Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Vision-Based Real-Time Indoor Positioning System for Multiple Vehicles ⋆

Maximilian Kloock * **Patrick Scheffe** * **Isabelle Tülleners** *
Janis Maczijewski * **Stefan Kowalewski** * **Bassam Alrifaee** *

* *Chair for Embedded Software, RWTH Aachen University, 52074 Aachen, Germany (corresponding e-mail: kloock@embedded.rwth-aachen.de).*

Abstract: We propose a novel external indoor positioning system that computes the position and orientation of multiple model-scale vehicles. For this purpose, we use a camera mounted at a height of 3.3 m and LEDs attached to each vehicle. We reach an accuracy of about 1.1 cm for the position and around 0.6 ° for the orientation in the mean. Our system is real-time capable with a soft deadline of 20 ms. Moreover, it is robust against changing lighting conditions and reflections.

Keywords: Autonomous Mobile Robots, Multi-vehicle systems, Localization, Indoor Positioning

SUPPLEMENTARY MATERIAL

A demonstration video of this work is available at `https://youtu.be/k6aD5G9DW4o`.

More information about the CPM Lab is provided at `https://cpm.embedded.rwth-aachen.de`

1. INTRODUCTION

Applications for indoor positioning are, e.g., humanoid robots or (model-scale) autonomous vehicles. We are building the Cyber-Physical Mobility Lab (CPM Lab) with 20 model-scale vehicles (μCars) to develop and to evaluate algorithms for networked and autonomous vehicles. For this purpose, we developed an Indoor Positioning System (IPS), since the knowledge of the vehicle positions is significant as computations of trajectories and the interaction between vehicles highly depends on their current positions. Additionally, trajectory control depends on the accuracy of positioning. The precision of the IPS is improved by sensor fusion with dead reckoning data. Nevertheless, the IPS is the only absolute reference system for the positioning. Therefore, to test functionalities in the field of networked and autonomous vehicles using model-scale vehicles, an accurate IPS is required.
There are already many systems providing the position of indoor robots. Some approaches are wave-based using WLAN or Radio Frequent Identification (RFID), e.g., (Ladd et al., 2005; Chawla et al., 2010), or ultrasonic and Radio Frequency (RF), e.g., (Diab et al., 2015; Fukuju et al., 2003). Overall, those approaches have the deficiency of an accuracy in the decimeter to meter range. More precise approaches are vision-based.

⋆ This research is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within the Priority Program SPP 1835 "Cooperative Interacting Automobiles" and the Post Graduate Program GRK 1856 "Integrated Energy Supply Modules for Roadbound E-Mobility".

There are many different attempts for vision-based indoor positioning, e.g., (Mautz and Tilch, 2011). Some use feature detection, e.g., (Hile and Borriello, 2008; Lee and Song, 2007), some use Visible Light Communication (VLC), e.g., (Ghimire et al., 2018; Rátosi and Simon, 2018; Yoshino et al., 2008) and others use Simultaneous Localization and Mapping (SLAM) methods, e.g., (Ido et al., 2009; Noonan et al., 2018). However in those approaches the target object is equipped with a camera and positions itself depending on its view. This requires additional computation power on the robots. In order to keep the computation requirements on the vehicles low, we develop a system which externally determines the position and the orientation (pose) of the target object. Our approach is inline with the attempt of (Faessler et al., 2014). However, (Faessler et al., 2014) deals only with a single vehicle for positioning. Our IPS is able to determine the pose of multiple vehicles.
There are also approaches to detect tags, e.g. (Neunert et al., 2016; Olson, 2011). However, due to changing lighting conditions, the tags have to be larger than the model-scale vehicles. Therefore, such approaches are not suitable in our case.
We propose a new vision-based IPS that externally computes the poses of multiple model-scale vehicles. For this, we use LEDs attached to the autonomous vehicles and a camera mounted on the ceiling. The LEDs can be detected robustly even with changing lighting conditions using short exposure times and bright LEDs. To distinguish the vehicles, one of the LEDs flashes in a vehicle specific frequency. Furthermore, our system can be used in a real-time environment with a soft deadline of 20 ms.

The rest of the paper is structured as follows. Section 2 gives an overview of the infrastructure and the IPS algorithm. The correctness of this algorithm is shown in Section 3. Section 4 evaluates our IPS and Section 5 concludes this paper.

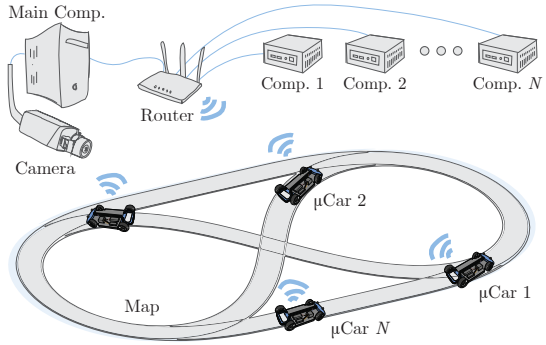


Fig. 1. System overview. μCars drive on a printed map and control trajectories planned by external computation units in a networked manner. The camera is used for positioning of the vehicles.

