



Automotive Perception System Evaluation with Reference Data from a UAV's Camera Using ArUco Markers and DCNN

Krzysztof Blachut¹ · Michał Danilowicz¹ · Hubert Szolc¹ · Mateusz Wasala¹ · Tomasz Kryjak¹ ·
Mateusz Komorkiewicz²

Received: 26 April 2021 / Revised: 7 December 2021 / Accepted: 13 December 2021 / Published online: 31 January 2022
© The Author(s) 2021

Abstract

Testing and evaluation of an automotive perception system is a complicated task which requires special equipment and infrastructure. To compute key performance indicators and compare the results with real-world situation, some additional sensors and manual data labelling are often required. In this article, we propose a different approach, which is based on a UAV equipped with a 4K camera flying above a test track. Two computer vision methods are used to precisely determine the positions of the objects around the car – one based on ArUco markers and the other on a DCNN (we provide the algorithms used on GitHub). The detections are then correlated with the perception system readings. For the static and dynamic experiments, the differences between various systems are mostly below 0.5 m. The results of the experiments performed indicate that this approach could be an interesting alternative to existing evaluation solutions.

Keywords UAV · Drone · LiDAR · Automotive · ADAS · Perception systems · Testing · Evaluation · Automatic labelling · ArUco markers · DCNN

The work presented in this paper was supported by the AGH University of Science and Technology project no. 16.16.120.773. We would like to especially thank Nikodem Pankiewicz and Nikodem Janik for help with the host vehicle and test track trials during the initial experiments.

✉ Krzysztof Blachut
kblachut@agh.edu.pl

Michał Danilowicz
danilowi@agh.edu.pl

Hubert Szolc
szolc@agh.edu.pl

Mateusz Wasala
wasala@agh.edu.pl

Tomasz Kryjak
tomasz.kryjak@agh.edu.pl

Mateusz Komorkiewicz
mateusz.komorkiewicz@aptiv.com

¹ Embedded Vision Systems Group, Computer Vision Laboratory, Department of Automatic Control and Robotics, AGH University of Science and Technology, Al. Mickiewicza 30, Krakow 30-059, Poland

² Aptiv, Krakow, Poland

1 Introduction

The problem of accurate information about the objects around the host vehicle is essential for the verification/testing of the automotive perception systems (cameras/radars/LiDARs) performance as well as for creating a dataset that can be used for the training of machine learning (ML) algorithms. In production projects, OEMs (Original Equipment Manufacturers) require a testing phase (e.g. performed on a test track) and the computation of key performance indicators (KPIs) for larger datasets. The testing phase usually comprises of different tests with the aim of additional data acquisition, algorithm verification, KPIs calculation or conducting Euro NCAP safety tests. It is worth mentioning that even in very similar tests, different manufacturers may have different requirements in terms of accuracy etc. Among the mentioned KPIs some may include binary information (e.g. if a collision happened or not), a metric value (e.g. a distance to another object in centimetres) or a numeric value (e.g. a number of correct object detections).

Generally, to verify the system performance in a quantitative form, not only the sensor output must be recorded but the reference must also be available (e.g. the position of the objects “visible” to the perception system). So far, the

problem mentioned is mostly solved by adding additional precise sensors to the vehicle tested such as LiDAR (Light Detection and Ranging) or GPS (Global Positioning System) etc., recording data, and manually labelling it. There are some solutions which use multiple, synchronised, precise, and expensive GPS receivers placed in a few vehicles which can provide the required information without labelling, but the number of objects monitored is very limited [1].

There are several problems with the solutions described above:

- Manual labelling requires a lot of man power, is a time-consuming process, and generates high costs and project risk.
- When additional sensors are used as a reference, they are mounted on the host vehicle and in almost all cases have fields of view and perspective the same as the sensors under test. This might lead to errors in labelling due to occlusions (e.g. a pedestrian partially covered by a lamp pole).
- Using multiple GPS receivers and tracking objects (e.g. during the testing phase) is expensive and requires significant vehicle modifications. Additionally, an accurate GPS signal is not available in all locations.
- Integrated reference systems are costly.
- What is most important is the relative position of the objects regarding the vehicle under test instead of the precise global position in the world. Thus, using expensive GPS systems seems to be unjustified in the considered case.

In this work, we propose a different approach to the issue of evaluation of perception systems used in advanced automatic driver assistance systems (ADAS) and autonomous vehicles (AV). It assumes the use of video sequences captured with a high-quality camera mounted on a UAV hovering above the host vehicle as reference data. On their basis, computer vision methods, including deep convolutional neural networks, are used to detect and locate objects in the surroundings of the vehicle.

This article is an extension of the conference paper [2] presented at DASIP'21 workshop in January 2021. We conducted further experiments for vehicle perception system evaluation with a drone's camera. We managed to upgrade the vision algorithms, eliminating some problems from the previous work. Some additional tests were also carried out to investigate the effect of multiple factors on detection accuracy with vision-based methods.

The main contributions of this paper:

- The proposal of a methodology for automotive perception systems evaluation with the use of sequences recorded by a UAV,

- The analysis of multiple factors which have an impact on accuracy of vision methods (ArUco markers and DCNN),
- The description of a static and dynamic experiment, the results of which are mostly below 0.5 m between different sensors, thus confirming the usefulness of the proposed solution.

We provide the source code of our application [3].

The rest of the paper is organised as follows. In Section 2 previous works on the vehicle perception systems evaluation are presented. Sections 3 and 4 describe our solution to the aforementioned problem: the equipment and the vision algorithms used in the project. Section 5 presents the tests carried out and the results obtained. Section 6 summarises the article with the conclusions.

2 Previous Work

The evaluation and testing of perception systems used in modern cars has received great attention in commercial solutions. As already mentioned in the introduction, they are based on different technologies while offering different degrees of accuracy and automation.

The RT-Range [4] solution is a coupled inertial (INS) and satellite (GNSS) navigation system. It requires expensive additional components to be installed inside the vehicle. These modules communicate with each other wirelessly, which enables simultaneous ground truth acquisition and the processing of relative position measurements. According to the manufacturer's information, the accuracy is 0.02 m for the distance in case of an RTK (Real-Time Kinematic) system and 0.4 m in case of a standard DGPS (Differential GPS) system, 0.2 km/h for the speed, and 0.1° for the rotation angle around the vertical axis.

A competitive solution is levelXdata [5]. It is a comprehensive system for processing 4K road traffic images recorded by a UAV. It enables the detection of various objects (mainly different vehicles, but also pedestrians and cyclists) and the determination of their trajectories using deep neural networks. The system discussed can operate in several modes. In one of them, the UAV can track the selected car, collecting information about the surrounding objects. The images captured by the drone are then processed off-line (i.e. on the ground). On their basis, the reference data can be prepared for the verification of the vehicle perception systems.

The main advantages and motivations of levelXdata are collected in [6], where the authors present its superiority over different existing solutions for automotive data recording and provide references to their earlier publications. Several details about the solution can be found in [7]. In this work the authors focused on a database creation with

the UAV, recording the road intersections from an altitude mainly between 60 and 100 m.

For the altitude of 100 m, 1 pixel corresponded to around 4×4 cm and the camera's field of view was around 140×70 m. Only in case of highway recordings the drone was hovering much higher to make it possible to monitor a longer part of the road (precisely 420 m). The authors applied video stabilisation to remove occurring movements of the drone and used expensive smoothing of the object trajectories over entire sequences separately. In terms of accuracy of the algorithm, the DCNN used by the authors achieved a positioning error of the objects mostly in the range of 10 cm compared with manual labels.

Despite these advantages, the solution described does not address the problem of the synchronisation between the camera and the sensors tested, which is a crucial element in the evaluation. Furthermore, the aim of the authors was to prepare a reliable dataset for testing purposes, so the proposed sequence-based smoothing that they used definitely improved the accuracy, but is rather not suitable for real-time operation. The authors also state that the accurate measurements require high video resolution, strong algorithms and high processing power, which is unavailable for online processing on the UAV.

In the scientific literature, most of the research in the vehicle perception field is focused on data processing and fusion – mainly on object detection and tracking. In this case, the well-known off-line comparison with the ground truth data (often manually labelled) is used. There is quite a significant shortage of articles devoted to other approaches to the evaluation of perception systems. In particular, we are not aware of solutions, which can work in real-time and process data in an on-line manner.

In [8] the authors have proposed the concept of using an artificial intelligence (AI) throughout the testing process. At first, the AI would be used to generate appropriate simulation scenarios based on a real-world data. Then it would also verify the results obtained. On this basis, it could prepare some additional scenarios, if necessary. As a result, the entire testing and validation process would be significantly automated and the human role would be limited only to the supervision of the results.

In the work [9] the authors proposed an innovative system for HIL (Hardware-in-the-Loop) testing. Its purpose is to enable the verification of all perception modules as a whole instead of separate components. According to the authors, this approach allows, to some extent, to replace costly road tests. At the same time, however, the proposed test bench lacks the ability to verify radar detections. This functionality is yet to be developed and will be added in the future.

