Simulation of Falling Rain for Robustness Testing of Video-Based Surround Sensing Systems

Dennis Hospach, Stefan Mueller, Wolfgang Rosenstiel, Oliver Bringmann
University of Tuebingen, Faculty of Science, Dept. of Computer Engineering
Email: {dennis.hospach, stefan.mueller}@uni-tuebingen.de

Abstract—**Recently, optical sensors have become a standard item in modern cars, raising questions with respect to the necessary testing under various ambient effects. In order to achieve a high test coverage of vision-based surround sensing systems, a lot of different environmental conditions need to be tested. Unfortunately, it is by far too time-consuming to build test sets of all relevant environmental conditions by recording real video data. This paper presents a novel approach for ambient-aware virtual prototyping and robustness testing. We propose a method to significantly reduce the needed on-road recordings being used for design and validation of vision-based Advanced Driver Assistance Systems (ADAS) and fully automated driving. Our approach facilitates the generation of comparable test sets by using largely reduced amounts of real on-road recordings and applying computer-generated variations of falling rain to it in a comprehensive virtual prototyping environment. In combination with the simulation of camera properties, which influence the visual effects of falling rain to a great extent, we are able to generate different rain scenarios under a wide variety of parameters. Our approach has been applied to an automotive lane detection system using a series of multiple rain scenarios. We have explored, how falling rain can influence such a system and how such behavior can be detected using simulated rain scenarios.**

I. Introduction

In recent years, a tight integration of the physical and digital world has taken place which was caused by advances in embedded systems and sensor technology. Such systems, having connections to the physical world through sensors and actors and also incorporating an embedded system for communication and processing, are often considered as Cyber Physical Systems (CPS). These CPS are accompanied by new challenges in design and verification. Many examples for CPS can be found in a modern car, especially in the field of Advanced Driver Assistance Systems (ADAS) appearing more and more in the latest generation of cars. These ADAS heavily rely on sensor perception, many of them are vision-based. The trend is that these systems are more and more taking over active control, ranging from emergency breaking over parking assistance to traffic jam assistance, where cars drive autonomously in slow-moving traffic. These systems significantly increase driving safety and comfort, but also raise new problems and challenges concerning the functional safety. This is also reflected by the road map of various automotive companies where one of the biggest problems to address is the amount of $10^7 - 10^9$ h (see [1], [2]) of on-road recordings necessary to validate a new ADAS. Concluding from that, the limits of classical testing approaches have been reached and new solutions are needed. One possible solution is to use simulation of relevant conditions instead of comprehensive recording. To ensure the validity of the simulation results it is necessary to validate the simulations against real data.

We contribute the simulation of falling rain for robustness testing of optical surround sensing and processing systems in a comprehensive virtual prototyping environment.

II. Related work

The challenges in safety evaluation of automotive electronic using virtual prototypes are summarized in [3]. As stated by the authors, the obstacles are the validation of early state virtual prototypes against real world conditions and to handle the complexity of stress simulation in these virtual prototypes. The contributions of [4], [5] and [6] use fully artificial input data from computer simulations to test and improve the underlying hardware and algorithms. Environmental influences, if part of the simulation approach, are simulated together with the rest of the scene. In [7] the authors generate artificial input for a lane detection assistant. The influence of sensor characteristics on ADAS in combination with the simulation of the behavior of another sensor on image data is presented in [8]. In [9] a method to search huge parameter spaces and to evaluate the behavior of systems under these parameters is given.

In the scope of the simulation of weather conditions in videos, only few work has been done until now. While recent work often dealt with visually convincing results for entertainment, we focus on the impact on sensors, hardware and software. In [10], Starik and Werman discuss how to simulate the visual appearance of rain in a video sequence without knowledge of the scene depth. They derived visual properties of rain streaks in videos and separated the rain streaks from the background to build a rain mask. Some more detailed results, including environmental lighting, are created by [11]. While not considering scene depth, these approaches are missing some of the effects of rain, e.g. depth-depending water vapor. A very detailed and physically convincing discussion of simulated rain, including depth information of the scene, is given by Garg and Nayar in [12] and [13]. They extract important properties and examine their interactions with respect to a camera and present a method for rendering photo-realistic artificial rain. Other papers examine the physical properties of rain, such as drop distributions with size [14][15], terminal velocity [16] and shape [17].

