The PSO algorithm finds the optimal path under multiple constraints, such as collision risk, path length, and smoothness. Then, the Self-adaptive Learning algorithm is deployed to enhance the capability of the PSO algorithm. The results proved that the SLPSO is better and more effective than PSO and GA. However, SLPSO is not used for the complex environment for path planning of the ADS. Furthermore, it is not applied in the 3D environment.

Zhang et al. innovated a hybrid algorithm called the Evolutionary Scatter Search Particle Swarm Optimization (ESS-PSO) algorithm. This algorithm uses the ESS algorithm to generate paths in a scatter search template. The PSO algorithm improves the paths generated from the ESS algorithm, where the velocity and the position of the car are implemented using the "ruin and recreate" concept. The algorithm gave excellent results concerning accuracy and speed. However, the algorithm has yet to be used for local planning [233].

Zhang et al. developed an Improved Localized Particle Swarm Optimization (ILPSO) algorithm for trajectory planning. Improvements in PSO parameters are made to increase convergence speed and prevent the ILPSO algorithm from falling into a local minimum. Furthermore, extended Gaussian distribution has been used to increase the diversity of particles. Compared to the basic PSO algorithm and A* algorithm, the ILPSO algorithm outperformed them. However, the ILPSO is not applied in complex,3D, and large-scale environments [234].

Wahab et al. presented a hybrid algorithm between the PSO algorithm and the Fringe Search Algorithm, named PSOFS, for the indoor environment. Generally, the PSO algorithm generates the path as a sequence of points. The role of the FS algorithm arises by generating a safe point if the generated point by the PSO algorithm is less than a one-meter distance from the obstacles. The PSOFS algorithm generated safer, shorter, and smoother paths than the ones obtained from PSOA and PSOD algorithms. Nevertheless, the algorithm needs more investigations against more state-of-the-art algorithms [235].

Qiuyun et al. introduced an Improved Particle Swarm Optimization (IPSO) algorithm for the material transportation problem in the indoor static environment. Two main modifications had been made to the original PSO; a crossover operator had been added to the particle position, and a mutation operator had been added to prevent falling into a local minimum. The IPSO algorithm outperformed the original PSO, GA, and ACO algorithms in material transportation. Nevertheless, dynamic obstacles are not considered, and the algorithm is not applied in the outdoor environment; it is problem-specific with high constraints [236].

Fusic et al. [237] implemented Satellite PSO (SPSO) and five other PSO variants to address the path optimization problem based on images from the satellite. Satellite map data are used to map the unknown environment. SPSO achieves the best among PSO variants. However, this study focused only on the PSO algorithm and its variants, neglecting other prominent meta-heuristic optimization algorithms.

Chai et al. proposed Enhanced Multi-Objective PSO (EMOPSO), where the original path planning problem is reformulated to a multi-criterion unconstrained problem by adding a normalized objective function that reflects the total amount of constraint violation. Moreover, two local exploration operations evolution RS and ϵ -bias selection method were added to improve the progress. The algorithm outperformed the Potential Field PSO (PFPSO) and Potential Field ABC (PFABC) algorithms. However, dynamic obstacles are not considered, and the algorithm needs to be tested on a more complex environment [238].

Zhang et al. designed a hybrid algorithm between Improved PSO, Improved Artificial Potential Field (IAPF), and Simulated Annealing (SA) algorithm, called the IPSO-APF/SA algorithm. The improved PSO algorithm, with adaptive weights, is used for global path planning. The improved APF algorithm has been deployed for local path planning and avoiding collisions. The parameters of the APF algorithm are optimized using the SA algorithm. The practical and simulation results showed that the IPSO outperformed the original and Mutation PSO (MPSO) algorithms. However, the dynamic obstacles are not considered, and more statistical analysis is required [239]

9.5. Ant Colony Optimization(ACO)

The ACO algorithm is a swarm-based meta-heuristic optimization algorithm introduced by Dorigo in 1997 [240]. ACO algorithm mimics the ants' behavior in organized groups while searching for the food source. It is required that the ants find the shortest path to the source of food. Recent research adopted the ACO algorithm to the path-planning problem.

Chen et al. improved the ACO algorithm by combining the A* and ACO algorithms. The results proved that the new hybrid algorithm gave more efficient and accurate solutions than the ones from the ACO algorithm. However, the algorithm still has high computational time [241].

Ali et al. developed a hybrid algorithm among the A* Multi-directional algorithm, the ACO algorithm, and the Markov Decision Process (MDP) model. The A* Multi-directional algorithm searches and stores the best nodes between the start and the endpoints. Next, the ACO algorithm optimizes the generated path to avoid blind searches. Then, the MDP scheme is implemented to reduce the sharpness of the global path. The algorithm gave better results than the other algorithms. However, the algorithm did not take into consideration the moving obstacles. Moreover, it has high computational time [242].

Wu et al. presented an Improved ACO algorithm optimized by the Particle Swarm Algorithm (IACO-PSO). Not only is the path length taken into consideration, but also the flow of the incoming traffic and the number of lanes are considered. The ACO algorithm is deployed for dynamic path planning, whereas the PSO algorithm is used to optimize the parameters of the ACO algorithm. The practical results showed that the IACO algorithm reduced the average rate of congestion from 13.63% to 9.73% compared to the ACO algorithm. However, no comparison had been held with other algorithms [243].

Pohan et al. designed a hybrid algorithm between the RRT algorithm and the Ant Colony System (ACS) algorithm, named the RRT-ACS algorithm. The RRT algorithm generates a quick sub-optimal path. After that, the optimal path is generated from the sub-optimal paths using the ACS algorithm. The RRT-ACS algorithm outperformed the original RRT, RRT*, and the other RRT variants. However, the algorithm became too slow to find the optimal path, especially when the effect of the ACS

algorithm was minimal. Furthermore, the dynamic environment is not considered [244].

Zhang et al. introduced the Adaptive Improved ACS algorithm using the population information Entropy (AIACSE) algorithm. The information entropy concept is added to the ACO algorithm to increase the diversity of the population. The population is initialized based on non-uniform distribution to avoid blind searches at the beginning. Moreover, the adaptive parameters and pheromone diffusion model have been applied to ensure an acceptable balance between exploitation and exploration. The AIACSE algorithm outperformed the original ACS, the Rank-based Ant System (RAS), and PS-ACO algorithms. Nevertheless, the dynamic environment is not considered, and the algorithm needs parameter improvements [245].

Jiang et al. implemented a hybrid algorithm between ACO and PSO based on feedback from the interaction with the environment to make the algorithm adaptive. The algorithm gave better and faster solutions than the traditional ACO algorithm. However, this algorithm is compared to the original ACO for low dimensions problem [246].

Zhou et al. developed a hybrid algorithm between the Dijkstra Algorithm and the ACO algorithm, named ACO-DA, for the indoor airport environment. The ACO algorithm can find a global path using the feedback information. Dijkstra's algorithm can avoid obstacles while selecting the best path. The ACO-DA algorithm outperformed the A*, Dijkstra, and ACO-A* algorithms. Nevertheless, the algorithm is not used in the outdoor environment, and it needs to be tested on more scenarios and benchmark functions with deep statistical analysis [247].

9.6. Artificial Bee Colony (ABC) algorithm

The ABC algorithm is a popular swarm-based meta-heuristic optimization algorithm based on honey bees' natural activity while searching for a food source. Three types of bees play an essential role in the ABC algorithm. The scout bees search for food sources. Then, the employed bees test and evaluate the quality of nectar food sources obtained from the scout bees. The onlooker bees obtain the data from the employed bees and choose the best food source. This algorithm was first proposed by Kharaboga in 2005 [248].

Nayyar et al. designed the Arrhenius ABC (aABC) algorithm, improving the balance between exploration and exploitation using the Arrhenius equation. The results showed that the aABC algorithm outperformed the traditional ABC, DE, and PSO algorithms. However, the algorithm needs more driving scenarios to be tested in the ADS [249].

Kamil et al. introduced the Adaptive Dimension Limit-ABC (ADL-ABC) Algorithm, in which an adaptive limit parameter has been used instead of a fixed, stable limit. The cubic polynomial interpolation technique is used to make the path smoother. The ADL-ABC algorithm outperforms the traditional ABC algorithm. However, more complex scenarios should be tested to judge the level of significance [161].

Xu et al. developed the Coevolution framework with the Global best Leading ABC (Co-GLABC) algorithm. In this algorithm, the Differential Evolution (DE) search equation is introduced to the ABC algorithm to speed up the convergence speed. Moreover, a crossover technique is integrated with the global best position to handle the dimension-dependent part. Furthermore, an innovative setting for the food source has been applied to boost performance. The Co-GLABC outperformed the state-of-the-art algorithms. However, dynamic obstacles are not considered, and more complex scenarios need to be tested [250].

Kumar and Sikande proposed a hybrid improved Artificial Bee Colony with Evolutionary Programming (EP), named iABC–EP algorithm. The improved ABC algorithm generates the path. Then, evolutionary programming (EP) is applied to improve the path achieved by the iABC algorithm. The simulation results proved that the iABC–EP algorithm can give shorter paths than the ABC–EP algorithm by 5.75%. However, more complex scenarios should be tested [251].

9.7. Grey Wolf Optimization (GWO) algorithm

GWO algorithm is a recent swarm-based meta-heuristic optimization algorithm that was first introduced by Mirjalili et al. in 2014. The GWO algorithm is inspired by the hierarchy of the wolves and the hunting group's behavior. The GWO algorithm is widely used because of its simplicity, but it has a slow convergence rate and can be stuck into a local minimum for some problems [252].

Liu et al. proposed the Improved Grey Wolf Optimization (IGWO) algorithm, where further improvements had been made to the original GWO algorithm; the lion optimizer optimizes the parameters to improve search capability, and the weights became dynamic to add more diversities to the wolves. The IGWO algorithm outperformed the Chicken Swarm Optimizer (CSO) algorithm, the original GWO algorithm, the Butterfly Optimization Algorithm (BOA), and the Whale Optimizer Algorithm (WOA) on a maximum of 30 dimensions problems. However, dynamic obstacles are not considered, and a more complex environment needs to be validated [253].

Kiani et al. proposed three hybrid algorithms between the Grey Wolf Optimization (GWO) and the Adaptive Rapidly-exploring Random Tree (ARRT) algorithm. The ARRT algorithm can be hybridized with one of the following GWO variants: GWO (ARRT-GWO), Incremental GWO (ARRT-IGWO), and the Expanded GWO (ARRT-ExGWO) algorithms. The GWO algorithm constructs the path, while the RRT algorithm avoids possible obstacles. The ARRT-ExGWO algorithm outperformed the other GWO variants, PSO, Whale Optimization Algorithm (WOA), and the improved BA algorithm under four scenarios. However, these algorithms are not tested on complex benchmark functions. Moreover, dynamic and 3D environments are not considered [254].

Zafar et al. implemented a hybrid algorithm between the GWO and APF algorithms, namely the GWO-APF algorithm. This algorithm has been applied to on-time path planning of ADS in a static environment. The APF algorithm finds obstacle-free locations. Then, the GWO algorithm minimizes the path and locations generated by the APF algorithm. The results revealed that the GWO-APF algorithm outperformed the Surrounding Point Set (SPS) [255]. However, the algorithm was not applied in a dynamic environment and complex benchmark functions [256].

9.8. Chicken Swarm Optimization (CSO) algorithm

The CSO algorithm is a recent and effective swarm-based meta-heuristic optimization algorithm introduced by Meng et al. in 2014. This algorithm mimics the natural behavior of chicken groups. However, the CSO algorithm can fall in a local minimum, especially in complex optimization problems [257].

Liang et al. designed the Improved CSO (ICSO) algorithm as a meta-heuristic optimizer to get the shortest path for the autonomous car. They improved the original CSO algorithm by adding Levy flight to the update rule of the hen's location to increase the population's diversity and exploration. Moreover, a nonlinear weight reduction strategy is added to the update rule of the chicken's position to improve the ability of self-learning. The results showed that the ICSO is faster and more accurate than the PSO and original CSO algorithms. However, the algorithms are only tested on a few benchmark functions and only one path planning scenario [258].

9.9. Cat Swarm Optimization Algorithm (CSOA)

The classical Cat Swarm Optimization (CSO) algorithm is a recent swarm-based meta-heuristic optimization algorithm that mimics the cats' behavior. However, the algorithm does not fully consider the other optimal cat position in the update process. Therefore, the classical CSO lacks population diversity, which may lead to falling into local minimum [259].

Zhao et al. introduced the MOCMCSO-APFM, a hybrid algorithm between the APF Method (APFM) and the Multi-Objective Cauchy Mutation Cat Swarm (MOCMCSO) algorithm. In the MOCMCSO algorithm, they introduced the Cauchy mutation operators to improve the search patterns to obtain the shortest path. The APFM algorithm is used to avoid collisions in the indoor environment. The results proved that the MOCMCSO algorithm outperformed the Multi-Objective Particle, Swarm Optimization (MOPSO) and the original Multi-Objective Cat Swarm Optimization (MOCSO) algorithms. However, the path points are repeated in the inspection case, and anti-collision efficiency needs to be increased [260].

9.10. Firefly Algorithm (FA)

FA algorithm is a nature-based meta-heuristic optimization algorithm used to solve optimization problems. The FA algorithm mimics the fireflies' natural flashing behavior. Fireflies belong to the winged beetle family, which is called the lightning bug, because of their ability to generate light. Fireflies use that light to select their mate without wasting heat energy. Sometimes, they use the light to make their enemies scary to avoid them. The goal is to find the proper mate and avoid the enemy. Yang first proposed the FA algorithm in 2008 [261]. It has been used recently as a meta-heuristic approach in trajectory planning.

Zhou et al. introduced the Modified Firefly Algorithm (MFA), in which the step factor α is adaptive, and the convergence speed is controlled by the parameter β_0. The results indicated that the MFA algorithm outperformed the Genetic Algorithm (GA) and the traditional Firefly Algorithm (FA). However, More scenarios and benchmark functions should be applied to test the algorithm's performance [262].

Bisen and Kaundal applied the original FA algorithm to the path planning for autonomous cars. They applied the FA algorithm to multiple scenarios. The algorithm managed to find an acceptable path. Nevertheless, the algorithm is not compared to any other algorithm. The performance of the traditional algorithm needs to be enhanced [263].

Li et al. developed the self-adaptive population size firefly algorithm (SPSFA), where the population size is adaptive to the collision degree using two nonlinear functions. An adaptively adjusted parameter is added to control the population size. The results proved that the SPSFA algorithm outperformed the traditional FA algorithm with a fixed population size. Nevertheless, the SPSFA algorithm is slower than the traditional FA algorithm [264].

Abbas proposed a hybrid algorithm between the D* algorithm and the FA algorithm, named (the FA-D*) algorithm. The D* algorithm has been deployed to find the shortest path. The FA algorithm is applied to generate a trajectory of intermediate points in the free space to achieve the path from the D* algorithm. Finally, a Quadratic parametric equation is used to make the path smooth. The algorithm managed to find collision-free and safe paths. However, the algorithm is not compared to any other algorithm in literature [265].

9.11. Cuckoo Search Algorithm (CSA)

The CSA is a recent nature-based meta-heuristic optimization algorithm introduced by Yang and Deb in 2009 [266]. The CSA algorithm is based on the cuckoo bird behavior, which lays eggs in another host bird. The main goal of the cuckoo bird is to mimic the host bird's eggs and avoid being discovered by the host bird. Hence, the optimization goal is reached if the host bird cannot discover the cuckoo egg. Researchers widely apply the CSA algorithm to solve various optimization problems, including path planning problems.

Alireza et al. designed the Enhanced Mutated Cuckoo Optimization Algorithm (EMCOA). The experimental results indicated that the EMCOA algorithm is better than the GA and A* algorithms. However, the statistical analysis needs to be more robust to judge the performance of the algorithms. Local planning is not considered as well [267].

Reda et al. proposed a new discrete CSA variant named the Discrete Damped Cuckoo Search (DDCS) algorithm to solve the order-picking routing problem inside warehouses. The proposed algorithm improved the traditional CSA algorithm by combining 2-opt move, crossover operators, and random key encoding. The DDCS algorithm was customized to address the OPR problem inside a warehouse environment. Results indicated the significant performance of the DDCS algorithm over GA, PSO, and ACO algorithms, especially in complex problems. However, this algorithm is not tested for local or global path planning [80].

9.12. Beetle Antennae Search (BAS) algorithm

BAS Algorithm is a new nature-based meta-heuristic optimization algorithm proposed in 2017 by Jiang and Li [268]. The navigation of longhorn beetles inspires the algorithm and mimics the random walking mechanism of beetles and the function of antennae in nature. However, the convergent BAS algorithm is based heavily on the direction of a random beetle in every iteration. Therefore, this algorithm was improved by Wang and Chen in 2018, who proposed the Beetle swarm antennae search (BSAS) algorithm. The BSAS algorithm overcomes the drawbacks of the BAS algorithm. It combines the feedback-based step-size update technique with the swarm intelligence algorithm [269].

