Self-Driving Cars: A Survey

Preprint · January 2019

12 authors, including:

Ranik Guidolini
Universidade Federal do Espírito Santo
15 PUBLICATIONS   **72** CITATIONS

SEE PROFILE

Raphael Vivacqua Carneiro
Universidade Federal do Espírito Santo
7 PUBLICATIONS   **28** CITATIONS

SEE PROFILE

Pedro Azevedo
Universidade Federal do Espírito Santo
7 PUBLICATIONS   **31** CITATIONS

SEE PROFILE

Vinicius Brito Cardoso
Universidade Federal do Espírito Santo
15 PUBLICATIONS   **91** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Intelligent Trading Architecture (ITA) View project

Parallel and Distributed Systems View project

Self-Driving Cars: A Survey

Claudine Badue, Rânik Guidolini, Raphael V. Carneiro, Pedro Azevedo, Vinicius B. Cardoso, Avelino Forechi, Luan F. R. Jesus, Rodrigo F. Berriel, Thiago M. Paixão, Filipe Mutz, Thiago Oliveira-Santos, Alberto F. De Souza

Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brasil
{claudine, ranik. carneiro.raphael, pedro, vinicius, avelino, luan.jesus, rodrigo.berriel, thiagopx, filipe , todsantos, alberto}@lcad.inf.ufes.br

Abstract—We survey research on self-driving cars published in the literature focusing on autonomous cars developed since the DARPA challenges, which are equipped with an autonomy system that can be categorized as SAE level 3 or higher. The architecture of the autonomy system of self-driving cars is typically organized into the perception system and the decision-making system. The perception system is generally divided into many subsystems responsible for tasks such as self-driving-car localization, static obstacles mapping, moving obstacles detection and tracking, road mapping, traffic signalization detection and recognition, among others. The decision-making system is commonly partitioned as well into many subsystems responsible for tasks such as route planning, path planning, behavior selection, motion planning, and control. In this survey, we present the typical architecture of the autonomy system of self-driving cars. We also review research on relevant methods for perception and decision making. Furthermore, we present a detailed description of the architecture of the autonomy system of the UFES's car, IARA. Finally, we list prominent autonomous research cars developed by technology companies and reported in the media

Index Terms— Self-driving cars, perception, obstacle mapping, road mapping, localization, moving obstacle detection and tracking, traffic signalization detection and recognition, decision making, route planning, behavior selection, motion planning, control.

Summary

I. INTRODUCTION

Autonomous cars (also known as driverless cars and self-driving cars) have been studied and developed by many universities, research centers, car companies, and companies of other industries around the world since the middle 1980s. Important examples of self-driving-cars' research platforms in the last two decades are the vehicle of the University of the Bundeswehr Munich (UniBw Munich) [DIC87], Navlab's mobile platform [THO91], UniBw Munich's and Daimler-Benz's vehicles "VaMP" and "VITA-2" [GER14], University of Pavia's and University of Parma's car "ARGO" [BRO99], and UBM's vehicles "VaMoRs" and "VaMP" [GRE00].

In order to spur technology for the development of self-driving cars, the Defense Advanced Research Projects Agency (DARPA) organized three competitions in the last decade. The first, named DARPA Grand Challenge, was realized at the Mojave Desert, USA, in 2004, and required driverless cars to navigate a 142 mi long course throughout desert trails within a 10 h time limit. All competing cars failed within the first few miles.

The DARPA Grand Challenge [BUE07] was repeated in 2005 and required robotic cars to navigate a 132 mi long route through flats, dry lake beds, and mountain passes, including three narrow tunnels and more than 100 sharp left and right turns. This competition had 23 finalists and 4 cars completed the route within the allotted time limit. The Stanford University's car, "Stanley" [THR07], claimed first place, and the Carnegie Mellon University's cars "Sandstorm" and "H1ghlander" finished in second and third places, respectively.

The third competition, known as the DARPA Urban Challenge [BUE09], was held at the former George Air Force Base, California, USA, in 2007, and required self-driving cars to navigate a 60 mi long route throughout a simulated urban environment, together with other self-driving and human driven cars, within a 6 h time limit. The cars had to obey California traffic rules. This competition had 11 finalists and 6

cars completed the route within the allotted time limit. The Carnegie Mellon University's car, "Boss" [URM08], claimed first place, the Stanford University's car, "Junior" [MON08], finished in second, and the Virginia Tech's car, "Odin" [BAC08], came in third place. Even though these competitions presented challenges much simpler than those typically seen in everyday traffic, they have being hailed as milestones for the development of self-driving cars.

Since the DARPA challenges, many self-driving-car competitions and trials have been performed. Relevant examples include: the European Land-Robot Trial (ELROB) [ELR18], which has being held from 2006 to the current year; the Intelligent Vehicle Future Challenge [XIN14], from 2009 to 2013; the Autonomous Vehicle Competition, from 2009 to 2017 [AUT17]; the Hyundai Autonomous Challenge, in 2010 [CER11]; the VisLab Intercontinental Autonomous Challenge, in 2010 [BRO12]; the Grand Cooperative Driving Challenge (GCDC) [GCD16], in 2011 and 2016; and the Public Road Urban Driverless Car Test, in 2013 [BRO15]. At the same time, research on driverless cars has accelerated in both academia and industry around the world. Notable examples of universities conducting research on robotic cars comprise Carnegie Mellon University [CAR18], Stanford University [STA18], MIT [MIT17], Virginia Tech [VIR18], and FZI Research Center for Information Technology [FZI18]. Notable examples of companies include Google [WAY18], Uber [UBE18], Baidu [APO18], Lyft [LYF18], Aptiv [APT18], Tesla [TES18], Nvidia [NVI18], Aurora [AUR18], Zenuity [ZEN18], Daimler and Bosch [BOS18], Argo AI [ARG18], Renesas Autonomy [REN18], Almotive [AIM18], AutoX [AUT18], Mobileye [MOB18], Ambarella [AMB18], Pony.ai [PON18], JD [JD18], Idriverplus [IDR18], Toyota [TOY18], Ford [FOR18], Volvo [VOLV18], and Mercedes Benz [MER18].

Although most of the university research on self-driving cars has been conducted in the United States of America, Europe and Asia, some relevant investigations have been carried out in China, Brazil and other countries. Relevant examples of self-driving-cars' research platforms in Brazil are the Universidade Federal de Minas Gerais (UFMG)'s car, CADU [LIM10] [SAB10] [LIM13] [DIA15a], Universidade de São Paulo's car, CARINA [FER14] [MAS14] [SHI16] [HAT17], and the Universidade Federal do Espírito Santo (UFES)'s car, IARA [MUT16] [CAR17] [GUI16] [GUI17]. IARA was the first Brazilian driverless car to travel autonomously 74 km on urban roads and highways.

To gauge the level of autonomy of self-driving cars, SAE International (formerly simply SAE, or Society of Automotive Engineers) published a classification system based on the amount of human driver intervention and attentiveness required by them, in which the level of autonomy of a self-driving car may range from level 0 (the car's autonomy system issues warnings and may momentarily intervene but has no sustained car control) to level 5 (no human intervention is required in any circumstance) [SAE16]. In this paper, we survey research on self-driving cars published in the literature focusing on autonomous cars developed since the DARPA

challenges, which are equipped with an autonomy system that can be categorized as SAE level 3 or higher [SAE16].

The architecture of the autonomy system of self-driving cars is typically organized into two main parts: the perception system, and the decision-making system [PAD16]. The perception system is generally divided into many subsystems responsible for tasks such as self-driving-car localization, static obstacles mapping, moving obstacles detection and tracking, road mapping, traffic signalization detection and recognition, among others. The decision-making system is commonly partitioned as well into many subsystems responsible for tasks such as route planning, path planning, behavior selection, motion planning, and control, though this partitioning is somewhat blurred and there are several different variations in the literature [PAD16].

In this survey, we present the typical architecture of the autonomy system of self-driving cars. We also review research on relevant methods for perception and decision making.

The remainder of this paper is structured as follows. In Section II, we present an overview of the typical architecture of the autonomy system of self-driving cars; comment on the responsibilities of the perception system, decision making system, and their subsystems; and present an overview of architectures of some autonomous vehicles reported in the literature. In Section III, we present research on important methods for the perception system, including obstacle mapping, road mapping, localization, moving obstacle detection and tracking, and traffic signalization detection and recognition. In Section IV, we present research on relevant techniques for the decision-making system, comprising the route planning, path planning, behavior selection, motion planning, and control subsystems. In Section V, we present a detailed description of the architecture of the autonomy system of the UFES's car, IARA. Finally, in Section VI, we list prominent autonomous research cars developed by technology companies and reported in the media.

II. OVERVIEW OF THE ARCHITECTURE OF SELF-DRIVING CARS

In this section, we present an overview of the typical architecture of the automation system of self-driving cars and comment on the responsibilities of the perception system, decision making system, and their subsystems.

Figure 1 shows a block diagram of the typical architecture of the automation system of self-driving cars, where the perception and decision making systems [PAD16] are shown as a collection of modules of different colors. The perception system is responsible for estimating the state of the car and for creating an internal (to the self-driving system) representation of the environment, using data captured by on-board sensors, such as Light Detection and Ranging (LIDAR), Radio Detection and Ranging (RADAR), camera, Global Positioning System (GPS), Inertial Measurement Unit (IMU), odometer, etc., and prior information about the sensors' models, road network, traffic rules, car dynamics, etc. The decision making

system is responsible for navigating the car from its initial position to the Final Goal defined by the user, considering the car state and the internal representation of the environment, as well as traffic rules and passengers' comfort.

In order to navigate the car throughout the environment, the decision-making system needs to know where the car is in it. The **Localizer** module (Figure 1) is responsible for estimating the car state (pose, linear velocities, angular velocities, etc.) in relation to static maps of the environment. These static maps, (or **Offline Maps**, Figure 1), are computed automatically before the autonomous operation, typically using the sensors of the self-driving car itself, although manual annotations (i.e. the position of pedestrian crossing or of traffic lights) or editing (i.e. for removing non-static objects captured by the sensors) are usually required. A self-driving car may use one or more different Offline Maps, such as occupancy grid maps, remission maps, or landmark maps, for localization. We survey the literature on methods for generating such maps in Section III.B.

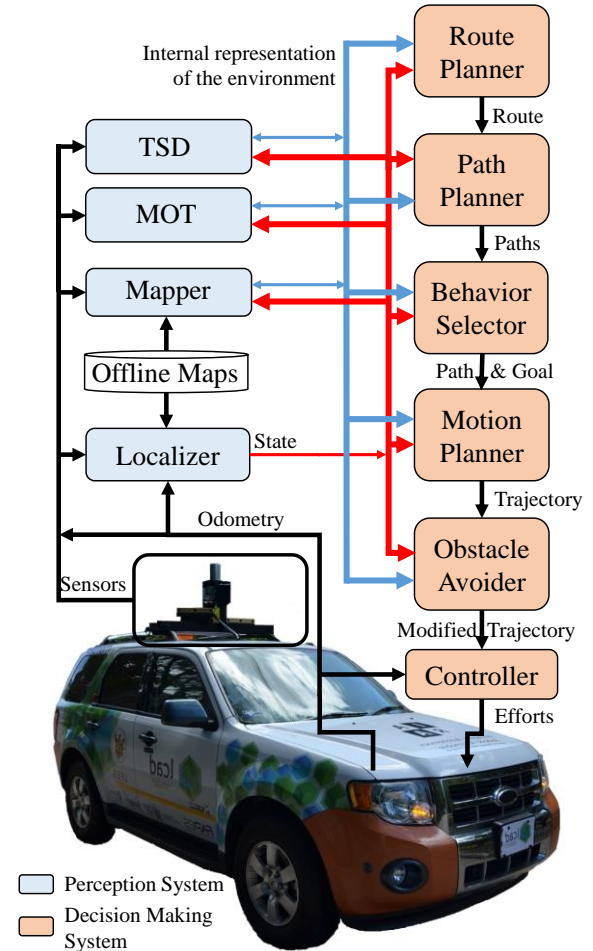Figure 1.Overview of the typical hierarchical architecture of self-driving cars. TSD denotes Traffic Signalization Detection and MOT, Moving Objects Tracking

The Localizer module receives as input the Offline Maps, sensors' data and the platform's odometry, and generates as output the State of the self-driving car (Figure 1). It is important to note that, although GPS may help the localization

process, it alone is not enough for proper localization in urban environments due to interferences caused by tall trees, building, tunnels, etc., that makes GPS positioning unreliable. We survey the literature on localization techniques in Section III.A.

The **Mapper** module receives as input the Offline Maps and the State, and generates as output the online map. This online map is typically a merge of information present in the Offline Maps and an occupancy grid map computed online using sensors' data and the current State. We survey the literature on methods for computing the online map in Section III.B. It is desirable that the online map contains only a static representation of the environment, since this may help the operation of some modules of the Decision Making system. To allow the detection and removal of moving objects of the online map, a Moving Objects Tracking module, or **MOT** (Figure 1), is typically employed. We survey the literature on methods for moving objects detection and tracking in the context of self-driving cars in Section III.D.

Horizontal (lane markings) and vertical (i.e. speed limits, traffic lights, etc.) traffic signalization must be recognized and obeyed by self-driving cars. The Traffic Signalization Detection module, or **TSD** (Figure 1), is responsible for the detection and recognition of traffic signalization. We survey the literature on methods for traffic signalization detection and recognition in Section III.E.

Given a Final Goal defined in the Offline Maps by the user, the **Route Planner** module computes a Route, W, in the Offline Maps, from the current State to the Final Goal. A Route is a sequence of way points, i.e. $W = \{w_1, w_2, ..., w_{|W|}\}$, where each way point, w_i, is a coordinate pair, i.e. $w_i = (x_i, y_i)$, in the Offline Maps. We survey the literature on methods for route planning in Section IV.A.

Given a Route, the **Path Planner** module computes, considering the car State and the internal representation of the environment as well as traffic rules, an odd set of Paths, $\mathbf{P} = \{P_1, P_2, ..., P_c, ..., P_{|\mathbf{P}|}\}$. A Path is a sequence of poses, i.e. $P_j = \{p_1, p_2, ..., p_{|P|}\}$, where each pose, p_i, is a coordinate pair in the Offline Maps, and the desired car's orientation at the position defined by this coordinate pair, i.e. $p_i = (x_i, y_i, \theta_i)$. The central Path, P_c, is aligned as best as possible with W, while Paths at its left side ($\mathbf{P}^l = \{P_1, P_2, ..., P_{c-1}\}$) and at its right side ($\mathbf{P}^r = \{P_{c+1}, ..., P_{|\mathbf{P}|}\}$) are Paths with the same initial pose of P_c, and with remaining poses departing from P_c to the left and to the right with different levels of aggressiveness. We survey the literature on methods for path planning in Section IV.B.1).

The **Behavior Selector** module is responsible for choosing the current driving behavior, such as lane keeping, intersection handling, traffic light handling, etc. It does so by selecting a Path, P_j, in $\mathbf{P}$, a pose in P_j a few seconds (about 5 s) ahead of the current State – the *decision horizon*, and the desired velocity at this pose. The pair pose in P_j and associated velocity is called Goal, and a $\text{Goal}_g = (x_g, y_g, \theta_g, v_g)$. The Behavior Selector chooses a Goal considering the current driving behavior and avoiding collisions with static and moving obstacles in the environment within the decision horizon time frame.

The **Motion Planner** module is responsible for computing a Trajectory, T, from the current car's state to the current Goal, which follows the Path defined by the Behavior Selector, satisfies car's kinematic and dynamic constraints, and provides comfort to the passengers. A Trajectory $T = \{c_1, c_2, ..., c_{|T|}\}$ is sequence of commands, $c_k = (v_k, \varphi_k, t_k)$, where v_k is the desired velocity at time k, φ_k is the desired steering angle at time k and t_k is the duration of c_k. A Trajectory takes the car from its current State to the Goal smoothly and safely. We survey the literature on methods for motion planning in Section IV.B.2).

The **Obstacle Avoider** module receives the Trajectory computed by the Motion Planner and changes it (typically reducing the velocity), if necessary, to avoid collisions. There is no much literature on methods for performing the functions of the Obstacle Avoider module. We discuss some relevant literature on this subject in Section IV.B.

Finally, the **Controller** module receives the Motion Planner trajectory, eventually modified by the Obstacle Avoider, and computes and sends Efforts commands to the actuators of the steering wheel, throttle and brakes in order to make the car execute the Modified Trajectory as best as the physical world allows. We survey the literature on methods for low level car control in Section IV.C.

In the following, we detail each one of these modules and the techniques used to implement them and their variants, grouped within perception and decision-making systems.

III. PERCEPTION

In this section, we present research on important methods proposed in the literature for the perception system of self-driving cars, including localizer (or localization), offline obstacle mapping, road mapping, moving obstacle tracking, and traffic signalization detection and recognition.

A. Localization

The localization module is responsible for estimating the self-driving car pose (position and orientation) relative to a map or road (e.g., represented by curbs or road marks). Most general-purpose localization subsystems are based on GPS. However, by and large, they are not applicable to urban self-driving cars, because the GPS signal cannot be guaranteed in occluded areas, such as under trees, in urban canyons (roads surrounded by large buildings) or in tunnels.

Various localization methods that do not depend on GPS have been proposed in the literature. They can be mainly categorized into three classes: LIDAR-based, LIDAR plus camera-based, and camera-based. LIDAR-based localization methods rely solely on LIDAR sensors, which offer measurement accuracy and easiness of processing. However, despite LIDAR industry efforts to reduce production costs, it still has a high price if compared to cameras. In typical LIDAR plus camera-based localization methods, LIDAR data

is used only to build the map, and camera data is employed to estimate the self-driving car's position relative to the map, which reduces costs. Camera-based localization approaches are cheap and convenient, even though typically less precise and/or reliable.

1) LIDAR-Based Localization

Levinson and Thrun [LEV10] proposed a localization method that uses offline grid maps of probability distributions over environment reflectance to LIDAR laser rays (laser remission grid maps, Figure 2); they have used the Velodyne HDL-64E LIDAR in their work. An unsupervised calibration method is used to calibrate the Velodyne HDL-64E' laser beams so that they all respond similarly to the objects with the same brightness, as seen by the LIDAR. A 2-dimension histogram filter [THR05] is employed to estimate the self-driving car position. As usual, the filter is comprised of two parts: the motion update (or prediction), to reduce confidence in our estimate based on motion, and the measurement update (or correction), to increase confidence in our estimate based on sensor data. In the motion update, the car motion is modeled as a random walk with Gaussian noise drift from a dead reckoning coordinate system (computed using the inertial update of an Applanix LV-420 navigation system) to the global coordinate system of the offline map. In the measurement step, they use, for different displacements, the similarity between the remission map computed online, with the remission map computed offline. Each displacement corresponds to one cell of the histogram of the histogram filter. To summarize the histogram into a single pose estimate, they use the center of mass of the probability distribution modeled by the histogram. The authors do not describe how they estimate the orientation, though. Their method has shown a root mean squared (RMS) lateral error of 9 cm and a RMS longitudinal error of 12 cm.
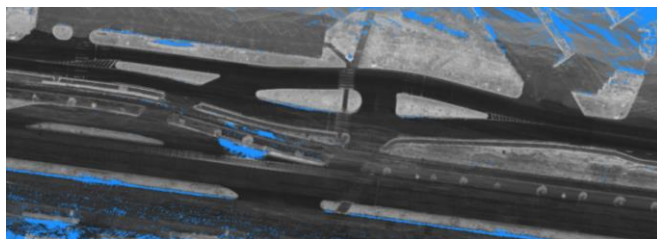

Figure 2. Remission map

Veronese et al. [VER15] proposed a MCL localization method that compares satellite aerial maps with re-emission maps. Aerial maps are downloaded offline from sources in the Internet, like OpenStreetMap, and remission maps are built online from LIDAR reflectance intensity data. The MCL algorithm is employed to estimate car's pose by matching remisusion maps to aerial maps using the normalized mutual information (NMI) measure to calculate the particles likelihood. The method was evaluated on a 6.5 km dataset collected by the robotic car "IARA" and achieved position estimation accuracy of 0.89 m. One advantage of this method

is that it does not require building a map specifically for the method.

Hata and Wolf [HAT16b] proposed a localization method based on road feature detection. Their curb detection algorithm uses ring compression analysis and least trimmed squares [HAT16b] to analyze the distance between consecutive concentric measurements (or rings) formed by a multilayer LIDAR (Velodyne HDL-32E) scan. The road marking detection algorithm uses Otsu thresholding [OTS79] to analyze LIDAR reflectance intensity data. Curb and road marking features are stored in a grid map. A Monte Carlo Localization (MCL) algorithm is employed to estimate the car pose by matching road features extracted from multilayer LIDAR measurements to the grid map. The method was evaluated on the autonomous vehicle "CARINA" [FER14], and has shown lateral and longitudinal localization estimation errors of less than 0.30 m.