To the authors' best knowledge, the use of UAVs for an on-line perception systems evaluation has not been considered in the scientific literature so far. The most similar

approach was presented by the levelXdata authors, but they focused rather on a dataset creation than a real-time (on-line) evaluation of perception systems – therefore, it is unknown whether their solution can be used in real-time processing tasks. However, other “automotive problems” could be addressed with the use of UAVs. One of the examples is a road traffic monitoring. In the work [10] the authors present a survey of systems dedicated to this subject. They also propose their own solution, using the images recorded by cameras mounted on a UAV. The algorithm designed makes it possible to detect the individual cars, determine their speed and possible violations of the traffic rules.

3 The Proposed System

The proposed solution is schematically presented in Fig. 1. The perception systems of the car on the left were evaluated. For the purposes of this test, the host vehicle was equipped with a LiDAR (exemplar perception system) and an accurate differential GPS – both integrated within the ROS system (Robot Operating System). In addition, the car was equipped with LEDs for data synchronisation with the UAV and an ArUco marker on the roof (see Fig. 2a). The remaining vehicles had ArUco markers placed on their roofs. The scene was recorded by the UAV that was essentially above the host vehicle.

At the current stage of the project, we have focused mainly on vehicle detection. We did some initial tests for pedestrian detection, however, the detection of other objects is planned as future work. We have assumed that two vision methods of measuring the vehicle position will be compared: based on the ArUco markers and object segmentation with a deep convolutional neural network (DCNN).

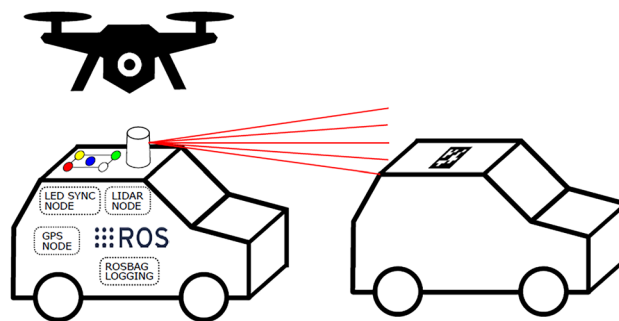


Fig. 1 Scheme of the proposed solution [2]. The host vehicle, presented on the left, is equipped with a LiDAR (exemplar perception system), differential GPS – both integrated within the ROS, and LEDs for data synchronisation with the UAV. ArUco markers are placed on the host and other vehicles.

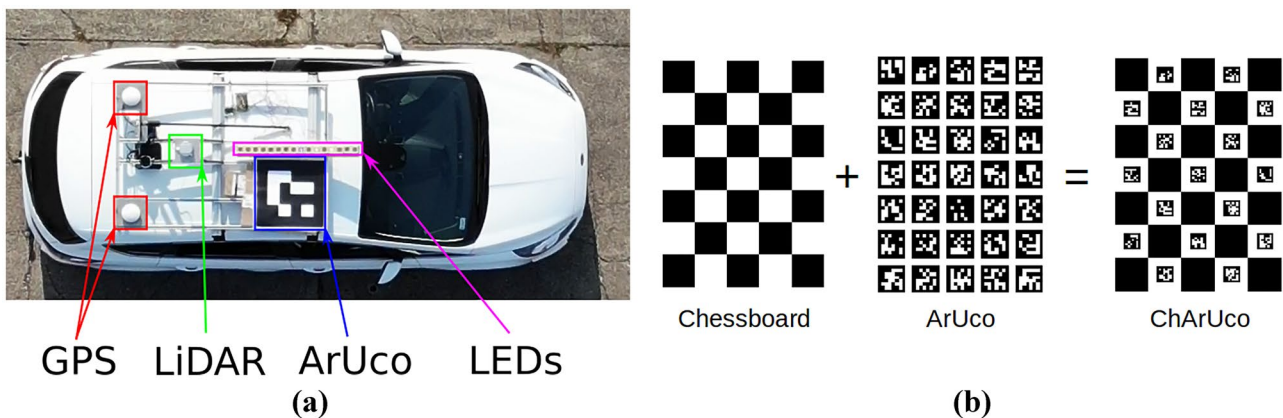


Fig. 2 (a) The host vehicle and its sensors [2], (b) ChArUco calibration board [11].

3.1 Vehicle Setup

An instrumented vehicle was used to perform sensor data recording. An Hesai Pandar 40p LIDAR (10 Hz rotation) together with an ArUco marker and LEDs were mounted on the roof rack (c.f. Fig. 2a). The vehicle was also equipped with a precise differential GPS system (also 10 Hz, accuracy of 3 cm) with RTK corrections from SwiftNavigation, which we used in our initial experiments. All sensors were connected to a logging PC with ROS used for data acquisition.

3.2 UAV Setup

A DJI Mavic Air UAV was used to record the test sequences. It is equipped with an on-board 4K (Ultra High Definition – 3840×2160 pixels, 30 fps – frames per second) camera, which is mounted on a 3-axis gimbal stabiliser. The camera also supports lower resolutions, e.g. Full HD (1920×1080 pixels) in slow motion format, i.e. 120 fps. It has a 12 megapixel CMOS sensor and an $f/2.8$ lens with an equivalent focal length of 24 mm and an 85° field of view. The recorded video is saved in the UAV's internal memory or on an SD card, as well as on a smartphone as a lower resolution backup. Additionally, it is possible to preview the image on the smartphone in real-time, which was useful during the experiments. With a single battery, a flight of around 20 minutes is possible.

According to the manufacturer, the maximum speed of the drone is 68.4 km/h, which is enough for the majority of tests, among which some would only require hovering above the tested vehicles. Only some special cases require higher velocities, e.g. testing an ACC (Adaptive Cruise Control) on a highway. However, despite the above-mentioned advantages, we assume the use of a custom-made drone in the target system. This issue is discussed in more detail in the summary section.

3.3 Vehicle-UAV Synchronisation Concept

If the data from different sensors (i.e. LiDAR, GPS and cameras) is used during the vehicle perception system evaluation, the synchronisation of these sensors is necessary. Given the available hardware setup, we have chosen to use a vision-based approach. We have prepared and constructed a controllable LED system shown in Fig. 2a.

The hardware part of the LED system consists of a 1 m long rail with sixteen 6×6 cm boxes. Every second box contains a high-brightness LED of a selected colour (red, green, blue, or white). There are empty boxes between the LEDs, which are used as a reference to cope with different lighting conditions. Everything is connected to a Raspberry Pi 3B+ mini-computer, which controls the LEDs. To synchronise consecutive image frames with the LiDAR point clouds, an 8-bit binary number is displayed on the LEDs, which is then recorded by the camera on the UAV.

The source of the synchronisation signal is a custom ROS node running on a PC computer. It subscribes to LiDAR point cloud messages and with every new LiDAR message an internal counter is incremented and sent over the USB2RS232 conversion board to the RPi3 with a protocol ensuring proper error correction. This value is then instantly displayed on the LEDs.

The above-described LED-based system was a direct consequence of the use of the off-the-shelf UAV. As the evaluated perception system (LiDAR) works at 10 Hz, a very precise timestamp synchronisation with 0.01 or 0.001 s accuracy was not needed. However, a much simpler and more reliable approach (less prone to all typical issues related to vision systems) would be a wireless communication channel between the vehicle (ROS) and the UAV (preferably a custom-built one). Ultimately, a timestamp from each of the evaluated sensors (LiDAR, radar, camera, and GPS) should be attached to the recorded video sequence.

4 The Video Processing Methods Used

In the target solution, the reference information about the location of the objects in the vicinity of the host vehicle would be obtained in real-time from a camera mounted on a UAV to evaluate its on-board sensors. This would allow to conduct different tests using various sensors with calculating appropriate KPIs and then to verify these results with the ones obtained from the UAV's camera and vision-based algorithms. Exemplary KPIs that could be verified this way include time to collision with an object, driving lane departure, velocity of the vehicle, number of stops in a particular section of the road or vehicle throughput at an intersection.

However, the primary aim of this work was to compare the measurements from the UAV's camera with the ground truth to assess the suitability of the proposed solution. Two approaches described below were used and evaluated – the ArUco markers [12] and a DCNN segmentation. The secondary objective was the initial evaluation of one of the vehicle's sensors – precisely the LiDAR that was mounted on our host car. For this task we decided to make a comparison between its measurements and the results obtained from vision-based methods.

