2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

Augmenting Image Data Sets With Water Spray Caused by Vehicles on Wet Roads

Alexander von Bernuth[1], Georg Volk[1], and Oliver Bringmann[1]

Abstract— **Adverse weather conditions challenge object detection neural networks, because they are mostly trained on clean data sets that were taken in good weather. But autonomous vehicles rely on accurate detection and classification of other road users for safe and reliable operation. Capturing new data sets in rough weather is time consuming and expensive. We therefore propose a method to augment existing image data sets with physically realistic water spray swirled up by vehicles—one of the most influential disturbances for even human drivers on wet roads in or after heavy rain. Using a wide range of newly augmented images, we evaluate the influence on an established convolutional neural network object detection and showcase the potential of training with these new augmented data sets.**

I. INTRODUCTION

In the light of recent reports of road accidents that involve at least one autonomous vehicle, car manufacturers strive to limit the damage done to their images. These incidents not only reduce trust into the responsible company—trust into autonomous transportation as a whole is shaken by reports of vehicles that collide head on, injuring unsuspecting passengers as well as bystanders.

This circumstance is reinforced by the fact that humans tend to blame machines more than other humans for their roles in accidents [1]. In otherwise equal scenarios, a vehicle driven by a machine learning algorithm is far more likely to be held accountable for its errors than a human driver would be. When the outcome of the scenario was more severe, the blame increased.

But it does not need real accidents to decrease the perceived reliability of self driving cars: even just prompting the passenger to take over—because of system failure, adverse weather conditions, or other road users—lowers the overall reaction time to those disengagements, pointing to a increased lack of trust [2].

To regain this trust, autonomous vehicles have to operate without error for a prolonged time and avoid handing the control back to the passenger. Disengagements are caused by adverse weather conditions about as frequently as by other road users and twice as frequent as by roadwork sites. But image data sets, widely used to train machine learning algorithms, mostly contain images taken under ideal weather conditions. Only slowly are data sets published that contain adverse conditions like rain, snow, or fog.

This work has been partially funded by the German Research Foundation (DFG) in the priority program 1835 under grant BR2321/5-2.

[1]University of Tübingen, Faculty of Science, Department of Computer Science, Embedded Systems Group {alexander.von-bernuth, georg.volk, oliver.bringmann}@uni-tuebingen.de

One method to quickly and accurately create those data sets is to simulate a broad range of weather conditions and, with the results, augment existing data sets. In this work we set out to simulate a byproduct of rain, the most frequent adverse weather condition: the spray dispersed by quickly moving vehicles during rain or on wet roads. Spray reduces visibility and consequently reaction times, making it the most frequent reasons for accidents in rain [3]. Adding this spray to training data sets improves the overall resilience against rainy situations and on the other hand shows where existing machine learning algorithms are lacking.

In section II we highlight the relevant previous work and show that no other simulation handles the augmentation of images with spray. Then we describe how our simulation generates and then renders the spray behind vehicles in section III. Using our simulation we showcase the rendered spray and elaborate the impact on machine learning object detection algorithms in section IV. Finally, section V concludes this paper and suggests further research topics.

II. RELATED WORK

Multiple data sets, containing images, depth information, and even LiDAR measurements, already exist. Most prominent are the KITTI data set [4] and the Cityscapes data set [5]. Both contain hundreds of manually labeled images, enabling machine learning algorithms to learn object detection, depth perception, and semantic segmentation—but neither contain any images taken under adverse weather conditions. A variety of research was carried out based on these and similar data sets training neural networks to near perfection, neglecting disturbances introduced by rain, snow, or fog. Some research exists dealing with the removal of adverse weather conditions from input images [6], [7], but these expensive neural networks are not fit for real time applications in automated vehicles.

To combat this shortcoming, many data sets containing scenes with visible weather conditions emerged, either by augmenting existing ones or by creating completely new ones. Kenk et al. collected a selection of scenes in snow, rain, and fog, complete with annotations [8]. Similarly, Volk et al. gathered a multitude of images from publicly available dashcam videos during light to heavy rain and provided bounding boxes for all vehicles [9].