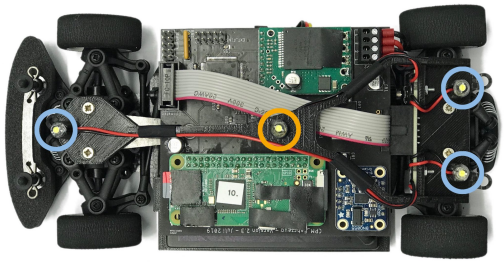


Fig. 2. μCar with four mounted LEDs. The blue-marked LEDs (outer ones) are the positioning LEDs where the yellow-marked LED (inner one) describes the identification LED.

2. INDOOR POSITIONING SYSTEM

2.1 System Overview

Fig. 1 sketches our system overview. Up to 20 vehicles drive on a map of 4mx4.5m size. One external computation device is provided for each vehicle. A camera records the whole map and a main computer performs the image processing. A router allows for wireless communications between the external computation devices, the main computer, and the vehicles. For more details of the CPM Lab, see (Kloock et al., 2020).

We mount the camera at a height of 3.3 meters straight downwards. Its field of view covers the whole map. Furthermore, we attach four LEDs to each vehicle as shown in Fig. 2. The three outer LEDs marked in blue are arranged in a non-equilateral triangle and used to determine the pose of the vehicle. Since the triangle is not equilateral, the direction of the vehicle can be depicted. We use the fourth LED marked in yellow to distinguish the vehicles. For this purpose, it flashes in a vehicle specific frequency. This LED lays in the borders of the triangle of the outer LEDs to simplify the clustering of the points to a vehicle. Since a frequency cannot be determined in a single image, we consider a sequence of images. However for pose determination, we only use the latest one. For more details of the vehicle construction, see (Scheffe et al., 2020).

2.2 Algorithm Overview

Our algorithm consists of five steps. In a first one, a camera stream is received. In the second step, we extract the LED points. For each image, we cluster the detected points belonging to a single vehicle afterwards. In a fourth step, the detected vehicles in the current image are mapped to the vehicles from the previous one. With this, we gain the sequence of points belonging to each vehicle in different images. Having this sequence, we compute the pose and the ID of each vehicle in a last step. In the following, those steps are explained in detail.

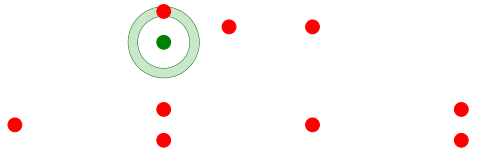


Fig. 3. The green marked annulus is the area in the vehicle-width up to a tolerance to the green marked point. It includes points that may belong to the same vehicle back.

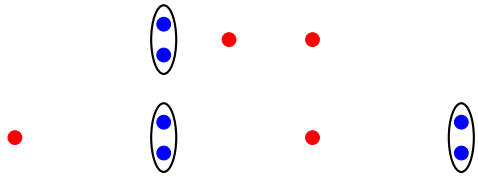


Fig. 4. The black ellipses surround all vehicle backs found with the help of the vehicle-width. The blue points are the LEDs of the vehicle back.

Get Image The camera takes images and writes them in a queue. Those images are taken equidistant with a short exposure time. The time difference between the images is constant.

Find Points In a second step, we search for the LED points in the image. As the images are taken with short exposure times, the LEDs are detected robustly using OpenCV (Bradski, 2000). We detect the contours of the LED points and determine their moments. The moments describe the centers of the blob. We filter the detected contours by size to remove disturbance points, i.e., blobs which are bigger or smaller than the expected size are dropped.

Find Vehicles In a next step, we match points to vehicles. For this purpose, we use the known distances between the LEDs as a measure. We propose two methods. First, we know the distance between the two LEDs on the back of the vehicle. In our case, the basis of the triangle. In the following, this distance is called vehicle-width. Furthermore, the longest distance between two LEDs on the same vehicle is known. This one is called vehicle-length. We use the vehicle-width to determine the vehicle back, i.e. the two LEDs mounted on the back of the vehicle.

For each point, we compute all points exactly in its vehicle-width up to a tolerance. This is shown exemplary in Fig. 3. From the geometry of the LEDs on the vehicle, it holds that if a point has another point in its vehicle-width, they form a vehicle back. Hence, we can compute all vehicle

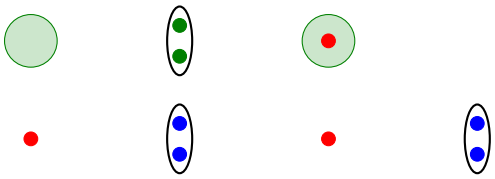


Fig. 5. The green cycles are the areas in which points belonging to the green vehicle back points are placed. They are in the vehicle-length orthogonal to the vehicle back, up to a tolerance.