It is worth adding here that the entire camera-based system is designed for good weather conditions, which generally means the weather without rain, snow, fog, or strong wind. Typical UAVs are not suitable for flying in such bad weather, because water or high humidity can cause the electronics to fail, which results in erroneous behaviour or loss of functionalities (only specialised UAVs are adapted to such conditions). Fog can reduce visibility or result in condensation on the camera lens. Rain or strong wind may cause the loss of control or communication and eventually result in severe damage of a UAV. Therefore, inadequate weather conditions may cause a threat to the UAV's equipment and the participants, or at least result in reduced accuracy of the experiment. On the other hand, using a specialised UAV, robust to weather conditions, could be an interesting idea for future experiments mainly due to the requirement of perception system evaluation in different, even unfavourable conditions.

4.1 Camera Calibration

For the correct operation of a vision system, especially when it involves measurements, it is necessary to calibrate the camera used. In our experiments, for this purpose, we have used a so-called ChArUco board available in the OpenCV library [11]. It is created by combining a typical chessboard widely used for image calibration with the ArUco markers placed on its white fields (c.f. Fig. 2b). In comparison to

the standard solution (i.e. traditional chessboard), it allows to increase the calibration precision. The detection of the corners is carried out for ArUco markers first and these results are used for interpolation of the chessboard corners. Such an approach results in slightly more accurate corner positions of the entire ChArUco board than in the case of a traditional chessboard without additional markers. Thus, as the correct calibration is crucial for determining real distances based on the camera, the ChArUco-based method is a better choice than the traditional chessboard.

To perform the calibration, the functions from the OpenCV library were used – the main one was *calibrateCameraCharucoExtended*. As a result, the distortion coefficients and the camera matrix of our optical system were obtained. Thanks to this, it was possible to remove the distortion introduced by the camera and more precisely determine the distances between objects visible on the scene (closer to real-world measurements). It should be noted that the calibration is necessary for both vision-based detection methods – ArUco and DCNN.

4.2 ArUco Detection

The ArUco markers [12, 13], well known in the image processing community, were used to detect the vehicles in the camera image. A typical ArUco marker is a big black square with multiple smaller white and black squares inside it. These small squares define the code of the marker (c.f. Fig. 2a). The detection procedure works in the same way for all markers, while the possibility of coding a unique identifier allows the use of a large number of easily distinguishable codes. Therefore, these markers are commonly used to detect various objects. Moreover, they are not very susceptible to interference (e.g. changes in lighting).

In this project, 4 markers of 55×55 cm with 4×4 small squares inside were used. They were printed on a magnetic foil and placed on the vehicles' roofs to enable their detection and determination of mutual distances. In addition, the information about the actual dimensions of the marker allowed us to convert the distances expressed in pixels to metres (i.e. from the image domain to the real-world measurements).

4.2.1 Marker Detection

Firstly, the distortion introduced by the camera optics was removed. Next, the gamma correction with gamma value set to 2 was applied to the image, as its usage increased the accuracy of the detections. Then, the *detectMarkers* function from the OpenCV library was used. The corners positions determined with precision of 5 decimal places and IDs of the detected markers were obtained, as well as a list of detections considered incorrect.

In lighting conditions, in which the sequences were recorded (quite harsh sun), not all markers were correctly detected. However, a modification of the default parameters of the mentioned OpenCV function allowed us to mostly eliminate this problem. We carried out some further tests in different weather conditions (harsh sun, cloudy). A different method of detecting marker corners was tested – *APRILTAG* instead of *SUBPIX*, which we used earlier [2]. This modification, as well as tuning some other detector parameters, significantly accelerated the calculations and improved the results. Some further speedups would be possible with different sets of parameters but at a cost of a decreased detection accuracy. For more details about modified ArUco detection parameters, please refer to the provided source code [3].

Different lighting (i.e. very cloudy, partly cloudy, sunny) is generally not a big problem in case of the ArUco markers as they are very robust to illumination changes. Therefore, even in more demanding situations, the additional tuning of the parameters should be sufficient for correct detections. However, the determination of the universal set of parameters for different lighting conditions is quite complicated – here some kind of automatic calibration would be a good idea for the future.

Then, for the detected markers, the *estimatePoseSingleMarkers* function was used to determine their orientation and position in the coordinate system associated with the camera. These positions were expressed in metres in relation to the centre of the image with the precision of 8 decimal places. In addition, the distance of each marker from the camera was determined. Then, the centre of each marker and its size in pixels were computed – also with a high precision to take advantage of results returned from OpenCV functions. For the marker size calculation, the average length of its 4 sides was used.

The next step of the algorithm was to determine a bounding box for each of the vehicles. As they had various dimensions, slightly different scaling factors were used to calculate the length and width of their bounding boxes with respect to the detected marker. This was done due to the data format usually returned by the vehicle perception systems – bounding boxes or cuboids. Therefore, the final comparison could be done with well-known approaches like IoU (Intersection over Union).

After the initial tests, one more problem was noted – when the car is visible not directly from above, but from an angle, it is difficult to correctly establish the bounding box. So an additional correction was introduced, which takes into account the angle at which the vehicle is visible in the image. This change allowed us to calculate particular bounding boxes with higher accuracy. But generally this issue is complicated (as it also depends on the altitude of the drone that makes the recording) and it requires further investigation in the future to increase the accuracy of the algorithm.

4.2.2 LEDs Detection

Additionally, for the host car, the LEDs detection had to be performed. For this purpose, their known positions in relation to the centre of the ArUco marker were used. These distances, known in metres, were projected into pixels and the approximate LEDs positions in the image were obtained. More details are included in the provided source code [3]. In this way, it was possible to avoid the problem of the LEDs detection in the whole image.

Then the pixel brightness in the designated locations was verified. At first, the image was converted to grayscale. If the value of the pixel brightness in the calculated position was greater than a preset threshold (its default range is between 0 and 255), the LED was considered ON and otherwise OFF. After the preliminary tests, the threshold value was set as the sum of 190 and the altitude of the drone rounded to metres, which made it possible to correctly distinguish the activated LEDs from the deactivated ones. Such a sum was the result of additional tests in different weather conditions, as it turned out that the big influence on the correct decoding of the LED state has the altitude of the UAV above the host car. It is worth mentioning that in case of different weather conditions the summed threshold may be slightly higher or lower.

An exemplary result of the LED state detection was presented in Fig. 3. To increase the robustness of the LEDs positioning to minor disturbances, the average pixel brightness in its small vicinity (exactly 5×5) was used instead of a single value. As a result, the situations, in which the designated positions of the LEDs were inaccurate, were limited. The information about the state of 8 LEDs was then translated into a binary code and a decimal number, which finally enabled the synchronisation with the LiDAR's data.

However, in case of a high altitude of the drone, the entire LED system is quite small and blurred in the image, which makes it difficult to properly encode its ID for



Fig. 3 Results of a correct LED system state detection. Red circle around the diode means that it is detected as OFF, while the green circle means that diode is detected as ON. The most significant bit of a binary number is on the right side of the image, while the least significant is on its left side. After the conversion to a decimal number, ID = 252 is obtained.

automatic synchronisation. Even though we managed to slightly update the LED system, the wireless communication channel seems to be a more reliable way of synchronisation than the vision-based approach, thus it will be used in the target solution instead of the LEDs.

4.2.3 Distance Calculation

The final step of the video processing was the determination of the distances between individual vehicles using a simple Euclidean norm. As the markers (and vehicles) are visible in different parts of the image, their dimensions in pixels are not identical, although in reality they are of the same size (55×55 cm). To scale the measurements from pixels to metres, two approaches were analysed and compared. For both methods, the maximum precision of calculated positions was used to avoid accumulation of rounding errors.

The first one was based on calculating the distances straight from the positions in metres returned by *estimatePoseSingleMarkers* OpenCV function. These positions were expressed in a 3D coordinate system with the camera as its centre. In case of this method, we noticed bigger variability of the results, as even small inaccuracies in the marker detection procedure had a significant impact on calculated 3D positions.

In the other one, the distances in pixels were calculated on a 2D image plane and then converted to metric units. The average size of the two analysed markers was calculated to obtain a more reliable scaling factor. For example, to determine the distance between markers having ID = 1 and ID = 2, Equation 1 was used. For this method, the distances obtained were much more stable and the simple averaging increased the robustness of this approach even further. As a result of the comparison between both methods, we decided to use this approach.

$$dist_m = \frac{dist_{pix} \cdot ML}{0.5 \cdot (ms_1 + ms_2)} \quad (1)$$

where: $dist_m$ is the distance in metres, $dist_{pix}$ is the distance in pixels, ML is the size (length) of a printed marker in metres, while ms_1 and ms_2 are the sizes of the markers in pixels obtained from the detection.

