Simulating Photo-realistic Snow and Fog on Existing Images for Enhanced CNN Training and Evaluation

Alexander von Bernuth, Georg Volk and Oliver Bringmann
University of Tübingen, {bernuth, volkg, bringmann}@informatik.uni-tuebingen.de

***Abstract*— Verification and robustness testing of machine learning algorithms for autonomous driving is crucial. Due to the increasing complexity and quantity of those systems in a single vehicle, just driving the required distance with a newly developed vehicle is not feasible anymore: billions of hours on the street without failure are necessary to qualify for industry standards like ISO 26262. That is where simulation comes into play: machine learning algorithms are trained and evaluated on well known image data sets like KITTI or Cityscapes. But today's data sets mostly contain images taken under perfect weather conditions and therefore do not harden optical object detection algorithms against various weather conditions. This paper focuses on reusing these established and labeled data sets by augmenting them with adverse weather effects like snow and fog. Those effects are rendered physically correct and life like while being added to existing real world images. Thanks to easy parametrization the weather influences may be varied as necessary and allow for finely tuned learning and optimization processes. The weather effects are evaluated with regard to realism and impact on an established object detection algorithm. These newly created weather-influenced images may be used to validate or train new object detection algorithms.**

I. INTRODUCTION

Object detection plays a key role in autonomous driving applications. When failing, the self driving vehicle must come to a full stop or give control to the driver, who may be inattentive. While falling back to manual driving might be an option in level one to three autonomous driving, the overall goal is to achieve full autonomy. Even very frequent weather conditions like rain and fog disrupt today's autonomous vehicles on a daily basis – Waymo taxis, usually self driving, will have to rely on human assistants in rainy weather [1].

To combat this deficiencies numerous tests have to be passed under millions of different condition combinations. According to ISO 26262, a self driving system has to absolve 10^9 km of driving on the road with at most one failure [2]. Other research states that 10^8 km suffice depending on the quality of the tests [3]. Hoping to be able to drive, record and verify this enormous distance for every iteration of all components in such a vehicle is ludicrous. Waiting for every one of this situations happen just to check if a single traffic scenario gets mastered by the system is impossible.

Currently used image data sets like KITTI and Cityscapes are recorded largely under good weather conditions [4], [5]. Even data set which specifically try to cover all possible environmental influences, like [6], only contain very few images showing snowy roads and none showing falling snow.

This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG) in the priority program 1835 under grant BR2321/5-1.

This is where simulation comes into play. Both Software and Hardware in the Loop (SiL and HiL) are necessary in order to reduce the amount of real world tests required to qualify for ISO 26262. Traffic scenarios only become manageable when all vehicles are under control of one testing instance; let alone the millions of combinations existing when looking at possible variations of sunshine, rain, wind, fog, snow and more.

While some research was already done simulating rain or dust on real video data taken from a test vehicle, none created synthetic yet realistic correct fog or snow and examined their effects on object detection. This paper explores the possibilities of scene reconstruction in combination with physically correct simulation of fog and snow, comparing both effects with the real world and showing the degradation of object detection quality with growing intensity of the weather conditions. By utilizing existing data sets like KITTI or Cityscapes as well as simulators like Vires VTD or CARLA and newly created images and videos, we show the wide applicability of our approach.

First we examine research that touches the matter of weather simulation on images as well as snow and fog in particular. Then both snow and fog generation are presented in more detail beginning with the scene reconstruction and ending with the rendering in OpenGL. Finally, the simulated influences are tested for realism and impact on state of the art visual object detection algorithms.

II. RELATED WORK

Several attempts have been made to simulate different physical equipment and various external influences on video cameras. Hospach et al. described a camera model that is used to transform high quality video recordings into shots as if they were made with another camera [7]. This allows for one single test drive to be recorded and later morphed to all possible cameras that may be embedded in vehicle prototypes, eliminating the need for new test drives for every hardware iteration. Together with Mueller he created a framework that allows for environmental influences to be added as plugin, creating a simple way to test multiple variations thereof quickly [8]. Finally, they introduced a physically correct simulation of falling rain and examined the impact on lane detection algorithms [9]. However, only camera characteristics and the influence of rain were investigated, ignoring other mayor factors like snow and fog.

