Computers & Graphics 84 (2019) 173–184

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on SIBGRAPI 2019

Handling pedestrians in self-driving cars using image tracking and alternative path generation with Frenét frames

Renan Sarcinelli[a,*], Rânik Guidolini[a], Vinicius B. Cardoso[a], Thiago M. Paixão[a,b], Rodrigo F. Berriel[a], Pedro Azevedo[a], Alberto F. De Souza[a], Claudine Badue[a], Thiago Oliveira-Santos[a]

[a] Laboratório de Computação de Alto Desempenho, Departamento de Informática, Universidade Federal do Espírito Santo, Brazil
[b] Instituto Federal do Espírito Santo, Brazil

ARTICLE INFO

Article history:
Received 16 April 2019
Revised 28 July 2019
Accepted 15 August 2019
Available online 21 August 2019

Keywords:
Pedestrian tracking
Crosswalk
Convolutional neural networks
Deep learning
Self-driving car

ABSTRACT

The development of intelligent autonomous cars is of great interest. A particular and challenging problem is to handle pedestrians, for example, crossing or walking along the road. Since pedestrians are one of the most fragile elements in traffic, a reliable pedestrian detection and handling system is mandatory. The current pedestrian handling system of our autonomous cars suffers from the limitation of the pure detection-based systems, i.e., it limits the autonomous car system to make decisions based only on the very present moment. This work improves the pedestrian handling systems by incorporating an object tracker with the aim of predicting the pedestrian's behavior. With this knowledge, the autonomous car can better decide the time to stop and to start moving, providing a more comfortable, efficient, and safer driving experience. The proposed method was augmented with a path generator, based on Frenét Frames, and incorporated to our self-driving car in order to enable a better decision making and to enable overtaking pedestrians. The behaviour of our self-driving car was evaluated in both simulated and real-world scenarios. Results showed the proposed system is safer and more efficient than the system without tracking functionality due to the early decision capability.

1. Introduction

The development of intelligent autonomous cars (or self-driving cars) is of large interest to the automotive industry, and has been widely addressed in artificial intelligence literature [1]. Typically, self-driving cars rely on several sensors (e.g., LiDAR, RADAR, or cameras) to provide the necessary data for creating an internal representation of the environment, and then to make decisions about how to behave within different traffic scenarios. For instance, a self-driving car is required to perceive static and moving objects in its vicinity and avoid collision with them. In particular, a self-driving car should regard very carefully pedestrians crossing or moving around roads, especially in the areas intended for this purpose, that is, crosswalks (Fig. 1). Several works in the literature have addressed the problem of detecting pedestrians in traffic scenes [2–5]. However, performing detection-only leaves to the remainder of the self-driving car system (in particular the Behavior Selector module [1], Fig. 3) all the responsibility for making necessary decisions to handle pedestrians. To reduce the burden of the Behavior Selector, temporal information may be used for keeping track of the state (position, velocity, and orientation) of pedestrians [6,7], which allows the prediction of the near-future pedestrians' states(~3–5 s ahead).

Obviously, it is unfeasible to properly predict all possible near-future pedestrians' states due to the nature of human behavior. Nevertheless, in most scenarios, the pedestrian's position, velocity, and orientation can facilitate the behavior selection decision making as, for example, in cases that a pedestrian is standing on the pavement near a crosswalk without intending to cross it.

Currently, the majority of the developed systems for handling pedestrians are restricted to the detection and tracking of pedestrians in a short-distance range. For longer distances, some works in the literature proposed coupling tracking systems with deep learning detection systems based on camera images [8–10], but none of these approaches provide precise localization of the pedestrians in the world this information is of vital importance to the Behavior Selector.

Previous work [4] has already presented a pedestrian handling system for a self-driving car in order to deal with pedestrians in crosswalks. Besides of being restricted to crosswalk areas, the handling system has also the same limitation of the pure detection-based systems. However, it is able to pinpoint the pre-

* Corresponding author.
E-mail address: rsarcinelli@lcad.inf.ufes.br (R. Sarcinelli).

https://doi.org/10.1016/j.cag.2019.08.004
0097-8493/

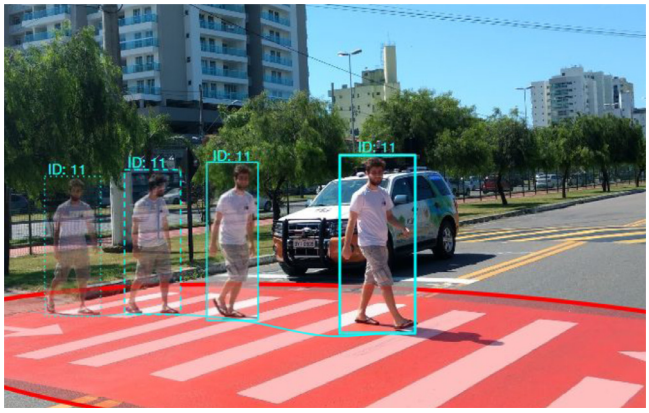

Fig. 1. The self-driving car handling a pedestrian in a crosswalk.

cise position of the pedestrian in the 3D world, as required by the Behavior Selector. Basically, it integrates 3D LiDAR point clouds with pedestrian detections in 2D images performed by a Convolutional Neural Network (CNN), so that the positions of the pedestrians in the 3D world are estimated "from scratch" at every image frame. This work improves on current pedestrian handling systems [4,8–10] by incorporating a pedestrian tracking system, which aims to predict the near-future pedestrians' states. Tracking the pedestrians helps predicting whether the pedestrian intends to cross the road or walk along the road, for example. With this knowledge, the Behavior Selector can better decide when to stop, start moving, deviate from the pedestrian, etc., providing a more comfortable, efficient, and safer driving experience.

In this work, the tracking system presented in [11] was integrated to the pedestrian detection system presented in [4], and incorporated to our self-driving car. Together, these systems form now a Pedestrians' State Estimator (PSE), which was added to the Moving Obstacles Tracker (MOT) module (Fig. 3) of our self-driving car. Also, a Crosswalk State Estimator (CSE) and an Alternative Paths Generator (APG), based on Frenét Frames [12], were added to the Path Planner module (Fig. 3) of our self-driving car. The CSE determines the current state of the crosswalk ahead in the car's path (busy or free) and the APG provides many alternatives of paths based on the single path we used previously. Using the alternative paths, the crosswalks' states and the near-future pedestrians' states, the Behavior Selector prunes the many alternative paths into a smaller set of valid paths and, considering traffic rules and safety parameters, selects the best plan available for crosswalks or other traffic scenarios involving pedestrians.

To evaluate the proposed methods for handling pedestrians, the behaviour of our self-driving car with the new approach and with the old approach was examined in both simulated and real-world scenarios. The conducted experiments depict normal conditions, as well as potentially dangerous situations that were not properly handled by the old approach. Results show that the proposed approach for handling pedestrians is safer and more efficient than the old one, especially due to the prediction of near-future pedestrians' states and availability of alternative paths.

2. Related work

Pedestrian detection has been widely addressed in the past years, especially in the context of autonomous vehicles. Li et al. [5] noticed that the features learned by deep networks may have a large variance depending on the size of the pedestrian in the image. To improve pedestrian detection, they proposed a method that combines networks trained for people in different scales. Benenson et al. [3] presented an algorithm that handles different scales in monocular camera images to improve the detection speed, achieving 50 frames per second (fps), as well as presented an improvement that takes advantage of better exploiting the depth information in stereo images, making the detection even faster. Guidolini et al. [4] used YOLOv2 CNN to detect people in a monocular camera image and fused this data with a LiDAR-generated point cloud to obtain precise position of each pedestrian. Premebida et al. [2] investigated several classifiers to detect pedestrians based only on LiDAR features. Although all the described methods can be used to detect pedestrians in traffic scenes, only Guidolini et al. [4] and Premebida et al. [2] can also provide accurate localization of the detected pedestrians. However, the aforementioned methods only detect pedestrian using single frame information and, therefore, do not provide temporally dependent data of a pedestrian, such as velocity.

To account for temporality, some works in the context of autonomous driving combine detection with tracking algorithms. Wang et al. [6] detect and track pedestrians from LiDAR data by clustering the point cloud, classifying each cluster using support vector machine (SVM), and then applying a simple Kalman filter to enable tracking. Baek et al. [7] proposed a lightweight system in which four fisheye images are combined to track objects near to an autonomous car in a 360 degrees field of view. While in [7] the fisheye camera makes far objects too small to be detected, the Wang's system [6] can be used for long-range detection. Nevertheless, the performance in [6] relies on the amount of points available for clustering and further classification. Therefore, far object detection should leverage higher resolution sensors or a combination of sensors.

Recent works developed networks that are able to retrieve more sophisticated and complex information. Segmentation networks, such as [13,14], are based on a convolutional and deconvolutional process to classify each pixel individually. This provides a detailed contour of each object instead of a region delimited for a bounding box. However, this process demands more computational efforts and using such networks in real-time applications is almost impracticable nowadays. Other interesting related works focus on the human near-future pose prediction from videos [15–18] and 2D pose regression [19,20]. Near-future pose prediction works are based on encoding-decoding networks. They take information from the most recent past to predict the future, using recurrent networks or other methods. Pose regression focuses on detecting key points to predict the person's joints and estimate his pose. Both techniques could be applied to a pedestrian handling system and would provide more complete information than a image detection an tracking system. Nevertheless, they are specific for detecting and predicting humanoid poses, and would require additional networks to detect other traffic elements such as cars or signs. Since the self-driving car has to consider many elements, it is desirable that a single network detects them all.