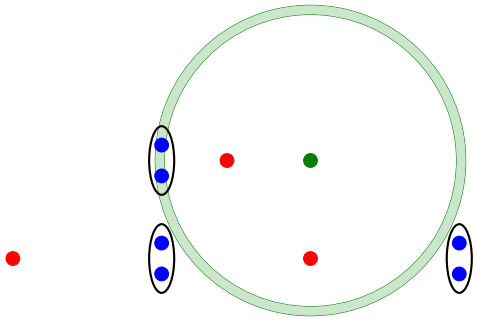


Fig. 6. The green marked annulus includes the area in the vehicle-length from the green point. This is the area, in which vehicle backs are placed that may belong to the same vehicle.

backs as illustrated in Fig. 4, i.e. the geometry avoids ambiguities in scenarios with multiple vehicles. Since the LEDs are mounted in a non-equilateral triangle, we receive the direction of the vehicle from the vehicle backs. With this knowledge, we can reduce the search space for the remaining points. Those points are orthogonal to the vehicle back in the vehicle-length. For this, we only consider those points including a tolerance, as shown in Fig. 5. That means for all vehicle backs, we know with which other points they can form a vehicle, namely those inside the green shaped area in Fig. 5. Now, we consider all points that do not belong to a vehicle back. For those, we have no restriction in direction to determine the points with which they can form a vehicle. Here, the only restriction is the vehicle-length. Hence, for each non-vehicle-back point, we consider all points in the vehicle-length as possible points with which it can form a vehicle. Those areas are shown for one point in Fig. 6. Now, we know for each point with which other points it can form a vehicle. Thus, if two points are not in the area of the other, they cannot be mounted on the same vehicle. That also means, if two LEDs are mounted on the same vehicle, the corresponding points are in the area of each other. We use this to determine the vehicles. For this purpose, we choose one point with the least possible matches. Then, we compute the intersection of its matches, including itself, with the matches of all points in its area. If this intersection is equivalent to the matches of the chosen point, the intersection only contains points belonging to the same vehicle.

Since the points from the intersection are mapped to a vehicle, we remove all those points from the remaining areas. Then, the procedure of choosing a point and intersecting the areas is repeated until all points are mapped or no progress is reached.

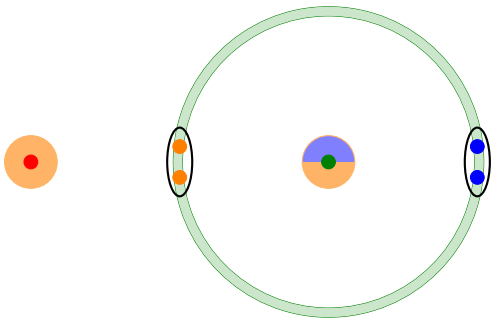


Fig. 7. The annulus of the green point and the areas of the orange vehicle back points and the blue vehicle back points. The double-colored cycle is in the area of the orange vehicle back points and the blue vehicle back points. The second blue cycle is omitted due to space reasons.

However, there may be points that are in each others area that are not mounted on the same vehicle, see Fig. 7. The green point has two vehicle backs in its area and is in the area of both vehicle backs, orange and blue. Therefore, the intersection of the sets of possible matches of the green point and the orange vehicle back is different from the original sets. Hence, they cannot be matched yet. Nevertheless, the intersection of the sets of the green point and the blue vehicle back is equal to the original set of the blue vehicle back. Thus, they may be matched. Once removed from the pool of unmatched points, the remaining points can be matched unambiguously.

For a small amount of vehicles it may be feasible to check every combination of points and check if its distances are feasible to match a vehicle. In this case, conflicts may occur where ambiguities occur in the previous procedure, e.g. in the example in Fig. 7. The check of combinations would detect three vehicles, the ones shown in the Fig. 7 and a vehicle containing of the orange vehicle-back and the green front LED. Therefore, those conflicts have to be resolved until each LED point is part of only one vehicle.

Match Vehicles Now, we have found all vehicles in one image. To determine the frequency of the identification LED, we need the sequence of points for each vehicle. For this purpose, we need to match the vehicles in the current image to the corresponding vehicles in the previous image. We use that we receive the images with a high frequency. Hence, vehicles can only move a short distance between the images. We match each vehicle to the nearest vehicle in the previous image. To avoid false matching, some plausibility checks are done. For the position and the orientation, we check whether the observed change between the previous and the current image is physically possible. For identification, we check whether the recalculation of the identity yields the same result as before.

Compute Pose By now, we have the sequence of points for each vehicle in the different images. As a resulting step, we compute the pose of the vehicles. For this, we consider the three positioning points from the current image. If the identification point is visible in the latest image, it can be filtered by the sum of the distances of one point to all other. To compute the orientation, we consider the straight line between two LED points. Then, we can compute

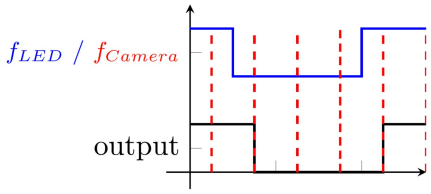
the angle between this line and the x-axis. Since we are interested in the orientation to the side of the vehicle, we determine the offset of the straight line to the side of the vehicle and add it to the previous computed angle. For more robustness, we compute the orientation for all three pairs of LEDs and take the median as result. In this way, we can remove outliers. The position of a vehicle is defined as its midpoint. Hence for positioning, we compute the midpoint of the two back points and shift this point to the midpoint of the vehicle using the previously calculated orientation.

