information

Article

Architectural Framework to Enhance Image-Based Vehicle Positioning for Advanced Functionalities

Iosif-Alin Beti [1,†], Paul-Corneliu Herghelegiu [2,†] and Constantin-Florin Caruntu [1,*,†]

[1] Department of Automatic Control and Applied Informatics, Gheorghe Asachi Technical University of Iasi, 700050 Iasi, Romania; iosif-alin.beti@student.tuiasi.ro
[2] Department of Computer Science and Engineering, Gheorghe Asachi Technical University of Iasi, 700050 Iasi, Romania; paul-corneliu.herghelegiu@academic.tuiasi.ro
* Correspondence: caruntuc@ac.tuiasi.ro
† These authors contributed equally to this work.

Abstract: The growing number of vehicles on the roads has resulted in several challenges, including increased accident rates, fuel consumption, pollution, travel time, and driving stress. However, recent advancements in intelligent vehicle technologies, such as sensors and communication networks, have the potential to revolutionize road traffic and address these challenges. In particular, the concept of platooning for autonomous vehicles, where they travel in groups at high speeds with minimal distances between them, has been proposed to enhance the efficiency of road traffic. To achieve this, it is essential to determine the precise position of vehicles relative to each other. Global positioning system (GPS) devices have an intended positioning error that might increase due to various conditions, e.g., the number of available satellites, nearby buildings, trees, driving into tunnels, etc., making it difficult to compute the exact relative position between two vehicles. To address this challenge, this paper proposes a new architectural framework to improve positioning accuracy using images captured by onboard cameras. It presents a novel algorithm and performance results for vehicle positioning based on GPS and video data. This approach is decentralized, meaning that each vehicle has its own camera and computing unit and communicates with nearby vehicles.

Keywords: computer vision; traffic optimization; vehicle platooning; vehicle positioning; V2V communication

Citation: Beti, I.-A.; Herghelegiu, P.-C.; Caruntu, C.-F. Architectural Framework to Enhance Image-Based Vehicle Positioning for Advanced Functionalities. *Information* **2024**, *15*, 323. https://doi.org/10.3390/info15060323

Academic Editors: Dejiu Chen, Fredrik Warg, Anders Thorsén and Anders Cassel

Received: 24 April 2024
Revised: 22 May 2024
Accepted: 28 May 2024
Published: 31 May 2024

1. Introduction

We live in a constantly developing world, where cities are becoming more and more crowded, and the number of cars is constantly increasing. Traffic becomes more and more congested because the road infrastructure can no longer cope with the increasing number of vehicles. This means more fuel consumption, more pollution, longer journeys, stressed drivers, and, most importantly, an increase in the number of accidents. Pedestrians, cyclists, and motorcyclists are the most exposed to road accidents. According to a report from 2017 by the World Health Organization, every year, 1.25 million people die in road accidents, and millions more are injured [1]. The latest status report from 2023 [2] indicates a slight decrease in the number of road traffic deaths to 1.19 million per year, highlighting the positive impact of efforts to enhance road safety. However, it underscores that the cost of mobility remains unacceptably high. The study described in [3] tracked the progress of reducing the number of car accidents since 2010 in several cities. It concluded that very few of the studied cities are improving road safety at a pace that will reduce road deaths by 50% by 2030, in line with the United Nations' road safety targets.

Autonomous vehicles can avoid some errors made by drivers, and they can improve the flow of traffic by controlling their pace so that traffic stops oscillating. They are equipped with advanced technologies such as global positioning systems (GPS), video cameras, radars, light detection and ranging (LiDARs), and many other types of sensors.

They can travel together, exchanging information about travel intentions, detected hazards and obstacles, etc., through vehicle-to-vehicle (V2V) or vehicle to everything (V2X) communication networks.

To increase the efficiency of road traffic, the idea of grouping autonomous vehicles into platoons through information exchange was proposed in [4]. Vehicles should consider all available lanes on a given road sector when forming a group and travel at high speeds with minimal safety distances between them. However, this is possible only if a vehicle can determine its precise position with respect to other traffic participants.

In recent years, image processing and computer vision techniques have been widely applied to solve various real-world problems related to traffic management, surveillance, and autonomous driving. In particular, the detection of traffic participants such as vehicles, pedestrians, and bicycles [5] plays a crucial role in many advanced driver assistance systems (ADAS) and smart transportation applications.

Image processing is important in optimizing traffic by being used to develop functionalities that reduce the number of accidents, increase traffic comfort, and group vehicles into platoons. Several approaches have been proposed over time to detect traffic participants with video cameras using convolutional neural networks [6,7], but most of them require a significant amount of computing power and cannot be used in real-time due to increased latency.

In this paper, a proof of concept algorithm to solve a part of the image-based vehicle platooning problem is proposed. It uses a decentralized approach, with each vehicle performing its own computing steps and determining its position with respect to the nearby vehicles. This approach relies on images acquired by the vehicle's cameras and the communication between vehicles. To test our approach, we used cheap commercial dashboard cameras equipped with a GPS sensor. No other sensors were used, mainly because they would have greatly increased the hardware cost. Each vehicle computes an image descriptor for every frame in the video stream, which it sends as a message along with other GPS information to other vehicles. Vehicles within communication range receive this message and attempt to find the frame in their own stream that most closely resembles the received one. The novelty of this approach lies in calculating the distance between the two vehicles by matching image descriptors computed for frames from both vehicles, determining the time difference at which the two frames were captured, and considering the traveling speeds of vehicles.

The rest of the paper is organized as follows. Section 2 presents some vehicle grouping methods for traffic optimization, then reviews applications of image processing related to street scenes, and, lastly, presents several image descriptors. The method proposed in this paper is detailed in Section 3, while in Section 4, the implementation of the algorithm is presented. In Section 5, preliminary results are presented, demonstrating the feasibility of the proposed algorithm. Finally, Section 6 presents the main conclusions of this study and directions for future research.

2. Related Work

The aim of this study is to describe a system architecture for positioning nearby vehicles using image processing techniques. The related work section is divided into three subsections: traffic optimization, street scene image processing, and image descriptors, as detailed in the following subsections.

2.1. Traffic Optimization

The majority of driver errors can be avoided by autonomous vehicles. They can reduce traffic oscillations by maintaining a safe distance, while exchanging information in real-time. Single-lane platooning solutions, proposed in [8,9], prove that vehicle platooning improves traffic safety and increases the capacity of existing roads.

To further increase traffic flow, ref. [4] extends the idea of single-lane platoons to multi-lane platoons [10–12]. Vehicles should consider all available lanes on a given road

sector when creating a group of vehicles and travel with small distances between them at high speeds. The platoon should be dynamic to allow new vehicles to join or leave the group, be able to overcome certain obstacles encountered on the road, and also allow faster-moving vehicles to overtake. Through the vehicle-to-vehicle communication network, they exchange information about travel intentions, dangers, and detected obstacles.

Vehicle movement control is divided into two parts: lateral control and longitudinal control. Lateral control is in charge of changing lanes, whereas longitudinal control is in charge of actions in the current lane. These two, when combined, must ensure that vehicle collisions are avoided. Maintaining formations and joining new members is made easier with lateral control via a lane change solution. At the same time, longitudinal control is used to keep a safe distance between vehicles. As such, the triangle formation strategy inspired by [13] is usually chosen because it offers several advantages, such as the stability of each member within the group and quick regrouping of the members in case of a dissolving scenario.

The way platoons are formed is based on the fields of swarm robotics and flocking, which are inspired by nature, more precisely, by the way fish, birds [14], insects, and mammals interact in a group [15,16]. An individual possesses poor abilities, but within a group, they contribute to the formation of complex group behaviors and, thus, provide flexibility and robustness, such as route planning and task allocation.

2.2. Street Scene Image Processing

Vehicle detection plays a very important role in modern society, significantly impacting transportation efficiency, safety, and urban planning. It optimizes traffic flow by refining signal timings and reducing congestion, as evidenced in [17]. Moreover, advancements in vehicle detection technology have facilitated features like automatic collision avoidance and pedestrian detection, contributing to a decrease in accidents [5].

Law enforcement also benefits from vehicle detection systems, aiding in tasks like license plate identification and stolen vehicle tracking [18]. Additionally, these systems support efficient parking management by monitoring parking spaces and guiding drivers to available spots [19].

Regarding the topic of advanced driver-assistance systems (ADAS), recent studies provide comprehensive reviews of vision-based on-road vehicle detection systems [20,21]. These systems, mounted on vehicles, face challenges in handling the vast amounts of data from traffic surveillance cameras and necessitate real-time analysis for effective traffic management.

Addressing challenges in traffic monitoring requires precise object detection and classification, accurate speed measurement, and interpretation of traffic patterns. Techniques proposed in studies offer efficient approaches for detecting cars in video frames, utilizing image processing methods [22].

While vision-based solutions have made significant advances in the automotive industry, they remain vulnerable to adverse weather conditions [23]. Weather elements like heavy rain, fog, or low lighting can potentially impact the accuracy and reliability of these systems, thus necessitating further research and development efforts.

Moreover, communication between vehicles is an emerging area of research, as highlighted in [24]. Understanding and optimizing vehicle-to-vehicle communication techniques are essential for enhancing road safety and traffic efficiency. Implementing robust communication protocols can facilitate cooperative driving strategies, leading to smoother traffic flow and reduced congestion levels.

2.3. Image Descriptors

Image descriptors are essential in computer vision and image processing. They extract robust and distinctive features from images for tasks like matching, recognition, and retrieval. Rather than processing the entire image, these techniques focus on specific key points. Each key point is associated with a descriptor that describes its properties. Examples of such descriptors are provided below.

2.3.1. Scale-Invariant Feature Transform (SIFT) Descriptor

The scale-invariant feature transform (SIFT) [25] is a widely used feature descriptor in computer vision and image processing. It extracts distinctive features using the scale-space extrema detection and the difference in the Gaussian (DoG) method. SIFT provides invariance to image scaling, rotation, and robustness against changes in viewpoint, illumination, and occlusion.