Towards the improvement of the BAS algorithm, the Beetle Swarm Optimization (BSO) algorithm was introduced by Wang and Yang in 2018 [270]. The BSO algorithm is a hybrid algorithm between Beetle Antennae Search (BAS) and the PSO algorithm (BAS-PSO). The BSO algorithm combines the PSO algorithm's update strategy and the BAS algorithm's search mechanism, thus increasing convergence speed and avoiding the local minimum trap.

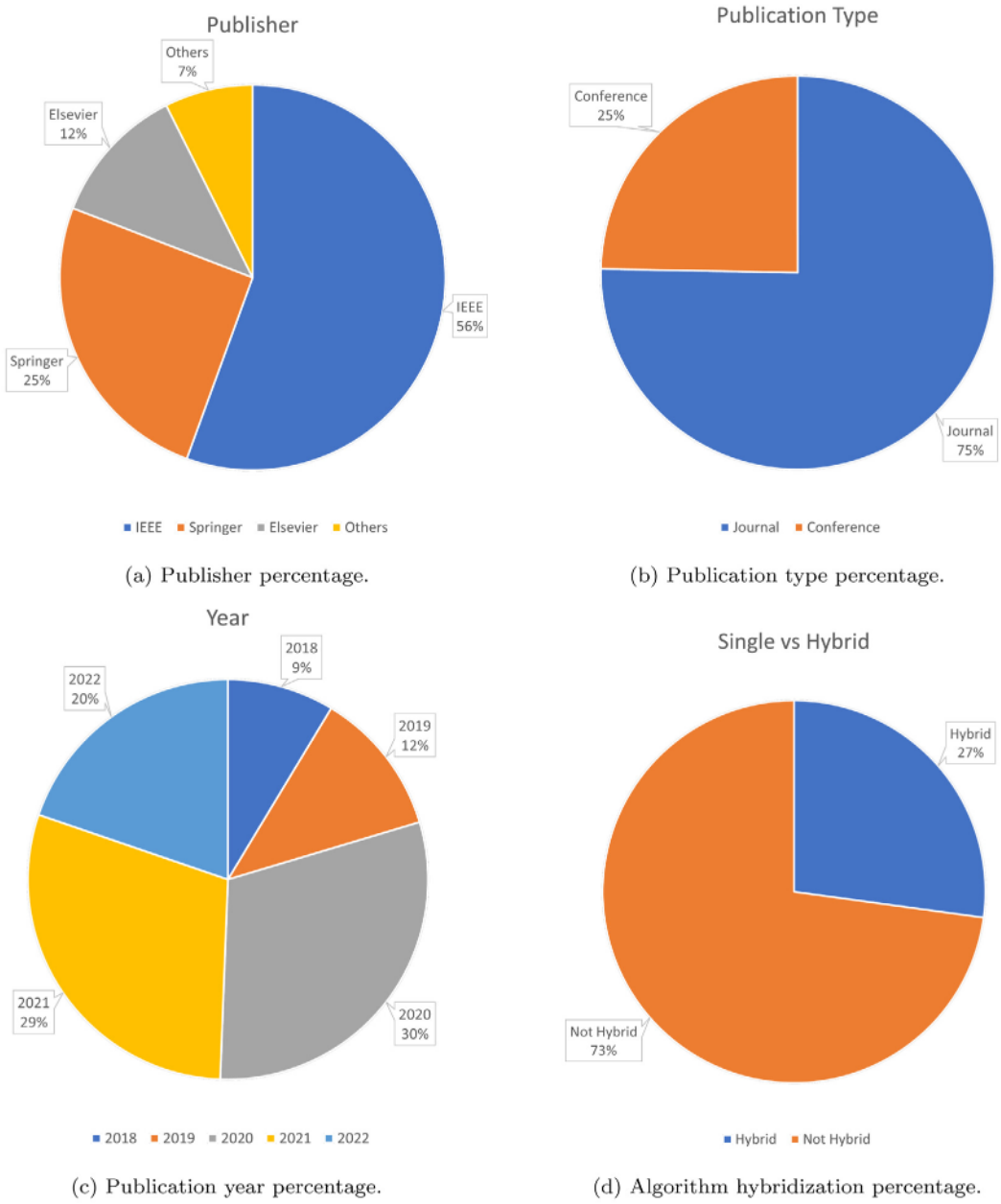
Mu et al. presented an improved version of the BSO (IBSO) algorithm to address the three-dimensional route planning problem, where a penalty term has been added to the fitness function. Next, the cubic spline interpolation technique has been applied to make the paths smoother. Results showed that the IBSO algorithm gave shorter paths by 90% of those obtained from the PSO algorithm. However, the statistical analysis needed to be more comprehensive to judge the performance in a complex environment. [160].

Jiang et al. developed three variants based on the BAS algorithm for addressing the 2D and 3D environments. The first one is called the Local Fast Search with BAS (LFS-BAS) algorithm, which improves the exploration and the convergence speed of path finding and can avoid missing early solutions. The second one is called ACO-BAS, in which ACO initializes the population to obtain a fast local optimal path in real-time applications. The third one is called Searching Information Orientation with BAS (SIO-BAS), which guarantees the stability between speed and accuracy of the path-finding. The results proved that the three variants are better than the original ACO and BAS algorithms. However, dynamic and complex obstacles are not considered. Moreover, the statistical analysis needs to be stronger to judge the performance and needs to be compared with more algorithms [271].

9.13. Teaching-Learning-Based Optimization (TLBO) algorithm

The TLBO Algorithm is a nature-based meta-heuristic optimization algorithm inspired by the idea of learning between teachers and learners. A population is a group of learners. The TLBO algorithm consists of two phases: the teacher phase, where the learners learn from the teacher, and the learner phase, where learners learn from each other. The TLBO algorithm was first proposed by Rao et al. in 2011 [272]. This algorithm is used in many optimization problems.

Sabiha et al. designed a TLBO algorithm to address the online path planning for the ADS. LiDAR and IMU sensors are used in the perception layer. The TLBO algorithm took the static obstacles and vehicle dynamics into consideration. The algorithm is compared with the PSO, GA, and hybrid GA–PSO algorithms. The robot operating system (ROS)



(a) Publisher percentage.

(b) Publication type percentage.

(c) Publication year percentage.

(d) Algorithm hybridization percentage.

Fig. 7. General Statistics about the state-of-the-art path planning algorithms.

results showed the effectiveness of the TLBO algorithm. Nevertheless, the algorithm did not consider dynamic obstacles. Moreover, it did not deal with obstacles with irregular shapes. The algorithm did not use cameras that reduce the system vision [273].

9.14. Moth–Flame Optimization (MFO) algorithm

The MFO algorithm is a new nature-based meta-heuristic optimization algorithm inspired by the natural movement mechanism of moths towards the moon at night. However, if there is any other light source, this disturbs this movement by attracting the moths to it. This concept is used as an optimization algorithm in various applications. The main disadvantage of the algorithm is the slow convergence speed in the late iterations, which leads to a fall in local minimum [275].

Dai and Wei proposed the Improved Moth–Flame Optimization (IMFO) algorithm to solve path planning problems in the ADS in a static environment. The first improvement to the MFO algorithm is the historical best flame concept to enhance the ability of the updated law and avoid local minimum. The second improvement is quasi-opposition-based learning (QOBL) to ensure the diversity of the population. The results proved that the IMFO algorithm is better than the original MFO, PSO, GWO, and DA algorithms. However, the algorithm is tested only on a maximum of 10 Dimensions benchmark functions. Moreover, the dynamic environment is not considered [274]. Meta-heuristic optimization algorithms are summarized in Table 10.

Meta-heuristic optimization Techniques have been widely used in the literature to address the path planning problems in the ADS system. These algorithms act as general problem solvers, whatever the complexity of the problem, making them a general template that can adapt to any driving scenario. However, the main challenge of these algorithms is to find the balance between exploring new solutions and exploiting known reasonable solutions, which is the key to the efficiency of these algorithms.

Table 10
Literature Summary for Meta-heuristic Optimization Path Planning techniques.

Genetic Algorithm (GA)			
Algorithm	Idea	Pros	Cons
MCGA [223]	modified Crossover by comparing children to parents based on fitness, the best will survive.	can generate optimal paths based on the desired conditions.	no comparison at all, no dynamic obstacles.
GA-PF [224]	Hybrid; GA for the global path, PF for the local path.	Avoid local minimum.	optimal path not recalculated, not admissible paths/
Differential Evolution (DE)			
Algorithm	Idea	Pros	Cons
DE-based [226]	DE solves the path planning in the MPC model.	achieved experimental suitable control performance.	no theoretical statistical analysis, no comparison to other algorithms.
Simulated Annealing (SA) algorithm			
Algorithm	Idea	Pros	Cons
SA [228]	traditional SA	faster solutions than PSO and ACO	Gives high-cost solutions compared to PSO.
SAPSO [229]	Hybrid SA with PSO for better convergence.	better than PSO and SA-TLBO algorithms.	no dynamic environment, no numerical statistical analysis.
ISA [230]	added neighborhood transformation techniques to SA	verified solutions in multi-constraints path.	High computational time, not tested on high dimensions problems.
Particle Swarm Optimization (PSO) algorithm			
Algorithm	Idea	Pros	Cons
SLPSO [232]	Hybrid; PSO generates path, Self Learning Alg. improves PSO.	Better than GA PSO.	Not applied to complex 3D environment
ESS-PSO [233]	Hybrid; ESS generate initial path, PSO improves the path.	managed to solve path planning with time window.	Not used for local path planning yet.
ILPSO [234]	improved PSO parameters, extended Gaussian distribution is used for particles' diversity.	outperformed PSO and A* algorithms.	not applied in complex,3D, and large-scale environment.
PSOFS [235]	hybrid PSO and FS; if PSO generates unsafe points, FS generates safer ones.	outperformed PSOA and PSOD algorithms.	need more investigations against more state-of-the-art algorithms.
IPSO [236]	added crossover and mutation operations in PSO	outperformed PSO, GA, ACO in material transportation indoors.	No dynamic obstacles, not generalized to outdoor path planning problems.
SPSO [237]	PSO with satellite images for mapping	unknown environment using satellite images, best variant (SPSO).	neglected other algorithms (PSO only).
EMOPSO [238]	normalized objective function, local exploration operations (ϵ-bias, RS).	outperformed PFPSO and PFABC algorithms.	no dynamic obstacles, need a more complex environment.
IPSO-APF/SA [239]	Hybrid; PSO for global planning, APF for local planning, SA to optimize APF.	outperformed PSO and MPSO algorithms.	no dynamic obstacles, need more statistical analysis.
Ant Colony Optimization (ACO) algorithm			
Algorithm	Idea	Pros	Cons
ACO-A* [241] ACO-A*-MDP [242] '	Hybrid ACO with A*. Hybrid; A* stores best nodes, ACO optimizes the path, MDP smooths the path.	improved shortcomings of ACO. More accurate paths	high computational time. No dynamic objects, high computational time.
IACO-PSO [243]	optimize ACO parameters using PSO algorithm.	IACO reduce congestion rate from 13.63%. to 9.73% compared to ACO.	No comparison to other state-of-the-art algorithms.
RRT-ACS [244]	Hybrid ACS, RRT ; RRT finds sub-optimal path, ACS optimizes sub-optimal path.	outperformed all RRT variants.	too slow when ACS contribution is minimal, no no dynamic environment.
AIACSE [245]	entropy concept for diversity, biased initial population, diffusion model for balance.	outperformed RAS, ACS, PS-ACO, MRCACO algorithms.	no dynamic environment , needs parameter improvements.
ACO-PSO [246]	Hybrid ACO with PSO, adaptive based on environment interaction.	fast convergence speed, better than ACO	only compared to ACO for low dimensions problem
ACO-DA [247]	hybrid ACO, Dijkstra; ACO finds global path, DA avoids obstacles in selecting the best path.	outperformed ACO-A* , ,and Dijkstra algorithms.	not used in outdoor, more scenarios and statistics are required.

(*continued on next page*)

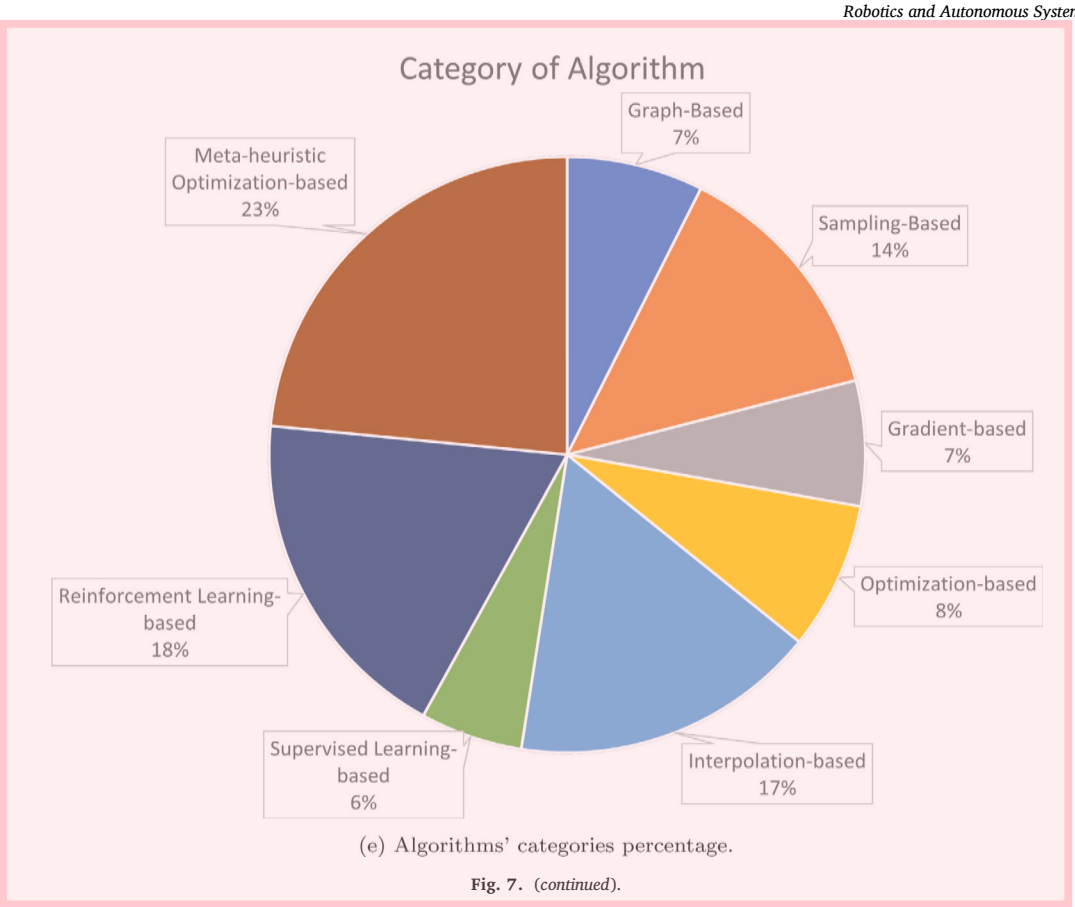
Table 10 (*continued*).