Completely new data sets were produced by Zhou et al., who specifically drove on a campus at different times and weather conditions to capture a wide variety of scenes [10]. The same was done by Tung et al., who focused on rides during dusk and night, especially during rain [11].

978-1-7281-9142-3/21/$31.00

Collecting new data sets is expensive and time consuming—not only because of the cost to outfit a vehicle with sensors, but also because the recorded data has to be (often manually) labeled. Additionally, the vehicle and its driver have to stand by, waiting for the required weather to happen. Therefore augmenting already labeled data sets is a viable option, especially when the simulated weather conditions can be varied widely to create an even larger number of new images.

By augmenting Cityscapes with synthetic fog, Sakaridis et al. created Foggy Cityscapes [12]. Simulating rain and adding it to preexisting images was done by Sindagi et al., who applied a simple mask onto the images [13], or by Halder et al., who used simulated physics and optics to create realistic rainy images [14].

Different adverse weather conditions were simulated by a range of researchers, for example rain [15], [16], snow [17], [18], and fog [17], [16].

Lacking from all previous simulation methods is the spray that is dispersed by vehicles that drive on wet surfaces. To simulate this physically correct, we used the work of Kooij et al. regarding the size of spray [19], a multitude of research regarding the movement of spray [20], [21], [22], [23], [24], [25], and the fast OpenGL render method for water droplets by Slomp et al. [26].

III. SPRAY SIMULATION

Spray occurs when a vehicle drives on a surface that is covered by a layer of water. This happens even on the best asphalt roads when the rain is heavy enough. Small water droplets are carried up by the wheels of the vehicle and are released into the air behind it.

For a mechanically accurate and optically exact spray simulation, we have to first position all drops of the spray correctly, and then render all generated drops quickly. The rendering step is based on a single data set image like provided by Cityscapes or KITTI. The challenge comes from augmenting an existing 2D image instead of fully rendering a synthetic scene as can be done in Carla or other 3D simulations. To maintain feasibility, both steps are optimized and, if the results are not affected, approximated.

A. Drop Positioning

Numerical physics simulations, like the ones carried out by Kabanovs et al. [20] or Kuthada et al. [27], show that spray drop diameters vary between $10\,\mu\mathrm{m}$ and $500\,\mu\mathrm{m}$. The simulations by Kuthada et al. use a fixed diameter of $200\,\mu\mathrm{m}$ as their results match experimental results best at this value. While vehicle speed does have an impact on the drop diameter, that minor impact allows us to neglect the influence on the diameter for a simpler simulation. We used a mean diameter of $200\,\mu\mathrm{m}$ with a standard deviation of $10\,\mu\mathrm{m}$ and sampled each droplet from this normal distribution. The range of very small fog-like droplets have to be taken into account, too. This will be done at the end in a separate step.

The initial velocity of a droplet originating from a spinning wheel is assumed to be equal to the rotational velocity of that wheel. As the wheel is as fast as the vehicle it is attached to, the initial droplet velocity is equal to the vehicle velocity v, conversely. Together with the angle α the droplet is ejected relative to the ground, the initial droplet velocity vector is defined as

$$\vec{v}_0 = \begin{pmatrix} v \cdot \cos\alpha \\ v \cdot \sin\alpha \end{pmatrix}. \tag{1}$$

After a spray drop is detached from the wheel, it is subject to aerodynamics and gravity. Air resistance acts directly opposed to the drop's velocity vector, slowing the droplet down along the way. The slower the droplet gets, the weaker the aerodynamic force becomes, as the drag force F_W is defined as

$$F_W = c_W A \frac{\varrho v^2}{2}, \tag{2}$$

where c_W is the drag coefficient ($c_W = 0.45$ for a small sphere), A is the area exposed to the drag ($A = 2r\pi$ with the droplet radius r), ϱ is the mass density of the surrounding liquid ($\varrho = 1.293\,\mathrm{kg\,m^{-3}}$ for air), and $v = \|\vec{v}\|_2$ is the absolute velocity of the droplet [28]. As v decreases, F_W decreases as well and slows the droplet down less. The drag deceleration vector $\vec{a}_{\text{drag}}$ can be determined by solving $F_W = m\vec{a}_{\text{drag}}$ (Newton's Second Law) for a and substituting $m = \frac{4}{3}\pi r^3 \rho$ (where ρ is the mass density of water):

$$\vec{a}_{\text{drag}} = \frac{F_W}{\frac{4}{3}\pi r^3 \rho} \cdot \frac{-\vec{v}}{\|\vec{v}\|_2}. \tag{3}$$