Rohde et al. [ROH16] proposed a multilayer adaptive Monte Carlo localization (ML-AMCL) method that operates in combination with 3D point registration algorithms. For estimating the car pose, horizontal layers are extracted from 3D LIDAR measurements and separate AMCL instances are used to align layers with a 2D projection of a 3D point cloud map built using 3D point registration algorithms. For every pose estimate, a consistency check against a series of odometry measurements is performed. Consistent pose estimates are fused to a final pose estimate. The method was evaluated on real world data and achieved position estimation errors of 0.25 m relative to the GPS reference. Their map is, however, expensive to store, since it is a 3D map.

Veronese et al. **[VER16]** proposed a localization method based on the MCL algorithm that corrects particles' poses by map-matching between 2D online occupancy grid-maps and 2D offline occupancy grid-maps, as illustrated in Figure 3. Two map-matching distance functions were evaluated: an improved version of the traditional Likelihood Field distance between two grid-maps, and an adapted standard Cosine distance between two high-dimensional vectors. An experimental evaluation on the IARA self-driving car demonstrated that the localization method is able to operate at about 100 Hz using the Cosine distance function, with lateral and longitudinal errors of 0.13 m and 0.26 m, respectively.

(a)   (b)

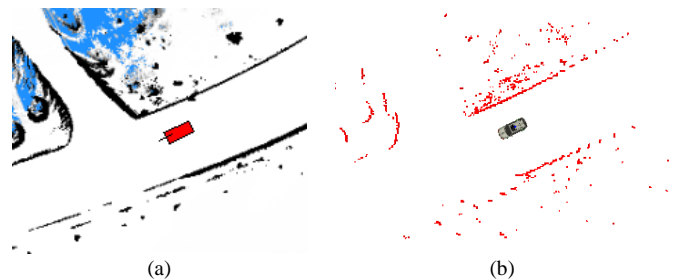Figure 3. Localization method proposed by Veronese et al. [VER16]. (a) Offline occupancy grid-map – the red rectangle is the car's localization, black cells contain obstacles, white cells are obstacle-free, and blue cells are regions untouched by sensors during mapping. (b) Online occupancy grid-map. The online map is matched against the offline map to compute the self-driving car's precise localization.

Wolcott and Eustice [WOL17] proposed a probabilistic localization method that models the world as a multiresolution map of mixture of Gaussians. Their Gaussian mixture maps represent the height and reflectance intensity (remission) distribution of the environment measured by multilayer LIDAR scanners (Velodyne HDL-32E). An extended Kalman filter (EKF) localization algorithm is used to estimate the car's pose by registering 3D point clouds against the Gaussian mixture multiresolution-maps. The method was evaluated on two driverless cars in adverse weather conditions and presented localization estimation errors of about 0.15 m.

2) LIDAR plus Camera-Based Localization

Some methods use LIDAR data to build a map and camera data to estimate the location of the self-driving car relative to this map. Xu et al. [XU17] proposed a localization method that matches stereo images to a 3D point-cloud map. The map was generated by a mapping company (http://www.whatmms.com) and it is composed of geometric data (latitude, longitude and altitude) and remission data acquired from odometer, RTK-GPS and 2D LIDAR scanners. Xu et al. transform map's 3D-points from the real-world coordinate system to the camera coordinate system, and extract depth and intensity images from them. A MCL algorithm is used to estimate the car location by matching stereo depth and intensity images taken from the car's camera to depth and intensity images extracted from the 3D point-cloud map. The method was evaluated on real world data and presented location estimation errors between 0.08 m and 0.25 m.

Viswanathan et al. [VIS16] proposed a method for self-driving-car localization that matches ground panoramic-images to satellite images captured in different seasons of the year. In their method, LIDAR data is classified into ground/non-ground categories. Next, ground images captured by a panoramic camera in the self-driving car are segmented into ground/non-ground regions using the LIDAR data, and then warped to obtain a bird's-eye view. The satellite image is also segmented into ground/non-ground regions using k-means clustering. A MCL is then used to estimate the pose by matching bird's-eye images to the satellite images. The method was evaluated on the NavLab11 self-driving car and achieved position estimation errors between 3 m and 4.8 m.

3) Camera-Based Localization

Some methods rely mainly on camera data to localize self-driving cars. Brubaker et al. [BRU16] proposed a localization method based on visual odometry and road maps. They use the OpenStreetMap, extracting from it all crossings and all drivable roads (represented as piece-wise linear segments) connecting them in the area of interest. They, then, build a graph-based representation of this road map and a probabilistic model of how the car traverses this graph. Using this probabilistic model and visual odometry measurements, they estimate the car displacement relative to the road map.

A recursive Bayesian filtering algorithm is used to perform inferences in the graph by exploiting its structure and the model of how the car moves, as measured by the visual odometry. This algorithm is able to pinpoint the car's position in the graph by increasing the probability that the current pose lies in a point of the graph that is correlated with latest car movements (distance traveling straight and recent curves) and by decreasing the probability that it is in a point that is not correlated. The localization method was evaluated on the KITTI visual odometry dataset and was able to localize the vehicle after 52 s of driving with an accuracy of 4 m on an 18 km^2 map containing 2,150 km of drivable roads.

Some methods use camera data to build a feature map. Ziegler et al., in [BER14], describe the localization methods used by the autonomous vehicle Bertha to drive autonomously on the historic Bertha-Benz-Memorial-Route. Two complementary vision based localization techniques were developed, named point feature based localization (PFL) and lane feature based localization (LFL). In PFL, the current camera image is compared with images of a sequence of camera images that is acquired previously during mapping using DIRD descriptors extracted from them. A global location estimate is recovered from the global position of the images captured during mapping. In LFL, the map, computed semi-automatically, provides a global geometric representation of road marking features (horizontal road signalization). The current camera image is matched against the map by detecting and associating road marking features extracted from a bird's-eye view of the camera image with horizontal road signalization stored in the map. Location estimates obtained by PFL and LFL are, then, combined by a Kalman filter (the authors do not provide an estimate of the combined localization error). Localization methods similar to LFL were proposed by Jo et al. [JO15], Suhr et al. [SUH17], and Vivacqua et al. [VIV17].

Some methods employ camera data to construct a feature map, but adopt alternative types of features. Radwan et al. [RAD16] proposed a localization method based on textual feature detection. Off-the-shelf text extraction techniques are used to identify text labels in the environment. A MCL algorithm is employed to integrate multiple observations. The method was evaluated on real world data and presented location estimation errors between 1 m and 25 m. Spangenberg et al. [SPA16] proposed the use of pole-like landmarks as primary features, because they are distinctive, long-term stable, and detectable by stereo cameras. Furthermore, they allow a memory efficient map representation. The feature detection is performed mainly by a stereo camera. Localization is performed by a MCL algorithm coupled with a Kalman filter for robustness and sensor fusion. The method was evaluated on an autonomous vehicle and achieved position estimation errors between 0.14 m and 0.19 m.

Some methods propose the use of neural networks to localize self-driving cars [LYR15] [OLI17]. They consist of correlating camera images and associated global positions. In the mapping phase, the neural network builds a representation of the environment. For that, it learns a sequence of images and global positions where images were captured, which are

stored in a neural map. In the localization phase, the neural network uses previously acquired knowledge provided by the neural map to estimate global positions from currently observed images. These methods present a meter scale error and have difficulty in localizing autonomous vehicles on large areas.

B. Offline Obstacle Mapping

The offline obstacle mapping subsystem is responsible for computing a map of obstacles in the environment of the self-driving car. This subsystem is fundamental to allow autonomous vehicles the ability to navigate on public roads safely without colliding with obstacles (e.g., signposts, curbs). An obstacle map contains information pertaining to the places that the car may or may not be able to navigate, distinguishing free (traversable) from occupied spaces. The car must always be in the free space. The obstacle map is constructed from sensor data during a mapping phase and stored for later use during the autonomous operation phase.

Representations of the state space are often distinguished between topological [CUM08] [MIL12] [FORE18] and metric [HOR13] [MUT16] [SCH18] representations. Topological representations model the state space as a graph, in which nodes indicate significant places (or features) and edges denote topological relationships between them (e.g., position, orientation, proximity, and connectivity). The resolution of these decompositions depends on the structure of the environment. Metric representations usually decompose the state space into regularly spaced cells. This decomposition does not depend on the location and shape of features. Spatial resolution of metric representations tends to be higher than that of topological representations. Such versatility and efficiency make them the most common space representation. For a review on the main vision-based methods for creating topological representations, readers are referred to Garcia-Fidalgo and Ortiz [FID15]. Here, we survey the most important methods for computing metric representations, which can be further subdivided into discrete and continuous space representations.

1) Discrete Space Metric Representations

For self-driving cars, one of the most common representations of the state space is the Occupancy Grid Map (OGM) proposed by Moravec and Elfes [ELF85]. An OGM discretizes the space into fixed size cells, usually of the order of decimeters. Each cell contains the probability of occupation of the region associated to it. The probability of occupation is updated independently for each cell using sensor data. 3D sensor measurements that characterize obstacles can be projected onto the 2D ground plane, for simplicity and efficiency purposes. The assumption of independence makes the OGM algorithm fast and easy. However, it generates a sparse state space representation, because only those cells reached by the sensor are updated [KIM13]. Thrun et al. [THR05] present a localization method on a 2D OGM using a Monte Carlo algorithm. Kuemmerle et al. [KUE07] apply

Monte Carlo localization on multi-level surface maps that represent different height ranges occupied in a 2D OGM. Levinson and Thrun [LEV10] presented a GraphSLAM algorithm that optimize an objective function in which adjacent vehicle poses are linked by inertial and odometry data. Then, they use a straightforward algorithm for generating a probabilistic map of 15x15cm grid cells. Mutz et al [MUT16] propose to integrate data captured by an odometer, 3D LIDAR Velodyne HDL-32E, IMU, and low-cost GPS in a GraphSLAM algorithm [THR05] in order to build an OGM for the autonomous vehicle IARA. GPS data is used to identify and remove systematic errors in odometer data. GPS data is also employed to detect revisited regions and a Generalized Iterative Closest Point (GICP) algorithm [SEG09] to estimate displacements between poses in subsequent visits. Data from GPS, IMU, calibrated odometer, and loop closures are used to introduce restrictions to the GraphSLAM algorithm, which calculates the most likely set of poses given sensor data. Finally, 3D LIDAR data and their poses calculated by GraphSLAM are used to build an OGM. Konolige et al. [KON00] and Lee et al. [LEE08] use OGMs as input to motion planning algorithms, such as A-star and RRT. A side effect of optimization-based motion planning algorithms is the generation of paths very close to obstacles, since their objective function is to minimize traveled distance. If we consider that OGMs of interest to driverless cars should cover large urban areas (tens of thousands of squared meters), it would be prohibitive to store and query OGMs with uniformly distributed cells of centimeter size. For this reason, the most used resolution is of the order of decimeters. At this resolution (e.g., 0.2m), it is not uncommon for obstacles to appear at the frontier between two cells of an OGM, making vehicle's maneuvers prone to hazards when paths are very close to obstacles.

An alternative metric representation is the Octree based map proposed by Hornung et al. [HOR13], which stores information with varied 3D resolutions. Compared to OGMs with varied 3D resolutions, OctoMaps (octree-based map) store only the space that is observed and, consequently, are more efficient in terms of memory consumption. However, OctoMaps treat updation of sensor data and estimation of obstacle occupancy in a uniform and discrete manner. Consequently, they are slower than OGMs with uniform occupancy [CHE17]. Although OctoMaps represent a significant advantage in terms of memory consumption, intense computational complexity makes them unviable in real-time scenarios of driverless cars.

Another alternative metric representation is a hybrid map proposed by Droeschel et al. [DRO17], which stores occupancy and distance measurements with varied resolutions. For this, measurements are stored in grid cells of increasing size from the center of the car. Thus, computational efficiency is gained by having a high resolution in the proximity of the sensor and a lower resolution as the distance from the sensor grows. This follows characteristics of some sensors with respect to distance and measurement density (e.g., angular resolution of laser sensors).

2) Continuous Space Metric Representations

Traditional methods for computing OGMs are constrained by the assumption that the probability of occupancy of each cell of the map is modeled as an independent random variable [THR05]. Intuitively, this assumption is not true, because real environments have some inherent structure. An alternative metric representation is the Gaussian Process Occupancy Map (GPOM) proposed by Doherty et al. [DOH16]. A GPOM uses a Gaussian Process (GP) to learn the structure of the environment from sparse sensor measurements in a training dataset and, subsequently, estimate the probability of occupation of OGM cells that were not directly intercepted by the sensor. Experiments have shown that localization errors are more than three times lower as those estimated by a particle filter localization and OGM [HAT16a]. However, this inference carries a high computational cost of $O(N^3)$, where N is the number of samples in the training dataset, and, therefore, it is not directly applicable to large-scale scenarios of self-driving cars [KIM13].

Another alternative metric representation is the Hilbert map proposed by Ramos and Ott [RAM16]. Experiments showed that a Hilbert map presents accuracy comparable to a Gaussian Process Occupancy Map (GPOM), but has time complexity linear with the number of samples in the training dataset.

A further alternative metric representation is the Discrete Cosine Transform (DCT) map proposed by Schaefer et al. [SCH18]. A DCT map is represented in the discrete frequency domain and converted to a continuously differentiable field in the position domain using the continuous extension of the inverse DCT. A DCT map assigns to each point of the space a LIDAR decay rate, which models the local permeability of the space for laser rays. In this way, the map can describe obstacles of different laser permeability, from completely opaque to completely transparent. DCT maps represent LIDAR data more accurately than OGM, GPOM, and Hilbert map regarding the same memory requirements. Nonetheless, the above continuous metric representations are still slower than OGM and not widely applicable to large-scale and real-time driverless scenarios yet.

C. Road Mapping

The road mapping subsystem is responsible for gathering information of roads and lanes in the surroundings of the self-driving car, and representing them in a map with geometrical and topological properties, including interconnections and restrictions. The main topics of the road mapping subsystem are the map representation and the map creation.

1) Road Map Representation

As with obstacle maps (Section III.B), road maps are usually distinguished between metric and topological maps.

a) Metric Representation

A simple metric representation for a road map is a grid map, which discretizes the environment into a matrix of fixed size cells that contain information about belonging to a road or not, and costs of moving to its neighboring cells. Road grid maps are simple and easy to understand. However, if movement costs are uniform across large areas of a road map, then using a grid representation might require a wasteful use of memory space and processing time.

Sequences of waypoints are an alternative to compress the path description in a large road grid map. A waypoint is a point along a path in a road grid map. Sequences of waypoints can be defined manually or automatically extracted from a road grid map. For the 2005 DARPA Grand Challenge, it was proposed the Route Data Definition File (RDDF) [DAR05], which is a formatted file that contains waypoint coordinates and other associated information (latitude, longitude, lateral boundary offset, and course speed) that specify the path for operation of autonomous vehicles.

Carneiro et al. [CARN18] propose a road map to infer the position and relevant properties of lanes in urban roads for the driverless car IARA, which uses both a road grid map and RDDF path, as illustrated in Figure 4. IARA's road grid map contains square cells of 0.2×0.2 m. A nonzero code is assigned to each cell that belongs to a lane. Codes ranging from 1 to 16 represent relative distances from cells to the center of the lane and types of lane markings (broken, solid, or none) present in the cells. IARA's RDDF path contains waypoints that are 0.5 m spaced and automatically extracted from the road grid map by an algorithm which rewards cells that are closer to the lane center. IARA's road grid map and RDDF path were successfully tested throughout an autonomous journey of 3.7 km on the ring road of the main campus of the Universidade Federal do Espírito Santo (UFES).
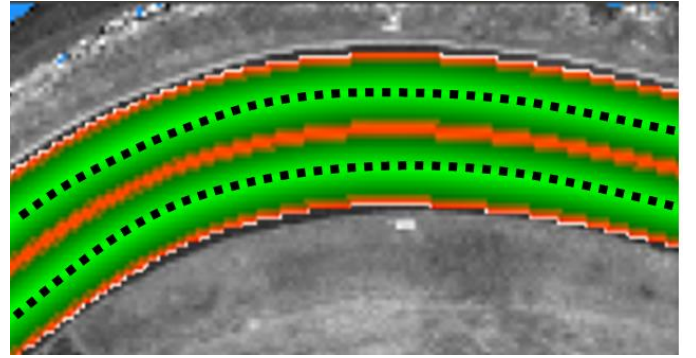
Figure 4. Road grid map and RDDF path used by the driverless car IARA [CARN18]. Shades of green and red denote the road grid map and black dots denote RDDF waypoints, which are automatically extracted from the road grid map.

b) Topological Representation

A more sophisticated representation for a road map is a topological map, which depicts the environment as a graph-like model, in which vertexes denote places and edges denote topological relationships between them. Topological maps can hold more complex information, including multiple lanes, lane crossings, and lane mergers.

For the 2007 DARPA Urban Challenge, it was proposed the Route Network Definition File (RNDF) [DAR07], which is a topological map defined as a formatted file that specifies road segments for operation of driverless cars. According to this file, the road network includes one or more segments, each of which comprises one or more lanes. A segment is characterized by the number of lanes, street name, and speed limit. A lane is characterized by the width of the lane, lane markings, and set of waypoints. Connections between lanes are characterized by exit and entry waypoints. Urmson et al. [URM08] use a graph model of the RNDF for the self-driving car Boss (Carnegie Mellon University's car that claimed first place in the 2007 DARPA Urban Challenge). Each node in the graph denotes a waypoint and directional edges denote lanes that connect the node to all other waypoints it can reach. Costs are assigned to the edges based on a combination of several factors, including expected time to traverse the lane associated to the edge, length of the lane, and complexity of the environment. Ramm et al. [RAM11] propose the OpenStreetMap (OSM), which models the environment with topological maps using three primitives, namely nodes, ways and relations. Nodes denote geographical points, ways denote lists of nodes (polylines), and relations consist of any number of members that may be of any of the three types and have specified roles. Other road properties, like the driving direction and the number of lanes, are given as properties of the elements.

Bender et al. [BEN14] propose a highly detailed topological road map, called lanelet map, for the autonomous vehicle Bertha. The lanelet map includes both geometrical and topological features of roads, such as roads, lanes, and intersections, using atomic interconnected drivable road segments, called lanelets, as illustrated in Figure 5. The geometry of a lanelet is defined by a left and a right bound, each one corresponding to a list of points (polyline). This representation implicitly defines the width and shape of each lane and its driving orientation. The adjacency of lanelets composes a weighted directed graph, in which each lanelet denotes a vertex and the length of a lanelet denotes the weight of its outgoing edges. Other elements describe constraints, such as speed limits and traffic regulations, like crossing and merging rights. The lanelet map was successfully tested throughout an autonomous journey of 103 km on the historic Bertha Benz Memorial Route.

High-Definition maps (HD maps) are a new generation of topological maps that are powering driverless cars. HD maps have high-precision at centimeter-level and contain rich information, such as lane positions, road boundaries, and road curvatures. Since the cost incurred to create HD maps is significant, there are platforms available to provide HD maps as a service. Dharia [DHA16] assessed and ranked the top vendors, namely Google, HERE, TomTom, and Apple.

2) Road Map Creation

The simplest method to create road maps is the manual annotation of road shapes extracted from aerial imagery. However, the high manual effort required by large urban road networks might turn manual annotation unfeasible. For this reason, methods for automated generation of road maps from aerial images have been proposed.
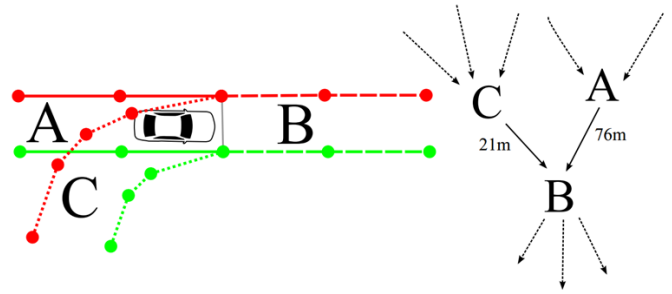

Figure 5. A graph model of a lanelet map used by the autonomous vehicle project [BEN14]. Red and green dots denote the polylines of left and right bounds of lanelets A, B and C, respectively. The graph shows the merge of A and C into B.

a) Manual Annotation

Urmson et al. [URM08] used manual annotation of road shapes extracted from aerial imagery in order to create a road map for the self-driving car Boss. The obtained local road shapes were accurate; however, global positions were not so accurate, due to the image resolution and the global registration. For this, their localization method dealt with road model errors using position filtering. Bender et al. [BEN14] also adopted manual annotation of all elements and properties of the lanelet map for the autonomous vehicle Bertha. Virtual top-view images were used as a foundation for the manual annotation of *lanelets* using the OSM format and Java OSM editor.

b) Automated Generation

Many methods have been proposed for automated generation of road maps from aerial images. Wegner et al. [WEG15] use higher-order conditional random fields (CRF) to model the structures of the road network by segmenting images into superpixels and adding paths to connect these superpixels. Mnih and Hinton [MNI10] use convolutional neural networks (CNN) to obtain road segments. A complementary task to road segmentation is lane detection from top-view or front-facing images. For reviews on recent advances in this topic, readers are referred to Yenikaya et al. [YEN13] and Hillel et al. [HIL14].

