IEEE Author Portal IEEE Access

IEEE

Society Section: IEEE Vehicular Technology Society Section

Trajectory Planning in Low-structured and Unstructured Environments for Autonomous Cars: A Systematic Review

Submission ID b97b783c-e83e-4284-9241-b0ea1172c167

Submission Version Initial Submission

PDF Generation 09 Oct 2024 13:21:37 EST by Atypon ReX

Authors

Mr. CRISTIANO SOUZA DE OLIVEIRA
Corresponding Author
Submitting Author

Affiliations
- Department of Computer Sciences, Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil

Mr. RAFAEL DE S. TOLEDO

Affiliations
- Department of Computer Sciences, Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil

Mr. VICTOR H. TULUX

Affiliations
- Department of Computer Sciences, Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil

Dr. ALDO VON WANGENHEIM

Affiliations
- National Institute for Digital Convergence (INCoD), Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil

Additional Information

Keywords

Autonomous vehicles

Computer vision

Motion planning

Trajectory optimization

Trajectory planning

Trajectory tracking

Subject Category

Computational and artificial intelligence

Intelligent transportation systems

Robotics and automation

Vehicular and wireless technologies

Files for peer review

All files submitted by the author for peer review are listed below. Files that could not be converted to PDF are indicated; reviewers are able to access them online.

Name	Type of File	Size	Page
Trajectory Planning SLR.pdf	Main Document - PDF	1.3 MB	Page 3

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Trajectory Planning in Low-structured and Unstructured Environments for Autonomous Cars: A Systematic Review

CRISTIANO SOUZA DE OLIVEIRA[1], (Member, IEEE), RAFAEL DE S. TOLEDO[2], VICTOR H. TULUX[3], ALDO VON WANGENHEIM[4]

[1]Department of Computer Sciences, Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil (e-mail: c.s.oliveira@posgrad.ufsc.br)
[3]Department of Computer Sciences, Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil (e-mail: rafael.toledo@posgrad.ufsc.br)
[3]Department of Computer Sciences, Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil (e-mail: vitor.tulux@posgrad.ufsc.br)
[4]National Institute for Digital Convergence (INCoD), Federal University of Santa Catarina, Florianopolis, SC 88040900 Brazil (e-mail: aldo.vw@ufsc.br)

Corresponding author: Cristiano Souza de Oliveira (e-mail: c.s.oliveira@posgrad.ufsc.br, cristianoo@ieee.org).

ABSTRACT Autonomous driving research, particularly on public roads, has received increasing attention in recent years. However, less attention has been paid to unstructured scenarios. Environments with poor driving conditions, such as pathways without proper signalization, broken pavement, no lane marks, and variations in shape and size, are commonly found in off-road scenarios and suburban areas in developing countries. They often present challenges for autonomous vehicles, particularly in finding a feasible path while handling unreliable data. This study provides a systematic review of the state-of-the-art trajectory planning for car-like vehicles in low- and unstructured environments. We investigated the last 10 years, presenting and classifying commonly adopted techniques and typical design decisions regarding system architecture, environment perception sensors, and data representation types. We identify common design risks owing to special terrain characteristics, sensor noise, or as the typical result of a planning approach and discuss their possible solutions presented in the selected studies.

INDEX TERMS autonomous vehicles, low-structured environments, off-road, path planning, perception, self-driving, sensors, trajectory planning, unstructured environments

I. INTRODUCTION

Autonomous driving (AD) research has received increasing attention in recent years owing to its potential to reshape transportation systems. This reveals the huge demand for robust and collision-free trajectory planning in complex and highly dynamic environments [22]. AD has been intensively studied on public roads over the past decade. In contrast, less attention has been paid to unstructured scenarios, which include off-road driveways, streets with poor maintenance, and other rough paths such as open-mining areas and unpaved roads in countryside. At the same time, most of the road network in economically underdeveloped nations can be represented by low or unstructured pathways, where semi-autonomous driving could dramatically improve goods distribution and commodity transportation logistics. Unstructured scenarios often include unknown obstacles, large curvature, and narrow passages. The goals and challenges of off-road AD can be quite different from the urban scenarios. In rough environments, driving rules such as lane-keeping are not as crucial and, depending on the situation, not possible due to variations in road shape, width, and absence of lane marks.

AD engineering can be roughly divided into a classical pipeline approach, where core functions are implemented by modules such as localization and mapping, perception, assessment, trajectory planning and decision making, vehicle control, and human-machine interface [65] and an end-to-end approach, in which the entire system, including its intermediate representations, is optimized together [10]. The ability to reliably plan trajectories in unstructured environments with rough conditions remains an obstacle [20].

Trajectory planning can be divided into global planning, local planning, and motion control. These functionalities are complementary to each other. In global planning, interactions with offline maps build a path to a user-defined goal location; however, no information regarding the environment is considered. In local planning, a subset of the global path is used as an input to build a path that allows the vehicle to traverse a region towards the final destination while dealing with local issues

such as obstacles. Motion control is responsible for guiding the vehicle through a planned local path by comparing its current state with the next intended state.

The objective of this study is to present the state-of-the-art in the field of trajectory planning for car-like autonomous navigation in unstructured or low-structured environments, showing and discussing the results of a systematic literature review (SLR) with a focus on research published in the last 10 years.

The contributions of this paper are as follows:

1) Present and classify the most commonly adopted techniques for trajectory planning in rough environments in the last 10 years using a systematic review approach.
2) Identify common design risks and discuss their possible solutions.
3) Show typical designing decisions regarding system architecture, environment perception sensors and data representation types.

This paper is structured as follows: In Section 2, we present an overview of the adopted methodology for the systematic review. Section 3 shows a typical AD system architecture. In Section 4, we discuss the state-of-the-art in trajectory planning. In Section 5, we discuss environment perception, presenting typical designing decisions regarding sensor choice and decisions regarding data representation. In Section 6, we present the identified risks and discuss their possible solutions; Section 7 concludes with highlights of the state-of-the-art possible future work.

A. RELATED WORK

[6] presents a survey on SAE level 3 or higher [30] self-driving cars since the DARPA challenges, dividing the architecture into perception and decision-making layers. Perception systems are generally subdivided into features such as localization, static obstacle mapping, moving obstacle detection and tracking, road mapping, traffic signalization detection, and recognition. Decision-making systems are commonly partitioned into route planning, path planning, behavior selection, motion planning, and control. Graph search and interpolation curve-based techniques were compared in terms of data structure size, scanned vertices, and execution time in microseconds.

[48] decompose the decision-making system into four components: route planning, behavioral layer, motion planning, and local feedback control. At the highest level, a route is planned through the road network. This is followed by a behavioral layer, which decides on a local driving task that progresses the car towards the destination and abides by the rules of the road. The motion-planning module then selects a continuous path through the environment to accomplish a local navigational task. The control system corrects errors in the execution of the planned motion. The motion planning problems adopting numerical solutions are divided into three main categories: (1) variational methods, which represent the path as a function parameterized by a finite-dimensional vector, and the optimal path is sought by optimizing over the vector parameter using nonlinear continuous optimization techniques; (2) graph-search methods, which seek a path by performing a search for a minimum-cost path in a graph; and (3) incremental search methods that sample the configuration space and incrementally build a reachability graph, maintaining a discrete set of reachable configurations and feasible transitions, and finding a solution once the graph is large enough so that at least one node is in the goal region.

[46] performed a survey particularly focused on deep learning methods. Convolutional neural networks (CNN) have emerged as one of the most common approaches for both monocular and binocular vision, object detection, lane detection, and traffic sign interpretation. Recurrent Neural Networks (RNN) and Deep Stacked Auto Encoders (DAE) are also used, along with hybrid solutions. For scene classification and understanding, which refers to judging the current traffic scene and environmental information of the vehicle, the use of a multi-resolution CNN is proposed, following the work of [63]. Other applications, particularly path planning and motion control, were also highlighted. In path planning, particle swarm optimization (PSO), genetic algorithms (GA), and methods based on deep reinforcement learning are highlighted. The work of [58] on stacking CNN and LTSM layers as a composite neural network for motion control is also presented herein. [3] presented a survey focusing on Deep Reinforcement Learning for motion planning, evaluating Deep-Q Learning Networks, Deep Neural Networks, Deep Deterministic Policy Gradients (DDPG), and other techniques, classified by observation, action, and rewarding type.

II. SYSTEMATIC ANALYSIS

In the context of this SLR, trajectory planning is understood as the act of defining a set of motions that are to be followed by a vehicle to self-move from a starting point to an ending point, respecting constraints linked to the physical characteristics of the vehicle and terrain or related to safety limitations.

We define *unstructured* and *low-structured* environments as outdoor pathways that lack partially or totally driving structural objects, such as signalization, pavement, tarmac, and lane marks, and present variations in road shape and size, such as what is commonly found in off-road scenarios. We excluded heavily cluttered outdoor environments, such as forests and crop fields, from this research.

A. METHODOLOGY

We adopted a five-step approach in this SLR, following the protocols proposed by Kitchenham [33] and Biolchini [7] for SLRs in the Software Engineering field:

1) Question: what is the state-of-the-art in trajectory planning for autonomous car-like robots in unstructured or low-structured environments?
2) Population: papers on trajectory planning or path planning for self-driving vehicles planning using at least one source of environment perception sensor.