The algorithm consists of several steps [26]:

1. Scale-space extrema detection: this step identifies potential points of interest that are invariant to orientation using the difference of Gaussians (DoG) function.
2. Key point localization: the algorithm establishes the location and scale of key points to measure their stability.
   - Contrast threshold: following the selection of key points, the algorithm sets a contrast threshold to ensure stability. By considering the DoG function as a contrast function, key points with a DoG value less than 0.03 (after normalizing the intensity [0 1]) are excluded from the list.
   - Eliminating edge response: the next approach for localizing key points involves the elimination of edge responses. This is achieved by employing the Hessian matrix derived from the DoG function,

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}. \tag{1}$$

   Due to variations in the DoG function, there are significant changes in curvature at edges perpendicular to the direction of interest. To ensure robustness, key points lacking a stable maximum are excluded. This is achieved by predicting the Hessian matrix and computing the sum of its eigenvalues.

$$T_r(H) = D_{xx} + D_{yy} = \alpha + \beta, \tag{2}$$

$$Det(H) = D_{xx}D_{yy} - D_{yx}D_{xy} = \alpha\beta. \tag{3}$$

   In rare cases, the curvatures may have different signs when the determinant is negative. Let r denote the ratio of the largest to the smallest magnitude eigenvalue, then the expression $(r+1)^2/r$ reaches its minimum value when the two eigenvalues are equal, and it rises as r increases. To ensure the principal curvatures ratio stays below a threshold r, only verifying r is necessary.

$$\frac{T_r(H)}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}, \tag{4}$$

$$\frac{T_r(H)^2}{Det(H)} < \frac{(r+1)^2}{r}. \tag{5}$$

   Therefore, key points with a ratio between the primary curvatures greater than 10 are disregarded when $r = 10$.
3. Orientation assignment: local image gradient directions are assigned to each key point position.
4. Key point descriptor: descriptors are obtained from the region surrounding each key point, incorporating local image gradients and scale information to represent significant shifts in light and local shape distortions.

2.3.2. Sped-Up Robust Feature (SURF) Descriptor

The sped-up robust feature (SURF) descriptor [27] is a faster and more efficient alternative to SIFT for feature extraction in computer vision and image processing. It is based on Haar wavelet responses and the determinant of the Hessian matrix. SURF achieves

comparable performance to SIFT in matching and recognition tasks while significantly improving processing efficiency. Unlike SIFT, which utilizes the difference of Gaussian (DoG) technique to approximate the Laplacian of Gaussian (LoG), SURF employs box filters. This approach offers computational advantages, as box filters can be efficiently computed, and calculations for different scales can be performed simultaneously.

To handle orientation, SURF calculates Haar-wavelet responses in both the x and y directions within a $6s$ neighborhood around each key point, with s being proportional to scale. The orientation is determined by summing the responses in a sliding scanning area.

For feature extraction, a 20 s $\times$ 20 s neighborhood is extracted around each key point and divided into 4 $\times$ 4 cells. Haar wavelet responses are computed for each cell, and the responses from all cells are concatenated to form a 64-dimensional feature descriptor.

The SURF algorithm's implementation involves the following key steps [28]:

- Identifying salient features like blobs, edges, intersections, and corners in specific regions of the integral image. SURF utilizes the fast Hessian detector for feature point detection.
- Utilizing descriptors to characterize the surrounding neighborhood of each feature point. These feature vectors must possess uniqueness while remaining robust to errors, geometric deformations, and noise.
- Assigning orientation to key point descriptors by calculating Haar wavelet responses across image coordinates.
- Ultimately, SURF matching is conducted using the nearest-neighbor approach.

2.3.3. Oriented FAST and Rotated BRIEF (ORB) Descriptor

The oriented FAST and rotated BRIEF (ORB) descriptor [29] is an efficient algorithm for feature extraction in computer vision and image processing. It combines the FAST key point detector [30] with the binary robust independent elementary features (BRIEF) descriptor [31] and introduces rotation invariance. This makes ORB robust to image rotations and enhances its performance in matching and recognition tasks [32]. It achieves comparable performance to other popular descriptors like SIFT and SURF while being significantly faster in computation time.

The ORB method utilizes a simple measure for corner orientation, namely the intensity centroid [33]. First, the moments of a patch are defined as follows:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y). \tag{6}$$

With these moments, the centroid, also known as the 'center of mass' of the patch, can be determined as follows:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right). \tag{7}$$

One can construct a vector from the corner's center O to the centroid $\vec{OC}$. The orientation of the patch is then provided as follows:

$$\theta = \text{atan2}(m_{01}, m_{01}). \tag{8}$$

After calculating the orientation of the patch, it can be rotated to a canonical position, enabling the computation of the descriptor and ensuring rotation invariance. BRIEF (9) lacks rotation invariance; hence, ORB employs rotation-aware BRIEF (rBRIEF) (11). ORB integrates this feature while maintaining the speed advantage of BRIEF:

$$f(n) = \sum_{1 \le i \le n} 2^{i-1} \tau(p; x_i, y_i), \tag{9}$$

where $\tau(p; x, y)$ is defined as follows:

$$\tau(p; x, y)) = \begin{cases} 1 \ : \ p(x) < p(y), \\ 0 \ : \ p(x) \geq p(y), \end{cases} \tag{10}$$

and $p(x)$ is the intensity p at a point x.

The steered BRIEF operator is obtained as follows:

$$g_n(p, \theta) = f_n(p) \mid (x_i, y_i) \in S_\theta. \tag{11}$$

2.3.4. Boosted Efficient Binary Local Image Descriptor (BEBLID)

The boosted efficient binary local image descriptor (BEBLID) [34] is a newer binary descriptor that encodes intensity differences between neighboring pixels. It provides an efficient representation for feature matching and recognition. BEBLID enhances the performance of the real-valued descriptor, boosted efficient local image descriptor (BELID) [35], improving both matching efficiency and accuracy.

The proposed algorithm assumes that there is a training dataset consisting of pairs of images, denoted as $\{(x_i, y_i, l_i)\}_{i=1}^{N}$, where $x_i, y_i \in X$ are labeled with $l_i \in \{-1, 1\}$. Here, $l_i = 1$ indicates that two blocks belong to the same image structure, while $l_i = -1$ indicates that they are different. The objective is to minimize the loss using AdaBoost:

$$L_{\text{BELID}} = \sum_{i=1}^{N} \exp\left(-\gamma \ell_i \sum_{k=1}^{K} \alpha_k h_k(x_i) h_k(y_i) \right), \tag{12}$$

where $g_s(x_i, y_i) = \sum_{k=1}^{K} \alpha_k h_k(x_i) h_k(y_i)$, and γ represents the learning rate parameter. The function $h_k(z) \equiv h_k(z; f; T)$ corresponds to the k-th weak learner (WL) in g_s combined with α_k. The WL depends on the feature extraction function $f(x)$ and threshold T, defined as follows:

$$h_k(z; f; T) = \begin{cases} 1 \ : \ f(x) \leq T, \\ -1 \ : \ f(x) > T, \end{cases} \tag{13}$$

The BEBLID feature extraction function $f(x)$ is defined as follows:

$$f(x; p_1, p_2, s) = \frac{1}{s^2} \left(\sum_{q \in R(p_1, s)} I(q) - \sum_{q \in R(p_2, s)} I(r) \right), \tag{14}$$

where $I(q)$ represents the gray value of pixel q, and $R(p, s)$ is the square image frame centered at p, with an area of s. Thus, $f(x)$ computes the average gray values of pixels in $R(p_1, s)$ and $R(p_2, s)$ and thresholds it. To output binary values $0, 1, -1$ are represented as 0 and $+1$ as 1, resulting in the BEBLID binary descriptor.

$$L_{\text{BEBLID}} = \sum_{i=1}^{N} \exp\left(-\gamma \ell_i \sum_{k=1}^{K} h_k(x) h_k(y) \right). \tag{15}$$

BEBLID has demonstrated superior performance compared to other state-of-the-art descriptors such as SIFT [25], SURF [27], ORB [29], and convolutional neural networks (CNNs) [36] in terms of speed, accuracy, and robustness.

Compared to CNNs, a powerful deep learning approach for feature extraction and recognition, BEBLID is a lightweight and efficient alternative, ideal for low-resource applications. It also offers easier interpretation and debugging due to its binary string representation. Hence, this paper utilizes the advantages and performance of BEBLID as the chosen algorithm.

3. Precise Localization Algorithm

In this paper, an algorithm is proposed to help vehicles position themselves with respect to other nearby vehicles. The approximate distance between two nearby vehicles is computed using GPS data. The exact position between two vehicles cannot be computed using only GPS data because all commercial GPS devices have an intended positioning error [37]. This error might increase further according to various specific conditions, like the number of available satellites, nearby buildings, trees, driving into tunnels, etc. For example, when using a smartphone, the GPS error can be, on average, as much as 4.9 m [38]. Such errors can lead to potentially dangerous situations if any relative vehicle positioning system relies solely on GPS data. For this reason, the aim of this paper is to increase the positioning accuracy using images captured by cameras mounted on each vehicle. Thus, the proposed solution aims to find two similar frames from different vehicles within a certain distance range. Each vehicle sends information about multiple consecutive frames while also receiving similar information from other vehicles for local processing. By using an algorithm to match image descriptors calculated based on these frames, a high number of matches indicates that the vehicles are in relatively the same position. Using the timestamps associated with the two frames, we can determine the moment each vehicle was in that position, allowing us to calculate the distance between them by considering their traveling speed and the time difference between the two.

The proposed approach is decentralized, meaning that each vehicle acts as an independent entity. It has to cover the information exchange with the other vehicles as well as processing the self-acquired and received data. In our model, vehicles employ a V2X communication system with the broadcast information, but they will also use a V2V communication model if the distance to a responding nearby vehicle is below a pre-defined threshold (Figure 1). Each vehicle will broadcast processed information, not being aware if any other vehicle will receive it.

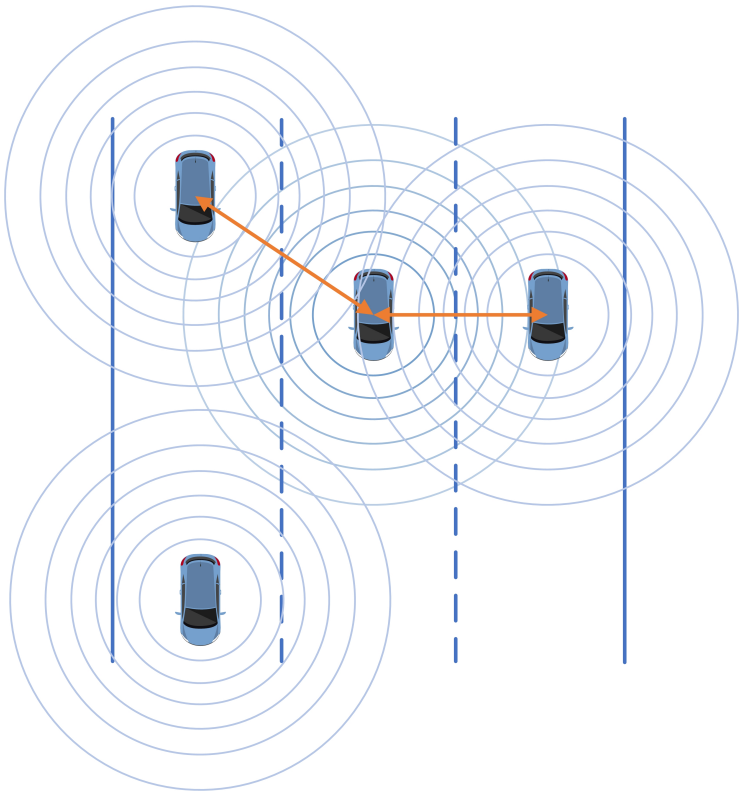


Figure 1. Vehicle communication. First, each vehicle broadcasts data in a V2X communication model (blue circles). Depending on the computed distance between two vehicles, they can start V2V communication (orange arrows).

