A Simulated Car-Park Environment for the Evaluation of Video-Based On-Site Parking Guidance Systems

Marc Tschentscher[1], Ben Pruß[1], Daniela Horn[1]

Abstract— **Developing image-processing algorithms based on machine learning is a challenging problem concerning the huge amount of thoroughly annotated data needed. The internet provides many already tagged images for basic classification problems like vegetables or different cars, but not for more narrow problems. In order to extend and evaluate the previously presented parking guidance system from our previous work, in this paper, we propose a simulation system based on Unreal Engine 4. We developed an artificial camera which implements all features of a real camera, *e.g.*, lens distortion, motion blur etc. to export video data from the simulated environment. This data is then compared to real-world video footage by using our classification module that distinguishes occupied and free parking lots. We reached a classification rate between 92.28 % and 99.72 % depending on the parking rows' distance using *DoG*-features and a support vector machine.**

I. INTRODUCTION

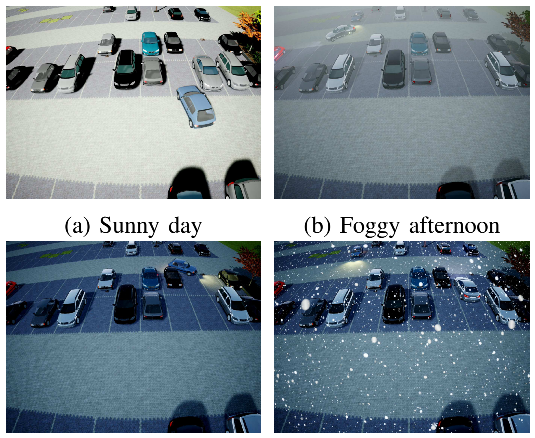
These days, developing image-processing algorithms based on machine learning methods is a problematic undertaking due to the huge amount of data needed [1]. Many images can be found on the internet to train classifiers if they are already tagged (*e.g.*, vegetables, different cars etc.). For a more distinct problem, as, in the present case, classifying occupied and vacant parking lots, it is extremely difficult to find representative images. In the past, hours of recording sessions with a huge amount of video data and manual labeling were required to extract a good data set.

Using a simulated environment can overcome these problems. It is possible to create video data reproducing natural scenes (*e.g.*, huge parking areas) and use them for the training and / or testing phase of image processing algorithms.

In this paper, we propose a simulation system based on Unreal Engine 4. We established this system to generate training and test data which are close to reality. This data will be used to further improve and evaluate our video-based parking classification and routing system [2], [3], [4].

Generating data for different weather and lighting conditions and using them for training and testing improves the robustness of our current system. We are able to simulate various scenarios of parking and leaving vehicles (*cf.* Fig. 1) to test our algorithms in every imaginable situation without having to record a large amount of sequences, or having to wait for special scenarios or environmental conditions in the real world. Furthermore, we can directly extract the ground truth data *e.g.*, for the occupancy of the observed parking lot and compare it with the results of the classification algorithm.

[1] University of Bochum, Institute for Neural Computation
`marc-philipp.tschentscher at ini.rub.de`



(a) Sunny day (b) Foggy afternoon

(c) Rainy afternoon (d) Afternoon snowfall

Fig. 1: The simulated environment showing different lighting and weather conditions

The challenge for the simulation is to be as close to reality as possible. This is a requirement for the development of the video-based parking guidance system based on the simulation and its use in real-world scenarios. Therefor we recorded a few scenes on a real parking lot, which we rebuilt in the simulation. We compare the results of our algorithm (*e.g.*, lot-occupancy-classification) of the real-world scenario with the artificial one. This leads to a decision on how exact the simulation is and whether it is realistic enough to develop and test new algorithms.

This paper is organized as follows: We first give an overview of what is done in the field so far concerning artificial cameras and simulation systems in general (*cf.* Sec. II). Afterwards we describe our proposed system's setup (*cf.* Sec. III). The simulated environment is explained in Sec. III-A, while Section III-B focuses on the artificial camera which we implemented to extract video material from the virtual environment. Experiments, which were executed to evaluate the system, can be found in Sec. IV. The paper closes with a conclusion and outlook in Sec. V.