III. Ambient-aware robustness testing

In the field of CPS, there is a tight coupling of the physical world with the environment. Thus, environmental effects have to be part of the modeling of a simulation. One possibility to include the ambient factors into the simulation is to use virtual sensors as connection between device and environment.

A. Simulation principles and data acquisition

In contrast to the approaches in [4], [5] and [6], we use artificially modified real data as input to the surround sensing devices. Accordingly, we stay closer to the reality and only need to simulate the effects we like to add. However, it imposes some requirements on the recorded data. First, the source material needs to be recorded with a low noise level and at known exposure time and aperture settings. The lens should be focused to infinity as well as the aperture should be kept as small as possible, to yield a wide depth of field. Additionally, the recording environment should be close to the one that is to be simulated. Further, the appearance of rain streaks changes with the depth of the scene. Thus, a simulation of falling rain drops without knowledge of the scene depth can only yield coarse results. Therefore, we do depth reconstruction from stereo data, resulting in a depth map of the scene. This is done using the StereoSGBM algorithm of OpenCV[1].

For data acquisition we use a Point Grey Research Bumblebee XB3 stereo camera. The rectified stereo pair is used as the input for the depth reconstruction stage. The left color image along with the depth map are then forwarded to the rain simulation.

For the simulation of different environmental conditions, including rain, a framework has been developed by the authors of this paper. It takes streamed image and video data as input and is able to modify the images by application of filters. A detailed description of how this framework is constructed, can be found in [18].

B. *Properties of rain drops*

Raindrops are uniformly distributed in space but the drop size follows another distribution (see [14], [12]). Additionally, the drop size distribution together with the terminal velocities are subject to change with the rain rate. The drop sizes define the visibility of drops in the final image of a camera system to a great extent. In contrast, rain drops are normally sufficiently small, so that physically correct reflection and refraction of light rays going through them may be neglected. Even the largest drops directly in front of the camera are falling fast enough to generate a motion blur, that makes these effects unimportant.

C. *Camera model and consequences for the simulation*

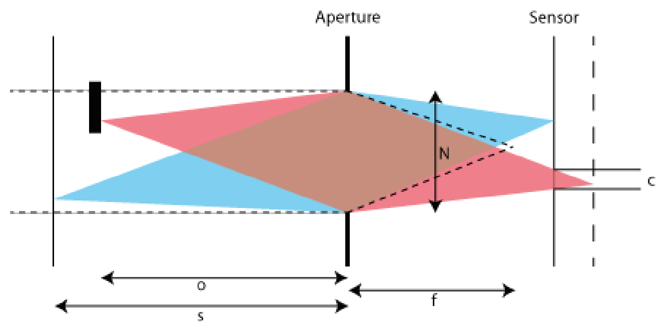

Fig. 1. The camera model and the parameters. The blue cone shows how light rays from the focused distance hit the sensor at one point. The red cone, in contrast, shows that light rays that originate from another depth, are spread over some area of the sensor and therefore appear blurred.

The standard camera model used in many computer graphics applications is the so-called pinhole camera model. Its simple geometric relations allow fast and in most cases accurate projective imaging. However, some effects that occur with real cameras are not representable in this easy model. The most important and most obvious effect is the depth blur, resulting from the limited depth-of-field of a camera, that can be observed in real images. It has severe impact on visual realism for the human eye but also on the realism when simulating the imaging process of an optical sensor. Therefore, standard pinhole camera models like the one used by OpenGL are not sufficient for this simulation and we formulate a way to enhance it.

1) Depth blur enhancement of the pinhole camera model: From the values of aperture and focus of the virtual camera, the circle of confusion of a point with scene depth o can be determined. The circle of confusion describes the concept of blur on the sensor plane introduced through scene points that are out of focus. Real cameras can focus on exactly one plane in space. The further a point is from that plane, the more blur is added to its projection on the sensor plane. The circle of confusion c is the projected size of an imaged scene point in pixels on the sensor, given by the following formula:

$$c = \frac{(o-s)*f^2}{o*(s-f)*N}$$

where o is the object distance, s the focused distance, f the focal length and N the f-number. Figure 1 shows these dimensions graphically. If $c > 1\ px$, the color of a scene point is spread over an area larger than one pixel, which leads to the observable blur effect.