Several tracking methods can be incorporated into an autonomous vehicle system in order to handle pedestrians. Some of the state-of-the-art pedestrian trackers were evaluated on the challenge promoted by the authors of the Multiple Object Tracking (MOT) benchmark [21], whose objective is tracking multiple people in monocular camera images. For instance, Sheng et al. [10] fuse high-level detections and low-level image evidence for target association. Leal-TaixÃ©et al. [8] use a siamese deep network to learn complex associations between input image patches, and then feed a simple tracker algorithm with the learned matching probabilities. Henschel et al. [9] estimated the position of a track in the image by fusing head and body detections. Despite the advance in the field, these methods do not provide accurate pedestrians' position in the 3D world, which is an important information for collision avoidance.

Handling pedestrians is an essential task for any autonomous vehicles and some researches can be found in the literature. For example, Li et al. [22] proposed a real-time trajectory planner for an autonomous car including obstacle avoidance and specific treatment for pedestrians. Their system was tested in a real world scenario using a self-driving car. Rentería [23] modeled and controlled an electric vehicle at 50km/h in a simulated environment to avoid a pedestrians. Fernández et al. [24] described a autonomous navigation and avoidance system for a micro-bus. During the experiments, the autonomous micro-bus performed some laps in a private circuit including a pedestrian simulated by a dummy. In addition to the academic research, many companies (such as Google, Uber, and others) are also investigating this problem. The best results of these researches are available on their autonomous vehicles. Some of them are already under testing and performing demonstrations. For instance, demonstration videos for pedestrians handling and other functionalities can be found at Torc Robotics web site [25] and at Waymo [26] channel. Unfortunately, most of these works are proprietary and, therefore, are not available in the literature.

In this work, advanced computer graphics techniques are used for integrating information of different sensors, detecting pedestrians, tracking their trajectory, and, finally, planning an alternative and smooth trajectory for an autonomous car. All these steps comprise a complex system for enabling an autonomous car handling pedestrians.

3. Self-driving car's architecture

Our self-driving car (Fig. 2) is named Intelligent Autonomous Robotic Automobile (IARA) and it was developed at Laboratório de Computação de Alto Desempenho (LCAD) from the Federal University of Espírito Santo (Universidade Federal do Espírito Santo UFES). The car is a Ford Escape Hybrid, which was adapted to enable electronic actuation of the steering, throttle and brakes, and to provide power supply for computers and sensors. These computers and sensors include 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).

The architecture of the autonomous system of our self-driving car can be roughly divided into two main parts: the Perception System and the Decision-Making system. The Perception System is currently divided into four main modules responsible for the tasks of localizing the self-driving car, mapping of obstacles, detection and tracking of moving obstacles, and detection and recognition of traffic signalization. The Decision-Making system is currently divided into six main modules responsible for the tasks of route planning, path planning, behavior selection, motion planning, obstacle avoidance, and control.

Fig. 2 shows a block diagram of the autonomous system architecture of our self-driving car. The Mapper module [27] constructs occupancy grid maps [28] that represent obstacles of the environment. The cells of an obstacle map store the probability of the associated regions of the environment being occupied by obstacles. The Mapper operates in offline and online modes. In the offline mode, the Mapper receives as input data from multiple sensors (odometer, LiDAR, IMU, and GNSS), which are stored while our self-driving car is driven by a driver along a path of interest - this only needs to be done once. It then estimates the self-driving-car's states along the path and the values of the offline map cells (Offline Map) using the GraphSLAM algorithm [27,28]. In the online mode, the Mapper receives as input the offline map, the sensors' data (odometer, LiDARs, and IMU) and the car's state, and computes the online map. The online map is a mix of information present on the offline map and an occupation grid map computed online using the sensors' data and the car's state. In the online mode, the Mapper also computes the Obstacle Distance Grid Map (ODGM), in which each cell contains the coordinates and the distance to the nearest obstacle.

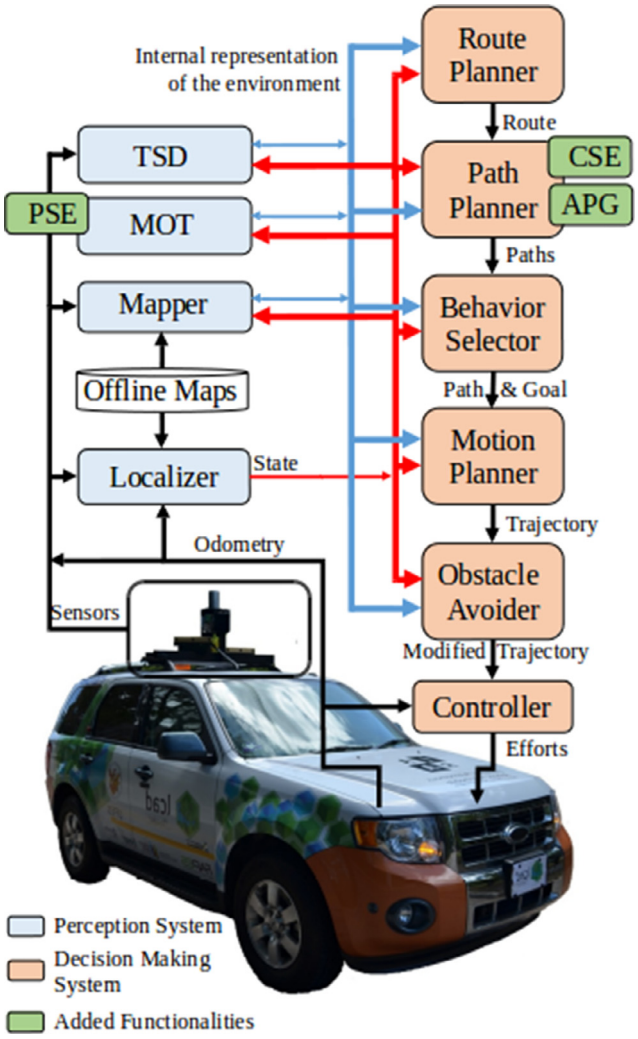

Fig. 2. Overview of the autonomous system architecture of our self-driving car. The perception and decision making systems are shown as a collection of modules of different colors, where TSD denotes Traffic Signalization Detector, and MOT, Moving Obstacles Tracker. The added functionalities presented in this paper are shown in green, where CSE denotes Crosswalk State Estimator, PSE denotes Pedestrians' State Estimator and APG denotes Alternative Paths Generator.

The Localizer module [29] estimates the self-drivingcar's state in relation to the offline map. When initialized (global location [28]), the Localizer receives as input the initial state of the car from the GNSS RTK an approximation of the initial state of the car can also be provided manually. At each subsequent stage, it receives as input the last state of the car, the offline map, and the sensors' data (odometer, LiDAR, and IMU), and calculates a local occupancy grid map and estimates the car's state, by combining the local map with the offline map, using for this a particle filter.

The Moving Obstacles Tracker (MOT) module receives the ODGM, the self-driving-car's state and a path segment, and calculates the pose and velocity of the nearest moving obstacles being tracked. The MOT also handles obstacles in motion by altering the longitudinal velocity of the self-driving car in order to (a) maintain adequate distance from obstacles in motion and (b) allow a