The proposed algorithm assumes that each vehicle is equipped with an onboard camera with GPS and a computing unit. The GPS indicates the current position of the vehicle in terms of latitude and longitude, as well as the timestamp at that time and the vehicle's speed. The vehicle computing unit will handle all computations and communications, so it will process the data and send it to the other vehicles. Also, the processing unit will receive data from other vehicles that are nearby. The processing unit must determine, based on the received information, if a V2V communication can start between the two vehicles. If it can, it will begin an information exchange with the other vehicle and will process the subsequent received data. This means that each vehicle has two roles: the first one involves data processing and communication, while the second involves receiving messages from other nearby vehicles and analyzing them. As the paper does not focus on the communication model itself but rather on the image processing part, we employed a very simple and straightforward communication model. This model cannot be used in real-world applications, where security, compression, and other factors must be taken into consideration. Our main focus when developing the communication model was the main processing steps needed from the image processing point of view. The send and receive roles are described in the following subsection.

3.1. Message Transmission Procedure

To avoid sending large amounts of irrelevant data between vehicles, a handshake system must be defined first. This will prevent congestion in any communication technique. The handshake system allows all vehicles to broadcast their GPS position. As a low-ranged communication system is assumed, only nearby vehicles will receive this message. Any nearby vehicle that receives this broadcast message will compute the distance between the sending vehicle and itself, and if the distance is lower than a threshold, it will send back a start communication message. The distance threshold should be around 15 to 20 m. This will take into consideration both the GPS errors and the minimum safety distance between two vehicles. Also, note that, as most GPS systems record data once per second and the video records at a much greater rate, synchronization between the GPS coordinates and each frame must be performed. In other words, if the camera records 30 frames per second, it means that 30 frames will have attached the same GPS coordinate. For example, a car driving at 60 km/h will travel 16.6 m/s during this time interval. Thus, during these approximately 17 m, the messages sent by the vehicle will have the same GPS position, leading to potentially dangerous situations if data from images is not taken into consideration.

After establishing that the two vehicles are close enough and that image data has to be sent between them, the next question to be answered is exactly which data should be exchanged. Sending the entire video stream is unfeasible due to high bandwidth requirements, so the most straightforward approach is to compute key points for each frame. Then, for each detected key point, a descriptor is computed. All these steps are presented in the flowchart illustrated in Figure 2. Once the descriptors have been computed, every piece of information related to the current frame is serialized and sent to other paired vehicles. This includes the timestamp, latitude, longitude, speed, key points, and descriptors.

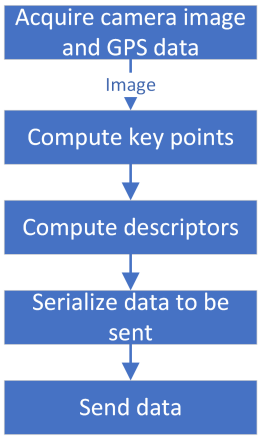


Figure 2. Send message architecture overview.

3.2. Message Reception Procedure

After passing the handshake system described before, each vehicle will only receive messages from nearby vehicles. The message will contain data about GPS positioning and processed image data. The steps these data will take are presented in Figure 3 and described in detail in the following subsection.

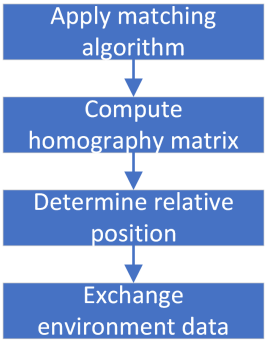


Figure 3. Receive message architecture overview.

Each vehicle will also have its own video stream that is processed locally. This means that, for each frame, the key points and their descriptors are computed. These will have to be matched against the key points and descriptors received from other vehicles. There are various algorithms developed for feature matching, but the most used ones are brute-force matcher (BFMatcher) [39] and fast library for approximate nearest neighbors (FLANN) [40]. Brute-force matcher matches one feature descriptor from the first set with all features in the second set using distance computation to find the closest match. FLANN is an optimized library for fast nearest neighbor search in large datasets and high dimensional features, which works faster than BFMatcher for large datasets and requires two dictionaries specifying the algorithm and its related parameters.

The matching algorithm will output the matched descriptors, which are the descriptors that correspond to two matched pixels in the input images. Usually, filtering is carried out to remove the outliers, i.e., points that have the same descriptors and do not correspond to matching pixels in the input images. Having the two sets of matched descriptors, the next step is to determine the relative position between them. In other words, at this point, a set of matched points is available, but the location of the points from the received image (their descriptors) in the current vehicle's frame is unknown. This will determine where the two vehicles are positioned with respect to each other: if the points are located in the center of the image, it means that the two vehicles are in the same lane, one in front of the other. If the points are located to the side of the image, it means that the two vehicles are on separate lanes, close to each other. In other words, if corresponding matched points are

located on the right side of the first image and on the left side of the second image, then the first vehicle is on the right side of the second vehicle.

One way to determine the points' relative position is to compute a homography matrix that transforms a point in the first image into a point in the second image. Once the homography matrix is computed, it can be applied to the points from the first image to see where those points are in the second image. In this way, the two vehicles can be relatively positioned in relation to each other.

Of course, it might happen that the current frame from the receiving vehicle corresponds to an older or newer frame from the receiving car. In these cases, multiple comparisons between the current frame and the received frames must be performed. By combining all these pieces of information, the vehicle can detect the location of each nearby vehicle accurately and efficiently, and the exact steps for these are described in detail in the following section regarding algorithm implementation.

4. Framework for Algorithm Implementation

The following section details the implementation of the algorithm, which involves several steps: resizing the image dimensions, extracting the camera-displayed time, simulating the communication process, detecting the corresponding frame, defining the distance calculation formula, and outlining the hardware equipment utilized.

4.1. Adjust Image Dimensions

Considering that the camera captures a significant part of the car dashboard (see, for example, Figure 4), this aspect can negatively influence the image-matching algorithm. It is important to note that this particular area is not relevant for the intended purposes. Furthermore, in that area, information from the camera is displayed, such as the camera name, current date, and speed. These elements can also affect the performance of the used image descriptors. For this reason, the decision was made to crop out the respective area from the original image, thus eliminating irrelevant information and retaining only the essential data for the intended purposes. The cropped area corresponds to the region beneath the red line in Figure 4, and this approach enables greater precision in image matching and enhances the algorithm's performance regarding the specific intended objectives.

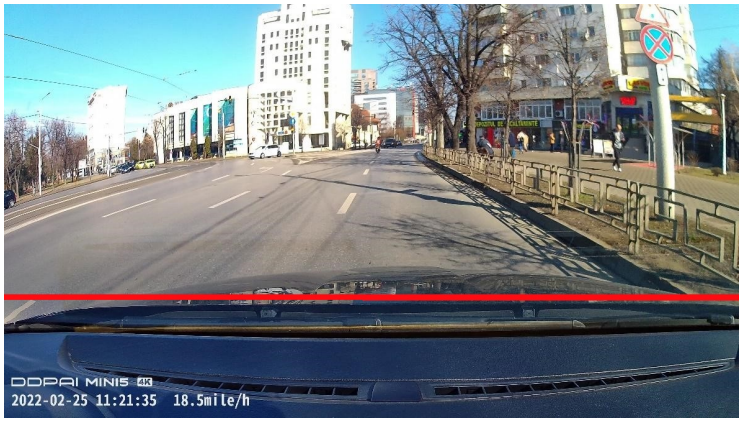


Figure 4. Cropping out non-relevant image areas: enhancing data relevance and algorithm efficiency.

4.2. Extract Camera Displayed Time

It is necessary to consider that most GPS systems record data once per second, while video recording is carried out at a much higher rate. Therefore, meticulous synchronization between GPS coordinates and each video frame is required. To put it simply, if the camera records 30 frames per second, it means that 30 frames will have the same GPS coordinate attached. To address this issue and ensure the accuracy of our data, optical character recognition (OCR) technology was utilized to extract time information from the images provided by the camera. We specified the exact area where the time is displayed in the image and

assigned the corresponding GPS coordinates to that moment. This synchronization process has allowed us to ensure that GPS data are accurately correlated with the corresponding video frames, which is essential for the subsequent analysis and understanding of our information.

To extract text from the image, we used Tesseract version 5.3.0, an open-source optical character recognition engine [41]. Tesseract is renowned for its robustness and accuracy in converting images containing text into editable content.

4.3. Simulation of Vehicle-to-Vehicle Communication

In our system, a vehicle extracts key points from each frame to obtain significant information about the surrounding environment. Additionally, information is retrieved from the GPS system, providing data about latitude, longitude, and crucial details about the number of lanes and vehicle speed, using the specialized function.

To ensure the exchange of information with other involved vehicles, a function was developed to simulate message transmission. As stated in Section 3, our paper focuses mainly on image processing and not on the communication itself. This is why we use a simulation model, in order to prove the feasibility of the proposed method. The function that simulates the communication is responsible for transmitting the processed data to other vehicles. This vehicle-to-vehicle communication is simulated through a file where the information is stored and later read.

To receive and process messages from other vehicles, a function for message reception is utilized. This simple function reads information from the specific file and extracts relevant data to make decisions and react appropriately within our vehicle communication system.

Overall, this architecture enables us to successfully collect, transmit, and interpret data to facilitate efficient communication and collaboration among vehicles in our project.

4.4. Detection of the Corresponding Frame

To detect the frame received from another vehicle within the current vehicle's frame sequence, we used an approach relying on the number of matches between the two frames. Thus, the frame with the highest number of matches in the video recording was selected.

When received information about a frame is searched in a video, the closer it gets to the corresponding frame, the higher the number of matches increases (as observed in Figure 5). Essentially, the closer the current vehicle gets to the position where that frame was captured, the more similar the images become. With some exceptions that will be discussed at the end of the paper, this approach proved valid during all our tests.

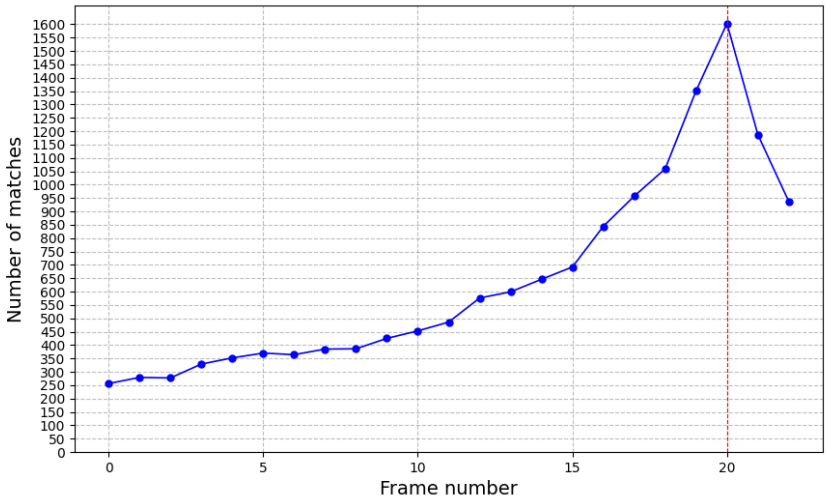


Figure 5. Observing frame proximity in video analysis: closer vehicle positioning correlates with increased image similarity and match frequency.