This deceleration opposes the velocity vector.

At the same time, gravity acts on the up-component of $\vec{v}$, bringing the upward motion to a stop and accelerating the drop back down—until being counteracted by drag again (or stopped by the floor). The simple resulting acceleration vector can be described as

$$\vec{a}_{\text{g}} = \begin{pmatrix} 0 \\ -g \end{pmatrix}, \tag{4}$$

where g is the acceleration of gravity. The combined acceleration acting on a droplet is then

$$\vec{a} = \vec{a}_{\text{drag}} + \vec{a}_{\text{g}}. \tag{5}$$

Because, given a time since the beginning of the flight, a droplet's position depends on the current deceleration it experiences by drag—which in turn depends on the velocity—it is easiest to numerically approximate the whole process. For this, we updated all accelerations, the velocity, and therefore the position in very small time steps. An exemplary resulting flight path is shown in Figure 1.

After determining the flight path of a single droplet in 2D space, we can now transfer this simulation into the 3D space. We added jitter to the positions of the droplets at every time step, gaining standard deviation the longer the time of flight. An exemplary drop distribution can be seen in Figure 2.

Having calculated the position of each droplet, we can now distribute the spray clouds in the 3D scene, each originating

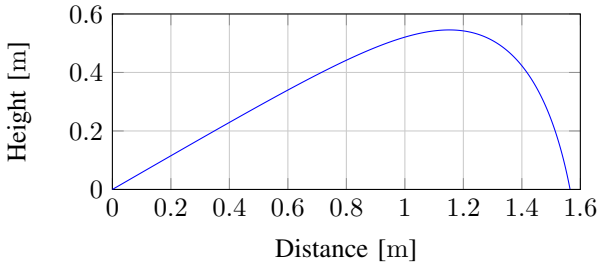

Fig. 1: Side view of a single spray droplet. After being flung from a rotating wheel, a droplet is affected by gravity (down) and aerodynamic drag (opposing the velocity vector of the droplet). The latter leads to a fast decrease in velocity in the x-direction. This example uses an initial velocity of $36\,\mathrm{m\,s^{-1}}$ and a spray angle of 30°.

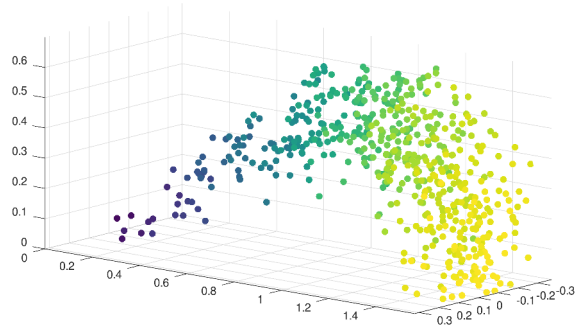

Fig. 2: Example of a drop distribution behind an imaginary wheel positioned at the origin. To maintain visibility, this plot largely reduced the number of drops. Colors indicate the longitudinal distance from the origin and aid spatial vision. The axes show meters.

at a relevant wheel in the scene. The wheel positions can either be detected by computer vision algorithms, neural networks, LiDAR matching, or be read from a 3D-labeled data set.