Aeberhard et al. [AEB15] use ground grid maps, in which each cell represents the probability of a ground location with high reflectivity, for the BMW's self-driving car. The ground grid maps are used to extract road boundaries using a second-degree polynomial model. Lane localization is used in conjunction with a digital map in order to obtain a higher level understanding of the environment. The map consists of mainly two layers: a semantic geometric layer and a localization layer. The semantic geometric layer contains the lane model geometry and the high-level semantic information, such as lane connectivity. The localization layer contains lane markings and road boundaries, which, together with GPS and vehicle odometry, can be used to match the vehicle onto the map.

Lee et al. [LEE18] also used LIDAR remission data to detect lane markings and camera images, in case lane makings are not well defined. The lane marking on the road was made to look good with headlight at night by using a special paint with good reflection on the light. With this feature, road marking could be detected with LIDAR, even in the case of changes in illumination due to rain or shadow. The lane marking detection technique based on camera image was run only in vulnerable situations (e.g., backlight and low light). This approach was successfully tested in a course 2 km in Seoul, Korea.

Carneiro et al. [CARN18] use deep neural networks (DNN) to infer the position and relevant properties of lanes with poor or absent horizontal signalization for the autonomous vehicle IARA. The DNN performs a segmentation of LIDAR remission grid maps into road grid maps, assigning nonzero codes (from 1 to 16) to map cells that belong to a lane, which denote relative distances to the center of the lane and types of lane markings in the cell. A dataset of tens of kilometers of marked road lanes was used to train the DNN, making it achieve an accuracy good enough for real-world autonomous driving of IARA.

A road map is not directly provided by road segmentation, which defines if map cells are part of a road or not. A complex post-processing pipeline is necessary in order to interpret the road segmentation, extract topological structures, and construct the road map. Bastani et al. [BAS18] propose the Road Tracer method that seeks to produce the road map directly from the CNN, rather than relying on intermediate image representations. It uses an iterative graph construction process that adds individual road segments one at a time and uses the CNN to decide on the next segment to be added. The test on aerial images covering 24 km^2 of 15 cities achieved an average error of 5% on a junction-by-junction matching metric.

D. Moving Objects Tracking

The Moving Objects Tracking (MOT) subsystem (also known as Detection and Tracking Multiple Objects - DATMO) is responsible for detecting and tracking the pose of moving obstacles in the environment around the self-driving car. This subsystem is essential to enable autonomous vehicles to take decisions and to avoid collision with potentially moving objects (e.g., other vehicles and pedestrians). Moving obstacles' positions over time are usually estimated from data captured by ranging sensors, such as LIDAR and RADAR, or stereo cameras. Images from monocular cameras are useful to provide rich appearance information, which can be explored to improve moving obstacle hypotheses. To cope with uncertainty of sensor measurements, Bayes filters (e.g., Kalman and particle filter) are employed for state prediction. Various methods for MOT have been proposed in the literature. Here, we present the most recent and relevant ones published in the last ten years. For earlier works, readers are referred to Petrovskaya et al. [PET12], Bernini et al. [BER14], and Girão et al. [GIR16].

Methods for MOT can be mainly categorized into six classes: traditional, model based, stereo vision based, grid map based, sensor fusion based, and deep learning based.

1) Traditional Based MOT

Traditional MOT methods follow three main steps: data segmentation, data association, and filtering [PET12]. In the data segmentation step, sensor data are segmented using clustering or pattern recognition techniques. In the data association step, segments of data are associated with targets (moving obstacles) using data association techniques. In the filtering phase, for each target, a position is estimated by taking the geometric mean of the data assigned to the target. Position estimates are usually updated by Kalman or particle filters. Amaral et al. [AMA15] propose a traditional method for detection and tracking of moving vehicles using 3D LIDAR sensor. The 3D LIDAR point cloud is segmented into clusters of points using the Euclidean distance. Obstacles (clusters) observed in the current scan sensor are associated with obstacles observed in previous scans using a nearest neighbor algorithm. States of obstacles are estimated using a particle filter algorithm. Obstacles with velocity above a given threshold are considered moving vehicles. Zhang et al. [ZHA13] build a cube bounding box for each cluster and use box dimensions for distinguishing whether a cluster is a vehicle or not. Data association is solved by an optimization algorithm. A Multiple Hypothesis Tracking (MHT) algorithm is employed to mitigate association errors. Hwang et al. [HWA16] use images captured by a monocular camera to filter out 3D LIDAR points that do not belong to moving objects (pedestrians, cyclists, and vehicles). Once filtered, object tracking is performed based on a segment matching technique using features extracted from images and 3D points.

2) Model Based MOT

Model-based methods directly infer from sensor data using physical models of sensors and geometric models of objects, and employing non-parametric filters (e.g., particle filters) [PET12]. Data segmentation and association steps are not required, because geometric object models associate data to targets. Petrovskaya and Thrun [PET09] present the model-based method for detection and tracking of moving vehicles adopted by the self-driving car "Junior" [MON08] (Stanford University's car that finished in second place in the 2007 DARPA Urban Challenge). Moving vehicle hypotheses are detected using differences over LIDAR data between consecutive scans. Instead of separating data segmentation and association steps, new sensor data are incorporated by updating the state of each vehicle target, which comprises vehicle pose and geometry. This is achieved by a hybrid formulation that combines Kalman filter and Rao-Blackwellized Particle Filter (RBPF). The work of Petrovskaya and Thrun [PET09] was revised by He et al. [HE16] that propose to combine RBPF with Scaling Series Particle Filter (SSPF) for geometry fitting and for motion estimate throughout the entire tracking process. The geometry became a tracked variable, which means that its previous state

is also used to predict the current state. Vu and Aycard [VU09] propose a model-based MOT method that aims at finding the most likely set of tracks (trajectories) of moving obstacles, given laser measurements over a sliding window of time. A track is a sequence of object shapes (L-shape, I-shape and mass point) produced over time by an object satisfying the constraints of a measurement model and motion model from frame to frame. Due to the high computational complexity of such a scheme, they employ a Data Driven Markov chain Monte Carlo (DD-MCMC) technique that enables traversing efficiently in the solution space to find the optimal solution. DD-MCMC is designed to sample the probability distribution of a set of tracks, given the set of observations within a time interval. At each iteration, DD-MCMC samples a new state (set of tracks) from the current state following a proposal distribution. The new candidate state is accepted with a given probability. To provide initial proposals for the DD-MCMC, dynamic segments are detected from laser measurements that fall into free or unexplored regions of an occupancy grid map and moving obstacle hypotheses are generated by fitting predefined object models to dynamic segments. Wang et al. [WAN15] adopt a similar method to the model-based one, but they do not assume prior categories for moving objects. A Bayes filter is responsible for joint estimation of the pose of the sensor, geometry of static local background, and dynamics and geometry of objects. Geometry information includes boundary points obtained with a 2D LIDAR. Basically, the system operates by iteratively updating tracked states and associating new measurements to current targets. Hierarchical data association works in two levels. In the first level, new observations (i.e., cluster of points) are matched against current dynamic or static targets. In the second level, boundary points of obstacles are updated.

3) Stereo Vision Based MOT

Stereo vision based methods rely on color and depth information provided by stereo pairs of images for detecting and tracking moving obstacles in the environment. Ess et al. [ESS10] propose a method for obstacle detection and recognition that uses only synchronized video from a forward-looking stereo camera. The focus of their work is obstacle tracking based on the per-frame output of pedestrian and car detectors. For obstacle detection, they employ a Support Vector Machine (SVM) classifier with Histogram of Oriented Gradients (HOG) features for categorizing each image region as obstacle or non-obstacle. For obstacle tracking, they apply a hypothesize-and-verify strategy for fitting a set of trajectories to the potentially detected obstacles, such that these trajectories together have a high posterior probability. The set of candidate trajectories is generated by Extended Kalman Filters (EKFs) initialized with obstacle detections. Finally, a model selection technique is used to retain only a minimal and conflict-free set of trajectories that explain past and present observations. Ziegler et al. [ZIE14a] describe the architecture of the modified Mercedes-Benz S-Class S500 "Bertha", which drove autonomously on the historic Bertha-Benz-Memorial-Route. For MOT, dense disparity images are reconstructed from stereo image pairs using Semi-Global Matching (SGM). All obstacles within the 3D environment are approximated by sets of thin and vertically oriented rectangles called super-pixels or stixels. Stixels are tracked over time using a Kalman filter. Finally, stixels are segmented into static background and moving obstacles using spatial, shape, and motion constraints. The spatio-temporal analysis is complemented by an appearance-based detection and recognition scheme, which exploits category-specific (pedestrian and vehicle) models and increases the robustness of the visual perception. The real-time recognition consists of three main phases: Region Of Interest (ROI) generation, obstacle classification, and object tracking. Chen et al. [CHEN17] compute a disparity map from a stereo image pair using a semi-global matching algorithm. Assisted by disparity maps, boundaries in the image segmentation produced by simple linear iterative clustering are classified into coplanar, hinge, and occlusion. Moving points are obtained during ego-motion estimation by a modified Random Sample Consensus (RANSAC) algorithm. Finally, moving obstacles are extracted by merging super-pixels according to boundary types and their movements.

4) Grid Map Based MOT

Grid map based methods start by constructing an occupancy grid map of the dynamic environment [PET12]. The map construction step is followed by data segmentation, data association, and filtering steps in order to provide object level representation of the scene. Nguyen et al. [NGU12] propose a grid-based method for detection and tracking of moving objects using stereo camera. The focus of their work is pedestrian detection and tracking. 3D points are reconstructed from a stereo image pair. An inverse sensor model is used to estimate the occupancy probability of each cell of the grid map based on the associated 3D points. A hierarchical segmentation method is employed to cluster grid cells into segments based on the regional distance between cells. Finally, an Interactive Multiple Model (IMM) method is applied to track moving obstacles. Azim and Aycard [AZI14] use an octree-based 3D local occupancy grid map that divides the environment into occupied, free, and unknown voxels. After construction of the local grid map, moving obstacles can be detected based on inconsistencies between observed free and occupied spaces in the local grid map. Dynamic voxels are clustered into moving objects, which are further divided into layers. Moving objects are classified into known categories (pedestrians, bikes, cars, or buses) using geometric features extracted from each layer. Ge et al. [GE17] leverage a 2.5D occupancy grid map to model static background and detect moving obstacles. A grid cell stores the average height of 3D points whose 2D projection falls into the cell space domain. Motion hypotheses are detected from discrepancies between the current grid and the background model.

5) Sensor Fusion Based MOT

Sensor fusion-based methods fuse data from various kinds of sensors (e.g., LIDAR, RADAR, and camera) in order to

explore their individual characteristics and improve environment perception. Darms et al. [DAR09] present the sensor fusion-based method for detection and tracking of moving vehicles adopted by the self-driving car "Boss" [URM08] (Carnegie Mellon University's car that finished in first place in the 2007 DARPA Urban Challenge). The MOT subsystem is divided into two layers. The sensor layer extracts features from sensor data that may be used to describe a moving obstacle hypothesis according to either a point model or a box model. The sensor layer also attempts to associate features with currently predicted hypotheses from the fusion layer. Features that cannot be associated to an existing hypothesis are used to generate new proposals. An observation is generated for each feature associated with a given hypothesis, encapsulating all information that is necessary to update the estimation of the hypothesis state. Based on proposals and observations provided by the sensor layer, the fusion layer selects the best tracking model for each hypothesis and estimates (or updates the estimation of) the hypothesis state using a Kalman Filter. Cho et al. [CHO14] describe the new MOT subsystem used by the new experimental autonomous vehicle of the Carnegie Mellon University. The previous MOT subsystem, presented by Darms et al. [DAR09], was extended for exploiting camera data, in order to identify categories of moving objects (e.g., car, pedestrian, and bicyclists) and to enhance measurements from automotive-grade active sensors, such as LIDARs and RADARs. Mertz et al. [MER13] use scan lines that can be directly obtained from 2D LIDARs, from the projection of 3D LIDARs onto a 2D plane, or from the fusion of multiple sensors (LADAR, RADAR, and camera). Scan lines are transformed into world coordinates and segmented. Line and corner features are extracted for each segment. Segments are associated with existing obstacles and kinematics of objects are updated using a Kalman filter. Byun et al. [BYU15] merge tracks of moving obstacles generated from multiple sensors, such as RADARs, 2D LIDARs, and a 3D LIDAR. 2D LIDAR data is projected onto a 2D plane and moving obstacles are tracked using Joint Probabilistic Data Association Filter (JPDAF). 3D LIDAR data is projected onto an image and partitioned into moving obstacles using a region growing algorithm. Finally, poses of tracks are estimated or updated using Iterative Closest Points (ICP) matching or image-based data association. Xu et al. [XU15] describe the context-aware tracking of moving obstacles for distance keeping used by the new experimental driverless car of the Carnegie Mellon University. Given the behavioral context, a ROI is generated in the road network. Candidate targets inside the ROI are found and projected into road coordinates. The distance-keeping target is obtained by associating all candidate targets from different sensors (LIDAR, RADAR, and camera). Xue et al. [XUE17] fuse LIDAR and camera data to improve the accuracy of pedestrian detection. They use prior knowledge of a pedestrian height to reduce false detections. They estimate the height of the pedestrian according to pinhole camera equation, which combines camera and LIDAR measurements.

6) Deep Learning Based MOT

Deep learning based methods use deep neural networks for detecting positions and geometries of moving obstacles, and tracking their future states based on current camera data. Huval et al. [HUV15] propose a neural-based method for detection of moving vehicles using the Overfeat Convolutional Neural Network (CNN) [SER13] and monocular input images with focus on real-time performance. CNN aims at predicting location and range distance (depth) of cars in the same driving direction of the ego-vehicle using only the rear view of them. Mutz et al. [MUT17] address moving obstacle tracking for a closely related application known as "follow the leader", which is relevant mainly for convoys of autonomous vehicles. The tracking method is built on top of the Generic Object Tracking Using Regression Networks (GOTURN) [HEL16]. GOTURN is a pre-trained deep neural network capable of tracking generic objects without further training or object-specific fine-tuning. Initially, GOTURN receives as input an image and a manually delimited bounding box of the leader vehicle. It is assumed that the object of interest is in the center of the bounding box. Subsequently, for every new image, GOTURN gives as output an estimate of the position and geometry (height and width) of the bounding box. The leader vehicle position is estimated using LIDAR points that fall inside the bounding box and are considered to be vehicle.

E. Traffic Signalization Detection and Recognition

The traffic signalization detection and recognition subsystem is responsible for detecting and recognizing signs defined in the traffic rules so that the car can take correct decisions according to the traffic law. There are many tasks related to traffic signalization, and in this review, we explore three main topics: traffic lights, traffic signs, and pavement markings in the environment around the self-driving car. Each of these topics are described in detail in the next subsections.

1) Traffic Light Detection and Recognition

Traffic light detection and recognition involve detecting the position of one or more traffic lights in the environment around the car (e.g., represented in an image) and recognizing their states (red, green, and yellow). Various methods for traffic light detection and recognition have been proposed in the literature. Here, we review only the most recent and relevant ones. For a more comprehensive review, readers are referred to Jensen et al. [JEN16].

Methods for traffic light detection and recognition can be mainly categorized into two classes: model-based and learning-based. Traffic lights have a well-defined structure in terms of color and shape information. A common traffic light has three bulbs (one for each state: red, green and yellow) and a well-defined form. Therefore, earlier, most of the approaches for traffic light detection and recognition were model-based. These approaches relied on hand-crafted feature engineering, which tried to leverage information humans have about the color and shape of the object to build a model capable of detecting and/or recognizing it. Methods that used color

[CAB12] [CAB15] and shape [OMA10] [TRE14] [SOO14] information were not robust when assumptions were not strictly observed. To increase their robustness, a combination of different features (e.g., color, shape, and structure) was proposed [KOU12] [ZHA14] [GOM14]. For example, in [ZHA14], the authors proposed a multi-feature system that combines both color (using color segmentation), shape/structure (using black box detection), and geographic information (by only using the system when known traffic lights are expected). Their system, however, suffer from the high number of hyper-parameters common on model-based approaches, which usually implicates the need of recalibration under certain circumstances. The authors performed the experiments on an in-house private data set and stated that failures were due to over-exposure, occlusions, non-standard installation of traffic lights, and several other situations that are not unusual in real-world cases. This combination, in the context of model-based approaches, showed not to be enough. Therefore, researchers began to introduce learning-based approaches.

In learning-based approaches, features were still hand-crafted, but detection and/or recognition processes were changed from rule-based to learning-based. Cascade classifiers [LIN04] were probably the first attempt to learning-based approaches. Eventually, popular combinations of HoG and Gabor features with classifiers (such as SVM [JAN14], AdaBoost [GON10], and JointBoost [HAL15]) were also investigated. More recently, end-to-end methods (i.e., without the need of hand-crafted features) outperformed most of the model-based ones. In John et al. [JOH14], GPS data and a traffic light location database were used to identify a region of interest in the image, and a convolutional neural network (CNN) was employed to recognize the traffic light state. Furthermore, state-of-the-art general object detectors [REN17] [LIU16] [RED17] were successfully applied to the detection of traffic lights (often without recognizing their states). These general-purpose deep object detectors (or simply deep detectors, as they are often called), comprehensively, do not provide a breakdown on the performance of the traffic light detection and recognition task. Even though, unlike the model-based approaches, these deep detectors tend to be more robust to over-exposure, color distortions, occlusions, and others. A more complete discussion of these deep detectors applied to the traffic light detection can be found in Jensen et al. [JEN17]. There, the authors apply the YOLO [RED17] on the LISA [BOR15] dataset and achieve 90.49\% AUC when using LISA's training set. However, the performance drops to 58.3% when using training data from another dataset. Despite of the fact, it is still an improvement over previous methods, and it demonstrates that there is still a lot to be done. Learning-based approaches, especially those using deep learning, require large amounts of annotated data. Only recently large databases with annotated traffic lights are being made publicly available, enabling and powering learning-based approaches. Nowadays, the most common databases are LaRA [LAR11b] (11,179 frames), LISA (43,007 frames), Bosch Small Traffic Lights [BEH17] (13,427 frames), BDD [XIA18] (100,000 frames),

and Udacity [UDA18] (13,063 frames). Note that some of these datasets report the total number of frames including the ones with background only.

Despite of the advances on traffic light detection and recognition research, little is known about what is being used by research self-driving cars. Perhaps, one of the main reasons for this is that there were no traffic lights in the 2007 DARPA Urban Challenge. First place and second place finishers of this challenge (the Carnegie Mellon University's team with their car "Boss" [URM08] and the Stanford University's team with their car "Junior" [MON08], respectively) recognized that traffic lights contribute to the complexity of urban environments and that they were unable to handle them at that time. In 2010, the "Stadtpilot" project's presented their autonomous vehicle "Leonie" [NOT11] on public roads of Braunschweig, Germany. Leonie used information about traffic light positions from a map and Car-to-X (C2X) communication to recognize traffic light states. However, during demonstrations a co-driver had to enter the traffic light state when C2X was not available. In 2013, the Carnegie Mellon University tested their driverless car [WEI13] on public roads for over a year. The Carnegie Mellon's car used cameras to detect traffic lights and employed vehicle-to-infrastructure (V2I) communication to retrieve information from DSRC-equipped traffic lights. In 2013, Mercedes-Benz tested their robotic car "Bertha" [ZIE14a] on the historic Bertha-Benz-Memorial-Route in Germany. "Bertha" used vision sensors and prior (manual) information [LIN04] to detect traffic lights and recognize their states. However, they state that the recognition rate needs to be improved for traffic lights at distances above 50m.

2) Traffic Sign Detection and Recognition

Traffic sign detection and recognition involve detecting the locations of traffic signs in the environment and recognizing their categories (e.g., speed limit, stop, and yield sign.). For reviews on methods for traffic sign detection and recognition, readers are referred to Mogelmose et al. [MOG12] and Gudigar et al. [GUD16].