II. RELATED WORK

This section covers related work in this field. It can be divided into two parts:

On the one hand, the use of virtual environments within the field of traffic modeling and their purposes have to be considered. Simulation tools like *AnyLogic* are commonly used to model high traffic situations or parking scenarios in order to find and optimize bottlenecks [5], [6]. For this purpose, AnyLogic also offers a special *Road Traffic Library*, which comprises a number of traffic-related assets that are useful for a number of traffic scenarios. The library can be customized ad lib to fit personal requirements.

Companies like the PTV Group build virtual models using their own tool, *PTVissim*, to demonstrate simulations of actually implemented parking guidance systems, *e.g.*, at airports, or to model and optimize high traffic situations at roundabouts or crossroads [7]. Especially in industry, simulations are used to visualize products in order to make them more comprehensible for potential clients.

It is conspicuous that these simulations show traffic scenarios in a highly stylized, simplified form. They focus on modeling predefined behavior rather than imitating reality in looks or scenarios. Human shortcomings, such as parking in an oblique angle using two parking spaces at a time, are not modeled at all. Also realistic rendering techniques still seem to be mainly reserved to video game environments.

On the other hand, the process of generating useful video footage from a virtual environment is equally important when constructing a system for synthetic data generation. There are a number of approaches covering virtual camera models based on different requirements.

Von Neumann-Cosel et al. introduce an idealized camera model for training a traffic lane tracing algorithm [8], which does not cover significant properties of real cameras (*e.g.*, depth of field, motion blur and effective exposure). Williams and Lee simulate a camera merely defined through its pose and focal length for the reconstruction of 3D objects from images [9].

Considering the combination of detailed simulated environments and virtual camera usage, Noth proposes a complex graphic simulation environment with integrated virtual cameras, focussing on the production of realistic camera images [10]. While his system is capable of generating different advanced effects like realistic shadow rendering, depending on manually defined light spots, some key elements of real cameras which are essential for generating highly realistic images, *e.g.*, image noise or distortions, are missing.

III. SYSTEM SETUP

This section gives a complete overview of our system, which consists of a simulated environment (*cf.* Sec. III-A) and an artificial camera (*cf.* Sec. III-B).

Different to real-world data, the simulated environment can comply with various requirements to fulfill a given task. In this use case, a lively parking area is portrayed, containing several parking rows and many cars driving around and parking on the lots (*cf.* Sec. III-A). There are various possibilities of bringing the environment to life, either by driving a single car across the given area, by adding a second player with a separate car, or by including automatically driven cars, which follow customized paths around the car park.

The artificial camera is implemented as an actor within the simulated environment. Thus, a variable number of artificial cameras can be placed at the same time to observe a simulated scene, *e.g.*, to cover a broader field of view, or to restore 3D information of the scene. The camera simulates several physical features of a real camera, such as lens distortion, different aperture sizes etc. The images are recorded as either color or grayscale images and are saved to the shared memory block of the PC's working memory. We implemented a module for the used image-processing framework, which reads the image from shared memory and makes it available for further processing (*cf.* Fig. 2). This step

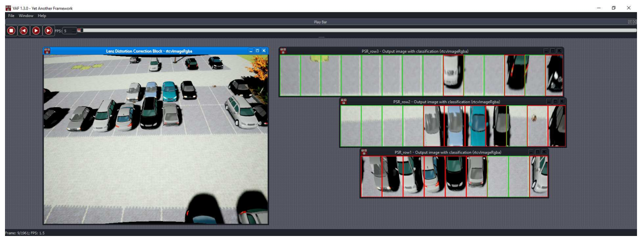


Fig. 2: The software framework YAF, which is used for image processing. *Left:* (Artificial) camera image with lens undistortion applied, *Right:* Visualization of classification results for first, second and third row (bottom-up)

is not presented in this paper as it is a special requirement for the given use-case. It is imaginable to implement any other tool to read the data block from the shared memory (*e.g.*, for another framework or a standalone application).

For the experiments (*cf.* Sec. IV) we evaluated the previously published classification module [2], [3] on a real and a simulated sequence, for which the simulated sequence is exactly the same as the real sequences concerning the cars and the maneuvers performed.