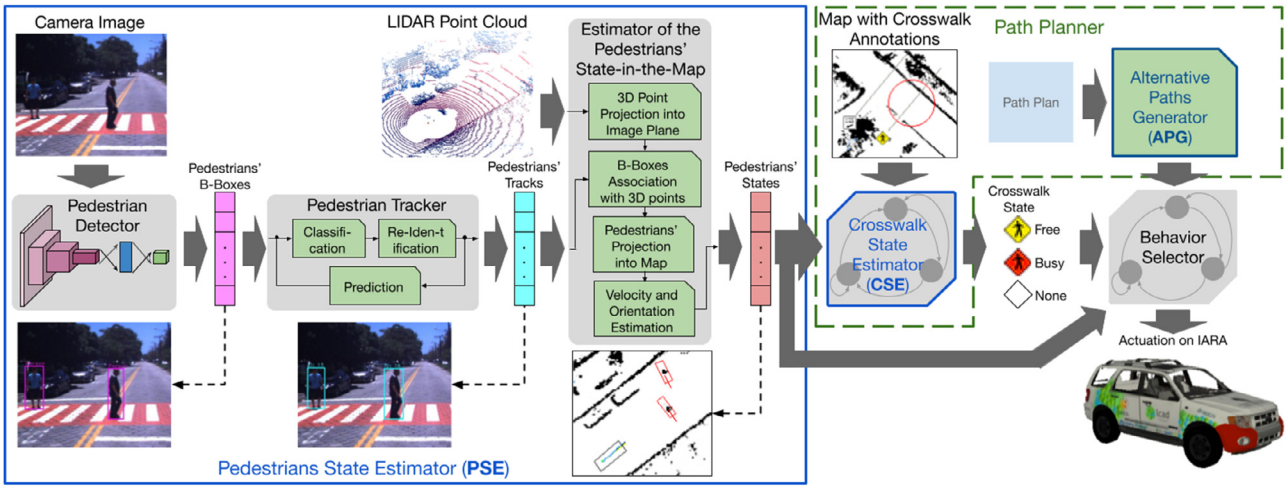

Fig. 3. Overview of the architecture of the proposed new approach for handling pedestrians. The Pedestrians' State Estimator (PSE) uses a CNN-based Pedestrian Detector to detect candidate pedestrian-regions in images (Pedestrians' B-Boxes). Subsequently, these Pedestrians' B-Boxes are feed to a Pedestrian Tracker, which creates or updates Pedestrians' Tracks in the 2D image space. The Pedestrians' Tracks are feed to the Pedestrians' State-in-the-Map Estimator, which: (i) projects 3D point clouds of a LiDAR into the images, (ii) associates the Pedestrians' B-Boxes with the LiDAR 3D points and calculates their position in the 3D world; (iii) projects the 3D pedestrians' positions into the 2D world map, and (iv) computes the Pedestrians' States (pedestrians' positions, orientations and velocities). The Pedestrians' States and crosswalks' positions are feed to the Crosswalk State Estimator (CSE), which determines the Crosswalk State (Free or Busy). The Pedestrians' States, Crosswalk State and Alternative Paths are feed to the Behavior Selector, which actuates in IARA to handle the situation accordingly. The dashed arrows indicate the views of partial results in different points of the system.

smooth steering experience. In addition, it changes the ODGM seen by other modules of the self-driving car, in order to hide obstacles in motion from then on, which significantly simplifies its implementation.

The Traffic Signalization Detector (TSD) [30–32] receives the sensors' data (camera and LiDAR) and the self-driving-car's state, and detects the position of traffic signalizations and recognizes their class or status. Traffic signalization includes horizontal traffic signalizations (lane markings) and vertical signalizations (speed limits, traffic lights, etc.) that must be recognized and obeyed by self-driving cars.

The Path Planner creates a path from the current state of the self-driving car to a local goal state. The Path Planner receives the car's state and a Road Definition Data File (RDDF) file as input. The RDDF is composed of a sequence of car's states, which are stored while our self-driving car is driven by a human along a path of interest. The Path Planner extracts a path from the RDDF, which consists of a sequence of equally spaced states that goes from the current state of the car to a goal state a few meters ahead.

The Behavior Selector module establishes a local goal state in the path according to the real-world instantaneous scenario. The Behavior Selector receives the ODGM, the self-driving car's state, the path, and a set of annotations (made previously) from the offline map, such as speed bumps', safety barriers', crosswalks' and speed limits' positions, as well as traffic lights' positions and states. It then sets a local goal state along the path a few seconds ahead of the current state of the car, and adjusts the pose and velocity of the goal, so that the car behaves according to the instantaneous scenario, such as stopping at traffic lights, and reduces velocity or stop on busy crosswalks, etc.

The Motion Planner module [33] calculates a trajectory from the current state of the self-driving car to a local goal state in the path. The Motion Planner receives as input the ODGM, the car's state, the path and the goal state in the path. It then calculates a path from the current state of the car to the goal state in the path that follows the path and avoids the obstacles represented in the ODGM using a predictive approach. The trajectory consists of a sequence of control commands, each composed of linear velocity, steering angle and execution time.

The Obstacle Avoider module [34] checks and eventually changes the trajectory if a collision is to be avoided. The Obstacle Avoider receives as input the ODGM, the self-driving-car's state and the trajectory. It then simulates the execution of the trajectory along the ODGM. If the trajectory collides with an obstacle, the Obstacle Avoider decreases the velocity commands of the trajectory to avoid the accident.

Finally, the Controller module [35] calculates the commands that will be sent directly to the actuators of the self-driving car in the steering wheel, accelerator and brakes, employing a Proportional Integral Derivative (PID) controller. The Controller receives as input the path and the odometer data.

The added functionalities presented in this paper are shown in green in Fig. 2, where CSE denotes Crosswalk State Estimator, PSE denotes Pedestrians' State Estimator and APG denotes Alternative Paths Generator. The PSE is part of the MOT module and provides Pedestrians' States to the Path Planner and Behavior Selector modules. The CSE is part of the Path Planner module and provides the Crosswalk State to the Behavior Selector module. The APG is part of the Path Planner module and provides Alternative Paths for the Behavior Selector to choose, considering pedestrians and other challenges of the real world. PSE, CSE and APG are detailed in the sections below.

4. Pedestrian's state estimator (PSE)

To estimate the state of pedestrians, the proposed approach (see Fig. 3 for an overview) first uses a CNN-based Pedestrian Detector (YOLOv3 [36,37]) to detect candidate pedestrian-regions in images, i.e., Pedestrians' Bounding Boxes, or Pedestrians' B-Boxes for short. Subsequently, these Pedestrians' B-Boxes are feed to a Pedestrian Tracker [11], which creates or updates Pedestrians' Tracks (if existing) in the 2D space of images. The Pedestrians' Tracks are feed to the Pedestrians' State-in-the-Map Estimator, which: (i) projects 3D point clouds of a LiDAR into the images, (ii) associates the Pedestrians' B-Boxes with the LiDAR 3D points in this way, finding their position in the 3D world and projecting these 3D world positions into the 2D world map, and (iii) computes the Pedestrians' States, which is a vector of triplets (position, orientation and velocity) with one element per pedestrian tracked. We describe each one of these steps in the following subsections.

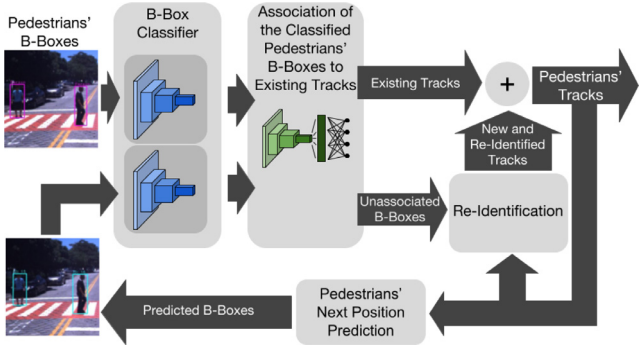

Fig. 4. Overview of the tracking system proposed in [11].

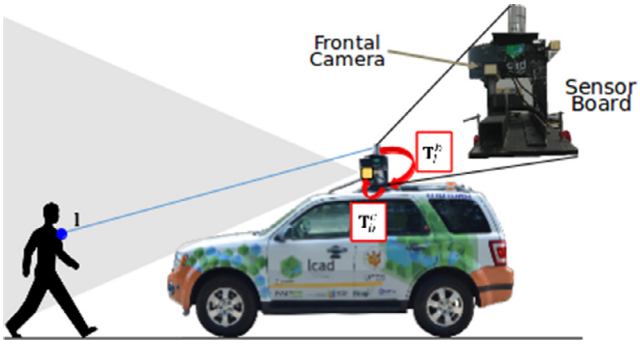

Fig. 5. Projection of 3D points captured by LiDAR into the image plane.

4.1. Pedestrian detector

Since in the autonomous driving context the pedestrians are not the only object of interest to be detected, simpler networks that can handle multiple object detection are desirable to reduce the computational overhead. YOLOv3 CNN [36,37] is one of the state-of-the-art systems for real-time object detection that is based on a feature extractoion and a bounding box (B-Box) regression. Therefore, it has been chosen as pedestrian detector. The feature extractor was pre-trained on ImageNet [38], whereas the bounding box regressor was trained on the general-purpose dataset Common Objects in Context (COCO) [39]. The pre-trained YOLOv3 was used in this work without any modification. In other words, we assume that the original training process is enough for the detection of pedestrians. Therefore, for each input image, YOLOv3 outputs a list of objects' bounding boxes and their respective classes. Only the objects classified as pedestrians are used in this work, even though other object classes could also be used.

4.2. Pedestrian tracker

To keep track of pedestrians, the proposed approach combines YOLOv3 with a multi-object tracking system proposed by Long Chen et al. [11] for the 2017 Multiple Object Tracking Challenge (MOT17 [21]). We decided to use this tracker because it operates in real-time and achieved good accuracy in MOT17. In summary, their system receives candidate bounding-boxes of people present in scenes and, then, classifies each bounding box as either a new person, or as a previously seen person that is currently being tracked by the system.

The block diagram of Fig. 4 illustrates the Pedestrian Tracker operation. First, it receives Pedestrians' B-Boxes provided by YOLOv3 as candidate people present in the scene. Second, it verifies whether the candidates in the B-Box are persons by using a Region-Based Fully Convolutional Neural Network (R-FCNN [40]) and assigns a classification score to each candidate (please note that the Chen et al. [11] R-FCNN B-Box Classifier is trained only for true-false candidate classification, not for bounding box regression; we used YOLOv3 for bounding box regression). Third, the Pedestrian Tracker performs the Association of the Classified Pedestrians' B-Boxes to Existing Tracks. For that, it extracts and compares appearance features of these B-Boxes using the neural network proposed in [41]. Finally, if there are no existing tracks, or a B-Box detected by YOLOv3 is not associated with any Existing Tracks, the Unassociated B-Boxes are considered for Re-Identification. This may result in New or Re-Identified Tracks; please see [11] for details.