3) Intervention: Analysis and classification of the techniques and design decisions present in the population.
4) Results: Evaluation of the published methods and design decisions.
5) Context: Papers from the following digital libraries: ACM, IEEEXplore, Science Direct, Springer link and Wiley between jan/2014 and sept/2024.

B. INCLUSION AND EXCLUSION CRITERIA

The *inclusion criteria* for this review are the following:

- Papers written in English the areas of Computer Science, Engineering or Robotics;
- Papers on self driving path planning, motion planning or trajectory planning in low or unstructured environments, on vehicles that are self-guided and perform environment perception using at least one source of input sensor.

The following *exclusion criteria* were employed to eliminate a candidate study from this review

- Short papers such as abstracts or expanded abstracts;
- Works on indoor environments, heavily cluttered environments, and other environments not typically drivable by humans, such as forests, harvesting fields, or microscopic environments, such as in medical applications;
- Papers unrelated to car-like vehicle trajectory planning, such as non-wheeled robots, bipedal or quadruped, aerial or aquatic vehicles and motion planning of robot arms or other robot parts not related to driving or collaborative planning for multi-agent or multi-vehicle scenarios;
- Papers that do not present any test case, neither in a Simulator nor on Unmanned Vehicles or that do not report the reaching of a feasible path as their solution.

C. SEARCH EXECUTION

We employed a three-step approach to select papers. Initially, a search was performed with the query strings defined in Appendix A (see Table 6), which resulted in 2381 candidate papers. As a second step, we filtered the initial data, excluding papers that did not have at least one of the following words in their title or abstract: "unstructured, low-structured, semi-structured, off-road, off-road, non-urban, and nonurban". This result was then filtered, eliminating repetitions in which the same paper was published in more than one source. This was done by comparing titles and DOI. Any paper that presented one of the following keywords in its title or abstract was excluded:

- Underwater, UAV, indoor, SUAS, aerial, Humanoid, cellular networks, femtocell, robot arm, multi-arm, 6-DOF, human environment, UVMS, climbing robot, CLIBO, snake-like, homotopy, multi-material, construction, Editorial:, urban, exoskeleton, multi-robot, surgical, robotic construction, History of, healthcare, Industry 4.0, excavator, Logistics, jumping robots, Riverine, submerged, social, networks, unstructured meshes, governance, agile, scrum

Every paper was manually filtered by title, abstract, keywords, images, introduction, and conclusion, following the inclusion and exclusion criteria, resulting in the selection of *27 primary sources* for this review.

III. SYSTEM ARCHITECTURE

Autonomous Driving Systems (ADSs) are designed as either standalone or connected multi-agent systems. Furthermore, each autonomous unit can be realized using two alternative approaches: modular and end-to-end driving [66]. In this review, the most commonly adopted approach is modular architecture, which divides the system into two major layers: perception and decision-making (see figure 1).

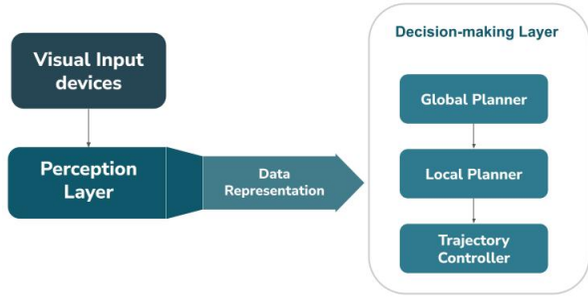


FIGURE 1. The most commonly adopted architecture.

A. PERCEPTION LAYER

The Perception Layer is responsible for perception of the surroundings, such as computer vision and sensor data gathering. Visual perception may come from different types of image-capturing devices, such as monocular cameras, which typically produce RGB frames at a given frame rate; stereo cameras, which can combine two sensors to estimate depth, thereby producing RGBD frames; light detection and ranging sensors (LiDAR), which actively emit pulsed light waves that bounce off surrounding objects and return to the sensor; and infrared sensors, which work similarly to LiDARs and are made of pyroelectric materials that produce waves in the infrared spectrum.

Computer vision plays a significant role in environmental perception by interpreting raw input data from cameras or LiDAR as meaningful data that can be used to decide which actions a vehicle should take next. Different techniques are used in self-driving car design, including object detection, object tracking, semantic segmentation, multiview 3D reconstruction, and stereo reconstruction [31].

B. DATA REPRESENTATION

Perception data, as a result of semantic segmentation or object detection, must be stored in such a way that it can be used as a search space for local planning. We define this data structure as a data representation, which is further discussed in V-B

There are several types of data representations in the literature. The most commonly used in low-structured environ-

ments are: Occupancy Grid, Traversability Map, Elevation Map, TreeMap, Cost-Map, State Lattice, and Mesh.

C. DECISION-MAKING LAYER

The decision-making layer is responsible for performing trajectory planning. It receives a goal pose from a human operator and produces a series of control commands to the vehicle's actuators, such as steering angle definition, acceleration, and brake activation, which aim to take the vehicle from a starting pose to the chosen goal pose. It also performs other decision tasks such as behavior selection, responsible for choosing the driving behavior such as lane keeping, intersection handling, traffic light handling, etc.; and traffic signalization detector, responsible for interpreting driving rule information such as allowed maximum speed [6]. Those functions can vary depending on the type of environment that the AD is designed to operate in.

IV. TRAJECTORY PLANNING

We now focus on trajectory planning, which is a central function of AD systems. We define it as the function responsible for receiving a destination from the user and producing a series of control outputs that ultimately bring the car to the given destination using exteroception and proprioception sensor data. We subdivided this into three sub-functionalities: global planning, local planning, and motion control.

A. GLOBAL PLANNING

Global planning is the function responsible for receiving a global destination and then building a list of sparse vehicle goal reference locations that represent points of interest or milestones to be achieved to reach the destination. This goal list can become invalid if the subsequent planning activities are unable to reach their reference points for some reason, such as a roadblock.

A Global Planner (GP) is responsible for interacting with offline maps to plan a path for the long run, which means that the path accounts mostly for a distance outside of the local perception range, so obstacles will most likely be unknown at this time.

Implementations of GP are found in [40], which computes intermediate goals (waypoints) from a global topological map, graph search based D* lite [45] in [8] and AD* [42] is used in [34]; [47] adopts sampling-based RRT. [55] combines satellite-based GP with local mapping to help future local planning by referencing the local map with the global coordinates.

B. LOCAL PLANNING

Local planning is the function responsible for receiving the list of global locations representing the milestones and navigating the list by building several lists of dense reference locations for each pair of milestones that have not yet been navigated, accounting for local issues such as obstacles and nonholonomic constraints. This dense list is often called "ideal path" [2], because it represents the path that the vehicle should follow. However, owing to the actual vehicle dynamics, inaccurate sensor data, or control errors, the actual trajectory can drift from the planned path.

We classified the techniques selected in this review into graph search, dynamic window approach, multi-path generation and selection, incremental search, optimization-based, and machine learning.

1) Graph Search

Graph search in trajectory planning is a technique that consists of traversing a graph or a data representation that is seen as a graph to find a path from a starting node to one or more goal nodes. A path is optimal when the total cost of reaching a goal node is minimal. Depending on the cost definition, suboptimal paths can result in better driving conditions or a better trade-off between the required execution time and optimality. In this review, it is the most common approach adopted by 42% of the analyzed papers.

In recent years, Hybrid A* [12] has been increasingly used, particularly when associated with optimization approaches. In the last five years, 70% of graph search local planners have adopted it. [23] uses Hybrid A* as the initial guess, then optimizes the path with all constraints removed, analytically deriving the gradients, and finally solving the transformed unconstrained optimization via the quasi-Newton method. Static collision avoidance is performed by combining a collision-free path with environmental information to generate a free convex polygon. Dynamic collision-free driving is guaranteed by ensuring that the minimum distance between the ego vehicle and obstacle convex polygons at each moment of the trajectory is greater than a collision-free threshold. [1] combines area optimization and Hybrid A* to optimize a recorded non-optimal path by dividing the optimization into smaller sub-problems and then applying Hybrid A* to solve them, as a divide-and-conquer approach. [52] also uses Hybrid A* to initially generate a rough path, which is used to build the optimal path by optimizing the time-energy cost function. It achieves collision avoidance by first representing the vehicle using two circles with the same radius and dilating obstacles to build a quad-tree map, allowing the assignment of a maximum rectangular box that does not intersect any obstacles for all trajectory points. The local convex space is constructed using an improved version of the CFS-handling quad-trees.

[43] fuses a 2D grid and 2.5D elevation map to create a hybrid map accounting for overhanging structures and uses a variation of Hybrid A*, which adds heading angle and traversability to each node. It achieves collision avoidance by first representing the vee using two circwith the same radius and dilating obstacles to build a quad-tree map, allowing the assignment of a maximum rectangular box that does not intersect any obstacles for all trajectory points. The local convex space is constructed using an improved version of the CFS-handling quad-trees.

[43] fuses a 2D grid and 2.5D elevation map to create a hybrid map accounting for overhanging structures and uses a variation of Hybrid A*, which adds heading angle and traversability to each node. The traversability is calculated from the terrain roughness, real pitch, and roll angle, which helps identify more suitable paths. The integration between data representation and path planning works at node expansion, where cells located on the 2D map are directly deleted because their corresponding static traversability is zero, whereas those on the 2.5D map are queried.