2) Consequences for OpenGL: The application of blur influences the simulation of rain in two cases: First, the background of the scene is blurred as if it was recorded with the settings of the simulation camera, according to the theory of circle of confusion above. Here, it is very important, that the depth-of-field of the source material is large enough, since already present blur cannot be undone. Second, the rain drops themselves are blurred depending on their depth during the simulation. In this manner, the OpenGL pinhole camera model for projective imaging can be extended as if the camera model had a lens with aperture and focus settings.

D. *Simulation of falling rain*

1) Scene reconstruction: The simulation in OpenGL starts with a color image, a depth map and the target camera parameters as input. The depth map is generated in a preceding filter from the calibrated stereo images and contains metric distances of the image pixels from the image plane. The values are transferred into an OpenGL-generated virtual camera space by mapping the metric distances to the virtual distances so that the minimum and maximum virtual distances (the near and far plane) match the minimum and maximum real distances, found in the depth map. Additionally, the field of view of both cameras and the sensor resolution are matched. Using this relation between source and target camera, the depth-reconstructed background is placed into the virtual camera space. For this purpose, each originally sensed pixel's color, taken from the input color image, is set as the color of a rectangle, that internally consists of two triangles - as usual in computer graphics. The background rectangles are then rendered at the position in virtual camera space that corresponds to the pixel position in the source image and the depth from the depth map. At rendering time, OpenGL keeps track of the depth values of the background through its depth buffer, which gets important at a later stage.

2) Scene partitioning: In a second step, the viewing volume of the virtual camera is divided into n space partitions with a separate vertex buffer each, to account for restrictions of OpenGL regarding the number of rendering primitives per buffer. In each such partition, rain drops are placed uniformly distributed in x, y and z coordinates. The drop size follows the Marshal-Palmer-Distribution [14] for a given rain rate r in mm/hr. Additionally, we make the assumption that the drops are already at their terminal velocity, which is reasonable because the falling distance should have been sufficiently long.

3) Rain streak geometry: The terminal velocity of a drop, which comes from a lookup table, combined with the shutter speed of the virtual camera defines the falling distance in that frame and thus the length of the visible rain streak.

Each rain drop is represented by a rectangle of appropriate size. The horizontal virtual dimension is calculated by remapping the metric radius of the drop to the virtual extent:

[1]http://www.opencv.org

$$w_{virt} = d_{virt}/d_{real} * w_{real}$$

where w_{virt} is the virtual width, w_{real} the metric radius, d_{virt} the virtual depth of the plane and d_{real} the metric depth of the plane. In case the virtual width of a drop is below a threshold of 0.01% of the plane width, it is discarded from simulation in favor of simulation time, because the visual effects are no longer detectable. The virtual length is calculated by multiplying the falling speed of the drop with the shutter speed t_{cam} and then applying the same remapping operation.

By setting an angular deviation from zero degrees, the falling angle of the drop can be changed. In cases where the drop is not falling purely vertical, the length of the drop during the exposure time of the camera stays the same but start and end points are horizontally displaced.

4) Rain streak color: Next, the color of the rain streak needs to be determined. It can be shown ([12]), that the color fraction a rain drop leaves at one pixel of the sensor is dependent on its velocity, its diameter and the camera shutter speed. During the exposure time t_{cam}, the drop covers a distance of length l_{virt}, as stated above. The color fragment it leaves at a pixel is then $\alpha = d_{virt}/l_{virt}$. This value denotes the transparency value of a rain drop due to its motion blur. The base color of a drop, which in reality is scene and illumination dependent, is then weighted with α. Without knowing details on the scene illumination, the base color of a rain streak has been empirically set to a semi-transparent white with opacity factor 0.5. This has shown to produce good results. If more detail of scene illumination is known, the color value can be specified more accurate, though.

