Fast Change Detection for Camera-based Surveillance Systems

Matthias Michael[1] Christian Feist[2] Florian Schuller[2] Marc Tschentscher[1]

Abstract— **Many parking garages and open parking spaces today are already equipped with surveillance cameras to increase the security of pedestrians or to record potentially illegal actions. An additional use case for such multi-camera surveillance systems is the automatic extraction of 3D-positions of objects and pedestrians. The safety of autonomous vehicles could benefit from this information in cases where the on-board sensors might be unable to detect potentially dangerous situations due to occlusion. Since the used cameras are installed statically, change detection is often employed as the first operation and all subsequent processing steps rely on its quality. Different scenarios impose specific challenges to the respective algorithms. In this paper we present an efficient algorithm for change detection which is tailored to the difficulties arising in an indoor surveillance scenario and demonstrate its applicability by adapting an existing pipeline and improving overall performance.**

I. INTRODUCTION

Modern surveillance systems have the task of inferring information about the environment they are monitoring. These information can range from simply extracting the location of movable objects and persons in the environment up to more complex statements like the detection of potentially dangerous situations *e.g.*, in the context of autonomously driving vehicles. Often a multi-stage processing pipeline is necessary regardless of the system's ultimate purpose. The first step of such a pipeline often consists of identifying moving objects and areas in the camera images. Since surveillance cameras are usually installed permanently and statically, methods of change detection (also known as background/foreground segmentation or background subtraction) can be utilized for this task without additional knowledge about the objects in the scene.

The basic idea of change detection is to build a representation of the static elements in a scene. New camera images can then be compared to this representation which allows the detection of changes with respect to the static model. Areas that display changes are then denoted as foreground. This concept is visualized in Fig. 1.

Since only those areas that are identified as foreground are considered for further processing, change detection is a crucial step and the performance of the entire system depends on its quality. If the algorithm fails to identify moving areas, the rest of the pipeline might be missing decisive information. On the other hand, if too many pixels are segmented erroneously the rest of the system is getting irrelevant information as input. It would need to be capable of identifying that these areas should not be considered for generating statements about the scene which in turn might slow processing down significantly, simply because larger areas of the image have to be processed. Overall, a functioning change detection allows for a more simple and streamlined design of the pipeline.

Even though the identification of moving areas is a common task in computer vision and many algorithms attempt to solve it, no decisive state-of-the-art has been established. This is most likely due to the various problems like moving background, moving camera, and shadows, which are different in each specific scenario. It is very difficult for a single algorithm to address all possible aspects at the same time while maintaining a reasonable processing speed – especially when real-time requirements are present. Therefore a promising approach is to identify the challenges specific to the scenario at hand and choose or design a specialized algorithm.

In this paper we present a change detection algorithm that is tuned for indoor surveillance systems in an urban environment with a non-moving camera. In Sec. II the specific challenges of this scenario are identified and the performance of existing algorithms is examined. Section III describes the concept of the algorithm with Sec. IV investigating its performance on a popular benchmark. Besides standard performance measures like precision and recall, it is also important to evaluate performance with respect to an existing surveillance system. Therefore we adapt the system presented in [2] which is a multi-camera surveillance system designed for localizing arbitrary objects in an indoor parking garage. We substitute their change detection step with our method and adjust a few other details. The entire pipeline as well as our changes are described in Sec. V with a conclusion being given in Sec. VI.

II. RELATED WORK

There are a few general approaches that can be seen standard due to the fact that they are relatively simple to implement and produce acceptable results in a short amount

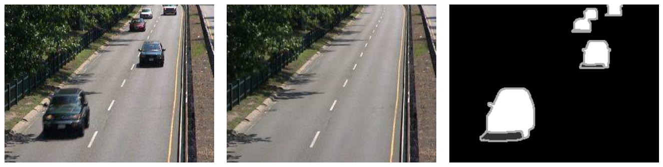

Fig. 1. *Left:* Single frame of a video sequence. *Center:* Representation of the static elements of the scene – here displayed by an image of the scene without moving objects. *Right:* Ground-truth mask of moving areas. White pixel denote movement while black pixel belong to non-moving background. (Images taken from [1]; Dataset *Baseline – Highway*.)