B. Drop Rendering

After the positions of each wheel and each spray droplet are determined, the latter have to be rendered as spheres [29]. With a radius of around $200\,\mu\mathrm{m}$, the droplets are just large enough to influence visible light geometrically, opposed to Mie scattering that occurs when the droplets are smaller, like in fog [30]. Geometric reflection happens many times in the scene; between the light source, objects in the scene, and in between the spray droplets. The geometric refraction, that occurs inside of the single droplets, can itself lead to total internal reflections or secondary reflections after the first refraction. To keep the computational load manageable, we focused on the primary reflections and refractions that happen at a single droplet.

Even when only calculating the primary light rays that color each pixel in the resulting image, these complex calculations can be very time intensive. For each pixel one (or even 4, 8, or 16, if super sampling is activated to battle aliasing) rays would have to be shot into the scene. Each ray would have to be checked for collisions with a droplet, then both reflection and refraction on the droplet would have to be calculated. Finally, the reflected and refracted rays would have to be collided with the 3D scene.

Instead, we utilized the work of Slomp et al. [26]. In place of the expensive ray tracing, they precalculate many possible reflection and refraction vectors depending on both distance of a drop to the camera and the position where the drop is hit by a ray. Being implemented in OpenGL, these precalculated vectors are then stored in textures, using the RGB channels of a texture image as XYZ coordinates for the directional vectors. This yields a three dimensional texture, a mipmap texture, for each reflection and refraction. The textures are shown above the arrow in Figure 3.

Water drops are then represented by billboard quads—squares, that always face the camera. Compared to creating the rather complex spherical shape of a drop using triangles, the usage of these billboard quads only requires two triangles to be instantiated. Mapped onto these quads are the reflection and refraction textures. When rendering the quads with OpenGL, the correct direction vectors are just looked up in the textures and do not have to be calculated every single time.

Having looked up both reflection and refraction vectors, the renderer has to know what that ray would hit if it flew in this direction. For this we created a cubemap from the input image. A cubemap is usually generated from a 360° image taken with specialized cameras and describes the colors of the environment around a given point in space. As we only have a single image to work with, we cut this image and stitch it together for a simple approximated cubemap. This cubemap then serves as lookup source for the renderer.

The resulting reflected and refracted color are then mixed according to the Fresnel effect. This effect describes the phenomenon that the obtuser the angle in which a ray hits a reflecting and refracting surface, the more reflected light contributes to the final color.

To account for minor disturbances on the surface of a moving water droplet, we approximated the very small waves that would form on the surface [29]. Instead of directly modifying the surface, we apply a pseudo-random three dimensional vector to all vectors that are read from the vector textures. As the random vectors may not have any abrupt value changes, we used an GLSL implementation of simplex noise [31]. Simplex noise is a gradient noise, meaning that continuous changes in the input result in continuous changes in the output—perfect for our use case.

Geometrically added to the original ray direction—and then normalized—this new vector behaves as if the droplet surface was under the influence of wind. The process is detailed in Figure 4 and the resulting droplet with disturbed surface is shown in Figure 3.

Finally, the magnitude of droplets that are too small to qualify for geometric reflection and refraction has to be taken