[60] uses a Hybrid A* version which combined with a global planner developed using Lanenet2. The search policy is replaced to make it harder to decide to go backward, as it is also not a common decision for a human driver. In [50], some extensions to the planner were made to plan for the next two goal waypoints instead of one, avoiding dead ends among other features. [53] uses a two-step approach, where a partial motion planner (PMP), similar to [12], uses motion primitives based on the kinodynamic car model to generate child nodes, guided by Dubins curves as first heuristics. Multilayer optimization based on discrete points, spiral curves, and spline curves is used to smooth the resulting path. The search policy is replaced to make it harder to decide to go backwarpredicteds also not a common decision for a human driver. In [50], some extensions to the planner were made to plan for the next two goal waypoints instead of one, avoiding dead ends among other features. [53] uses a two-step approach, where a partial motion planner (PMP), similar to [12], uses motion primitives based on the kinodynamic car model to generate child nodes, guided by Dubins curves as first heuristics. Multilayer optimization based on discrete points, spiral curves, and spline curves is used to smooth the resulting path. It performs collision checking for dynamic objects by transforming each predicted trajectory into several line segments and computing the intersection of line segments.

Less recent graph search local planners have adopted variations of classical A* [24] such as [61], where it is combined with convolutional neural networks for energy cost estimation from a point cloud input. In [19], adaptive A* is used to improve speed when searching for a feasible path, and a root-mean-square is used for cost evaluation. In [39], the authors suggested an approach based on a deep-first forward search (DF-FS) combined with a graph search algorithm such as A* or ARA*, where the first one is used when the vehicle is not in a certain range of waypoints with good positioning. In [13], a modified A* admits many leaf goal points. The least-cost path is returned once a timeout occurs; therefore, there is no guarantee of an optimal result, but the processing time is clamped.

[40] uses an incremental variation of A*, where a multi-feature LP that uses Determined Finite Automata (DFA) performs several tasks based on the vehicle state, and local planning is performed by Field D* [14] when in a corner or intersection or by generating

parabolic functions or detecting the centerline under normal driving conditions. Similarly, [44] performs path planning using a sample-based algorithm called Rapid Semi-Optimal Motion Planning (RASMO). It expands search trees according to the physical model under the control input, where each node in the tree represents the state of the robot model, including the pose and velocity. The optimal path is obtained by finding the tree that has the nearest node to the target state. [67] modifies A* to use a set of 14 elementary energy-aware motion primitives for each start state with map resolution 0.1 m and heading angle resolution $\pi/8$. The cost function then selects the motion primitives that lead to a lower total energy. Testing shows that for a small increase in the total length, the energy cost for moving is reduced by nearly 26.9%.

Local planning based on Dijkstra's method is much less common, but is also present. [4] builds a three-layer traversability detection using multi-layer perceptrons, decision trees, and random forests, each outputing a traversable path.Then, Dijkstra is used to find an optimal path from the given result.

[37] adopts a hybrid solution based on combining deep learning, graph search and optimization. A ResNet network is initially trained to find the priority search area using a dataset built with the output from A*. This search area and a 3 channel frame representing the state, starting point, goal point, edges, obstacles, and ROI are used as inputs for a Monte Carlo Tree Search (MCTS). MCTS attempts to generate a coarse path. If it fails, A* is used instead. A cubic spline is used to smooth the path and serves as the input for the path optimization model, which uses sequential quartic optimization to reduce the path cost.

2) Dynamic Window Approach (DWA)

It is a technique that reduces the search space to the dynamic window, which consists of velocities reachable within a short time interval and also allows the vehicle to stop safely [15]. It is well suited for proximity sensors such as ultrasonic transducers. For obstacle detection using cameras and infrared sensors, the resulting proximity estimate is accurate only if the obstacle touches the floor. Obstacles at different heights lead to an overestimation of distance, which may cause the robot to collide. [34] uses DWA for local planning combined with AD* for global planning. In [69], a deep neural network (DNN) fuse data of semantic segmentation from stereo cameras and LiDAR to build a traversability map, and the local trajectory is planned using DWA.

3) Multi-path generation and selection

This approach is based on the generation of several parallel paths from a reference path or waypoint-following motion constraints, which are then subjected to a selection process to choose the best path. [68] generates a series of lines with equal intervals on a map and then fuses lanes and obstacles that are all parallel to a line generated by the geographic information system (GIS) in order to find the optimal path. [64] follows a reference path by first using a general regression neural

network (GRNN) to find the centerline, which is checked for reliability. If the centerline is not reliable, the LP is switched to sampling a set of terminal states in the control space. A cubic Bezier curve is used to generate the trajectories for each terminal state. An obstacle cost function evaluates the path candidates, from which the optimal path with the lowest cost can be extracted.

4) Sampling-based search

A sampling search consists of techniques that use random sampling in the configuration space to build a reachability graph. Once this graph is sufficiently large to touch the goal point or goal region, a feasible path can be obtained. The most commonly used sampling-based technique is RRT and its derivations. [59] Modifies RRT* to account for safety first by using Potential Field to prevent RRT from generating paths too close to obstacles, which is critical in off-road scenarios. [56] proposes a hierarchical planner capable of anytime planning, which uses a simplified vehicle model and a variant of RRT* that works by sampling a pose in the continuum within some radius and yaw tolerance of a random waypoint. When a feasible path is found, the rest of the free time is spent to optimize it.

5) Optimization-based approach

The optimal-based approach involves a technique that obtains optimal paths by performing optimizations over a function representation of a path. It is often associated with other approaches such as graph search, which serves as a coarse input path. [41] suggests using model predictive control (MPC) to achieve more complex and predictable behavior. As it relies heavily on an accurate representation of the system dynamics model, which is often not possible, a learning-based Gaussian Process version (GP-MPC) was adopted. [5] proposes a planning solution based on information theory, where the mutual information (MI) and motion costs are computed over the values from a probability density function (PDF), allowing integrating the measurements into a probabilistic map representation using Bayesian updates. The planning builds a control sequence for the vehicle model that maximizes the ratio of mutual information between the map and future sensor observations and the motion cost of the planned robot trajectory. [21] treats the trajectory planning as an optimal control problem (OCP) and transforms it into a nonlinear programming (NLP) problem. As the OCP is highly nonconvex and nonlinear, it is dependent on a proper initial guess. An initialization strategy decoupling the original scheme into two subtasks is proposed, reducing the equation violation by mainly optimizing vehicle positions and orientations and trying to find a solution that strictly satisfies all inequalities by optimizing only the total travel time. [26] uses a cost-valley approach where, waypoints given by a global planner are optimized by a line simplification algorithm Douglas-Peucker (DP) and by comparing the difference in the first-order derivative of the current path element and of the consecutive path elements. The path is chosen by following a cost-based version similar to a potential-field approach using Breshenham's algorithm.

6) Machine Learning (ML)

In the classical pipeline, each model serves a standalone component and corresponds to a specific task, which facilitates interpretability and debugging, but also allows error propagation between each module, which could result in information loss. In contrast, end-to-end autonomous systems combine prediction and planning into a single model that can be jointly trained such that the entire system is optimized towards the ultimate task [10]. [28] proposes a training-based approach using convolutional neural networks to visually predict the future path that a vehicle will take as an end-to-end solution. [8] proposes to use Support Vector Machine (SVM) based on the intuition that the safest trajectory for cruising among obstacles is the one that preserves the maximum margin from all of them. Thus, the obstacles are clustered such that the minimum distance between their groups is greater than the vehicle embodiment.

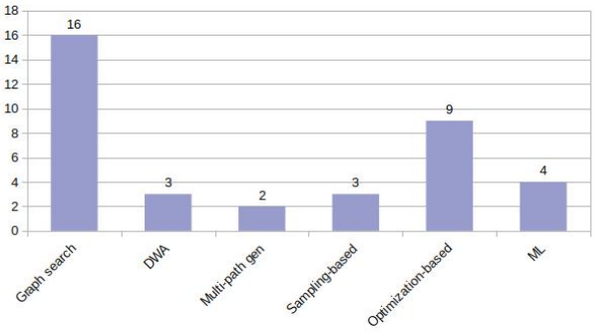


FIGURE 2. Local planning by frequency of adoption.

C. MOTION CONTROL (MC)

When the local planner finishes producing the next local path, the self-driving car is expected to follow the output perfectly. Because this is often impossible due to imprecision in determining its own location or wheel slip, the actual position with respect to the planned path may differ. The motion controller is therefore a function responsible for continuously correcting the car's course to follow the planned path as best as possible by comparing the estimated current EGO position with the ideal path. The typical implementations include combinations of model prediction control [17], Stanley, Pure Pursuit, and PID controllers. [6]. [1] adopts MPC, modeling lateral dynamics by applying a time-state control form (T-SCF) transformation to describe it as a system of linear, differential equations, and approximates the longitudinal dynamics with a double integrator model, whose states are the traveled distance and the vehicle speed, as in [23]. In [47] the planned path is converted into a set of line segments representing the input for moving a virtual robot, which is treated as a path reference. MC compares the real state of the vehicle with

this reference to issue commands and collects new states, resulting in a feedback loop. Final movement is controlled by a model-reference adaptive controller (MRAC) and a proportional controller. The error between the ideal virtual path and the real path is measured and minimized as much as possible. In [68], a reference path is also followed. A speed controller defines the velocity based on the perceived distance from the front obstacle; therefore, the controller works with both a reference path from the path planner and a reference speed from the speed planner. [2] uses a Distributed Control System (DCS) to perform reference path correction with precision in navigation. Several different control algorithms run in parallel, and their outputs are compared. The one with the best concordance rate of sensor readings and the ideal path database is selected as a high priority to control the vehicle.