A very simple method to add snow into a scene is proposed by Wang and Wade [10]. It consists of a structure sur-

Fig. 1. Example of a street image before and after augmentation with both snow and fog. On the left half the image is shown unaltered, on the right side we simulated falling snow at a vehicle speed of $50\,\mathrm{km\,h^{-1}}$, a snow fall speed of $1\,\mathrm{m\,s^{-1}}$ and crosswinds of $1\,\mathrm{m\,s^{-1}}$ as well as light fog with the method proposed in this paper. Motion blur is visible especially towards the edges of the scene. Image depth was calculated from a pair of stereo images.

rounding the virtual camera, on which a transparent texture containing many 2D snowflakes is applied. After z-buffer-testing and blending, the scene looks like there is snow falling in front of the camera. Of course this method is rudimentary and not realistic, as neither depth nor falling speeds are considered. Another simplifying method was proposed by Zhuo and Libaicheng: they draw the falling snowflakes as billboards that always point towards the observer [11]. Thus they are visible at all times, but lack the realism of rotating and flat flakes. Aleshin et al. built a ski simulator that accounts for falling snow as obstruction for the athlete [12]. Lv and Liu describe a method to model snow flakes very simplified based on wind and gravity, but do not render a scene [13]. Various authors examine the way snow falls and covers the ground as well as objects that are present [14]–[17].

While their research is thorough, none paid close attention to the realism of the flakes while they are falling. Additionally none of the works mentioned above is applicable on images that already exist, not to mention real world scenes.

This gets addressed by Langer and Zhang, who use a trick to render noise as if it was falling snow: they apply a Fast Fourier Transform on an existing image, add noise in the frequency domain and the inverse the transform [18]. This leads to snow-like artifacts with respect to "depth" – but no real distance is calculated in the process. Jaeger tried adding a snow cover on existing images as well, coloring every pixel that is closer to the camera as the pixel above it white [19]. This results in interesting however unrealistic images.

Regarding fog much more research has been carried out. Examples for very basic virtual fog simulation can be found from Sellers or Aleshin [12], [20] who approximate fog using a single equation, not taking anything else but the distance from the camera to the objects into account. Even modern works like the one of Sakaridis et al. use very simple light attenuation formulas [21]. While Zdrojewska uses Perlin noise to simulate different fog densities throughout the scene, Chen evaluates fog integrals for light rays traveling in the scene and Biri and Michelin even involve wind in their calculations [22], [23]. To get even more advanced, multiple other authors introduce volumetric fog in their render approaches [24], [25]. Finally, Dumont or Jensen follow Monte-Carlo-driven methods in which they shoot multiple rays per pixel into a scene filled with fog in order to have results that are virtually realistic [26], [27].

Maximum realism is only achieved by Colomb et al. [28]. They built a full sized fog chamber where they can simulate real fog at all lighting conditions and any still scenario. Using real vehicles and pedestrians, all measurements translate perfectly to the real world. Unfortunately, moving scenarios as well as other environmental influences are impossible to simulate.

While much research was done on isolated topics like snow cover formation fall paths of single snowflakes, fog rendering in virtual scenes or volumetric lighting effects, none of the authors mentioned above investigated the possibility to render snow or fog realistically onto existing images. Also, no research was done to examine the influence of those simulated weather conditions on object detection.

This paper explores the ways in which varying fog and snow can be rendered physically correct on real scenes (see Fig. 1) and how these conditions impact object detection. Integrated in a simulation and testing framework, the proposed methods allow for fast function tests on newly created or recently updated perception algorithms.

III. SNOW GENERATION

Rendering the snow flakes requires multiple steps. First, the 3D scene has to be reconstructed in OpenGL using either stereo images and disparity matching or a single image with depth information from a different origin. Into this scene virtual snow flakes are distributed according to physical and meteorological principles. Finally, OpenGL is used to render the scene together with the snow flakes and generates a realistic image. Motion blur, caused by the flakes motion relative to the viewer (due to wind, gravitation and vehicle speed), as well as light attenuation are respected.

A. Scene Reconstruction

Introducing snow into an existing real scene requires the reconstruction of that scene – otherwise depth effects and occlusion of both snow flakes and road objects will not unfold.

Rebuilding a scene requires depth information, either from stereo images, lidar or other sensors. Stereo vision calculates the depth based on two images taken by cameras next to each other. After camera calibration every pair of images can be processed and the distance of each pixel to the camera can be determined.

Both Cityscapes and KITTI data sets provide stereo images (Cityscapes even comes with precomputed depth maps) that are suitable for 3D reconstruction.

Lidar or radar sensor information may be used to determine the distance of objects in front of a vehicle as well – but the point clouds representing the environment have to be mapped into view space, which can be challenging.