A. Simulated Environment

As one of our main requirements for the simulated environment was the realistic appearance, we chose Unreal Engine 4 (UE4) for implementation, one of the most widely used game engines. This was especially necessary as one of our main goals was to model a real-world location for a direct comparison of both reality and simulation (*cf.* Sec. IV-A).

We adopted a combination of UE4's newly introduced Blueprints Visual Scripting tool and the engine's native C++ code to build both visual assets and functionality within the simulated scene. Blueprints were especially used to add functionality to parking spaces in order to obtain and store information on the occupancy status of a given spot.

In order to create a more realistic feel, objects within the modeled scene were coated with materials which had been created using physically-based rendering (PBR). This approach refers to a number of concepts rather than strict rules which describe realistic shading and lighting models by using measured surface values to mimic the behavior of real-world materials and add a form of standardization to a rather artistic work [11], [12].

Within the various possibilities to create realistic materials, we have chosen a four-layer approach. These layers are: *albedo*, *gloss*, *specular*, and *normal*. Required values for the creation of the first three layers were taken from an open library, which offers three principal values: albedo, microsurface, and reflectivity, for a small number of material categories (*cf.* [11]). However, libraries which are liable to costs offer finer distinctions of materials and value types in order to provide photo-realistic quality.

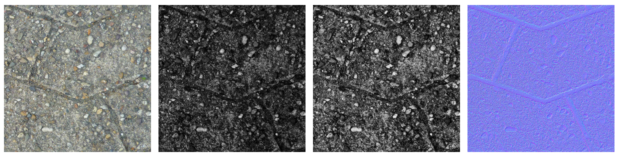


Fig. 3: Material layers used in PBR approach

The albedo layer (see Fig. 3a) represents all color information of a material. It is also referred to as diffusion. From an optics point of view, a natural object receives its color by absorbing and scattering light, depending on its wavelength. In this context, albedo describes the fractions of light which scatter back from a surface.

Gloss, also known as microsurface, handles information on a material's roughness. In this grayscale image (*cf.* Fig. 3b) lighter shading defines a rather smooth surface, while darker pixels represent a rougher surface. It is noteworthy that the actual roughness of a material is not the gloss map itself but rather the inverted map. This is due to the fact that gloss and roughness complement each other in the real world, as well.

A material's reflectivity is set by its specular map (*cf.* Fig. 3c). Analogous to the gloss layer, lighter colors hint at a higher degree of reflectivity than darker ones. The fourth material layer (*cf.* Fig. 3d) is a normal map. This layer stores height information of the material as RGB values. Its purpose is to add the impression of three-dimensionality to an actually flat surface, thus creating the illusion of more detailed objects.

While each of the layers stores separate aspects of information on the resulting material, the layers also affect each other when combined to a material. The mutual exclusiveness of diffusion and reflection in natural lighting is reproduced within the physically-based rendering approach by the term energy conservation.

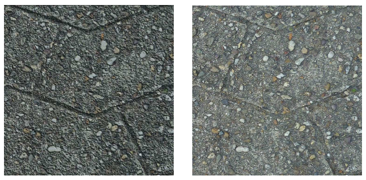


Fig. 4: Comparison of non-PBR and PBR material in UE4

A direct comparison of a non-PBR material with a PBR material (*cf.* Fig. 4) shows a clear difference in quality and closeness to reality. As the left image merely consists of the same original photograph and a normal map, it cannot depict the difference between the rather smooth pebbles and the rough main substance of the paving stone. The prominent dark lines result from shading problems within Unreal Engine due to missing information. Microscopic unevenness are excessively shaded, resulting in a rather harsh, dark resemblance. The PBR material on the other hand shows a much smoother appearance of the material. While depth information are preserved, they do not appear as promiment as with the simple approach. When viewing this material from different angles in UE4, differences in gloss and specular become clear, and show a quite natural reflective behavior. Although the PBR approach requires more time, both for creation and computation, the improvement in looks justifies these concessions.