Identification For the identification, we use the sequence of points. This sequence is received from the order of images received by the camera. For each vehicle, that sequence is an order of sets of three or four points each belonging to this vehicle in the respective image. To determine the identity, we count in how many consecutive images the identification LED is on num_{on} and in how many it is off num_{off}. Using the camera frequency f_{Camera}, the time in which it is on or off t_{on}/t_{off}, respectively, can be computed by

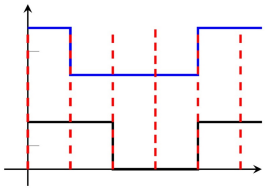
$$t_{on/off} = \frac{num_{on/off}}{f_{Camera}}$$

A mapping of LED frequencies to concrete IDs has to be provided to our system beforehand. For this approach, it is important that the LED frequencies have such high distances that they can be recognized uniquely. For this purpose, we choose the LED frequencies depending on the camera frequency and the number of images in which we expect the LED to be on or off. To guarantee that there is no overlapping, we only consider every third number for a sequence of images. For example, we expect that for the first ID it is on in two following up images and for the next ID that it is on in five following up images. With this, we can handle sampling during a switch of the LED value, as illustrated in Fig. 8. In Fig. 8(a), the camera samples while the LEDs are on or off, while Fig. 8(b) and 8(c) show scenarios where the camera samples while the LEDs are turning on or off. For robust recognition, we map the frequencies detected with the intermediate numbers to the nearest number. In the end, we have for each ID a number of images n in which we expect it to be on. From this number we can compute the frequency of the LED with the camera frequency $f_{LED} = \frac{f_{Camera}}{n}$ and the interval of detected frequencies which we map to this ID $[\frac{f_{Camera}}{n+1}, \frac{f_{Camera}}{n-1}]$. Please note that we do not check the last images again. Instead, the number of images in which the identification LED is on and the overall amount of images are tracked during operation. This approach requires some initial time to initiate the IDs of the vehicles, i.e., the minimum number of images to distinguish all used IDs.

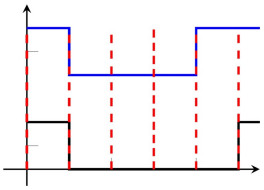
The algorithm is summarized in Algorithm 1 where $[v_1, \dots, v_n]$ describes a vehicle consisting of the image points $v_1, \dots, v_n$ belonging to its LED points, $g_1, \dots, g_n$ are the reference LED points as image points from the vehicle geometry, *dis* computes all distances of the points in the provided set sorted in ascending order and n_j is the size of the vehicle-mapping of v_j with the corresponding points $p_{j1}, \dots, p_{jn_j-1}$.



(a) Sampling when the LED is on or off.



(b) Sampling when the LED is turning on or off.



(c) Sampling when the LED is turning on or off.

Fig. 8. Possible situations while sampling. The dashed red lines depict the time points when an image with the camera is taken. The blue line describes the signal which is sampled and the black line the result of the sampling.

Algorithm 1: One step of matching a set of points to a vehicle.

Input : Vehicle Mapping for each point and a point p
Output: Whether p belongs to a vehicle, is a disturbance point or cannot be mapped, yet. If p belongs to a vehicle, the corresponding vehicle.

get $\{[p], p_1, \dots, p_{n-1}\}$;
if $n < 3$ **then return** disturbance point;
if $n > 4$ **then return** cannot be mapped yet;
$S = \bigcap_{i=1}^{n-1} \{[p_i], p_{i1}, \dots, p_{im_i}\} \cap \{[p], p_1, \dots, p_{n-1}\}$;
if $S == \{[p], p_1, \dots, p_{n-1}\}$ **then**
  if $dis(S) \approx dis(\{g_1, \dots, g_n\})$ **then return** Vehicle $[p, p_1, \dots, p_{n-1}]$;
  if $n == 3$ **then return** disturbance point;
  return cannot be mapped yet;
end
if $n == 3$ **then return** disturbance point;
return cannot be mapped yet;

3. CORRECTNESS

In the following, we prove that this computation is correct. That means, we prove the following theorem.

Theorem 1. Assume that disturbance points do not form exactly a vehicle geometry with other points. Then, it holds that:
If our system detects a vehicle consisting of the points $v_1, \dots, v_n$ with $n = 3$ or $n = 4$, the LEDs corresponding to these points are mounted on the same vehicle.

First, we formalize some properties. From the reality, we know that if we have a vehicle, all LEDs on the vehicle are at most in the vehicle-length to each other and the distance of the LED points match the distances of the vehicle geometry. The assumption yields the other direction. We

assume that if a set of points is in the vehicle-length of each other and all the distances match the vehicle geometry, the set of points belongs to one vehicle. Formally,

$$[v_1, \dots, v_n] \Leftrightarrow \forall i = 1 \dots n. \forall j = 1 \dots n, j \neq i. \\ v_i \in \{[v_j], p_{j1}, \dots, p_{jn_j-1}\} \\ \wedge dis(\{v_1, \dots, v_n\}) = dis(\{g_1, \dots, g_n\}) \tag{1}$$

For proving the Theorem 1, we first show the following lemma.