[1] University of Bochum, Institute for Neural Computation
{firstname.lastname}@ini.rub.de
[2] AUDI AG, Ingolstadt, Germany
{firstname.lastname}@audi.de

of time. Examples are calculating the mean [3] or median [4] of each pixel over a certain amount of time and comparing it to a new observation. Gaussian mixture models [5] represent another popular approach.

However, these algorithms are typically not able to handle more complex situations like dynamically changing backgrounds or shadows. Since our approach is aimed at urban surveillance scenarios with a focus on moving and parking cars, the most common challenges that occur in typical scenes are identified as listed below.

a) Global Changes in Lighting: Our algorithm will most likely be incorporated in a system installed in an indoor parking garage which is an environment with rather controlled lighting conditions. Yet it cannot be guaranteed that the illumination of the monitored scene will always be constant. Lights might be switched on and off depending on the time of day and opening and closing doors and gates might introduce additional variation to the lighting. If the parking garage is not build underground the illumination is also influenced by the weather and the time of day.

b) Headlights and Shadows: Many elements can lead to a local change in the image without being related to an actual moving object in the scene. Focussed lighting emitted by the headlights of a car or cast shadows are the most prominent example for this - especially since it is mandatory to switch the lights on when entering a parking garage. A rather severe example of such changes is given in Fig. 2. Here

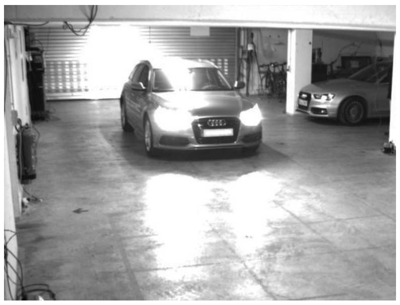

Fig. 2. The headlights of the car cast distinctive light cones onto the ground. Even though this leads to drastic changes in the image, a change detection algorithm should be able to detect, that this change does not correspond to an actual moving object.

an algorithm should be able to distinguish between pixels that are deviating from the background representation due to actual moving object and pixels that are just temporarily exposed to different illumination.

c) Long-term changes in the scene: This challenging situation is actually divided in two different types of changes. The first type occurs, when a moving object that has been identified as such stops its movement and becomes stationary. An example is a car that parks in a parking spot. In such a case the object should probably become part of the background representation. However, a decision has to be made regarding the time frame after which an object should become part of the background. If it only stops moving for a few seconds it should remain part of the foreground. The second type of change concerns the opposite sequence of action: An object that was stationary before and therefore part of the background suddenly starts to move and leaves its former position. An example would be a car that has been parked overnight and leaves its place in the morning. In this case the car itself would be (correctly) identified as foreground, however, the pixels at its former position would also display change since they do not correspond to the background model which still contains a representation of the parked car. Such a situation is shown in Fig. 3. In any case, the background representation of a suitable algorithm needs to be adaptable to permanent changes in the scene.

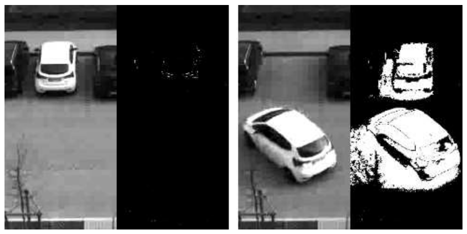

Fig. 3. *Left:* The initial situation. The white car is parked, part of the background representation and therefore not segmented as foreground. *Right:* After the car moves, it is still part of the background representation at its initial position. Therefore the pixels at its old location are classified as changed and, thus, designated as part of the foreground.

An overview as well as a benchmark of current algorithms for change detection is given in [1]. Prominent examples of advanced non-parametrized algorithms are ViBe (*Visual Background Extractor*) [6] and ViBe+ [7]. ViBe uses a sample-based background model and a voting scheme together with randomized updates of the background model to differentiate between foreground and background pixels. However, the voting is solely based on the absolute distance of gray values an therefore not able to account for shadows and global changes in lighting. ViBe+ mainly introduced methods of post processing to close holes in segmented regions as well as the option to detect blinking pixels and exclude them from the update of the background model which is not as relevant in the context of indoor parking garages.