Especially synthetic images generated by simulators like Vires VTD or CARLA provide perfect depth information [29], [30]. Although those simulations come with some weather models, neither provides physically correct snow nor fog or their influences on the optical sensor.

After obtaining the depth map for a given image, reconstruction is a matter of trigonometry. Each pixel gets projected from an image plane into the 3D space. At that point in space a quad (a rectangle made out of two triangles), parallel to the image plane, is created and colored according to the original pixel color. Each quad is scaled so that if rendered it is visible in exactly one pixel in the rendered image. Hence, when rendering all quads at once the resulting rendered image is exactly the same as the original image but is now ready for 3D manipulations.

B. Snow Distribution

With the 3D scene prepared the next step is to place up to many billion single snow flakes in front of the camera. To limit the execution time to a reasonable level, some restrictions have to be made in terms of the maximum number of particles. Since single snow flakes become indistinguishable to an optical sensor as soon as they are far enough away, only a limited space in front of the camera has to be populated with flakes. Light attenuation effects that are caused by the consolidated snow masses are taken care of in a later step. Depending on the general motion vector of the snow, the space behind and left and right of the camera may be ignored as well.

The single parameter describing all properties of falling snow directly and indirectly is the snow fall rate (precipitation, R_s [$\mathrm{mm\,h^{-1}}$]). Once set, the snow mass concentration in the air (M_s [$\mathrm{g\,m^{-3}}$]) can be calculated according to Koh and Lacombe [31]:

$$M_s = 0.30 \cdot R_s \tag{1a}$$

$$M_s = 0.47 \cdot R_s \tag{1b}$$

Equation (1a) comes into play when the snow is “dense” (e.g. in snow storms) whereas (1b) describes regular snow.

Together with the mean mass of a single snow flake (0.2 g, [32]) this density can be converted into the number of flakes per volume (N_s [$\mathrm{m^{-3}}$]):

$$N_s = \frac{M_s}{0.2\,\mathrm{g}} \tag{2}$$

Using this ratio we populate a cuboid, placed in front of the camera, with snow flakes. The cuboid’s dimensions are arbitrarily chosen and primarily control render time. It has to be sufficiently deep so that the farthest flakes are not distinguishable anymore. Assuming that all particles are distributed uniformly simple random numbers suffice to set all necessary coordinates. One could use a frustum instead a cuboid to save some render time.

With the flakes’ positions set, in the next step their size has to be determined. As not all snow flakes share the same diameter, a distribution has to be applied. According to Gunn and Marshall (and corrections from Skehon and Srivastava), snow flake size may be calculated from the precipitation and is described by an exponential formula [33], [34]. Let D be the snow flake diameter in [cm] and R the snow precipitation in [$\mathrm{mm\,h^{-1}}$], then the relative frequency of a flake with that diameter N_D can be calculated as follows:

$$N_D = N_0 \cdot e^{-\Lambda D} \tag{3a}$$

$$N_0 = 2.50 \times 10^3 \cdot R^{-0.94} \qquad [\mathrm{m^{-1}\,m^{-3}}] \tag{3b}$$

$$\Lambda = 22.9 \cdot R^{-0.45} \qquad [\mathrm{cm^{-1}}] \tag{3c}$$

Using a piece-wise defined weighted probability distribution function (weighted by N_D), an appropriate diameter is assigned to each generated snow flake.

C. Rendering Snow Flakes

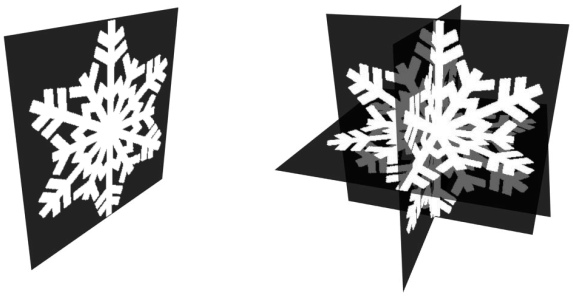
Having determined size and position of each pixel and snowflake, those parameters are fed into an OpenGL render pipeline. Additionally, the pixel colors as well as snow flake textures are sent to the graphics card.

Finally, the geometric properties of a snowflake are defined: we either assume that snow flakes are flat crystals and render them as quads, or we view them as thick aggregated flakes and render them as three pairwise perpendicular quads [14]. Each quad is assigned a texture, see Fig. 2.