Artificial Bee Colony Optimization (ABC) algorithm			
aABC [249]	improve balance between exploration and exploitation in ABC using the Arrhenius equation.	outperformed PSO, DE and basic ABC algorithms.	more scenarios need to be tested in ADS.
ADL-ABC [161]	improve ABC, used adaptive limit parameter, Cubic polynomial smooths path.	outperform the traditional ABC algorithm.	more complex scenarios need to be tested.
Co-GLABC [250]	DE search eq. speeds up convergence, crossover handles, dimension-dependent, novel food source setting.	Co-GLABC outperformed the state-of-the-art algorithms.	no dynamic obstacles, needs more scenarios.
iABC–EP [251]	improved ABC finds path, Evolutionary Programming refines the path.	gave shorter path than ABC–EP algorithm by 5.75%.	more complex scenarios need to be tested.
Grey Wolf Optimization (WFO) algorithm			
IGWO [253]	optimized parameters by the lion optimizer, dynamic weights for diversity.	outperformed GWO, CSO, BOA, WOA algorithms.	no dynamic obstacles, need more complex environment (> 30 Dim.).
ARRT-ExGWO [254]	Hybrid RRT with ExGWO; RRT avoids obstacles, GWO finds the optimal path.	outperformed all other algorithms on 4 scenarios.	no benchmark test, no dynamic and 3D environment.
GWO-APF [256]	Hybrid GWO, APF; APF finds obstacle-free locations, GWO optimizes the path.	outperformed (SPS) algorithm in [255].	no dynamic environment, no complex benchmark functions.
Chicken Swarm Optimization (CSO) algorithm			
ICSO [258]	improved CSO by adding Levy flight and nonlinear weight reduction to update rules.	better than PSO and original CSO algorithms.	tested on few benchmark functions, and on only one scenario.
Cat Swarm Optimization algorithm (CSOA)			
MOCMCSO-APFM [260]	hybrid; MOCMCSO to find path with Cauchy Mutation, APFM to avoid obstacles.	better than MOCSO and MOPSO algorithms.	repeated points in the inspection case, low anti-collision.
Firefly Algorithm (FA)			
MFA [262]	step factor α becomes adaptive, convergence parameter is used β_0.	outperformed GA and FA algorithms.	More scenarios and benchmark is needed.
FA [263]	Applied traditional FA to multiple scenarios.	Managed to find acceptable paths.	No comparison, performance needs to be improved.
SPSFA [264]	population size is adaptive to collision degree using two nonlinear functions.	outperformed traditional FA algorithm.	slower than the traditional FA algorithm.
FA-D* [265]	D* finds path, FA finds intermediate path, Quadratic fn. smooths path.	managed to find collision-free path.	not compared to the state-of-the-art algorithms.
Cuckoo Search Algorithm (CSA)			
EMCOA [267]	added mutation operator for diversity	better than GA and A*	weak statistical analysis no local planning
DDCS [80]	added 2-opt move, crossover, random-key encoding to CSA	Solved TSP and path planning in indoor environment.	Not tested in path planning in outdoor environment.
Beetle Antennae Search (BAS) algorithm			
IBSO [160]	Improved BSO; added penalty term, to fitness fun., cubic spline interpolation for a smooth path.	better than PSO algorithm by 90%	poor statistical analysis.
LFS/ACO/SIO [271]	hybrid BAS; LFS to find early path, ACO for a fast path, SIO for a stable path.	better than BAS and ACO algorithms.	no dynamic, complex obstacles, poor statistics.
Teaching Learning Based Optimization (TLBO) algorithm			
TLBO [273]	applied traditional TLBO	outperformed GA, PSO in static environment	no dynamic irregular obstacles
Moth Flaming Optimization (MFO) algorithm			
IMFO [274]	Improved MFO; historical best flame, quasi-opposition based learning concepts are added.	better than MFO, PSO, GWO, and DA algorithms.	only max. of 10 Dim. functions, no dynamic environment.

Most research direction tackles this balance problem via adaptive rather than static parameters. This adaptive method will eliminate the need to tune the parameters before applying the algorithm to new driving scenarios. Moreover, the adaptive method will add the proper balance to the algorithm, leading to an optimal path. Furthermore, hybridization with some local search techniques can improve the convergence speed, making them suitable choices for real-life driving scenarios.

10. Statistics of the review paper

This section explores the numerical statistics about the state-of-the-art publications of the path planning algorithms in the ADS. The number of references used in the path planning problem is 162 reference. There are five metrics used to categorize these references: publisher, publication type, year, hybridization, and category of the algorithm.



(e) Algorithms' categories percentage.

Fig. 7. (*continued*).

10.1. General statistics

The general statistics show the percentage of each metric. Fig. 7 presents the general statistics about the state-of-the-art path planning algorithms in the ADS discussed in this paper. Concerning the publishers, 56% of the references were published in IEEE, 25% were published in Springer, 12% were published in Elsevier, and 7% were published via other publishers such as arXiv, AAAI, Hindawi, and SAGE UK, as shown in Fig. 7a. Fig. 7b shows that 75% of the publications are journal papers, while 25% are conference papers. Concerning the publication year, Fig. 7c represents that 9% of the publications were published in 2018, 12% in 2019, 30% in 2020, 29% in 2021, and 20% in 2022.

Concerning algorithm hybridization, Fig. 7d shows that 27% of the algorithms are hybrid with another algorithm of an elementary category. On the other hand, 73% of the algorithms are not hybrid with another primary algorithm, but they may include some concepts from another algorithm embedded in the main steps. Fig. 7e demonstrates the percentages of algorithm categories as follows: 7% of the algorithms are graph-based, 14% are sampling-based, 7% gradient-based, 8% optimization-based, 16% interpolation-based, 6% are supervised learning-based, 19% are reinforcement learning-based, and 23% are meta-heuristic optimization-based.

10.2. Chronicle order

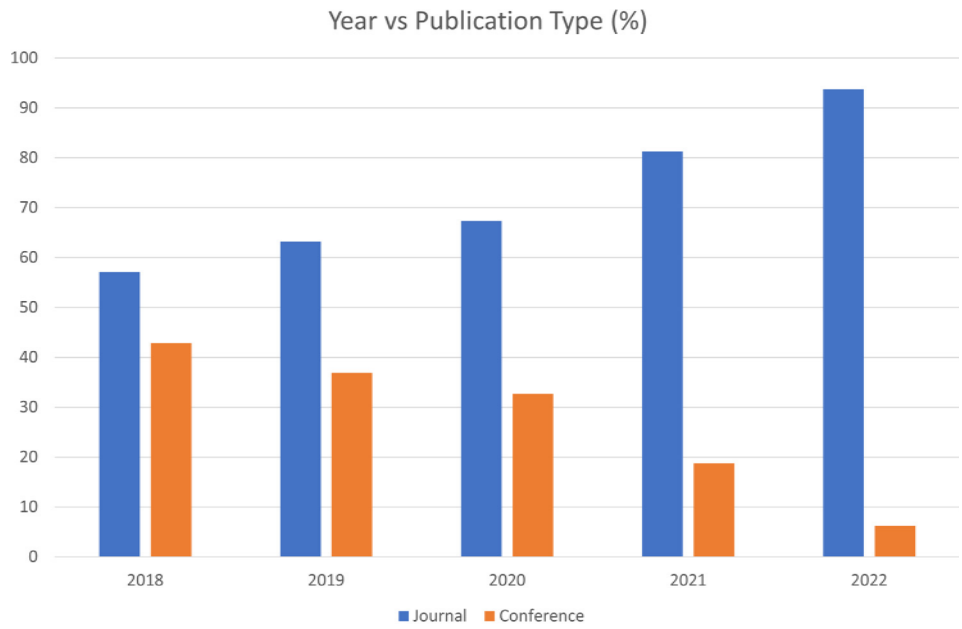
This section explores the metrics from a chronicle point-of-view as shown in Fig. 8. Fig. 8a shows the publication type percentages with time. In 2018, 57.14% of the publications were journal papers, while 42.86% were conference papers. In 2019, 63.16% of the publications were journal papers, while 36.84% were conference papers. In 2020, 67.45% of the publications were journal papers, while 32.65% were conference papers. In 2021, 81.25% of the publications were journal papers, while 18.75% were conference papers. In 2022, 93.75% of the publications were journal papers, while 6.25% were conference papers.

Fig. 8b demonstrates the algorithm hybridization percentages with time. In 2018, 14.29% of the algorithms were hybrid, while 85.71% were not hybrid. In 2019, 21.05% of the algorithms were hybrid, while 78.95% were not hybrid. In 2020, 32.65% of the algorithms are hybrid, while 67.35% are not hybrid. In 2021, 25% of the algorithms are hybrid, while 75% are not hybrid. In 2022, 31.25% of the algorithms are hybrid, while 68.75% are not hybrid.

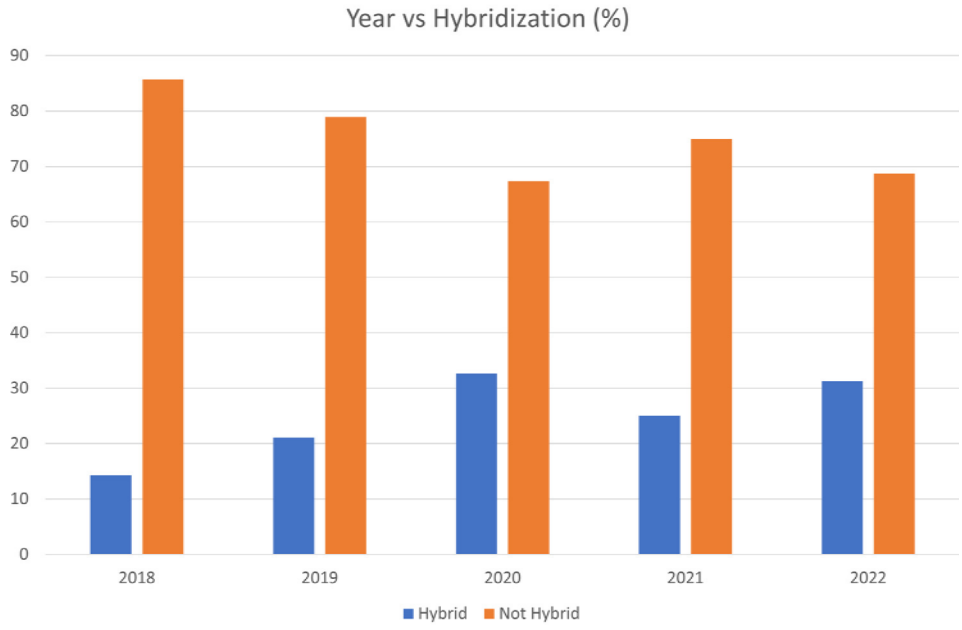
Fig. 8c shows the publisher percentage with time. In 2018, 78.57% of the publications are in IEEE, 14.29% are in Springer, and 7.14% are in Elsevier. The percentage of publishers in 2019 is 73.86%, 5.26%, 15.79%, and 5.09% in IEEE, Springer, Elsevier, and others, respectively. The publisher percentages in 2020 are 59.18%, 16.32%, 10.20%, and 14.30% in IEEE, Springer, Elsevier, and others, respectively. In 2021, the publishers' distribution was 54.17%, 29.17%, 10.42%, and 6.24% in IEEE, Springer, Elsevier, and others, respectively. In 2022, the publisher distribution was 31.25%, 50%, 15.63%, and 3.12% in IEEE, Springer, Elsevier, and others, respectively.

Fig. 8d presents the algorithm types percentages with time. In 2018, 7.14% of the algorithms are graph-based, 21.43% are sampling-based, 7.14% gradient-based, 14.29% optimization-based, 7.14% interpolation-based, 7.14% are supervised learning-based, 14.29% are reinforcement learning-based, and 21.43% are meta-heuristic optimization-based. In 2019, 10.53% of the algorithms are graph-based, 5.26% are sampling-based, 15.79% optimization-based, 36.84% interpolation-based, 10.53% are reinforcement learning-based, and 21.05% are meta-heuristic optimization-based.

In 2020, 6.12% of the algorithms are graph-based, 12.25% are sampling-based, 4.08% gradient-based, 8.16% optimization-based, 16.33% interpolation-based, 6.12% are supervised learning-based, 18.37% are reinforcement learning-based, and 28.57% are meta-heuristic optimization-based. In 2021, 6.25% of the algorithms are graph-based, 10.42% are sampling-based, 10.42% gradient-based, 4.17% optimization-based, 12.50% interpolation-based, 8.33% are supervised learning-based, 22.91% are reinforcement learning-based, and



(a) Publication type percentage with time.



(b) Hybridization percentage with time.

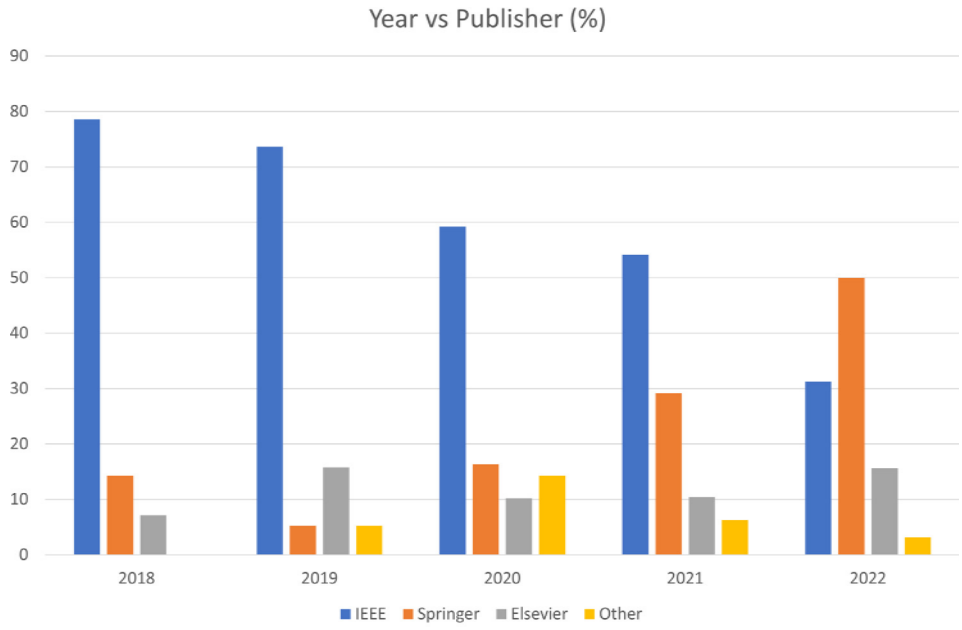
Fig. 8. Chronicle statistics of the state-of-the-art path planning algorithms.

25% are meta-heuristic optimization-based. In 2022, 9.38% of the algorithms are graph-based, 21.88% are sampling-based, 9.38% gradient-based, 6.25% optimization-based, 15.63% interpolation-based, 3.13% are supervised learning-based, 18.75% are reinforcement learning-based, and 15.60% are meta-heuristic optimization-based.

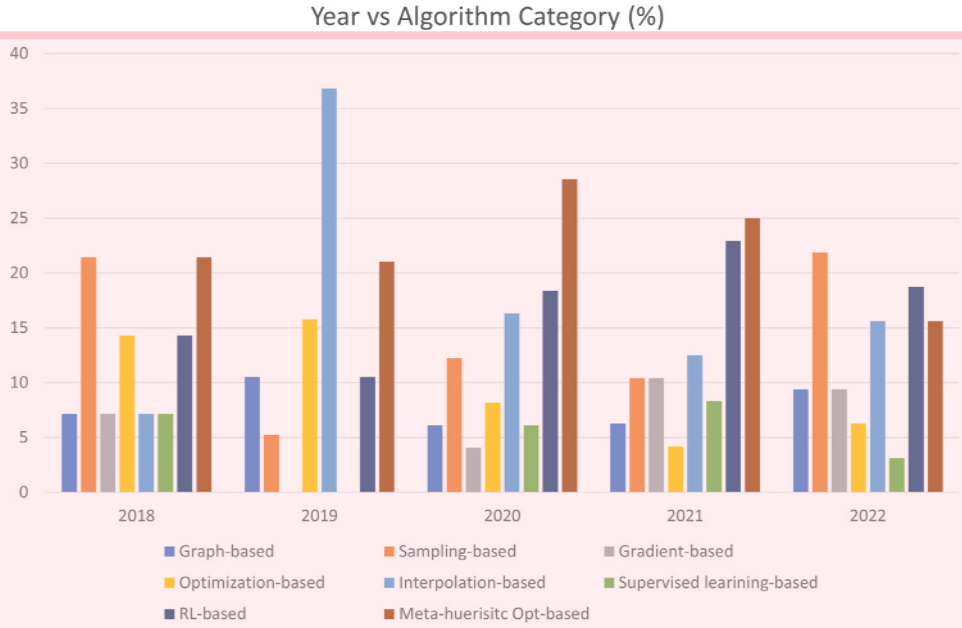
10.3. Algorithm category point of view

This section discusses the metrics from the algorithm category point-of-view as shown in Fig. 9. Fig. 9a demonstrates the publication year percentage for each algorithm type. For graph-based algorithms, 8.33% of the graph-based algorithms were published in 2018, 16.67% were published in 2019, 25% were published in 2020, 25% were published in 2021, and 25% were published in 2022. For sampling-based algorithms, 13.64% were published in 2018, 4.55% were published in 2019, 27.27% were published in 2020, 22.73% were published in 2021, and 31.81% were published in 2022.

For gradient-based algorithms, 9.09% were published in 2018, 18.18% in 2020, 45.45% in 2021, and 27.28% in 2022. For optimization-based algorithms, 15.38% were published in 2018, 23.08% in 2019, 30.77% in 2020, 15.38% in 2021, and 15.39% in 2022. For interpolation-based algorithms, 3.70% of them were published in 2018, 25.93% in 2019, 29.63% in 2020, 22.22% in 2021, and 18.52% in 2022. For supervised learning-based algorithms, 11.11%



(c) Publisher percentage with time.



(d) Algorithm categories percentage with time.

Fig. 8. (*continued*).

were published in 2018, 33.33% in 2020, 44.44% in 2021, and 11.12% in 2022. For reinforcement learning-based algorithms, 6.67% of them were published in 2018, 6.67% in 2019, 30% in 2020, 36.67% in 2021, and 19.99% in 2022. For meta-heuristic optimization algorithms, 7.89% of them were published in 2018, 10.53% in 2019, 36.84% in 2020, 31.58% in 2021, and 13.15% in 2022.