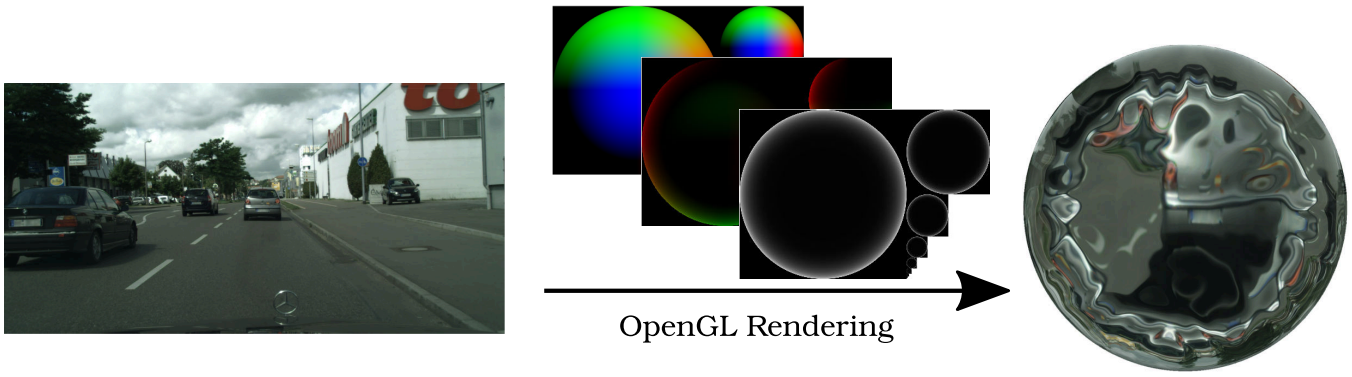

Fig. 3: The render pipeline visualized using a single large drop in front of the camera. From a single image, an environment map is created. Then, using precalculated vector masks saved in textures, a quick approximation of primary refraction and reflection is created. Both are then blended according to the Fresnel effect to produce the drop on the right. The wavy effect is achieved by adding continuous simplex noise to the reflection and refraction vectors. The exemplary input image is taken from Cityscapes [5].

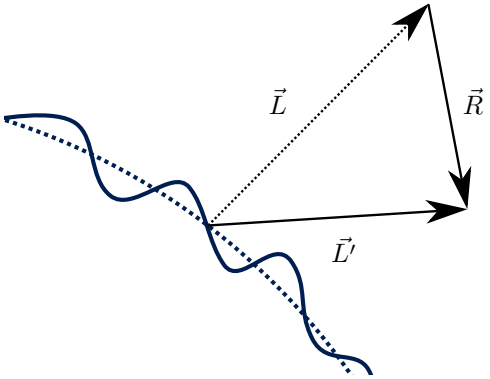

Fig. 4: Adding three dimensional simplex noise in form of a random vector $\vec{R}$ to the light ray direction $\vec{L}$ after reflection and refraction. The resulting vector $\vec{L}'$ lets the reflection and refraction behave as if the droplet surface (dashed blue line) was disturbed by wind (solid blue line). All lengths and heights are exaggerated for better visibility.

into account. Being basically a locally bounded fog, they experience in and out scattering [30]. Instead of rendering the very high number of those micro droplets, we accounted for them by mixing the sky color into the final droplet color. The sky color is assumed to be the color sampled from the cubemap by an up-pointing vector.

IV. RESULTS

To test and compare the results, we used two different types of input data: the Cityscapes data set [5] and the KITTI 3D data set [32]. We chose these quite different data sets to show the adaptability of our method.

The Cityscapes data set consists of stereo images taken from a vehicle, looking into the direction of travel. It is accompanied by manual labeled ground truth in form of image-space bounding boxes around objects. To get it working with our simulation, we used the provided disparity maps to get the necessary information about the distance of each pixel in the images. Each image-space bounding box can then be approximately positioned in three dimensional space. In case of vehicles driving in front of the ego vehicle, these boxes mark the rear surface of those vehicles. By assuming that the vehicles have two wheels and that these are located at the bottom of the vehicle, left and right, we can calculate the spacial positions of our spray origins. Because each pixel has an associated depth, we can even reconstruct the complete scene in three dimensional space before we add our spray to it [17]. This way we achieve occlusion of the rendered spray by objects that are closer to the camera. An exemplary image can be seen in Figure 5. We compare it to an image from the Realrain data set [9]. The comparison only focuses on the occlusion that the spray provides—all other circumstances like fog in the image, or different lighting conditions cannot be compared directly. This stems from the lack of clean spray data sets—the very issue we want to improve with this work.

On the other hand, the KITTI 3D data set provides single images of road scenes, but comes with hand-labeled LiDAR space coordinates, dimensions, and headings for all road users, as well as image-space bounding boxes. We read the locations of all road users, determined their respective direction of travel, and looked up their length along this direction. Going back half that length, down half their height, and left and right half their width, we found their wheel positions.

Next, we used the RESIST testing framework [33] to include our render pipeline into a evaluation environment. With help of the framework, we read all KITTI 3D images and ground truth data and fed the images into YOLOv3 [34], an object detection convolutional neural network. Then we assessed the performance of YOLOv3 on the clean, unaltered KITTI 3D images. As metric we used the average precision with an Intersection over Union threshold of 0.5.