Implementing this algorithm involved monitoring the number of matches for each frame and retaining the frame with the most matches. An essential aspect was identifying two consecutive decreases in the number of matches, at which point the frame with the highest number of matches up to that point was considered the corresponding frame.

4.5. Compute Distance

After detecting the corresponding frame in the video stream, the next step involves computing the distance between vehicles and determining their positions. This can be achieved using information from the two vehicles associated with these frames, such as the timestamp and speed.

Thus, if the timestamp from the current vehicle is greater than that of the other vehicle, it indicates that the latter is in front of the current one. The approach to comparing the two timestamps is presented in Figure 6. By knowing the speed of the front vehicle and the time difference between the two vehicles, the distance between them is computed.

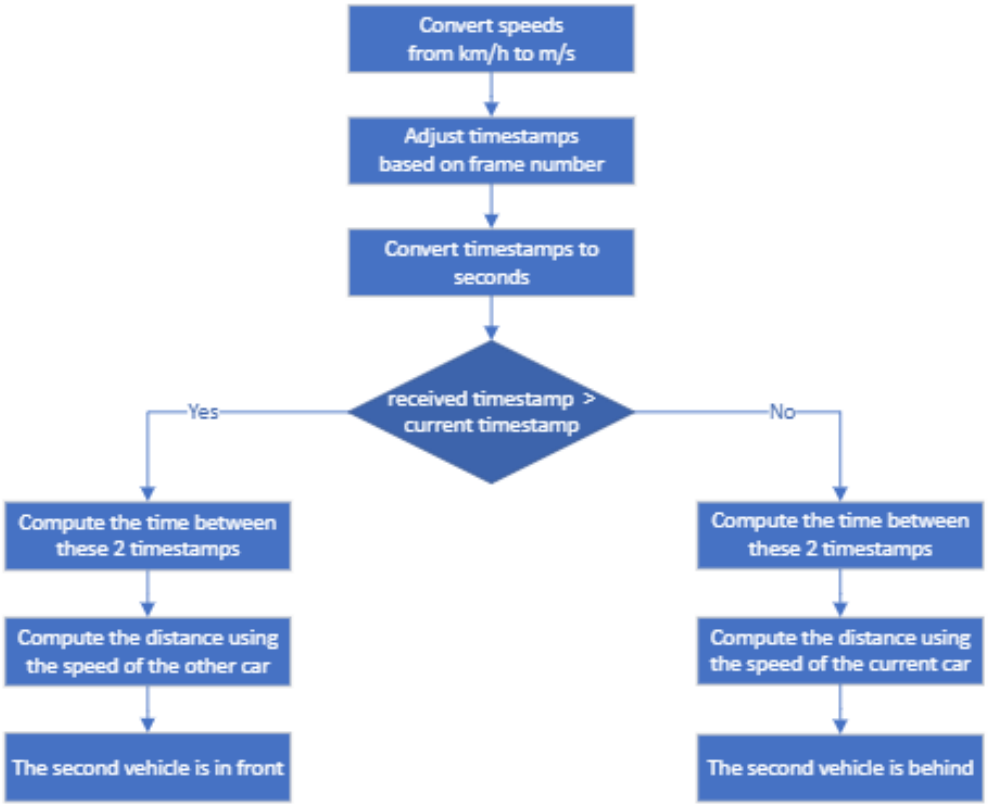


Figure 6. Determining neighboring vehicle relative position using matched frame timestamps.

Conversely, if the timestamp from the current vehicle is smaller than that of the other vehicle, it suggests that the latter is behind the current one. With the speed of the current vehicle and the time difference between the two vehicles, the distance between them can still be computed.

Given that the video operates at a frequency of 30 frames per second and GPS data is reported every second, each of the 30 frames contains the same set of information. However, this uniformity prevents the exact determination of distance because both frame 1 and frame 30 will have the same timestamp despite an almost 1-second difference between the two frames.

To enhance the accuracy of distance computation between the two vehicles, adjustments are made to the timestamp for the frames from both vehicles. In addition to other frame details, the frame number reported with the same timestamp (ranging from 1 to 30) is transmitted. In the distance computation function, the timestamp is adjusted by adding

the current frame number divided by the total number of frames (30). For instance, if the frame number is 15, 0.5 s are added to the timestamp.

In Figure 7, the method of computing distance assuming that Vehicle 1 is in the front and Vehicle 2 is behind is detailed. Frame V1 from Vehicle 1, which is the x-th frame at timestamp $T1$, is detected by Vehicle 2 as matching with frame V2, which is the y-th frame at timestamp $T2$. To determine the position relative to Vehicle 1, the other vehicle needs to compute the distance traveled by the first vehicle in the time interval from timestamp $T1$ to the current timestamp $T2$, taking into account its speed.

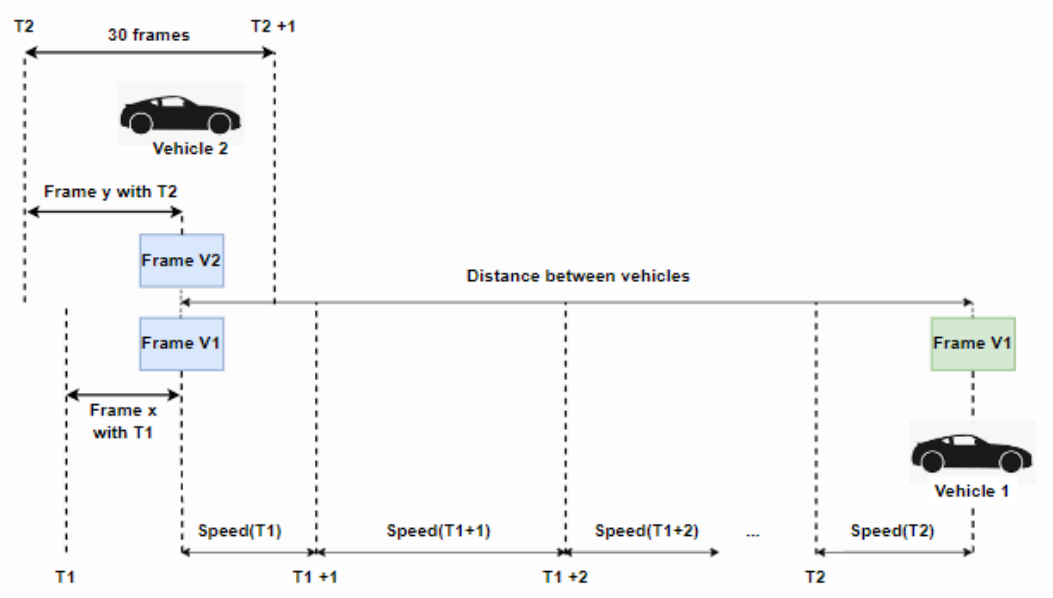


Figure 7. Distance estimation between two vehicles through frame matching and timestamp comparison.

To compute the distance as accurately as possible, the speed reported at each timestamp is considered, and the calculation formula is presented in Equation (16). Since Frame V1 is the x-th frame at timestamp $T1$, and considering that there are 30 frames per second, the time remaining until timestamp $T1 + 1$ second can be determined. Then, this time interval is multiplied by speed $S1$ at timestamp $T1$ to determine the distance traveled in this interval. The distance traveled from timestamps $T1 + 1$ to $T2 - 1$ is determined by multiplying the speeds $S1$ at these timestamps by 1 s each. To determine the distance traveled from $T2$ to the frame y-th, the speed $S1$ at $T2$ is multiplied by $y/30$. By summing all these distances, the total distance is obtained.

$$\begin{aligned} \text{distance} = & \sum_{j=x-1}^{29} \frac{1}{30}(S_1(T_1) + j\frac{S_1(T_1+1) - S_1(T_1)}{30}), \\ & + \sum_{T_i=T_1+1}^{T_2-1} \sum_{j=0}^{29} \frac{1}{30}(S_1(T_i) + j\frac{S_1(T_i+1) - S_1(T_i)}{30}), \\ & + \sum_{j=0}^{y-1} \frac{1}{30}(S_1(T_2) + j\frac{S_1(T_2+1) - S_1(T_2)}{30}). \end{aligned} \tag{16}$$

4.6. Hardware Used

For the developed solution, 2 DDPAI MOLA N3 cameras were utilized, each featuring a 2k resolution and operating at a frame rate of 30 frames per second. These cameras feature built-in GPS functionality that accurately records the vehicle's location and speed. The advantage of these cameras lies in their GPS data storage format, which facilitates the seamless retrieval of this information. Cameras with identical resolutions were selected

to ensure consistency, as not all image descriptors maintain scale invariance, which could otherwise affect algorithm performance.

5. Performed Experiments and Test Results

Based on the implementation presented in the previous section, a series of tests were conducted to demonstrate both the feasibility of the algorithm and its performance. The performances of the BEBLID, ORB, and SIFT descriptors were tested, as well as how the number of features influences frame detection. Finally, a comparison between the distance calculated by the proposed algorithm, the one calculated based on GPS data, and the measured distance is presented to illustrate the preciseness of the proposed algorithm in real-world applications. This comparison shows that the algorithm reflects a high degree of accuracy when validated against physically measured distances, which demonstrates its potential effectiveness for applications requiring precise distance calculations, e.g., vehicle platooning applications.

5.1. Test Architecture

To prove the feasibility and robustness of the proposed algorithm, we conducted various tests in real-world scenarios. For the first test, a vehicle equipped with a dashboard camera was used to make two passes on the same streets, resulting in two video recordings. The main purpose of this test was to determine what descriptors work best and if the proposed system performs well when eliminating the errors caused using multiple cameras. For this purpose, for a frame extracted from the first video (left picture from Figure 8), we had to find the corresponding frame in the second video (right picture from Figure 8). Additionally, the performances of three different descriptors were compared, namely SIFT, ORB, and BEBLID, in terms of matching accuracy and speed. This comparison allows us to evaluate the strengths and limitations of each descriptor in the context of our experiment.