In the snow shader all flakes are rotated randomly and three velocity vectors (wind, gravity, car) together with a time stamp determine the temporal position of each flake. An additional random vector is added to simulate turbulence in the air, its length based on the velocity vector.

In the next step all pixels and all snow flakes are rendered from far to near in order to account for some OpenGL opacity pitfalls. Utilizing super-sampling, aliasing is mitigated.

As video or photo cameras always open their aperture for a short but not negligible amount of time to take a picture, motion blur has to be considered. We use knowledge about the camera setup to determine the exposure time or fall back to a healthy default of 16.7 ms (≈ 60 fps). This time frame is divided into a customizable number of inter-frames (30 is sufficient for Full HD resolution). For each inter-frame all snow flakes are repositioned according to their



(a) Light Snow (b) Heavy Snow

Fig. 2. Models of a single snow flake, depending on the precipitation rate: heavy snow rather has big flakes while light snow tends to consist of thin flakes. A flake texture is added to each quad instead of modelling the flakes exactly to dramatically reduce the number of vertices to draw. The black quads are only drawn for visibility reasons in this figure and usually are fully transparent. Texture image from pngimg.com (CC BY-NC 4.0).

motion vectors and the current time stamp, then the scene is rendered. Weighted addition of each inter-frame then leads to realistic motion blur.

IV. FOG GENERATION

The second environmental phenomenon we simulate is fog. Fog can dramatically reduce visibility and lead to accidents even for experienced drivers, let alone self driving vehicles.

While theoretically no 3D reconstruction is needed to simulate fog (without using extensive rendering techniques like volumetric particles), we implemented our light attenuation algorithms in the pixel shader known from the snow simulation. This brings the benefits of massive parallelism at no extra cost or effort. For that, physical basics and Mie scattering (how light scatters on small drops) have to be revisited first before light falloff can be simulated correctly.

A. Physical Basics

When light traverses fog, which is just very fine water spray, the rays are affected by those very small water drops. Either they do not intersect with any particles and reach the viewer without any interaction, or they are absorbed completely, or scattered in a new direction with a new wavelength [35].

Simulating trillions of fog water particles and the corresponding light rays as well as their interactions is, while possible, terribly time consuming. Instead, one can assume that every light ray passes through a fixed number of fog particles per distance traveled, depending on the fog density. Additionally, different fog densities could occur in the 3D space.

While travelling through the fog, a certain amount of light is absorbed or scattered – thus extincted and therefore not reaching the observer's eye. This phenomenon can be very simply described by

$$I_t = I_i e^{-\alpha_{ext} d} \tag{4}$$

where I_t denotes the intensity of the light reaching the observer, I_i the incident light intensity, α_{ext} ($[\mathrm{m}^{-1}]$) an extinction factor and d ([m]) the distance the light travels through the fog.

Very simple extinction can be tied to the meteorological measure of visibility V as the distance a human can distinguish between a black object and the sky [36]:

$$0.05 = e^{-\alpha_{ext} V}, \tag{5a}$$

$$\alpha_{ext} \approx \frac{3}{V}. \tag{5b}$$

The value of 0.05 is often assumed to be the contrast threshold a human observer has in this scenario. With this simple formula a given visibility (distance) can be chosen and the appropriate extinction coefficient can be calculated. Given an image and a sky color I_s, each pixel with depth d is then assigned its new color

$$I = I_i e^{-\alpha_{ext} d} + I_s(1 - e^{-\alpha_{ext} d}). \tag{6}$$

B. Mie Scattering

While this basic blending with a fog color results in presentable images [21], real fog affects every light wavelength (λ, [nm]) differently instead of applying the same effects on all three RGB channels.

More specifically, each image color channel gets extincted according to its own special α_{ext}, depending on fog particle radius (r [µm]) and concentration in the air (d [$\mu\mathrm{m}^{-3}$]). This is covered by the Mie theory [37] (which would also apply to snow if we examined the extinction of infrared light [38]).

The extinction efficiency for each wavelength Q_e is a function of $x = 2\pi r/\lambda$ and translates directly to the extinction coefficient

$$\alpha_{ext} = Q_e(x) \cdot \pi r^2 d. \qquad [\mathrm{m}^{-1}] \tag{7}$$

C. Rendering Fog

Using the work of Wang and Jacques [39], we are able to calculate the necessary α_{ext} for each of the R, G and B channel, assuming corresponding wavelengths of 650, 550 and 450 nm. In the shaders attached to the environmental scenes these values are then combined with the distance of each pixel to the camera. The pixel colors are blended accordingly with the color of the sky.