V. ENVIRONMENT PERCEPTION

Environmental perception is performed by the perception layer, which is responsible for the perception of the surroundings, such as computer vision and sensor data gathering. Visual perception may originate from different types of image-capturing devices, such as monocular cameras, stereo cameras, light detection and ranging sensors (LiDAR), and infrared sensors.

Computer vision (CV) plays a significant role in environmental perception by interpreting raw input data from cameras or LiDAR as meaningful data that can be used to decide which actions the vehicle should take next. Different techniques are used in self-driving car design, including object detection, object tracking, semantic segmentation, multiview 3D reconstruction, and stereo reconstruction [31].

A. INPUT SENSORS

The input sensor choice plays a significant role in the design of autonomous driving systems. Given their affordability, availability, and widespread research, cameras have been widely used. However, they have limitations, such as problems with excess or lack of illumination, because detection occurs in the projected image space and the scale of the scene is unknown. 3D LiDAR offers an alternative by solving the scale problem while being less susceptible to intemperate weather at the expense of distortions bound by distance [65]. In this review, the most commonly adopted input sensors were monocular cameras, stereo cameras, LiDAR, and infrared sensors.

1) Monocular cameras

Monocular cameras can provide a relatively inexpensive passive solution for computer vision. Some studies used them as a single source of input [47, 39, 56, 13, 19] whereas [1, 41] used it in combination with LiDAR. The most commonly adopted data representation for monocular cameras is the occupancy grid (Table 1).

2) Stereo cameras

Stereo cameras employ two or more cameras to estimate depth. Most studies with stereo cameras adopted the Cost Map as a data representation [13, 50, 62]. A stereo camera is also used in [28], but with occupancy grid.

3) LiDAR

Light detection and ranging (LiDAR) is an active sensor that typically emits pulsed light waves that bounce off surrounding objects and return to the sensor. By computing the time taken to return to the sensor, it is possible to estimate the travel distance, map the surroundings, and build a point cloud for object detection and depth calculation. LiDAR-only is the most commonly adopted choice, and the occupancy grid is the most frequently used [37, 53, 43, 60, 8]; however, it is also used with traversability and elevation maps [64].

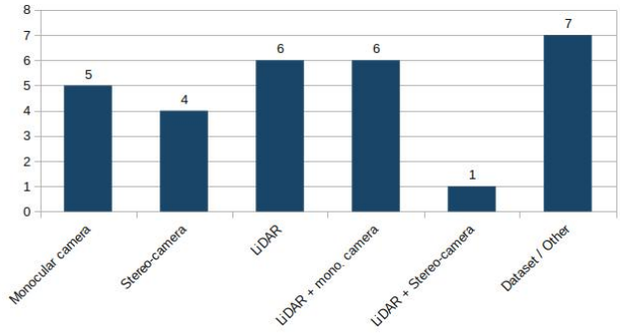


FIGURE 3. Input sensors by frequency of adoption.

B. DATA REPRESENTATION (DR)

The data obtained from visual input sources represent the main input for most local planning techniques. Different types of data structures are adopted, some of which have the simple role of holding sensor data such as RGB frames or point cloud raw output, while others carry pre-computed planning information such as vector directions, traversability, and cost mapping. Here, we discuss the most common findings in this review. Table 1 classifies the data representation according to the corresponding type of visual input sensor found in this study.

1) Occupancy Grid

Occupancy Grid (OG) is a tessellated 2D grid that stores information regarding which areas of a robot's operating environment are occupied and which are empty on each cell by setting a certainty factor relating to the confidence that the particular cell is occupied. Its main advantage is its simplicity, particularly in terms of memory management [11]. This was present in 71% of the studies included in this review. [39, 13] combine it with graph search local planning, using monocular cameras as input. [47] also combines it with a monocular camera but adopts RRT instead. In [40, 55], a local graph search was performed using OG data obtained from stereo

TABLE 1. Classification of data representation by visual input type

	Monocular camera	Stereo camera	LiDAR	LiDAR and camera	LiDAR and Stereo-camera	Other
Occupancy Grid	[47, 39, 13]	[28]	[37, 53, 64, 8]	[1, 67, 41]	-	[26]
Traversability Map	-	-	[43]	-	[69]	
Elevation Map	-	-	[43]	-	[64]	[4]
CostMap	-	[40, 44, 34]	-	[68]	-	
TreeMap	-	[34]	-	[5]	-	[52]
State Lattice	[56]	-	-	-	-	[61]
Mesh	[19]	-		-	-	
Other	-	-	[60]	-	-	[22, 59, 21]

cameras. LiDAR and OG were used to perform a graph search on global planning in [8] and an incremental search in [64]. An optimization-based approach was adopted for OG data obtained from a combination of LiDAR and camera inputs in [41].[28] implements an end-to-end machine learning solution using OG as the base input. [67] fuses data from a monocular camera and LiDAR but uses a 2D OG map, not measuring terrain elevation.

2) Traversability Map

Traversability Maps hold information about a vehicle's traversability, which can be defined as the capability to reside over a terrain region under an admissible state wherein it is capable of entering given its current state, taking into account the terrain model and vehicle models, the kinematic constraints, and a set of optimality criteria. Geometry-based methods comprise the majority of the methodologies for analyzing traversability [49]. [69] fuses LiDAR and stereo-camera images, using a traversability map combined with an elevation map and using a dynamic window approach. [64] uses a traversability map to estimate the optimal path with a hybrid behavior selection approach that combines deep-learning and offset curve generation.

3) Elevation Map (EM)

Elevation Maps are 3-D representations of rugged terrain, often implemented as grid-based representations, where the position of a point in the cartesian coordinate system can be derived from the measured range, obtained using passive or active sensing, and the direction of the bean at that point. Stereo cameras are typically used in passive sensing. In active sensing, a direct measure is taken, suffering less interference from outside illumination; therefore, it is preferred under this condition. Once the range image is obtained, the values θ and ϕ of the horizontal and vertical angles are computed using the step between two consecutive values, and the x-,y-, and z-coordinates are calculated using trigonometry [35]. [53] uses EM for path optimization to account for the terrain altitude as a cost parameter. [41] adopts a GP-MPC and works the planning on EM data as an optimization problem.

4) TreeMap

TreeMaps are frameworks for 3-D representation that seek to perform probabilistic representation, model unmapped areas, and maintain memory and computing efficiencies [29]. They are based on QuadTrees, Octrees, or similar data structures, where the initialization of map volumes is often delayed until measurements need to be integrated; thus, the map only contains volumes that have been measured. The probability of a leaf node n being occupied given a measurement is estimated by an equation that takes a prior probability, the previous estimated value, and the current measurement as inputs. As measurements are taken, the probability changes towards a greater or lesser chance of being occupied. When the threshold is reached, it is assumed to be either occupied or free. [34] uses octrees with a prune technique to reduce memory consumption for its graph search in stereo camera input. [9] performs optimization-based planning using information theory and fuses a LiDAR and a monocular camera with an octree. [37] combines a quad-tree map with a convex feasible set method to construct a local convex feasible space efficiently.

5) Cost-Map

A Cost-Map is a grid where each cell contains one or many values that represent the average cost of traversing an area [36]. The total average cost of a path can be obtained by summing the values for each cell. Usually, the cost values vary from 0 for a fully traversable area to ∞ for an obstacle or nontraversable area. Any value between them corresponds to a certain degree of traversability [32]. [40] uses it with stereo-cameras and a graph search approach, [44] uses it with incremental search. [34] uses it with a dynamic window approach. [68] uses it to fuse LiDAR and monocular camera data.

6) State Lattice

A State Lattice is a search space built to satisfy motion constraints [51], representing a set of all reachable configurations, built by discretizing the C-space into a hyperdimensional grid, where for every node a feasible connection is attempted, resulting in a set of all possible feasible paths. This can be viewed as a generalization of the grid. [56] uses

TABLE 2. Classification of local planning techniques by data representation type

	Graph search	DWA	Multi-path	Optimization-based search	Sampling-based	Machine Learning
Occupancy Grid	[1, 37, 67, 53, 39, 13]	-	[64]	[1, 37, 53, 39, 13]	[47]	[37, 28, 8]
Traversability Map	[43]	[69]	-	-	-	-
Elevation Map	[43, 4]	[69]	-	-	-	[4]
CostMap	[40, 44]	[34]	[68]	-	-	
TreeMap	[52]	[34]	-	[52, 5]	-	-
State Lattice	[61]	-	-	-	[56]	-
Mesh	[19]	-	-	-	-	-
Other	[22, 59, 53]	[2]	-	[22, 21]	[59]	-

Obs.: A local planner may implement more than one data representation type.

it with a monocular camera and a sampling-based approach with RRT as a local planner. [61], uses it with graph search (A*).

7) Mesh

Mesh is a very common technique used in computer graphics to represent 3-D data in a series of shapes. In particular, triangular meshes are composed of irregular triangles and can be used as a simplification for a 3-D raw point cloud. Raw data can be converted into meshes by applying functions, such as 2D Delaunay. In this case, the memory required to hold the data is less than the initial raw data [18]. Compression techniques [38] can also be applied to reduce memory consumption. [19] used it for camera frames and adopted adaptive A* as the local planning approach.