After all these procedures, the Pedestrian Tracker outputs Pedestrians' Tracks, which are represented by bounding boxes and associated labels (i.e., IDs). The bounding boxes give the localization of the pedestrians in each image, whereas the labels allow identification of each pedestrian throughout a series of image frames. The output is also used as a feedback for Pedestrian Next Position Prediction and updating appearance features used in the Re-Identification.

Since YOLOv3 accurately detect the pedestrians, we adopted a lower classification threshold for the B-Box Classifier (0.1 instead of the default of 0.4).

4.3. Estimator of the Pedestrians' State-in-the-Map

To compute precise pedestrians' positions, image and LiDAR data were fused as in [4]. LiDAR can give accurate positions, but the greater the distance between the object and the laser, the sparser the point cloud will be. This situation is a problem for methods relying solely on LiDAR points [2,6]. On the other hand, image-based detectors [3,5] have shown the ability of detecting small objects, which can be explored to detect people far from the camera. Most of these methods could be integrated with stereo vision for depth estimation; however, their precision is not as accurate as the LiDAR's.

The Estimator of the Pedestrians' State-in-the-Map projects the 3D points captured by a LiDAR into the image plane of the camera, associates these points with B-Boxes of Pedestrians' Tracks, projects the 3D points belonging to pedestrians into the map, and estimates the pose and velocity of the pedestrians in map coordinates using the history of the pedestrians' poses in the map. We detail each one of these steps in the following.

To describe how we perform the projection of 3D points captured by a LiDAR into the image plane of the camera, we are going to use the following notation.

Notation: We write scalars in lowercase italic letters (a), vectors in bold lowercase ($\mathbf{a}$), and matrices using boldface capitals ($\mathbf{A}$). Scalars may have subscripts to better characterize them (i.e., a_x). A 3D rigid-body transformation matrix that takes points from coordinate system a to coordinate system b will be denoted by $\mathbf{T}_a^b$, with $\mathbf{T}$ for 'transformation'. We have 3 coordinate systems: LiDAR, camera and sensor board (in our self-driving car, the sensors are mounted in a sensor board). They are all oriented as follows: x = forward, y = left, z = up. The camera images are rectified prior to use.

A 3D point, $\mathbf{l}$, in LiDAR coordinates is projected to a point, $\mathbf{c}$, in camera coordinates using Eq. (1),

$$\mathbf{c} = \mathbf{T}_b^c \mathbf{T}_l^b \mathbf{l} \tag{1}$$

where $\mathbf{T}_l^b$ is the transformation matrix that maps a point from LiDAR coordinates onto the sensor board coordinates, and $\mathbf{T}_b^c$ is the transformation matrix that takes a point from the sensor board coordinates to the camera coordinates (Fig. 5). The point $\mathbf{c} = (x, y, z)$ is projected to a point $\mathbf{p} = (u, v)$ in the camera image plane using

Eqs. (2) and (3) [42],

$$u = \frac{f_u}{s}\frac{y}{x} + c_u \tag{2}$$

$$v = \frac{f_v}{s}\frac{-z}{x} + c_v \tag{3}$$

where f_u and f_v are the camera's focal lengths in meters, s is the pixel size in meters, and c_u and c_v are the coordinates of the camera's principal point in pixels.

Using Eqs. (1), (2), and (3) we filter the LiDAR points that lie outside the camera's field. Using information of the mapping system (i.e., which LiDAR points hit obstacles or not [27]), we are also able to remove points that hit the pavement. We then associate the remaining points, $\mathbf{p}$, with the B-Boxes of Pedestrians' Tracks by checking which of these points lie within the B-Boxes. At this stage, the cloud of points is partitioned in N groups, where N is the number of detected pedestrians. However, a partition potentially includes 3D points related to other objects, since the bounding boxes are approximate representations. To extract only pedestrian's points, each partition is clustered and the largest cluster (i.e., that with the highest number of points) is assigned as the pedestrian. The pedestrian's 3D position is given by the average of the associated cluster points.

We project the 3D points representing each pedestrian into the map using the following equation:

$$\mathbf{m} = \mathbf{T}_i^w \mathbf{T}_b^i \mathbf{T}_l^b \mathbf{l} \tag{4}$$

where $\mathbf{T}_l^b$ is the transformation matrix that takes a point from LiDAR coordinates to the sensor board coordinates,$\mathbf{T}_b^i$ is the transformation matrix that takes a point from the sensor board coordinates to the car coordinates (center of the rear axle), and $\mathbf{T}_i^w$ is the transformation matrix that takes a point from the car coordinates to the world coordinates. In the autonomous system of our self-driving car, a world point is noted in the Universal Transverse Mercator (UTM) coordinate system. Since the map is only 2D, the z coordinate of $\mathbf{m}$ was discarded.

Pedestrians are uniquely identified with their respective track ID provided by the Pedestrian Tracker. For each ID, the system keeps a list with the last 10 positions of the respective pedestrian. The Pedestrians' States, i.e., the velocity (scalar) and orientation (angle with respect to the x axis of the map) of the pedestrian in the map are computed using the (x, y) coordinates of the elements of each list. For the velocity, we first calculate a set of 9 base velocities (distance/time interval) by pairing the 10 consecutive observations (i.e., the positions stored in the list). These estimations are filtered so that the outliers are removed. In this work, outliers are defined as values beyond 1.5 standard deviations away from the mean. The final velocity is then computed as the average of the remaining values. To estimate the orientation, a line is fitted onto the pedestrian's positions, resulting in two possible (complementary) orientations. The ambiguity is resolved by choosing the orientation that is closer to the average orientation, considering the last 9 poses.

The Pedestrians' States computed by the PSE are used as input of the Crosswalk State Estimator (CSE) and the Behavior Selector module. CSE is detailed in the section below.

5. Crosswalk state estimator (CSE)

In the autonomous system of our self-driving car (Fig. 2), the Path Planner is the module responsible for publishing to the remaining modules the traffic signalization that must be dealt with at each occasion. This is because, in the process of computing a path, this module has to examine the Offline Map, which in our self-driving car stores traffic signalization that can be stored offline.

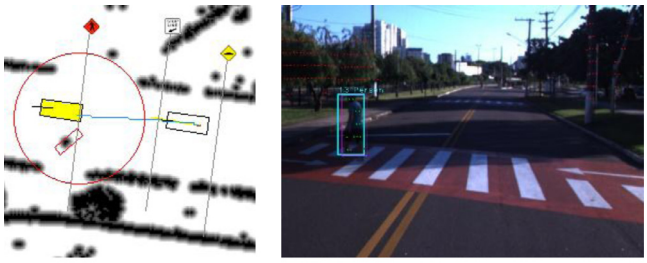

Fig. 6. (a) Occupancy grid map with a crosswalk (red circle) annotation, a bumper annotation (marked with the bumper traffic sign), a pedestrian (red rectangle), the planned car's path (blue line connecting two rectangles), and the car's goal state (yellow filled rectangle). (b) The camera image with the detected pedestrian highlighted with a rectangle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Therefore, it was convenient to place the Crosswalk State Estimator (CSE) inside the Path Planner, since it pertains to traffic signalization.

The CSE determines the current state of the crosswalk ahead in the car's path; i.e., whether the crosswalk is busy (has one or more pedestrians demanding right of passage) or free. To accomplish this, the CSE analyzes several data, such as the car's velocity, the distance between the car and the crosswalk, and the presence of pedestrians around the crosswalk area.

5.1. Crosswalk representation

The proposed system relies on the precise crosswalk position (world coordinates), previously annotated in the occupancy grid map. The annotation process consists of positioning the car on the crosswalks and recording the positions provided by car's system. Later, in an offline process (i.e., without the car), an annotator assigns a circle to each crosswalk in order to delimitate the (approximate) crosswalk area. The radius of the circle is set manually so that the circle line reaches the sidewalk. In addition to the crosswalk, the stop line is also manually annotated. This annotation could be performed automatically off-line, however, by leveraging an image-based crosswalk detection system [43,44] for example.

Fig. 6 (a) shows a crosswalk in our system's user interface. It is represented as the red circle centered on the map. A pedestrian in the crosswalk is shown as a red rectangle. Fig. 6 (b) illustrates the detection of the pedestrian in the map of Fig. 6 (a) in the camera image.

5.2. Pedestrian intention

We incorporate a measure of the pedestrian's intention of crossing the road to define the crosswalk state. It is modeled as a Boolean variable, therefore, (or simply) means that at least one pedestrian intends to cross the road, and (or) means the opposite. As depicted in Fig. 7, two scenarios (I and II) should be analyzed to assert that a specific pedestrian intends to cross the road.