A benefit of using virtual environments for data generation is the possibility to extract ground truth data on the fly. Therefor each parking space was equipped with a trigger volume which notices objects entering or leaving the respective parking space. The parking space id, its current status, *i.e.*, available or occupied, and the timestamp of a change are then written to a log file. As a recorded video sequence within the virtual environment saves a timestamp for each recorded frame as well, the gathered data can later be compared to the video data in order to create ground truth data for a video sequence.

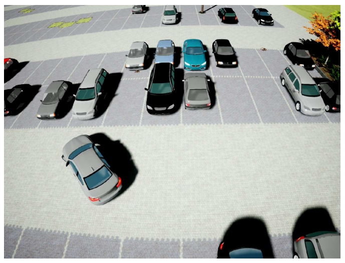


Fig. 5: Example image of the developed car park

While many of the cars in Fig. 5 have been added as static objects in order to fill the scene, one car can be steered freely across the parking area. However, as a more lively parking situation is more interesting for the evaluation of the resulting material, two different ways for additional animation within the car park level have been created. A basic AI has been implemented to add one or more moving vehicles to the scene. The chosen amount of vehicles can move around the area following predefined paths which can be set with the help of waypoint markers in the level. For a less deterministic behavior, a local multi-player setup has been implemented as well. This allows another person to drive a second car within the area at the same time with the help of a classical split-screen mode. A further addition of a network-based multiplayer mode could be interesting for other scenarios in which more than two user-controlled vehicles seem advantageous.

A huge benefit of simulated environments is the full control over environmental parameters. A change of weather, different daytime lighting conditions or obstructed parking spaces or lanes due to construction works – every situation that is required for relevant data material can be set in the environment. We've successfully used exactly this advantage to create video sequences with various weather conditions for further evaluation of our classifier (*cf.* Sec. IV-B and IV-C).

B. Artificial Camera

In order to generate highly realistic camera images from the simulated scene, it is not only necessary to extract certain parts of the environment, but also to translate physical restrictions to our virtual camera actor which bring on limitations to the images produced by cameras in the real world. These restrictions are *e.g.*, depth of field, image noise, and motion blur. Therefore we implemented a realistic camera model which modifies the image generation process within the Unreal Engine interface and performs optional final post processing operations on the images generated by Unreal Engine's API. Our model is based on the same parameters which apply to modern camera systems and consists of transform functions between the parameter values and the Unreal Engine rendering options for each of the parameters: focal length, aperture opening, film speed, exposure time and focus distance.

Using the aperture opening number k, film speed S, and exposure time t_{exp}, a scalar exposure value EV [13] representing the change of brightness in the image caused by the effects of these parameters can be calculated as shown in Equation 1.

$$EV = log_2(\frac{1\,[sec]}{t_{exp}}) + log_2(\frac{100}{S}) + log_2(k^2) \tag{1}$$

The total brightness of the scene captured by the virtual camera sensor is reduced by a factor of 2^{EV} which corresponds quite accurately to the behavior of real-world cameras (*e.g.*, image brightness doubles when doubling the exposure time).

Real camera systems offer different methods to automatically determine the exposure parameters in order to guarantee correct lighting conditions in the generated image. The most common functions measure the amount of incoming light to establish an estimate for the optimal value of the exposure time so that the image is neither under- nor overexposed. The measurement is either performed in the concrete scene (*e.g.*, using a light meter) or based on the images generated with a fixed set of parameters.

We are using the second approach by retrieving information about the exposure from the grayscale histograms of the different color channels with the following criterion for correctly exposed images: In each histogram the lowest gray value which occurs at least once shall be as far away from position 0 as the highest gray value occurring at least once is away from position 255. The difference must not be more than 20. If an image is not matching the criterion, the next image will be generated using either an increased or decreased exposure time, depending on whether the current image was too bright or too dark.