Having established a baseline, we gradually added spray to all images on the fly, starting with an assumed vehicle

Fig. 5: Qualitative comparison of real spray (taken from the Realrain data set [9]) on the left, and our simulated spray on the right. Because of the lack of clean spray data sets, we can only compare the occlusion of the lower end of the vehicle. Here, we can observe similar behavior: parts of the wheel are not visible, as well as part of the rear end and parts of the rear lights. The spray conceals vital image features that might be used for robust object detection. The spray color blends in with the background and the color of the street; it reaches the same height as the real spray.

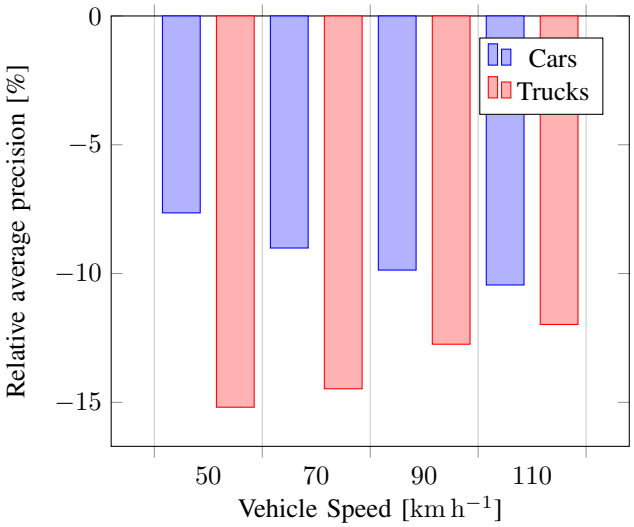

Fig. 6: Comparison of the average precision of the object detection, grouped by vehicle speed and vehicle type. The baseline for each relative average precision was taken from clean images. The average precision is lower than the baseline at every speed. The faster a car drives, the higher the spray is swirled up, and the more that car is occluded by its spray. Even trucks are not detected as well when they produce spray while driving on wet roads. The truck comparison's significance suffers from the low number of trucks in the data set (only about 1.100 trucks, compared to 28.700 cars).

speed of $50\,\mathrm{km\,h^{-1}}$ (the standard speed allowed within city limits in Germany), going up to $70\,\mathrm{km\,h^{-1}}$, $90\,\mathrm{km\,h^{-1}}$, and $110\,\mathrm{km\,h^{-1}}$ (a sane vehicle speed on very wet, rainy highways). We then compared the object detection average precision of each run with the baseline values, separate for cars and trucks. The relative decrease in average precision compared to the baseline, is graphed in Figure 6. Looking at the average precision of the detection of just cars, we can observe a drop of $7.6\,\%$ at the lowest and a drop of $10.4\,\%$ at the highest speeds. This indicates an important gap in the training of the neural network, which could be bridged by using a data set augmented by our method. Because there are comparatively few trucks labeled in the KITTI 3D data set, the drop in average precision is not as meaningful as the drop regarding cars—but a trend is visible: even the detection of trucks, which are usually taller than cars, are affected by the spray they produce.

V. Conclusion & Outlook

In this paper we examined the physics behind spray caused by vehicles that drive on wet roads. Based on this numerical simulation, we positioned individual spray droplets behind every vehicle in an existing data set image. We then showed that the influence of the augmented adverse weather condition has severe influence on object detection algorithms, because those are trained on mostly clean images. This highlights the need of more diverse training data sets for machine learning algorithms.

Our work can provide the edge for the training process, as we can augment existing data sets with our simulated spray, leading to cheap, accurate, and customizable new data sets. Combined with other augmentations like falling rain, raindrops on the windshield, or wet roads, a comprehensive simulation framework can be utilized to create even more realistic weather conditions.

A next step would be the quantitative evaluation of our simulations, both for visual realism and the impact on other state of the art object detection algorithms. For that, a real world spray data set would be the foundation.