If the pedestrian is outside the crosswalk area (scenario I), the system first verifies whether the car will reach the crosswalk (i.e., its midpoint) before the pedestrian reaches the crosswalk area (i.e., the red circle in Fig. 7). If it is the case, the system assumes, for simplicity, that the pedestrian is not intending to cross the road. Otherwise, intention exists if, and only if, the pedestrian is moving in the approximate direction ($\pm 30^\circ$) of the center of the crosswalk area.

If the pedestrian is inside the crosswalk area (scenario II), intention exists if, and only if, (i) the pedestrian is in the car's path or (ii) if velocity is equal or greater than 0.3 m/s and the pedestrian

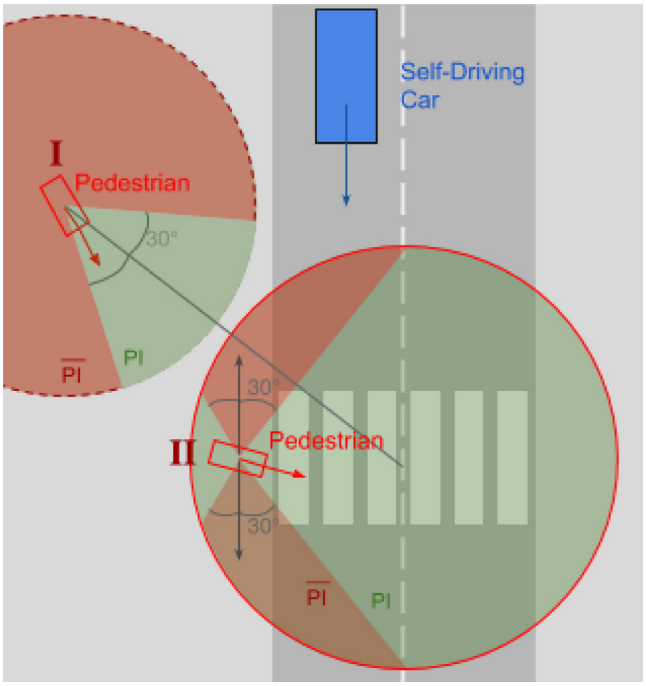

Fig. 7. Pedestrian Intention (PI) to cross the crosswalk. PI is true in areas shaded red and false in areas shaded green. Scenario I (dashed circle) when the pedestrian is outside the crossing area (solid circle): PI is true if the movement orientation is within $\pm 30^\circ$ of the line that connects the pedestrian to the centre of the crosswalk. Scenario II when the pedestrian is inside the crossing area (solid circle): PI is true if (i) the pedestrian is in the car's path or (ii) if velocity is equal or greater than 0.3 m/s and the pedestrian is not moving in the approximate direction ($\pm 30^\circ$) of the road. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is not moving in the approximate direction ($\pm 30^\circ$) of the road (assumed as the car's direction), i.e., not moving parallel to the road.

The angle range in both presented scenarios were empirically defined considering normal human behaviour on these situations. The range that defines whether a pedestrian is going to enter in a crosswalk could be dynamically defined but was chosen to be static to avoid noise problems due to imprecision on the pose prediction. These values were respectively chosen to covers visible pedestrians within a reasonable distance from the crosswalk and to safely avoid assuming a pedestrian is walking parallel when he is not.

5.3. CSE state machine

The CSE behavior was modelled with a state machine considering the pedestrians' intentions. Fig. 8 shows the diagram of the state machine we have designed. The circles represent the states and, inside each circle, there are the name of the states (Far, Near, Stopping, Stopped, Leaving) and the output of the state machine for each state (None, Free, or Busy). The arrows depict the state transitions and the conditions for changing from one state to another. If none of the conditions for leaving a state is true, the state remains the same. The condition variables that appear in the state machine are: START, which is true when the system is turned on; P, which is true when a pedestrian is present inside the crosswalk area (i.e., the red circle in Fig. 7); PI, which is true when the pedestrian has the intention of crossing the road; v, which is the car velocity; and D, which is distance from the car to the center crosswalk area in front of it (it is negative when the car has crossed the crosswalk center and left it behind). The conditions for changing states are evaluated every time a new car pose is published by the Localizer module and received by the Path Planner module (Fig. 2).

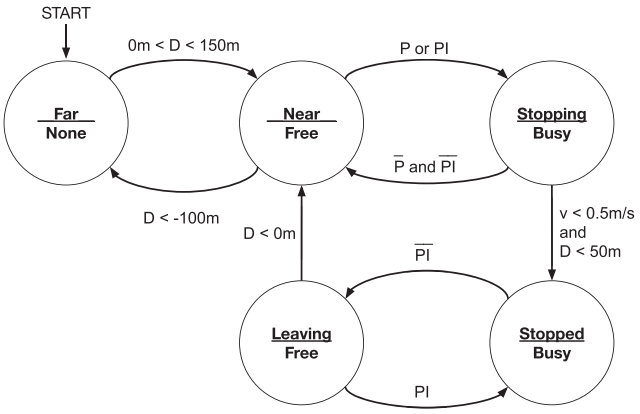

Fig. 8. CSE state machine.

The state machine starts in the Far state, which means that the car is far from any crosswalk; in this state the output of CSE is None. If 0m $<$ D $<$ 150m, there is a crosswalk ahead and the state changes to Near, while the output of CSE changes to Free (the car is free to cross the crosswalk). If the PSE detects any pedestrian in front of the car who is inside of the crosswalk area (P $=$ true) or has the intention to cross the road (PI $=$ true), the state changes to Stopping; the output of CSE is now Busy. If the speed of the car, v, is smaller than 0.5 m/s and D is smaller than 50 m, the state changes to Stopped: the CSE output is still busy in this state, however. If there is no pedestrian with intention to cross the road (PI $=$ false), the state changes to Leaving; the output of CSE is now Free. When the car crosses the crosswalk (D $<$ 0), the state changes back to Near and the output remains Free. Finally, when D $<$ -100, the state machine goes back to the Far state.

The distances and speed thresholds used in the state machine were empirically defined along the years of development of the self-driving car. A threshold of 0.5 m/s minimum velocity is considered to avoid that noisy sensor measurements falsely indicates that the vehicle is moving when it is not. The distances of 150 m ahead and 100 m behind are used to avoid unnecessary verification of crosswalk annotations that are too far from the vehicle's current position. The 50 m was chosen to link the stop event with the current crosswalk.

Using the output of CSE, the Behavior Selector module chooses a goal in the current path, and the desired velocity at this goal, so that the car moves smoothly and safely while considering all states of the state machine of CSE. For more details on how the Behavior Selector module accomplishes that, the reader can refer to [4]. It is important to stress that the Obstacle Avoider module is constantly running and reacts in case of an untreated or unpredicted state of the overall system.

6. Alternative Paths Generator (APG)

Ano ther contribution of this work is the treatment of pedestrians in places far from crosswalks. If there are pedestrians in the road near the path of the car, it has to select another path that is smooth and is at a safe distance of the pedestrians. To achieve that, we included in our Path Planner module an Alternative Paths Generator (APG) which provides many alternatives of paths based on the single path we used previously. The alternative paths are computed using Frenét Frames [12].

Frenét Frames is a well-known approach in tracking theory for modeling the path generation as the problem of generating a sequence of moving frames using a frame as reference. In this work, we follow the approach proposed in [12] that uses such moving

frames to plan alternative paths by combining different lateral and longitudinal cost functions aiming to mimic a human-like driving behavior. These functions minimize the lateral and longitudinal jerk so that the car can overtake an obstacle considering a safe lateral distance and then rejoin the original trajectory in a smooth way. The overall cost of a path is calculated as a weighted sum of the two, lateral and longitudinal contributions.

The reference frame is basically modeled by two vectors, one tangential and one normal, that can be calculated for each point of a curve representing the planned path for an obstacle-free road. The method uses these vectors to generate a set of possible trajectories and then chooses the one with the best cost and that complies with the obstacle's constraints in the map. Please refer to [12] for details.

As shown in Fig. 3, the Behavior Selector module receives, from PSE, the Pedestrians' States, and, from the Path Planner, the original path and the set of alternative paths computed by APG. If there is a pedestrian in front of the car and it obstructs the original path, the Behavior Selector chooses the best-cost path from the set of alternative paths.

7. Experimental methodology

The presented pedestrian handler was evaluated in a real-world scenario using a self-driving car. The experiments were divided in two common scenarios where the self-driving car and the pedestrians are present. The first considers the pedestrian is near a crosswalk, whereas the second depict common situations in Brazil in which the pedestrian is in the lane, but there is no crosswalk nearby. The remainder of this section describes the experiments in the two scenarios, near-crosswalk and no-crosswalk.

7.1. Near-crosswalk scenario experiment

The system was evaluated in situations considering pedestrians approaching the crosswalk area and considering pedestrians in the crosswalk area with and without intention of crossing it. The evaluation considered the expected behavior of the car in such situations. This set of experiments was firstly conducted in a simulated environment in order to enable comparison with the Pedestrian Handling System (PHS) proposed by Guidolini et al. [4]. Finally, a set of real-world experiments were performed with the self-driving car.