The *Pixel-based Adaptive Segmenter* (PBAS) [8] improves on the ideas of ViBe by extending the distance measure to incorporate the local gradient. It also adjusts the update rate for the background model of each pixel individually which has not been proven useful in our target scenario.

SuBSENSE (*Self-balanced Sensitivity Segmenter*) [9] introduced the idea of using Local Binary Similarity Patterns for change detection to reduce the impact of shadows, however, the exact distance function incorporating gray values and LBSPs is not mentioned in the original work. While the update rate of each pixel is refined further, the problem of long term changes and the resulting wrong segmentation remains largely unregarded.

III. ALGORITHM

Our algorithm attempts to solve the problems identified in Sec. II by combining several aspects of PBAS [8] and SuBSENSE [9] with other techniques like an optional static background model. It consists of two distinct steps: The decision which pixels belong to the foreground and the update of the background model depending on the computed foreground mask. Both steps however require an existing background model which will be explained first.

A. Background Model

Our algorithm follows a sample-based approach. For each pixel there exists a database of observations that represent samples of the background. This is in contrast to parametrized approaches where the characteristics of a background pixel are modelled by a pre-defined probability distribution of which the parameters need to be estimated.

In our case the background model consists of a three-dimensional matrix of the size $W \times H \times (N_s + N_d)$ where W and H are the width and the height of the image, respectively. N_s is the size of the static part of the model for each pixel and N_d is the size of the dynamic part. This amounts to $(N_s + N_d)$ entries stored for each pixel.

A single entry consists of a gray value g_x^i and a *Local Binary Similarity Pattern* (LBSP) p_x^i where x is the position of the related pixel in the image and i is the index in the database for the pixel at position x.

The static part is completely optional – as mentioned before – and can be used to circumvent the problem of long-term object changes identified in Sec. II *e.g.*, segmenting parking spaces after the parked cars have left. It can be initialized by capturing images of the empty scene. This can be done right before processing starts, however most times it is easier to capture images off-line during the physical set-up of the system when the environment is not currently in use. The usefulness of the static model can be increased when images are captured in different lighting conditions.

The static background model is initialized based on the N_s previously captured images by computing an entry (g_x^i, p_x^i) for each pixel x and each image i. g_x^i is just the simple gray value and no further computation is necessary. The process to compute the LBSP p_x^i is illustrated in Fig. 4. A 5×5 mask is placed on every possible image location x and a vector with 16 binary elements is allocated for the result. According to the sequence shown in Fig. 4 the gray value at x is compared with its neighbours. If the difference between the values is larger than a threshold t_{lbsp} the element in the vector is set to 1, otherwise it is set to 0. The actual order of comparison is arbitrary, however it is important that the same order is applied every time the pattern is computed. In this case the neighbours are sorted according to their euclidean distance to x.

The dynamic model can be initialized in the same way using the first captured frame after processing starts. $\frac{N_d}{2}$ entries are computed from the pixel at location x while the rest is taken from random pixels in the neighbourhood of x to increase the robustness of the segmentation in the first few frames.

Large values of N_d allow for a background model with large variance since more samples can be stored but also lead to a longer processing time for each frame since more comparisons need to be made. In practice values between 10 and 20 seem sufficient.

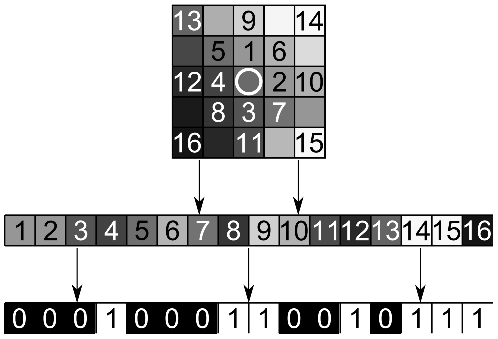

Fig. 4. Illustration of the calculation of Local Binary Similarity Patterns at 5×5 image location. *Top:* The access pattern in which sequence the pixels are compared with the center pixel. *Middle:* The pixel values stored in a linear array according to the sequence. *Bottom:* The final LBSP with 16 elements. If the difference between a gray value and the center pixel is larger than a certain threshold, the bit at this location in the array is set to 1, otherwise it is set to 0.