References

[1] J. Hong, "Why Is Artificial Intelligence Blamed More? Analysis of Faulting Artificial Intelligence for Self-Driving Car Accidents in Experimental Settings," *International Journal of Human–Computer Interaction*, vol. 36, no. 18, pp. 1768–1774, Nov. 2020, https://doi.org/10.1080/10447318.2020.1785693.

[2] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous Vehicles: Disengagements, Accidents and Reaction Times," *PLOS ONE*, vol. 11, no. 12, p. e0168054, Dec. 2016, https://dx.plos.org/10.1371/journal.pone.0168054.

[3] T. Berg, "Aerodynamic Designs Help Cut Water Spray," http://www.truckinginfo.com/156364/aerodynamic-designs-help-cut-water-spray, Oct. 2015.

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, http://arxiv.org/abs/1604.01685.

[6] Z. Shi, E. Tseng, M. Bijelic, W. Ritter, and F. Heide, "ZeroScatter: Domain Transfer for Long Distance Imaging and Vision through Scattering Media," *arXiv:2102.05847 [cs]*, Feb. 2021, http://arxiv.org/abs/2102.05847.

[7] L. Gao, W. Long, Y. Li, H. Liu, X. Yu, and J. Li, "RASWNet: An Algorithm That Can Remove All Severe Weather Features from a Degraded Image," *IEEE Access*, vol. 8, pp. 76 002–76 018, 2020, https://ieeexplore.ieee.org/document/9075235/.

[8] M. A. Kenk and M. Hassaballah, "DAWN: Vehicle Detection in Adverse Weather Nature Dataset," *arXiv:2008.05402 [cs]*, Mar. 2020, http://arxiv.org/abs/2008.05402.

[9] G. Volk, S. Müller, A. von Bernuth, D. Hospach, and O. Bringmann, "Towards Robust CNN-based Object Detection through Augmentation with Synthetic Rain Variations," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 285–292.

[10] W. Zhou, J. S. Berrio, C. De Alvis, M. Shan, S. Worrall, J. Ward, and E. Nebot, "Developing and Testing Robust Autonomy: The University of Sydney Campus Data Set," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 4, pp. 23–40, Jun. 2020, https://ieeexplore.ieee.org/document/9109704/.

[11] F. Tung, J. Chen, L. Meng, and J. J. Little, "The Raincouver Scene Parsing Benchmark for Self-Driving in Adverse Weather and at Night," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2188–2193, Oct. 2017, http://ieeexplore.ieee.org/document/7970170/.

[12] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic Foggy Scene Understanding with Synthetic Data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, Sep. 2018, http://arxiv.org/abs/1708.07819.

[13] V. A. Sindagi, P. Oza, R. Yasarla, and V. M. Patel, "Prior-based Domain Adaptive Object Detection for Hazy and Rainy Conditions," *arXiv:1912.00070 [cs]*, Jul. 2020, http://arxiv.org/abs/1912.00070.

[14] S. S. Halder, J.-F. Lalonde, and R. de Charette, "Physics-Based Rendering for Improving Robustness to Rain," in *International Conference on Computer Vision*, Seoul, Korea, 2019, p. 19, https://team.inria.fr/rits/computer-vision/weather-augment/.

[15] A. von Bernuth, G. Volk, and O. Bringmann, "Rendering Physically Correct Raindrops on Windshields for Robustness Verification of Camera-based Object Recognition," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 922–927, https://embedded.uni-tuebingen.de/assets/publications/vonBernuth-Volk-Bringmann_Raindrops.pdf.

[16] S. Hasirlioglu, "A Novel Method for Simulation-based Testing and Validation of Automotive Surround Sensors under Adverse Weather Conditions," Dissertation, Universität Linz, 2020, https://epub.jku.at/obvulihs/download/pdf/4837383?originalFilename=true.

[17] A. von Bernuth, G. Volk, and O. Bringmann, "Simulating Photo-realistic Snow and Fog on Existing Images for Enhanced CNN Training and Evaluation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 41–46, https://embedded.uni-tuebingen.de/assets/publications/vonBernuth-Volk-Bringmann_Snow_Fog.pdf.