The simulated scenario disregards the detection and tracking procedures; therefore, it focuses on assessing the logic behind the CSE. For this scenario, a pedestrian simulator was developed in order to prove the concept of the proposed method. This simulator is a secondary contribution of this work that allows to insert 'fake' pedestrians into a virtual environment. In this context, Pedestrians' States and car's information (e.g., velocity, position, and direction) comes for free, which makes possible the evaluation of the car's behavior with both the system of [4] and the proposed Crosswalk State Estimator.

The experiments for the simulated scenario comprises five situations: (a) the pedestrian walks into the crosswalk area direction (he is initially in outside the area) reaching the crosswalk long before the car; (b) similar to the first situation, except for the fact that the pedestrian and the car reach the crosswalk area almost simultaneously; (c) similar to the first situation, but now the car reaches the crosswalk long before the pedestrian; (d) the pedestrian is inside the crosswalk area and moves parallel to the road direction (i.e., without intention of crossing the road); and, (e) the pedestrian is standing on the pavement and inside the crosswalk area, but with no intention of crossing it.

A second set of experiments was conducted to assess the proposed CSE in the real-world. The car was driven around predefined routes in self-drive mode with the proposed CSE enabled. During these experiments, the full system was running, which includes the detection and tracking procedures. No comparative evaluation was conducted since the replication of the physical environment's conditions for online tests is impracticable. Due to the sensors' maximum range, the first scenario of the simulated results could not be replicated on the real-world. Therefore, this set of experiments comprise four situations: (a) the pedestrian (initially outside the crosswalk area) reaches the crosswalk area almost simultaneously to the car; (b) the car reaches the crosswalk long before the pedestrian; (c) the pedestrian moves (inside the crosswalk area) parallel to the road direction; and, (d) the pedestrian is standing on the pavement and inside the crosswalk area, but with no intention of crossing it.

7.2. No-crosswalk scenario experiment

One common scenario in Brazilian low-traffic roads is the presence of pedestrians walking along the road. Therefore, this set of experiments tries to replicate such scenarios that have to deal with if the self-driving car are made to interact with pedestrians. Since there is no software available for direct comparisons in a simulated environment, only real-worlds experiments with our car were conducted. In these experiments, the system has to be able to correctly generate alternate paths to overtake the pedestrian in the road.

Therefore, the car was driven in self-drive mode around predefined routes considering situations with pedestrians in the road. During these experiments, the full system is running, which includes the detection, tracking procedures and the proposed APG. For this scenario the experiments are conducted considering three major situations: (a) the pedestrian is stopped directly in the car's path; (b) the pedestrian is walking in the lane but respecting its direction, common situation where there is no sidewalk; (c) the pedestrian crosses the lane on an unappropriated area, far from a crosswalk.

8. Results and discussion

This section presents the results for the experiments for the near-crosswalk and no-crosswalk scenarios.

8.1. Near-crosswalk scenario experiment

Fig. 9 shows the PHS proposed by Guidolini et al. [4] in the top row and the proposed CSE in the bottom row, for the simulated environment. In the first situation (Fig. 9 (a)), the pedestrian reaches the crosswalk area long before the car. Initially, PHS considered that the crosswalk was free, since the pedestrian was outside the crosswalk area. When the pedestrian entered in the crosswalk area, the car was able to stop. This was possible only because the car had enough time (i.e., it was far enough given its speed) to change its behavior. The proposed Crosswalk State Estimator, on the other hand, considered the crosswalk was busy since the pedestrian was moving towards its area. Note that the car planned to stop before the stop line from the initial moment, which is particularly interesting for a smoother driving experience.

In the second situation (Fig. 9 (b)), nevertheless, the car reaches the crosswalk area almost simultaneously to the pedestrian. Therefore, in the PHS, the car overpassed the stop line since it started the stop maneuver when it was already in the crosswalk area. This late decision compromises the pedestrian safety, as well as the safety of vehicles behind the car due to the sudden braking. Differently, the proposed Crosswalk State Estimator was able to stop the car before the stop line because it perceived that the pedestrian was moving towards the crosswalk area.

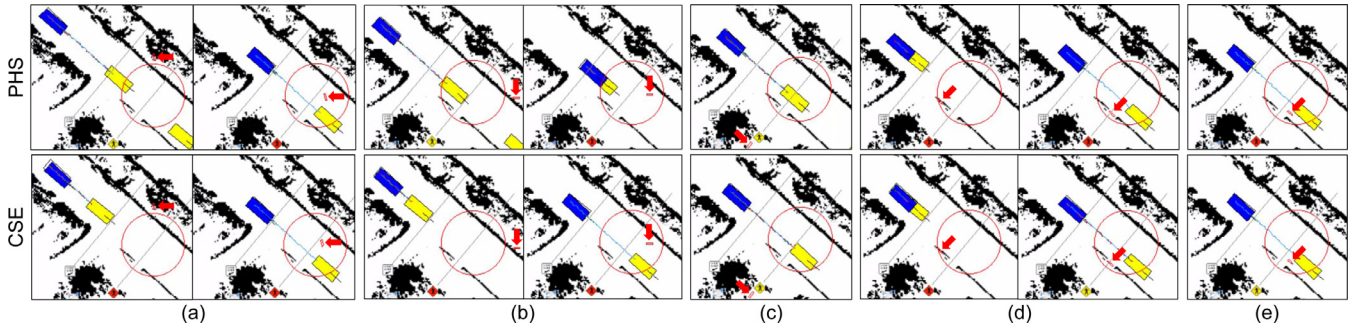

Fig. 9. Results for the simulated experiments. The top and bottom rows represent, respectively, the PHS [4] and the proposed CSE approaches. The five depicted situations are: (a) the pedestrian reaches the crosswalk long before the car; (b) the pedestrian and the car reach the crosswalk area almost simultaneously; (c) the car reaches the crosswalk long before the pedestrian; (d) the pedestrian moves parallel to the road direction; and (e) the pedestrian is standing on the crosswalk area. In the columns (a), (b), and (d), the two images placed side-by-side denote sequential configurations. Six elements of these illustrations are regarded in the discussion: the current car's position (filled blue, and most left rectangle, in simulated scenarios, white in real-world scenarios), the planned car's position (yellow filled boxes), the current pedestrian's position (box indicated by the red arrow), the stop line (the line closest to top-left corner), the crosswalk area (red circle), and the crosswalk state flag (red/yellow diamond). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

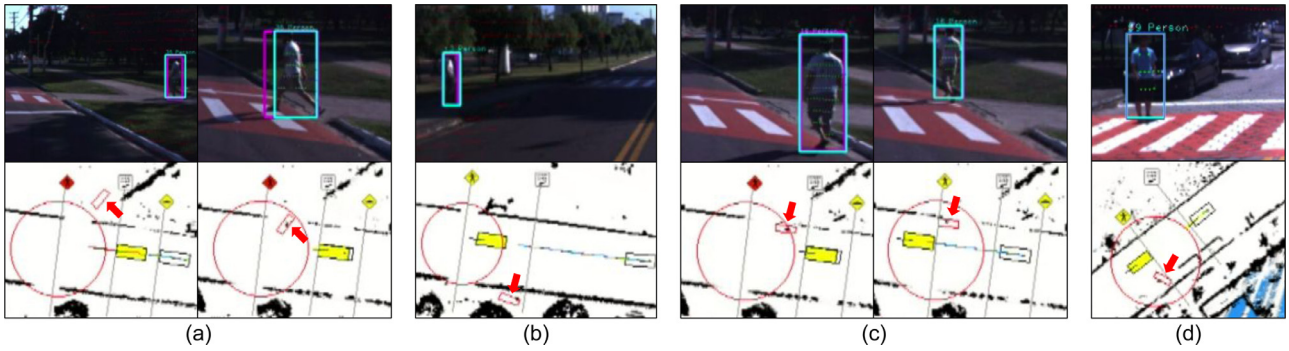

Fig. 10. Results for the real-world experiments near crosswalks. The top images present the camera of the self-driving car, the cyan and magenta boxes represent, respectively, the detected and tracked positions. The bottom images present the maps described in Fig. 9. The four illustrated situations are: (a) the pedestrian reaches the crosswalk area almost simultaneously to the car; (b) the car passes completely through the crosswalk before the pedestrian reaches the crosswalk area; (c) the pedestrian moves (inside the crosswalk area) parallel to the road direction; and (d) the pedestrian is static before the crosswalk.

In the third situation (Fig. 9 (c)), both Crosswalk State Estimators have the same apparent behavior. The pedestrian was far away from the crosswalk, therefore the car passed completely by the area before the pedestrian reached it. However, the proposed system's decision of keep moving was based on the current pedestrian's velocity and on the current distance between the car and the pedestrian. This is safer than simply checking the presence of a pedestrian in the crosswalk area.

When the pedestrian was inside the crosswalk area moving parallel to the road (Fig. 9 (d)), both Crosswalk State Estimators considered that the crosswalk was busy. However, after some time, the proposed Crosswalk State Estimator set the crosswalk as free because the pedestrian (still inside the crosswalk area) was detected as moving parallel to the road (i.e., not intending to cross). The PHS, on the contrary, got stuck until the pedestrian was outside the crosswalk area. This behavior was like the fifth situation (Fig. 9 (e)), but now the presence of a static pedestrian was what prevented the car from starting to move. With the PHS, a pedestrian stopped (e.g., talking on the phone) on the pavement inside the crosswalk area could leave the car waiting forever. Please see the video of this experiment to have a better feeling of the results[1].