Fig. 9b demonstrates the publisher percentage for each algorithm type. For graph-based algorithms, 58.33% are published in IEEE, 25% in Springer, and 16.67% in other publishers. For sampling-based algorithms, 50% are published in IEEE, 22.73% in Springer, 13.64% in Elsevier, and 13.63% in other publishers. For gradient-based algorithms, 72.73% are published in IEEE, 18.18% in Springer, and 9.09% in Elsevier. For optimization-based algorithms, 76.92% are published in IEEE, 15.38% in Springer, and 7.70% in Elsevier.

For interpolation-based algorithms, 48.15% are published in IEEE, 33.33% in Springer, 11.11% in Elsevier, and 7.41% in other publishers. For supervised learning-based algorithms, 44.44% of them are published in IEEE, 33.33% in Springer, 11.11% in Elsevier, and 11.12% in other publishers. For reinforcement learning-based algorithms, 56.67% are published in IEEE, 26.67% in Springer, 13.33% in Elsevier, and 3.33% in other publishers. For meta-heuristic optimization algorithms, 52.63% of them are published in IEEE, 23.68% in Springer, 15.79% in Elsevier, and 7.90% in other publishers.

Fig. 9c explores the publication type percentage for each algorithm type. For graph-based algorithms, 75% of the publications are journal

papers, while 25% are conference papers. For sampling-based algorithms, 72.72% of the publications are journal papers, while 27.28% are conference papers. For gradient-based algorithms, 90.9% of the publications are journal papers, while 9.1% are conference papers. For optimization-based algorithms, 46.15% of the publications are journal papers, while 53.85% are conference papers.

For interpolation-based algorithms, 77.78% of the publications are journal papers, while 22.22% are conference papers. For supervised learning-based algorithms, 66.67% of the publications are journal papers, while 33.33% are conference papers. For reinforcement learning-based algorithms, 73.33% of the publications are journal papers, while 26.67% are conference papers. For meta-heuristic optimization algorithms, 84.21% of the publications are journal papers, while 15.79% are conference papers.

Fig. 9d explores the hybridization percentage for each algorithm type. For graph-based algorithms, 16.67% are hybrid, while 83.33% are not hybrid. Only 4.55% of the sampling-based algorithms are hybrid. 18.18% of gradient-based algorithms are hybrid. 46.15% of optimization-based algorithms are hybrid. 44.4% of interpolation-based algorithms are hybrid. 100% of the supervised learning algorithms are not hybrid. 20% of reinforcement learning algorithms are hybrid. 39.47% of meta-heuristic optimization algorithms are hybrid.

11. Conclusion

This research has intensively explored the recent state-of-the-art algorithms in the AD system. The stages of the AD system have been discussed in detail, starting from sensors, perception, localization, mapping, risk assessment, and path planning. The stages of the AD system are well-researched. However, path planning is still one of the most challenging problems in the AD system [7].

There are three main groups of path-planning algorithms: Traditional, Machine learning, and Meta-heuristic optimization techniques. The most common traditional techniques are graph-based search methods such as Dijkstra and A* algorithms. These methods give accurate solutions for limited search space. However, they are very slow for large search spaces and give jerky routes. Future research in graph-based algorithms could focus on hybridizing them with sampling-based and optimization techniques for enhanced search capability for complex search spaces.

The sampling-based methods such as RRT have been used to solve the path planning problem. These algorithms are faster than graph search, but they give jerky solutions. Improving Sampling-Based Planning methods should involve integrating optimization and interpolation techniques for enhanced efficiency and smoothness.

A gradient-based algorithm such as the APF algorithm has been used to address the trajectory planning in the ADS. The main merit of this technique is that it can produce collision-free trajectories within a small computation time. Nevertheless, the main demerit of the algorithm is the high possibility of falling into a local minimum because of the gradient behavior. The future direction for Gradient-Based methods should focus on overcoming the challenge of local minima via hybridization with global search methods that can add diversity to the solutions.

Interpolation-based algorithms should primarily be utilized to smooth out paths generated by other methods. However, they are too slow and have high computational time if used as an independent technique. Numerical optimization methods such as quadratic programming can deal with complex search spaces and give high-quality solutions but have high computational time. Future developments in optimization-based methods should reduce computational complexity using appropriate optimizers, making them problem-dependent.

Supervised and deep learning techniques have been deployed in two manners: they can be used only in the perception layer to deal with images and sensory data, or they can be used as an End-To-End driving scheme to include the path-planning phase. It can give a high-speed solution for familiar scenarios during the application phase. However, the main challenge is the need for vast, complex training data to build reliable models. Moreover, the need to re-train the whole model to make any updates to the model is too time-consuming. The future of Deep Supervised Learning Techniques should focus on enhancing real-time processing capabilities by using pre-trained models with the capability of real-time training, which reduces reliance on extensive training datasets.

Reinforcement learning techniques have recently been deployed to provide optimal and fast solutions for complex search spaces in the long term. Unlike traditional supervised learning, they do not need training data to learn. They interact with the environment to update the policy. The main challenge of traditional reinforcement learning is that generating a state–action table will be too massive for complex problems, which require more memory and high computational time to search for suitable action.

Deep Reinforcement learning can best use neural networks as a function approximator instead of storing a vast state–action Q-table. However, this adds more parameter tuning complexity to train the neural network, which is too time-consuming. The future of Deep Reinforcement Learning techniques should emphasize using pre-trained deep Q networks, which can make it faster; however, it is still challenging.

Meta-heuristic optimization techniques such as GA, ABC, PSO, SA, ACO, and CSA have recently been widely used to solve path-planning problems, 23% of the literature. The main advantage of meta-heuristic optimization algorithms is that they are considered general problem-solver and problem-independent. They can give an optimal solution for any complex search space. Therefore, it can be applied to any problem with few modifications. Furthermore, they rarely get stuck in the local minimum because of the ability to explore different search spaces and utilize local search spaces. Nevertheless, they can provide different paths for the same problem if run multiple times. The reason behind this problem is the random behavior in specific stages of the algorithms. The main challenge of these algorithms is the parameter tuning to balance exploitation and exploration. The most trending methodology to address this challenge is using adaptive rather than static parameters; this adds the appropriate balance between global and local search.

In this study, we claim that we proposed a comprehensive and extensive review with high algorithm diversity in the path planning problem in the ADS. This paper can mark a starting point for future research on solving the path planning problem. This review can give an intuition about the most recent techniques used in solving the path planning problem of the ADS. Based on the cons of the recent state-of-the-art algorithms, new ideas can be generated to solve the path planning problem more efficiently. Therefore, as described in this paper, the research community can start from where the literature ended.

CRediT authorship contribution statement

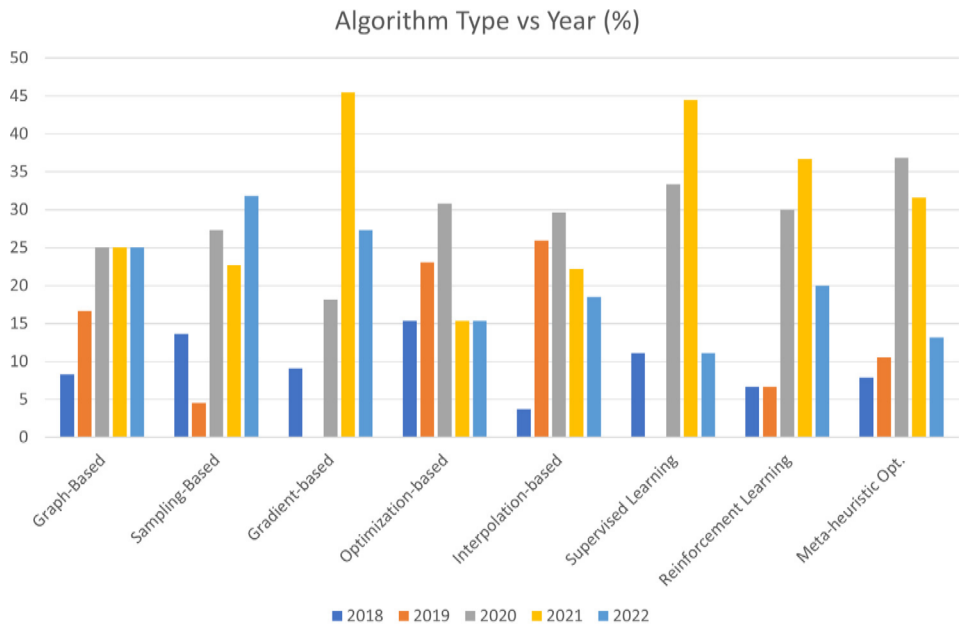
Mohamed Reda: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft. **Ahmed Onsy:** Conceptualization, Funding acquisition, Investigation, Project administration, Resources, Supervision, Writing – review & editing. **Ali Ghanbari:** Investigation, Project administration, Supervision, Writing – review & editing. **Amira Y. Haikal:** Investigation, Supervision, Writing – review & editing.

Declaration of competing interest

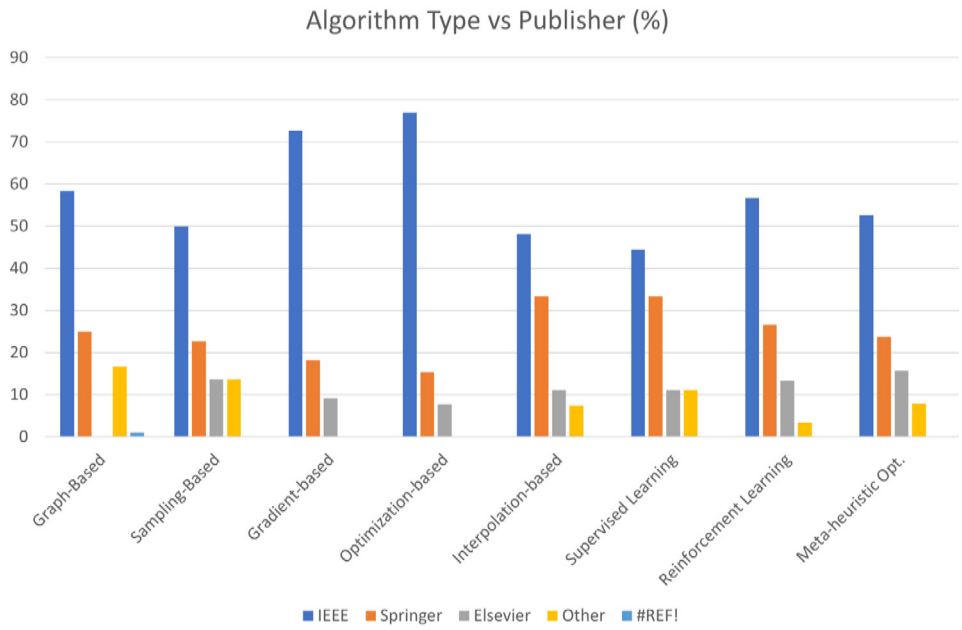
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

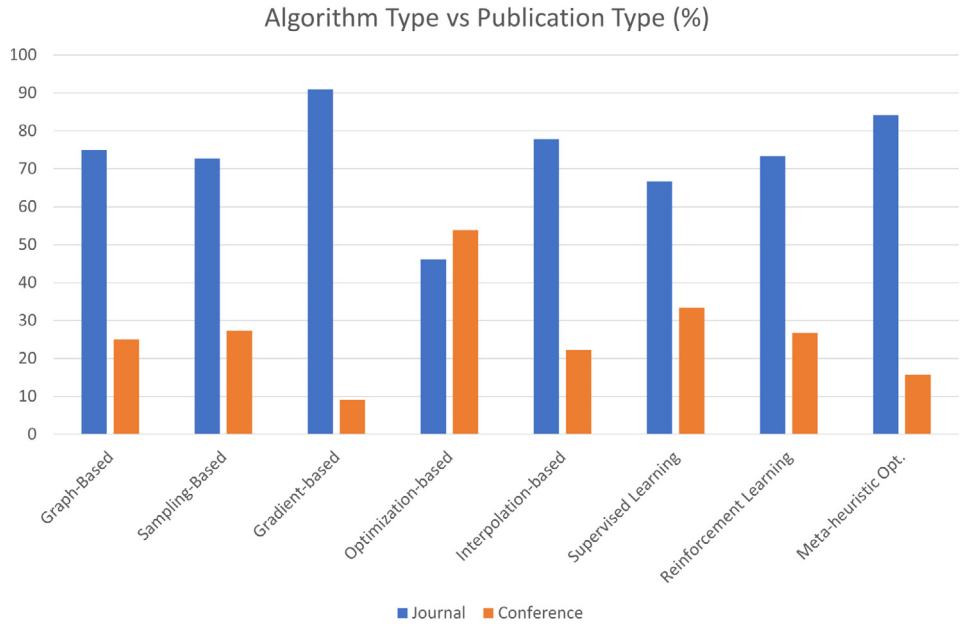


(a) Year percentage with algorithm type.

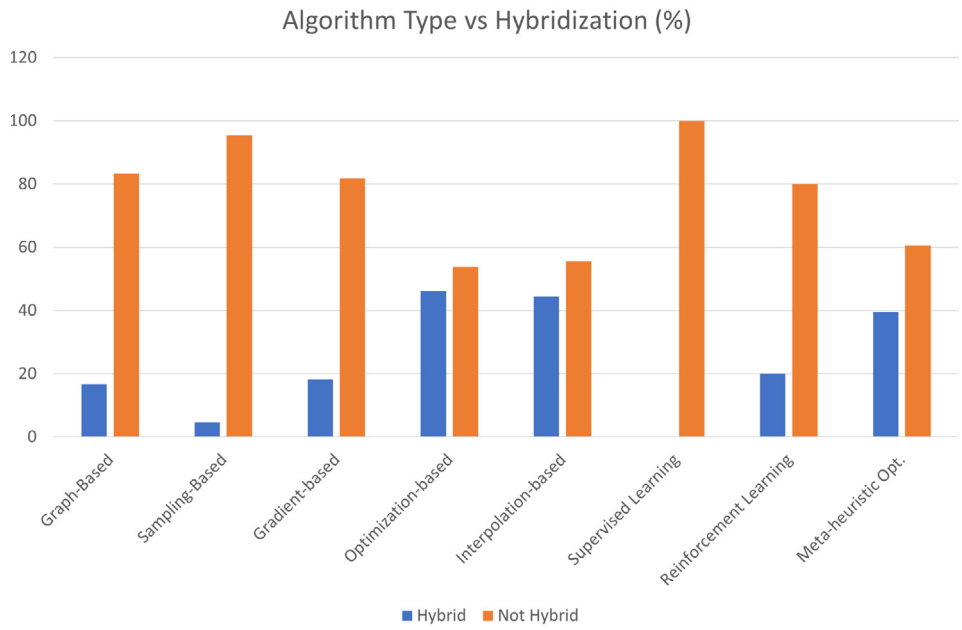


(b) Publisher percentage with algorithm type.

Fig. 9. Algorithm prospective of the state-of-the-art path planning algorithms.



(c) Publication type percentage with algorithm type.



(d) Hybridization percentage with algorithm type.

Fig. 9. (*continued*).

Table 11
List of abbreviations.