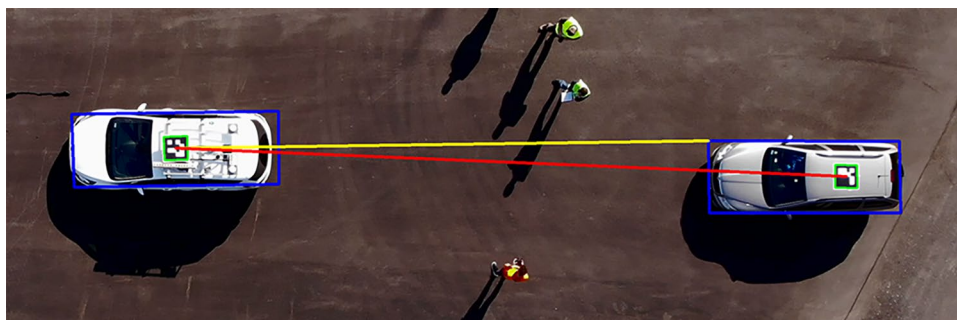
Finally, two distances were calculated between the host car and every other vehicle, just like in Fig. 4. The first one (red line) defined the distance between the centre of the host vehicle's ArUco marker and the corresponding point on each of the other cars. The second one (yellow line) expressed the approximate minimum distance from the same point (i.e. the centre of the ArUco marker on the host vehicle) to the bounding box of a given vehicle (which roughly corresponds to its bumper). For more information, please refer to the source code [3]. These measurements could be later compared with the results obtained with DCNN or LiDAR detections, as well as the ground truth.

4.2.4 Performance Analysis

In the target solution, the algorithm presented above should be implemented and run on an embedded platform. In this work, a software version of the algorithm written in Python 3.7 runs on Intel Core i7-6700K 4.00 GHz CPU and takes around 300 ms to process one 4K frame from the sequence. Reading the frame by using *videoCapture* takes around 20 ms. Frame preprocessing takes around 85 ms – conversions between colour spaces, needed for gamma correction, and into grayscale take 55 ms, LUT-based gamma correction 10 ms, while removal of camera distortion 20 ms. The core function of the algorithm (*detectMarkers*) takes 200 ms on average, but depending on number of detected marker candidates, this time can be around 50 ms shorter or longer. The rest of operations, including pose estimation, distance calculation or LEDs detection, need just a few milliseconds, which is a very small fraction of a total processing time.

The main problem lies in the image resolution – the processing of a 4K image on a CPU is very time-consuming. Of course resizing down the image or using parameters in *detectMarkers* function that do similar down-sampling helps to reduce ArUco detection time by an order of magnitude,

Fig. 4 Exemplary measurements between the host car and other vehicle.



but at a cost of decreased number of detections and their accuracies.

In terms of a hardware implementation it is possible to easily accelerate/parallelise some of the functions on GPU or FPGA. So as a simple test we ran this algorithm on the Nvidia Jetson AGX Xavier embedded GPU and obtained mostly longer processing times. After rewriting the “frame preprocessing” functions to use the available CUDA cores, we were able to reduce their processing times and obtained the following results: *videoCapture* 45 ms, colour conversions 30 ms, distortion removal 10 ms, *detectMarkers* 125 ms. So the speedup for colour conversions and distortion removal is around 2. However, accelerating the ArUco detection function is more difficult and requires more investigation into parallelising particular operations of the detection procedure. The entire algorithm requires on average 250 ms to process one frame, which translates to 4 fps. This value is slightly better than on a CPU, but is definitely not enough for real-time processing (at least 20–25 fps). Therefore, a real-time implementation of the proposed algorithm in 4K resolution poses a challenge, but this simple test shows that there is a room for improvement over the software version.

4.3 Vehicle Detection and Tracking with DCNN

The second of the considered vehicle detection methods was based on a deep convolutional neural network (DCNN). In recent years, some successful DCNN detection frameworks were introduced. One-stage detectors, like SSD (Single Shot Detector) [14] or YOLO (You Only Look Once) [15], contain a single network that is used for classification and localisation during a single forward pass on the input image. These algorithms aim for a high processing speed (exceeding 30 fps), while maintaining competitive detection accuracy among other state-of-the-art methods. YOLO network has already been successfully used for object detection from drone’s perspective [16]. The authors added extra convolutional filters and retrained the net using a dataset with the images from a UAV. On the other hand, detectors using R-CNN architectures (Faster R-CNN [17], Mask R-CNN [18]) use separate networks for feature extraction, proposal generation, and classification. The Mask R-CNN detector based on the convolutional ResNet architecture was used in this work, as it precisely segments the detected objects along their contours, which was crucial in our experiments.

4.3.1 Initial experiments with DCNN

The model trained by the authors of the Mask R-CNN solution on the COCO train2017 set, containing 80 classes of objects, including the car class, was used. Initially, we followed the work of [19], which investigated the effectiveness of the Mask R-CNN detector, learned on the COCO set

[20], on video sequences recorded by high-mounted security cameras. Their results indicated a high detection accuracy on selected sequences (95% *true positives ratio* for the car class), even though the COCO training set contains photos of the cars presented only from the perspective of a standing human or other ground vehicles. Our initial experiments have shown, however, that such a trained detector is not able to effectively recognise the cars or pedestrians from the videos recorded by a UAV (at an altitude of 30 m and above).

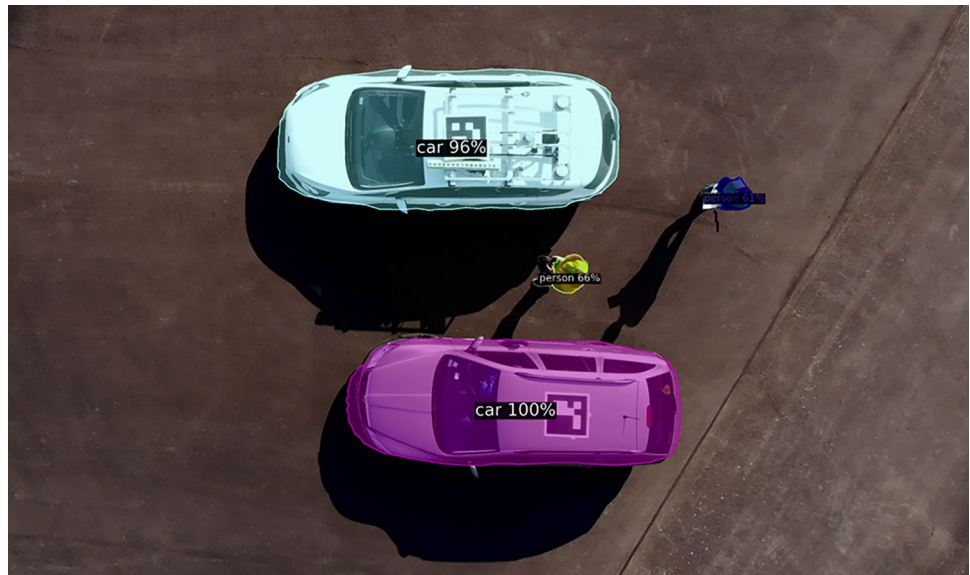
4.3.2 Mask R-CNN Fine-Tuning

In the next step, we have decided to fine-tune a 50-layer ResNet model pretrained on the COCO train2017 dataset. We used the COCO as well as the VisDrone dataset [21] containing over 10000 photos and 260000 video frames captured by drone-mounted cameras, on which vehicles and pedestrians were annotated. We applied the SGD (Stochastic Gradient Descent) algorithm with the learning coefficient 0.02, the momentum 0.9 and a batch size equal to eight images. The ground truth of the VisDrone set contained bounding boxes of the cars, so we only trained the ResNet, RPN (Region Proposal Network) and classifier networks. To train the segmentation sub-network of the Mask R-CNN detector, only the COCO train2017 set was used, from which images containing cars and people were selected.

4.3.3 Results

Sample results of the trained detector operation are shown in Fig. 5. The model is able to generate precise binary masks of vehicles seen from above, even though the training set for segmentation fine-tuning (COCO) contained only segmentation masks of vehicles from the front view. Some of the recorded test sequences were quite demanding due to the lighting conditions. The occurring shadows made it difficult to localise the vehicles and segment them correctly. However, these problems were significantly reduced compared with our previous test [2] by using data augmentation with random changes in brightness, saturation, and contrast of the training images. The training process applied resulted in a significant improvement in the quality of the detection and classification of the cars from the drone’s perspective. The model evaluated with the COCO API yielded bounding box mAP (mean Average Precision) of 0.36 for cars and person classes in our test sequences and 0.23 for car, truck, bus, pedestrian, person classes in the VisDrone dataset. In our test sequences, the detector was able to successfully recognise people at altitudes of up to 40 m with missed detections and false positives starting above that altitude. In terms of the vehicles, the results were much better – they were correctly detected for the entire range of tested altitudes, up to 100 m.

Fig. 5 Vehicle and pedestrian detections obtained with the fine-tuned Mask R-CNN.



Our fine-tuned model of the network used in this project is available in the source code [3]. The use of the VisDrone together with the COCO dataset made it possible to correctly detect vehicles and pedestrians seen from above, while maintaining the ability to detect them from the front view.