The image area supposed to be sharp is determined by two distances d_{near} and d_{far} which both define a curved plane relative to the pivot point of the camera actor [14]. Objects which are located in between these planes are expected to appear satisfactorily sharp. The distances depend on the focal length f, the chosen focus distance d_f, and on the hyperfocal distance d_h, which itself is dependent on the focal length, aperture opening number k, and on the maximum permissible circle of confusion diameter c (*cf.* Eq. 2).

$$\begin{aligned} d_h &= \frac{f^2}{k * c} + f \\ d_{near} &= \frac{d_f * (d_h - f)}{d_h + d_f - 2f} \\ d_{far} &= \frac{d_f * (d_h - f)}{d_h - d_f} \end{aligned} \tag{2}$$

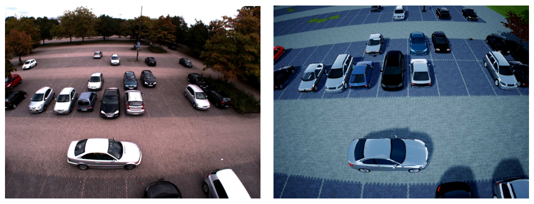
Using these distances, the engine is parameterized with the focus distance and two transition regions before and beyond the focus distance to define the area which will be rendered sharply in contrast to the remaining area, which is blurred.

Images generated by real cameras suffer from various sources of error such as sensor related image noise, radial distortions caused by wide-angle lenses, or motion blur. To reproduce these effects in the virtual images, we implemented post processing steps which we apply to the output image generated by the Unreal Engine functions.

We used a simple symmetric model to describe a radial distortion transformation which is only dependent on the focal length f of the virtual camera and a hardware related correction factor b_d which we determined empirically for our reference camera system. This factor is used to align the distortion strength s_d gradation of the virtual camera with a certain system in the real world by giving a starting point for it at a fixed focal length of 55 mm.

$$s_d = b_d * \frac{55\,[mm]}{f} \tag{3}$$

Eq. 3 describes the relation between the focus length and the total strength of the distortion effect with regard to the correction factor, which is similar to the gradation of the distortion strength in real-world cameras, where a halved focal length results in a doubled strength of distortion. The described transformation can easily be undone preserving most of the relevant image information, by using the method proposed by Zhang [15].



(a) Original camera image (b) Virtual camera image

Fig. 6: Comparison of distorted camera images generated *a* by a real-world camera system and *b* by the proposed virtual camera system

A direct comparison of the radial distortions caused by using wide-angle lenses in real-world camera systems and the distortion effects generated by our virtual system is depicted in Fig. 6. Both system use the same set of parameters.

The artificial image noise we used to model the effects of the film speed on the resulting image is generated by picking values from a random variable with normal distribution of mean 0 and variance σ^2 (*cf.* Eq. 4) with correction factor b_g

$$\sigma^2 = b_g * \frac{S}{100} \tag{4}$$

depending on the hardware which is supposed to be virtually replicated. For each color channel of each image pixel a single value is picked and added to the original scalar gray value.

The engine is producing snapshots of the scene with no temporal extension. Thus the generated images cannot contain the motion blur effect which is caused by fast moving objects within the covered image area during the exposure time. Because exact image-based calculations of motion blur effects rely on meta information about the objects within the simulation context (such as complete object segmentation, position, moving direction, and movement speed) which is unavailable or at least hard to calculate, we implemented a concept that approximates this effect by taking multiple snapshots during a period of exposure and calculating an average image based on these snapshots.

IV. EXPERIMENTS

In this section the experiments to evaluate the proposed system are described. We used the same setting of the car park as in [3] (3 parking rows containing 36 parking lots in total). The following tables show the accuracy (in percent) for each row (from near to far) and each feature (color $\hat{=}$ best performing color feature in *HSV* color space, gray $\hat{=}$ best performing *Difference of Gaussian (DoG)*-feature) classifier presented in [3]. The best accuracy for each row and each scenario is highlighted. For a first test we used the third sequence from [3] and reconstructed the given setting in the simulated environment (*cf.* Sec.IV-A) to estimate the quality of the simulation. Section IV-B provides information of experiments with different lighting conditions (*e.g.*, sunny, cloudy and foggy) to show the robustness of the system. Finally, it is possible to simulate precipitation, *e.g.*, rain and snow. The results are presented in Section IV-C.