B. Segmentation

When a new frame is captured, a decision between background and foreground must be made for each pixel x. For this all gray values g_x and LBSPs p_x are computed. These values are compared with all entries in the background model (g_x^i, p_x^i). For g_x the absolute difference is applied while the hamming distance $h(\cdot,\cdot)$ is used for p_x. According to [8] a voting scheme can be applied to determine the actual affiliation of a pixel. For each $i \in [1, N_s + N_d]$ it is checked if

$$|g_x - g_x^i| < t_g \tag{1}$$

$$\vee \quad h(p_x, p_x^i) < t_p, \tag{2}$$

where t_g and t_p are configurable thresholds for the respective distances. It is important to notice that the values for t_p and t_{lbsp} can (and in most cases should) be different. If one of these conditions is met for a certain i, a voting counter c_x is increased. As soon as c_x exceeds a voting threshold t_v, the pixel at x is added to the background and processing can be stopped. It is also feasible to immediately stop processing with a positive result if an entry in the static model votes for the new sample since the static entries are definitive representations of the background. If the entire background model is traversed without reaching the threshold, x is seen as foreground since it does not match enough samples of previous backgrounds. Since the calculation of the LBSPs is rather costly in time, processing can be sped up by triggering its calculation only when the comparison of g_x yields a negative result.

The voting scheme is only one possibility to determine a pixel's affiliation to fore- or background. Another one – which is better suited for parallel processing hardware like GPUs – is a weighted distance to all samples. However, this method is not as robust when it comes to multi-modal backgrounds.

	t_g	t_p	t_{lbsp}	t_v	r	N_d
Value	15	3	10	2	92	10

TABLE I
THE CHOSEN PARAMETER VALUES FOR EVALUATION ON THE CHANGEDETECTION.NET BENCHMARK.

C. Background Update

As soon as long-term changes are expected in the scene that should not lead to a permanent classification as foreground, the background model needs to be adaptable. For sample-based approaches adaptation can be achieved by replacing samples in the database for each pixel. When a pixel is classified as background, its model is updated with probability $\frac{1}{r}$ where r is the configurable update rate. If an update is performed, the entry (g_x^i, p_x^i) at a random location i is replaced with the new entry (g_x, p_x). Due to the randomized update the chance of it still being present in the k-th update step is $\left(\frac{N_d-1}{N_d}\right)^k$.

In order to include objects in the background model after they have stopped moving, another update is necessary. If a regular update has taken place, a random pixel x' next to x is updated with the same probability $\frac{1}{r}$. The update can be made either with (g_x, p_x) or with $(g_{x'}, p_{x'})$. We argue to update location x' with a newly computed entry $(g_{x'}, p_{x'})$ at its location. Otherwise objects that differ greatly from their surrounding background would always remain in the foreground.

D. Post Processing

The LBSPs are able to detect if an image region is just a color-scaled version of another region due to lighting, since the ratios between neighbouring pixels remain similar. However, depending on the threshold t_{lbsp} they are sensitive to stronger noise to a point where a pixel is erroneously classified as foreground if the gray values differ too much. These outliers can be removed by an opening of the segmentation which, however, also might remove some correctly segmented pixels. In general, an opening leads to an improved precision while decreasing the recall.

Another aspect is, that a pixel is only classified as foreground if it differs in color and structure from the background. This means that areas with different color but nearly identical structures (*e.g.*, the surface of a black car in front of a light gray wall) are classified as background and only the areas at the edges of objects are assigned to the foreground. In the intended use case this does not negatively impact performance. Since the segmentation primarily contains edges and contours, bounding boxes of the corresponding objects can easily be generated.

In cases where a full segmentation of a moving object is required, a clustering can be used to identify segmented regions belonging to a single object. Then it is possible to find and fill small non-segmented areas inside of the convex hull of the cluster.