For the real-world experiments near crosswalks the car, equipped with the proposed Crosswalk State Estimator, demonstrated the expected behavior. In Fig. 10 (a), the car stopped before the stop line even entering the crosswalk area at (almost) the same time as the pedestrian, as in the simulated scenario. Although Pedestrian Detector had failed for some frames, the Pedestrian Tracker was able to keep the car aware of the pedestrian. In Fig. 10 (b), the Crosswalk State Estimator decided that the car could proceed since the pedestrian would not reach the crosswalk area on time. The third situation, in Fig. 10 (c), shows two moments. Initially, the pedestrian was correctly detected inside the crosswalk area and the Crosswalk State Estimator assumed the crosswalk was busy. After verifying the pedestrian was moving parallel to the road, the crosswalk state was set to "Free", and the car started to move. Finally, Fig. 10 (d) shows the situation in which the pedestrian was static inside the crosswalk area. The car had already stopped before the stop line, and then started to move after 2 seconds when it perceived the pedestrian was not intending to cross the road. Please see the video of this experiment to have a better feeling of the results[2].

8.2. No-crosswalk scenario experiment

Fig. 11 shows the results for the real-world experiments with the self-driving car in situations with pedestrians on the road without crosswalks nearby. Fig. 11 (a) shows the situation where the pedestrian was stopped in the border of the lane obstructing the car's original path. As shown in the map, the APG generated a series of paths that avoids the pedestrian, then the Behavior Selector chose a path respecting the safety distance to the pedestrian. In this situation, the pedestrian was static during the car's

[1] https://youtu.be/PLlkphSvopw

[2] https://youtu.be/dncJtDzaz9Q

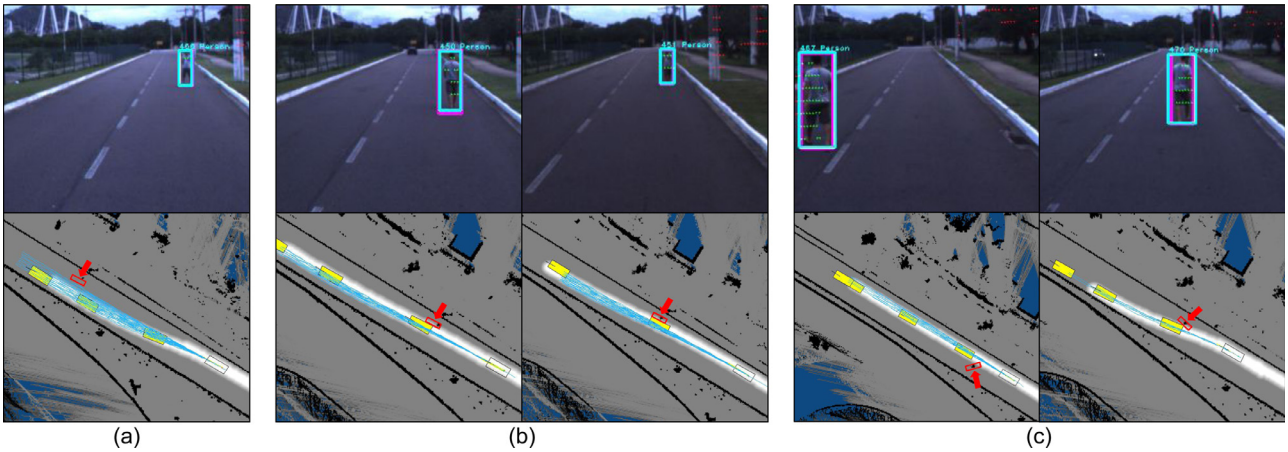

Fig. 11. Results for the real-world experiments without crosswalks. The top images were captured with the camera of the self-driving car, the blue and magenta boxes represent, respectively, the detected and tracked positions. The bottom images present the maps described in Fig. 9, with the addition of the blue lines that represents the paths generated by the APG and the highlighted trace represent the chosen path by the Behavior Selector. The three illustrated situations are: (a) the pedestrian stopped in the border of the lane; (b) the pedestrian walking parallel to the lane; (c) the pedestrian crosses the road in an unappropriated area.

maneuver, therefore, a static path choice would be enough to address this situation.

In Fig. 11 (b), the pedestrian walks in the lane respecting its direction. This is a common situation when there are no sidewalks. The left and the right images depict the pedestrian walking in the same and opposite car's orientation respectively. In this scenario, the paths must be dynamically recalculated to readjuste to the pedestrians' trajectories. This explains why the APG generates new paths for each car's pose.

In Fig. 11 (c), the pedestrian crosses the road in an unappropriated area where there are no crosswalks nearby. In the left image, the pedestrian crosses from the same to the opposite lane and in the right image, the pedestrian crosses from the opposite to the same lane of the car. In both situations, the APG generates a set of paths that deviates from the pedestrian, but the pedestrian keeps moving from one lane to another. Because of this, in a certain point the APG cannot generate any safe path. With no alternative paths available, the Behavior Selector slows down the car until that the APG calculates new paths. In the scenario depicted by the left image, after the pedestrian crosses, the self-driving car keeps the original path, whereas, in the scenario of the right, the car avoids the pedestrian since it continues in the originally planned car's path. Please see the video of this experiment to have a better feeling of the results[3].

8.3. Qualitative analysis with literature

Most path planners and pedestrian handlers in the literature do not provide enough details for implementation and have specific constraints for the sensors. Therefore, their experiments could not be replicated and only a discussion about the qualitative results of the literature could be presented.

The results presented by Li et al. [22] show that their path generator have a special treatment for pedestrians, but the authors do not address crosswalks. The situation depicted by the paper is a pedestrian walking in the lane without caring about the autonomous vehicle, very similar to the situation presented in Fig. 11 (c). Their behavioral planner decided to slowdown and wait for the pedestrian to cross, avoiding aggressive maneuvers, whereas our method performs a soft transition to and from the other lane.

The solution proposed by Fernández et al. [24] generates an alternative path from the original plan to avoid an artificial and static pedestrian dummy on the lane. The micro-bus performed the maneuver smoothly several times ensuring the passengers comfort and safety. Fig. 11(a) shows how our APG handles this situation in a similar manner. One limitation of their approach is that the micro-bus cannot handle dynamic pedestrians (that was suggested as a future work), which is already handled by Li et al. [22] and by our method as shown in Fig. 11(b) and (c).

In [23], the authors only performed simulated experiments, but they considered higher speeds when compared to the works mentioned in this section. Their work handles pedestrians crossing the street without considering the trajectory of the pedestrian. As a result, an unconventional maneuver might be performed by the car, such as trying to deviate using a new trajectory that passes where the pedestrian is going to. It is worth noting that their system is able to brake the car in time even when such situations happen. The most similar situation present in our experiments is depicted in Fig. 11(c). There, the car slows down and diverges from the pedestrian, but considering a speed approximately 20km/h slower than in the simulation presented by Rentería et al. [23].

9. Conclusion and future work

The proper handling of pedestrians on the road is very challenging and essential for safe autonomous driving. This paper proposes a system to handle pedestrians considering image tracking information. Such track information enables predicting pedestrian intention and, therefore, taking better decision in self-driving cars. The pedestrian handler was designed to address pedestrians not only near crosswalk (which is the usual place for pedestrians to cross streets), but also in places without nearby crosswalks, which is very usual in low-traffic Brazilian roads. For these situations, an alternative paths generator was proposed to find a route that can safely overtake the pedestrian on the road.

Experimental results demonstrated the feasibility of the proposed approach. By analyzing the pedestrian's intention, the car's path planner was able to stop earlier, which implies in smoother and safer driving. In addition, the analysis of intention is important to prevent the car to get stuck before the crosswalk when the pedestrian is static or moving along the road direction. The former is very common when the pedestrian is talking on the phone or chatting with friends on the pavement. In situations with no

[3] https://youtu.be/aR9twCHOGh8

crosswalks, the APG was able to deviate of the pedestrians in all the presented situations.

Despite of the enhanced behavior, some adjustments are necessary to improve the overall system's performance. First, a camera with larger field of view combined with a larger range LiDAR would be indicated to enhance the short-range detection of pedestrians. In addition, a higher resolution LiDAR should provide more accurate pedestrian's localization, and, therefore, more robustness in the tracking procedure. The system also has some software limitations: (i) the detection is based on image and, therefore, might be affected by strong light variations in the environments; (ii) occluded or partially visible pedestrians may also hinder detection; (iii) system could only be tested at 30 km/h due to the resolution of the LiDAR point cloud that limits the distance for pose estimation. It is worth noting that the self-driving car has a Obstacle Avoider module to cope with any situation that might not get covered by any of these limitations, and it would stop the car if any obstacle gets in the path.

Near-future work includes the evaluation of the proposed method in crowded scenarios. Additionally, the tracking of pedestrians in 3D world coordinates should be investigated as an alternative for tracking in the 2D camera plane and then transferring for 3D coordinates. This could yield a more reliable estimation of the pedestrian's velocity and orientation.