Lemma 1. Assume Equation (1). Then, it holds that:
If Algorithm 1 forms a vehicle $[p, v_1, \dots, v_{n-1}]$ with $n = 3$ or $n = 4$ for a provided point p, the LEDs corresponding to these points are mounted on the same vehicle.

Proof. We prove that if the algorithm matches a point to a vehicle, the found vehicle is indeed a vehicle by a case distinction on n. So let p be an arbitrary point with

$$\{[p], p_1, \dots, p_{n-1}\}. \tag{2}$$

Case 1. Let $n < 3$.
p is filtered out as disturbance point by line 2 of Algorithm 1. Thus, no vehicle is detected and especially no wrong vehicle.

Case 2. Let $n = 3$ with $\{[p], q, r\}$.

a) Let the LEDs corresponding to p, q, r be mounted on the same vehicle
Thus, we know from Equation (1) that

$$p \in \{[r], r_1, \dots, r_{m-1}\} \\ p \in \{[q], q_1, \dots, q_{k-1}\} \\ q \in \{[r], r_1, \dots, r_{m-1}\} \\ r \in \{[q], q_1, \dots, q_{k-1}\}$$

with this and Equation (2) we have

$$\{p, q, r\} \subseteq \{[r], r_1, \dots, r_{m-1}\} \cap \{[q], q_1, \dots, q_{k-1}\}. \tag{3}$$

Furthermore, the set S from Algorithm 1 line 4 intersecting the above intersection with the mapping of p is

$$S = \{[r], r_1, \dots, r_{m-1}\} \cap \{[q], q_1, \dots, q_{k-1}\} \cap \{[p], q, r\}.$$

Thus, from Equation (3) we receive that

$$S = \{[p], q, r\}.$$

Now, the algorithm checks in line 6 the distances. Thus, the distances of the points $dis(\{p, q, r\})$ are compared to the distances of the position points $dis(\{g_1, g_2, g_3\})$. Since the LEDs corresponding to the points p, q, r are mounted on the same vehicle, we know from Equation (1) that the distances are similar. Therefore, the Algorithm 1 finds the vehicle $[p, q, r]$ which is indeed a vehicle.

b) Let the light points corresponding to the points p, q, r do not be mounted on the same vehicle.

i) $\exists s \in \{p, q, r\}. \exists t \in \{p, q, r\} \backslash \{s\}.$
$s \notin \{[t], t_1, \dots, t_{l-1}\}$
Thus, the intersection does not contain one of the points p, q, r that means

$$\{p, q, r\} \not\subseteq S \\ = \{[r], r_1, \dots, r_{m-1}\} \cap \{[q], q_1, \dots, q_{k-1}\} \\ \cap \{[p], q, r\}$$

Therefore, $S \neq \{[p], q, r\}$. In that case, the Algorithm 1 determines p as a disturbance point in line 10 and does not detect a wrong vehicle.

ii) $\forall s \in \{p, q, r\}. \forall t \in \{p, q, r\} \backslash \{s\}.$
$s \in \{[t], t_1, \dots, t_{l-1}\}$
From Equation (1), we then know that

$$dis(\{[r], r_1, \dots, r_{m-1}\} \cap \{[q], q_1, \dots, q_{k-1}\} \\ \cap \{[p], q, r\}) \\ \neq dis(\{g_1, g_2, g_3\}).$$

Thus, the Algorithm 1 detects p as disturbance point in line 7 and does not detect a wrong vehicle.

Case 3. Let $n = 4$ with $\{[p], q, r, s\}$.

a) Let the LEDs corresponding to p, q, r, s be mounted on the same vehicle
This case is equivalent to Case 2a). The set S is equal to $\{[p], q, r, s\}$ and since the points p, q, r, s belong to a vehicle, the distances are comparable to $dis(\{g_1, g_2, g_3, g_4\})$. Thus, the algorithm finds the vehicle $[p, q, r, s]$ which is indeed a vehicle.

b) Let the light points corresponding to the points p, q, r, s do not be mounted on the same vehicle.
Analog to Case 2b), we need to distinguish two further cases.

i) $\exists u \in \{p, q, r, s\}. \exists t \in \{p, q, r, s\} \backslash \{u\}.$
$u \notin \{[t], t_1, \dots, t_{l-1}\}$
Thus analog to Case 2b)i), the set $S \neq \{[p], q, r, s\}$. Therefore, the Algorithm 1 cannot map the point p yet and terminates in line 11 without finding a wrong vehicle.

ii) $\forall s \in \{p, q, r\}. \forall t \in \{p, q, r\} \backslash \{s\}.$
$s \in \{[t], t_1, \dots, t_{l-1}\}$
Thus with the same argumentation as in Case 2b)ii), the algorithm cannot map p yet and terminates in line 8 without finding a wrong vehicle.

Case 4. Let $n > 4$
p cannot be mapped yet by Algorithm 1 and thus it terminates in line 3 without finding a wrong vehicle.

Thus, in all cases if a vehicle was found, it was indeed a vehicle.