E. Extensibility

The presented algorithm offers great extensibility. Additional features can be added to the decision function. Currently only gray-scale images are processed, but color information can be added as an additional feature. Haar-Features or other image descriptors are also conceivable.

The decision function itself is also variable. The or-function combined with the voting scheme held optimal results for our scenarios. Other distance measures between feature vector and a database of feature vectors are also applicable.

IV. EXPERIMENTS

For the evaluation of our algorithm we chose the benchmark presented in [1]. It provides 53 labelled sequences in 11 categories highlighting different challenges for change detection algorithms like shadows and moving cameras. We chose evaluation categories according to the design goals identified in Sec. II to determine whether these goals have been met by the algorithm. The chosen categories are *badWeather*, *baseline*, *intermittentObjectMotion*, *lowFramerate*, and *shadow*. The other were omitted since they contain challenges the algorithm was not designed to solve.

We executed the algorithm as described in Sec. III with the parameters displayed in Tab. I on each sequence in the chosen categories. An opening was employed as post processing to remove outliers that occurred due to camera noise and compression artefacts in the input images. The resulting mean values for precision and recall for each category are shown in Tab. II. Overall a mean precision of 0.90 could be achieved with 10 out of 24 sequences yielding a precision of over 0.95. Since we aimed at minimizing the false positive rate, the results for the recall are not as good with a mean value of 0.25. This outcome can be explained by the distance function incorporating LBSPs. An input pixel needs to differ in both gray value and local structure from the samples in the background model to be regarded as foreground. Therefore a white pixel on the uniform surface of a car might be classified as background if the real background consists of darker pixels on the uniform surface of the road.

This behaviour makes it possible to exclude shadows and illuminated regions from the segmentation. However, in most cases different gray values and colors can be found at object boarders – either of the object in the foreground or objects in the background – which is sufficient to identify the moving object itself. These structural differences also occur at hard edges of shadows, but the center region as well as fading shadows can successfully be removed from the segmentation images. An example is given in Fig. 6. On an Intel Core i7-6820HQ CPU with 2.7 GHz processing of an 1360×1024 video requires 30 to 50 ms per frame depending on the amount of segmented pixels and how many LBSPs need to be calculated. A 680×512 video requires 8 to 13 ms per frame. An equivalent implementation on an NVIDIA Quadro M1000M GPU (using a weighted sum instead of a voting scheme as distance function) requires only 14 ms for 1360×1024 pixels.

	Precision	Recall
badWeather	0.92	0.15
baseline	0.96	0.36
intermittentOM	0.95	0.17
lowFramerate	0.74	0.25
shadow	0.93	0.30

TABLE II
MEAN PRECISION AND RECALL FOR ALL SEQUENCES IN EACH CATEGORY.

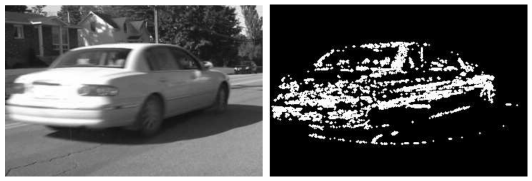

Fig. 6. *Left:* Frame 141 of the sequence *bungalow* in the category *shadow*. *Right:* Corresponding segmentation image produced by our algorithm. Segmentation is mainly present at the edges of the car and where the structure of the car differs from the structure of the background.

V. PIPELINE

As mentioned in Sec. I we adapted the system of [2] by exchanging their simpler method of change detection by our algorithm. In the following we give a quick overview of the entire pipeline that estimates the pose of objects in a parking garage. A diagram of all components of the system can be found in Fig. 5.

The first step is gathering input images from an arbitrary number of cameras which are calibrated and have a known position and orientation. The following processing steps are executed for each camera separately. Our change detection algorithm including post processing is executed to gain a mask of moving objects in the foreground. On the camera image, optical flow is calculated. Here the mask is used (with a certain amount of padding of the identified foreground regions) to restrict the search-area for significant points for flow calculation. This leads to a speed-up of the calculation and lowers the amount of irrelevant or even wrong flow vectors.