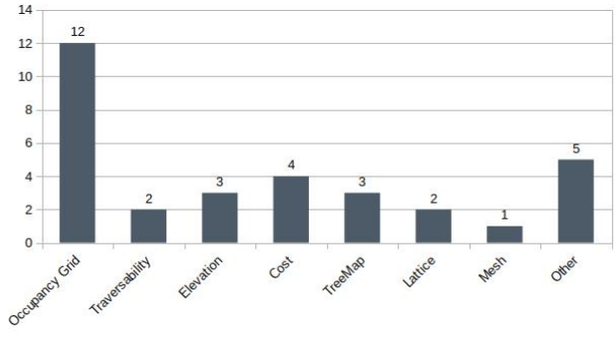


FIGURE 4. Data representation by frequency of adoption.

VI. IDENTIFIED RISKS AND SOLUTIONS

Planning and correctly following a local trajectory in rough scenarios often presents many problems owing to terrain variations or sensor faults. These problems can vary according to the design decisions, such as the local planner technique, sensor type, or special terrain characteristics. In this section, we highlight some interesting risks and problems, along with their proposed solutions, with the aim of presenting practical ways of dealing with them.

A. THE PLANNER CHOOSES A VALID BUT LESS-TRAVERSABLE TRAJECTORY.

When a local planner uses cost metrics that do not consider sufficient information about the terrain and its shape, such as path planning based solely on length, it can produce less traversable trajectories. These trajectories, despite being valid in terms of collision avoidance, minimal distances, and non-holonomic constraints, can cross unwanted terrains, such as those with too high or too low elevations or too slippery. Therefore, the local planner must collect other information regarding free space and take traversability into account when optimizing the final path.

Path smoothness is an important measure of driving comfort. It can be enhanced during the planning phase by using interpolations, such as splines in [25] and Bézier curves in [64]. Similarly, [1] improved trajectory smoothness by modifying the cost function of Hybrid A* by adjusting the last two terms to be proportional to steering control. Post-processing optimization can also be applied to improve the smoothness of a rough initial path from the local planner. [60] uses Catmull-Rom interpolation on the output of a hybrid A* local planner. Similarly, [22] and [52] generate an initial rough path using hybrid A*, followed by path optimization to enhance traversability. [53] uses an approach similar to hybrid A*, whereas [37] adopts a pre-trained deep neural network to provide an initial guess for the path optimization step.

In terms of data representation, [9] aimed to address segmentation uncertainty by deriving a closed-form, efficiently computable lower bound for the Shannon mutual information between a multiclass occupancy map and a set of range-category measurements. This allows for a better reevaluation of paths and balancing uncertainty reduction with efficient exploration. [34] uses OctoMap, where each cell indicates the probability of being occupied or free. A 50% probability signifies an unknown state, whereas a probability of 12% or lower indicates a known free space, and 97% or higher denotes a known occupied space. This approach prevents sudden path changes owing to faulty image segmentation, as subsequent measurements are unlikely to drastically alter these values.

B. THE FINAL MOTION PATH IS DIFFERENT FROM THE PLANNED PATH

This can occur because of loose terrain, sensor faults, overspeed, and other instabilities. Another concern is that the majority of studies use nonholonomic constraints based on kinematics, which assume no wheel slip when modeling the vehicle's motion. However, wheel slip is common in unstructured environments. [64] and [68] use reference path correction, while [47] uses a model reference adaptive controller (MRAC) [57] which aims at updating the parameters of the control law such that the behavior of the resulting closed-loop system becomes as close as possible to that of a given reference model. [1] and [22] use model prediction control, which utilizes a model of the vehicle to find the numerical solution to a constrained optimal control problem over a finite time horizon [54].

C. UNRELIABILITY OF VISUAL DATA

Visual input data is a crucial component of the local planning decision-making process. However, depending on the source of the visual input, issues such as incorrect image segmentation, sensor failures, and inaccurate distance estimations can occur unexpectedly. Probability density measurements can be used to flag certain regions containing obstacles to prevent a single faulty image frame from compromising a vehicle's perception of the environment. [41] builds an occupancy grid with a two-layer risk map: the static risk layer for terrain costs and the dynamic risk layer for uncertainties like moving objects. [4] uses a traversability map to calculate the cost of movement and incorporates machine learning to learn from past driving experiences.

Stereo cameras can make autonomous driving systems prone to errors under strong lighting conditions. To address this issue, [44] employed a semi-global matching 3D terrain representation [27] that uses a pixel-wise matching cost based on mutual information to compensate for radiometric differences between input images.

D. THE LOCAL PLANNING IS NOT FIT FOR A PARTICULAR SCENARIO

AD driving in unstructured environments needs to adapt to different terrain environments, such as country roads, woodland roads, slope roads, and wading roads; therefore, the behavior of the vehicle might need to change to adapt to different driving scenarios. [40] proposes a deterministic finite automaton to change the planning behavior based on the surroundings, such as detection of uphill or downhill, detection of road or slope road. Each mode has a special planner to adapt to the particular characteristics of the terrain. In [2] a distributed control system performs the path tracking decision using several parallel modules that follow a hierarchy of priority based on the concordance rate of the sensor readings and the database. The module with the highest concordance rate is selected to control the navigation over the next stretch of the planned path.

E. DRIVING AT HIGHER SPEEDS REQUIRES A LONGER LINE OF SIGHT AHEAD OF THE VEHICLE.

When driving at higher speeds, the length of the line of sight ahead must be sufficient to allow any newly detected obstacle to be within the minimal required distance to stop the car safely. However, simply extending it reduces planning efficiency, which ultimately affects the car's overall response time. [68] solves this problem by only extending further the line of sight ahead for object detection. This allows the vehicle to lower its speed, ensuring that the local obstacle avoidance plan will have enough time to execute properly. This allowed their cars to drive up to 60 km/h and could theoretically support a linear velocity of up to 26.853 km/h.

VII. DISCUSSION AND CONCLUSION

This review examines state-of-the-art trajectory planning for car-like vehicles in low-structured and unstructured environments over the last 10 years using a systematic methodology. Unstructured scenarios often include unknown obstacles, large curvatures and narrow passages. The goals and challenges of off-road driving can be quite different from those of urban scenarios. For example, driving rules such as lane-keeping are not as crucial and, depending on the situation, are not possible owing to variations in road shape, width, and absence of lane marks.

In general, trajectory planning relies heavily on several system functions such as environment perception, state estimation, and motion control. Some interesting risks and problems regarding trajectory planning and its supporting functionalities that repeat the studies in this review are discussed in Section VI. These problems should be addressed properly because they can affect the performance and overall safety of driving tasks.

Visual environment perception is a challenge faced by AD systems. It is performed by capturing image frames with devices such as monocular and stereo cameras, producing point cloud data with LiDAR, and using other raw data sensors such as infrared and radar. Computer vision plays a significant role in perception by interpreting raw input as meaningful data that can be used to decide which actions the vehicle should take next. LiDAR is the most commonly used input sensor. It is adopted standalone and in combination with cameras in the majority of studies, but no clear preference for a specific input set was identified, as shown in figure 3. Perception data must be stored in such a way that it can be used as a search space for local planning Data representation techniques vary, with each addressing a particular set of functionalities. As shown in figure 4, there is a clear preference for using an occupancy grid. It is a versatile data structure that is used with all the local planning methodologies in this review, as shown in Table 2.

Trajectory planning is responsible for receiving a destination from the user and producing a series of control outputs that ultimately bring the car to a given destination. We subdivided this into three sub-functions: global planning, local planning, and motion control. Local planning was the main focus of the papers selected for this review. A global

planner was presented in seven studies [41, 47, 26, 56, 40, 8, 34]. However, there is no clear preference for a particular global planner algorithm. Similarly, the motion controller was presented in only seven studies with no clear preference for a particular technique [1, 22, 47, 19, 64, 69, 68].

We classify the local planning techniques by the following categories: Graph search, Dynamic Window Approach, Multi-path generation and selection, Sampling-based search Optimization-based and Machine Learning methods, as shown in Table 2. In figure 2, we see a clear preference for graph search techniques, representing 41% of adoption. It is important to note that a study can use more than one type of local planning approach, such as graph search and optimization in [1, 37, 53], graph search, and machine learning [4, 37].

Comparing local planning performance is a difficult task because external factors such as the type of hardware and set of implemented features, types of data representation, and offline pre-computing tasks may interfere with metrics such as the total execution time, which can be misleading. Therefore, we focused on presenting the improvements and weaknesses identified in the selection of studies. See Tables 4 and 5.

When examining the studies chronologically, we see an increase in the use of optimization-based methods (see Table 3). Over the last five years, these methods as standalone solutions or in combination with graph search represent 70% of the studies in this review, which is a great increase from 2014 to 2019. They are gaining attention owing to their continuously increasing embedded computing power and wide choice of numerical optimization tools, offering the possibility of explicitly capturing the vehicle model in the form of kinematic, dynamic, and actuator constraints [1]. However, guaranteeing real-time performance and convergence is a significant challenge when dealing with nonconvex collision constraints. On the other hand, methods such as graph search or sampling-based search can guarantee convergence and perform well in real time; however, they rely on finite-dimensional state spaces, often ignoring high-order information such as velocity and acceleration, such that using a constant speed when planning is common. [52]. Therefore, we observed an increase in combining optimization-based techniques with heuristics-based techniques, such as graph search, as in [1, 22, 52].