Earlier, most of the approaches for traffic sign detection and recognition were model-based [GAO06, BAR08] and used simple features (e.g., colors, shapes, and edges). Later, learning-based approaches (such as SVM [LAF10], cascade classifiers [PET08], and LogitBoost [OVE11]) started leveraging simple features, but evolved into using more complex ones (e.g., patterns, appearance, and templates). However, these approaches commonly tended to not generalize well. In addition, they usually needed fine-tuning of several hyper-parameters. Furthermore, some methods worked only with recognition and not with detection, perhaps because of the lack of data. Only after large databases were made available (such as the well-known German Traffic Sign Recognition (GTSRB) [STA12] and Detection (GTSDB) [HOU13] Benchmarks, with 51,839 and 900 frames, respectively) that learning-based approaches [MAT13] [HOU13] could finally show their power, although some of them were able to cope with fewer examples [SOU13a]. With

the release of even larger databases (such as STSD [LAR11] with over 20,000 frames, LISA [BOR15] with 6,610 frames, BTS [MAT13] 25,634 frames for detection and 7,125 frames for classification, and Tsinghua-Tencent 100K [ZHU16] with 100,000 frames), learning-based approaches improved and achieved far better results when compared to their model-based counterparts. Some of these datasets report the number of frames including frames with background only. Following the rise of deep learning in general computer vision tasks, convolutional neural networks [ZHU16] are the state-of-the-art for traffic sign detection and recognition, achieving up to 99.71% and 98.86% of F1 score on the recognition task for the GTSRB and BTS respectively.

Once more, little can be said about what is being employed for traffic sign detection and recognition by research self-driving cars. Again, maybe, one of the main drivers *behind* this is that only the stop-sign had to be detected and recognized in the 2007 DARPA Urban Challenge, since the map had detailed information on speed limits and intersection handling [THR10]. Some researchers (such as those of "Bertha" [ZIE14a]) still prefer to rely on annotations about speed limit, right-of-way, among other signs. Other researchers [FER14] stated that their driverless cars can handle traffic signs but did not provide information about their method.

3) Pavement Marking Detection and Recognition

Pavement marking detection and recognition involve detecting the positions of pavement marking and recognizing their types (e.g., lane markings, road markings, messages, and crosswalks). Most researches deal with only one type of pavement marking at a time and not with all of them at the same time. This may happen because there is neither a widely used database nor a consensus on which set of symbols researchers should be focused on when dealing with pavement marking detection and recognition.

One important pavement marking is the lane definition in the road. Earlier, most of the methods for lane marking detection were model- or learning-based [MCC06]. Shape and color were the most usual features, and straight and curved lines (e.g., parabolic [JUN05] and splines [BER15, BER17c]) were the most common lane representations. In [BER17c], the authors propose a complete system for performing ego-lane analysis. Among the features of the systems, the authors claim to be able of detecting lanes and their attributes, crosswalks, lane changing events, and some pavement markings. The authors also release datasets for evaluating these types of systems. Deep learning is another popular method that have gained popularity lately and approaches like [GUR16] have become showed very good results. In [GUR16], the authors propose (i) to use two laterally-mounted down-facing cameras and (ii) to model the lateral distance estimation as a classification problem in which they employ a CNN to tackle the task. In this setting, they argue to achieve sub-centimeter accuracy with less than 2 pixels of Mean Absolute Error (MAE) in a private database. For a review on this type of methods, readers are referred to Hillel et al. [HIL14].

Many of the methods for lane marking detection were also attempted for road marking detection. They commonly use geometric and photometric features [WU12]. Furthermore, various methods for road marking detection and recognition use Inverse Perspective Mapping (IPM), which reduces perspective effect and, consequently, makes the problem easier to solve and improves the accuracy of results. More recently, several methods [LEE17] [BAI17] [AHM17] employed Maximally Stable Extremal Regions (MSER) for detecting regions of interest (i.e., regions that are likely to contain a road marking) and convolutional networks for recognizing road markings. In [BAI17], the authors propose the combination of IPM, MSER, and DBSCAN-based algorithm to perform the detection of the road markings and the combination of PCANet with either SVM or linear regression for the classification. While they achieve up to 99.1% of accuracy when evaluating the classification task alone, it drops to 93.1% of accuracy when the performance of detection and recognition is reported together.

In the context of road markings, messages are often handled separately. Some methods for message detection and recognition [AHM17] treat different messages as different categories (i.e., they firstly detect positions of messages in the scene and then recognize their categories), while most of the methods firstly recognize letters and then writings using OCR-based approaches [HYE16] [GRE15]. The former is usually more robust to weather and lighting conditions, but the latter can recognize unseen messages.

Still in the setting of road markings, pedestrian crossings are often investigated separately. Most of the methods for crosswalk detection exploit the regular shape and black-and-white pattern that crosswalks usually present [IVA08] [FOU11]. Therefore, in many practical applications, this task is set aside in favor of robust pedestrian detectors. For a review on these methods, readers are referred to Berriel et al. [BER17b]. Together with the review, Berriel et al., in [BER17b], present a deep learning-based system to detect the presence of crosswalks in images. The authors provide pre-trained models that can be directly applied for this task.

IV. DECISION MAKING

In this section, we present investigations on relevant techniques reported in the literature for the decision-making system of self-driving cars, comprising the route planning, behavior selection, motion planning, and control subsystems.

A. *Route Planning*

The route planning subsystem is responsible for computing a route through the road network from the self-driving car's initial position to the final position defined by a user operator. If the road network is represented by a weighted directed graph, whose edge weights denote the cost of traversing a road segment, then the problem of computing a route can be reduced to the problem of finding the shortest path in a weighted directed graph. However, for large road networks,

complexity times of classical shortest path algorithms, such as Dijkstra [DIJ59] and A* [HAR68], are impractical.

In the last decade, there have been significant advances in the performance of algorithms for route planning in road networks. Newly developed algorithms can compute driving directions in milliseconds or less, even at continental scales. For a review on algorithms for route planning in road networks that are applicable to self-driving cars, readers are referred to Bast et al. [BAS15].

Methods for route planning in road networks provide different trade-offs in terms of query time, preprocessing time, space usage, and robustness to input changes, among other factors. They can be mainly categorized into four classes [BAS15]: goal-directed, separator-based, hierarchical, bounded-hop, and combinations.

1) Goal-Directed Techniques

Goal-directed techniques guide the search from the source vertex to the target vertex by avoiding scans of vertices that are not in the direction of the target vertex. A* is a classic goal-directed shortest path algorithm. It achieves better performance than the Dijkstra's algorithm by using a lower-bound distance function on each vertex, which causes vertices that are closer to the target to be scanned earlier. ALT (A*, landmarks, and triangle inequality) algorithm [GOL05] enhances A* by picking a small set of vertices as landmarks. During the preprocessing phase, distances between all landmarks and all vertices are computed. During the query phase, a valid lower-bound distance for any vertex is estimated, using triangle inequalities involving landmarks. The query performance and correctness depend on the wise choice of vertices as landmarks.

Another goal-directed algorithm is Arc Flags [HIL09]. During the preprocessing phase, the graph is partitioned into cells with a small number of boundary vertices and a balanced (i.e., similar) number of vertices. Arc flags for a cell i are computed by growing backwards a shortest path tree from each boundary vertex, setting the i-th flag for all arcs (or edges) of the tree. During the query phase, the algorithm prunes edges that do not have the flag set for the cell that contains the target vertex. The arc flags method has high preprocessing times, but the fastest query times among goal-directed techniques.

2) Separator-Based Techniques

Separator-based techniques are based on either vertex or edge separators. A vertex (or edge) separator is a small subset of vertices (or edges) whose removal decomposes the graph into several balanced cells. The vertex separator-based algorithm uses vertex separators to compute an overlay graph. Shortcut edges are added to the overlay graph such that distances between any pair of vertices from the full graph are preserved. The overlay graph is much smaller than the full graph and is used to accelerate the query algorithm. HPML (*High-Performance Multilevel Routing*) algorithm [DEL09] is a variant of this approach that significantly reduces query time, but at the cost of increasing space usage and preprocessing time, by adding many more shortcuts to the graph across different levels.

The arc separator-based algorithm uses edge separators to decompose the graph into balanced cells, attempting to minimize the number of cut edges, which connect boundary vertices of different cells. Shortcuts are added to the overlay graph in order to preserve distances between boundary vertices within each cell. CRP (*Customizable Route Planning*) algorithm [DEL15] is designed to meet requirements of real-world road networks, such as handling turn costs and performing fast updates of the cost function. Its preprocessing has two phases. The first phase computes the multilevel partition and the topology of the overlays. The second phase computes the costs of clique edges by processing the cells bottom-up and in parallel. Queries are processed as bidirectional searches in the overlay graph.

3) Hierarchical Techniques

Hierarchical techniques exploit the inherent hierarchy of road networks, in which main roads such as highways compound a small arterial subnetwork. Once the source and target vertices are distant, the query algorithm only scans vertices of the subnetwork. The preprocessing phase computes the importance of vertices or edges according to the actual shortest path structure. CH (*Contraction Hierarchies*) algorithm [GEI12] is a hierarchical technique that implements the idea of creating shortcuts to skip vertices with low importance. It repeatedly executes vertex contraction operations, which remove from the graph the least important vertex and create shortcuts between each pair of neighboring vertices, if the shortest path between them is unique and contains the vertex to be contracted. CH is versatile, thus serving as a building block for other point-to-point algorithms and extended queries.

REACH algorithm [GUT04] is a hierarchical technique that, during the preprocessing phase, computes centrality measures (reach values) on vertices and, during query phase, uses them to prune Dijkstra-based bidirectional searches. Let P be a shortest path from the source vertex s to the target vertex t that contains vertex v. The reach value of v with respect to P is $r(v, P) = \min\{distance(s, v), distance(v, t)\}$.

4) Bounded-Hop Techniques

Bounded-hop techniques precompute distances between pairs of vertices by adding virtual shortcuts to the graph. Since precomputing distances among all pairs of vertices is prohibitive for large networks, bounded-hop techniques aim to get the length of any virtual path with very few hops. A bounded-hop algorithm is HL (*Hub Labeling*) [COH03], which, during the preprocessing phase, computes a label $L(u)$ for each vertex u of the graph, which consists of a set of hub vertices of u and their distances from u. These labels are chosen such that they obey the cover property: for any pair (s, t) of vertices, the intersection of labels $L(s)$ and $L(t)$ must contain at least one vertex of the shortest path from s to t. During the query phase, the distance (s, t) can be determined in linear time by evaluating the distances between hub vertices

present in the intersection of labels $L(s)$ and $L(t)$. HL has the fastest known queries for road networks, at the cost of high space usage. HL-∞ algorithm [ABR12] exploited the relationship between hub labelings and vertex orderings, and developed preprocessing algorithms for computing orderings that yield small labels. The iterative range optimization algorithm for vertex ordering makes the query time of HL-∞ algorithm twice as fast as HL. It starts with some vertex ordering (e.g., the one given by CH) and proceeds in a given number of iterative steps, each reordering a different range of vertices in decreasing order of importance. HLC (*Hub Label Compression*) algorithm [DEL13a] reduces space usage by an order of magnitude, at the cost of higher query times, by combining common substructures that appear in multiple labels,

Another bounded-hop algorithm is TNR (*Transit Node Routing*) [ARZ13], which uses distance tables on a subset of the vertices. During the preprocessing phase, it selects a small set of vertices as transit nodes and computes all pairwise distances between them. From transit nodes, for each vertex u, it computes a set of access nodes. If there is a shortest path from u, such that v is the first transit node in it, then the transit node v is an access node of u. It also computes distances between each vertex and its access nodes. A natural approach for selecting the transit node set is to select vertex separators or boundary vertices of arc separators as transit nodes. During query phase, the distance table is used to select the path from the source vertex s to the target vertex t that minimizes the combined distance s–$a(s)$–$a(t)$ –t, where $a(s)$ and $a(t)$ are access nodes. If the shortest path does not contain a transit node, then a local query is performed (typically CH).

5) Combinations

Individual techniques can be combined into hybrid algorithms that exploit different graph properties. REAL algorithm [GOL09] combines REACH and ALT. ReachFlags algorithm [BAU10] combines REACH and Arc Flags. SHARC algorithm [BAU09] combines the computation of shortcuts with multilevel Arc Flags. CHASE algorithm [BAU10] combines CH with Arc Flags. TNR+AF algorithm [BAU10] combines TNR and Arc Flags. PHAST algorithm [DEL13b] can be combined with several techniques in order to accelerate them by exploiting parallelism of multiple core CPUs and GPUs.

Bast et al. [BAS15] evaluated experimentally many of the route planning techniques described here, using the well-known continent-sized benchmark Western Europe and a simplified model of real-world road networks. Table I shows the results of their experimental analysis. For each technique, Table 1 presents the total memory space usage, total preprocessing time, number of vertices scanned by a query on average, and average query time.

B. *Motion Planning*

The motion planning subsystem is responsible for computing a path or trajectory from the self-driving car's current state to the next local goal state defined by the behavior selection subsystem. The motion plan executes the local driving behavior, satisfies car's kinematic and dynamic constraints, provides comfort to the passenger, and avoids collisions with static and moving obstacles in the environment.

TABLE I.   PERFORMANCE OF ROUTE PLANNING TECHNIQUES ON WESTERN EUROPE [BAS15]

Algorithm	Data Structures		Queries	
	Space (GB)	Time (h:m)	Scanned Vertices	Time (µs)
Dijkstra	0.4	–	9,326,696	2,195,080
Bidirect. Dijkstra	0.4	–	4,914,804	1,205,660
CRP	0.9	1:00	2,766	1,650
Arc Flags	0.6	0:20	2,646	408
CH	0.4	0:05	280	110
CHASE	0.6	0:30	28	5.76
HLC	1.8	0:50	–	2.55
TNR	2.5	0:22	–	2.09
TNR+AF	5.4	1:24	–	0.70
HL	18.8	0:37	–	0.56
HL-∞	17.7	60:00	–	0.25
Table Lookup	1,208,358	145:30	–	0.06

A motion plan can be a path or a trajectory. The path is a sequence of car's states that does not define how car's states evolve over time. This task can be entrusted to other subsystem (e.g., the behavior selection subsystem) or a velocity profile can be defined as a function of curvature and proximity to obstacles. The trajectory is a path that specifies the evolution of car's states through time.

Various methods for motion planning have been proposed in the literature. We review those that were designed for on-road motion planning[1] and evaluated experimentally using real-world autonomous vehicles. For more comprehensive reviews on these methods, readers are referred to González et al. [GON16] and Paden et al. [PAD16].

1) Path Planning

Path planning involves generating a sequence of states from the car's current state to the next goal state, which does not define the evolution of car's states over time. Path planning is usually categorized into global and local path planning [KAM04]. In global path planning, a global path is computed before the car starts moving using an offline global map of the environment. In local path planning, a local path is generated while the car is moving using online local maps of the surroundings, which allows the car to deal with moving obstacles. Methods for path planning can be mainly categorized into two classes: graph search based and interpolating curve based [GON16, PAD16].

---

[1] On-road motion planning aims at planning a path or trajectory that follows a desired lane, which differs from unstructured motion planning, in which there are no lanes and, thus, the trajectory is far less constrained.

Graph search based techniques searches for the best path (sequence of states) between the car's current state and next goal state in a state space represented as a graph. They represent the search space as a graph. They discretize the search space and imposing a graph on an occupancy grid map with centers of cells acting as neighbors in the search graph. The most common graph search based techniques for path planning of self-driving cars are Dijkstra, A-star, and A-star variants.

The Dijkstra algorithm [DIJ59] finds the shortest path between the initial node and goal node of a graph. Dijkstra repeatedly examines the closest not yet examined node, adding its neighbors to the set of nodes to be examined, and stops when the goal node is attained. Dijkstra is suitable for global path planning. However, it has high computational cost in vast areas, because of the large number of nodes inspected, and its result is not continuous [GON16]. Arnay et al. [ARN16] use Dijkstra to generate a global path that is extended for building a trajectory for the autonomous vehicle "Verdino". Bacha et al. [BAC08] employ Dijkstra to construct a global path for the driverless car Odin [BAC08] to navigate toward and reversing out of a parking spot. Kala et al. [KAL13] use Dijkstra to generate global and local paths, which were tested only in computer simulations.

The A-star algorithm [HAR68] is an extension of Dijkstra, which performs fast graph search by assigning weights to nodes based on a heuristic estimated cost to the goal node. Nevertheless, it is not easy to find a solution [GON16]. Leedy et al. [LEE07] employ the A-star algorithm to build a local path for the driverless car "Rocky", which participated in the 2005 DARPA Grand Challenge. Ziegler et al. [ZIE08] propose a method for local path planning that combines A-star with two different heuristic cost functions, namely rotation-translation-rotation (RTR) metric and Voronoi. The first one accounts for kinematics constraints of the car, while the second one incorporates knowledge of shapes and positions of obstacles. The method was tested in the robotic car "AnnieWAY", which participated of the 2007 DARPA Urban Challenge.

Other authors propose variations of the A-star algorithm for path planning. Urmson et al. [URM08] propose the anytime D-star for the driverless car "Boss" (Carnegie Mellon University's car that claimed first place in the 2007 DARPA Urban Challenge). Dolgov et al. [DOL10] propose the hybrid-state A-star for the robotic car "Junior" [MON08] (Stanford University's car that finished in second place in the 2007 DARPA Urban Challenge). Both anytime D-star and hybrid-state A-star algorithms merge two heuristics – one non-holonomic, which disregards obstacles, and the other holonomic, which considers obstacles – and were used for path planning in an unstructured environment (parking lot). Chu et al. [CHU15] propose a variation of A-star to build a local path that considers car's kinematic constraints, which ignores the resolution of grid cells and creates a smooth path. Yoon et al. [YOO15] propose a variation of A-star that

accounts for kinematic constraints of the self-driving car "Kaist".

b) *Interpolating Curve Based Techniques*

Interpolating curve based techniques use the interpolation process, by which a new set of points is inserted in the range of a previously known set of points. They take a previously known set of points (e.g., waypoints that describes a road-map) and generate a new set of points that depicts a smoother path. The most usual interpolating curve based techniques for path planning of self-driving cars are spline curves.

A spline curve is a piecewise polynomial parametric curve divided in sub-intervals that can be defined as polynomial curves. The junction between each sub-segment is called knot (or control points), which commonly possess a high degree of smoothness constraint. This kind of curve has low computational cost, because its behavior is defined by the knots. However, its result might not be optimal, because it focuses more on achieving continuity between the parts instead of meeting road's constraints, and it depends on global waypoints [GON16]. Chu et al. [CHU12] and Hu et al. [HU18] use cubic spline curves for path planning. Both of them construct a center line from a set of waypoints obtained from a lane map. They generate a series of parametric cubic splines, which represent possible path candidates, using the arc length and offset to the center line. The optimal path is selected based on the weighted sum of function costs. A difference between these two approaches is that the first avoids only static obstacles, while the second avoids static and moving obstacles. Another difference is that the first was tested in an autonomous vehicle, the "A1", which won the Autonomous Vehicle Competition in 2010 at Korea, while the second was evaluated by computer simulations that modeled challenging scenarios, including single-lane and multi-lane roads with static and moving obstacles.

2) *Trajectory Planning*

Trajectory planning involves generating a sequence of states from the self-driving car's current state to the next goal state, which specifies the evolution of car's states through time. Methods for trajectory planning can be mainly categorized into four classes: graph search based, sampling based, interpolating curve based, and numerical optimization based [GON16, PAD16].

a) *Graph Search Based Techniques*

Graph search based techniques for trajectory planning extend those for path planning (Section IV.B.1)) in order to specify the evolution of car's states over time. The most common graph search based trajectory planning techniques of self-driving cars are state lattice, elastic-band (EB), and A-star.

A state lattice is a search graph in which vertices denote states and edges denote paths that connect states satisfying robot's kinematic constraints. Vertices are placed in a regular fashion such that the same paths can be used to connect all vertices. In this way, a path to the goal is likely to be

represented as a sequence of edges in the graph. This state lattice must be adapted for on-road path planning by representing only states that are likely to be in the solution. Furthermore, this state lattice must be extended to dynamic environments by adding time and velocity dimensions. State lattices are able to handle several dimensions, such as position, velocity, and acceleration, and suitable for local planning and dynamic environments. However, they have high computational cost, because it evaluates every possible solution in the graph [GON16]. McNaughton et al. [MCN11] propose a conformal spatiotemporal state lattice for on-road trajectory planning. They construct the state lattice around a centerline path. They define nodes on the road at a lateral offset from the centerline and compute edges between nodes using an optimization algorithm. This optimization algorithm finds the parameters of a polynomial function that define edges connecting any pairs of nodes. They assign to each node a state vector that contains a pose, acceleration profile, and ranges of times and velocities. The acceleration profile increases trajectory diversity at less cost than would the finer discretization of time and velocity intervals. Furthermore, the ranges of times and velocities reduce computational cost by allowing the assignment of times and velocities to the graph search phase, instead of graph construction phase. Xu et al. [XU12] propose an iterative optimization that is applied to the resultant trajectory derived from the state lattice, which reduces the planning time and improve the trajectory quality. Gu et al. [GU16] propose a planning method that fuses a state lattice trajectory planning with behavior selection. A set of candidate trajectories is sampled, from which different behaviors are extracted. The final trajectory is obtained by selecting a behavior and choosing the candidate trajectory associated with the selected behavior. Li et al. [LI16] generate candidate paths along a global path using a cubic polynomial curve. A velocity profile is also computed to be assigned to points of the generated paths. The generated trajectories are evaluated by a cost function and the optimal trajectory is selected.