Foreground regions with associated flow are considered for object tracking in the image in the next step. Here the image is divided into rectangular blocks of 30×30 pixels. Only those blocks which contain a certain amount of foreground pixels are considered for tracking. For each block the mean orientation of its associated flow vectors is calculated. Since some blocks (*e.g.*, on uniformly coloured surfaces) might not contain enough significant points to calculate any flow, the flow direction is iteratively propagated from blocks with flow to neighbouring blocks without flow. Based on the image location and the flow, blocks are clustered together to form object hypotheses. These hypotheses are classified with an SVM using orientation histograms as features and are tracked with an Alpha-Beta filter.

The reasoning behind this procedure is this: Using the flow as additional information for the clustering allows to differentiate between a car and a pedestrian moving in front or behind the car in a different direction or at a different pace. The classification uses orientation histograms since it can be expected that a lot more horizontal edges are present in a camera image of a car, while a pedestrian most likely will display a larger amount of vertical edges. To exclude the background from this calculation, the foreground mask can also be used to limit the area in which the orientation features are extracted.

After object hypotheses are consolidated, view rays are created from the center of the camera through the center of each active block inside of a tracked image region. Based on the movement direction and appearance of tracked regions in the image of each camera, view ray clusters are identified that most likely belong to the same object. All view rays of clusters that are associated to the same object are intersected. The intersection points (or the points that are closest to both view rays) for a single object are transformed into a 2D normal distribution which gives information about the location and shape of the object. The normal distributions are also tracked by an Alpha-Beta filter and additional measures are taken to prevent objects from decaying into multiple tracks *i.e.*, tracks that are displaying identical movement for a certain amount of time are merged into a single track.

A Lidar system is used as a ground truth to provide more accurate measures of the objects' locations which can be used as a reference to determine the quality of the results of the entire system as well as for comparison with the quality of the previous system. Evaluation of the enhanced system is performed for a vehicle and a pedestrian where first each object is moving solely in a scene and for a scene in which both are acting together. The resulting error graphs can be found in Fig. 7 for the pedestrian and in Fig. 8 for the vehicle.

It is noteworthy that evaluation can only be performed when positions estimates from both, the ground truth and our system are provided. The overall results for our system for each of the scenes can be found in Tab. III.

However, the change detection algorithm as well as the subsequent processing pipeline has been changed significantly during development which complicates the evaluation

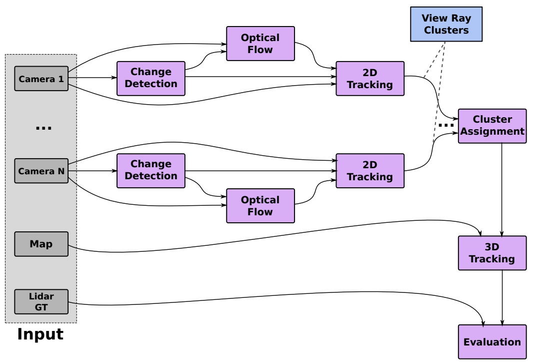

Fig. 5. The general structure of the processing pipeline to generate object hypotheses in the environment. For each camera change detection, flow calculation, 2D tracking, and view ray generation is performed separately. From this data 3D object hypotheses are generated and tracked over time. Evaluation can be performed with respect to a Lidar system which provides almost ground-truth data.

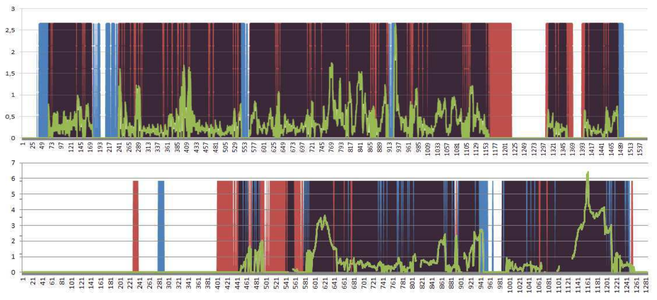

Fig. 7. *Top:* Evaluation for a single pedestrian moving in the scene. *Bottom:* Evaluation for a pedestrian moving together in the scene with a car. A blue background means that only the Lidar system provides position estimates while for a red background only our system has an estimate. The green line shows the error of the estimate relative to the Lidar system in meters.

of the performance regarding earlier instances of the system. The new approach can handle occlusions, sudden illumination changes and can be calculated in less than half the time as before. The trade-off is a slightly worse positioning error as in the previous work [2]. Considering the localization of pedestrians, there is a difference of 0.30 m to 0.60 m.