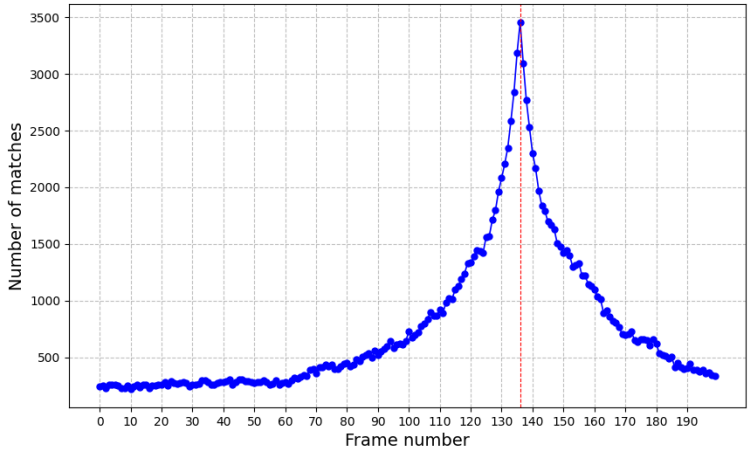
For the selected frame and each frame in the second video, the following steps were performed:

- In total, 10,000 key points were detected using the ORB detector for each frame.
- Based on these key points, the descriptors were computed, and the performances of SIFT, ORB, and BEBLID descriptors were compared.
- For feature matching, the brute-force descriptor matcher was used for ORB and BEBLID, which are binary descriptors. This technique compares binary descriptors efficiently by calculating the Hamming distance. As for SIFT, a floating-point descriptor, the FLANN descriptor matcher, was employed. FLANN utilizes approximate nearest neighbor search techniques to efficiently match floating-point descriptors.
- The frame with the highest number of common features with the selected frame from the first video is considered as its corresponding frame in the second video. This matching process is based on the similarity of visual features between frames, allowing us to find the frame in the second video that best corresponds to the reference frame from the first video. In Figure 8, an example of the identified corresponding frame is presented.

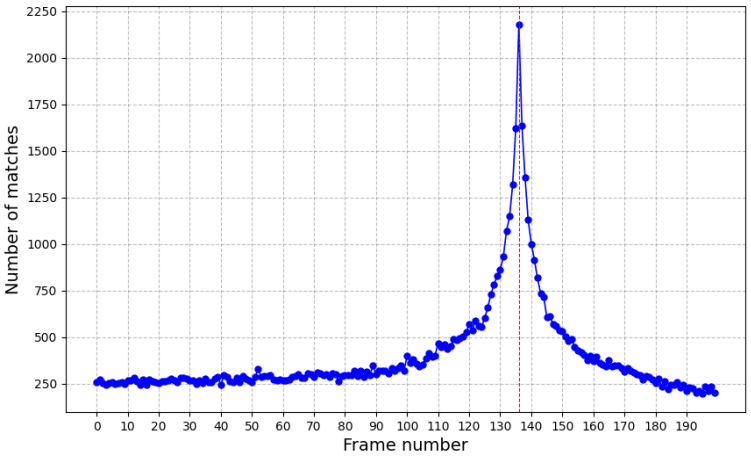
Figure 8. Two corresponding frames from two video sequences.

The SIFT descriptor is considered a gold-standard reference but requires a significant amount of computational power. It has a feature vector of 128 values. This descriptor managed to match the reference frame with frame 136 from the second video with a total of 3456 matches (Figure 9a). The ORB descriptor is considered one of the fastest algorithms and has a feature vector of 32 values. It also successfully detected frame 136 from the second video with 2178 matches, as shown in Figure 9b. According to the authors, the BEBLID descriptor achieves results similar to SIFT and surpasses ORB in terms of accuracy and speed, with a feature vector of 64 values. However, in our specific test case, the BEBLID descriptor managed to detect frame 136 but with a lower number of matches, specifically 2064 (as shown in Figure 9c) compared to ORB. This discrepancy could be attributed to the specific conditions of our experiment, such as variations in lighting, perspective, or the content of the frames.

As a result of this first experiment, all three descriptors considered successfully match two frames from different videos, but with the same camera, even if their content is slightly different. These differences, such as variations in traffic conditions, will also occur when video sequences form different vehicles are used.

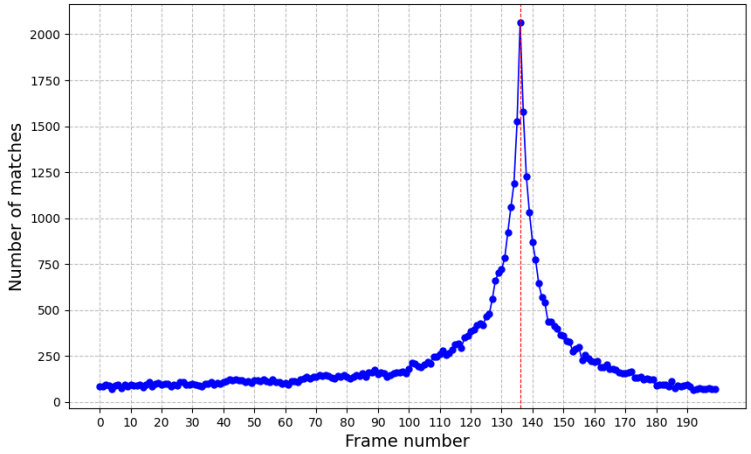


(**a**) SIFT Matches



(**b**) ORB Matches

Figure 9. *Cont.*



(**c**) BEBLID Matches

Figure 9. Comparison of Matching Results for SIFT, ORB, and BEBLID.

5.2. Descriptor Performance Test

The underlying concept of this test involved the deployment of two vehicles equipped with dashboard cameras driving on the same street. As they progressed, the cameras recorded footage, resulting in two distinct videos.

Using these video recordings, the objective of the test was to identify 50 consecutive frames from the leading vehicle within the footage captured by the trailing vehicle. Each frame from the first video was compared with 50 frames from the second video, and the frame with the highest number of matches was taken into consideration. This aimed to ascertain the algorithm's capability to consistently detect successive frames, thus showcasing its robustness. Furthermore, a secondary aim was to evaluate the performance of the three descriptors used in the process. In Figure 10a, one of the frames from the car in front (left) and the matched frame from the rear car are presented (right). In the frame on the right, the front car is also visible.

(**a**) Frame 1 Video 1 (**b**) Frame 26 Video 2

Figure 10. Two matched frames from vehicles.

In Table 1, the results of the three descriptors for a total of 20,000 features are presented. BEBLID successfully detected 39 frames correctly, with instances of incorrect detections shown in blue in the table. These incorrect detections exhibit a minor deviation by detecting a frame either preceding or following the actual frame, which poses no significant concern.

Table 1. Results of descriptor analysis with color indications: blue for incorrect detections, red for frames detected from behind, and orange for differences exceeding 1 frame.

Frame Video 1	Frame Video 2	BEBLID	ORB	SIFT
1	26	26–3992 matches	26–4964 matches	26–5935 matches
2	27	26–3848 matches	26–4885 matches	26–5970 matches
3	28	28–3929 matches	28–4970 matches	29–5984 matches
4	29	29–3921 matches	29–5053 matches	29–6052 matches
5	30	30–3868 matches	29–4918 matches	30–5958 matches
6	31	31–3890 matches	31–4953 matches	32–5878 matches
7	32	33–3923 matches	32–5016 matches	32–5957 matches
8	33	33–3991 matches	33–4996 matches	34–5927 matches
9	34	34–3900 matches	33–4864 matches	35–5884 matches
10	35	35–3974 matches	35–4949 matches	36–5957 matches
11	36	36–3884 matches	36–4927 matches	35–5893 matches
12	37	37–3894 matches	37–5061 matches	37–5925 matches
13	38	38–3895 matches	38–5044 matches	38–5875 matches
14	39	39–3837 matches	38–4964 matches	40–5838 matches
15	40	40–3906 matches	40–4846 matches	40–5817 matches
16	41	41–3848 matches	41–4937 matches	41–5886 matches
17	42	42–3966 matches	42–4995 matches	42–5880 matches
18	43	43–3764 matches	43–4835 matches	44–5980 matches
19	44	45–3744 matches	45–4772 matches	44–5787 matches
20	45	45–3780 matches	45–4903 matches	45–5862 matches
21	46	46–3809 matches	45–4776 matches	46–5796 matches
22	47	47–3891 matches	47–5009 matches	48–5998 matches
23	48	48–3939 matches	48–4956 matches	49–6043 matches
24	49	50–3645 matches	48–4802 matches	49–6017 matches
25	50	50–3678 matches	50–4592 matches	50–6011 matches
26	51	51–3695 matches	51–4737 matches	50–6019 matches
27	52	52–3672 matches	52–4741 matches	52–5986 matches
28	53	53–3544 matches	52–4599 matches	53–5900 matches
29	54	54–3755 matches	54–4709 matches	55–6031 matches
30	55	55–3793 matches	55–4796 matches	55–5976 matches
31	56	55–3582 matches	55–4560 matches	55–5971 matches
32	57	56–3486 matches	56–4591 matches	58–5926 matches
33	58	58–3600 matches	57–4582 matches	58–5985 matches
34	59	59–3639 matches	59–4584 matches	60–6079 matches
35	60	60–3628 matches	60–4704 matches	59–6052 matches
36	61	61–3584 matches	61–4609 matches	62–5965 matches
37	62	62–3650 matches	62–4575 matches	63–5851 matches
38	63	63–3653 matches	63–4523 matches	63–5946 matches
39	64	63–3584 matches	63–4512 matches	63–5909 matches
40	65	65–3435 matches	65–4402 matches	66–5875 matches
41	66	65–3449 matches	66–4349 matches	67–5745 matches
42	67	67–3565 matches	67–4410 matches	68–5967 matches
43	68	68–3393 matches	68–4320 matches	68–6003 matches
44	69	70–3343 matches	69–4356 matches	71–5825 matches
45	70	70–3477 matches	70–4399 matches	71–5940 matches
46	71	71–3434 matches	71–4389 matches	71–5929 matches
47	72	73–3288 matches	72–4282 matches	73–5971 matches
48	73	73–3135 matches	73–4119 matches	73–5960 matches
49	74	75–3193 matches	74–4048 matches	73–5884 matches
50	75	75–3199 matches	75–4079 matches	75–5955 matches
Number of Correct Detections		39	38	23

Note that, the frames shown in blue in Table 1 might be caused by the fact that a perfect synchronization between frames of the used videos cannot be accomplished. For example, the first car traveled at 21.9 km/h or 6.083 m/s records a frame every 0.2 m (considering 30 frames per second). This sampling rate might, in our opinion, cause some

of the slightly incorrect detections presented in blue in Table 1. This is the reason to use the blue color, because they might result from the sampling rate of the used cameras and not by an actual error in the matching algorithm.

Similarly, ORB shows good performance by correctly detecting 38 frames. However, the performance of SIFT falls short of expectations, with only 23 out of 50 frames being detected accurately. Additionally, for SIFT, there are cases when it detected a frame from behind after the detection of a subsequent frame, indicated in red in the table. Moreover, the case when the difference between the correct frame and the predicted one is greater than 1 frame is highlighted in orange in the table. Another downside of using SIFT is that it has more cases with three consecutive detections of the same frame than the other two descriptors (BEBLID-0, ORB-1 (frame 45), SIFT-4 (frames 55, 63, 71, 73)). Also, when using SIFT, there are cases when two consecutive detected frames differ by three frames (frames 29, 58, 66 and 71), which is a case that was not encountered when using BEBLID of ORB.