Optimization based elastic-band techniques for path planning methods represent the state space by a graph with elastic nodes and edges. An elastic node is defined by augmenting the spatial node with an in edge and out edge that connect the neighboring spatial nodes. The path is found by an optimization algorithm that balances two forces – repulsive forces generated by external obstacles and contractive forces generated by the neighboring points to remove band slackness. This method demonstrates continuities and stability, it has non-deterministic run-time and requires a collision-free initial path.

Gu et al. [GU15] propose a decoupled space-time trajectory planning method that performs path planning and trajectory planning separately. The trajectory planning is divided into three phases. In the first phase, a collision-free path is computed considering road and obstacles constraints, and feasible path is generated using a pure-pursuit controller and kinematic car model. In the second phase, a velocity profile is suggested under several constraints, namely speed limit,

obstacle proximity, lateral acceleration and longitudinal acceleration. Finally, given the path and velocity profile, trajectories are computed using parametric path spirals. Trajectories are evaluated against all static and moving obstacles by simulating their future movements.

The A-star algorithm is commonly used for path planning (Section IV.B.1)) or unstructured trajectory planning. Fassbender et al. [FAS16] propose two novel node expansion schemes to A-star for on-road trajectory planning. The first scheme tries to find a trajectory that connects the car's current node directly to the goal node using numerical optimization. The second scheme uses a pure-pursuit controller to generate short edges (i.e., short motion primitives) that guide the car along the global reference path.

b)   *Sampling Based Techniques*

Sampling based techniques randomly sample the state space looking for a connection between the car's current state and the next goal state. The most usual sampling based techniques for trajectory planning of self-driving cars are rapidly-exploring random tree (RRT).

RRT methods for trajectory generation [LAV01] incrementally build a search tree from the car's current state using random samples from the state space. For each random state, a control command is applied to the nearest vertex of the tree in order to create a new state as close as possible to the random state. Each vertex of the tree represents a state and each directed edge represents a command that was applied to extend a state. Candidate trajectories are evaluated according to various criteria. RRT methods have low computational cost for high-dimensional spaces and always find a solution, if there is one and it is given enough time. However, its result is not continuous and jerky [GON16]. Radaelli et al. [RAD14] propose a RRT method for trajectory planning of the autonomous vehicle "IARA". They present new variants to the standard RRT in order to bias the location of random states toward the lane region, select the most promising control commands to extend states, choose the best trajectories, discard non-promising states, and reuse part of the trajectory built in the previous planning cycle. Du et al. [DU16] propose a RRT method that uses the drivers' visual search behavior on roads to guide the state sampling of RRT. Drivers use both a "near point" and a "far point" to drive through curved roads. They employ this characteristic of drivers' visual search behavior shown on curved roads to guide the RRT method. Furthermore, they adopt a post-processing based on B-spline to generate smooth, continuous, and feasible trajectories.

c)   *Interpolating Curve Based Techniques*

Interpolating curve based techniques interpolate a previously known set of points (e.g., road-map waypoints) and build a smoother trajectory that considers car's kinematic and dynamic constraints, comfort, obstacles, among other parameters. The most common interpolating curve based techniques for trajectory planning of self-driving cars are clothoid curves.

A clothoid curve allows defining trajectories with linear variable curvature, so that transitions between straight to curved segments are smooth. However, a clothoid curve has

high computational cost, because of the integrals that define it, and it depends on global waypoints [GON16]. Alia et al. [ALI15] use clothoid tentacles for trajectory planning. Tentacles are computed for different speeds and different initial steering angles, starting from the car's center of gravity and taking the form of clothoids. Tentacles are classified as navigable or not navigable using an occupancy grid map. Among the navigable tentacles, the best tentacle is chosen based on several criteria. Mouhagir et al. [MOU16] [MOU17] also use clothoid tentacles for trajectory planning. However, they use an approach inspired in Markov Decision Process to choose the best tentacle.

d)   Numerical Optimization Based Techniques

Numerical optimization based techniques minimize or maximize a function with constrained variables. The most common numerical optimization based techniques for trajectory planning of self-driving cars are function optimization and model-predictive methods.

Function optimization methods find a trajectory by minimizing a cost function that considers trajectory's constraints, such as position, velocity, acceleration, and jerk. These methods can easily consider car's kinematic and dynamic constraints and environment's constraints. However, they have high computational cost, because the optimization process takes place at each motion state, and depend on global waypoints [GON16]. Ziegler et al. [ZIE14b] uses a function optimization method for trajectory planning of the self-driving car "Bertha". They find the optimal trajectory that minimizes a cost function and obeys trajectory's constraints. The cost function is composed of a set of terms that make the trajectory to follow the middle of the driving corridor at a specified velocity, penalize strong acceleration, dampen rapid changes in acceleration, and attenuate high yaw rates.

Model predictive methods for trajectory planning [HOW07] produce dynamically feasible control commands between car's current state and next goal state. They can be used to solve the problem of generating parameterized control commands that satisfy state constraints whose dynamics can be expressed by differential equations. Ferguson et al. [FER08] uses a model-predictive method for trajectory planning of the autonomous vehicle "Boss" [URM08] (Carnegie Mellon University's car that claimed first place in the 2007 DARPA Urban Challenge). They generate trajectories to a set of goal states derived from the centerline path. To compute each trajectory, they use an optimization algorithm that gradually modifies an initial approximation of trajectory control parameters until the trajectory end point error is within an acceptable bound. The trajectory control parameters are the trajectory length and three knot points of a spline curve that defines the curvature profile. A velocity profile is generated for each trajectory based on several factors, including the velocity limit of the current road, maximum feasible velocity, and goal state velocity. The best trajectory is selected according to their proximity to obstacles, distance to the centerline path, smoothness, end point error, and velocity error. Li et al. [LI17] use a state sampling-based

trajectory planning scheme that samples goal states along a global reference path. A model-predictive path planning method is applied to produce paths that connect the car's current state to the sampled goal states. A velocity profile is used to assign a velocity to each of the states along generated paths. A cost function that considers safety and comfort is employed to select the best trajectory. Cardoso et al. [CAR17] use a model-predictive method for trajectory planning of the autonomous vehicle "IARA". To compute the trajectory, they use an optimization algorithm that finds the trajectory control parameters that minimize the distance to the goal state, distance to the centerline path, and proximity to obstacles. The trajectory control parameters are the trajectory time and four knot points that define a spline curve which specifies the steering angle profile.

C. Control

In the field of self-driving cars, control refers to the theory behind automatic control of the engineering area, which covers the application of mechanisms to operate and regulate processes without continuous direct human intervention. In the simplest type of automatic control, a control subsystem compares the process's output with a desired input and uses the error (difference between the process's output and the desired input) to change the process's input, such that the process stays at its set point despite disturbances. In autonomous vehicles, the automatic control theory is generally applied to path tracking and hardware actuation methods. The role of a path tracking method is to stabilize the execution of the motion plan in the presence of inaccuracies in the car's model, among others. The role of the hardware actuation control is to compute steering, throttle and brake actuator inputs that execute the motion plan in the presence of inaccuracies in actuators' models and others.

Path tracking methods are also referred to as control techniques, because they employ the automatic control theory and consider the path as the signal to be controlled. However, in the area of autonomous vehicles, it is more appropriate to refer to them as path tracking methods, in order to discern them from hardware actuation control methods. Both of them are detailed in the next subsections.

1) Path Tracking Methods

Path tracking methods stabilize (i.e., reduce the error in) the execution of the motion plan computed by the motion planning subsystem (Section IV.B), in order to reduce inaccuracies caused mainly by car's motion model. They can be considered simplified trajectory planning techniques, although they do not deal with obstacles. The pure pursuit method [PAD16] is broadly used in self-driving cars for path tracking, because of its simple implementation. It consists of finding a point in the path at some look-ahead distance from the current path and turning the front wheel so that a circular arc connects the center of the rear axle with the point in the path [PAD16], as shown in Figure 6. There is a set of variants proposed to improve the pure pursuit method. Samson et al.

[SAM95] propose to use the rear wheel position as the hardware output to stabilize the execution of the motion plan. Thrun et al. [THR07] present the control approach of the autonomous vehicle Stanley, which consists of taking the front wheel position as the regulated variable. Stanley's control subsystem was able to drive the car at up to 60 km/h.

The Model Predictive Control (MPC) method is widely employed in driverless cars. It consists of selecting control command inputs that will lead to desired hardware outputs, using the car's motion model to simulate and optimize outputs over a prediction horizon in the future [GUI17]. Koga et al. [KOG13] use the MPC method for path tracking of a small electrical car. They employ the standard bicycle model to predict car's motion. The electrical car's control subsystem was able to drive the car at up to 20 km/h. Kritayakirana and Gerdes [KRI12] present the control approach of an Audi TTS, in which a dynamic motion model of the car [BRO17] [LAU17] is used to enable path tracking at the limits of handling. The car's dynamic motion model considers the deviation from the path plan due to tire slippage. Experimental results demonstrated that the Audi TTS could achieve a speed of up to 160 km/h.
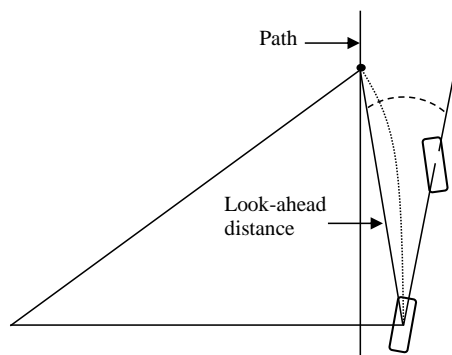

Figure 6. Pure Pursuit approach geometry.

2) Hardware Actuation Control Methods

Hardware actuation control methods compute inputs for the steering, throttle, and brake actuators of the car, which execute the motion plan computed by the motion planning subsystem (Section IV.B) and mitigate inaccuracies caused primarily by actuators' models. One of the most common hardware actuation control methods for self-driving cars is the feedback control. It involves applying a control command input, observing the hardware output, and adjusting future inputs to correct errors in actuators' models. Funke et al. [FUN12] used a feedback control method to implement the control subsystem of an Audi TTS. Hardware structures of the Audi TTS were designed to achieve hard real-time control at 200 Hz. This high-speed hardware enabled their control subsystem to drive the car at up to 160 km/h. Ziegler et al. [ZIE14a] employed a feedback control method for the control subsystem of "Bertha", which drove autonomously on the historic Bertha-Benz-Memorial-Route in Germany. Bertha's control subsystem was able to drive the car at up to 100 km/h through the 103 km long Bertha-Benz-Memorial-Route. Li et al.

[LI17] adopted the same control method in a Toyota Land Cruiser, which was able to drive the car at up to 25 km/h.

Another popular hardware actuation method for autonomous vehicles is the Proportional Integral Derivative (PID). It involves specifying a desired hardware input and an error measure that accounts for how far the hardware output is from the desired hardware input [AST08]. In possession of this information, a PID control method computes the hardware input, which is K_p times the error measure (proportional to the error according to K_p), plus K_i times the integral of the error, plus K_d times the derivative of the error, where K_p, K_i and K_d are parameters of the PID control method. Similar to the feedback control technique, the PID control method can deal with the current error, besides the accumulated past errors over time and future errors [GUI17]. Zhao et al. [ZHA12] used an adaptive PID method based on the generalized minimum variance technique to implement the control subsystem of the driverless car "Intelligent Pionner". Intelligent Pionner's control subsystem was able to drive the car at up to 60 km/h. Levinson et al. [LEV11] employed a mixture of a MPC method, based on well-known physically based car models, and a feedforward PID control technique for the control subsystem of the robotic car "Junior" [MON08] (Stanford University's car that finished in second place in the 2007 DARPA Urban Challenge). Junior's control subsystem was able to drive the car at up to 56 km/h. Guidolini et al. [GUI17] propose a neural based model predictive control (N-MPC) method to tackle delays in the steering wheel hardware of the self-driving car "IARA". They used the MPC method to reduce the impact of steering hardware delays by anticipating control command inputs that would timely move the car according to the trajectory. However, standard techniques for predicting IARA's steering hardware output did not worked well due to its non-linearities and delays. Then, they modeled IARA's steering hardware using a neural network and employed the neural-based steering model in the N-MPC steering control method. The proposed solution was compared to the standard solution based on the PID control method. The PID control subsystem worked well for speeds of up to 25 km/h. However, above this speed, delays of IARA's steering hardware were too high to allow proper operation. The N-MPC subsystem reduced the impact of IARA's steering hardware delays, which allowed its autonomous operation at speeds of up to 37 km/h – an increase of 48% in maximum stable speed.

V. ARCHITECTURE OF THE UFES'S CAR "IARA"

In this section, we present a detailed description of the architecture of the autonomy system of the UFES's car, IARA.

The Intelligent and Autonomous Robotic Automobile (IARA, Figure 7) follows the typical architecture of self-driving cars. IARA is a research autonomous vehicle that was developed at the Laboratory of High Performance Computing (*Laboratório de Computação de Alto Desempenho* – LCAD) of the Federal University of Espírito Santo (*Universidade Federal do Espírito Santo* – UFES). IARA was the first

Brazilian driverless car to travel autonomously 74 km on urban roads and highways. The 74 km route from Vitória to Guarapari was travelled by IARA at the night of May 12, 2017.

IARA is based on a 2011 Ford Escape Hybrid, which was adapted for autonomous driving. The actuation on the steering wheel is made by the factory-installed electric power steering system, which was modified so that an electronic module we have built can send signals equivalent to those sent by the original torque sensor to the electric power steering system. Our electronic module also sends signals equivalent to the signals the throttle pedal sends to the car's electronics. The actuation on the brakes is made by an electric linear actuator attached to the brakes' pedal. The breaks' pedal can be used normally even in autonomous mode, though, thanks to the way the linear actuator is attached to it (the actuator only touches the pedal and can push it). The gear stick is originally attached to a series of electric switches that informs the car's electronics the current gear. A series of relays we have installed allows either connecting our autonomy system to the car's electronics or the original switches controlled by the gear stick. Similarly, the electric power steering wires and throttle pedal wires can be connected to the original car's wiring or to our autonomy system. At the press of a button, we can select the original manual factory operation, or autonomous operation.

To power the computers and sensors that implement the autonomy system, we take advantage of the 330 Volts battery used by the electric powertrain of the Ford Escape Hybrid. We use a DC-DC converter to reduce it to 54 Volts that charges the battery of standard no-breaks that provides 120V AC inside the car. These no-breaks can also be connected to the 120V AC mains, which allows powering the computers and sensors of IARA while it is in the garage without requiring to turn on the Ford Escape Hybrid's engine.

IARA's computers and sensors comprise a workstation (Dell Precision R5500, with 2 Xeon X5690 six-core 3.4GHz processors and one NVIDIA GeForce GTX-1030), networking gear, two LiDARs (a Velodyne HDL-32E and a SICK LD-MRS), three cameras (two Bumblebee XB3 and one ZED), an IMU (Xsens MTi) and a dual RTK GPS (based on the Trimble BD982 receiver).

Following the typical architecture of self-driving cars, IARA's architecture is organized into perception and decision making systems. The perception system is divided mainly into obstacle mapping and localization subsystems. The decision making system is partitioned mainly into path planning, behavior selection, trajectory planning, obstacle avoidance, and control subsystems. Figure 1 shows a block diagram of the main subsystems of IARA's architecture, which depicts how each subsystem connects with each other.

The obstacle mapping (referred to as Obstacle Mapper in Figure 1) subsystem [MUT16] builds an occupancy grid map that represents obstacles of the environment. Obstacle map cells' stores the probability of the associated region being occupied by an obstacle. The Obstacle Mapper operates in offline and online modes. In the offline mode, it receives as

input multiple sensor data (odometer, LIDAR and IMU), which were stored while IARA was conducted by a driver along a path of interest. Then, it estimates IARA's states along the path and the obstacle map cells' values using the GraphSLAM algorithm. In the online mode, it receives as input IARA's state and LIDAR data, and updates the map cells' values.

Figure 7. Intelligent and Autonomous Robotic Automobile (IARA) that was the first Brazilian self-driving car to travel autonomously 74 km on urban roads and highways. A video that shows IARA operating autonomously is available at https://youtu.be/iyKZV0ICysc.

The localization (Localizer in Figure 1) subsystem estimates IARA's state relative to the obstacle map. In the first step, it receives as input IARA's initial state from the GPS; an approximation of the initial state can also be given manually. At each subsequent step, it receives as input IARA's last state, the offline map, and odometer, LIDAR and IMU data. Then, it computes a local obstacle map and estimates IARA's current state, by matching the local map to the offline map (the obstacle map constructed by the Obstacle Mapper in the offline mode), using the particle filter algorithm.

The path planning (Path Planner) subsystem builds a path from IARA's current state to a local goal state. It receives as input IARA's current state and a road definition data file (RDDF). The RDDF is composed of a sequence of IARA's states, which were stored while IARA was conducted by a driver along a path of interest. Then, the path planner extracts from the RDDF a path, which is composed by a sequence of equally spaced states that goes from IARA's current state to a goal state some meters ahead. The path planner follows the 2005 DARPA Grand Challenge pattern, instead of using an approach to obtain the path online, such as A* search algorithm.

The behavior selection (Behavior Selector) subsystem establishes a local goal state in the path according to the scenario. It receives as input the online map, IARA's current state, the path, and a set of annotations in the online map, such as speed bumps', security barriers', crosswalks' and speed limits' positions, and traffic lights' positions and states. Then,

it defines a goal state in the path some seconds ahead of IARA's current state, and adjusts the goal state's pose and velocity, in order to make the car behave properly according to the scenario, such as stopping on red traffic lights, reducing velocity or stopping in busy crosswalks.

The trajectory planning (Trajectory Planner) subsystem [CAR17] computes a trajectory from IARA's current state to a local goal state in the path. It receives as input the online map, IARA's current state, the path and the goal state in the path. Then, it computes a trajectory, from the current IARA's state to the goal state in the path, which follows the path and avoids obstacles represented in the online map, using a model-predictive approach. The trajectory is composed by a sequence of control commands, each one comprised of linear velocity, steering wheel angle and execution time.

The obstacle avoidance (Obstacle Avoider) subsystem [GUI16] verifies and eventually changes the trajectory, in case it becomes necessary to avoid a collision. It receives as input the online map, IARA's current state, and the trajectory. Then, it simulates the trajectory execution along the online map. If the trajectory crashes into an obstacle, the obstacle avoider decreases the linear velocity commands of the trajectory to prevent the accident.

Finally, the control (Controller) subsystem [GUI17] computes the commands that will be sent directly to IARA's actuators in the steering wheel, throttle and brake, employing a Proportional Integral Derivative (PID) approach. It receives as input the trajectory and odometer data. Then, it computes an error measure that accounts for how far actual IARA's steering wheel angle and velocity are from those specified in the trajectory. The error is used to compute the actuation commands that will be applied directly on IARA to minimize this error.

VI. SELF-DRIVING CARS UNDER DEVELOPMENT IN THE INDUSTRY

In this section, we list prominent autonomous research cars developed by technology companies and reported in the media.

Several companies demonstrated interest in developing self-driving cars, and/or investing in technology to support and profit from them. Enterprises range from manufacturing cars and creating hardware for sensing and computing, to developing software for assisted and autonomous driving, entertainment, and in-car advertisement.

We provide an overview of companies doing research and development in self-driving cars. The list is not presented in any specific order, since we aim at making it as impartial and complete as possible. The information was acquired by inspecting companies' websites and news published in other media.

Torc was one of the pioneers in developing cars with self-driving capabilities. The company was founded in 2005. In 2007, it joined Virginia Tech's team to participate in the 2007 DARPA Urban Challenge with their autonomous car "Odin" [BAC08], which reached third place in the competition. The technology used in the competition was improved since then and it was successfully applied in a variety of commercial ground vehicles, from large mining trucks to military vehicles [TOR18].