One could also just apply those steps on an image directly without using OpenGL scenes, but since we already are in fragment space we can just as well utilize the massive parallelity of graphics cards for this task.

V. RESULTS

As a first quality assessment we took a series of photos of a scene containing a road and diverse depth during snowy weather. With an array of two cameras fixed to a metal profile bar – the cameras have been calibrated and all parameters for stereo imaging calculated – these photos could then be used to extract the depth of the scene. During autumn many images with diverse fog densities were taken. Over the course of the winter an additional data set of images with various snow intensities and wind directions as well as light situations was formed.

(a) Real Snow (b) Simulated Snow

Fig. 3. Visual comparison of real and simulated snow flakes. The images on the left were taken during snowy weather, on the right snow was added on images that were taken on days without any snow fall.

A. Comparison

In Fig. 3 we compare real snow flakes found on our images to flakes that were simulated onto images without any snow. While it was not only possible to recreate the general look of a snowy scene, we could also imitate single snowflakes and infer snow falling speed and flake size during this process. Even the combination of falling speed and crosswind speed could be determined only by manually approximating the synthetic image to the real image. Conversely, the simulated images share striking resemblance with the real images.

Reproducing foggy scenes from real images is more difficult as the images' white balance, lighting conditions and time of day have to match exactly. This lead to no exact match between our real fog images and images without any fog. Therefore we show an example from the Cityscapes data set on which we applied our fog (Fig. 4). Objects with greater distance to the camera are clearly faded to a larger extend than closer objects. Because even tiny imperfections in the depth map cause the generated fog to have very visible bright or dark spots, the quality of those maps is critical. Until depth maps improve fog simulation will be inconsistent.

B. Evaluation

To evaluate the impact of our simulated weather environment on state-of-the-art object detection we conducted a series of experiments. Using Recurrent Rolling Convolution (RRC) as introduced by Ren et al. [40], we chose a random subset (468 images which were not used for training) of the KITTI image data set [5] and examined the object detection average precision. Varying both the snow fall rate and the fog density automatically we created 69 different derived image sets and used the RRC to detect all trained objects.

Fig. 4. The upper image is from the Cityscapes image data set, the lower figure shows that image with our fog applied. Distant objects are harder to spot than closer ones, as they are affected more by the fog.

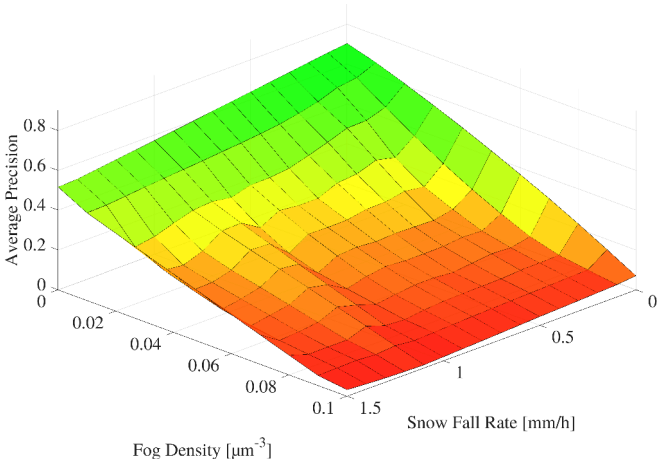


Fig. 5. CNN average detection precision which changes under the influence of our simulated snow and fog. With both increasing fog density and snow fall rate the average detection precision plummets from 70.5% to 3.7%. The plotted surface is interpolated from 69 sampled points.

Fig. 5 shows the significant loss of average precision with increasingly strong weather influences. Especially growing fog densities had a mayor impact on the detection rates, reducing average precision by 94.8%. Although the CNN is placed seventh in the KITTI object detection benchmark (as of April '19) we observe a huge drop in average recognition precision. This shows that CNNs would greatly benefit from training against those augmented images.

VI. CONCLUSIONS

This paper has argued that conventional testing of autonomous driving functions that are based on machine learning has reached the limit of feasibility due to the huge financial effort and expenditure of time. We proposed a new method to enhance existing image data sets with both realistic and variable snow and fog. The realism necessary to imitate adverse weather conditions for optical sensors was show by comparing real snow images with the ones our framework created. Object recognition rates were sig-

nificantly reduced when applying the weather effects on the images before the recognition process started. The insights gained from this experiments may lead to improved machine learning algorithms as their training data sets can be augmented with our methods. This aspect is addressed by many parallel research activities.