4.3.4 Object Tracking

The detector was then used to create a multi-object tracking-by-detection system. The problem of the detection association in subsequent frames was solved by using an additional *fully-connected* neural network (single layer). It returned the object identification vectors on the basis of the features from the ResNet network. The Euclidean metric was used to determine the similarity of the two identification vectors. This method, described in [22], is a popular solution in the latest tracking-by-detection systems. The final detection assignment was done using the Hungarian algorithm and based on the similarity metric. The analysed test sequences, however, were not demanding from the tracking point of view: there were no occlusions, no rotations of objects outside the image plane, or changes in lighting.

4.3.5 Performance Analysis

In terms of performance, we compared the processing times of our modified network on Intel Core i7-6700K 4.00 GHz CPU and Nvidia GTX 1070 GPU. Using the CPU we achieved on average 2400 ms for one frame (around 0.4 fps), while using the GPU we managed to accelerate it to 156 ms, which equals 6.4 fps. Similarly to the ArUco-based method, we ran the network on Nvidia Jetson AGX Xavier using CUDA cores and achieved around 500 ms on average for the processing of one frame, which translates to 2 fps.

5 Experiments & Results

To assess the usefulness of the proposed concept, multiple tests were carried out – some initial ones, the static experiments and the dynamic ones. They were executed on a dedicated test track in conditions similar to a real road situation.

5.1 Initial Tests

Before recording the planned testing scenarios, some initial experiments were conducted. They were motivated by the fact that when analysing the results from our previous tests described in [2], we found a few problems with distance measurements based on the vision methods (ArUco and DCNN).

5.1.1 Altitude Test

The aim of the first test was to investigate the effect of the altitude of the UAV at which the recording is done on ArUco marker detection. For this purpose, we placed two markers on the ground at a distance of 10 m (measured with a tape). Next, the UAV hovered above the markers at an altitude ranging from 10 to 100 m. With the ArUco-based algorithm, the distances between the markers on subsequent frames were calculated. The results of vision-based measurements in relation to the altitude of the drone (and its camera) are presented in Fig. 6a.

On its basis, at least three conclusions can be made. Firstly, the calculated distances exceed 10 m and the error increases with the altitude of the UAV, which corresponds to increasing distance between the markers (on the ground) and the camera. These errors are mostly caused by two reasons – characteristics of the camera used and inaccuracies

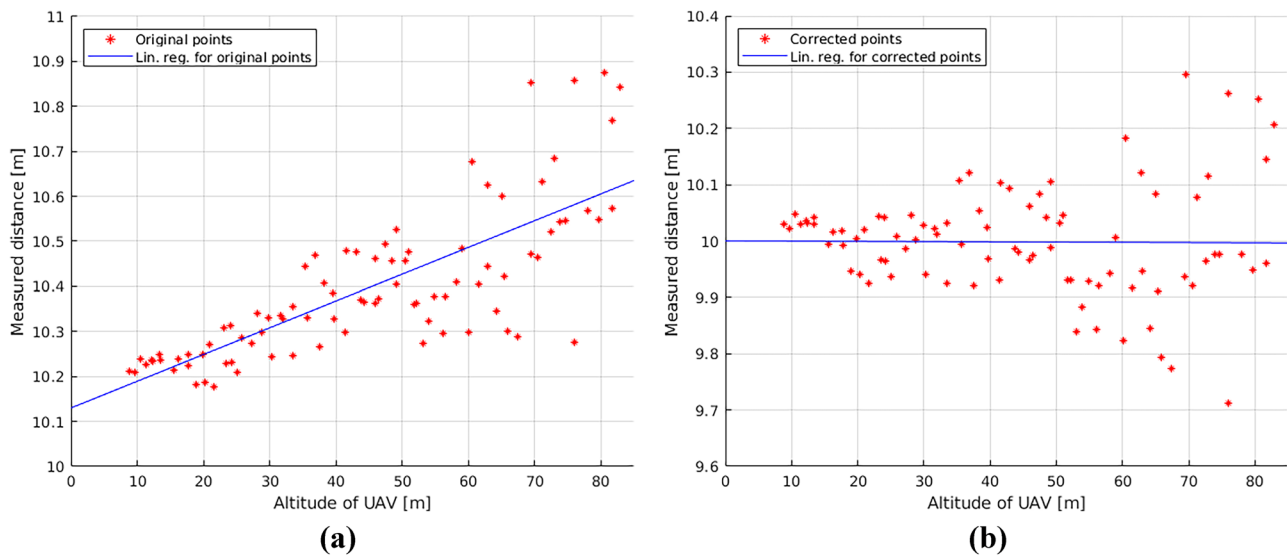


Fig. 6 Results of distance measurements between markers (real distance – 10 m) based on altitude of the UAV's camera recording (a) original (without any correction), (b) corrected (dependent on UAV's altitude).

of marker detection. If a corner of the marker is misplaced, it has an influence on the scaling factor from pixels to metres, so on the distances in metres as well.

Therefore, some kind of scaling correction factor can be introduced, taking into account the current altitude of the drone. Approximately, the error grows linearly with the altitude, so it can be significantly reduced with the calculation as in Equation 2. It presents the way to calculate the marker size after applying the correction based on the drone's altitude. The coefficient 0.00057 (which equals 0.057%) can be interpreted as a change of the perceived by the vision system marker size for one metre of altitude difference.

$$\text{newML} = \text{ML} \cdot (1 - 0.00057 \cdot h) \quad (2)$$

where: *newML* is the marker size after correction, *ML* is the real marker size, and *h* is the altitude of the UAV above the ground in metres.

The distances between the markers obtained after the correction are presented in Fig. 6b. The results are significantly better and much closer to the real distance (10 m), independently of the altitude of the drone. Below 50 m of altitude the errors are mostly in the range of 1% of the real distance between the markers. Above that altitude they are slightly bigger (up to 3%), but they are much lower than without the correction (errors up to 9%).

Thus, the usage of the introduced correction factor is well justified and it was taken into account in static and dynamic experiments.

Secondly, the measurements made from a high altitude are more prone to errors. The reason for that is the smaller

size of the objects in pixels. At the same time, the edges of the objects are blurred, so it is more demanding to correctly detect their positions, which is crucial in marker corners detection. In some cases, the ArUco markers were not detected at all, when it was not possible to correctly decode their IDs. Therefore, selecting the “proper” altitude of a UAV is a compromise between the field of view and the accuracy of the calculated distances.

Thirdly, the maximum altitude of the UAV at which the detection of the markers used by us is still possible is around 100 m. In Fig. 6 we limited it to around 80 m as up to that altitude we can consider detections as “stable” in various conditions. In comparison to our initial work [2], in which we used different detection parameters, this result is much better – previously, the maximum altitude was around 40 m. However, such a limit is generally influenced by a number of factors – the camera used, the parameters of the detector and the lighting conditions. The last factor (e.g. harsh sun) has an effect of reducing the contrast of the image and thus limiting the detection accuracy with increasing altitude.

5.1.2 Marker Orientation Test

The existing inaccuracies in marker corners detection inspired us to conduct another test. Its aim was to analyse the effect of marker orientation with regard to the camera. Just like in the altitude test, we placed two markers on the ground at a distance of 10 m. The UAV was hovering above them at an altitude between 10 and 80 m. Next, we changed the orientation of one marker by a small angle and did a similar flight. We did the same steps a few times, recording the

markers with different orientations. For each of them, the distances between the markers were calculated at multiple altitudes. It turned out, however, that the influence of the orientation is so small that the results are nearly identical. Therefore, there was no need to take into account any correction based on the marker's orientation.

5.1.3 Centroid Test

For the vision method based on the DCNN an additional experiment was carried out. Its aim was to investigate the calculation of the object centroid in relation to the position and the size of the object in the image. This test was motivated by the requirement to convert the vehicle's centroid coordinates to its ArUco marker, to allow a direct comparison of both methods, which was not done in our previous tests.

On the basis of manual measurements, the centre of the vehicle (precisely – the host) was established in a view directly from above. Additionally, its position in relation to the ArUco marker was calculated. Next, a flight above the vehicle was made in a way to make the car visible in different parts of the image. Then, for both vision methods the centroid of the host car was calculated. In case of the ArUco-based method, after detecting the marker, the displacement obtained before was used to establish the centroid of the vehicle. For the DCNN-based method, the centroid of the segmented vehicle mask was obtained. Both results were then compared. A few frames from the test sequence are presented in the form of a superposition of the host car in Fig. 7 – in these positions the analysed vehicle was visible on particular frames.