5) Final rendering: Each partition with rain streaks is then rendered with the depth buffer activated and with the depth values of the background rendering stage still loaded, so that only rain streaks that lie between the camera and the background are rendered. The target of this rendering operation is a texture, that has been initialized with transparent black. Afterwards, the texture contains the rain streaks with correct transparency values. Next, the blur that is introduced by the circle of confusion is added to the texture of rain streaks using a gaussian blur shader. This rendering operation is performed with the frame buffer as target, such that after blending the blurred texture with the already rendered background, the final image is ready.

After rendering, a copy of the frame buffer (the output image) is given back to the framework. As we incorporate the idea of temporally independent frames, described in [10], we do not track single rain drops over time but render the next frame with new random drop locations and sizes.

A resulting image of the rain filter is shown in figure 2. Notice the influence of scene depth on visible drop count and contrast of the background.

IV. Validation

To gain valid testing results, the simulation is expected to produce effects that are similar to those that emerge in real rain images. Observing and isolating these effects in a complex system like lane detection, that consists of multiple basic computer vision algorithms, is difficult.

To solve this problem, the verification of the simulation against real data has been done by applying basic algorithms of computer vision, that are part of many of the target applications. Arguing with the results of the metrics of these basic algorithms is by far easier and the problem of mutual interference can be avoided. Algorithms from different categories have been selected to get a broad overview. These incorporate sobel filtering, canny edge detection, harris corner detection, histogram evaluation and SURF feature detection.

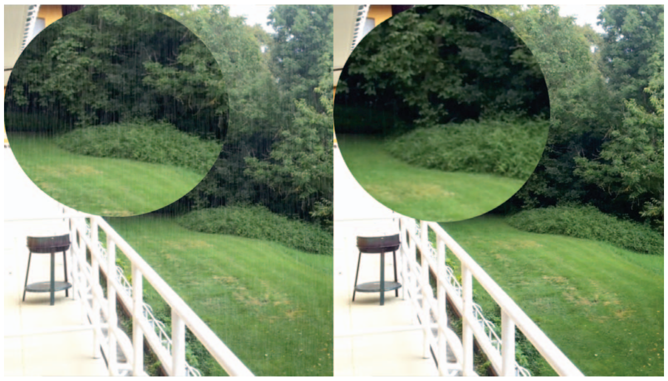

Fig. 2. Rain with an artificial depth map. The depth map contains two levels of depth for illustrative purposes, containing higher depth values in the left part. The number of drops between a scene object and the virtual camera depends on the scene depth. Therefore, in the left part, more drops are visible than in the right part.

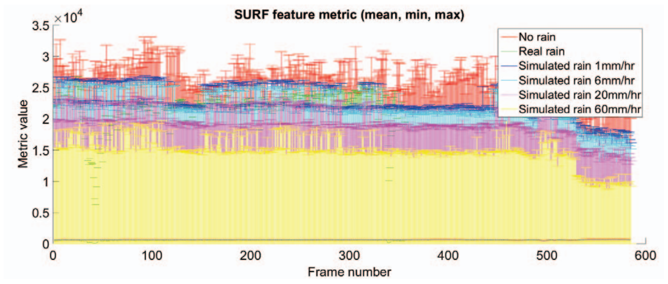

Fig. 3. Evaluation results of the SURF feature metric on test sets with real and simulated rain and approximately 600 frames each. At the bottom, the mean metric values are drawn as solid lines. The minimum and maximum values are drawn as bars. The higher the real or simulated rain rate is, the lower is the maximum confidence metric. The drop of the metric of real rain around frames 50 and 350 is due to motion blur.

The MATLAB implementation of each of those basic features has been applied to a series of real rain images and a series of images with simulated rain. The simulated rain series has been calculated from a series of images without rain that has been recorded temporally close to the real rain series and with fixed camera parameters to minimize external influences through changes of illumination etc. Furthermore, the recording of validation data has been done with different lenses and parameter sets to account for effects emerging from these sources and the simulation has been done with different rain intensities, ranging from 1 mm/hr to 60 mm/hr. The rain rate of the real rain series has been coarsely determined by taking the mean of surrounding weather stations at the time of recording. Its value has been around 8 mm/hr. In each run, the minimum, maximum and mean of the metric (if available) and the feature detection rate has been recorded and drawn into a figure for final evaluation.