Google's self-driving car project began in 2009 and was formerly led by Sebastian Thrun, who also led the Stanford University's team with their car "Stanley" [THR07], winner of the 2005 DARPA Grand Challenge. In 2016, Google's self-driving car project became an independent company called Waymo [WAY18]. The company is a subsidiary of the holding Alphabet Inc., which is also Google's parent company. Waymo's self-driving car uses a sensor set composed of LIDARs to create a detailed map of the world around the car, RADARs to detect distant objects and their velocities, and high resolution cameras to acquire visual information, such as whether a traffic signal is red or green [WAY18b].

Baidu, one of the giant technology companies in China, is developing an open source self-driving car project with codename Apollo [APO18]. The source code for the project is available in GitHub [APOL18]. It contains modules for perception (detection and tracking of moving obstacles, and detection and recognition of traffic lights), HD map and localization, planning, and control, among others. Several companies are partners of Baidu in the Apollo project, such as TomTom, Velodyne, Bosch, Intel, Daimler, Ford, Nvidia, and Microsoft. One of the Apollo project's goals is to create a centralized place for original equipment manufacturers (OEMs), startups, suppliers, and research organizations to share and integrate their data and resources. Besides Baidu, Udacity, an educational organization founded by Sebastian Thrun and others, is also developing an open source self-driving car, which is available for free in GitHub [UDA18].

Uber is a ride-hailing service and, in 2015, they partnered with the Carnegie Mellon University to develop self-driving cars [UBE15]. A motivation for Uber's project is to replace associated drivers by autonomous software [UBE18].

Lyft is a company that provides ridesharing and on-demand driving services. Like Uber, Lyft is doing research and development in self-driving cars [LYF18]. The company aims at developing cars with level 5 of autonomy.

Aptiv is one of Lyft's partners. Aptiv was created in a split of Delphi Automotive, a company owned by General Motors. Aptiv's objective is to build cars with level 4 and, posteriorly, level 5 of autonomy [APT18]. Besides other products, the company sells short-range communication modules for vehicle-to-vehicle information exchange. Aptiv recently acquired two relevant self-driving companies, Movimento and nuTonomy.

Didi is a Chinese transportation service that bought Uber's rights in China. Didi's self-driving car project was announced in 2017 and, in February of 2018, they did the first successful demonstration of their technology. In the same month, company's cars started being tested in USA and China. Within a year, Didi obtained the certificate for HD mapping in China. The company is now negotiating a partnership with Renault, Nissan, and Mitsubishi to build an electric and autonomous ride-sharing service [ENG18].

Tesla was founded in 2003 and, in 2012, it started selling its first electric car, the Model S. In 2015, the company enabled the autopilot software for owners of the Model S. The software has been improved since then and its current version, the so called enhanced autopilot, is now able to match speed with traffic conditions, keep within a lane, change lanes, transition from one freeway to another, exit the freeway when the destination is near, self-park when near a parking spot, and be summoned to and from the user's garage [TES18]. Tesla's current sensor set does not include LIDARs.

A Chinese company called LeEco is producing self-driving luxury sedans to compete with the Tesla Model S. The company is also backing up Faraday Future for the development of a concept car. LeEco is also partnering Aston Martin for the development of the RapidE electric car [VERG16].

Besides developing hardware for general-purpose high-performance computing, NVDIA is also developing hardware and software for self-driving cars [NVI18]. Although their solutions rely mostly on artificial intelligence and deep learning, they are also capable of performing sensor fusion, localization in HD maps, and planning [NVID18].

Aurora is a new company founded by experienced engineers that worked in Google's self-driving project, Tesla, and Uber [AUR18]. The company plans to work with automakers and suppliers to develop full-stack solutions for cars with level 4 and, eventually, level 5 of autonomy. Aurora has independent partnerships with Volkswagen group (that owns Volkswagen Passenger Cars, Audi, Bentley, Skoda, and Porsche) and Hyunday [FORT18].

Zenuity is a joint venture created by Volvo Cars and Autoliv [ZEN18]. Ericsson will aid in the development of Zenuity's Connected Cloud that will use Ericsson's IoT Accelerator [ERI18]. TomTom also partnered with Zenuity and provided its HD mapping technology [TOM18]. TomTom's HD maps will be used for localization, perception and path planning in Zenuity's software stack.

Daimler and Bosch are joining forces to advance the development of cars with level 4 and 5 of autonomy by the beginning of the next decade [BOS18]. The companies already have an automated valet parking in Stuttgart [DAA18] and they also have tested the so called Highway Pilot in trucks in USA and Germany [DAB18]. Besides partnering with Bosch, Daimler also merged its car sharing business, Car2Go, with BMW's ReachNow, from the same business segment, in an effort to stave off competition from other technology companies, such as Waymo and Uber [DAC18]. The new company will include business in car sharing, ride-hailing, valet parking, and electric vehicle charging.

Argo AI founders led self-driving car teams at Google and Uber [ARG18]. The company received an investment of U$ 1 billion from Ford with the goal of developing a new software platform for Ford's fully autonomous vehicle (level 4) coming in 2021. They have partnerships with professors from Carnegie Mellon University and Georgia Tech [ARB18].

Renesas Autonomy develops several solutions for automated driving assistant systems (ADAS) and automated driving [REN18]. The company has a partnership with the University of Waterloo [RENE18].

Honda revealed in 2017 plans for introducing cars with level 4 of autonomy by 2025. The company intends to have vehicles with level 3 of autonomy by 2020 and it is negotiating a partnership with Waymo [HON18].

Visteon is a technology company that manufactures cockpit electronic products and connectivity solutions for several vehicle manufacturers [VIS18]. In 2018, Visteon introduced its autonomous driving platform capable of level 3 of autonomy and, potentially, higher levels. The company does not aim at producing cars and sensors, but at producing integrated solutions and software.

AImotive is using low cost components to develop a self-driving platform [AIM18]. Its solution relies strongly on computer vision, but it uses additional sensors. The company is already able to perform valet parking and navigate in highways. AImotive also developed a photorealistic simulator for data collection and preliminary system testing, and chips for artificial intelligence-based, latency-critic, and camera-centric systems.

As AImotive, AutoX is avoiding to use LIDARs in its solution. However, the company is going further than Almotive and trying to develop a level 5 car without using RADARs, ultrasonics, and differential GPS's. AutoX's approach is to create a full-stack software solution based on artificial intelligence [AUT18].

Mobileye is also seeking to develop a self-driving solution without using LIDARs, but relying mostly in a single-lensed camera (mono-camera). Mobileye is one of the leading suppliers of software for Advanced Driver Assist Systems (ADAS), with more than 25 partners among automakers. Beyond ADAS, Mobileye is also developing technology to support other key components for autonomous driving, such as perception (detection of free space, driving paths, moving objects, traffic lights, and traffic signs, among others), mapping, and control. The company partnered with BMW and Intel to develop production-ready fully autonomous vehicles, with production launch planned for 2021 [MOB18].

Ambarella also does not use LIDAR, but only RADARs and stereo cameras. The company joined the autonomous driving race in 2015 by acquiring VisLAB. Different from other companies, Ambarella does not aim at becoming a tier-one supplier or selling complete autonomous-driving systems. Instead, they plan to sell chips and software to automakers, suppliers, and software developers. Ambarella's current research and development guidelines include detection of vehicles, obstacles, pedestrians, and lanes; traffic sign recognition; terrain mapping; and issues related to technology commercialization, such as system calibration, illumination, noise, temperature, and power consumption [AMB18].

Pony.ai was founded in December of 2016 and, in July of 2017, it completed its first fully autonomous driving demonstration. The company signed a strategic agreement with one of the biggest Chinese car makers, the Guangzhou Auto Group (GAC) [PON18].

Navya [NAA18] and Transdev [TRA18] are French companies that develop self-driving buses. Navya has several of their buses being tested in Europe, Asia, and Australia. Their sensor set consists of two multi-layer 360º LIDARs, six 180º mono-layer LIDARs, front and rear cameras, odometer (wheels encoder + IMU), and a GNSS RTK [NAB18]. Transdev is also already demonstrating their self-driving buses for the public [TRA18].

JD is a Chinese e-commerce company interested in building self-driving delivery vehicles [JD18]. JD's project, started in 2016, is being developed together with Idriverplus [IDR18], a Chinese self-driving car startup.

In March, 2018, Toyota announced an investment of U\$ 2.8 billion in the creation of a new company called the Toyota Research Institute-Advanced Development (TRI-AD) with the goal of developing an electric and autonomous car until 2020 [TOYT18]. Besides Toyota [TOY18], other car manufacturers, such as Ford [FOR18], Volvo [VOLV18], and Mercedes-Benz [MER18], have also recently presented their plans for self-driving cars. Ford defined 2021 as a deadline for presenting a fully autonomous vehicle ready for commercial operation.

REFERENCES

[ABR12]   I. Abraham, D. Delling, A. V Goldberg, and R. F. Werneck, "Hierarchical hub labelings for shortest paths", European Symposium on Algorithms, pp. 24–35, 2012.

[AEB15]   M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen, "Experience, results and lessons learned from automated driving on Germany's highways", IEEE Intelligent Transportation Systems Magazine, vol. 7, no. 1, pp. 42–57, 2015.

[AHM17]   T. Ahmad, D. Ilstrup, E. Emami, and G. Bebis, "Symbolic road marking recognition using convolutional neural networks", in Intelligent Vehicles Symposium (IV), 2017, pp. 1428–1433.

[AIM18]   AIMotive, "The scalable future of autonomy", 2016. Available: https://aimotive.com. [Accessed: 17- Apr- 2018].

[ALI15]   C. Alia, T. Gilles, T. Reine, and C. Ali, "Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method", IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 674–679.

[AMA15]   E. Amaral, C. Badue, T. Oliveira-Santos, A. F. De Souza, "Detecção e Rastreamento de Veículos em Movimento para Automóveis Robóticos Autônomos", Simpósio Brasileiro de Automação Inteligente, 2015, pp. 801–806.

[AMB18]   Ambarella. "Our Vision is Computer Vision", 2018. Available: https://www.ambarella.com/about/about-ambarella/the-ambarella-story. [Accessed: 17- Apr- 2018].

[APO18]   Apollo, "Promote Open Capability - Share Resources - Accelerate Innovation - Sustain Mutual Benefit", 2018. Available: http://apollo.auto/. [Accessed: 17- Apr- 2018]

[APOL18]   Available: https://github.com/ApolloAuto/apollo. [Accessed: 17- Apr- 2018]

[APT18]   Aptiv, "Making mobility work", 2018. Available: https://www.aptiv.com/our-journey. [Accessed: 17- Apr- 2018]

[ARB18]   The Verge, "An inside look at Ford's \$1 billion bet on Argo AI", 2017.Available: https://www.theverge.com/2017/8/16/16155254/argo-ai-ford-self-driving-car-autonomous. [Accessed: 17- Apr- 2018].

[ARG18]   Argo AI, "Our vision for a safer tomorrow", 2018. Available: https://www.argo.ai/. [Accessed: 17- Apr- 2018].

[ARN16]   R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L.

            Acosta, "Safe and reliable path planning for the autonomous vehicle verdino", IEEE Intelligent Transportation Systems Magazine, vol. 8, no. 2, pp. 22–32, 2016.

[ARZ13]   J. Arz, D. Luxen, and P. Sanders, "Transit Node Routing Reconsidered", 12th International Symposium on Experimental Algorithms (SEA'13), pp. 55–66, 2013.

[AST08]   K. J. Aström and R. M. Murray, "Feedback Systems: An Introduction for Scientists and Engineers", Princeton University Press, 2008.

[AUR18]   Aurora, "We do self-driving cars", 2018. Available: https://aurora.tech/. [Accessed: 17- Apr- 2018].

[AUT17]   SparkFun, "Autonomous Vehicle Competition", 2017. Available: https://avc.sparkfun.com/. [Accessed: 22- May- 2018]

[AUT18]   AutoX, "Democratizing autonomy", 2018. Available: https://www.autox.ai/. [Accessed: 17- Apr- 2018].

[AZI14]   A. Azim and O. Aycard, "Layer-based supervised classification of moving objects in outdoor dynamic environment using 3D laser scanner", IEEE Intelligent Vehicles Symposium, 2014, pp. 1408–1414.

[BAC08]   A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. Van Covern, and M. Webster, "Odin: team victortango's entry in the darpa urban challenge", Journal of Field Robotics, vol. 25, no. 8, pp. 467–492, 2008.

[BAI17]   O. Bailo, S. Lee, F. Rameau, J. S. Yoon, and I. S. Kweon, "Robust Road Marking Detection and Recognition Using Density-Based Grouping and Machine Learning Techniques", in Winter Conference on Applications of Computer Vision (WACV), 2017, pp. 760–768.

[BAR08]   N. Barnes, A. Zelinsky, and L. S. Fletcher, "Real-Time Speed Sign Detection Using the Radial Symmetry Detector", IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 2, pp. 322–332, 2008.

[BAS15]   H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route Planning in Transportation Networks", arXiv preprint arXiv:1504.05140, 2015.

[BAS18]   F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D. Dewitt, "RoadTracer: automatic extraction of road networks from aerial images", preprint MIT, 2018.

[BAU09]   R. Bauer and D. Delling, "SHARC: Fast and robust unidirectional routing", ACM Journal of Experimental Algorithmics, vol. 14, no. 2.4, pp. 1–29, 2009.

[BAU10]   R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner, "Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm", ACM Journal of Experimental Algorithmics, vol. 15, no. 2.3, pp. 1–31, 2010.

[BEH17]   K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification", in International Conference on Robotics and Automation (ICRA), 2017, pp. 1370–1377.

[BEN14]   P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving", in Intelligent Vehicles Symposium Proceedings, 2014 IEEE, pp. 420-425, 2014.

[BER14]   J. Ziegler, H. Lategahn, M. Schreiber, C. G. Keller, C. Knoppel, J. Hipp, M. Haueis, and C. Stiller, "Video based localization for bertha", IEEE Intelligent Vehicles Symposium (IV), 2014, pp. 1231–1238.

[BER14]   N. Bernini, M. Bertozzi, L. Castangia, M. Patander, and M. Sabbatelli, "Real-time obstacle detection using stereo vision for autonomous ground vehicles: a survey", International IEEE Conference on Intelligent Transportation Systems 2014, pp. 873–878.

[BER15]   R. F. Berriel, E. Aguiar, V. V. S. Filho, and T. Oliveira-Santos, "A Particle Filter-based Lane Marker Tracking Approach using Cubic Spline Model", SIBGRAPI Conference on Graphics, Patterns and Images, Salvador, Brazil, 2015.

[BER17b]   R. F. Berriel, F. S. Rossi, A. F. de Souza, and T. Oliveira-Santos, "Automatic large-scale data acquisition via crowdsourcing for crosswalk classification: A deep learning

approach", Computers & Graphics, vol. 68, pp. 32–42, 2017.

[BER17c] R. F. Berriel, E. de Aguiar, A. F. de Souza, and T. Oliveira-Santos, "Ego-Lane Analysis System (ELAS): Dataset and algorithms", Image and Vision Computing, vol. 68, pp. 64–75, 2017.

[BMW18] BMW, "BMW: The Path to Autonomous Driving", 2018. Available: https://www.bmw.com/en/automotive-life/autonomous-driving.html. [Accessed: 22- May- 2018]

[BOR15] M. Borno Jensen, M. P. Philipsen, A. Mogelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives", IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, pp. 1800-1815, 2015.

[BOS18] Daimler, "Future mobility. Bosch and Daimler join forces to work on fully automated, driverless system", 2018. Available: https://www.daimler.com/innovation/case/autonomous/bosch-cooperation.html. [Accessed: 17- Apr- 2018].

[BRO12] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, "The vislab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles", International Journal of Vehicle Autonomous Systems, vol. 10, no. 3, p. 147, 2012.

[BRO15] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti, "PROUD—public road urban driverless-car test", IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3508–3519, 2015.

[BRO17] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles", Control Engineering Practice, vol. 61, pp. 307–316, 2017.

[BRO99] A. Broggi, M. Bertozzi, and A. Fascioli, "ARGO and the millemiglia in automatico tour", IEEE Intelligent Systems, vol. 14, no. 1, pp. 55–64, 1999.

[BRU13] A. Viswanathan, B. R. Pires, and D. Huber, "Vision-based robot localization across seasons and in remote locations", IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 4815–4821.

[BRU16] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 4, pp. 652–665, 2016.

[BUE07] M. Buehler, K. Iagnemma, and S. Singh, Eds., The 2005 DARPA grand challenge: the great robot race, Springer-Verlag Berlin Heidelberg, 2007.

[BUE09] M. Buehler, K. Iagnemma, and S. Singh, Eds., The DARPA urban challenge: autonomous vehicles in city traffic, Springer-Verlag Berlin Heidelberg, 2009.

[BYU15] K. Na, J. Byun, M. Roh, and B. Seo, "RoadPlot-DATMO: moving object tracking and track fusion system using multiple sensors", International Conference on Connected Vehicles and Expo, 2015, pp. 142–143.

[CAB12] M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina, "Suspended Traffic Lights Detection and Distance Eestimation Using Color Features", in 15th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1315–1320, 2012.

[CAB15] M. Diaz-Cabrera, P. Cerri, and P. Medici, "Robust Real-Time Traffic Light Detection and Distance Estimation Using a Single Camera", Expert Systems with Applications, vol. 42, no. 8, pp. 3911–3923, 2015

[CAR17] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese, and A. F. De Souza, "A model-predictive motion planner for the iara autonomous car", in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 225–230.

[CAR18] Carnegie Mellon University, "General Motors-Carnegie Mellon autonomous driving collaborative research lab", 2018. Available: https://www.ri.cmu.edu/robotics-area/general-motors-carnegie-mellon-autonomous-driving-collaborative-research-lab/. [Accessed: 22- May- 2018]

[CARN18] R. V. Carneiro, R. C. Nascimento, R. Guidolini, V. B. Cardoso, T. Oliveira-Santos, C. Badue and A. F. De Souza, "Mapping road lanes using laser remission and deep neural networks", arXiv preprint arXiv:1804.10662, 2018.

[CER11] P. Cerri, G. Soprani, P. Zani, J. Choi, J. Lee, D. Kim, K. Yi, and A. Broggi, "Computer vision at the hyundai autonomous challenge", 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2011, pp. 777–783.

[CHE08] B. Chen and G. Quan, "Np-hard problems of learning from examples", 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008, pp. 182–186.

[CHE17] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation", 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3656–3663.

[CHEN17] L. Chen, L. Fan, G. Xie, K. Huang, and A. Nuchter, "Moving-object detection from consecutive stereo pairs using slanted plane smoothing", IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 11, p. 3093–3102, 2017.

[CHO14] H. Cho, Y.-W. Seo, B. V. K. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments", IEEE International Conference on Robotics and Automation, 2014, pp. 1836–1843.

[CHU12] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles", IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1599–1616, 2012.

[CHU15] K. Chu, J. Kim, K. Jo, and M. Sunwoo, "Real-time path planning of autonomous vehicles for unstructured road navigation", International Journal of Automotive Technology, vol. 16, no. 4, pp. 653–668, 2015..

[COH03] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-hop labels", SIAM Journal on Computing, vol. 32, no. 5, pp. 1338–1355, 2003.

[CUM08] M. Cummins and P. Newman, "FAB-MAP: probabilistic localization and mapping in the space of appearance", The International Journal of Robotics Research, vol. 27, no. 6, pp. 647–665, 2008.

[DAA18] Daimler, "Driverless in the parking lot. Automated Valet Parking", 2018. Available: https://www.daimler.com/innovation/case/autonomous/driverless-parking.html. [Accessed: 17- Apr- 2018].

[DAB18] Daimler, "Highway Pilot. The Autopilot for Trucks", 2018. Available: https://www.daimler.com/innovation/case/autonomous/highway-pilot-2.html. [Accessed: 17- Apr- 2018].

[DAC18] The Verge, "BMW and Daimler are putting their differences aside to beat Uber", 2018. Available: https://www.theverge.com/2018/4/2/17188374/bmw-daimler-merger-car2go-reachnow-mobility. [Accessed: 17- Apr- 2018].

[DAR05] DARPA, "DARPA grand challenge 2005 route data definition file", Aug. 2005. [Online]. Available: http://archive.darpa.mil/grandchallenge05/RDDF_Document.pdf

[DAR07] DARPA, "Urban challenge route network definition file (RNDF) and mission data file (MDF) formats", Mar. 2007. [Online]. Available: https://www.grandchallenge.org/grandchallenge/docs/RNDF_MDF_Formats_031407.pdf

[DAR09] M. S. Darms, P. E. Rybski, C. Baker, and C. Urmson, "Obstacle detection and tracking for the urban challenge", IEEE Transactions on Intelligent Transportation Systems, 2009.