Furthermore, it is noteworthy to highlight that a higher number of matches, as observed in the case of SIFT, does not necessarily translate to better performance. Despite BEBLID having a lower number of matches compared to the other two descriptors, it achieved the highest performance in this test.

These findings underscore the importance of not only relying on the quantity of matches but also considering the accuracy and robustness of the detection algorithm. In this context, BEBLID stands out as a promising descriptor for its ability to deliver reliable performance even with a comparatively lower number of matches.

It is worth mentioning that, the frames written in orange and red are most likely errors in the detection algorithm and can lead to potentially dangerous situations if their number increases.

5.3. Influence of the Number of Features

In this test, the objective was to analyze the influence of the number of features associated with each descriptor on its performance. As the computational time increases with a higher number of features, we examined the performance of the three descriptors across a range of feature numbers, from 20,000 down to 5000.

The test methodology involved detecting 20 frames from the first video against 20 frames from the second video. This approach facilitated an assessment of how varying feature counts affected the accuracy and efficiency of frame detection for each descriptor.

As observed in Table 2, the number of matches per frame decreased as the number of features decreased. For BEBLID, if the number of features decreased from 20,000 to 10,000, the performance did not decline considerably. In fact, for a feature count of 16,000, we achieved the highest number of correctly detected frames, with 18 out of 20. However, if the number of features dropped below 10,000, performance deteriorated significantly.

The results for ORB can be observed in Table 3. For a feature count of 12,000 and 10,000, we achieved 16 out of 20 correctly detected frames. However, if the feature count dropped below 10,000, the performance deteriorated.

From Table 4, it is clear that the number of correctly detected frames varies depending on the number of features for SIFT. However, overall, this descriptor exhibits poor performance in all cases.

Based on the outcomes of the last two tests, we can conclude that BEBLID generally achieves better results, with the exception being when the number of features is 12,000, where ORB detects 16 frames correctly compared to BEBLID's 15, see Figure 11. ORB also shows satisfactory results, whereas the performance of SIFT is not as commendable. For the presented reasons, we will use only the BEBLID descriptor in further tests.

Table 2. BEBLID—The influence of the number of features on the frame detection: blue for incorrect detections, red for frames detected from behind.

Frame Video 1	Frame Video 2	20,000	18,000	16,000	14,000	12,000	10,000	8000	6000	5000
1	26	26– 3992	26–3637	26–3263	26–2840	26–2442	26–2013	26–1672	26–1284	26–1065
2	27	26–3848	26–3483	26–3106	26–2755	26–2374	27–1974	26–1605	26–1252	26–1047
3	28	28–3929	28–3578	28–3218	28–2785	28–2392	28–2007	28–1667	28–1298	28–1095
4	29	29–3921	29–3575	29–3206	29–2815	29–2449	29–2052	30–1653	29–1274	30–1065
5	30	30–3868	30–3546	30–3210	30–2798	30–2442	30–2042	30–1671	29–1262	30–1074
6	31	31–3890	31–3539	31–3151	31–2753	32–2368	31–2016	30–1633	30–1252	30–1039
7	32	33–3923	32–3560	32–3183	32–2837	32–2434	33–2017	33–1643	33–1221	33–1037
8	33	33–3991	33–3616	33–3235	33–2857	33–2481	33–2111	33–1711	33–1289	33–1083
9	34	34–3900	34–3560	34–3183	33–2803	33–2442	35–2045	35–1643	34–1249	35–1056
10	35	35–3974	35–3609	35–3258	35–2887	35–2447	35–2049	35–1652	35–1244	35–1043
11	36	36–3884	35–3534	36–3159	36–2789	36–2415	36–2011	35–1629	36–1233	36–1048
12	37	37–3894	37–3508	37–3181	37–2793	37–2407	38–2025	38–1619	38–1248	36–1069
13	38	38–3895	38–3514	38–3153	38–2774	38–2407	38–2016	38–1640	38–1230	38–1023
14	39	39–3837	39–3512	39–3168	39–2813	39–2406	39–1986	39–1608	39–1227	39–1024
15	40	40–3906	40–3562	40–3167	40–2822	40–2441	40–2014	40–1650	40–1248	40–1047
16	41	41–3848	41–3483	41–3084	41–2717	41–2364	41–1922	42–1585	42–1201	42–1008
17	42	42–3966	42–3564	42–3203	42–2800	42–2410	42–2037	42–1619	42–1285	42–1088
18	43	43–3764	43–3431	43–3071	43–2679	44–2299	43–1937	43–1564	43–1208	42–993
19	44	45–3744	43–3364	43–3022	45–2692	43–2326	45–1967	45–1609	45–1218	43–1016
20	45	45–3780	45–3426	45–3069	45–2715	45–2336	45–1958	45–1569	45–1238	45–1017
Correct Detection		17	17	18	17	15	16	11	13	11

Table 3. ORB—The influence of the number of features on the frame detection: blue for incorrect detections, and orange for differences exceeding 1 frame.

Frame Video 1	Frame Video 2	20,000	18,000	16,000	14,000	12.000	10,000	8000	6000	5000
1	26	26–4964	26–4481	26–4016	26–3542	26–3052	26–2559	26–2170	26–1652	26–1379
2	27	26–4885	26–4425	28–3951	26–3487	28–3026	27–2538	26–2104	26–1618	28–1352
3	28	28–4970	28–4533	28–4095	28–3622	28–3118	28–2614	28–2161	28–1670	28–1412
4	29	29–5053	29–4614	29–4135	29–3664	29–3169	29–2668	29–2162	29–1660	29–1394
5	30	29–4918	30–4485	30–4069	30–3570	30–3104	30–2609	30–2125	29–1612	30–1330
6	31	31–4953	30–4480	30–4066	30–3566	30–3070	30–2592	30–2121	29–1600	30–1338
7	32	32–5016	32–4579	32–4079	32–3614	32–3159	32–2583	32–2119	32–1599	33–1348
8	33	33–4996	33–4555	33–4103	33–3632	33–3175	33–2688	33–2197	33–1660	33–1388
9	34	33–4864	33–4445	33–4008	33–3538	34–3105	34–2568	34–2107	33–1572	33–1320
10	35	35–4949	35–4534	35–4089	35–3628	35–3102	35–2621	35–2092	35–1593	35–1337
11	36	36–4927	35–4495	35–4033	36–3541	36–3069	36–2569	35–2100	36–1592	35–1346
12	37	37–5061	37–4566	37–4089	37–3615	37–3137	37–2615	36–2092	36–1602	36–1373
13	38	38–5044	38–4560	38–4055	37–3574	38–3142	38–2642	38–2143	38–1639	38–1366
14	39	38–4964	38–4482	39–4046	38–3545	38–3108	38–2619	38–2108	38–1606	38–1362
15	40	40–4846	40–4410	39–3942	39–3522	40–3010	39–2517	39–2095	39–1597	39–1312
16	41	41–4937	41–4477	41–3954	41–3510	41–3036	41–2479	40–2051	40–1570	40–1326
17	42	42–4995	42–4507	42–4057	42–3608	42–3055	42–2559	42–2113	42–1647	42–1384
18	43	43–4835	43–4394	43–3971	42–3510	43–3044	43–2584	42–2061	42–1635	42–1349
19	44	45–4772	43–4310	43–3879	45–3433	43–2987	43–2531	43–2062	43–1575	43–1349
20	45	45–4903	45–4439	45–4005	45–3521	45–3018	45–2536	45–2030	45–1566	45–1293
Correct Detection		15	14	14	12	16	16	11	10	9

Table 4. SIFT—The influence of the number of features on the frame detection: blue for incorrect detections, red for frames detected from behind, and orange for differences exceeding 1 frame.

Frame Video 1	Frame Video 2	20,000	18,000	16,000	14,000	12,000	10,000	8000	6000	5000
1	26	25–5994	25–5474	25–5002	26–4504	26–3967	25–3385	26–2844	26–2169	25–1869
2	27	27–6042	28–5556	27–5054	27–4574	27–4059	27–3430	27–2942	27–2206	28–1871
3	28	28–6036	30–5499	28–4996	28–4511	29–3944	30–3413	28–2871	28–2196	30–1859
4	29	29–6066	31–5441	29–5007	31–4544	29–4050	29–3485	31–2904	31–2204	29–1839
5	30	30–5954	31–5496	29–4990	31–4475	30–4023	30–3459	31–2852	31–2250	31–1899
6	31	32–5904	32–5441	31–4978	32–4506	31–3980	31–3396	30–2847	31–2229	31–1912
7	32	32–5926	32–5437	33–4944	32–4503	33–3977	33–3430	33–2822	33–2188	32–1874
8	33	34–5874	34–5419	34–4972	32–4437	32–3900	34–3399	32–2817	33–2200	34–1903
9	34	35–5873	35–5450	35–4891	35–4439	35–3944	35–3410	34–2782	34–2155	35–1846
10	35	36–5953	35–5469	36–4960	35–4558	35–4027	35–3404	36–2806	35–2163	36–1845
11	36	36–5938	35–5474	35–4979	35–4441	35–3926	35–3411	36–2760	36–2174	36–1839
12	37	37–5892	37–5407	37–4964	37–4463	37–3930	38–3369	37–2775	36–2205	39–1836
13	38	38–5930	38–5468	38–4953	38–4400	38–3901	40–3369	38–2784	39–2159	39–1881
14	39	40–5836	40–5352	40–4902	40–4343	40–3893	41–3373	40–2785	39–2203	39–1853
15	40	40–5831	40–5425	40–4978	40–4386	39–3900	39–3346	40–2793	39–2247	39–1917
16	41	42–5928	42–5482	43–4970	41–4484	41–3902	42–3356	42–2786	41–2203	42–1877
17	42	42–5894	42–5501	42–4928	42–4410	42–3890	42–3428	42–2788	42–2195	42–1867
18	43	44–5981	44–5503	42–4947	44–4449	44–3978	42–3385	42–2803	43–2173	43–1844
19	44	44–5757	45–5310	45–4858	45–4421	44–3846	45–3360	45–2715	45–2162	43–1814
20	45	46–5855	46–5408	45–4978	45–4451	45–3904	45–3387	45–2789	45–2148	43–1847
Correct Detection		11	6	9	11	12	7	10	13	7

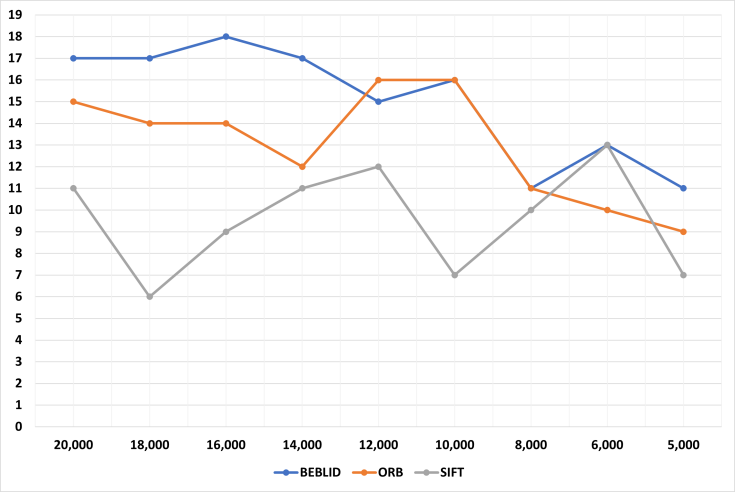


Figure 11. Comparison of correct detection rates for BEBLID, ORB, and SIFT descriptors across different feature counts.