From Lemma 1, we can conclude that the Theorem 1 holds. As the lemma shows that for one point p only a correct vehicle is found by Algorithm 1, we can extend this to all points since p was an arbitrary point and Algorithm 1 is the only part of our system that recognizes vehicles.
Thus, we have shown that if a vehicle is found by our algorithm, these points indeed form a vehicle.

4. EVALUATION

We evaluate our system in different scenarios in simulations as well as in experimental tests. In simulations, we simulate the camera and feed our system with black images with white points at the positions where the LED would be observed. This has the advantages over the real tests that we exactly know the ground truth. Hence, we have a perfect reference value. However, the real test provides more realistic results for the accuracy of our system.

Table 1. Accuracy in the simulation for all 175 scenarios. There is a small error in the simulation results, since calibration is used for comparison to experiments.

	Position Error in cm	Orientation Error in °
Mean	0.449673	0.256081
Max	0.569379	0.876293
Std dev	0.0405785	0.153931

4.1 Scenarios

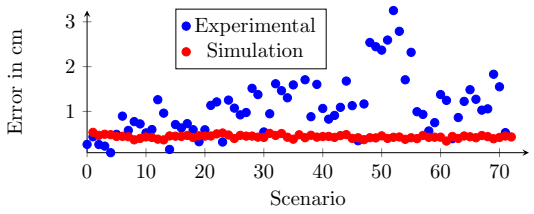
First, we have static scenarios. Here, we place the vehicle at nine different positions in the room. In each position, we place the vehicles in eight different orientations. We choose the angles $0, \pm\frac{\pi}{4}, \pm\frac{\pi}{4}, \pm\frac{3\pi}{4}$ and π. Furthermore, we evaluate dynamic scenarios. First, we force the vehicles drive along a straight line. Here, we test eight different lines in both directions, each with 0.5 m/s, 0.8 m/s and 1 m/s. In the experimental test, we additionally evaluate a vehicle driving along a circle with 0.5 m/s, 0.8 m/s, 1 m/s and 1.5 m/s in both directions. We evaluate a higher velocity in the circle experiments, as the distance is larger than in the straight line experiments. In the simulation, we test a vehicle driving an ellipse and afterwards an eight. Here, we simulate up to 20 vehicles. In a first step, we simulate them standing in small clusters and in a second step driving the ellipse and the recumbent eight. Moreover, we evaluate scenarios with two vehicles simulation right-hand or left-hand traffic, two parallel lines or passing a parking vehicle in different angles. Overall, we simulate 175 scenarios and test 128 in experiments.

4.2 Simulation Results

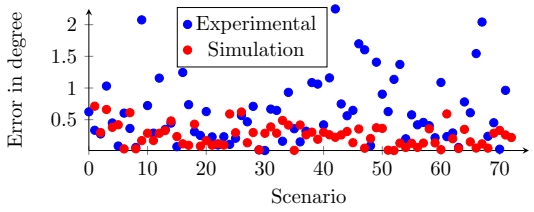
To create the images that are the input to our system, we compute the positions of each vehicle in the world depending on the path which they follow. Then, we calculate the LED positions from the vehicle positions. With the help of a camera calibration, those world point can be translated in the image plane. At those positions, we draw white blobs in a black image and provide it to the system. Hence, we only simulate the camera. For the identification LED, we compute whether the LED is on or off at a specific time point. Each image gets a time stamp.
To compute the accuracy of our system, we compare the expected position from which the LED positions are computed with the position received by our system. For this purpose, we compute the euclidean error. The results of all 175 scenarios as described above are summarized in Table 1.

4.3 Experimental Results

Beside the simulation, we evaluate different scenarios in experimental tests with a model-scale vehicle and a camera in a height of about 3 m. For the static scenarios, we place a vehicle at specific poses in the room. Then, we compare the pose gained by our system with the one that we intended to place. In contrast to the simulation, we cannot guarantee that the vehicle is placed exactly in the intended pose. Hence, an error may be introduced. In Fig. 9, the computations in the simulations are compared



(a) Comparison of the positioning error. The error is given in cm.



(b) Comparison of the orientation error. The error is given in degree.

Fig. 9. Comparison of the accuracy results of the simulations to the results of the experiments for all 72 static scenarios.

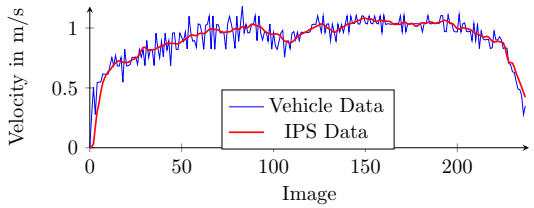


Fig. 10. Comparison between the velocity measured by the vehicle and the velocity detected by our system in the straight line experiment. Here, the distances of the image to its predecessor is illustrated

to the results in the experiments for all 72 static scenarios. In the dynamic scenarios the placement in reality gets even worse. Beside the inaccurate placement, we need to know the poses at the different time points depending on the velocity. For this, we use data from odometer and Inertial Measurement Unit (IMU) on the vehicle to compute the reference value. To force the vehicle driving a straight line, we use a rail. For the circle, we fix a midpoint and attach a cord to this midpoint as well as to the vehicle. If the vehicle drives straight ahead, it is pulled along a circle. In Table 2, the accuracy results for the static test as well as for the lines and the circles are shown. Fig. 10 compares the speed received from the odometer to the velocity computed from our IPS for the straight line experiment. Here, we can see that those are comparable. However, there are some differences, e.g. at the start of the experiment. The tires start spinning, but the vehicle is not moving. The speed measured by the odometer rises, but the IPS does not measure any movement. The errors for the static tests and the straight lines are comparable. However, the errors of the circles are worse. This is because the placement of the circles was the most difficult. If the length of the cord is measured only a few millimeters too short or too long, the error of the intended position to the actual position increases with covered distance. Furthermore since the vehicle drives straight ahead, the actual angle of the vehicle does not fit to the angle of the tangent to the circle.