Because this review focuses on trajectory planning techniques in drivable-by-human unstructured environments, we have excluded other important scenarios such as harvesting fields and heavy, cluttered spaces such as forests. Future studies should cover other scenarios, in which transportation automation may also be an economically relevant factor.

VIII. DECLARATIONS

A. COMPLIANCE WITH ETHICAL STANDARDS

This manuscript is original and was not submitted or published elsewhere in any form or language. The study was conducted impartially without any financial, personal, or professional relationships that could have biased the findings. The authors also affirm no conflicts of interest that could have influenced the publication, interpretation of results, or conclusions of this research.

B. CONSENT FOR PUBLICATION

All authors have approved the manuscript and agree with its submission.

C. FUNDING

The authors declare that no funding was received for conducting this review.

D. AVAILABILITY OF DATA AND MATERIAL

The authors state that all pictures and images included in this review belong to the authors. There are no copyright conflicts.

E. CONFLICT OF INTEREST

The authors have no conflicts of interest to declare.

References

[1] Antonio Acernese et al. "Informed Hybrid A Star-based Path Planning Algorithm in Unstructured Environments". In: *2024 European Control Conference (ECC)*. 2024, pp. 1039–1044. DOI: 10 . 23919 / ECC64448.2024.10590977.

[2] Naoki Akai et al. "Development of magnetic navigation method based on distributed control system using magnetic and geometric landmarks". In: *ROBOMECH Journal* 1.1 (Nov. 2014), p. 21. ISSN: 2197-4225. DOI: 10 . 1186 / s40648 - 014 - 0021 - 8. URL: https : //doi.org/10.1186/s40648-014-0021-8.

[3] Szilárd Aradi. "Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 23.2 (2022), pp. 740–759. DOI: 10 . 1109 / TITS.2020.3024655.

[4] Paolo Arena, Luca Patanè, and Salvatore Taffara. "Learning risk-mediated traversability maps in unstructured terrains navigation through robot-oriented models". In: *Information Sciences* 576 (2021), pp. 1–23. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.06.007. URL: https://www.sciencedirect.com/science/article/pii/S0020025521005946.

[5] Arash Asgharivaskasi and Nikolay Atanasov. "Active Bayesian Multi-class Mapping from Range and Semantic Segmentation Observations". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1–7. DOI: 10 . 1109 / ICRA48506 . 2021.9561711.

[6] Claudine Badue et al. "Self-driving cars: A survey". In: *Expert systems with applications* 165 (2021), p. 113816.

[7] Jorge Biolchini et al. *Systematic Review in Software Engineering*. Jan. 2005.

TABLE 3. Local planning approach by year

Year	Graph search	Graph search + Optimization-based	DWA	Multi-path gen	Optimization-based	Sampling	ML
2024	-	[1]	-	-	-	-	-
2023	[67]	[37](+ML), [22]	-	-	[59]	-	-
2022	[43]	[52, 53]	-	-	[21, 41]	-	-
2021	[61]	-	-	-	[5]	-	[4](+ Graph search)
2020	-	-	-	-	-	[47]	-
2019	[60, 19]	-	[69]	[64]	[26]	-	-
2018	[39]	-	-	-	-	-	[28]
2017	[40]	-	-	[68]	-	[56]	-
2016	-	-	-	-	-	[44]	-
2015	-	-	-	-	-	-	[8](+ Graph search)
2014	[13]	-	[2, 34]	-	-	-	-

TABLE 4. Comparing the local planner techniques by improvements and weaknesses (1/2)

Local planner	Variation	Improvements	Weakness	Nonholonomic constraints
Hybrid A*	Hybrid A* as the first step	Hybrid A* is used as the first step, creating an initial guess. The trajectory planning is then solved as an optimization problem [1, 52]. In [22], it handles both static and dynamic collisions.	It needs to pre-compute the initial path to be optimized, so the total planning time is the sum of the initial path step and the optimization step.	Reeds-Sheep
	Traversability as cost for the heuristics	Generates safer paths than pure Hybrid A* by checking the traversability information of a node [43].	Requires 3D environment perception.	Reeds-Sheep
	Catmull-Rom interpolation	[60] uses the heading information when computing cost heuristics and applies Catmull-Rom interpolation to smooth the final path.	Requires manual parametrization of backwards cost.	Reeds-Sheep
Optimization-based	Deep-learning	[37] uses a trained DNN to generate the initial guess for the optimization problem. The final result runs about as fast as A*, RRT*, and JPS with a smoother path and a smaller jerk.	Requires pre-training. No nonholonomic constraints.	-
	OPC	[21] can handle low- and high-speed dynamic obstacles	The accuracy decreases with the predicted distance. No nonholonomic constraints.	-
	GP-MPC	[41] only optimizes critical variables to improve performance.	May not converge depending on the initial guess. No nonholonomic constraints.	-
RRT	Potential Field, RRT*	[59] plans for safety first by using Potential Field to prevent chosing paths too close from obstacles. Outperforms RRT and RRT* up to 61% on the initial solution, with a smoother and safer path.	The final path length is longer than RRT. The final solution takes much longer than RRT/RRT*.	Kinodynamics
	-	[47] executes obstacle avoidance in parallel and MRAC to control motion. RRT can produce anytime partial results.	Unpredictable execution time. Low smoothness due to RRT and no nonholonomic constraints.	-
	RRT*	[56] uses a global planner for long-distant planning. Anytime response due to RRT.	The search space is built along the entire mission, from start to goal, which may limit the mission size.	Dubins
A*	Deep-learning	[61] uses a DNN to estimate the end-to-end driving energy from point clouds, which is used as one of the cost-functions by A*	It requires pre-training and does not implement behavior for terrains outside the initial scope.	Kinodynamics
	Adaptive A*	[19] uses a single underlying data structure for terrain modeling and path planning.	No nonholonomic constraints.	-
	Kinodynamics, velocity profile	[13] supports kinodynamic constraints and computes velocity profiles.	The cost function uses weights that are manually tuned.	Kinodynamics
	energy-aware	[67] implements an energy-aware A* planning, capable of reducing energy costs by 26.9%	It does not take the terrain height into account. Dynamic objects require re-planning, which reduces efficiency.	Kinodynamics

TABLE 5. Comparing the local planner techniques by improvements and weaknesses (2/2)

Local planner	Variation	Improvements	Weakness	Nonholonomic constraints
PMP	-	[53] uses an approach similar to Hybrid A*. It applies an optimization process to smooth the final path. Outperforms hybrid-state A* and wait and go (HAWG)	Does not account for uncertainty in prediction.	Kinodynamics / Dubins
Based on Information Theory		[5] quickly evaluates many potential robot trajectories online.	It does not scale to large environments. No non-holonomic constrains	-
Dijkstra	Random Forest, Decision Tree, Neural Networks	[4] uses a light shallow network that reaches F1 score comparable to deep networks such as CNN to build a direction-based traversability map.	Requires pre-training. Does not implement planning behavior for terrains outside the initial scope. Does not account for non-holonomic constraints. Dijkstra is slower than other heuristics-based graph search such as A*	-
DWA		[69] fuses semantic information with geometric information to represent traversability. [34] solution is robust to noisy data from stereo cameras and degradation of the estimated positions. Global planning can perform incremental re-planning.	DWA is well-suited for proximity sensors such as ultrasonic transducers but does not behave well in long distances [16]. Does not account for non-holonomic constraints.	-
DF-FS		[39] is robust in the absence of a local goal waypoint and the uncertainty of positioning and perception	When there are few obstacles on both sides, some sporadic obstacles can cause deviation from the expected route.	Kinodynamics
End-to-end Neural Networks		[28] uses an end-to-end solution, which simplify the system design. Can be trained to account for new scenarios.	It needs robust training data. End-to-end inherently lack precise mathematical guarantees regarding safety. Risks of a covariance shift during the training phase and causal confusion [10].	-
Multiple planners	Field D*, interpolation	In [40] multiple specialized planning approaches handle different driving scenarios. Field D* for corner or intersection. Otherwise, parabolic function and middle of the road detection are used to improve smoothness	Correct planning depends on scenario identification. It does not take non-holonomic constraints into account.	-
Multiple path generation and selection	Douglas-Peucker optimization	Multiple path generation allows for fast planning. In [68] the prospect planning is extended to allow the car to run faster, but only for object detection, which speeds up reaction to obstacles, allowing the car to theoretically handle up to 26.853 km/h.	The current local planning can only handle simple overtake at up to 25 km/h. It does not account for non-holonomic constraints.	
	GNRR	[64] performs fast planning by first using a General Regression Neural Network to find the centerline and then use Bézier curves to correct by multipath offset, which ensures path smoothness	No testing in complex scenarios with multiple obstacles. GNRR must be trained. If centerline is not correctly identified, the local planner may fail to converge.	-
SVM		In [8], Support Vector Machine is able to perform collision avoidance for obstacles on both sides, while maintaining an equally distant, secure trajectory. Fast local planner.	Contrary to the local planner, the global planner is very slow and executes an update every 11 minutes, which raises total planning time to up to 80s. It does not account for non-holonomic constraints.	-
Breshenham		[26] uses a cost-valley approach that performs well in narrow passages and dirty roads.	The trajectory feasibility check takes only minimal distance into account. It does not account for non-holonomic constraints.	-
RASMO		[44] takes speed into consideration as a velocity-dependent cost to avoid vibration	It is slow and does not account for non-holonomic constraints.	-

[8] Konstantinos Charalampous, Ioannis Kostavelis, and Antonios Gasteratos. "Thorough robot navigation based on SVM local planning". In: *Robotics and Autonomous Systems* 70 (2015), pp. 166–180. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2015.02.010. URL: https://www.sciencedirect.com/science/article/pii/S0921889015000342.