Figure 3 shows the results of one of the SURF feature evaluations. It depicts the mean values (solid lines at the bottom) of the SURF feature metric in a series of about 800 frames together with the minimum and maximum values as bars. Since no thresholding has been done, the minimum is always close to zero. The maximum values of the confidence metric drop with higher rain rates. The highest values are reached without rain (red) and are lower when real rain at 8 mm/hr is observed (green). The metric is falling consistently when evaluating with simulated rain (blue, cyan, magenta and yellow bars) and reaches the green level somewhere between blue and cyan, where it should be on grounds of the rain rates.

The rest of the validation results will be given in a summarized fashion. The edge detection algorithms, sobel and canny, show falling confidence metrics with increasing rain rate. This effect is inverted if thresholding is used, mainly due to the removal of weak edges and therefore rising mean. In the test set with real rain, this effect could also be observed. Harris corner detection behaves the same way. In contrast, the hough circle detection metric shows rising values with rising rain rates, but this does not result in more recognized circles due to the robustness of hough.

In summary, the validation of the simulation with basic algorithms showed only minor differences, that occurred especially when the source images and the real rain images were illuminated slightly different.

V. Results

In the first part of this paper we have shown how test sets with artificial rain can be built and discussed the validation results. However, to be sure that our artificial test sets are helpful for robustness testing, we have also tested the simulation results with typical algorithms in the scope of surround sensing systems. As an example, we applied the implementation of a lane detection algorithm (see [19]) that is available as source code, to a test set of simulated rain, varied in two parameters. As input sequences, the accompanying ground truth image sets have been used. We applied artificial rain with rain intensities between $0\ mm/hr$ and $40\ mm/hr$ and falling angles between -45 to 45 degrees to it and then fed each set into the lane detection.

The recognition results are shown in figure 4. At first sight, you may notice that there is a drop of the recognition rate when the simulated rain angle is near 0 degrees and a steady drop with increasing rain rate. Rain streaks with a falling angle of approximately zero degrees are falling nearly vertical. To find an explanation for the drop of recognition rate, we looked into the details of the implementation and found, that the algorithm does a perspective remapping to transform the front view images to a top view image. This leads to rain streaks that appear as lines parallel to potential lane markers and therefore to more difficulties for the algorithm to distinguish rain from lane markers.

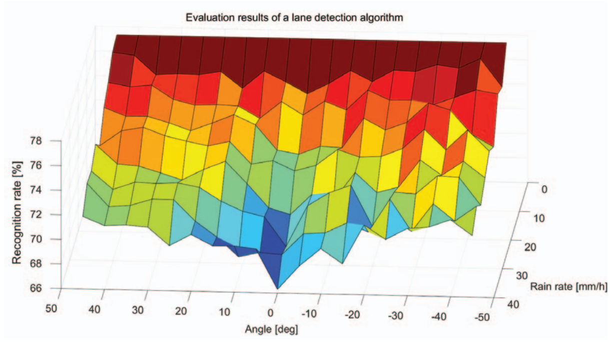

Fig. 4. Results of the lane detection on a test set of simulated rain. You can clearly see the drop of recognition rate around a falling angle of 0 degrees.

VI. Conclusion and future work

We presented a way to generate and apply synthetic falling rain to real on-road recordings to evaluate the robustness of vision-based surround sensing systems against such influences, combining both the real world and the computer generated effects. We evaluated the correctness of the simulation against ground-truth results of real rain using basic computer vision algorithms. Finally, we showed by example that the results can disclose strengths and weaknesses of such systems in a sufficiently high speed, gaining hints on difficulties of the system under rainy conditions.

In future work, we are planning to combine the simulation of rain with further environmental effects like dust simulation of camera lenses and to evaluate system behavior under a wider set of effects and combinations of parameters. Further, the simulation will be extended to include some fluid water simulation on wind shields and camera lenses to further increase simulation quality. Including more information of light sources and illumination by image analysis could also increase the simulation quality in special illumination cases.