5.4. Distance Computation Test

The objective of this test was to evaluate the distance calculation algorithm and compare the distance calculated based on the proposed algorithm with the distance calculated based on GPS coordinates.

Thirty frames were used as a reference from the car in front, and attempts were made to detect them in the video stream from the car behind using the BEBLID descriptor. As can be observed in Table 5, the detected frames are mostly consecutive. Additionally, it is evident that the distance calculated based on GPS coordinates is significantly larger than the one calculated by the algorithm.

Table 5. First test—The computed distance between the vehicles (Car A in Front, Car B Behind) for 30 consecutive frames.

Frame Video 1	Frame Video 2	GPS Distance	Computed Distance
1	33–2267 matches	34.153 m	20.694 m
2	33–2215 matches	34.153 m	20.516 m
3	35–2175 matches	34.153 m	20.681 m
4	37–2162 matches	34.153 m	20.846 m
5	38–2134 matches	34.153 m	20.839 m
6	39–2108 matches	34.153 m	20.832 m
7	40–2207 matches	34.153 m	20.825 m
8	41–2120 matches	33.463 m	20.653 m
9	43–2151 matches	33.463 m	21.005 m
10	45–2139 matches	33.463 m	21.181 m
11	44–2167 matches	33.463 m	20.827 m
12	45–2128 matches	33.463 m	20.826 m
13	47–2186 matches	33.463 m	21.000 m
14	49–2128 matches	33.463 m	21.173 m
15	49–2123 matches	33.463 m	20.996 m
16	50–2089 matches	33.463 m	20.993 m
17	52–2100 matches	33.463 m	21.164 m
18	52–2122 matches	33.463 m	20.987 m
19	53–2128 matches	33.463 m	20.984 m
20	54–2026 matches	33.463 m	20.980 m
21	56–1985 matches	33.463 m	21.148 m
22	56–2008 matches	33.463 m	20.972 m
23	57–1980 matches	33.463 m	20.968 m
24	58–2079 matches	33.463 m	20.963 m
25	60–2065 matches	33.463 m	21.128 m
26	62–2020 matches	33.463 m	21.292 m
27	62–2079 matches	33.463 m	21.117 m
28	64–2105 matches	33.463 m	21.279 m
29	64–2013 matches	33.463 m	21.104 m
30	66–2133 matches	33.463 m	21.265 m

A second test was conducted by reversing the order of the two cars. What is observed in this test is that, in this case, the distance calculated based on GPS coordinates is smaller than the distance calculated by the proposed algorithm. These results are presented in Table 6.

Table 6. Second Test—The computed distance between the vehicles(Reversed Order: Car B in Front, Car A Behind) for 30 consecutive frames.

Frame Video 1	Frame Video 2	GPS Distance	Computed Distance
1	25–2255 matches	9.085 m	24.025 m
2	26–2222 matches	9.085 m	24.006 m
3	27–2232 matches	9.085 m	23.987 m
4	29–2184 matches	9.184 m	23.513 m
5	30–2180 matches	9.184 m	23.333 m
6	31–2177 matches	9.184 m	23.336 m
7	32–2191 matches	9.184 m	23.338 m
8	32–2148 matches	9.184 m	23.161 m
9	33–2270 matches	9.184 m	23.164 m
10	34–2202 matches	9.184 m	23.166 m
11	35–2176 matches	9.184 m	23.167 m
12	36–2186 matches	9.184 m	23.168 m
13	38–2202 matches	9.184 m	23.341 m
14	38–2205 matches	9.184 m	23.168 m
15	39–2193 matches	9.184 m	23.168 m

Table 6. *Cont.*

Frame Video 1	Frame Video 2	GPS Distance	Computed Distance
16	41–2182 matches	9.184 m	23.336 m
17	41–2166 matches	9.184 m	23.165 m
18	43–2210 matches	9.184 m	23.329 m
19	44–2127 matches	9.184 m	23.325 m
20	44–2144 matches	9.184 m	23.156 m
21	45–2030 matches	9.184 m	23.152 m
22	47–2081 matches	9.184 m	23.310 m
23	48–1912 matches	9.184 m	23.304 m
24	49–1964 matches	9.184 m	23.297 m
25	49–2036 matches	9.184 m	23.132 m
26	50–1946 matches	9.184 m	23.126 m
27	51–2014 matches	9.184 m	23.119 m
28	51–2036 matches	9.184 m	22.955 m
29	54–2025 matches	9.184 m	23.256 m
30	54–2161 matches	9.184 m	23.094 m

5.5. Accuracy of the Computed Distance in a Real-World Scenario

The last test that was conducted aims to ascertain the accuracy of the distance between two vehicles computed by the proposed algorithm. For this, we use a simple but very effective real-world testing scenario. This scenario allows us to measure the exact distance between two vehicles and compare it with the computed distance using the presented approach.

For this test, we used two vehicles, each equipped with a video camera. The street where we recorded the videos was a one lane street. First, we recorded the videos used by our positioning algorithm. Next, for the same frames we have computed the distance, we positioned the cars in the exact same location and measured the exact distance using a measuring tape. During this test, the cars were traveling with speeds between 17.8 and 20.3 km/h.

We did this two times, the only difference being that we switched the car order. The frames detected were in the same area for both cases. The results are presented in Table 7 and in Table 8. In these tables, we included the frame from the first video (from the car in front), the detected frame in the video from the car behind, the computed distance relying solely on the GPS coordinates, the distance computed by the proposed algorithm, and the real measured distance.

Table 7. First Test—Comparison between the measured distance and the computed distance for 3 frames (Car A in Front, Car B Behind).

Frame Video 1	Frame Video 2	GPS Distance	Computed Distance	Measured Distance	Difference	Percent
1	33–2267 matches	34.153 m	20.694 m	21.5 m	−0.806 m	3.748%
16	50–2089 matches	33.463 m	20.993 m	21.45 m	−0.457 m	2.130%
25	60–2065 matches	33.463 m	21.128 m	20.38 m	0.748 m	3.670%

The data presented in these two tables confirm the hypothesis that the distance computed only using the GPS coordinates presents a significant error compared to the real distance and should not be used in car platooning applications. Also, the data indicate that the distance computed using the proposed algorithm outperforms the GPS distance by a great margin, with small differences compared to the real distance. All the differences between the distances computed using the proposed algorithm and the real distances were under 1 m, compared to around 10 m using the GPS distance.

Table 8. Second Test—Comparison between the measured distance and the computed distance for 3 frames (Reversed Order: Car B in Front, Car A Behind).

Frame Video 1	Frame Video 2	GPS Distance	Computed Distance	Measured Distance	Difference	Percent
1	25–2255 matches	9.085 m	24.025 m	23.95 m	0.075 m	0.313%
12	36–2186 matches	9.184 m	23.168 m	23.98 m	−0.812 m	3.386%
16	41–2182 matches	9.184 m	23.336 m	23.9 m	−0.564 m	2.359%

5.6. Limitations

In the conducted tests, we observed that for certain areas captured in the images, such as in Figure 12, the proposed algorithm detected very few matches between frames from the two cameras. This compromised the optimal functioning of the corresponding frame detection algorithm. As depicted in Figure 13, for frame 11, which should ideally have the highest number of matches, they amounted to only around 150. Due to the low number of matches, the algorithm fails to accurately identify the correct frame.

Figure 12. Area of sparse matches detected by the algorithm.

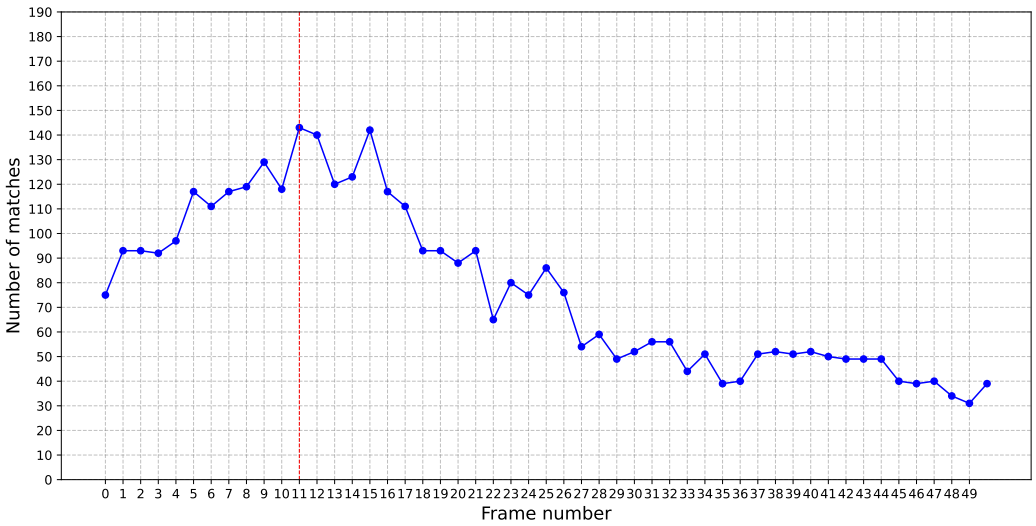


Figure 13. Correct frame detection.

One of the reasons for this issue could be lower brightness in these areas, where descriptors may struggle to extract and match significant features between images, resulting in a reduced number of matches. Nevertheless, such cases can be labeled as failed detections and not to be used in further vehicle platooning applications.

6. Conclusions

Increasing urbanization and vehicle density have led to escalating traffic congestion and a rise in road accidents. With millions of lives lost or injured annually, urgent measures are required to enhance road safety. This underscores the necessity for effective vehicle positioning algorithms to mitigate these challenges.

A robust vehicle positioning algorithm is crucial for effective traffic management and enhanced road safety. With the rising number of vehicles, there is an increased need to optimize traffic flow, minimize delays, and enable intelligent control systems. By accurately determining the position of vehicles, advanced functionalities can be developed to reduce the risk of accidents, improve commuting experiences, and facilitate efficient resource allocation. Implementing such an algorithm is vital for creating a safer and more efficient transportation system.