A. Reconstructed Setting

This setting is a reconstruction of a previously used sequence in [3] for evaluating the parking space classification algorithm. Figure 6 juxtaposes a real-world image to an image of the reconstructed scenario. The classification result including a comparison to the previously calculated real-world accuracy is shown in Table I. It is noteworthy that the algorithm reached an accuracy on the reconstructed scenario which is comparable to the real-world scenario. Concerning row 2, the accuracy drops significantly in this scenario using both, the color features and the *DoG*-features.

	reconstruction		real world	
	color	**gray**	**color**	**gray**
row 1	93.68	99.53	96.68	99.96
row 2	72.73	82.78	98.95	96.99
row 3	100.00	100.00	91.83	92.33

TABLE I: Accuracy for the reconstructed scene compared to the real world

	sunny		cloudy		foggy	
	color	**gray**	**color**	**gray**	**color**	**gray**
row 1	99.72	99.78	98.25	99.77	80.31	99.70
row 2	98.48	93.17	77.20	88.74	56.42	97.18
row 3	92.28	99.82	98.77	99.27	4.35	58.44

TABLE II: Accuracy for scenarios with adjusted lighting conditions (*sunny, cloudy, foggy*)

B. Adjusted Lighting Conditions

In the next phase of experiments we focused on the adjustment of the lighting conditions. We tested three different states: Sunny, cloudy and foggy. The results are presented in the following.

1) Sunny: This sequence represents a typical summer's day at noontide. Figure 7a shows an image which is included in the sequence. The results gained from the classification algorithms are shown in table II.

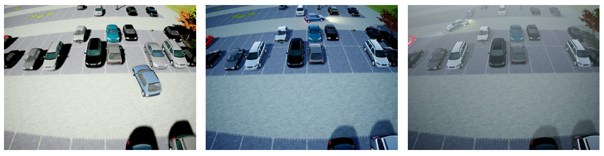

(a) sunny (b) cloudy (c) foggy

Fig. 7: Images of the three different lighting states

Similar to the previous experiment, there is a drop in the classification result in row 2. However, this holds only for the gray features. In contrast, the accuracy slightly drops in row 3 using color features but provides good results for row 2.

2) Cloudy: This scenario shows a cloudy day in the afternoon. The lighting is darker than in the previous scenario so the driving cars turned their lights on (*cf.* Fig. 7b). Table II presents the achieved results.

Regarding the second parking row, the typical accuracy drop can be observed.

3) Foggy: An example of this sequence is visualized in Figure 7c. The scenario contains dense fog; again the vehicles have turned on their lights. The classification results are shown in Table II.

The accuracy of the third parking row has decreased. This can be explained by the density of the fog, which increases with the camera distance.

The color features are not suitable to deal with the problem of fog. On operation, the system should use gray features in foggy situations instead.

	rain		snow	
	color	**gray**	**color**	**gray**
row 1	93.04	99.83	72.40	71.41
row 2	62.90	82.77	40.94	52.11
row 3	65.01	43.11	25.02	25.75

TABLE III: Accuracy for the rain and snow scene with cars having turned their lights on

C. Precipitation

Precipitation is an everyday occurrence. Therefor we evaluated the algorithm for the two most frequent common categories: Rain and snow.

1) Rain: In this experiment, it is raining heavily and the cars have to turn on their light. An example can be found in Figure 8a. The performance of the algorithm is presented in Table III.

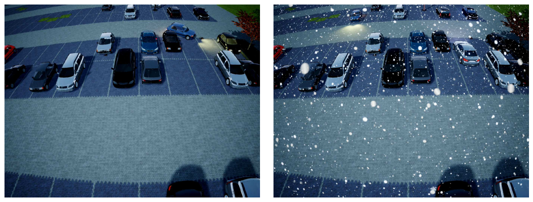

(a) rain (b) snow

Fig. 8: Images of the two different states

Again, the gray features perform better than color features. Interestingly enough, this holds only for the first and second parking row.

2) Snow: An even harder challenge for classification is snow. In this experiment, we implemented falling snow which significantly affects the camera image (*cf.* Fig. 8b). The results of the classification are shown in Table III.

The hardness of the problem can be seen in the relatively low accuracy, which is 70 % concerning the first parking row using both color and gray features. Only the second row differs notably in accuracy regarding the two features.

V. CONCLUSION

In this paper we proposed a simulation system based on Unreal Engine 4. The system contains the simulated environment, which represents a real world parking scenario and an artificial camera, implementing many of the features a real camera has. The camera captures images and provides them via shared memory.