[DEL09] D. Delling, M. Holzer, K.l Müller, F. Schulz, and D. Wagner, "High-performance multi-level routing", The Shortest Path Problem: Ninth DIMACS Implementation Challenge, vol. 74 of DIMACS Book, American Mathematical Society, pp. 73–92, 2009.

[DEL13a] D. Delling, A. V. Goldberg, and R. F. Werneck, "Hub Label Compression", 12th International Symposium on Experimental Algorithms (SEA'13), pp. 18–29, 2013.

[DEL13b] D. Delling, A. V. Goldberg, A. Nowatzyk, and R. F. Werneck, "PHAST: Hardware-accelerated shortest path trees", Journal of Parallel and Distributed Computing, vol. 73, no. 7, pp. 940–952, 2013.

[DEL15]   D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, "Customizable route planning in road networks", Transportation Science, vol. 51, no. 2, pp. 566–591, 2015.

[DHA16]   N. Dharia, "Location Platform Index: Mapping and Navigation, 1H16", OVUM, 2016.

[DIA12]   M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina, "Suspended Traffic Lights Detection and Distance Estimation Using Color Features", in International Conference on Intelligent Transportation Systems (ITSC), 2012, pp. 1315–1320.

[DIA15a]   J. E. A. Dias, G. A. S. Pereira, and R. M. Palhares, "Longitudinal model identification and velocity control of an autonomous car", IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 1–11, 2014.

[DIA15b]   M. Diaz-Cabrera, P. Cerri, and P. Medici, "Robust Real-Time Traffic Light Detection and Distance Estimation Using a Single Camera", Expert Systems with Applications, vol. 42, no. 8, pp. 3911–3923, 2015.

[DIC87]   E. D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision 1", IFAC Proceedings Volumes, vol. 20, no. 5, pp. 221–226, 1987.

[DID18]   DiDi Labs, "DiDi Labs is working to solve the world's most challenging problems", 2018. Available: http://www.didi-labs.com/. [Accessed: 17- Apr- 2018]

[DIJ59]   E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[DIJ59]   E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[DOH16]   K. Doherty, J. Wang, and B. Englot, "Probabilistic map fusion for fast, incremental occupancy mapping with 3d 26ilbert maps", 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1011–1018.

[DOL10]   D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments", The International Journal of Robotics Research, vol. 29, no. 5, pp. 485–501, 2010.

[DRO17]   D. Droeschel, M. Schwarz, and S. Behnke, "Continuous mapping and localization for autonomous navigation in rough terrain using a 3d laser scanner", Robotics and Autonomous Systems, vol. 88, pp. 104–115, 2017.

[DU16]   M. Du, T. Mei, H. Liang, J. Chen, R. Huang, and P. Zhao, "Drivers' visual behavior-guided RRT motion planner for autonomous on-road driving", Sensors, vol. 16, no. 1, p. 102, 2016.

[ELF85]   H. Moravec and A. Elfes, "High resolution maps from wide angular sensors", Proceedings of the IEEE International Conference On Robotics and Automation ICRA85, vol. 2, 1985, pp. 116–121.

[ELF89]   A. Elfes, "Using occupancy grids for mobile robot perception and navigation", Computer, vol. 22, no. 6, pp. 46–57, 1989.

[ELR18]   ELROB, "The European Land Robot Trial (ELROB)", 2018. Available: http://www.elrob.org/. [Accessed: 22- May- 2018]

[ENG18]   Engadget, "Renault-Nissan and Didi plan self-driving ride service in China ", 2018. Available: https://www.engadget.com/2018/02/07/renault-nissan-mitsubishi-didi-chuxing-china/. [Accessed: 17- Apr- 2018]

[ERI18]   Ericsson, "Ericsson and Zenuity team up for self-driving cars ", 2017. Available: https://www.ericsson.com/en/press-releases/2017/9/ericsson-and-zenuity-team-up-for-self-driving-cars. [Accessed: 17- Apr- 2018].

[ESS10]   A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments", International Journal of Robotics Research, vol. 29, no. 14, pp. 1707–1725, 2010.

[FAS16]   D. Fassbender, B. C. Heinrich, and H.-J. Wuensche, "Motion planning for autonomous vehicles in highly constrained urban environments", 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4708–4713.

[FER08]   D. Ferguson, T. Howard and M. Likhachev, "Motion Planning in Urban En-vironments", Journal of Field Robotics, vol. 25, no. 11–12, pp. 939–960, 2008.

[FER14]   L. C. Fernandes, J. R. Souza, G. Pessin, P. Y. Shinzato, D. Sales, C. Mendes, M. Prado, R. Klaser, A. C. Magalhães, A. Hata, D. Pigatto, K. Castelo Branco, V. Grassi, F. S. Osorio, and D. F. Wolf, "CaRINA intelligent robotic car: architectural design and applications", Journal of Systems Architecture, vol. 60, no. 4, pp. 372–392, 2014.

[FID15]   E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey," Robotics and Autonomous Systems, vol. 64, pp. 1–20, 2015.

[FOR18]   Ford, "Looking Further: Ford will Have a Fully Autonomous Vehicle in Operation by 2021", 2018. Available: https://corporate.ford.com/innovation/autonomous-2021.html. [Accessed: 22- May- 2018]

[FORE18]   A. Forechi, T. Oliveira-Santos, C. Badue, and A. F. De Souza, "Visual Global Localization with a Hybrid WNN-CNN Approach", 2018 International Joint Conference on Neural Networks (IJCNN), 2018.

[FORT18]   Fortune, "Aurora Is The Secretive Startup From the Best Minds in Self-Driving Cars ", 2017. Available: http://fortune.com/2017/04/28/aurora-innovation-startup/. [Accessed: 17- Apr- 2018]

[FOU11]   P. Foucher, Y. Sebsadji, J. Tarel, P. Charbonnier, and P. Nicolle, "Detection and recognition of urban road markings using images", in International Conference on Intelligent Transportation Systems (ITSC), 2011, pp. 1747–1752.

[FUN12]   J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler and B. Huhnke, "Up to the Limits: Autonomous Audi TTS", IEEE Intelligent Vehicles Symposium (IV), 2012, pp. 541-547.

[FZI18]   FZI, "Research for the mobility of the future", 2018. Available: https://www.fzi.de/en/research/research-sectors/anwendungsfelder-en/afeld/mobilitaet. [Accessed: 29-Aug-2018]

[GAO06]   X. W. Gao, L. Podladchikova, D. Shaposhnikov, K. Hong, and N. Shevtsova, "Recognition of traffic signs based on their colour and shape features extracted using human vision models", Journal of Visual Communication and Image Representation, vol. 17, no. 4, pp. 675–685, 2006.

[GAR15]   E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey", Robotics and Autonomous Systems, vol. 64, pp. 1–20, 2015.

[GCD16]   TNO, "Grand Cooperative Driving Challenge", 2016. Available: http://www.gcdc.net. [Accessed: 22- May- 2018]

[GE17]   Z. Ge, P. Wang, J. Wang, and Z. Chen, "A 2.5D grids based moving detection methodology for autonomous vehicle", International Conference on Robotics and Automation Engineering, 2017, pp. 403–407.

[GEI12]   R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies", Transportation Science, vol. 46, no. 3, pp. 388–404, 2012.

[GER14]   J. Becker, M.-B. A. Colas, S. Nordbruch, and M. Fausten, "Bosch's Vision and Roadmap Toward Fully Autonomous Driving", in Road Vehicle Automation, G. Meyer and S. Beiker, Eds., Springer International Publishing, pp. 49-60, 2014.

[GIM17]   S. Gim, L. Adouane, S. Lee, and J.-P. Dérutin, "Clothoids composition method for smooth path generation of car-like vehicle navigation", Journal of Intelligent & Robotic Systems, vol. 88, no. 1, pp. 129–146, 2017.

[GIR16]   P. Girão, A. Asvadi, P. Peixoto, and U. Nunes, "3D object tracking in driving environment: a short review and a benchmark dataset", IEEE Conference on Intelligent Transportation Systems, 2016, pp. 7–12.

[GM18]   Cruise, "Cruise: The Driverless Revolution", 2018. Available: https://getcruise.com. [Accessed: 22- May- 2018]

[GOL05]   A. V. Goldberg and C. Harrelson, "Computing the shortest path: A* search meets graph theory", 16th Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, pp. 156–165, 2005.

[GOL09]   A. V. Goldberg, H. Kaplan, and R. F. Werneck, "Reach for A*: Shortest path algorithms with preprocessing", The Shortest Path Problem: Ninth DIMACS Implementation Challenge, vol. 74 of DIMACS Book, American Mathematical Society, pp.

93–139, 2009.

[GOM14] A. E. Gomez, F. A. R. Alencar, P. V Prado, F. S. Osorio, and D. F. Wolf, "Traffic lights detection and state estimation using Hidden Markov Models", in Intelligent Vehicles Symposium (IV), 2014, pp. 750–755.

[GON10] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, and H. Chen, "The recognition and tracking of traffic lights based on color segmentation and CAMSHIFT for intelligent vehicles", in Intelligent Vehicles Symposium (IV), 2010, pp. 431–435.

[GON16] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles", IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135–1145, 2016.

[GRE00] R. Gregor, M. Lützeler, M. Pellkofer, K. H. Siedersberger, and E. D. Dickmanns, "EMS-vision: a perceptual system for autonomous vehicles", IEEE Transactions on Intelligent Transportation Systems, vol. 3, no. 1, pp. 48–59, 2002.

[GRE15] J. Greenhalgh and M. Mirmehdi, "Detection and Recognition of Painted Road Surface Markings", in International Conference on Pattern Recognition Applications and Methods (ICPRAM), 2015, pp. 130–138.

[GU15] Tianyu Gu, J. Atwood, Chiyu Dong, J. M. Dolan, and Jin-Woo Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, vol. 2015–Decem, pp. 250–256.

[GU16] T. Gu, J. M. Dolan, and J.-W. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, vol. 2016–Novem, pp. 5474–5480.

[GUD16] A. Gudigar, S. Chokkadi, and U. Raghavendra, "A Review on Automatic Detection and Recognition of Traffic Sign", Multimedia Tools and Applications, vol. 75, no. 1, pp. 333–364, 2016.

[GUI16] R. Guidolini, C. Badue, M. Berger and A. F. De Souza, "A Simple Yet Effective Obstacle Avoider for the IARA Autonomous Car", 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), Rio de Janeiro, Brazil, 2016.

[GUI17] R. Guidolini, A. F. De Souza, F. Mutz, and C. Badue, "Neural-based model predictive control for tackling steering delays of autonomous cars", International Joint Conference on Neural Networks (IJCNN), 2017, pp. 4324–4331.

[GUI18] R. Guidolini, L. Scart, L. F. R. Jesus, T. Oliveira-Santos, C. Badue, and A. F. De Souza, "Handling pedestrians in crosswalks using deep neural networks in the IARA autonomous car", 2018 International Joint Conference on Neural Networks (IJCNN 2018), Rio de Janeiro, Brazil, 2018.

[GUR16] A. Gurghian, T. Koduri, S. V Bailur, K. J. Carey, and V. N. Murali, "DeepLanes: End-To-End Lane Position Estimation Using Deep Neural Networks", in Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 38–45.

[GUT04] R. J. Gutman, "Reach-based routing: A new approach to shortest path algorithms optimized for road networks", Proceedings of the 6th Workshop on Algorithm Engineering and Experiments, SIAM, pp. 100–111, 2004.

[HAL15] V. Haltakov, J. Mayr, C. Unger, and S. Ilic, "Semantic Segmentation Based Traffic Light Detection at Day and at Night", in German Conference on Pattern Recognition (GCPR), 2015, pp. 446–457.

[HAR68] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[HAT14] A. Hata and D. Wolf, "Road marking detection using lidar reflective intensity data and its application to vehicle localization", 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 584–589.

[HAT16a] A. Y. Hata, D. F. Wolf, and F. T. Ramos, "Particle filter localization on continuous occupancy maps", Springer, Cham, 2017, pp. 742–751.

[HAT16b] A. Y. Hata and D. F. Wolf, "Feature detection for vehicle localization in urban environments using a multilayer lidar", IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pp. 420–429, 2016.

[HAT17] A. Y. Hata, F. T. Ramos, and D. F. Wolf, "Monte carlo localization on gaussian process occupancy maps for urban environments", IEEE Transactions on Intelligent Transportation Systems, pp. 1–10, 2017.

[HE16] M. He, E. Takeuchi, Y. Ninomiya, and S. Kato, "Precise and efficient model-based vehicle tracking method using Rao-Blackwellized and scaling series particle filters", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016, pp. 117–124.

[HEL16] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks", European Conference on Computer Vision, Springer: Cham, pp. 749–765, 2016.

[HIL09] M. Hilger, E. Köhler, R. H. Möhring, and H. Schilling, "Fast point-to-point shortest path computations with arc-flags", The Shortest Path Problem: Ninth DIMACS Implementation Challenge, vol. 74 of DIMACS Book, American Mathematical Society, pp. 41–72, 2009.

[HIL14] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey", Machine Vision and Applications, vol. 25, no. 3, pp. 727–745, 2014.

[HON18] The Verge, "Honda reveals its plans for autonomous vehicles ", 2017. Available: https://www.theverge.com/2017/6/8/15761272/honda-self-driving-cars-autonomous-level-4-date#1ff68c563b3a. [Accessed: 17- Apr- 2018].

[HOR13] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees", Autonomous Robots, vol. 34, no. 3, pp. 189–206, 2013.

[HOU13] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark", in International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–8.

[HOW07] Howard, T. M., & Kelly, A. (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. International Journal of Robotics Research, 26(2), 141–166.

[HU18] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles", Mechanical Systems and Signal Processing, vol. 100, pp. 482–500, 2018.

[HUV15] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An empirical evaluation of deep learning on highway driving", arXiv preprint arXiv: 1504.01716, 2015.

[HWA16] S. Hwang, N. Kim, Y. Choi, S. Lee, and I. S. Kweon, "Fast multiple objects detection and tracking fusing color camera and 3d LIDAR for intelligent vehicles", International Conference on Ubiquitous Robots and Ambient Intelligence, 2016, pp. 234–239.

[HYE16] D. Hyeon, S. Lee, S. Jung, S.-W. Kim, and S.-W. Seo, "Robust road marking detection using convex grouping method in around-view monitoring system", in Intelligent Vehicles Symposium (IV), 2016, pp. 1004–1009.

[IDR18] IDriverPlus, "IDriverPlus", 2018. Available: http://www.idriverplus.com/. [Accessed: 17- Apr- 2018].

[IVA08] V. Ivanchenko, J. Coughlan, and H. Shen, "Detecting and locating crosswalks using a camera phone", In Computer Vision and Pattern Recognition Workshops (CVPRW), 2008, pp. 1–8.

[JAN14] C. Jang, C. Kim, D. Kim, M. Lee, and M. Sunwoo, "Multiple exposure images based traffic light recognition", in Intelligent Vehicles Symposium (IV), 2014, pp. 1313–1318.

[JD18] Jingdong, "Jingdong", 2018. Available: http://www.jd.com/. [Accessed: 17- Apr- 2018].

[JEN16] M. B. Jensen, M. P. Philipsen, A. Mogelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives", IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, pp. 1800–1815, 2016.

[JEN17] M. B. Jensen, K. Nasrollahi, and T. B. Moeslund, "Evaluating State-of-the-Art Object Detector on Challenging Traffic Light Data", in Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 882–888.

[JO15] K. Jo, Y. Jo, J. K. Suhr, H. G. Jung, and M. Sunwoo, "Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3377–3392, 2015.

[JOH14] V. John, K. Yoneda, B. Qi, Z. Liu, and S. Mita, "Traffic light recognition in varying illumination using deep learning and saliency map", in International Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 2286–2291.

[JUN05] C. R. Jung and C. R. Kelber, "Lane following and lane departure using a linear-parabolic model", Image and Vision Computing, vol. 23, no. 13, pp. 1192–1202, 2005.

[KAL13] R. Kala and K. Warwick, "Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization", Journal of Intelligent & Robotic Systems, vol. 72, no. 3–4, pp. 559–590, 2013.

[KAM04] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, and Heng-Ming Tai, "Autonomous local path planning for a mobile robot using a genetic algorithm," in Proceedings of the 2004 Congress on Evolutionary Computation, 2004, pp. 1338–1345.

[KAR17] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles", 2017 American Control Conference (ACC), 2017, pp. 4023–4028.

[KIM13] S. Kim and J. Kim, "3D Continuous Occupancy Maps using Overlapping Gaussian Processes", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4709–4714, 2013.

[KOG13] A. Koga, H. Okuda, Y. Tazaki, T. Suzuki, K. Haraguchi and Z. Kang, "Realization of Different Driving Characteristics for Autonomous Vehicle by Using Model Predictive Control", IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 722-728.

[KON00] K. Konolige, "A gradient method for realtime robot control", Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), vol. 1, pp. 639–646, 2000.

[KOU12] E. Koukoumidis, M. Martonosi, and L.-S. Peh, "Leveraging Smartphone Cameras for Collaborative Road Advisories", IEEE Transactions on Mobile Computing, vol. 11, no. 5, pp. 707–723, 2012.

[KRI12] K. Kritayakirana and J. C. Gerdes, "Autonomous vehicle control at the limits of handling", International Journal of Vehicle Autonomous Systems, vol. 10, no. 4, pp. 271-296, 2012.

[KRO86] B. H. Krogh and C. E. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles", 1986 IEEE International Conference on Robotics and Automation, 1986, pp. 1664–1669.

[KUE07] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte Carlo localization in outdoor terrains using multilevel surface maps", Journal of Field Robotics, vol. 25, no. 6–7, pp. 346–359, 2008.

[LAF10] S. Lafuente-Arroyo, S. Salcedo-Sanz, S. Maldonado-Bascón, J. A. Portilla-Figueras, and R. J. López-Sastre, "A decision support system for the automatic management of keep-clear signs based on support vector machines and geographic information systems", Expert Systems with Applications, vol. 37, no. 1, pp. 767–773, 2010.

[LAR11] F. Larsson and M. Felsberg, "Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition", in Scandinavian Conference on Image Analysis (SCIA), 2011, pp. 238–249.

[LAR11b] "LaRA - La Route Automatisée", Available at http://www.lara.prd.fr/lara, 2011.

[LAU17] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction", American Control Conference (ACC), 2017, pp. 5586–5591.

[LAV01] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning", The International Journal of Robotics Research, vol. 20, no. 5, pp. 378–400, 2001.

[LEE07] B. M. Leedy, J. S. Putney, C. Bauman, S. Cacciola, J. M. Webster, and C. F. Reinholtz, "Virginia tech's twin contenders: a comparative study of reactive and deliberative navigation", Springer Tracts in Advanced Robotics, vol. 36, pp. 155–182, 2007.

[LEE08] J. Lee, C. Pippin, and T. Balch, "Cost based planning with RRT in outdoor environments", 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2008, pp. 684–689.

[LEE17] S. Lee, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. S.Hong, S.-H. Han, and I. S. Kweon, "VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition", in International Conference on Computer Vision (ICCV), 2017, pp. 1965–1973.

[LEE18] U. Lee, J. Jung, S. Jung, and D. H. Shim, "Development of a self-driving car that can handle the adverse weather", International Journal of Automotive Technology, vol. 19, no. 1, pp. 191–197, 2018.

[LEV10] J. Levinson, S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps", Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010. p. 4372-4378.

[LEV11] J. Levinson, J. Askeland, J. Becker, …, and S. Thrun, "Towards Fully Autonomous Driving: Systems and Algorithms", 2011 IEEE Intelligent Vehicles Symposium (IV), 2011, pp. 163-168.

[LI16] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications", IEEE/ASME Transactions on Mechatronics, vol. 21, no. 2, pp. 740–753, 2016.

[LI17] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles", Mechanical Systems and Signal Processing, vol. 87, pp. 118–137, 2017.

[LIM10] [D. A. de Lima and G. A. S. Pereira, "Um sistema de visão estéreo para navegação de um carro autônomo em ambientes com obstáculos", XVIII Congresso Brasileiro de Automática, 2010, pp. 224–231.

[LIM13] D. A. de Lima and G. A. S. Pereira, "Navigation of an autonomous car using vector fields and the dynamic window approach", Journal of Control, Automation and Electrical Systems, vol. 24, no. 1–2, pp. 106–116, 2013.

[LIN04] F. Lindner, U. Kressel, and S. Kaelberer, "Robust recognition of traffic signals", in Intelligent Vehicles Symposium (IV), 2004, pp. 49–53.

[LIU16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg,, "SSD: Single Shot MultiBox Detector", in European Conference on Computer Vision (ECCV), 2016, pp. 21–37.