Symbol	Acronym/Abbreviation
A*ESS	A* algorithm with Equal-Step Sampling
A*-PDWA	A* algorithm and Predictive-Dynamic Window Approach
aABC	Arrhenius Artificial Bee Colony
AAIRL	Augmented Adversarial Inverse Reinforcement Learning
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ACRL	Actor-Critic Reinforcement Learning
ACS	Ant Colony System
ADAPF	Azimuth and Distance Artificial Potential Field
ADL-ABC	Adaptive Dimension Limit-Artificial Bee Colony
ADP	Adaptive Dynamic Programming
ADPF-PP	Adaptive Potential Field for Path Planning
ADS	Automated driving system
AHPFs	Artificial Harmonic Potential Fields
AI	Artificial Intelligence
AIACSE	Adaptive Improved Ant Colony System algorithm based on population information Entropy
ANN	Artificial Neural Networks
APF	Artificial Potential Field
ARRT-ExGWO	Adapted Rapidly-exploring Random Tree with Extended Grey Wolf Optimization
ARRT-GWO	Adapted Rapidly-exploring Random Tree with Grey Wolf Optimization
ASMP-NCSS	Adaptive Sampling-based Motion Planning with a Non-Conservatively Defensive Strategy
BAS	Beetle Antennae Search
BAS-PSO	Beetle Antennae Search and Particle Swarm Optimization
BB	Bernstein–Bézier
BC-based	Bezier Curve-based
BCO	Bézier Curve Optimization
BFO	Bacterial Foraging Optimization
Bi-RRT	Bidirectional Rapidly-exploring Random Tree
BOA	Butterfly Optimization Algorithm
BSAS	Beetle swarm antennae search
BSCO	B-Spline Curve Optimization
BSO	Beetle Swarm Optimization
CBA	Clothoid-Based Algorithm
CCO	Clothoid Curves Optimization
CDQN	Conditional Deep Q-network
CDT	Constrained Delaunay Triangulation
CNM	Complete Node Mechanism
CNN	Convolution Neural Networks
Co-GLABC	Coevolution framework with the Global best Leading Artificial Bee Colony
C-PDF	Custom Probability Density Function
CSA	Cuckoo Search Algorithm
CSO	Chicken Swarm Optimization
CSOA	Cat Swarm Optimization Algorithm
CVAE	Conditional Variational Encoder
DC	dynamic cell
DCNN	Deep Convolutional Neural Network
DDCS	Discrete Damped Cuckoo Search
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q-network
DDQN-FSLAM	Dueling Deep Q-Network with a Fast active SLAM
DE	Differential Evolution
DL	Deep Learning
DNN	Deep Neural Network
DPCC	Dynamic Programming-based algorithm integrated with Clothoid Curve
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
D-RRT*	Directional Rapidly-exploring Random Trees*
DSUNet-PP	Depth-wise Separable UNet for Path Prediction
DVD-CC	Direct Visibility Diagram Algorithm with Clothoid Curves
DWA	Dynamic Window Approach
DynEFWA-APF	Dynamic Enhanced Firework Algorithm with Artificial Potential Field
EBHS	Experience-Based Heuristic-Search
EES-PSO	Evolutionary Scatter Search Particle Swarm Optimization
EMCOA	Enhanced Mutated Cuckoo Optimization Algorithm
EMOPSP	Enhanced Multi-Objective Particle Swarm Optimization
ERRT	Enhanced Rapidly-exploring Random Tree
ET-MPADP	Event-Triggered Model Predictive Adaptive Dynamic Programming
FA	Firefly Algorithm
FA-D*	Firefly algorithm and the D* algorithm
FDA-DP	Feasible Direction Algorithm integrated with Dynamic Programming (
FDP	Forward Dynamic Programming
FFNN	Feed Forward Neural Networks
FMCW	Frequency-Modulated Continuous Wave
FOBC	Fourth-Order Bézier Curves

(*continued on next page*)

Table 11 (*continued*).

Symbol	Acronym/Abbreviation
GA	Genetic Algorithm
GA-PF	Genetic Algorithm-Potential Field
GARL	Graph Attention Reinforcement Learning
GDA	Gradient Descent Algorithm
GDBC	Gradient Descent and B\'ezier Curve algorithm
GDM	Gradient Descent Method
GPS-IMU	Global Positioning System and Inertial Measurement Unit
GWO	Grey Wolf Optimization
HA*	Hybrid A*
HARL	Hybrid A* Algorithm and Reinforcement Learning
hDDQN	Hierarchical Double Deep Q-learning
HDM	Human Driver Model
HDM-MIQP	Human Driver Model integrated with Mixed-integer quadratic programming
HDR	High Dynamic Range
HIL	hardware-in-loop
HS	Harmony Search
IA*-DWA	Improved A* algorithm and Dynamic Window Approach (DWA)
IA*FC	Improved A* based on the Fuel Consumption
iABC–EP	Improved Artificial Bee Colony with Evolutionary Programming
IACO-PSO	Improved Ant Colony Algorithm optimized by Particle Swarm Algorithm
iADA*	Improved Anytime Dynamic A*
IAPF	Improved Artificial Potential Field
IAPF-GDM	Improved Artificial Potential Field and Gradient Descent Method
IBSO	Improved Beetle Swarm Optimization
ICP	Iterative closest point
ICSO	Improved Chicken swarm optimization
IDQNPER-ETE	Deep Reinforcement Learning in an End-To-End driving scheme
IDWAQ	Improved Dynamic Window Approach based on the Q-Learning
IGWO	Improved Grey Wolf Optimization
ILPSO	Improved Localized Particle Swarm Optimization
ILS	Iterated Local Search
IMFO	Improved Moth–Flame Optimization
IPF-PP	Improved Potential Field-based Path Planning
IPSO	Improved Particle Swarm Optimization
IPSO-APF/SA	Improved PSO, Improved Artificial Potential Field , and Simulated Annealing algorithm
IQL	Improved Q-learning
IRL	Integral Reinforcement Learning
IRRT	Improved Rapidly-exploring Random Trees
i-RRT	improved Rapidly-exploring Random Trees
IRRT-PI	Improved Rapidly-exploring Random Tree associated with PI controller
ISA	Improved Simulated Annealing
JPS	Jump Point Search
JQBC	Joint Quadratic Bézier Curves
KB-RRT*	Kinematic Constrained Bi-directional Rapidly-exploring Random Tree
LFS-BAS	Local Fast Search with Beetle Antennae Search
LiDAR	Light Detection And Ranging
LLC	Longitudinal and Lateral planning with Clothoid interpolation
LSAC-CPP	Lyapunov-based Soft Actor–Critics with Collision Probability Prediction
LSPP	Path Planning with a Line Segment
LSTM	long short-term memory
LTSTP	Long-Term and Short-Term Planner
MCAL-P	Mobile robot Collision Avoidance Learning with Path
MCGA	Modified Crossover Genetic Algorithm
MCQ	Monte Carlo Q-learning
MCS/AC	Monte Carlo Search and actor-critic
MDL	Modified Dogleg
MDP	Markov decision process
MEP-DIRL	Maximum Entropy Deep Inverse Reinforcement Learning algorithm
MFA	Modified Firefly Algorithm
MFO	Moth–Flame Optimization
ML	Machine Learning
MOCMCSO	Multi-Objective Cauchy Mutation Cat Swarm
MPC	Model Predictive Control
MPPP	Model Predictive Path Planning
MPSO	Mutation Particle Swarm Optimization
FDA-DP	Feasible Direction Algorithm integrated with Dynamic Programming (
FDP	Forward Dynamic Programming
FFNN	Feed Forward Neural Networks
FMCW	Frequency-Modulated Continuous Wave
FOBC	Fourth-Order Bézier Curves
GA	Genetic Algorithm
GA-PF	Genetic Algorithm-Potential Field
GARL	Graph Attention Reinforcement Learning
GDA	Gradient Descent Algorithm

(*continued on next page*)

Table 11 (*continued*).

Symbol	Acronym/Abbreviation
GDBC	Gradient Descent and B\'ezier Curve algorithm
GDM	Gradient Descent Method
GPS-IMU	Global Positioning System and Inertial Measurement Unit
GWO	Grey Wolf Optimization
HA*	Hybrid A*
HARL	Hybrid A* Algorithm and Reinforcement Learning
hDDQN	Hierarchical Double Deep Q-learning
HDM	Human Driver Model
HDM-MIQP	Human Driver Model integrated with Mixed-integer quadratic programming
HDR	High Dynamic Range
HIL	hardware-in-loop
HS	Harmony Search
IA*-DWA	Improved A* algorithm and Dynamic Window Approach (DWA)
IA*FC	Improved A* based on the Fuel Consumption
iABC–EP	Improved Artificial Bee Colony with Evolutionary Programming
IACO-PSO	Improved Ant Colony Algorithm optimized by Particle Swarm Algorithm
iADA*	Improved Anytime Dynamic A*
IAPF	Improved Artificial Potential Field
IAPF-GDM	Improved Artificial Potential Field and Gradient Descent Method
IBSO	Improved Beetle Swarm Optimization
ICP	Iterative closest point
ICSO	Improved Chicken swarm optimization
IDQNPER-ETE	Deep Reinforcement Learning in an End-To-End driving scheme
IDWAQ	Improved Dynamic Window Approach based on the Q-Learning
IGWO	Improved Grey Wolf Optimization
ILPSO	Improved Localized Particle Swarm Optimization
ILS	Iterated Local Search
IMFO	Improved Moth–Flame Optimization
IPF-PP	Improved Potential Field-based Path Planning
IPSO	Improved Particle Swarm Optimization
IPSO-APF/SA	Improved PSO, Improved Artificial Potential Field , and Simulated Annealing algorithm
IQL	Improved Q-learning
IRL	Integral Reinforcement Learning
IRRT	Improved Rapidly-exploring Random Trees
i-RRT	Improved Rapidly-exploring Random Trees
IRRT-PI	Improved Rapidly-exploring Random Tree associated with PI controller
ISA	Improved Simulated Annealing
JPS	Jump Point Search
JQBC	Joint Quadratic Bézier Curves
KB-RRT*	Kinematic Constrained Bi-directional Rapidly-exploring Random Tree
LFS-BAS	Local Fast Search with Beetle Antennae Search
LiDAR	Light Detection And Ranging
LLC	Longitudinal and Lateral planning with Clothoid interpolation
LSAC-CPP	Lyapunov-based Soft Actor–Critics with Collision Probability Prediction
LSPP	Path Planning with a Line Segment
LSTM	Long short-term memory
LTSTP	Long-Term and Short-Term Planner
MCAL-P	Mobile robot Collision Avoidance Learning with Path
MCGA	Modified Crossover Genetic Algorithm
MCQ	Monte Carlo Q-learning
MCS/AC	Monte Carlo Search and actor-critic
MDL	Modified Dogleg
MDP	Markov decision process
MEP-DIRL	Maximum Entropy Deep Inverse Reinforcement Learning algorithm
MFA	Modified Firefly Algorithm
MFO	Moth–Flame Optimization
ML	Machine Learning
MOCMCSO	Multi-Objective Cauchy Mutation Cat Swarm
MPC	Model Predictive Control
MPPP	Model Predictive Path Planning
MPSO	Mutation Particle Swarm Optimization
MRCACO	Multi-Role Adaptive Collaborative Ant Colony Optimization
MRRT*	Modified Rapidly-exploring Random Trees*
MSM	Minimum Snap Method
NDT	Normal distributions transform
NHTSA	National Highway Traffic Safety Administration
NOC	Nonlinear Optimal Control
NRRT*	Neural Rapidly-exploring Random Tree*
ODDs	Operational design domains
OSQP	Open Source Quadratic Programming
PAPF	Predictive Artificial Potential Field
PCAPF	Polynomial Curve and Artificial Potential Field
PCC	Parameterized Curvature Control
P-DWA	Predictive-Dynamic Window Approach
PF	Potential Field

(*continued on next page*)

Table 11 (*continued*).

Symbol	Acronym/Abbreviation
PFABC	Potential Field Artificial Bee Colony
PFPSO	Potential Field Particle Swarm Optimization
PJM	Piecewise Jerk Method
PRM	Probabilistic Roadmap method
PRRT	Pruning Rabidly-exploring Random Tree
PSO	particle swarm optimization
QLOR	Q-learning algorithm and Linear Output Regulation algorithm
QOBL	Quasi-opposition-based learning
QP	Quadratic programming
QPO	Quadratic Programming Optimization
QTBC	Quintic Trigonometric Bézier Curve
RAS	Rank-based Ant System
RHRL	Robust Hierarchical Reinforcement Learning
RL	Reinforcement Learning
RLDA	reverse labeling Dijkstra algorithm
RL-MOHH	Reinforcement Learning based Multi-Objective Hyper-Heuristic
RNN	Recurrent Neural Networks
ROS	Robot operating system
RPN	Region Proposal Networks
RRT	Rapidly-exploring Random Tree
RRT-CPDF	Rapidly-exploring Random Tree with a Custom Probability Density Function
RRT-DWA	Random Tree algorithm and the Dynamic Window Approach algorithm
RTQL	Real-time Q-learning
SA	Simulated Annealing
SAC	Soft Actor–Critics
SAE	Society of Automotive Engineers
SAPSO	Simulated Annealing algorithm and the Particle Swarm Optimization
SA-TLBO	Simulated Annealing Teaching Learning Based Optimization
SBDP-QP	Sampling-based algorithm, Dynamic Programming, and Quadratic Programming
SBP	Sampling-Based Planning
SDGT-PC	Stackelberg Differential Game Theory and Polynomial Curve
SEHS	Space Exploration Guided Heuristic Search
SIO-BAS	Searching Information Orientation with Beetle Antennae Search
SLAM	Simultaneous Localization And Mapping
S-NO	Sampling and Numerical Optimization algorithm
SPS	Surrounding Point Set
SPSO	Satellite Particle Swarm Optimization
SQP	Sequential Quadratic Programming
SR1	Symmetric Rank-1
SSD	Single Shot Detector
SUMO	Simulation of Urban Mobility
TCSC	Tentacle algorithm and the B-Spline Curve algorithm
TEB	Timed Elastic Band
TFA	Trajectory Following Approach
THA*	Traversability Hybrid A*
TLBO	Teaching-Learning-Based Optimization
TS	Tabu Search
TSQP	Two-Step Quadratic Programming
TT-MPADP	Time-triggered Model Predictive Adaptive Dynamic Programming
UAV	Unmanned Aerial Vehicle
U-PDF	Uniform Probability Density Function
VD	Visibility Diagram
VDQN	Vanilla Deep Q-Network
VFH-	Vector Field Histogram-
vSLAM	visual Simultaneous Localization And Mapping
WOA	Whale Optimizer Algorithm
YOLO	You Only Look Once

Appendix. List of abbreviations

See Table 11.

References

[1] É. Zablocki, H. Ben-Younes, P. Pérez, M. Cord, Explainability of deep vision-based autonomous driving systems: Review and challenges, Int. J. Comput. Vis. 130 (10) (2022) 2425–2452.

[2] C. Badue, R. Guidolini, R.V. Carneiro, P. Azevedo, V.B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T.M. Paixao, F. Mutz, et al., Self-driving cars: A survey, Expert Syst. Appl. 165 (2021) 113816.

[3] A. Chougule, V. Chamola, A. Sam, F.R. Yu, B. Sikdar, A comprehensive review on limitations of autonomous driving and its impact on accidents and collisions, IEEE Open J. Veh. Technol. (2023) 1–20, http://dx.doi.org/10.1109/OJVT.2023.3335180.

[4] J. Zhao, W. Zhao, B. Deng, Z. Wang, F. Zhang, W. Zheng, W. Cao, J. Nan, Y. Lian, A.F. Burke, Autonomous driving system: A comprehensive survey, Expert Syst. Appl. (2023) 122836.

[5] Kent County Council, Crash and Casualty Data, Tech. Rep., Kent County Council, 2022, https://www.kent.gov.uk/roads-and-travel/road-safety/crash-and-casualty-data. (Online; accessed 28 October 2022).

[6] T.J. Crayton, B.M. Meier, Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy, J. Transp. Health 6 (2017) 245–252.

[7] E. Yurtsever, J. Lambert, A. Carballo, K. Takeda, A survey of autonomous driving: Common practices and emerging technologies, IEEE Access 8 (2020) 58443–58469.

[8] M. Maurer, J.C. Gerdes, B. Lenz, H. Winner, Autonomous Driving: Technical, Legal and Social Aspects, Springer Nature, 2016.

[9] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, A. Mouzakitis, A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications, IEEE Internet Things J. 5 (2) (2018) 829–846.

[10] D. Gruyer, S. Demmel, V. Magnier, R. Belaroussi, Multi-hypotheses tracking using the Dempster–Shafer theory, application to ambiguous road context, Inf. Fusion 29 (2016) 40–56.

[11] C. Gold, M. Körber, D. Lechner, K. Bengler, Taking over control from highly automated vehicles in complex traffic situations: The role of traffic density, Human Factors 58 (4) (2016) 642–652.

[12] D. González, J. Pérez, V. Milanés, F. Nashashibi, A review of motion planning techniques for automated vehicles, IEEE Trans. Intell. Transp. Syst. 17 (4) (2015) 1135–1145.

[13] L. Claussmann, M. Revilloud, D. Gruyer, S. Glaser, A review of motion planning for highway autonomous driving, IEEE Trans. Intell. Transp. Syst. 21 (5) (2019) 1826–1848.

[14] S. Aradi, Survey of deep reinforcement learning for motion planning of autonomous vehicles, IEEE Trans. Intell. Transp. Syst. (2020).

[15] A. Yoganandhan, S. Subhash, J.H. Jothi, V. Mohanavel, Fundamentals and development of self-driving cars, Mater. Today: Proc. 33 (2020) 3303–3310.

[16] B. Hadi, A. Khosravi, P. Sarhadi, A review of the path planning and formation control for multiple autonomous underwater vehicles, J. Intell. Robot. Syst. 101 (4) (2021) 1–26.

[17] C. Zhou, B. Huang, P. Fränti, A review of motion planning algorithms for intelligent robots, J. Intell. Manuf. (2021) 1–38.

[18] A. Puente-Castro, D. Rivero, A. Pazos, E. Fernandez-Blanco, A review of artificial intelligence applied to path planning in UAV swarms, Neural Comput. Appl. (2021) 1–18.

[19] Y. Huang, D. Chen, Research progress of automatic driving path planning, in: 2021 2nd International Conference on Artificial Intelligence and Computer Engineering, ICAICE, IEEE, 2021, pp. 95–99.