For the scenario in which a car and a pedestrian are moving simultaneously in a scene there can be seen error spikes where the position estimate is off by several meters. These spikes are the results of a wrong matching between object hypotheses between multiple cameras. This is also not taken account for in [2] since evaluation is only done for the movement of one object at a time.

Nevertheless, our aim is to detect potential dangerous situations like people moving in front of the car since the car has a very high positioning accuracy itself. Therefore, a safety margin of about 1.5 m is employed around the detected person or object, to guarantee that the car can stop with enough clearance. In summary, we increased efficiency and reached a better performance and usability for our intended scenario.

VI. CONCLUSION

In this paper we developed a general purpose algorithm for change detection which focuses on a low false positive segmentation rate and a high precision. It is able to handle global as well as local illumination changes and provides an adaptable, sample-based background model. Due to the incorporation of an optional static background model the algorithm is also able to handle long-term changes in a scene without wrong segmentation. It is easy to implement and provides real-time capability on the GPU as well as for smaller image sizes (680×512) on the CPU and has been successfully deployed in an existing image processing

	Pedestrian	Vehicle
Separate	0.78 m	1.19 m
Simultaneous	1.07 m	0.97 m

TABLE III

THE MEAN POSITION ERROR TO THE LIDAR SYSTEM.

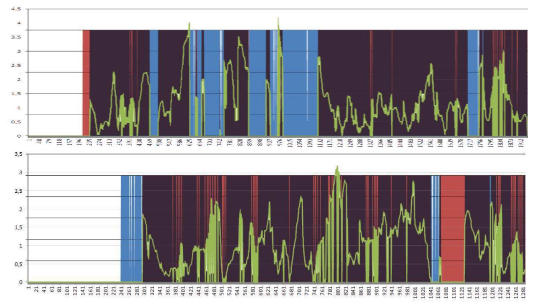

Fig. 8. *Top:* Evaluation for a single vehicle moving in the scene. *Bottom:* Evaluation for a vehicle moving together in the scene with a pedestrian. The same color-coding is used as in Fig. 7.

pipeline for multi-camera object localization.

Improvements can be made regarding the handling of shadows with hard edges where a segmentation cannot be fully suppressed. This is most likely not possible based on local image features but would require a high level representation of illuminated image regions. The sample-based background model also allows to change the update rate for each pixel individually. While not necessary for the intended scenario this can be used to account for strongly dynamic backgrounds.

However, most improvements regarding segmentation quality would negatively impact the required runtime of the algorithm. In its current state the algorithm can already easily be integrated in any image processing pipeline that requires fast and precise change detection.

REFERENCES

[1] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, “Changedetection.net: A new change detection benchmark dataset,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 1–8.

[2] A. Ibisch, S. Houben, M. Michael, R. Kesten, and F. Schuller, “Arbitrary object localization and tracking via multiple-camera surveillance system embedded in a parking garage,” in *Proceedings of the Electronic Imaging Conference*, 2015, pp. 9407–9412.

[3] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Real-time tracking of the human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, 1997.

[4] B. Lo and S. Velastin, “Automatic congestion detection system for underground platforms,” in *Proceedings of the International Symposium on Intelligent Multimedia, Video and Speech Processing*, 2001, pp. 158–161.

[5] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 1999, pp. 246–252.

[6] O. Barnich and M. Van Droogenbroeck, “Vibe: A powerful random technique to estimate the background in video sequences,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 945–948.

[7] M. Van Droogenbroeck and O. Paquot, “Background subtraction: Experiments and improvements for vibe,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 32–37.

[8] M. Hofmann, P. Tiefenbacher, and G. Rigoll, “Background segmentation with feedback: The pixel-based adaptive segmenter,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 38–43.

[9] P.-L. St-Charles and G.-A. Bilodeau, “Flexible background subtraction with self-balanced local sensitivity,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 414–419.