[18] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," *arXiv:1807.01697 [cs, stat]*, Jul. 2018, http://arxiv.org/abs/1807.01697.

[19] S. Kooij, R. Sijs, M. M. Denn, E. Villermaux, and D. Bonn, "What Determines the Drop Size in Sprays?" *Physical Review X*, vol. 8, no. 3, p. 031019, Jul. 2018, https://link.aps.org/doi/10.1103/PhysRevX.8.031019.

[20] A. Kabanovs, A. Garmory, M. Passmore, and A. Gaylard, "Investigation into the dynamics of wheel spray released from a rotating tyre of a simplified vehicle model," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 184, pp. 228–246, Jan. 2019, https://linkinghub.elsevier.com/retrieve/pii/S0167610518307669.

[21] X. Hu, L. Liao, Y. Lei, H. Yang, Q. Fan, B. Yang, J. Chang, and J. Wang, "A numerical simulation of wheel spray for simplified vehicle model based on discrete phase method," *Advances in Mechanical Engineering*, vol. 7, no. 7, p. 168781401559719, Jun. 2015, http://journals.sagepub.com/doi/10.1177/1687814015597190.

[22] I. O. Kamm and G. A. Wray, "Suppression of Water Spray on Wet Roads," in *1971 Automotive Engineering Congress and Exposition*, Feb. 1971, p. 710120, https://www.sae.org/content/710120/.

[23] A. Gaylard, A. Kabanovs, J. Jilesen, K. Kirwan, and D. Lockerby, "Simulation of rear surface contamination for a simple bluff body," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 165, pp. 13–22, Jun. 2017, https://linkinghub.elsevier.com/retrieve/pii/S0167610516300319.

[24] A. P. Gaylard, K. Kirwan, and D. A. Lockerby, "Surface contamination of cars: A review," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 231, no. 9, pp. 1160–1176, Aug. 2017, http://journals.sagepub.com/doi/10.1177/0954407017695141.

[25] A. P. Gaylard, "Vehicle Surface Contamination, Unsteady Flow and Aerodynamic Drag," Dissertation, University of Warwick, Warwick, United Kingdom, Oct. 2019, https://wrap.warwick.ac.uk/143065/1/WRAP_Theses_Gaylard_2019.pdf.

[26] M. Slomp, M. W. Johnson, T. Tamaki, and K. Kaneda, "Photorealistic real-time rendering of spherical raindrops with hierarchical reflective and refractive maps," *Computer Animation and Virtual Worlds*, vol. 22, no. 4, pp. 393–404, 2011, https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.421.

[27] T. Kuthada and S. Cyr, "Approaches to vehicle soiling," in *4th FKFS Conference on Progress in Vehicle Aerodynamics and Thermal Management: Numerical Methods*. Renningen: Expert–Verlag, 2006, pp. 111–123.

[28] A. Hammer, H. Hammer, and K. Hammer, *Physikalische Formeln und Tabellen*. München: Lindauer, 2009.

[29] W. Ren, J. Reutzsch, and B. Weigand, "Direct Numerical Simulation of Water Droplets in Turbulent Flow," *Fluids*, vol. 5, no. 3, p. 158, Sep. 2020, https://www.mdpi.com/2311-5521/5/3/158.

[30] G. W. Petty, *A First Course in Atmospheric Radiation*, 2nd ed. Madison, Wis: Sundog Pub, 2006, https://www.patarnott.com/atms749/pdf/ch12PettyScattering.pdf.

[31] K. Perlin, "Noise Hardware," in *SIGGRAPH 2002 Course 36 Notes. Real-Time Shading Languages*. Marc Olano, 2001, https://www.csee.umbc.edu/~olano/s2002c36/ch02.pdf.

[32] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[33] S. Mueller, D. Hospach, 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 (MBMV)*, 2015, pp. 145–154.

[34] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767 [cs]*, Apr. 2018, http://arxiv.org/abs/1804.02767.