[20] A.K. Tyagi, S. Aswathy, Autonomous intelligent vehicles (AIV): Research statements, open issues, challenges and road for future, Int. J. Intell. Netw. 2 (2021) 83–102.

[21] A. Vagale, R. Oucheikh, R.T. Bye, O.L. Osen, T.I. Fossen, Path planning and collision avoidance for autonomous surface vehicles I: A review, J. Mar. Sci. Technol. (2021) 1–15.

[22] F. Ye, S. Zhang, P. Wang, C.-Y. Chan, A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles, in: 2021 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2021, pp. 1073–1080.

[23] X. Xiao, B. Liu, G. Warnell, P. Stone, Motion planning and control for mobile robot navigation using machine learning: A survey, Auton. Robots (2022) 1–29.

[24] H.A. Ignatious, M. Khan, et al., An overview of sensors in autonomous vehicles, Procedia Comput. Sci. 198 (2022) 736–741.

[25] E. Marti, M.A. De Miguel, F. Garcia, J. Perez, A review of sensor technologies for perception in automated driving, IEEE Intell. Transp. Syst. Mag. 11 (4) (2019) 94–108.

[26] N. Pinchon, O. Cassignol, A. Nicolas, F. Bernardin, P. Leduc, J.-P. Tarel, R. Brémond, E. Bercier, J. Brunet, All-weather vision for automotive safety: Which spectral band? in: International Forum on Advanced Microsystems for Automotive Applications, Springer, 2018, pp. 3–15.

[27] R. O'Malley, M. Glavin, E. Jones, A review of automotive infrared pedestrian detection techniques, in: IET Irish Signals and Systems Conference, ISSC 2008, IET, 2008, pp. 168–173.

[28] B. Pueo, High speed cameras for motion analysis in sports science, J. Hum. Sport Exercise 11 (1) (2016) 53–73.

[29] R.A. Hamzah, H. Ibrahim, Literature survey on stereo vision disparity map algorithms, J. Sensors 2016 (2016).

[30] P. Garbat, W. Skarbek, M. Tomaszewski, Structured light camera calibration, Opto-Electron. Rev. 21 (1) (2013) 23–38.

[31] M. Hansard, S. Lee, O. Choi, R.P. Horaud, Time-of-Flight Cameras: Principles, Methods and Applications, Springer Science & Business Media, 2012.

[32] J. Binas, D. Neil, S.-C. Liu, T. Delbruck, DDD17: End-to-end DAVIS driving dataset, 2017, arXiv preprint arXiv:1711.01458.

[33] M. Schönbein, A. Geiger, Omnidirectional 3d reconstruction in augmented Manhattan worlds, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 716–723.

[34] D. Scaramuzza, R. Siegwart, Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles, IEEE Trans. Robot. 24 (5) (2008) 1015–1026.

[35] C. Jang, C. Kim, K. Jo, M. Sunwoo, Design factor optimization of 3D flash lidar sensor based on geometrical model for automated vehicle and advanced driver assistance system applications, Int. J. Automot. Technol. 18 (1) (2017) 147–156.

[36] Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, T. Harada, MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2017, pp. 5108–5115.

[37] J. Hasch, Driving towards 2020: Automotive radar technology trends, in: 2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, ICMIM, IEEE, 2015, pp. 1–4.

[38] D. Kissinger, Millimeter-Wave Receiver Concepts for 77 GHz Automotive Radar in Silicon-Germanium Technology, Springer Science & Business Media, 2012.

[39] M.E. Warren, Automotive LIDAR technology, in: 2019 Symposium on VLSI Circuits, IEEE, 2019, pp. C254–C255.

[40] M. Wang, W. Liu, Y. Lu, X. Zhao, B. Song, Y. Zhang, Y. Wang, C. Lian, J. Chen, Y. Cheng, et al., Study on the measurement of the atmospheric extinction of fog and rain by forward-scattering near infrared spectroscopy, Guang Pu Xue Yu Guang Pu Fen Xi=Guang Pu 28 (8) (2008) 1776–1780.

[41] P. Radecki, M. Campbell, K. Matzen, All weather perception: Joint data association, tracking, and classification for autonomous ground vehicles, 2016, arXiv preprint arXiv:1605.02196.

[42] J. Redmon, S. Divvala, R. Girshick, 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.

[43] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.

[44] J. Redmon, A. Farhadi, YOLO9000: Better, faster, stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.

[45] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: European Conference on Computer Vision, Springer, 2016, pp. 21–37.

[46] R. Rampriya, R. Suganya, et al., RSNet: Rail semantic segmentation network for extracting aerial railroad images, J. Intell. Fuzzy Systems 41 (2) (2021) 4051–4068.

[47] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2881–2890.

[48] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 801–818.

[49] H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1520–1528.

[50] R.B. Rusu, Semantic 3D object maps for everyday manipulation in human living environments, KI-Künstliche Intell. 24 (4) (2010) 345–348.

[51] W. Wang, K. Sakurada, N. Kawaguchi, Incremental and enhanced scanline-based segmentation method for surface reconstruction of sparse LiDAR data, Remote Sens. 8 (11) (2016) 967.

[52] S. Song, J. Xiao, Sliding shapes for 3d object detection in depth images, in: European Conference on Computer Vision, Springer, 2014, pp. 634–651.

[53] Y. Zhou, O. Tuzel, Voxelnet: End-to-end learning for point cloud based 3d object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.

[54] B. Li, T. Zhang, T. Xia, Vehicle detection from 3d lidar using fully convolutional network, 2016, arXiv preprint arXiv:1608.07916.

[55] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, Nuscenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11621–11631.

[56] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The Kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3354–3361.

[57] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al., Autonomous driving in urban environments: Boss and the urban challenge, J. Field Robot. 25 (8) (2008) 425–466.

[58] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, M.-M. Meinecke, Stereo-camera-based urban environment perception using occupancy grid and object tracking, IEEE Trans. Intell. Transp. Syst. 13 (1) (2011) 154–165.

[59] J. Shi, et al., Good features to track, in: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 1994, pp. 593–600.

[60] J. Lambert, L. Liang, L.Y. Morales, N. Akai, A. Carballo, E. Takeuchi, P. Narksri, S. Seiya, K. Takeda, Tsukuba challenge 2017 dynamic object tracks dataset for pedestrian behavior analysis, J. Robot. Mechatronics 30 (4) (2018) 598–612.

[61] J. Ziegler, P. Bender, T. Dang, C. Stiller, Trajectory planning for Bertha—A local, continuous method, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, IEEE, 2014, pp. 450–457.

[62] A. Petrovskaya, S. Thrun, Model based vehicle detection and tracking for autonomous urban driving, Auton. Robots 26 (2) (2009) 123–139.

[63] M. He, E. Takeuchi, Y. Ninomiya, S. Kato, Precise and efficient model-based vehicle tracking method using Rao-Blackwellized and scaling series particle filters, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2016, pp. 117–124.

[64] D. Held, S. Thrun, S. Savarese, Learning to track at 100 fps with deep regression networks, in: European Conference on Computer Vision, Springer, 2016, pp. 749–765.
[65] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, et al., An empirical evaluation of deep learning on highway driving, 2015, arXiv preprint arXiv:1504.01716.
[66] C. Fernandez, D. Fernandez-Llorca, M.A. Sotelo, A hybrid vision-map method for urban road detection, J. Adv. Transp. 2017 (2017).
[67] M. Paton, K. MacTavish, C.J. Ostafew, T.D. Barfoot, It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images, in: 2015 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2015, pp. 1519–1526.
[68] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, N. Akai, N. Kawaguchi, A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles, in: 2018 21st International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2018, pp. 497–504.
[69] A.S. Huang, D. Moore, M. Antone, E. Olson, S. Teller, Finding multiple lanes in urban road networks with vision and lidar, Auton. Robots 26 (2) (2009) 103–122.
[70] A.V. Nefian, G.R. Bradski, Detection of drivable corridors for off-road autonomous navigation, in: 2006 International Conference on Image Processing, IEEE, 2006, pp. 3025–3028.
[71] R. Labayrade, J. Douret, J. Laneurit, R. Chapuis, A reliable and robust lane detection system based on the parallel use of three algorithms for driving safety assistance, IEICE Trans. Inf. Syst. 89 (7) (2006) 2092–2100.
[72] R. Danescu, S. Nedevschi, Probabilistic lane tracking in difficult road scenarios using stereovision, IEEE Trans. Intell. Transp. Syst. 10 (2) (2009) 272–282.
[73] J. Levinson, M. Montemerlo, S. Thrun, Map-based precision vehicle localization in urban environments, in: Robotics: Science and Systems, vol. 4, Citeseer, 2007, p. 1.
[74] G. Bresson, Z. Alsayed, L. Yu, S. Glaser, Simultaneous localization and mapping: A survey of current trends in autonomous driving, IEEE Trans. Intell. Veh. 2 (3) (2017) 194–220.
[75] X. Qu, B. Soheilian, N. Paparoditis, Vehicle localization using mono-camera and geo-referenced traffic signs, in: 2015 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2015, pp. 605–610.
[76] M. Magnusson, The Three-Dimensional Normal-Distributions Transform: An Efficient Representation for Registration, Surface Analysis, and Loop Detection (Ph.D. thesis), Örebro universitet, 2009.
[77] N. Akai, L.Y. Morales, E. Takeuchi, Y. Yoshihara, Y. Ninomiya, Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching, in: 2017 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2017, pp. 1356–1363.
[78] E. Yurtsever, K. Takeda, C. Miyajima, Traffic trajectory history and drive path generation using GPS data cloud, in: 2015 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2015, pp. 229–234.
[79] C.M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, D. Cao, Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey, IEEE Trans. Intell. Transp. Syst. 19 (3) (2017) 666–676.
[80] M. Reda, A. Onsy, M.A. Elhosseini, A.Y. Haikal, M. Badawy, A discrete variant of cuckoo search algorithm to solve the travelling salesman problem and path planning for autonomous trolley inside warehouse, Knowl.-Based Syst. (2022) 109290.
[81] D. Delling, A.V. Goldberg, A. Nowatzyk, R.F. Werneck, PHAST: Hardware-accelerated shortest path trees, J. Parallel Distrib. Comput. 73 (7) (2013) 940–952.
[82] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.
[83] W. Yijing, L. Zhengxuan, Z. Zhiqiang, L. Zheng, Local path planning of autonomous vehicles based on A* algorithm with equal-step sampling, in: 2018 37th Chinese Control Conference, CCC, IEEE, 2018, pp. 7828–7833.
[84] R. Udomsil, T. Sangpet, T. Sapsaman, Environment generation from real map to investigate path planning and obstacle avoidance algorithm for electric vehicle, in: 2019 Research, Invention, and Innovation Congress, RI2C, IEEE, 2019, pp. 1–5.
[85] Y. Song, Z. Wang, Path planning simulation based on improved A* algorithm, J. Changchun Univ. Technol. 40 (2) (2019) 138–141.
[86] L. Yeong-Ho, K. Yeong-Jun, J. Da-Un, W. Ihn-Sik, Development of an integrated path planning algorithm for autonomous driving of unmanned surface vessel, in: 2020 20th International Conference on Control, Automation and Systems, ICCAS, IEEE, 2020, pp. 27–32.
[87] X. Zhong, J. Tian, H. Hu, X. Peng, Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment, J. Intell. Robot. Syst. 99 (1) (2020) 65–77.
[88] A.A. Maw, M. Tyan, J.-W. Lee, iADA*: Improved anytime path planning and replanning algorithm for autonomous vehicle, J. Intell. Robot. Syst. 100 (3) (2020) 1005–1013.
[89] M. Thoresen, N.H. Nielsen, K. Mathiassen, K.Y. Pettersen, Path planning for UGVs based on traversability hybrid A, IEEE Robot. Autom. Lett. 6 (2) (2021) 1216–1223.
[90] D.-D. Zhu, J.-Q. Sun, A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute, IEEE Access 9 (2021) 19761–19775.
[91] L.-s. Liu, J.-f. Lin, J.-x. Yao, D.-w. He, J.-s. Zheng, J. Huang, P. Shi, Path planning for smart car based on Dijkstra algorithm and dynamic window approach, Wirel. Commun. Mob. Comput. 2021 (2021).
[92] T. Liu, J. Zhang, An improved path planning algorithm based on fuel consumption, J. Supercomput. (2022) 1–31.
[93] D. Kim, G. Kim, H. Kim, K. Huh, A hierarchical motion planning framework for autonomous driving in structured highway environments, IEEE Access 10 (2022) 20102–20117.
[94] Y. Li, R. Jin, X. Xu, Y. Qian, H. Wang, S. Xu, Z. Wang, A mobile robot path planning algorithm based on improved A* algorithm and dynamic window approach, IEEE Access (2022).
[95] M. Elbanhawi, M. Simic, Sampling-based robot motion planning: A review, Ieee Access 2 (2014) 56–77.
[96] S.M. LaValle, J.J. Kuffner Jr., Randomized kinodynamic planning, Int. J. Robot. Res. 20 (5) (2001) 378–400.
[97] W. Lim, S. Lee, M. Sunwoo, K. Jo, Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method, IEEE Trans. Intell. Transp. Syst. 19 (2) (2018) 613–626.
[98] J. Wang, S. Wu, H. Li, J. Zou, Path planning combining improved rapidly-exploring random trees with dynamic window approach in ROS, in: 2018 13th IEEE Conference on Industrial Electronics and Applications, ICIEA, IEEE, 2018, pp. 1296–1301.
[99] A.M. Varghese, V. Jisha, Motion planning and control of an autonomous mobile robot, in: 2018 International CET Conference on Control, Communication, and Computing, IC4, IEEE, 2018, pp. 17–21.
[100] M. Werling, S. Kammel, J. Ziegler, L. Gröll, Optimal trajectories for time-critical street scenarios using discretized terminal manifolds, Int. J. Robot. Res. 31 (3) (2012) 346–359.
[101] W. Lim, S. Lee, M. Sunwoo, K. Jo, Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios, IEEE Trans. Intell. Transp. Syst. 22 (1) (2019) 341–355.
[102] Z. Li, W. Zhan, L. Sun, C.-Y. Chan, M. Tomizuka, Adaptive sampling-based motion planning with a non-conservatively defensive strategy for autonomous driving, IFAC-PapersOnLine 53 (2) (2020) 15632–15638.
[103] S. Feraco, S. Luciani, A. Bonfitto, N. Amati, A. Tonoli, A local trajectory planning and control method for autonomous vehicles based on the RRT algorithm, in: 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive, AEIT AUTOMOTIVE, IEEE, 2020, pp. 1–6.
[104] J. Chen, R. Zhang, W. Han, W. Jiang, J. Hu, X. Lu, X. Liu, P. Zhao, Path planning for autonomous vehicle based on a two-layered planning model in complex environment, J. Adv. Transp. 2020 (2020).
[105] J. Wang, W. Chi, C. Li, C. Wang, M.Q.-H. Meng, Neural RRT*: Learning-based optimal path planning, IEEE Trans. Autom. Sci. Eng. 17 (4) (2020) 1748–1758.
[106] Y. Zhang, J. Zhang, J. Zhang, J. Wang, K. Lu, J. Hong, A novel learning framework for sampling-based motion planning in autonomous driving, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 1202–1209.
[107] J. Rong, S. Arrigoni, N. Luan, F. Braghin, Attention-based sampling distribution for motion planning in autonomous driving, in: 2020 39th Chinese Control Conference, CCC, IEEE, 2020, pp. 5671–5676.
[108] C. Huang, H. Huang, P. Hang, H. Gao, J. Wu, Z. Huang, C. Lv, Personalized trajectory planning and control of lane-change maneuvers for autonomous driving, IEEE Trans. Veh. Technol. 70 (6) (2021) 5511–5523.
[109] X. Jin, Z. Yan, H. Yang, Q. Wang, A practical sampling-based motion planning method for autonomous driving in unstructured environments, IFAC-PapersOnLine 54 (10) (2021) 449–453.
[110] D. Rakita, B. Mutlu, M. Gleicher, Single-query path planning using sample-efficient probability informed trees, IEEE Robot. Autom. Lett. 6 (3) (2021) 4624–4631.
[111] J. Wang, B. Li, M.Q.-H. Meng, Kinematic constrained bi-directional RRT with efficient branch pruning for robot path planning, Expert Syst. Appl. 170 (2021) 114541.
[112] Y. Zhang, S. Wang, LSPP: A novel path planning algorithm based on perceiving line segment feature, IEEE Sens. J. 22 (1) (2021) 720–731.
[113] S. Ganesan, S.K. Natarajan, J. Srinivasan, A global path planning algorithm for mobile robot in cluttered environments with an improved initial cost solution and convergence rate, Arab. J. Sci. Eng. 47 (3) (2022) 3633–3647.