Declaration of Competing Interest

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

Acknowledgments

We would like to acknowledge the scholarships of Productivity on Research (grants 311120/2016-4 and 311504/2017-5) supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico(CNPq, Brazil). We also like to acknoledge the finance support from Programa de Apoio a Núcleos Emergentes (Pronem, grant 594/2018) supported by Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES, Brazil). This study was finance in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior 'Brasil (CAPES)' Finance Code 001. The authors would also like to thank the NVIDIA Corporation for their donation of GPUs and Sabrina Panceri for the help with the templates.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cag.2019.08.004.

References

[1] Badue C, Guidolini R, Carneiro RV, Azevedo P, Cardoso VB, Forechi A, Jesus LFR, Berriel RF, Paixão TM, Mutz F, et al. Self-driving cars: a survey, arXiv:190104407; 2019.
[2] Premebida C, Ludwig O, Nunes U. Exploiting lidar-based features on pedestrian detection in urban scenarios. In: Proceedings of the 12th international IEEE conference on intelligent transportation systems. IEEE; 2009. p. 1–6.
[3] Benenson R, Mathias M, Timofte R, Van Gool L. Pedestrian detection at 100 frames per second. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE; 2012. p. 2903–10.
[4] Guidolini R, Scart LG, Jesus LFR, Cardoso VB, Badue C, Oliveira-Santos T. Handling pedestrians in crosswalks using deep neural networks in the IARA autonomous car. In: Proceedings of the international joint conference on neural networks (IJCNN). IEEE; 2018. p. 1–8.
[5] Li J, Liang X, Shen S, Xu T, Feng J, Yan S. Scale-aware fast r-CNN for pedestrian detection. IEEE Trans Multimed 2017;20(4):985–96.
[6] Wang H, Wang B, Liu B, Meng X, Yang G. Pedestrian recognition and tracking using 3D liDAR for autonomous vehicle. Robot Autonom Syst 2017;88:71–8.
[7] Baek I, Davies A, Yan G, Rajkumar RR. Real-time detection, tracking, and classification of moving and stationary objects using multiple fisheye images. In: Proceedings of the ieee intelligent vehicles symposium (IV). IEEE; 2018. p. 447–52.
[8] Leal-Taixé L, Canton-Ferrer C, Schindler K. Learning by tracking: Siamese CNN for robust target association. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops; 2016. p. 33–40.
[9] Henschel R, Leal-Taixé L, Cremers D, Rosenhahn B. Fusion of head and full-body detectors for multi-object tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops; 2018. p. 1428–37.
[10] Sheng H, Zhang Y, Chen J, Xiong Z, Zhang J. Heterogeneous association graph fusion for target association in multiple object tracking. IEEE Trans Circuits Syst Video Technol 2018.
[11] Long C, Haizhou A, Zijie Z, Chong S. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: Proceedings of the ICME; 2018.
[12] Werling M, Ziegler J, Kammel S, Thrun S. Optimal trajectory generation for dynamic street scenarios in a Frenet frame. In: Proceedings of the IEEE international conference on robotics and automation. IEEE; 2010. p. 987–93.
[13] He K, Gkioxari G, Dollar P, Girshick R. Mask r-CNN. In: Proceedings of the IEEE international conference on computer vision (ICCV); 2017.
[14] Chen L-C, Zhu Y, Papandreou G, Schroff F, Adam H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV); 2018.
[15] Fragkiadaki K, Levine S, Felsen P, Malik J. Recurrent network models for human dynamics. In: Proceedings of the IEEE international conference on computer vision (ICCV); 2015.
[16] Butepage J, Black MJ, Kragic D, Kjellstrom H. Deep representation learning for human motion prediction and classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR); 2017.
[17] Walker J, Marino K, Gupta A, Hebert M. The pose knows: Video forecasting by generating pose futures. In: Proceedings of the ieee international conference on computer vision; 2017. p. 3332–41.
[18] Martinez J, Black MJ, Romero J. On human motion prediction using recurrent neural networks. In: Proceedings of the ieee conference on computer vision and pattern recognition; 2017. p. 2891–900.
[19] Cao Z, Simon T, Wei S-E, Sheikh Y. Realtime multi-person 2D pose estimation using part affinity fields. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR); 2017.
[20] Papandreou G, Zhu T, Kanazawa N, Toshev A, Tompson J, Bregler C, Murphy K. Towards accurate multi-person pose estimation in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 4903–11.
[21] Milan A, Leal-Taixé L, Reid I, Roth S, Schindler K. MOT16: a benchmark for multi-object tracking. arXiv: 1603.00831[cs].
[22] Li X, Sun Z, Cao D, He Z, Zhu Q. Real-time trajectory planning for autonomous urban driving: framework, algorithms, and verifications. IEEE/ASME Trans Mechatron 2015;21(2):740–53.
[23] Rentería LA, Oria JMP, Becerra VM, Avello AJ, Al-Hadithi BM. Modeling, simulation, and control of pedestrian avoidance maneuver for an urban electric vehicle. In: Proceedings of the IEEE European modelling symposium (EMS). IEEE; 2015. p. 201–6.
[24] Fernández C, Domínguez R, Fernández-Llorca D, Alonso J, Sotelo MA. Autonomous navigation and obstacle avoidance of a micro-bus. Int J Adv Robot Syst 2013;10(4):212.
[25] Torc Robotics, Virgina US. How we share the road with pedestrians. https://torc.ai/how-we-share-the-road-with-pedestrians/; 2018.
[26] Waymo, US. Waymo – waymo. https://waymo.com/; 2019.
[27] Mutz F, Veronese LP, Oliveira-Santos T, de Aguiar E, Cheein FAA, De Souza AF. Large-scale mapping in complex field scenarios using an autonomous car. Expert Syst Appl 2016;46:439–62.
[28] Thrun S, Burgard W, Fox D. Probabilistic robotics. MIT press; 2005.
[29] de Paula Veronese L, Guivant J, Cheein FAA, Oliveira-Santos T, Mutz F, de Aguiar E, Badue C, De Souza AF. A light-weight yet accurate localization system for autonomous cars in large-scale and complex environments. In: Proceedings of the IEEE 19th international conference on intelligent transportation systems (ITSC). IEEE; 2016. p. 520–5.
[30] Berger M, Forechi A, De Souza AF, De Oliveira NJ, Veronese L, Neves V, de Aguiar E, Badue C. Traffic sign recognition with wiSARD and VG-RAM weightless neural networks. J Netw Innov Comput 2013;1(1):87–98.
[31] De Souza AF, Fontana C, Mutz F, de Oliveira TA, Berger M, Forechi A, de Oliveira Neto J, de Aguiar E, Badue C. Traffic sign detection with VG-RAM weightless neural networks. In: Proceedings of the international joint conference on neural networks (IJCNN). IEEE; 2013. p. 1–9.
[32] Berriel RF, de Aguiar E, De Souza AF, Oliveira-Santos T. Ego-lane analysis system (ELAS): dataset and algorithms. Image Vis Comput 2017;68:64–75.
[33] Cardoso V, Oliveira J, Teixeira T, Badue C, Mutz F, Oliveira-Santos T, Veronese L, De Souza AF. A model-predictive motion planner for the IARA autonomous car. In: Proceedings of the ieee international conference on robotics and automation (ICRA). IEEE; 2017. p. 225–30.
[34] Guidolini R, Badue C, Berger M, de Paula Veronese L, De Souza AF. A simple yet effective obstacle avoider for the IARA autonomous car. In: Proceedings of the IEEE 19th international conference on intelligent transportation systems (ITSC). IEEE; 2016. p. 1914–19.

[35] Guidolini R, De Souza AF, Mutz F, Badue C. Neural-based model predictive control for tackling steering delays of autonomous cars. In: Proceedings of the international joint conference on neural networks (IJCNN). IEEE; 2017. p. 4324–31.
[36] Redmon J, Farhadi A. Yolo9000: better, faster, stronger. In: Proceedings of the ieee conference on computer vision and pattern recognition; 2017. p. 7263–71.
[37] Redmon J, Farhadi A. YOLOv3: an incremental improvement, arXiv:180402767; 2018.
[38] Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. ImageNet: a large-scale hierarchical image database. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE; 2009. p. 248–55.
[39] Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft COCO: common objects in context. In: Proceedings of the European conference on computer vision. Springer; 2014. p. 740–55.
[40] Dai J, Li Y, He K, Sun J. R-FCN: object detection via region-based fully convolutional networks. In: Proceedings of the advances in neural information processing systems; 2016. p. 379–87.
[41] Zhao L, Li X, Zhuang Y, Wang J. Deeply-learned part-aligned representations for person re-identification. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 3219–28.
[42] Hartley R, Zisserman A. Multiple view geometry in computer vision. Cambridge university press; 2003.
[43] Berriel RF, Lopes AT, De Souza AF, Oliveira-Santos T. Deep learning-based large-scale automatic satellite crosswalk classification. IEEE Geosci Remote Sens Lett 2017;14(9):1513–17.
[44] Berriel RF, Rossi FS, de Souza AF, Oliveira-Santos T. Automatic large-scale data acquisition via crowd sourcing for crosswalk classification: a deep learning approach. Comput Graph 2017;68:32–42.