[9] Benjamin Charrow et al. "Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping". In: *Proceedings of Robotics: Science and Sys-*

tems. Rome, Italy, July 2015, pp. 18, 27. DOI: 10.15607/RSS.2015.XI.003.

[10] Li Chen et al. "End-to-end Autonomous Driving: Challenges and Frontiers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), pp. 1–20. DOI: 10.1109/TPAMI.2024.3435937.

[11] Thomas Collins, J.J. Collins, and Donor Ryan. "Occupancy grid mapping: An empirical evaluation". In: *2007 Mediterranean Conference on Control & Automation*. 2007, pp. 1–6. DOI: 10.1109/MED.2007.4433772.

[12] Dmitri Dolgov et al. "Practical Search Techniques in Path Planning for Autonomous Driving". In: *AAAI Workshop - Technical Report* (Jan. 2008).

[13] Dennis Fassbender, André Mueller, and Hans-Joachim Wuensche. "Trajectory planning for car-like robots in unknown, unstructured environments". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3630–3635. DOI: 10.1109/IROS.2014.6943071.

[14] Dave Ferguson and Anthony Stentz. "Field D*: An Interpolation-Based Path Planner and Replanner". In: *Robotics Research*. Ed. by Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 239–253. ISBN: 978-3-540-48113-3.

[15] D. Fox, W. Burgard, and S. Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics and Automation Magazine* 4.1 (1997), pp. 23–33. DOI: 10.1109/100.580977.

[16] D. Fox, W. Burgard, and S. Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics and Automation Magazine* 4.1 (1997), pp. 23–33. DOI: 10.1109/100.580977.

[17] Carlos E. García, David M. Prett, and Manfred Morari. "Model predictive control: Theory and practice—A survey". In: *Automatica* 25.3 (1989), pp. 335–348. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(89)90002-2. URL: https://www.sciencedirect.com/science/article/pii/0005109889900022.

[18] David Gingras et al. "Rough Terrain Reconstruction for Rover Motion Planning". In: *2010 Canadian Conference on Computer and Robot Vision*. 2010, pp. 191–198. DOI: 10.1109/CRV.2010.32.

[19] Ueli Graf et al. "Optimization-Based Terrain Analysis and Path Planning in Unstructured Environments". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5614–5620. DOI: 10.1109/ICRA.2019.8794331.

[20] Dario Calogero Guastella and Giovanni Muscato. "Learning-Based Methods of Perception and Navigation for Ground Vehicles in Unstructured Environments: A Review". In: *Sensors* 21.1 (2021). ISSN: 1424-8220. DOI: 10.3390/s21010073. URL: https://www.mdpi.com/1424-8220/21/1/73.

[21] Yuqing Guo et al. "Down-Sized Initialization for Optimization-Based Unstructured Trajectory Planning by Only Optimizing Critical Variables". In: *IEEE Transactions on Intelligent Vehicles* 8.1 (2023), pp. 709–720. DOI: 10.1109/TIV.2022.3163335.

[22] Zhichao Han et al. "An Efficient Spatial-Temporal Trajectory Planner for Autonomous Vehicles in Unstructured Environments". In: *IEEE Transactions on Intelligent Transportation Systems* 25.2 (2024), pp. 1797–1814. DOI: 10.1109/TITS.2023.3315320.

[23] Zhichao Han et al. "An Efficient Spatial-Temporal Trajectory Planner for Autonomous Vehicles in Unstructured Environments". In: *IEEE Transactions on Intelligent Transportation Systems* 25.2 (2024), pp. 1797–1814. DOI: 10.1109/TITS.2023.3315320.

[24] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[25] Marcel Häselich et al. "Spline templates for fast path planning in unstructured environments". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 3545–3550. DOI: 10.1109/IROS.2011.6094756.

[26] Nina Felicitas Heide et al. "Performance Optimization of Autonomous Platforms in Unstructured Outdoor Environments Using a Novel Constrained Planning Approach". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2359–2364. DOI: 10.1109/IVS.2019.8813805.

[27] Heiko Hirschmuller. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 328–341. DOI: 10.1109/TPAMI.2007.1166.

[28] Christopher J. Holder and Toby P. Breckon. "Learning to Drive: Using Visual Odometry to Bootstrap Deep Learning for Off-Road Path Prediction". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 2104–2110. DOI: 10.1109/IVS.2018.8500526.

[29] Armin Hornung et al. "OctoMap: an efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* 34.3 (Apr. 2013), pp. 189–206. ISSN: 1573-7527. DOI: 10.1007/s10514-012-9321-0. URL: https://doi.org/10.1007/s10514-012-9321-0.

[30] SAE International. *SAE J3016TM LEVELS OF DRIVING AUTOMATIONT*. 2021. URL: https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf (visited on 04/20/2023).

[31] Joel Janai et al. "Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art". In: *Foundations and Trends® in Computer Graphics and Vision* 12.1–3 (2020), pp. 1–308. ISSN: 1572-2740.

DOI: 10.1561/0600000079. URL: http://dx.doi.org/10.1561/0600000079.

[32] Jennifer King and Maxim Likhachev. "Efficient cost computation in cost map planning for non-circular robots". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 3924–3930. DOI: 10.1109/IROS.2009.5354074.

[33] Barbara Kitchenham. *Procedures for Performing Systematic Reviews*. Tech. rep. Keele, UK, Keele Univ., Aug. 2004.

[34] Rafael Luiz Klaser, Fernando Santos Osório, and Denis Wolf. "Vision-Based Autonomous Navigation with a Probabilistic Occupancy Map on Unstructured Scenarios". In: *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*. 2014, pp. 146–150. DOI: 10.1109/SBR.LARS.Robocontrol.2014.13.

[35] I.S. Kweon and T. Kanade. "High-resolution terrain map from multiple sensor data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 278–292. DOI: 10.1109/34.121795.

[36] Jinhan Lee, Charles Pippin, and Tucker Balch. "Cost based planning with RRT in outdoor environments". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 684–689. DOI: 10.1109/IROS.2008.4651052.

[37] Han Li et al. "Trajectory Planning for Autonomous Driving in Unstructured Scenarios Based on Deep Learning and Quadratic Optimization". In: *IEEE Transactions on Vehicular Technology* 73.4 (2024), pp. 4886–4903. DOI: 10.1109/TVT.2023.3330581.

[38] Jiankun Li and C.-C.J. Kuo. "A dual graph approach to 3D triangular mesh compression". In: *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*. Vol. 2. 1998, 891–894 vol.2. DOI: 10.1109/ICIP.1998.723699.

[39] Ning Li et al. "DF -FS based path planning algorithm with sparse waypoints in unstructured terrain". In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2018, pp. 1783–1788. DOI: 10.1109/ROBIO.2018.8664797.

[40] Ning Li et al. "DFA based autonomous decision-making for UGV in unstructured terrain". In: *2017 IEEE International Conference on Unmanned Systems (ICUS)*. 2017, pp. 34–39. DOI: 10.1109/ICUS.2017.8278313.

[41] Zhiyuan Li et al. "A Learning-Based Model Predictive Trajectory Planning Controller for Automated Driving in Unstructured Dynamic Environments". In: *IEEE Transactions on Vehicular Technology* 71.6 (2022), pp. 5944–5959. DOI: 10.1109/TVT.2022.3159994.

[42] Maxim Likhachev et al. "Anytime Dynamic A*: An Anytime, Replanning Algorithm". In: *Proceedings of 15th International Conference on Automated Planning and Scheduling (ICAPS '05)*. June 2005, pp. 262–271.

[43] Jiayang Liu et al. "Hybrid Map-Based Path Planning for Robot Navigation in Unstructured Environments". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 2216–2223. DOI: 10.1109/IROS55552.2023.10341666.

[44] Sho Matsunaga et al. "Motion planning of mobile robot considering velocity-dependent cost and time". In: *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2016, pp. 619–624. DOI: 10.1109/ROBIO.2016.7866391.

[45] Khalid Al-Mutib et al. "D* Lite Based Real-Time Multi-Agent Path Planning in Dynamic Environments". In: *2011 Third International Conference on Computational Intelligence, Modelling & Simulation*. 2011, pp. 170–174. DOI: 10.1109/CIMSim.2011.38.

[46] Jianjun Ni et al. "A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods". In: *Applied Sciences* 10.8 (2020). ISSN: 2076-3417. DOI: 10.3390/app10082749. URL: https://www.mdpi.com/2076-3417/10/8/2749.

[47] Rodrigo W. S. M. de Oliveira et al. "A Robot Architecture for Outdoor Competitions". In: *Journal of Intelligent & Robotic Systems* 99.3 (Sept. 2020), pp. 629–646. ISSN: 1573-0409. DOI: 10.1007/s10846-019-01140-9. URL: https://doi.org/10.1007/s10846-019-01140-9.

[48] Brian Paden et al. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1 (Apr. 2016). DOI: 10.1109/TIV.2016.2578706.