Table 2. Accuracy of the dynamic scenarios evaluating experimental results compared to the results of the static scenarios evaluated experimentally.

	Static	Lines	Circles
Position [cm]			
Mean	1.119578024	1.41932	5.77446
Max	3.250616575	4.64408	19.4436
Std dev	0.663445809	0.804597	3.62245
Orientation [°]			
Mean	0.629930184	0.621901	2.10362
Max	2.248496847	10.4818	12.7505
Std dev	0.516798292	0.438002	1.73485

However for the straight lines, the results are comparable to the static tests. Here, we only have a single computation yielding a wrong result. This, we consider as an outlier. For all other computations, the maximal errors are in the range of the maximal errors of the static experiments.
Overall, the errors of the experimental test increase compared to the simulation. On the one hand, this is because the placement is more difficult as described above. On the other hand, the error of the calibration of the camera introduces an error in the positioning. Since no mapping between the world points of the camera and the world coordinate system used by the system is done in the simulation, there is no such calibration error.

4.4 Efficiency

The next property of our system that we evaluate is the efficiency. For this purpose, we determine the average and the worst case. In the worst case, we have all vehicles driving in a platoon. Then, we can determine in the 'Find Vehicle'—step only two vehicles at the same time namely the head and the tail of the platoon. Hence, we need many iterations to detect all vehicles. In contrast to this, the vehicles are clustered in small groups in the average case. In Fig. 11, the computations times for the different steps are shown for different number of vehicles in the worst case as well as in the average case. We can see that finding points in the image and computing the ID and the pose is constant for increasing number of vehicles as well as for the average and the worst case. Matching the vehicles is constant for the average case compared to the worst case, but increasing with the number of vehicles. This is because we need to match more vehicles. However, it has not influenced how those vehicles are positioned. The step with the most influence on the overall runtime is finding the vehicles. With increasing number of vehicles, the computation time increases. Furthermore in the worst case, the computations are higher than in the average case. This is because more vehicles need to be found with increasing number of vehicles and it is more difficult to find them in the worst case compared to the average case. Furthermore, we compare the runtimes with increasing number of vehicles for the average case and the worst case. The results can be seen in Fig. 12 and Fig. 13, respectively. In the average case, all computations terminate within the soft deadline of 20 ms. In the worst case, all computations for less than 14 vehicles also terminate within the deadline. For more than 14 vehicles, the deadline is exceeded.

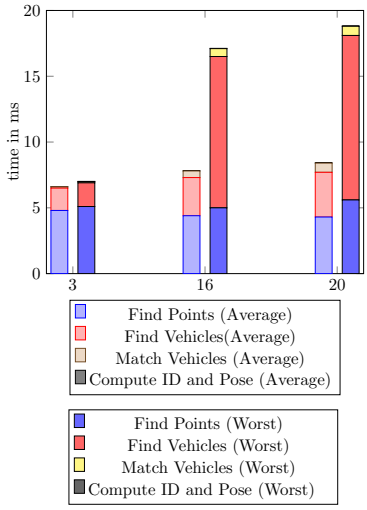


Fig. 11. Mean latencies of the single steps of the algorithm for 3, 16 and 20 vehicles in the average case and in the worst case.

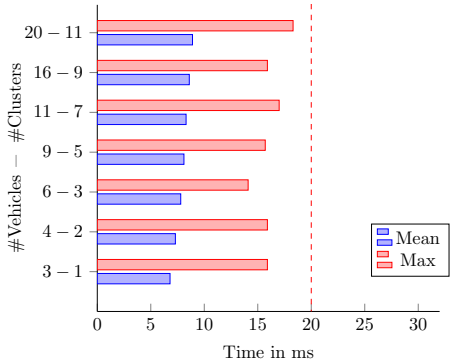


Fig. 12. Mean and max latencies of our algorithm in the average case (vehicles split in clusters) for different number of vehicles and clusters. The deadline gained from the vehicle cycle time is marked as dashed red line.

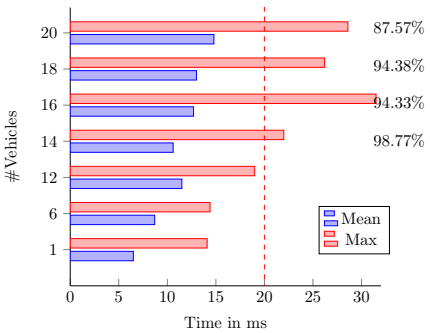


Fig. 13. Mean and max latencies of our algorithm in the worst case (driving in a platoon) for different number of vehicles. The deadline gained from the vehicle cycle time is marked as dashed red line. The percentages describe the amount of computations that terminate within the 20 ms.