[LYF18] Lyft, "We can change transportation together", 2018. Available: https://www.lyft.com/self-driving-vehicles. [Accessed: 17- Apr- 2018]

[LYR15] L. J. Lyrio, T. Oliveira-Santos, C. Badue, and A. F. De Souza, "Image-based mapping, global localization and position tracking using vg-ram weightless neural networks", IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 3603–3610.

[MAS14] C. M. Filho, D. F. Wolf, V. Grassi, and F. S. Osorio, "Longitudinal and lateral control for autonomous ground vehicles", in 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, no. Iv, pp. 588–593.

[MAT13] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic sign recognition: How far are we from the solution?", in International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–8.

[MCC06] J. C. McCall and M. M. Trivedi, "Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation", Transactions on Intelligent Transportation Systems, vol. 7, no. 1, pp. 20–37, 2006.

[MCN11] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice", IEEE International Conference on Robotics and Automation, 2011, pp. 4889–4895.

[MER13] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe, D. Duggins, and J. Gowdy, "Moving object detection with laser scanners", Journal of Field Robotics, vol. 30, no. 1, pp. 17–43, 2013.

[MER18] Mercedes-Benz, "Autonomous Driving: Mercedes-Benz", 2018. Available: https://www.mercedes-benz.com/en/mercedes-benz/innovation/autonomous-driving. [Accessed: 22- May- 2018]

[MIL12] M. J. Milford and G. F. Wyeth, "SeqSLAM: visual route-based navigation for sunny summer days and stormy winter nights", 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 1643–1649.

[MIT17] MIT, "Self-driving vehicles", 2017. Available: https://www.csail.mit.edu/research/self-driving-vehicles. [Accessed: 22- May- 2018]

[MNI10] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images", in European Conference on Computer Vision, 2010, pp. 210–223.

[MOB17] "Mobileye – Advanced Driver Assistance Systems", Available at https://www.mobileye.com/our-technology/adas/, 2017.

[MOB18] Mobileye, "About Us", 2018. Available: https://www.mobileye.com/about/. [Accessed: 17- Apr- 2018].

[MOG12] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey", IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1484–1497, 2012.

[MON08] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: the stanford entry in the urban challenge", Journal of Field Robotics, vol. 25, no. 9, pp. 569–597, 2008.

[MOU16] H. Mouhagir, R. Talj, V. Cherfaoui, F. Aioun, and F. Guillemard, "Integrating safety distances with trajectory planning by modifying the occupancy grid for autonomous vehicle navigation", IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 1114–1119.

[MOU17] H. Mouhagir, V. Cherfaoui, R. Talj, F. Aioun, and F. Guillemard, "Trajectory planning for autonomous vehicle in uncertain environment using evidential grid", IFAC-PapersOnLine, vol. 50, no. 1, pp. 12545–12550, 2017.

[MUT16] F. Mutz, L. P. Veronese, T. Oliveira-Santos, E. de Aguiar, F. A. Auat Cheein, and A. Ferreira De Souza, "Large-scale mapping in complex field scenarios using an autonomous car", Expert Systems with Applications, vol. 46, pp. 439–462, 2016.

[MUT17] F. Mutz, V. Cardoso, T. Teixeira, L. F. R. Jesus, M. A. Gonçalves, R. Guidolini, J. Oliveira, C. Badue, and A. F. De Souza, "Following the leader using a tracking system based on pre-trained deep neural networks", International Joint Conference on Neural Networks, 2017, pp. 4332–4339.

[NAA18] Navya, "La nouvelle mobilité est là. Autonome, partagée et électrique", 2018. Available: http://navya.tech. [Accessed: 17- Apr- 2018].

[NAB18] Navya, "Sensors", 2018. Available: http://navya.tech/en/autonom-en/autonom-shuttle/#storeLocator__bottomHalf. [Accessed: 17- Apr- 2018].

[NEV18] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards End-to-End Lane Detection: an Instance Segmentation Approach", arXiv preprint arXiv: 1802.05591, 2018.

[NGU12] T. N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M. M. Meinecke, "Stereo-camera-based urban environment perception using occupancy grid and object tracking", IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 1, pp. 154–165, 2012.

[NIS18] Nissan, "Nissan's self-driving car", 2018. Available: https://www.nissanusa.com/blog/autonomous-drive-car. [Accessed: 22- May- 2018]

[NOT11] T. Nothdurft, P. Hecker, S. Ohl, F. Saust, M. Maurer, A. Reschka, and J. R. Bohmer, "Stadtpilot: First fully autonomous test drives in urban traffic", in International Conference on Intelligent Transportation Systems (ITSC), 2011, pp. 919–924.

[NVI18] NVIDIA, "Driving Innovation: NVIDIA Automotive", 2018. Available: https://www.nvidia.com/en-us/self-driving-cars. [Accessed: 22- May- 2018]

[NVID18] NVidia, "NVIDIA DRIVE Scalable AI platform for Autonomous Driving", 2018. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/. [Accessed: 17- Apr- 2018].

[OLI17] G. L. Oliveira, N. Radwan, W. Burgard, and T. Brox, "Topometric localization with deep learning", arXiv preprint, 2017, arXiv:1706.08775.

[OMA10] M. Omachi and S. Omachi, "Detection of Traffic Light Using Structural Information", in International Conference on Signal Processing (ICSP), 2010, pp. 809–812.

[OND16a] I. Posner and P. Ondrúška, "Deep tracking: seeing beyond seeing using recurrent neural networks", Conference on Artificial Intelligence, pp. 3361–3367, 2016.

[OND16b] P. Ondrúška, J. Dequaire, D. Z. Wang, and I. Posner, "End-to-end tracking and semantic segmentation using recurrent neural networks", arXiv preprint arXiv: 1604.05091, 2016.

[OTS79] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Syst., Man Cybern., vol. SMC-9, no. 1, pp. 62–66, Jan. 1979

[OVE11] G. Overett and L. Petersson, "Large scale sign detection using HOG feature variants", in Intelligent Vehicles Symposium (IV), 2011, pp. 326–331.

[PAD16] B. Paden, M. Cap, S.Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles", IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, 2016.

[PET08] N. Pettersson, L. Petersson, and L. Andersson, "The histogram feature - a resource-efficient Weak Classifier", in Intelligent Vehicles Symposium (IV), 2008, pp. 678–683.

[PET09] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving", Autonomous Robots, vol. 26, no. 2–3, pp. 123–139, 2009.

[PET12] A. Petrovskaya, M. Perrollaz, L. Oliveira, L. Spinello, R. Triebel, A. Makris, J. D. Yoder, C. Laugier, U. Nunes, and P. Bessiere, "Awareness of road scene participants for autonomous driving", in Handbook of Intelligent Vehicles, London: Springer, pp. 1383–1432, 2012.

[PON18] Pony.ai, "Pony.ai", 2018. Available: https://www.pony.ai/. [Accessed: 17- Apr- 2018].

[PRO15] V. Protschky, C. Ruhhammer, and S. Feit, "Learning Traffic Light Parameters with Floating Car Data", in International Conference on Intelligent Transportation Systems (ITSC), 2015, pp. 2438–2443.

[RAD14] R. R. Radaelli, C. Badue, M. A. Gonçalves, T. O. Santos and A. F. De Sou-za, "A Motion Planner for Car-Like Robots Based on Rapidly-Exploring Random Trees", 4th Ibero-American Conference on Artificial Intelligence, 2014, pp. 469-480.

[RAD16] N. Radwan, G. D. Tipaldi, L. Spinello, and W. Burgard, "Do you see the bakery? leveraging geo-referenced texts for global localization in public maps", IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 4837–4842.

[RAM11] F. Ramm, J. Topf, and S. Chilton, "OpenStreetMap: using and enhancing the free map of the world", Cambridge: UIT Cambridge, 2011.

[RAM16] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent", International Journal of Robotics Research, vol. 35, no. 14, pp. 1717–1730, 2016.

[RED17] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger", in Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517–6525.

[REN17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

[REN18]   Renesas, "R-Car", 2018. Available: https://www.renesas.com/en-us/products/automotive-lsis/r-car.html. [Accessed: 17- Apr- 2018].

[RENE18]   Renesas, "Experience future-proof solutions for autonomous driving with the Renesas autonomy fleet", 2017. Available: http://renesasatces.com/autonomy/. [Accessed: 17- Apr- 2018].

[ROH16]   J. Rohde, I. Jatzkowski, H. Mielenz, J. M. Zöllner. "Vehicle pose estimation in cluttered urban environments using multilayer adaptive Monte Carlo localization", 19th International Conference on Information Fusion (FUSION), 2016, pp. 1774-1779.

[SAB10]   [1] M. M. S. Vitor B. Sabbagh, Elias J. R. Freitas, Guilherme M. M. Castro and G. A. S. P. Maurício F. Baleeiro, Tiago M. da Silva, Paulo Iscold, Leonardo A. B. Torres e Pereira, "Desenvolvimento de um sistema de controle para um carro de passeio autônomo", in XVIII Congresso Brasileiro de Automática, 2010, pp. 928–933.

[SAE16]   "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles", SAE International, 2016.

[SAM95]   C. Samson, "Control of chained systems application to path following and time-varying point-stabilization of mobile robots", IEEE Transactions on Automatic Control, vol. 40, no. 1, pp. 64–77, 1995.

[SCH18]   K. Doherty, J. Wang, and B. Englot, "Probabilistic map fusion for fast, incremental occupancy mapping with 3d hilbert maps", 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1011–1018.

[SEG09]   A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP", Robotics: Science and Systems V, 2009, vol. 5, pp. 168–176.

[SER13]   P. Sermanet, D. Eigen, M. Mathieu, X. Zhang, R. Fergus, and Y. Lecun, "OverFeat detection using deep learning", arXiv preprint arXiv: 1312.6229, 2013.

[SHA16]   G. Shafer, "A mathematical theory of evidence turns 40", International Journal of Approximate Reasoning, vol. 79, pp. 7–25, 2016.

[SHI16]   P. Y. Shinzato, T. C. dos Santos, L. A. Rosero, D. A. Ridel, C. M. Massera, F. Alencar, M. P. Batista, A. Y. Hata, F. S. Osorio, and D. F. Wolf, "CaRINA dataset: an emerging-country urban scenario benchmark for road detection systems", 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 41–46.

[SOO14]   S. Sooksatra and T. Kondo, "Red traffic light detection using fast radial symmetry transform", in International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2014, pp. 1–6.

[SOU13a]   A.F. De Souza, C. Fontana, F. Mutz, T.A. de Oliveira, M. Berger, A. Forechi, J. de Oliveira Neto, E. de Aguiar, and C. Badue, "Traffic sign detection with VG-RAM weightless neural networks", in International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–9.

[SOU13b]   O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning : a 2013 survey", International Conference on Industrial Engineering and Systems Management, 2013, pp. 1–8.

[SPA16]   R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 2161–2166.

[STA12]   J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition", Neural Networks, vol. 32, pp. 323–332, 2012.

[STA18]   Stanford University, "Center for automotive research at Stanford", 2018. Available: https://cars.stanford.edu/. [Accessed: 22- May- 2018]

[SUH17]   J. K. Suhr, J. Jang, D. Min, and H. G. Jung, "Sensor fusion-based low-cost vehicle localization system for complex urban environments", IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 5, pp. 1078–1086, 2017.

[TES18]   Tesla, "Full Self-Driving Hardware on All Cars", 2018. Available: https://www.tesla.com/autopilot. [Accessed: 22- May- 2018]

[THO91]   C. Thorpe, M. Herbert, T. Kanade, and S. Shafter, "Toward autonomous driving: the cmu navlab. ii. architecture and systems", IEEE Expert, vol. 6, no. 4, pp. 44–52, 1991.

[THR05]   S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics. Cambridge: MIT press, 2005.

[THR07]   S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: the robot that won the darpa grand challenge", in J. Field Robotics, vol. 23, no. June, 2007, pp. 1–43.

[THR10]   S. Thrun, "Toward robotic cars", Communications of the ACM, vol. 53, no. 4, p. 99, 2010.

[TOM18]   TomTom, "TomTom & Zenuity Join Forces On End-To-End Autonomous Driving System Software", 2018. Available: http://corporate.tomtom.com/releasedetail.cfm?releaseid=1053504. [Accessed: 17- Apr- 2018].

[TOR18]   Torc, "Our story: Giving birth to the future of autonomous technologies", 2018. Available: https://torc.ai/our-story. [Accessed: 17- Apr- 2018]. [Accessed: 17- Apr- 2018]

[TOY18]   Toyota, "Toyota: Automated Driving Technology", 2018. Available: http://www.toyota-global.com/innovation/automated_driving. [Accessed: 22- May- 2018]

[TOYT18]   The Wall Street Journal, "Toyota Announces New Company Devoted to Self-Driving Cars", 2018. Available: https://www.wsj.com/articles/toyota-announces-new-company-devoted-to-self-driving-cars-1519976923. [Accessed: 17- Apr- 2018].

[TRA18]   Transdev, "Transdev", 2018. Available: https://www.transdev.com/en/. [Accessed: 17- Apr- 2018].

[TRA18]   Transdev, "Métropole Rouen Normandie, Normandy Region, groupe Transdev, Groupe Renault, groupe Caisse des Dépôts and Groupe Matmut launch the Rouen Normandy Autonomous Lab project an on-demand mobility service on open roads with five autonomous electric vehicles for public use", 2017. Available: https://www.transdev.com/en/press/rouen-normandy. [Accessed: 17- Apr- 2018].

[TRE14]   G. Trehard, E. Pollard, B. Bradai, V. Bobigny, and F. F. Nashashibi, "Tracking both Pose and Status of a Traffic Light via an Interacting Multiple Model Filter", in International Conference on Information Fusion (FUSION), 2014, pp. 1–7.

[UBE15]   Uber, "Uber and Carnegie Mellon University: A Deeper Partnership", 2015. Available: https://www.uber.com/newsroom/cmupartnership/. [Accessed: 17- Apr- 2018]

[UBE18]   Uber, "Uber: Our Road to Self-Driving Vehicles", 2018. Available: https://www.uber.com/blog/our-road-to-self-driving-vehicles [Accessed: 22- May- 2018]

[UDA18]   Udacity, "The Udacity open source self-driving car project ", 2018. Available: https://github.com/udacity/self-driving-car. [Accessed: 17- Apr- 2018]

[URM08]   C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. "Red" Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge", Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[VER15]   L. de P. Veronese, E. de Aguiar, R. C. Nascimento, J. Guivant, F. A. A. Cheein, A. F. De Souza, and T. Oliveira-Santos, "Re-emission and satellite aerial maps applied to vehicle localization on urban environments", International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 4285–4290.

[VER16] L. de P. Veronese, J. Guivant, F. A. A. Cheein, T. Oliveira-Santos, F. Mutz, E. de Aguiar, C. Badue, and A. F. De Souza, "A light-weight yet accurate localization system for autonomous cars in large-scale and complex environments", 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 520–525.

[VERG16] The Verge, "This is the LeEco LeSEE Pro self-driving electric car ", 2016. Available: https://www.theverge.com/2016/10/19/13335242/eeco-lesee-pro-self-driving-electric-car-photos. [Accessed: 17- Apr- 2018]

[VIR18] Virginia Tech, "Center for advanced automotive research", 2018. Available: https://www.vtti.vt.edu/research/caar/index.php. [Accessed: 22- May- 2018]

[VIS16] A. Viswanathan, B. R. Pires, and D. Huber, "Vision-based robot localization across seasons and in remote locations", IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 4815–4821.

[VIS18] Visteon, "Visteon Introduces DriveCore™ Autonomous Driving Platform to Accelerate Adoption of Self-Driving Technology", 2018. Available: https://www.visteon.com/media/newsroom/2018/180109_story 1.html. [Accessed: 17- Apr- 2018].

[VIV17] R. Vivacqua, R. Vassallo, and F. Martins, "A low cost sensors approach for accurate vehicle localization and autonomous driving application", Sensors, vol. 17, no. 10, p. 2359, 2017.

[VOLK18] Volkswagen Group, "Mobility for All – at the Touch of a Button: Volkswagen Group Presents the Latest Version of SEDRIC", 2018. Available: http://www.discover-sedric.com/en. [Accessed: 22- May- 2018]

[VOLV18] Volvo, "Volvo: Autonomous Driving". Available: https://www.volvocars.com/en-om/about/our-innovation-brands/intellisafe/autonomous-driving, 2018. [Accessed: 17-Apr- 2018].

[VU09] Trung-Dung Vu and O. Aycard, "Laser-based detection and tracking moving objects using data-driven Markov Chain Monte Carlo", IEEE International Conference on Robotics and Automation, 2009, pp. 3800–3806.

[WAN15] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2D LIDAR", International Journal of Robotics Research, vol. 34, no. 7, pp. 1039–1063, 2015.

[WAY18] Waymo, "We're building the world's most experienced driver", 2018. Availble: https://waymo.com/

[WAY18b] Waymo, "Waymo 360° Experience: A Fully Self-Driving Journey", 2018. Available: https://www.youtube.com/watch?v=B8R148hFxPw. [Accessed: 17- Apr- 2018]

[WEG15] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler, "Road networks as collections of minimum cost paths", ISPRS Journal of Photogrammetry and Remote Sensing, vol. 108, pp. 128–137, 2015.

[WEI13] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform", in Intelligent Vehicles Symposium (IV), 2013, pp. 763–770.

[WOL17] R. W. Wolcott and R. M. Eustice, "Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving", The International Journal of Robotics Research, vol. 36, no. 3, pp. 292–319, 2017.

[WU12] T. Wu, and A. Ranganathan, "A practical system for road marking detection and recognition", in Intelligent Vehicles Symposium (IV), 2012, pp. 25–30.

[XIA18] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling", arXiv preprint arXiv: 1805.04687, 2018

[XIN14] J. Xin, C. Wang, Z. Zhang, and N. Zheng, "China Future Challenge: Beyond the Intelligent Vehicle", IEEE Intelligent Transportation Systems Society Newsletter, vol. 16, pp. 8–10, 2014.

[XU12] Wenda Xu, Junqing Wei, J. M. Dolan, Huijing Zhao, and Hongbin Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles", IEEE International Conference on Robotics and Automation, 2012, pp. 2061–2067.

[XU15] K. Na, J. Byun, M. Roh, and B. Seo, "RoadPlot-Datmo: moving object tracking and track fusion system using multiple sensors", International Conference on Connected Vehicles and Expo, 2015, pp. 142–143.

[XU17] Y. Xu, V. John, S. Mita, H. Tehrani, K. Ishimaru, and S. Nishino, "3D point cloud map based vehicle localization using stereo camera", IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 487–492.

[XUE17] J. Xue, D. Wang, S. Du, D. Cui, Y. Huang, and N. Zheng, "A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars", Frontiers of Information Technology & Electronic Engineering, vol. 18, no. 1, pp. 122–138, 2017.

[YEN13] S. Yenikaya, G. Yenikaya, and E. Düven, "Keeping the vehicle on the road", ACM Computing Surveys, vol. 46, no. 1, pp. 1–43, 2013.

[YOO15] S. Yoon, S. Yoon, U. Lee, and D. H. Shim, "Recursive path planning using reduced states for car-like vehicles on grid maps", IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 5, pp. 2797–2813, 2015.

[ZEN18] Volvo Car Group, "Volvo Cars and Autoliv announce the launch of Zenuity ", 2017. Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/202044/volvo-cars-and-autoliv-announce-the-launch-of-zenuity. [Accessed: 17- Apr- 2018].

[ZHA12] P. Zhao, J. Chen, Y. Song, X. Tao, T. Xu and T. Mei, "Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID", International Journal of Advanced Robotic Systems, vol. 9, no. 2, 44, 2012.

[ZHA13] L. Zhang, Q. Li, M. Li, Q. Mao, and A. Nüchter, "Multiple vehicle-like target tracking based on the Velodyne LIDAR", in IFAC Proceedings Volumes, vol. 46, no. 10, pp. 126–131, 2013.

[ZHA14] Y. Zhang, J. Xue, G. Zhang, Y. Zhang, and N. Zheng, "A multi-feature fusion based traffic light recognition algorithm for intelligent vehicles", in Chinese Control Conference (CCC), 2014, pp. 4924–4929.

[ZHU16] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-Sign Detection and Classification in the Wild", in Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2110–2118.

[ZIE08] J. Ziegler and M. Werling, "Navigating car-like robots in unstructured environments using an obstacle sensitive cost function", in 2008 IEEE Intelligent Vehicles Symposium, 2008, pp. 787–791.

[ZIE14a] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, Thao Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making Bertha Drive—An Autonomous Journey on a Historic Route", Intelligent Transportation Systems Magazine, vol. 6, no. 2, pp. 8–20, 2014.

[ZIE14b] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha; A local, continuous method", IEEE Intelligent Vehicles Symposium, 2014, pp. 450–457.