[49] Panagiotis Papadakis. "Terrain traversability analysis methods for unmanned ground vehicles: A survey". In: *Engineering Applications of Artificial Intelligence* 26.4 (2013), pp. 1373–1385. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2013.01.006. URL: https://www.sciencedirect.com/science/article/pii/S095219761300016X.

[50] Janko Petereit et al. "Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments". In: *ROBOTIK 2012; 7th German Conference on Robotics*. 2012, pp. 1–6.

[51] Mikhail Pivtoraiko and Alonzo Kelly. "Efficient constrained path planning via search in state lattices". In: *Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS '05)*. Sept. 2005, 4b.

[52] Huayan Pu et al. "A Trajectory Planning Method Based on Quad-tree Convex Feasible Set in Unstructured Environments". In: *2022 China Automation Congress (CAC)*. 2022, pp. 3915–3920. DOI: 10.1109/CAC57257.2022.10054714.

[53] Yao Qi et al. "Hierarchical Motion Planning for Autonomous Vehicles in Unstructured Dynamic Environments". In: *IEEE Robotics and Automation Letters*

8.2 (2023), pp. 496–503. DOI: 10.1109/LRA.2022.3228159.

[54] R. Rajamani. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer US, 2011. ISBN: 9781461414339. URL: https://books.google.com.br/books?id=cZJFDox4KuUC.

[55] Francisco Rovira-Más, John Reid, and Quanyi Zhang. ''Stereovision Data Processing With 3 d Density Maps For Agricultural Vehicles''. In: *Transactions of the ASABE* 49 (July 2006). DOI: 10.13031/2013.21721.

[56] Neal Seegmiller et al. ''The Maverick planner: An efficient hierarchical planner for autonomous vehicles in unstructured environments''. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2018–2023. DOI: 10.1109/IROS.2017.8206021.

[57] Amritash Shekhar and Abhijeet Sharma. ''Review of Model Reference Adaptive Control''. In: *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*. 2018, pp. 1–5. DOI: 10.1109/ICICET.2018.8533713.

[58] Chao Shen et al. ''Multi-receptive field graph convolutional neural networks for pedestrian detection''. In: *IET Intelligent Transport Systems* 13.9 (2019), pp. 1319–1328. DOI: https://doi.org/10.1049/iet-its.2018.5618. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-its.2018.5618. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-its.2018.5618.

[59] Hongqing Tian et al. ''Driving risk-aversive motion planning in off-road environment''. In: *Expert Systems with Applications* 216 (2023), p. 119426. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.119426. URL: https://www.sciencedirect.com/science/article/pii/S0957417422024459.

[60] Kangbin Tu et al. ''Hybrid A* Based Motion Planning for Autonomous Vehicles in Unstructured Environment''. In: *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2019, pp. 1–4. DOI: 10.1109/ISCAS.2019.8702779.

[61] Marco Visca et al. ''Conv1D Energy-Aware Path Planner for Mobile Robots in Unstructured Environments''. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2279–2285. DOI: 10.1109/ICRA48506.2021.9560771.

[62] Antony Waldock and David W. Corne. ''Exploiting Prior Information in Multi-objective Route Planning''. In: *Parallel Problem Solving from Nature - PPSN XII*. Ed. by Carlos A. Coello Coello et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 11–21. ISBN: 978-3-642-32964-7.

[63] Limin Wang et al. ''Knowledge Guided Disambiguation for Large-Scale Scene Classification With Multi-Resolution CNNs''. en. In: *IEEE Trans Image Process* 26.4 (Feb. 2017), pp. 2055–2068.

[64] Shaobo Wang et al. ''Reference Path Correction for Autonomous Ground Vehicles Driving Over Rough Terrain''. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2405–2410. DOI: 10.1109/IVS.2019.8813812.

[65] Ekim Yurtsever et al. ''A Survey of Autonomous Driving: Common Practices and Emerging Technologies''. In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149.

[66] Ekim Yurtsever et al. ''A Survey of Autonomous Driving: Common Practices and Emerging Technologies''. In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149.

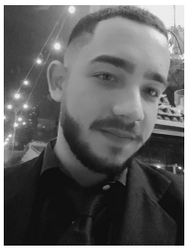
[67] Haojie Zhang et al. ''Energy efficient path planning for autonomous ground vehicles with ackermann steering''. In: *Robotics and Autonomous Systems* 162 (2023), p. 104366. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104366. URL: https://www.sciencedirect.com/science/article/pii/S0921889023000052.

[68] Kai Zhang et al. ''An efficient decision and planning method for high speed autonomous driving in dynamic environment''. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 806–811. DOI: 10.1109/IVS.2017.7995815.

[69] Yimo Zhao et al. ''Semantic Probabilistic Traversable Map Generation For Robot Path Planning''. In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 2576–2582. DOI: 10.1109/ROBIO49542.2019.8961533.

CRISTIANO SOUZA DE OLIVEIRA received his BSc degree in Computer Engineering from São Paulo University (USP), São Paulo, Brazil, in 2008. He is currently a researcher in trajectory planning for autonomous vehicle navigation. His research interests include computer vision, parallel algorithms, CUDA, artificial intelligence, sensor fusion and embedded systems. He also designs algorithms for distributed processing of large amounts of data for analytics, with a focus on performance improvements and data quality.

RAFAEL DE S. TOLEDO is a PhD candidate in Computer Science at the Universidade Federal de Santa Catarina (UFSC), Brazil. He received his Master's degree in Automation Engineering from UFSC in 2022, and his Bachelor's degree in Mechatronics Engineering from the Universidade do Estado do Amazonas in 2011. Rafael's research focuses on deep learning for autonomous vehicle navigation in unstructured environments and road distress detection. Rafael recently published his work at the conferences ENIAC 2021 and BRACIS 2024. He also addresses professional projects for background removal and image matting at Pixelcut.ai.



VICTOR H. TULUX OLIVEIRA VICTORIO received his BSc degree in Control and Automation Engineering from Universidade Católica Dom Bosco, Campo Grande, Brazil, in 2020. Currently, he is studying for an MSc degree in Computer Science with a focus on Autonomous Vehicle Navigation: Simultaneous Mapping and Localization. His research interests include computer vision, sensing, artificial intelligence, and embedded systems.

ALDO VON WANGENHEIM is a Professor at Federal University of Santa Catarina, where he also graduated in Computer Sciences in 1989. He received his Ph.D. degree in Computer Sciences from the University of Kaiserslautern in 1996. He coordinates the research on Autonomous Vehicles in rough environments at UFSC. He is also professor at the Computer Sciences, Information Systems and Medicine Undergraduate Programs at UFSC. He coordinates the Brazilian National Institute for Digital Convergence, a facility focused on advanced research.

APPENDIX. QUERY SEARCH STRINGS

TABLE 6. Query strings and the selection result for each location

Location	Search String	Result	Selected
Science Direct	(("unstructured" OR "offroad" OR "low-structured" OR "non-urban") AND ("path planning" OR "motion planning" OR "trajectory planning" OR "self-driving" OR "autonomous"))	470	2
ACM Digital Library	Title:("unstructured" OR "offroad" OR "low-structured" OR "non-urban") AND ("path planning" OR "motion planning" OR "trajectory planning" OR "self-driving" OR "autonomous") NOT "UAV" NOT "indoor" NOT "UVMS" NOT "water" NOT "submerged" NOT "SOCIAL" OR Abstract:("unstructured" OR "offroad" OR "semi-structured" OR "non-urban") AND ("path planning" OR "motion planning" OR "trajectory planning" OR "self-driving" OR "autonomous") NOT "UAV" NOT "indoor" NOT "UVMS" NOT "water" NOT "submerged" NOT "SOCIAL" OR Keyword:("unstructured" OR "offroad" OR "semi-structured" OR "non-urban") AND ("path planning" OR "motion planning" OR "trajectory planning" OR "self-driving" OR "autonomous") NOT "UAV" NOT "indoor" NOT "UVMS" NOT "water" NOT "submerged" NOT "SOCIAL"	600	8
IEEE Xplore	(("Document Title":"unstructured" OR "Document Title":"offroad" OR "Document Title":"semi-structured" OR "Document Title":"non-urban") AND ("Document Title":"path planning" OR "Document Title":"motion planning" OR "Document Title":"trajectory planning" OR "Document Title":"self-driving" OR "Document Title":"autonomous") OR Abstract:("Abstract":"unstructured" OR "Abstract":"offroad" OR "Abstract":"low-structured" OR "Abstract":"non-urban") AND ("Abstract":"path planning" OR "Abstract":"motion planning" OR "Abstract":"trajectory planning" OR "Abstract":"self-driving" OR "Abstract":"autonomous") OR AuthorKeywords:("Author Keywords":"unstructured" OR "Author Keywords":"offroad" OR "Author Keywords":"low-structured" OR "Author Keywords":"non-urban") AND ("Author Keywords":"path planning" OR "Author Keywords":"motion planning" OR "Author Keywords":"trajectory planning" OR "Author Keywords":"self-driving" OR "Author Keywords":"autonomous"))	78	17
Springer Link	(("unstructured" OR "offroad" OR "low-structured" OR "non-urban") AND ("path planning" OR "motion planning" OR "trajectory planning" OR "self-driving" OR "autonomous"))	266	0
Wiley	("path planning" OR "motion planning" OR "trajectory planning")	967	0
Total		2381	27