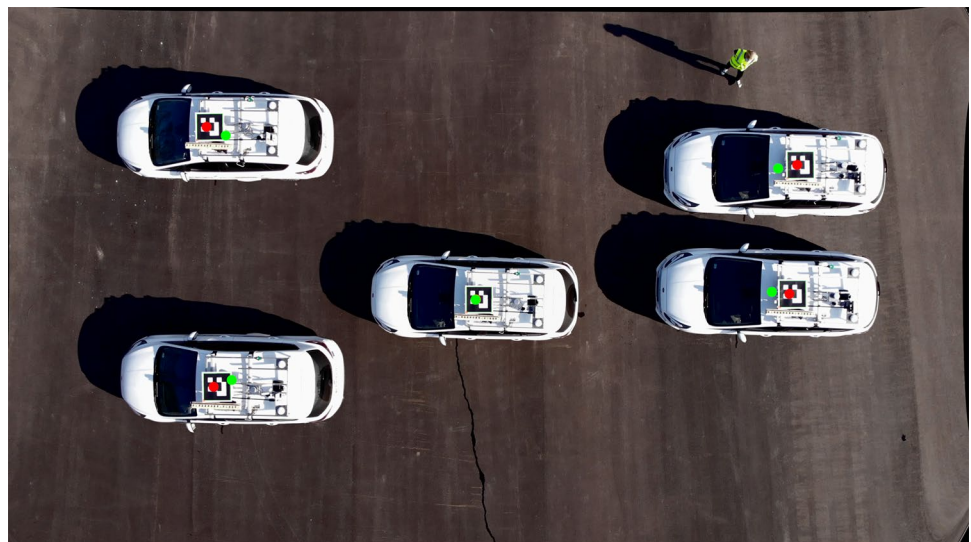
It turned out that when the car was not in the centre of the image, the DCNN centroid was closer to the centre of that image than the centroid from the ArUco method. It was

caused by the fact that the car was not visible directly from above, but from an angle, so other parts of the car (side, front, and rear) were visible as well. The difference between both methods was calculated and it turned out that the calculated positions for DCNN were around 7% smaller than for the ArUco method. Therefore, another correction was evaluated to enable a direct comparison between both vision methods. However, additional tests on other sequences at different altitudes did not confirm the results presented above (around 7% of difference). It turned out that varying altitude also has a big effect on DCNN centroids and the calculated percentage value varies as well.

Eventually, at this stage of the project we decided not to use any correction for the centroids. Not only is the evaluated correction factor related to the position of the object in the image, but also to the camera used and the altitude of the drone above the ground. Therefore, some inaccuracies of the measurements by the DCNN method may be caused by the problem discussed above and the further the car is from the centre of the image, the bigger that inaccuracy is. However, this negative effect is diminished when the UAV is at higher altitudes and the errors are then much smaller (like in levelXdata solution).

An additional source of errors is the situation in which only the part of the vehicle is visible in the image – then its centre does not match the reality, e.g. when the object is just the rear part of the car. However, a DCNN is at least able to correctly determine the distance to the closest point of this object. On contrary, the ArUco-based method may not be able to detect such a vehicle at all, when its marker is not entirely visible in the image. Therefore, both analysed vision methods have some pros and cons, so the idea of combining their advantages in one solution seems quite justified.

Fig. 7 Centroids calculated with the ArUco method (red circles) and the DCNN method (green circles). Note that when the vehicle is in the centre of the image, both centroids are basically in the same place.



5.1.4 Velocity Test

We have also conducted additional experiments to verify whether our vision-based algorithms can work with different vehicle and UAV velocities. At first, the vehicle under test was driving in a straight line at a constant speed of 30 km/h, while the UAV was operated manually to fly around 30 m above the car. Then, in next tests we increased the speed of the vehicle by 10 km/h up to 70 km/h, which was the maximum one for the drone, thus limiting the experiments. As the relative speed between the drone and the car was small, the images recorded were very sharp, thus enabling correct and accurate marker detections for both vision methods.

Later, we conducted another type of test – the drone was hovering above the road and the vehicle was driving on it at a constant speed, again ranging from 30 to 70 km/h. This time, with the increasing speed, there were some frames on which the car (with the marker) was blurred, resulting in problems with marker detection in ArUco-based method – this was happening at speeds over 50 km/h. One of the possible solutions to this problem would be to use an event camera, where the blur does not happen, but it would require a different approach to algorithm implementation. Anyway, better lighting conditions helped to acquire sharper images and obtain correct detections with vehicle speed of 70 km/h.

In case of DCNN-based method such problems did not occur, as the network was able to accurately detect and segment the vehicle at different speeds, independently of the existing blur that was problematic for the ArUco-based method – therefore, the DCNN seems to be better in this task.

In conclusion, the proposed system can be used in experiments with a variety of vehicle velocities, but at the higher ones the DCNN-based method seems like a better choice, as it is not as vulnerable to blur as the ArUco-based method.

5.2 Static Scenario

In this scenario, all vehicles remained stationary. The host car was placed in the middle of the scene, between other vehicles. Contrary to the initial tests, the results of which were published in our previous work [2], we decided not to generate an orthophotomap as it was not very useful in our experiments. The analysed scene is visualised in Fig. 8a, where the vehicles are marked with letters.

The main aim of the test was to estimate the accuracy of the measurements using both visual methods and compare them with ground truth and LiDAR (evaluated sensor) data. The secondary aim of the static test was to confirm the maximum height at which the registration of the ArUco markers and LEDs is possible. In case of the ArUco markers, just like in a dedicated altitude test 5.1.1, correct detections were possible up to 80 m (in favourable conditions even up to 100 m). For the LED system, the problems described in 4.2.2 appeared – it was difficult to correctly determine the position and the state of particular LEDs, when the UAV was above 40 m. It can be perceived as the limiting height up to which the entire system works properly. However, in case of using another method for synchronisation (a wireless communication channel), correct marker detection (so vehicle as well) is possible up to 80–100 m.

The reference data was obtained with manual measurements using a traditional tape measure. Contrary to our first experiments, we decided not to use a differential GPS as a ground truth. This was motivated by the fact that GPS measurements had to be shifted to a common coordinate system to compare them with other sensors and these displacements generated some extra errors. However, we will try to correct these errors or find a different solution in future work. The data from the LiDAR sensor was also acquired (Fig. 8b) as in this experiment it is the evaluated vehicle perception system. It was labelled manually with a 10 cm accuracy.

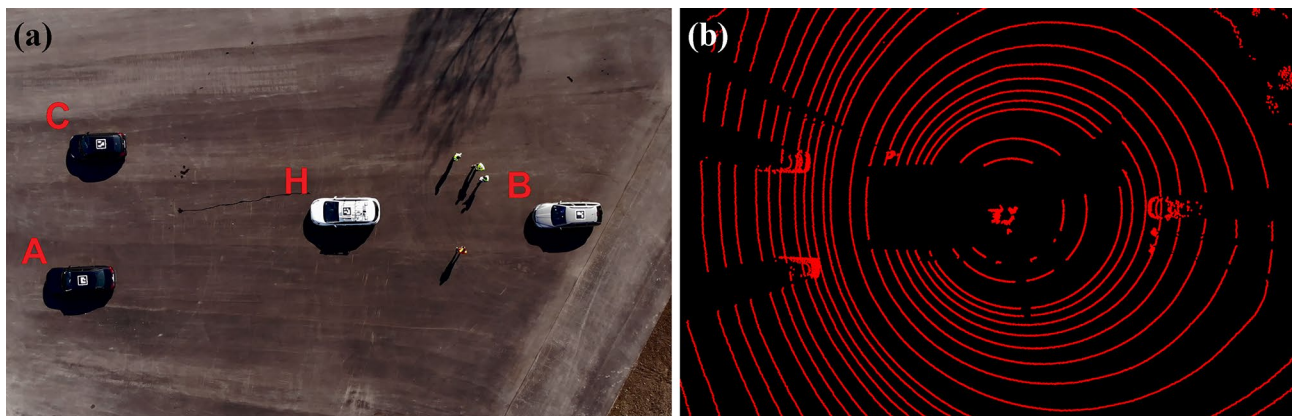


Fig. 8 (a) Camera image of the test scene, (b) corresponding LiDAR point cloud projected on the ground plane.

All of the vehicles used were visible in the image when the UAV was hovering at an altitude of around 30 m. From that point we increased the altitude up to 80 m. With such a change, the field of view increased significantly – from 37×21 metres to around 116×66 metres. For the images recorded in that range, we calculated the distances between vehicles, using the correction described in altitude test 5.1.1. For each vehicle two measurements from the host car were made – first to the centre of its ArUco marker and second to its closest point (which usually corresponds to its front or rear bumper). The results obtained in this experiment were averaged over the entire sequence (over 200 video frames) and summarised in Table 1.

5.2.1 Results Analysis

Several conclusions can be drawn from the information presented in Table 1. In general, the distance measurements obtained with the analysed methods are quite similar. The differences from the reference (manual) measurements may be caused by various reasons, depending on the evaluated method, which are briefly discussed below.

For the method based on the ArUco markers, the most important thing is the detection accuracy – not only does it apply to the centre of the marker, but even more to its dimensions. Accurately detected marker corners (i.e. position error equals fraction of a pixel) are the key elements that enable correct scaling of the distances to metric units. The altitude of the recordings also has an important effect on the results, however, compared with our initial research [2], we managed to reduce it by introducing the correction factor. Eventually, the maximum mean error (for one point) was just 0.26 m and the average errors for other points were below 0.2 m, which is a big improvement over our previous work [2]. What is more, the variability of the measurements is quite low – below 0.2 m, which translates to around 1% of the measured distances.