[114] G.O. Flores-Aquino, J.I. Vasquez-Gomez, O. Gutierrez-Frias, Custom distribution for sampling-based motion planning, J. Braz. Soc. Mech. Sci. Eng. 44 (3) (2022) 1–15.
[115] G. Huang, Q. Ma, Research on path planning algorithm of autonomous vehicles based on improved RRT algorithm, Int. J. Intell. Transp. Syst. Res. 20 (1) (2022) 170–180.
[116] G. Yang, Y. Yao, Vehicle local path planning and time consistency of unmanned driving system based on convolutional neural network, Neural Comput. Appl. 34 (15) (2022) 12385–12398.
[117] X. Zhang, T. Zhu, Y. Xu, H. Liu, F. Liu, Local path planning of the autonomous vehicle based on adaptive improved RRT algorithm in certain lane environments, Actuators 11 (4) (2022) 109.
[118] S. Lu, Path tracking control algorithm for unmanned vehicles based on improved RRT algorithm, in: 2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information, ICETCI, IEEE, 2022, pp. 1201–1204.
[119] S. Spanogiannopoulos, Y. Zweiri, L. Seneviratne, Sampling-based non-holonomic path generation for self-driving cars, J. Intell. Robot. Syst. 104 (1) (2022) 1–17.
[120] F. Bounini, D. Gingras, H. Pollart, D. Gruyer, Modified artificial potential field method for online path planning applications, in: 2017 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2017, pp. 180–185.
[121] H. Hongyu, Z. Chi, S. Yuhuan, Z. Bin, G. Fei, An improved artificial potential field model considering vehicle velocity for autonomous driving, IFAC-PapersOnLine 51 (31) (2018) 863–867.
[122] B. Lu, G. Li, H. Yu, H. Wang, J. Guo, D. Cao, H. He, Adaptive potential field-based path planning for complex autonomous driving scenarios, IEEE Access 8 (2020) 225294–225305.
[123] P. Lin, W.Y. Choi, S.-H. Lee, C.C. Chung, Model predictive path planning based on artificial potential field and its application to autonomous lane change, in: 2020 20th International Conference on Control, Automation and Systems, ICCAS, IEEE, 2020, pp. 731–736.
[124] Z. Huang, D. Chu, C. Wu, Y. He, Path planning and cooperative control for automated vehicle platoon using hybrid automata, IEEE Trans. Intell. Transp. Syst. 20 (3) (2018) 959–974.
[125] H. Li, C. Wu, D. Chu, L. Lu, K. Cheng, Combined trajectory planning and tracking for autonomous vehicle considering driving styles, IEEE Access 9 (2021) 9453–9463.
[126] C. Huang, C. Lv, P. Hang, Y. Xing, Toward safe and personalized autonomous driving: Decision-making and motion control with DPF and CDT techniques, IEEE/ASME Trans. Mechatronics 26 (2) (2021) 611–620.
[127] J. Ji, A. Khajepour, W.W. Melek, Y. Huang, Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints, IEEE Trans. Veh. Technol. 66 (2) (2016) 952–964.
[128] Z. Zhang, L. Zheng, Y. Li, P. Zeng, Y. Liang, Structured road-oriented motion planning and tracking framework for active collision avoidance of autonomous vehicles, Sci. China Technol. Sci. 64 (11) (2021) 2427–2440.
[129] H. Li, W. Liu, C. Yang, W. Wang, T. Qie, C. Xiang, An optimization-based path planning approach for autonomous vehicles using dynEFWA-artificial potential field, IEEE Trans. Intell. Veh. (2021).
[130] P. Lin, M. Tsukada, Model predictive path-planning controller with potential function for emergency collision avoidance on highway driving, IEEE Robot. Autom. Lett. 7 (2) (2022) 4662–4669.
[131] R. Szczepanski, T. Tarczewski, K. Erwinski, Energy efficient local path planning algorithm based on predictive artificial potential field, IEEE Access 10 (2022) 39729–39742.
[132] C. Wang, Z. Wang, L. Zhang, H. Yu, D. Cao, Post-impact motion planning and tracking control for autonomous vehicles, Chin. J. Mech. Eng. 35 (1) (2022) 1–18.
[133] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, 1999.
[134] E.V. Denardo, Dynamic Programming: Models and Applications, Courier Corporation, 2012.
[135] Í.B. Viana, N. Aouf, Distributed cooperative path-planning for autonomous vehicles integrating human driver trajectories, in: 2018 International Conference on Intelligent Systems, IS, IEEE, 2018, pp. 655–661.
[136] H. Kanchwala, Path planning and tracking of an autonomous car with high fidelity vehicle dynamics model and human driver trajectories, in: 2019 IEEE 10th International Conference on Mechanical and Aerospace Engineering, ICMAE, IEEE, 2019, pp. 306–313.
[137] P.F. Lima, R. Oliveira, J. Mårtensson, B. Wahlberg, Minimizing long vehicles overhang exceeding the drivable surface via convex path optimization, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2017, pp. 1–8.
[138] R. Oliveira, P.F. Lima, G.C. Pereira, J. Mårtensson, B. Wahlberg, Path planning for autonomous bus driving in highly constrained environments, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE, 2019, pp. 2743–2749.
[139] S. Zhu, B. Aksun-Guvenc, Trajectory planning of autonomous vehicles based on parameterized control optimization in dynamic on-road environments, J. Intell. Robot. Syst. 100 (3) (2020) 1055–1067.
[140] Y. Zhang, H. Sun, J. Zhou, J. Pan, J. Hu, J. Miao, Optimal vehicle path planning using quadratic optimization for baidu apollo open platform, in: 2020 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2020, pp. 978–984.
[141] J. Changhao, H. Miaohua, S. Liyang, An autonomous vehicle motion planning method based on dynamic programming, in: 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP, IEEE, 2020, pp. 394–398.
[142] Y. Zhang, H. Chen, S.L. Waslander, J. Gong, G. Xiong, T. Yang, K. Liu, Hybrid trajectory planning for autonomous driving in highly constrained environments, IEEE Access 6 (2018) 32800–32819.
[143] J. Li, J. Gong, G. Kong, Y. Zhao, X. Zhang, A hierarchical trajectory planning framework for autonomous driving, in: 2020 3rd International Conference on Unmanned Systems, ICUS, IEEE, 2020, pp. 428–434.
[144] C. Hu, L. Zhao, G. Qu, Event-triggered model predictive adaptive dynamic programming for road intersection path planning of unmanned ground vehicle, IEEE Trans. Veh. Technol. 70 (11) (2021) 11228–11243.
[145] L. Wang, B. Wang, C. Wang, Collision-free path planning with kinematic constraints in urban scenarios, J. Shanghai Jiaotong Univ. (Science) 26 (5) (2021) 731–738.
[146] P. Typaldos, M. Papageorgiou, I. Papamichail, Optimization-based path-planning for connected and non-connected automated vehicles, Transp. Res. C 134 (2022) 103487.
[147] Y. Jiang, Z. Liu, D. Qian, H. Zuo, W. He, J. Wang, Robust online path planning for autonomous vehicle using sequential quadratic programming, in: 2022 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2022, pp. 175–182.
[148] J.P. Rastelli, R. Lattarulo, 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.
[149] H. Fuji, J. Xiang, Y. Tazaki, B. Levedahl, T. Suzuki, Trajectory planning for automated parking using multi-resolution state roadmap considering non-holonomic constraints, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, IEEE, 2014, pp. 407–413.
[150] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, S. Tsugawa, Overview of platooning systems, in: Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012), 2012, pp. 2–7.
[151] P. Petrov, F. Nashashibi, Modeling and nonlinear adaptive control for autonomous vehicle overtaking, IEEE Trans. Intell. Transp. Syst. 15 (4) (2014) 1643–1656.
[152] H.-w. Wang, X.-c. Yu, H.-b. Song, Z.-h. Lu, J. Lloret, F. You, A global optimal path planning and controller design algorithm for intelligent vehicles, Mob. Netw. Appl. 23 (5) (2018) 1165–1178.
[153] X. Hu, L. Chen, B. Tang, D. Cao, H. He, Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles, Mech. Syst. Signal Process. 100 (2018) 482–500.
[154] G. Klančar, M. Seder, S. Blažič, I. Škrjanc, I. Petrović, Drivable path planning using hybrid search algorithm based on E* and Bernstein–Bézier motion primitives, IEEE Trans. Syst., Man, Cybern.: Syst. 51 (8) (2019) 4868–4882.
[155] C. You, J. Lu, D. Filev, P. Tsiotras, Autonomous planning and control for intelligent vehicles in traffic, IEEE Trans. Intell. Transp. Syst. 21 (6) (2019) 2339–2349.
[156] S. Sedighi, D.-V. Nguyen, P. Kapsalas, K.-D. Kuhnert, Fusing direct visibility diagram with clothoid curves for motion planning, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE, 2019, pp. 3579–3586.
[157] D. Piscini, E. Pagot, G. Valenti, F. Biral, Experimental comparison of trajectory control and planning algorithms for autonomous vehicles, in: IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society, Vol. 1, IEEE, 2019, pp. 5217–5222.
[158] J. Moreau, P. Melchior, S. Victor, L. Cassany, M. Moze, F. Aioun, F. Guillemard, Reactive path planning in intersection for autonomous vehicle, IFAC-PapersOnLine 52 (5) (2019) 109–114.
[159] A.Z. Zambom, B. Seguin, F. Zhao, Robot path planning in a dynamic environment with stochastic measurements, J. Global Optim. 73 (2) (2019) 389–410.
[160] Y. Mu, B. Li, D. An, Y. Wei, Three-dimensional route planning based on the beetle swarm optimization algorithm, IEEE Access 7 (2019) 117804–117813.
[161] R.T. Kamil, M.J. Mohamed, B.K. Oleiwi, Path planning of mobile robot using improved artificial bee colony algorithm, Eng. Technol. J. 38 (9) (2020) 1384–1395.
[162] S. Luo, X. Li, Z. Sun, An optimization-based motion planning method for autonomous driving vehicle, in: 2020 3rd International Conference on Unmanned Systems, ICUS, IEEE, 2020, pp. 739–744.
[163] X. Jin, Z. Yan, G. Yin, S. Li, C. Wei, An adaptive motion planning technique for on-road autonomous driving, IEEE Access 9 (2020) 2655–2664.

[164] Y. Dai, S.-G. Lee, Perception, planning and control for self-driving system based on on-board sensors, Adv. Mech. Eng. 12 (9) (2020) 1687814020956494.
[165] Á. Fehér, S. Aradi, T. Bécsi, Hierarchical evasive path planning using reinforcement learning and model predictive control, IEEE Access 8 (2020) 187470–187482.
[166] E.D. Lambert, R. Romano, D. Watling, Optimal smooth paths based on clothoids for car-like vehicles in the presence of obstacles, Int. J. Control Autom. Syst. 19 (6) (2021) 2163–2182.
[167] V. Bulut, Path planning for autonomous ground vehicles based on quintic trigonometric Bézier curve, J. Braz. Soc. Mech. Sci. Eng. 43 (2) (2021) 1–14.
[168] D. Zeng, Z. Yu, L. Xiong, J. Zhao, P. Zhang, Z. Fu, A Steerable Curvature Approach for Efficient Executable Path Planning for On-Road Autonomous Vehicle, Tech. Rep., SAE Technical Paper, 2019.
[169] Z. Li, L. Xiong, D. Zeng, Z. Fu, B. Leng, F. Shan, Real-time local path planning for intelligent vehicle combining tentacle algorithm and B-spline curve, IFAC-PapersOnLine 54 (10) (2021) 51–58.
[170] E. Horváth, C.R. Pozna, Clothoid-based trajectory following approach for self-driving vehicles, in: 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics, SAMI, IEEE, 2021, pp. 000251–000254.
[171] S. Shentu, Z. Gong, X.-J. Liu, Q. Liu, F. Xie, Hybrid navigation system based autonomous positioning and path planning for mobile robots, Chin. J. Mech. Eng. 35 (1) (2022) 1–13.
[172] D.S. Drake, Using Ensemble Learning Techniques to Solve the Blind Drift Calibration Problem (Ph.D. thesis), Old Dominion University, 2022.
[173] 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, 2016, arXiv preprint arXiv:1604.07316.
[174] S. Song, X. Hu, J. Yu, L. Bai, L. Chen, Learning a deep motion planning model for autonomous driving, in: 2018 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2018, pp. 1137–1142.
[175] P. Kicki, T. Gawron, P. Skrzypczyński, A self-supervised learning approach to rapid path planning for car-like vehicles maneuvering in urban environment, 2020, arXiv preprint arXiv:2003.00946.
[176] G. Moraes, A. Mozart, P. Azevedo, M. Piumbini, V.B. Cardoso, T. Oliveira-Santos, A.F. De Souza, C. Badue, Image-based real-time path generation using deep neural networks, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–8.
[177] L. Markolf, J. Eilbrecht, O. Stursberg, Trajectory planning for autonomous vehicles combining nonlinear optimal control and supervised learning, IFAC-PapersOnLine 53 (2) (2020) 15608–15614.
[178] N. Guo, C. Li, D. Wang, Y. Song, G. Liu, T. Gao, Local path planning of mobile robot based on long short-term memory neural network, Autom. Control Comput. Sci. 55 (1) (2021) 53–65.
[179] M. Sakurai, Y. Ueno, M. Kondo, Path planning and moving obstacle avoidance with neuromorphic computing, in: 2021 IEEE International Conference on Intelligence and Safety for Robotics, ISR, IEEE, 2021, pp. 209–215.
[180] D. Wang, C. Wang, Y. Wang, H. Wang, F. Pei, An autonomous driving approach based on trajectory learning using deep neural networks, Int. J. Automot. Technol. 22 (6) (2021) 1517–1528.
[181] D. Kalathil, V.K. Mandal, A. Gune, K. Talele, P. Chimurkar, M. Bansode, Self driving car using neural networks, in: 2022 International Conference on Applied Artificial Intelligence and Computing, ICAAIC, IEEE, 2022, pp. 213–217.
[182] D.-H. Lee, J.-L. Liu, End-to-end deep learning of lane detection and path prediction for real-time autonomous driving, Signal, Image Video Process. (2022) 1–7.
[183] S. Russell, P. Norvig, A modern, agent-oriented approach to introductory artificial intelligence, Acm Sigart Bull. 6 (2) (1995) 24–26.
[184] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A. Al Sallab, S. Yogamani, P. Pérez, Deep reinforcement learning for autonomous driving: A survey, IEEE Trans. Intell. Transp. Syst. (2021).
[185] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, C.-Y. Lee, Diversity-driven exploration strategy for deep reinforcement learning, Adv. Neural Inf. Process. Syst. 31 (2018).
[186] Y. Yao, Z. Peng, B. Xiao, Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city, IEEE Trans. Veh. Technol. 67 (11) (2018) 10307–10318.
[187] X.-H. Liu, D.-G. Zhang, H.-R. Yan, Y.-y. Cui, L. Chen, A new algorithm of the best path selection based on machine learning, IEEE Access 7 (2019) 126913–126928.
[188] C. Chen, J. Jiang, N. Lv, S. Li, An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge, IEEE Access 8 (2020) 99059–99069.
[189] X. Wang, X. Yu, W. Sun, A path planning and tracking control for autonomous vehicle with obstacle avoidance, in: 2020 Chinese Automation Congress, CAC, IEEE, 2020, pp. 2973–2978.
[190] P. Wang, D. Liu, J. Chen, H. Li, C.-Y. Chan, Decision making for autonomous driving via augmented adversarial inverse reinforcement learning, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 1036–1042.
[191] X. Liu, D. Zhang, T. Zhang, Y. Cui, L. Chen, S. Liu, Novel best path selection approach based on hybrid improved A* algorithm and reinforcement learning, Appl. Intell. 51 (12) (2021) 9015–9029.
[192] X. Liu, D. Zhang, T. Zhang, J. Zhang, J. Wang, A new path plan method based on hybrid algorithm of reinforcement learning and particle swarm optimization, Eng. Comput. (2021).
[193] H. Kim, W. Lee, Real-time path planning through Q-learning's exploration strategy adjustment, in: 2021 International Conference on Electronics, Information, and Communication, ICEIC, IEEE, 2021, pp. 1–3.
[194] L. Chang, L. Shan, C. Jiang, Y. Dai, Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment, Auton. Robots 45 (1) (2021) 51–76.
[195] E.S. Low, P. Ong, C.Y. Low, R. Omar, Modified Q-learning with distance metric and virtual target on path planning of mobile robot, Expert Syst. Appl. 199 (2022) 117191.
[196] P. Rousseas, C.P. Bechlioulis, K.J. Kyriakopoulos, Optimal motion planning in unknown workspaces using integral reinforcement learning, IEEE Robot. Autom. Lett. (2022).
[197] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, PMLR, 2014, pp. 387–395.
[198] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[199] J. Bernhard, R. Gieselmann, K. Esterle, A. Knol, Experience-based heuristic search: Robust motion planning with deep q-learning, in: 2018 21st International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2018, pp. 3175–3182.
[200] C. You, J. Lu, D. Filev, P. Tsiotras, Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning, Robot. Auton. Syst. 114 (2019) 1–18.
[201] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang, D. Cao, Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning, IEEE Access 8 (2020) 177804–177814.
[202] J. Zhao, T. Qu, F. Xu, A deep reinforcement learning approach for autonomous highway driving, IFAC-PapersOnLine 53 (5) (2020) 542–546.
[203] X. Liao, Y. Wang, Y. Xuan, D. Wu, AGV path planning model based on reinforcement learning, in: 2020 Chinese Automation Congress, CAC, IEEE, 2020, pp. 6722–6726.
[204] L. Chen, X. Hu, B. Tang, Y. Cheng, Conditional DQN-based motion planning with fuzzy logic for autonomous driving, IEEE Trans. Intell. Transp. Syst. (2020).
[205] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, L. Manfredi, Path planning for active SLAM based on deep reinforcement learning under unknown environments, Intell. Serv. Robot. 13 (2) (2020) 263–272.
[206] G. Li, S. Li, S. Li, Y. Qin, D. Cao, X. Qu, B. Cheng, Deep reinforcement learning enabled decision-making for autonomous driving at intersections, Automot. Innov. 3 (4) (2020) 374–385.
[207] V. Sainath, S. Reddy, et al., Deep learning for autonomous driving system, in: 2021 Second International Conference on Electronics and Sustainable Communication Systems, ICESC, IEEE, 2021, pp. 1744–1749.
[208] K.B. Naveed, Z. Qiao, J.M. Dolan, Trajectory planning for autonomous vehicles using hierarchical reinforcement learning, in: 2021 IEEE International Intelligent Transportation Systems Conference, ITSC, IEEE, 2021, pp. 601–606.
[209] J. Li, Y. Chen, X. Zhao, J. Huang, An improved DQN path planning algorithm, J. Supercomput. 78 (1) (2022) 616–639.
[210] Y. Peng, G. Tan, H. Si, J. Li, DRL-GAT-SA: Deep reinforcement learning for autonomous driving planning based on graph attention networks and simplex architecture, J. Syst. Archit. 126 (2022) 102505.
[211] Ó. Pérez-Gil, R. Barea, E. López-Guillén, L.M. Bergasa, C. Gómez-Huélamo, R. Gutiérrez, A. Díaz-Díaz, Deep reinforcement learning based control for autonomous vehicles in carla, Multimedia Tools Appl. 81 (3) (2022) 3553–3576.
[212] Z. Wang, J. Tu, C. Chen, Reinforcement learning based trajectory planning for autonomous vehicles, in: 2021 China Automation Congress, CAC, IEEE, 2021, pp. 7995–8000.
[213] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, Y. Zhao, Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles, IEEE Trans. Neural Netw. Learn. Syst. 32 (12) (2021) 5435–5444.
[214] C. Xu, W. Zhao, Q. Chen, C. Wang, An actor-critic based learning method for decision-making and planning of autonomous vehicles, Sci. China Technol. Sci. 64 (5) (2021) 984–994.