We showed that the system can help to evaluate a given problem, *e.g.*, parking lot classification under certain lighting and weather conditions. We compared a real-world scenario from [3] and reconstructed it considering the parking situation and the movement of the cars. We reached a classification rate of 99.53 % on the first parking row, which is comparable to the results on the real-world sequence being 99.96 %. Considering the results on varying lighting and weather conditions, the gray image features (*DoG*) perform better than the color image features (using the *HSV* color space). It is noteworthy that the algorithm can deal with most of the lighting and weather conditions as well as with cars turning on their lights although the classifier was trained and tested exclusively on real-world image snippets containing only examples of a sunny and a light foggy scene, without any precipitations and headlights.

Using simulated data to train and test a new parking lot classifier and verify it with different real-world scenarios, which we already recorded, could lead to an improvement of the classifier to deal with certain problems, *e.g.*, rain and snow, where the accuracy is not satisfying at the moment. We will test the algorithm for the routing and tracking module [4] on the simulation to build up the entire video-based on-site parking guidance system. Therefor we will implement an more powerful AI which is able to perform realistic maneuvers on the parking area autonomously. A car entering the area will receive the position of the next vacant parking lot and should drive on its own taking other driving and parking cars into account.

REFERENCES

[1] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] M. Tschentscher, M. Neuhausen, C. Koch, M. König, J. Salmen, and M. Schlipsing, "Comparing image features and machine learning algorithms for real-time parking-space classifiaction," in *Proceedings of the ASCE International Workshop on Computing in Civil Engineering*, 2013, pp. 363–370.

[3] M. Tschentscher, C. Koch, M. König, J. Salmen, and M. Schlipsing, "Scalable real-time parking lot classification: An evaluation of image features and supervised learning algorithms," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2015, pp. 1–8.

[4] D. Horn and M. Brüggenthies, "Video-based parking space detection: Localisation of vehicles and development of an infrastructure for a routeing system," in *Proceedings of the Forum Bauinformatik*, 2015, pp. 175–182.

[5] G. Merkuryeva and V. Bolshakovs, "Vehicle schedule simulation with anylogic," in *Proceedings of the International Conference on Computer Modelling and Simulation*, 2010, pp. 169–174.

[6] M. Kondratyev, "An object-oriented approach to port activity simulation," *Internation Journal Simulation and Process Modelling*, vol. 10, no. 1, 2015.

[7] PTV GROUP, "What keeps traffic flowing?" Brochure, 2017. [Online]. Available: http://vision-traffic.ptvgroup.com/fileadmin/files_ptvvision/Downloads_N/0_General/2_Products/2_PTV_Vissim/BRO_PTV_Vissim_EN.pdf

[8] K. von Neumann-Cosel, E. Roth, D. Lehmann, J. Speth, and A. Knoll, "Testing of image processing algorithms on synthetic data," in *Proceedings of the International Conference on Software Engineering Advances*, vol. 4. ICSEA, 2009, pp. 169–172.

[9] J. Williams and W. Lee, "Interactive virtual simulation for multiple camera placement," in *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, vol. 5, 2006, pp. 124–129.

[10] S. Noth, "A multi-user driving simulator for studying human driving," Ph.D. dissertation, University of Bochum, 2014.

[11] J. Wilson, "Physically-based rendering, and you can too!" 2015, [Online; posted 01-October-2015]. [Online]. Available: https://www.marmoset.co/posts/physically-based-rendering-and-you-can-too/

[12] J. Russel, "Basic theory of physically-based rendering," 2015, [Online; posted 01-November-2015]. [Online]. Available: https://www.marmoset.co/posts/basic-theory-of-physically-based-rendering/

[13] D. A. Kerr, "Apex-additive system of photographic exposure," *Issue*, vol. 7, no. 2007.08, p. 04, 2007.

[14] R. E. Jacobson, S. F. Ray, G. G. Attridge, and N. R. Axford, *The Manual of Photography: Photographic and Digital Imaging*. Oxford: Focal Press, 2000.

[15] Z. Zhang, "A flexible new technique for camera calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.