Acknowledgment

This work was partially funded by the State of Baden Wuerttemberg, Germany, Ministry of Science, Research and Arts within the scope of a Research Training Group. Additionally, it has been partially supported by the Deutsche Forschungsgemeinschaft (DFG) in the priority program 1835 under grant BR2321/5-1.

References

[1] A. Weitzel and S. Geyer, *Absicherungsstrategien für Fahrerassistenzsysteme mit Umfeldwahrnehmung*. Bundesanstalt für Straßenwesen, 2014.

[2] R. B. G. Nordbusch, Stefan, "Vision or Reality The Way to Fully Automated Driving," Berlin, 2014 Vortrag - edaForum14.

[3] N. Bannow, M. Becker, O. Bringmann, A. Burger, M. Chaari, S. Chakraborty, R. Drechsler, W. Ecker, C. Kuznik, H. M. Le, A. Mauderer, F. Poppen, H. Post, S. Reiter, W. Rosenstiel, S. Roth, U. Schlichtmann, and A. Viehl, "Safety Evaluation of Automotive Electronics Using Virtual Prototypes: State of the Art and Research Challenges," in *DAC 2014*, 2014.

[4] K. V. Neumann-Cosel, E. Roth, D. Lehmann, J. Speth, and A. Knoll, "Testing of Image Processing Algorithms on Synthetic Data," *2009 Fourth International Conference on Software Engineering Advances*, pp. 169–172, Sep. 2009.

[5] M. Nentwig and M. Stamminger, "Hardware-in-the-loop testing of computer vision based driver assistance systems," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, June 2011, pp. 339–344.

[6] M. Nentwig, M. Miegler, and M. Stamminger, "Concerning the applicability of computer graphics for the evaluation of image processing algorithms," *2012 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2012*, pp. 205–210, 2012.

[7] F. Coskun, O. Tuncer, M. E. Karsligil, and L. Guvenc, "Real time lane detection and tracking system evaluated in a hardware-in-the-loop simulator," *13th International IEEE Conference on Intelligent Transportation Systems*, pp. 1336–1343, Sep. 2010.

[8] D. Hospach, S. Mueller, J. Gerlach, O. Bringmann, and W. Rosenstiel, "Simulation and Evaluation of Sensor Characteristics in Vision Based Advanced Driver Assistance Systems," in *17th International IEEE Conference on Intelligent Transportation Systems*, 2014.

[9] S. Mueller, D. Hospach, J. Gerlach, O. Bringmann, and W. Rosenstiel, "Robustness Evaluation and Improvement for Vision-based Advanced Driver Assistance Systems," in *18th International IEEE Conference on Intelligent Transportation Systems*, 2015.

[10] S. Starik and M. Werman, "Simulation of rain in videos," *Texture Workshop, ICCV*, 2003.

[11] L. Wang, Z. Lin, and X. Yu, "Real-Time Rendering of Realistic Rain," Microsoft Research, Tech. Rep., 2005.

[12] K. Garg and S. K. Nayar, "Photorealistic rendering of rain streaks," *ACM Transactions on Graphics*, vol. 25, no. 3, p. 996, Jul. 2006.

[13] ——, "Vision and Rain," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 3–27, Feb. 2007.

[14] J. Marshall and W. Palmer, "The distribution of raindrops with size," *Journal of Meteorology*, 1948.

[15] P. Willis, "Functional fits to some observed drop size distributions and parameterization of rain," *Journal of the atmospheric sciences*, 1984.

[16] R. Gunn and G. Kinzer, "The terminal velocity of fall for water droplets in stagnant air," *Journal of Meteorology*, 1949.

[17] K. Beard and C. Chuang, "A new model for the equilibrium shape of raindrops," *Journal of the Atmospheric sciences*, 1987.

[18] S. Mueller, D. Hospach, J. Gerlach, O. Bringmann, and W. Rosenstiel, "Framework for varied sensor perception in virtual prototypes," in *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 03 2015.

[19] M. Aly, "Real time detection of lane markers in urban streets," *2008 IEEE Intelligent Vehicles Symposium*, pp. 7–12, Jun. 2008.