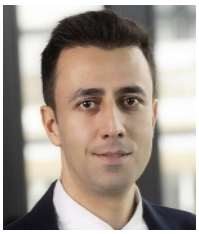
[215] J. Choi, G. Lee, C. Lee, Reinforcement learning-based dynamic obstacle avoidance and integration of path planning, Intell. Serv. Robot. 14 (5) (2021) 663–677.
[216] X. Tang, B. Huang, T. Liu, X. Lin, Highway decision-making and motion planning for autonomous driving via soft actor-critic, IEEE Trans. Veh. Technol. 71 (5) (2022) 4706–4717.
[217] M. Shehab, A.T. Khader, M.A. Al-Betar, A survey on applications and variants of the cuckoo search algorithm, Appl. Soft Comput. 61 (2017) 1041–1059.
[218] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, et al., A review: On path planning strategies for navigation of mobile robot, Def. Technol. 15 (4) (2019) 582–606.
[219] M. Reda, M. Elhosseini, A. Haikal, M. Badawy, A novel cuckoo search algorithm with adaptive discovery probability based on double Mersenne numbers, Neural Comput. Appl. 33 (23) (2021) 16377–16402.
[220] M. Reda, A.Y. Haikal, M.A. Elhosseini, M. Badawy, An innovative damped cuckoo search algorithm with a comparative study against other adaptive variants, IEEE Access 7 (2019) 119272–119293.
[221] H.J. Bremermann, et al., Optimization through evolution and recombination, Self-Organ. Syst. 93 (1962) 106.
[222] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, 1992.
[223] N.S. Utami, A. Jazidie, R.E.A. Kadier, Path planning for differential drive mobile robot to avoid static obstacles collision using modified crossover genetic algorithm, in: 2019 International Seminar on Intelligent Technology and Its Applications, ISITIA, IEEE, 2019, pp. 282–287.
[224] J.-B. Receveur, S. Victor, P. Melchior, Autonomous car decision making and trajectory tracking based on genetic algorithms and fractional potential fields, Intell. Serv. Robot. 13 (2) (2020) 315–330.
[225] R. Storn, K. Price, Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[226] H. Guo, D. Cao, H. Chen, Z. Sun, Y. Hu, Model predictive path following control for autonomous cars considering a measurable disturbance: Implementation, testing, and verification, Mech. Syst. Signal Process. 118 (2019) 41–60.
[227] P.J. Van Laarhoven, E.H. Aarts, Simulated annealing, in: Simulated Annealing: Theory and Applications, Springer, 1987, pp. 7–15.
[228] Y. Magdy, O.M. Shehata, M. AbdelAziz, M. Ghoneima, F. Tolbah, Metaheuristic optimization in path planning of autonomous vehicles under the ATOM framework, in: 2017 IEEE International Conference on Vehicular Electronics and Safety, ICVES, IEEE, 2017, pp. 32–37.
[229] J. Yin, W. Fu, A hybrid path planning algorithm based on simulated annealing particle swarm for the self-driving car, in: 2018 International Computers, Signals and Systems Conference, ICOMSSC, IEEE, 2018, pp. 696–700.
[230] Z. Shang, J. Gu, J. Wang, An improved simulated annealing algorithm for the capacitated vehicle routing problem, Comput. Integr. Manuf. Syst. 13 (2020).
[231] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.
[232] G. Li, W. Chou, Path planning for mobile robot using self-adaptive learning particle swarm optimization, Sci. China Inf. Sci. 61 (5) (2018) 1–18.
[233] J. Zhang, F. Yang, X. Weng, An evolutionary scatter search particle swarm optimization algorithm for the vehicle routing problem with time windows, IEEE Access 6 (2018) 63468–63485.
[234] L. Zhang, Y. Zhang, Y. Li, Mobile robot path planning based on improved localized particle swarm optimization, IEEE Sens. J. 21 (5) (2020) 6962–6972.
[235] M.N. Ab Wahab, C.M. Lee, M.F. Akbar, F.H. Hassan, Path planning for mobile robot navigation in unknown indoor environments using hybrid PSOFS algorithm, IEEE Access 8 (2020) 161805–161815.
[236] T. Qiuyun, S. Hongyan, G. Hengwei, W. Ping, Improved particle swarm optimization algorithm for AGV path planning, IEEE Access 9 (2021) 33522–33531.
[237] S.J. Fusic, G. Kanagaraj, K. Hariharan, S. Karthikeyan, Optimal path planning of autonomous navigation in outdoor environment via heuristic technique, Transp. Res. Interdiscipl. Perspect. 12 (2021) 100473.
[238] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, Solving constrained trajectory planning problems using biased particle swarm optimization, IEEE Trans. Aerosp. Electron. Syst. 57 (3) (2021) 1685–1701.
[239] T. Zhang, J. Xu, B. Wu, Hybrid path planning model for multiple robots considering obstacle avoidance, IEEE Access 10 (2022) 71914–71935.
[240] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[241] Y. Ge, T. Chen, X. Kong, L. Gao, Application of improved ant colony algorithm in car navigation, Control Eng. China 23 (1) (2016) 133–137.
[242] H. Ali, D. Gong, M. Wang, X. Dai, Path planning of mobile robot with improved ant colony algorithm and MDP to produce smooth trajectory in grid-based environment, Front. Neurorobotics 14 (2020) 44.
[243] C. Wu, S. Zhou, L. Xiao, Dynamic path planning based on improved ant colony algorithm in traffic congestion, IEEE Access 8 (2020) 180773–180783.
[244] M.A.R. Pohan, B.R. Trilaksono, S.P. Santosa, A.S. Rohman, Path planning algorithm using the hybridization of the rapidly-exploring random tree and ant colony systems, IEEE Access 9 (2021) 153599–153615.
[245] S. Zhang, J. Pu, Y. Si, An adaptive improved ant colony system based on population information entropy for path planning of mobile robot, IEEE Access 9 (2021) 24933–24945.
[246] C. Jiang, J. Fu, W. Liu, Research on vehicle routing planning based on adaptive ant colony and particle swarm optimization algorithm, Int. J. Intell. Transp. Syst. Res. 19 (1) (2021) 83–91.
[247] Y. Zhou, N. Huang, Airport AGV path optimization model based on ant colony algorithm to optimize Dijkstra algorithm in urban systems, Sustain. Comput.: Inf. Syst. 35 (2022) 100716.
[248] D. Karaboga, et al., An Idea Based on Honey Bee Swarm for Numerical Optimization, Tech. Rep., Technical report-tr06, Erciyes university, engineering faculty, computer ..., 2005.
[249] A. Nayyar, N.G. Nguyen, R. Kumari, S. Kumar, Robot path planning using modified artificial bee colony algorithm, in: Frontiers in Intelligent Computing: Theory and Applications, Springer, 2020, pp. 25–36.
[250] F. Xu, H. Li, C.-M. Pun, H. Hu, Y. Li, Y. Song, H. Gao, A new global best guided artificial bee colony algorithm with application in robot path planning, Appl. Soft Comput. 88 (2020) 106037.
[251] S. Kumar, A. Sikander, Optimum mobile robot path planning using improved artificial bee colony algorithm and evolutionary programming, Arab. J. Sci. Eng. 47 (3) (2022) 3519–3539.
[252] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Software 69 (2014) 46–61.
[253] J. Liu, X. Wei, H. Huang, An improved grey wolf optimization algorithm and its application in path planning, IEEE Access 9 (2021) 121944–121956.
[254] F. Kiani, A. Seyyedabbasi, R. Aliyev, M.U. Gulle, H. Basyildiz, M.A. Shah, Adapted-RRT: Novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms, Neural Comput. Appl. 33 (22) (2021) 15569–15599.
[255] J. Han, Y. Seo, Mobile robot path planning with surrounding point set and path improvement, Appl. Soft Comput. 57 (2017) 35–47.
[256] M. Zafar, J. Mohanta, A. Keshari, et al., GWO-Potential field method for mobile robot path planning and navigation control, Arab. J. Sci. Eng. 46 (8) (2021) 8087–8104.
[257] X. Meng, Y. Liu, X. Gao, H. Zhang, A new bio-inspired algorithm: Chicken swarm optimization, in: International Conference in Swarm Intelligence, Springer, 2014, pp. 86–94.
[258] X. Liang, D. Kou, L. Wen, An improved chicken swarm optimization algorithm and its application in robot path planning, IEEE Access 8 (2020) 49543–49550.
[259] B. Crawford, R. Soto, N. Berríos, F. Johnson, F. Paredes, Binary cat swarm optimization for the set covering problem, in: 2015 10th Iberian Conference on Information Systems and Technologies, CISTI, IEEE, 2015, pp. 1–4.
[260] D. Zhao, H. Yu, X. Fang, L. Tian, P. Han, A path planning method based on multi-objective cauchy mutation cat swarm optimization algorithm for navigation system of intelligent patrol car, IEEE Access 8 (2020) 151788–151803.
[261] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.
[262] J. Zhou, P. Chen, H. Liu, J. Gu, H. Zhang, H. Chen, H. Zhou, Improved path planning for mobile robot based on firefly algorithm, in: 2019 IEEE International Conference on Robotics and Biomimetics, ROBIO, IEEE, 2019, pp. 2885–2889.
[263] A.S. Bisen, V. Kaundal, Mobile robot for path planning using firefly algorithm, in: 2020 Research, Innovation, Knowledge Management and Technology Application for Business Sustainability, INBUSH, IEEE, 2020, pp. 232–235.
[264] F. Li, X. Fan, Z. Hou, A firefly algorithm with self-adaptive population size for global path planning of mobile robot, IEEE Access 8 (2020) 168951–168964.
[265] N.A.F. Abbas, Mobile robot path planning optimization based on integration of firefly algorithm and quadratic polynomial equation, in: The International Conference on Intelligent Systems & Networks, Springer, 2021, pp. 538–547.
[266] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing, NaBIC, Ieee, 2009, pp. 210–214.
[267] M. Alireza, D. Vincent, W. Tony, Experimental study of path planning problem using EMCOA for a holonomic mobile robot, J. Syst. Eng. Electron. 32 (6) (2021) 1450–1462.
[268] X. Jiang, S. Li, BAS: Beetle antennae search algorithm for optimization problems, 2017, CoRR arXiv: abs/1710.10724.
[269] J. Wang, H. Chen, BSAS: Beetle swarm antennae search algorithm for optimization problems, 2018, arXiv preprint arXiv:1807.10470.

[270] T. Wang, L. Yang, Beetle swarm optimization algorithm: Theory and application, 2018, arXiv preprint arXiv:1808.00206.
[271] X. Jiang, Z. Lin, T. He, X. Ma, S. Ma, S. Li, Optimal path finding with beetle antennae search algorithm by using ant colony optimization initialization and different searching strategies, IEEE Access 8 (2020) 15459–15471.
[272] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (3) (2011) 303–315.
[273] A.D. Sabiha, M.A. Kamel, E. Said, W.M. Hussein, Real-time path planning for autonomous vehicle based on teaching–learning-based optimization, Intell. Serv. Robot. (2022) 1–18.
[274] X. Dai, Y. Wei, Application of improved moth-flame optimization algorithm for robot path planning, IEEE Access 9 (2021) 105914–105925.
[275] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.

Mohamed Reda Mohamed received his B.Sc. and M.Sc. degrees in Computers and Control Systems Engineering from the Faculty of Engineering at Mansoura University, Egypt. He is currently pursuing his Ph.D. at the School of Engineering, University of Central Lancashire (UCLan) in the UK. Since 2016, he has been an Assistant Lecturer at the Faculty of Engineering, Mansoura University, and an Associate Lecturer at UCLan's School of Engineering since 2022. His primary research areas include artificial intelligence, meta-heuristic optimization techniques, evolutionary computation, and control engineering. Additionally, Mohamed has a strong interest in the application of AI and deep learning to embedded systems, biomedical systems, and robotics, with a particular focus on automated vehicles.

Ahmed Onsy awarded his Ph.D. from the School of Mechanical and Systems Engineering, Design Unit and Mechatronics Group, Newcastle University, UK. His main research interests are intelligent diagnostics and health management systems, intelligent maintenance systems, advanced mechatronics, and embedded systems, which can be directly applied to Intelligent Diagnostic and Health Management (DHM) and Predictive Health Monitoring (PHM) systems for oil well, wind turbine, aerospace (SHM & HUM), marine and automotive applications. He is a member of the Jost Institute for Tribotechnology. Ahmed has taken different roles at the University of Central Lancashire School of Engineering since 2014: Lecturer, Senior Lecturer, Principal Lecturer, and Academic Lead for the Mechanical and Maintenance Engineering area. Ahmed is currently the Deputy Head of the School of Engineering for Business Development and Partnerships, driving the School of Engineering UK and International Partnerships. Ahmed is a member of the University Research and Innovation Committee. He successfully placed the Mechanical Engineering Programmes in the top 15 universities in the UK in 2020 NSS results in overall student satisfaction.

Amira Y. Haikal received the B.Sc., M.Sc., and Ph.D. degrees from the Computers and Control Systems Engineering Department, Mansoura University, Egypt. She is the vice dean of environmental and community affairs and the head of the Computers and Control Systems Engineering Department, Faculty of Engineering, Mansoura University. Her major research interests include artificial intelligence fields: meta-heuristic optimization techniques, machine learning, deep learning, and smart systems engineering.



Ali Ghanbari received his Ph.D. in Mechanical Engineering from Amirkabir University of Technology in 2011. His research focuses on control of microrobots for biomedical applications. Dr Ghanbari is interested in how to make and model intelligent devices at small scale. He was a researcher in Robotics Engineering Department at Daegu Gyeongbuk Institute of Science and Technology (DGIST) and University of Leeds working on micro-/nanorobots and soft robotics. Dr Ghanbari was also a guest researcher at Multi-Scale Robotics Lab in ETH Zurich. Currently, he is a lecturer in the School of Engineering and Computing at the University of Central Lancashire.