In case of DCNN, as already mentioned, the correct object segmentation is essential. If some pixels belonging to the object are omitted or some background is attached to it, then the position of its centroid, which defines the location of each vehicle, is misplaced. The proper masks are also important for other measurements, as the distances between the reference point on the host car and the selected points from the other vehicles' masks are taken into account. However, the masks after re-training the network were pretty good – the mean errors calculated to their closest points (so more or less vehicle bumpers) did not exceed 0.1 m. A difficult task, which was described in 5.1.3, is to determine the relationship between the centroid and the position and size of the object in the image. Therefore, this problem is the main reason for much bigger errors (maximum of 0.73 m) in measurements to vehicle markers than in the ArUco-based method, especially for Vehicle B. Anyway, even in such problematic cases the measurement variability was not very high – below 0.3 m, so up to 1.5% of the measured distances. That means the main source of errors can be eliminated by effectively solving this issue (object's centroid calculation), bringing the results much closer to the reference values.

In case of the LiDAR, the differences may occur (especially for further objects) due to a lower density of the points and the resulting difficulties in the precise manual determination of proper points (uncertainty of around 10 cm). These are the main reasons for such big errors in calculating distances to the markers (up to 0.5 m) compared with the bumpers (up to 0.1 m).

Another potential source of distance differences, which is worth noting here, is the need to express all measurements in a common coordinate system. In the considered case, it was the coordinate system of the LiDAR (the evaluated perception system). Therefore, for the results of the vision algorithms, the appropriate displacements from their coordinate systems to the LiDAR one were necessary. One more reason for distance differences may

Table 1 Results of the static experiment (in metres). In columns 3–6 the distances calculated from the host vehicle and obtained with different methods are presented: the ground truth [R], two vision-based methods [V1], [V2] and LiDAR [L]. In case of vision-based methods ([V1], [V2]) the values presented in the table are mean distances over the entire sequence along with the corresponding standard deviations

Car	Point	Measurement [R]	ArUco [V1]	DCNN [V2]	LiDAR [L]	V1-R	V2-R	L-R
Vehicle A	marker	19.17	19.08 ± 0.07	18.71 ± 0.21	18.84	-0.09	-0.46	-0.33
	bumper	16.87	16.71 ± 0.15	16.83 ± 0.18	16.90	-0.16	-0.04	0.03
Vehicle B	marker	14.88	14.95 ± 0.07	14.15 ± 0.11	14.44	0.07	-0.73	-0.44
	bumper	11.92	11.77 ± 0.10	11.83 ± 0.16	11.82	-0.15	-0.09	-0.10
Vehicle C	marker	17.63	17.89 ± 0.10	17.92 ± 0.26	17.72	0.26	0.29	0.09
	bumper	16.12	16.18 ± 0.16	16.02 ± 0.20	16.16	0.06	-0.10	0.04

(all in metric values). In columns 7–9 the mean differences in results of the analysed methods ([V1], [V2], [L]) regarding the ground truth [R] are shown (for clarity without standard deviations). These differences are quite low for all the evaluated methods (mostly below 0.5 m, for ArUco-based below 0.3 m), confirming the correctness of the calculations made.

also be an imperfect calibration of the camera used – even if done in a proper way, some small inaccuracies may happen.

Summarising the results obtained in this test it can be stated that the measurements based on the camera (ArUco, DCNN) and the LiDAR are generally similar to the distances between the vehicles determined with manual, ground truth measurements. In most cases, the average error is less than 0.5 m. The system works correctly for a range of altitudes at which we recorded the scene – precisely between 30 and 80 m.

Although the errors are in a similar range as in [2] (mostly below 0.5 m), the updated algorithm is more versatile and maintains a good quality independently of the drone's altitude. Another thing is that we managed to calculate two distances for each vehicle (which was not done in previous work) and the errors are still in the same range. Therefore, it can be concluded that the use of vision methods is quite a good reference for a data from e.g. a LiDAR.

The comparison of different methods from the literature with our solutions is presented in Table 2. The proposed algorithm shows a comparable accuracy to the one from levelXdata [5], although the authors used there a slightly different methodology than us – comparing midpoint of the vehicle's bounding box with manually created labels, obtaining the positioning error mostly below 0.1 m. It is worth mentioning that this kind of manual reference is also prone to errors, especially when the vehicles in the image are very small.

In conclusion, both camera-based solutions have worse accuracy than the RTK system of RT-Range [4]. On the other hand, vision-based solutions ensure better accuracy than a traditional DGPS system from RT-Range.

5.3 Dynamic Scenario

In this scenario, the host car drives between static vehicles with the speed of around 20 km/h. At the same time, the UAV is controlled manually to fly with a similar

speed above the moving car in order to keep it around the centre of the image – it can be perceived as a kind of manual tracking. In a target solution, the tracking is planned to be automatic, i.e. without the human operator. An exemplary sequence of photos from this test is shown in Fig. 9. The aim of the experiment was to analyse the accuracy of the measurements in dynamic conditions that potentially negatively affect the detection of the ArUco markers and vehicles (using the DCNN method). In addition, the data from the LiDAR sensor is also changing – in this case, just like in the static scenario, the data was labelled manually.

Contrary to the initial dynamic experiment (described in [2]), we did not use a GPS system as a reference due to the required displacements needed for comparison and some additional errors resulting from them. Instead, we decided to compare the measurements from the LiDAR and vision methods in a direct way. We carried out multiple experiments during which the UAV was flying at different altitudes. The lower ones made it possible to determine the distances using a camera with higher accuracy (with an error of just a few cm), but at the same time the camera field of view was limited – for example, at 15 m of altitude it was just 20×11 metres. Therefore, not all analysed vehicles were visible on some frames. A LiDAR sensor works in a different way, so in its case that problem was non-existent as its range was the same in all experiments. The above-mentioned problem limited the number of camera frames on which the comparison with LiDAR could be done.

Therefore, we decided to use a scenario, in which the UAV was flying at an altitude of around 45 m. It enabled the comparison of the entire sequence with all vehicles visible in the image as the field of view was 53×30 metres. In our previous work [2] the marker size on subsequent frames was averaged over 10 frames as it filtered out single incorrect detections. However, in this work due to updated detection parameters the averaging was not necessary as the results with and without it were basically the same.

The results of the dynamic experiment are summarised in Table 3, which presents the mean differences of

Table 2 Comparison of measurement accuracy between different solutions.

Solution	Equipment	Method	Approx. accuracy [m]
RT-Range [4]	coupled inertial and satellite navigation, additional components in vehicles	RTK	0.02
		DGPS	0.4
levelXdata [5]	UAV with 4K camera	DCNN	0.1
Our previous work [2]	UAV with 4K camera	ArUco markers	0.3
		DCNN	1.0 (bumper only)
This work	UAV with 4K camera	ArUco markers	marker 0.3, bumper 0.2
		DCNN	marker 0.7, bumper 0.1



Fig. 9 ArUco-based detection and distance measurements on multiple frames from the dynamic test sequence.

the calculated distances using vision methods in relation to LiDAR measurements. The number of camera frames (synchronised with frames from the LiDAR working with a frequency of 10 Hz) differs for particular vehicles as not

on all LiDAR frames was it possible to choose a proper point (marker centre).

5.3.1 Results Analysis

For both vision methods the average differences with reference to LiDAR measurements were mostly below 0.5 m – precisely just in two cases (out of 12) the errors exceeded this value and the maximum one was 0.61 m. What is worth mentioning, such similar errors were achieved for both points (ArUco marker and bumper) for all analysed cars. However, the comparison of these results with the ones from the initial dynamic experiment [2] is not straightforward as we did not use a GPS system this time. Due to obvious reasons, manual measurements using a tape were not possible in this experiment. Therefore, it is not possible to directly

Table 3 Results of the dynamic experiment – comparison of vision methods with LiDAR measurements (in metres). The average differences between the analysed systems are mostly below 0.5 m.

Car	Point	Frames	LiDAR - ArUco	LiDAR - DCNN
Vehicle A	marker	115	0.15	0.61
	bumper	129	0.56	0.41
Vehicle B	marker	107	-0.30	0.39
	bumper	107	-0.10	-0.28
Vehicle C	marker	116	-0.01	0.16
	bumper	129	0.25	0.25

compare these results with the ones from the static experiment, where it was shown that LiDAR's data also has some inaccuracies with regard to the ground truth.

Nevertheless, the differences were again caused by various reasons. In case of the ArUco markers, just like in the static scenario, the main challenge is the accurate determination of the marker corners and its size. Compared with the initial test, we corrected the placing of the elements on the host car, so there were no shadows on the ArUco marker and it was detected more precisely during the entire sequence.