However, for 20 vehicles 87.57 % of the computations terminate within the deadline. After computation, there is a delay of about 5 ms for communications of the poses to the vehicles.

Fig. 14. A model-scale vehicle on an intersection, as seen from the IPS. The colors are inverted due to print reasons.

4.5 Robustness

We also evaluated the robustness of our system against changes of the lighting conditions. Due to the low exposure time and the light LEDs, the vehicles are detected robustly even in light environments. Fig. 14 shows an image of a vehicle on an intersection as seen from the IPS. Please note that the colors are inverted. The white lines of the road are light grey, while the LED points of the vehicle are black. Hence, we can easily filter LED points of the vehicles using a threshold.

However, if an led is broken or occluded, it can not be detected. For robustness aginst such errors, we implemented a timeout on the vehicles. If the IPS is not able to detect a vehicle, the points are omitted, as stated in Section 2. Then, the vehicle does not receive any pose update for some time. If the vehicle does not receive an update for 100 ms, i.e. of 5 consequtive images, the vehicle stops. If the identification LED is occluded, the IPS may detect a vehicle with a wrong ID. Therefore, one vehicle will not receive pose updates and will stop.

5. CONCLUSION

We developed a new indoor positioning system which externally computes the position and the orientation of multiple vehicles. We evaluated our system with 20 vehicles. Our system is real-time capable with a soft deadline of 20 ms. Moreover, we reach an accuracy of around 1 cm and of about 0.6 ° in the mean. We robustly detect all vehicles in the plane even with changing lighting conditions and reflections. While evaluating our system, no identification errors occurred. Hence, our indoor positioning system can be used in applications with model-scale autonomous vehicles in which the knowledge of the position of each vehicle is crucial.

REFERENCES

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chawla, K., Robins, G., and Zhang, L. (2010). Object localization using RFID. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, 301–306. IEEE.

Diab, H., Abel, D., and Kowalewski, S. (2015). *Experimental validation and mathematical analysis of cooperative vehicles in a platoon*. Ph.D. thesis, Fachgruppe Informatik, RWTH Aachen University.

Faessler, M., Mueggler, E., Schwabe, K., and Scaramuzza, D. (2014). A monocular pose estimation system based on infrared LEDs. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 907–913.

Fukuju, Y., Minami, M., Morikawa, H., and Aoyama, T. (2003). DOLPHIN: An autonomous indoor positioning system in ubiquitous computing environment. In *Proceedings IEEE Workshop on Software Technologies for Future Embedded Systems. WSTFES 2003*, 53–56. IEEE.

Ghimire, B., Seitz, J., and Mutschler, C. (2018). Indoor Positioning Using OFDM-Based Visible Light Communication System. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–8. IEEE.

Hile, H. and Borriello, G. (2008). Positioning and orientation in indoor environments using camera phones. *IEEE Computer Graphics and Applications*, 28(4).

Ido, J., Shimizu, Y., Matsumoto, Y., and Ogasawara, T. (2009). Indoor navigation for a humanoid robot using a view sequence. *The International Journal of Robotics Research*, 28(2), 315–325.

Kloock, M., Maczijewski, J., Scheffe, P., Kampmann, A., Mokhtarian, A., Kowalewski, S., and Alrifaee, B. (2020). Cyber-Physical Mobility Lab An Open-Source Platform for Networked and Autonomous Vehicles. *arXiv preprint arXiv:2004.10063*.

Ladd, A.M., Bekris, K.E., Rudys, A., Kavraki, L.E., and Wallach, D.S. (2005). Robotics-based location sensing using wireless ethernet. *Wireless Networks*, 11(1-2), 189–204.

Lee, S. and Song, J.B. (2007). Mobile robot localization using infrared light reflecting landmarks. In *2007 International Conference on Control, Automation and Systems*, 674–677. IEEE.

Mautz, R. and Tilch, S. (2011). Optical indoor positioning systems. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*.

Neunert, M., Bloesch, M., and Buchli, J. (2016). An open source, fiducial based, visual-inertial motion capture system. In *Information Fusion (FUSION), 2016 19th International Conference on*, 1523–1530. IEEE.

Noonan, J., Rotstein, H., Geva, A., and Rivlin, E. (2018). Vision-Based Indoor Positioning of a Robotic Vehicle with a Floorplan. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–8. IEEE.

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 3400–3407. IEEE.

Rátosi, M. and Simon, G. (2018). Real-Time Localization and Tracking Using Visible Light Communication. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–8. IEEE.

Scheffe, P., Maczijewski, J., Kloock, M., Derks, A., Kowalewski, S., and Alrifaee, B. (2020). Networked and Autonomous Model-scale Vehicles for Experiments in Research and Education. 21st IFAC World Congress.

Yoshino, M., Haruyama, S., and Nakagawa, M. (2008). High-accuracy positioning system using visible LED lights and image sensor. In *2008 IEEE Radio and Wireless Symposium*, 439–442. IEEE.