In this paper, an algorithm that accurately and robustly position vehicles on a road with respect to the position of other nearby vehicles was described. The algorithm presents a decentralized approach where each vehicle acts like an independent computational node and tries to position itself depending on data received from the other nearby vehicles.

The decentralized approach proposed in this paper can use a low-range communication system with a very high bandwidth, but each vehicle requires a high computational power to perform all the processing tasks in real time. A centralized approach (using cloud services, for example) can perform all the processing tasks in real-time, but it highly depends on communication between vehicles and the server, mainly because each vehicle will send the entire video stream to the server.

Based on the results obtained for the various performed tests, it was proven that the novel approach proposed in this paper is efficient and can be used to increase the accuracy of the computed distance between vehicles.

For the first future research direction, the goal is to detect whether vehicles are in the same lane or in different lanes based on the relative position of the two matched descriptors. Another research direction involves implementing a centralized approach, where each vehicle sends data to a server that utilizes cloud computing to process all the data in real-time. This way, each vehicle will have a clearer understanding of vehicles that are not within the considered distance threshold. Furthermore, we plan to expand the experiments and conduct them at higher speeds once we find a suitable road that allows for this, aiming to ensure minimal interference and achieve more accurate results.

Author Contributions: Conceptualization, I.-A.B. and P.-C.H.; methodology, I.-A.B. and P.-C.H.; software, I.-A.B.; validation, I.-A.B., P.-C.H. and C.-F.C.; formal analysis, I.-A.B. and P.-C.H.; investigation, I.-A.B. and P.-C.H.; resources, I.-A.B. and P.-C.H.; data curation, I.-A.B.; writing—original draft preparation, I.-A.B.; writing—review and editing, I.-A.B., P.-C.H. and C.-F.C.; visualization, I.-A.B.; supervision, P.-C.H. and C.-F.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GPS	Global Positioning System
LiDAR	Light Detection and Ranging
V2V	Vehicle-to-vehicle
V2X	Vehicle-to-everything

ADAS	Advanced Driver Assistance Systems
SIFT	Scale-Invariant Feature Transform
DoG	Difference of Gaussian
LoG	Laplacian of Gaussian
SURF	Speeded-Up Robust Feature
ORB	Oriented FAST and Rotated BRIE
BRIEF	Binary Robust Independent Elementary Features
rBRIEF	Rotation-aware BRIEF
BELID	Boosted Efficient Local Image Descriptor
BEBLID	Boosted efficient binary local image descriptor
CNN	Convolutional Neural Networks
BFMatcher	Brute-force Matcher
FLANN	Fast Library for Approximate Nearest Neighbors
OCR	Optical Character Recognition

References

1. World Health Organization. *Save Lives: A Road Safety Technical Package*; World Health Organization: Geneva, Switzerland , 2017; p. 60.
2. World Health Organization. *Global Status Report on Road Safety 2023*; World Health Organization: Geneva, Switzerland, 2023.
3. Forum, I.T. *Monitoring Progress in Urban Road Safety*; International Traffic Forum: Paris, France, 2018.
4. Caruntu, C.F.; Ferariu, L.; Pascal, C.; Cleju, N.; Comsa, C.R. Connected cooperative control for multiple-lane automated vehicle flocking on highway scenarios. In Proceedings of the 23rd International Conference on System Theory, Control and Computing, Sinaia, Romania, 9–11 October 2019; pp. 791–796. [CrossRef]
5. Sun, Y.; Song, J.; Li, Y.; Li, Y.; Li, S.; Duan, Z. IVP-YOLOv5: An intelligent vehicle-pedestrian detection method based on YOLOv5s. *Connect. Sci.* **2023**, *35*, 2168254. [CrossRef]
6. Ćorović, A.; Ilić, V.; Đurić, S.; Marijan, M.; Pavković, B. The Real-Time Detection of Traffic Participants Using YOLO Algorithm. In Proceedings of the 2018 26th Telecommunications Forum (TELFOR), Belgrade, Serbia, 20–21 November 2018; pp. 1–4. [CrossRef]
7. Joshi, R.; Rao, D. AlexDarkNet: Hybrid CNN architecture for real-time Traffic monitoring with unprecedented reliability. *Neural Comput. Appl.* **2024**, *36*, 1–9. [CrossRef]
8. Jia, D.; Lu, K.; Wang, J.; Zhang, X.; Shen, X. A Survey on Platoon-Based Vehicular Cyber-Physical Systems. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 263–284. [CrossRef]
9. Axelsson, J. Safety in Vehicle Platooning: A Systematic Literature Review. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1033–1045. [CrossRef]
10. Yang, H.; Hong, J.; Wei, L.; Gong, X.; Xu, X. Collaborative Accurate Vehicle Positioning Based on Global Navigation Satellite System and Vehicle Network Communication. *Electronics* **2022**, *11*, 3247. [CrossRef]
11. Kolat, M.; Bécsi, T. Multi-Agent Reinforcement Learning for Highway Platooning. *Electronics* **2023**, *12*, 4963. [CrossRef]
12. Gao, C.; Wang, J.; Lu, X.; Chen, X. Urban Traffic Congestion State Recognition Supporting Algorithm Research on Vehicle Wireless Positioning in Vehicle–Road Cooperative Environment. *Appl. Sci.* **2022**, *12*, 770. [CrossRef]
13. Lee, G.; Chong, N. *Recent Advances in Multi Robot Systems*; Chapter Flocking Controls for Swarms of Mobile Robots Inspired by Fish Schools; InTechOpen: London, UK, 2008; pp. 53–68. [CrossRef]
14. Reynolds, C.W. Flocks, Herds and Schools: A Distributed Behavioral Model. *SIGGRAPH Comput. Graph.* **1987**, *21*, 25–34. [CrossRef]
15. Tan, Y.; Yang, Z. Research Advance in Swarm Robotics. *Def. Technol.* **2013**, *9*, 18–39. [CrossRef]
16. Kennedy, J.; Eberhart, R.C.; Shi, Y. Swarm Intelligence. In *The Morgan Kaufmann Series in Artificial Intelligence*; Morgan Kaufmann: San Francisco, CA, USA, 2001. [CrossRef]
17. Mandal, V.; Mussah, A.R.; Jin, P.; Adu-Gyamfi, Y. Artificial Intelligence-Enabled Traffic Monitoring System. *Sustainability* **2020**, *12*, 9177. [CrossRef]
18. Sultan, F.; Khan, K.; Shah, Y.A.; Shahzad, M.; Khan, U.; Mahmood, Z. Towards Automatic License Plate Recognition in Challenging Conditions. *Appl. Sci.* **2023**, *13*, 3956. [CrossRef]
19. Rafique, S.; Gul, S.; Jan, K.; Khan, G.M. Optimized real-time parking management framework using deep learning. *Expert Syst. Appl.* **2023**, *220*, 119686. [CrossRef]
20. Tang, X.; Zhang, Z.; Qin, Y. On-Road Object Detection and Tracking Based on Radar and Vision Fusion: A Review. *IEEE Intell. Transp. Syst. Mag.* **2022**, *14*, 103–128. [CrossRef]
21. Umair Arif, M.; Farooq, M.U.; Raza, R.H.; Lodhi, Z.U.A.; Hashmi, M.A.R. A Comprehensive Review of Vehicle Detection Techniques Under Varying Moving Cast Shadow Conditions Using Computer Vision and Deep Learning. *IEEE Access* **2022**, *10*, 104863–104886. [CrossRef]
22. Kalyan, S.S.; Pratyusha, V.; Nishitha, N.; Ramesh, T.K. Vehicle Detection Using Image Processing. In Proceedings of the IEEE International Conference for Innovation in Technology, Bangluru, India, 6–8 November 2020; pp. 1–5.

23. Zhang, Y.; Carballo, A.; Yang, H.; Takeda, K. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *J. Photogramm. Remote Sens.* **2023**, *196*, 146–177. [CrossRef]
24. Lu, S.; Shi, W. Vehicle Computing: Vision and challenges. *J. Inf. Intell.* **2023**, *1*, 23–35. [CrossRef]
25. Lowe, D. Object recognition from local scale-invariant features. In Proceedings of the 7th IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157. [CrossRef]
26. Vaithiyanathan, D.; Manigandan, M. Real-time-based Object Recognition using SIFT algorithm. In Proceedings of the 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 5–7 April 2023; pp. 1–5. [CrossRef]
27. Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. Speeded-Up Robust Features SURF. *Comput. Vis. Image Underst.* **2008**, *110*, 346–359. [CrossRef]
28. Sreeja, G.; Saraniya, O. Chapter 3—Image Fusion Through Deep Convolutional Neural Network. In *Deep Learning and Parallel Computing Environment for Bioengineering Systems*; Sangaiah, A.K., Ed.; Academic Press: Cambridge, MA, USA, 2019; pp. 37–52. [CrossRef]
29. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G.R. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
30. Rosten, E.; Drummond, T. Machine Learning for High-Speed Corner Detection. In *Computer Vision—ECCV*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 430–443.
31. Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P. BRIEF: Binary Robust Independent Elementary Features. In *Computer Vision—ECCV*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 778–792.
32. Wu, S.; Fan, Y.; Zheng, S.; Yang, H. Object tracking based on ORB and temporal-spacial constraint. In Proceedings of the IEEE 5th International Conference on Advanced Computational Intelligence, Nanjing, China, 18–20 October 2012; pp. 597–600. [CrossRef]
33. Rosin, P.L. Measuring Corner Properties. *Comput. Vis. Image Underst.* **1999**, *73*, 291–307. [CrossRef]
34. Suárez, I.; Sfeir, G.; Buenaposada, J.M.; Baumela, L. BEBLID: Boosted efficient binary local image descriptor. *Pattern Recognit. Lett.* **2020**, *133*, 366–372. [CrossRef]
35. Suarez, I.; Sfeir, G.; Buenaposada, J.; Baumela, L. BELID: Boosted Efficient Local Image Descriptor. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2019; pp. 449–460. [CrossRef]
36. Tian, Y.; Fan, B.; Wu, F. L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6128–6136. [CrossRef]
37. Zhang, H.C.; Zhou, H. GPS positioning error analysis and outlier elimination method in forestry. *Trans. Chin. Soc. Agric. Mach.* **2010**, *41*, 143–147. [CrossRef]
38. van Diggelen, F.; Enge, P.K. The World's first GPS MOOC and Worldwide Laboratory using Smartphones. In Proceedings of the 28th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2015), Tampa, FL, USA, 14–18 September 2015.
39. OpenCV Modules. Available online: https://docs.opencv.org/4.9.0/ (accessed on 1 May 2024).
40. Muja, M.; Lowe, D. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *VISAPP* **2009**, *1*, 331–340.
41. Tesseract OCR. Available online: https://github.com/tesseract-ocr (accessed on 1 May 2024).