In case of DCNN, the distances between the objects in pixels were scaled to metres using the marker dimensions from a particular frame.

This turned out to be a slightly better solution than using averaged marker size from multiple frames and a much better solution than in the previous test, where we used the averaged marker size from the entire sequence. The neural network used is able to make a better segmentation of the vehicles than the ArUco-based method, which is entirely dependent on the accuracy of the marker detection. However, in some more demanding cases the cars were detected with their shadows, resulting in incorrect distances between them. These situations were significantly reduced in comparison with the initial test [2], but they still occasionally appear. But the most significant reason for differences seems to be the calculated position of the centroid for every car. This problem, described in 5.1.3, seems even more important in case of a dynamic test, where the vehicles move.

It is worth noting that with the ArUco-based method, such problems (regarding shadows and centroids) do not occur, because it only detects the marker instead of the entire object (i.e. car). Therefore, a solution that combines the advantages of both methods may be a good idea for the future – for example a network able to precisely segment the objects and correctly detect the markers.

In summary, the differences of both vision methods compared with LiDAR measurements were mostly below 0.5 m. This result seems much better than in the initial dynamic test. Even without a ground truth in this experiment, the improvements in ArUco detection and DCNN segmentation as well as more accurate scaling from pixels to metric units allowed us to achieve quite good results. Of course, the vision algorithms still require additional analysis and experiments in different conditions, especially in unfavourable ones.

The big advantage of the LiDAR sensor is the wide range and constancy of perspective – in case of the vision methods from a UAV's camera the range of the scene and the accuracy of measurements is dependent on the altitude of the drone and the camera itself. Thus, the evaluation of the LiDAR by a vision system is a challenge, especially when monitoring a bigger space around the vehicle.

6 Summary

In this article, we have presented experiments during which the possibility of using a 4K camera mounted on a UAV for the evaluation of autonomous vehicle perception systems was analysed. The vision algorithms applied in this work make it possible to use any off-the-shelf UAV equipped with a camera for this task.

As a test case, we assumed car detection using a LiDAR sensor (the evaluated vehicle perception system) and considered two vision methods: ArUco and DCNN. Two types of tests were carried out – static and dynamic ones. In the first one, the measurement accuracy of approx. 0.3 m for the ArUco-based method and 0.5 m for the DCNN-based method was obtained regarding the ground truth. It should be noted that the commercial solutions offer an accuracy of approx. 0.02 m (RT-range), while a vision-based approach of levelXdata around 0.1 m. However, in this solution the authors used slightly different methodology. Therefore, it is not possible to easily compare its results with our method.

In the dynamic experiment, as a ground truth was unavailable, the results of the vision methods were compared with the LiDAR measurements – they turned out to be in the range of around 0.5 m. Due to the multiple improvements to vision algorithms, these results are much better than in our initial work [2]. However, there is a room for further refinement, which should be done in the next versions of the system.

For the ArUco markers, inaccurate corner detections occur in some cases, which leads to erroneous positions of the objects and scaling the distances between them from pixels to metric units. For the DCNN method, localising the objects (vehicles) with their centroids makes it difficult to compare them with other methods, especially when they are visible in different parts of the image. However, in the target solution, where the bounding boxes obtained from the perception systems will be compared, the masks from the DCNN method seem to be the best reference.

Summing up, the accuracy obtained currently is better than in our initial work, so we strongly believe that the proposed approach is an interesting alternative to the existing solutions and therefore can be used in real evaluation scenarios to calculate various KPIs.

We have also implemented both vision methods on Nvidia Jetson Xavier AGX – for the ArUco-based method we achieved processing of 4 fps, while for the DCNN-based one 2 fps. With some additional improvements, the algorithms described in this work can be run on a hardware platform in real-time. Therefore, this work is the first step on the way to an on-line system for evaluation of vehicle perception systems.

6.1 Future Work

The experiments described in this article are an introduction to a fully automatic solution for the real-time evaluation of perception systems for an autonomous vehicle. On their basis, the next stages of further work can be outlined. First, we plan to concentrate on applying on-line image processing and solving the frame synchronisation issue. This can be done in two variants: on-ground using WiFi communication with the drone or embedded on the UAV. The first approach should be possible with the use of the API provided by DJI. However, transmitting a 4K video stream using WiFi can be a challenge, thus the system may have to work with a lower resolution. Then a ground PC, equipped with a GPU, could be responsible for all image processing (also tracking the host-vehicle) and control. This would speed up the next experiments, as most of the results would be available on-line. The advantage of this solution is also the use of an off-the-shelf UAV.

The second approach would require the use of a custom UAV platform. In this case, we would have full control over all equipment: camera, lens, and computing platform – the use of an embedded GPU and/or heterogeneous programmable SoC is assumed. Also, one of the ideas regarding the custom UAV is the use of a specialised drone, which is able to operate in unfavourable conditions (rain, strong wind, etc.).

Second, it is necessary to refine or combine the applied vision algorithms. The use of the Aruco markers has some limitations – for more accurate determination of their positions, bigger markers may be necessary, but this is limited by the width of the vehicle's roof. On the other hand, to be able to analyse a larger space around the vehicle, the UAV should be located even higher than in our experiments – up to 100–120 metres. Despite a good accuracy of the ArUco-based algorithm it seems better to use methods that do not require markers – like based on a DCNN. Moreover, in this case it would be possible to add detection and tracking of other classes of objects, e.g. pedestrians and cyclists (which is impossible with the ArUco markers).

The DCNN method should be refined to be even more robust in different meteorological conditions – especially in the case of shadows. Finally, it is necessary to conduct multiple additional tests to better assess the suitability of the proposed solution.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in

the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Oxford Technical Solutions (2020a). *Rt-range user manual*. <https://www.oxts.com/app/uploads/2017/07/rt-rangeman.pdf>
2. Blachut, K., Danilowicz, M., Szolc, H., Wasala, M., Kryjak, T., Pankiewicz, N., & Komorkiewicz, M. (2021). Automotive perception system evaluation with reference data obtained by a UAV. In *Workshop on Design and Architectures for Signal and Image Processing (14th Edition) DASIP '21* (p. 10–18). New York, NY, USA: Association for Computing Machinery.
3. (2021). *Software model of the proposed application*. https://github.com/vision-agh/apse_uav.
4. Oxford Technical Solutions (2020b). *Rt-range (for adas)*. <https://www.oxts.com/products/rt-range-hunter/>
5. fka GmbH (2020). *levelxdata*. <https://www.levelxdata.com/>.
6. Bock, J., Vater, L., Krajewski, R., & Moers, T. (2021). Highly accurate scenario and reference data for automated driving. *ATZ worldwide*, 123, 50–55.
7. Bock, J., Krajewski, R., Moers, T., Runde, S., Vater, L., & Eckstein, L. (2020). The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1929–1934).
8. Vishnukumar, H. J., Butting, B., Müller, C., & Sax, E. (2017). Machine learning and deep neural network artificial intelligence core for lab and real-world test and validation for adas and autonomous vehicles: Ai for efficient and quality test and validation. In *2017 Intelligent Systems Conference (IntelliSys)* (pp. 714–721).
9. Di Mare, G., Vico, F., Crisci, F., Montieri, A., Amoroso, D., Marino, B., et al. (2019). An innovative real-time test setup for adas's based on vehicle cameras. *Transportation Research Part F: Traffic Psychology and Behaviour*, 61, 252–258.
10. Khan, N. A., Jhanjhi, N., Brohi, S. N., Usmani, R. S. A., & Nayyar, A. (2020). Smart traffic monitoring system using unmanned aerial vehicles (uavs). *Computer Communications*, 157, 434–443.
11. Team, O. (2020). *Open source computer vision library*. <https://opencv.org/>.
12. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., & Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51, 481–491.
13. Romero-Ramirez, F., Muñoz-Salinas, R., & Medina-Carnicer, R. (2018). Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76, 38–47.
14. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., & Berg, A. C. (2015). *SSD: single shot multibox detector*. <https://arxiv.org/abs/1512.02325>.
15. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779–788).
16. Liu, M., Wang, X., Zhou, A., Fu, X., Ma, Y., & Piao, C. (2020). Uav-yolo: Small object detection on unmanned aerial vehicle perspective. *Sensors*, 20, 2238.
17. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 1137–1149.

18. He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). *Mask R-CNN*. <https://arxiv.org/abs/1703.06870>.
19. Ahmed, I., Din, S., Jeon, G., & Piccialli, F. (2020). Exploring deep learning models for overhead view multiple object detection. *IEEE Internet of Things Journal*, 7, 5737–5744.
20. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014* (pp. 740–755). Springer International Publishing.
21. Zhu, P., Wen, L., Du, D., Bian, X., Hu, Q., & Ling, H. (2020). *Vision meets drones: Past, present and future*. <https://arxiv.org/abs/2001.06303>.
22. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). *Facenet: A unified embedding for face recognition and clustering*. <https://arxiv.org/abs/1503.03832>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.