Next steps are going to include mounting a automotive-qualified stereo camera to a test vehicle and collecting more snow and fog data to compare to the simulated images. Furthermore, the training and evaluation of neural networks based on the newly created images might yield interesting findings. Finally, improving existing data set's depth maps might come in handy for better fog generation.

REFERENCES

[1] A. J. Hawkins, "We spoke to a Waymo One customer about how robot taxis get confused by rainstorms," https://www.theverge.com/2019/1/20/18175563/waymo-one-customer-interview-self-driving-arizona, Jan. 2019.

[2] A. Weitzel, H. Winner, C. Peng, S. Geyer, F. Lotz, and M. Sefati, *Absicherungsstrategien Fuer Fahrerassistenzsysteme Mit Umfeldwahrnehmung*. Fachverlag NW in der Carl Schuenemann Verlag GmbH, 2014, https://trid.trb.org/view/1339882.

[3] H. Winner, "Quo vadis, FAS?" in *Handbuch Fahrerassistenzsysteme*, H. Winner, S. Hakuli, F. Lotz, and C. Singer, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, pp. 1167–1186, http://link.springer.com/10.1007/978-3-658-05734-3_62.

[4] 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.

[5] 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.

[6] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, Jan. 2017.

[7] D. Hospach, S. Mueller, O. Bringmann, J. Gerlach, 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 (ITSC)*, 2014, pp. 2610–2615.

[8] 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.

[9] D. Hospach, S. Mueller, W. Rosenstiel, and O. Bringmann, "Simulation of Falling Rain for Robustness Testing of Video-Based Surround Sensing Systems," in *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 233–236.

[10] N. Wang and B. Wade, "Rendering falling rain and snow," in *ACM SIGGRAPH 2004 Sketches on - SIGGRAPH '04*, 2004, p. 14.

[11] T. Zhuo and Libaicheng, "A Simulation System of Snow Based on Particle System," in *International Conference on Intelligent Systems Research and Mechatronics Engineering (ISRME 2015)*, 2015, pp. 740–743.

[12] V. Aleshin, V. Afanasiev, A. Bobkov, S. Klimenko, V. Kuliev, and D. Novgorodtsev, "Visual 3D Perception of Motion Environment and Visibility Factors in Virtual Space," in *Transactions on Computational Science XVI*, M. L. Gavrilova and C. J. K. Tan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7380, pp. 17–33, http://link.springer.com/10.1007/978-3-642-32663-9_2.

[13] H. Y. Lv and F. Liu, "Real-Times Snowfall Simulation Based on Particle System and Pulverization," in *Applied Mechanics and Materials*, vol. 373-375, 2013, pp. 1168–1171.

[14] I. Saltvik, A. C. Elster, and H. R. Nagel, "Parallel methods for real-time visualization of snow," Ph.D. dissertation, 2006, http://dl.acm.org/citation.cfm?id=1775059.1775091.

[15] K. Muraoka and N. Chiba, "Visual simulation of snowfall, snow cover and snowmelt," in *Proceedings - 7th International Conference on Parallel and Distributed Systems: Workshops*, 2000, pp. 187–194.

[16] C. Wang, Z. Wang, T. Xia, and Q. Peng, "Real-time snowing simulation," *Visual Computer*, vol. 22, no. 5, pp. 315–323, 2006.

[17] N. V. Festenberg and S. Gumhold, "A geometric algorithm for snow distribution in virtual scenes," *Natural Phenomena*, pp. 17–25, 2009.

[18] M. S. Langer and L. Zhang, "Rendering falling snow using an inverse Fourier transform," *ACM SIGGRAPH technical sketches program*, 2003.

[19] M. Jaeger, "Enhancing virtual natural scenes using quick and dirty image based recipes," *Proceedings - 2012 IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications, PMA 2012*, pp. 164–171, 2012.

[20] G. Sellers, R. S. Wright, Jr., and N. Haemel, "Atmospheric Effects," in *OpenGL SuperBible: Comprehensive Tutorial and Reference, Seventh Edition*, 7th ed. Addison-Wesley Professional, July 2015.

[21] 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, Sept. 2018.

[22] D. Zdrojewska, "Real time rendering of heterogenous fog based on the graphics hardware acceleration," *Proceedings of CESCG*, vol. 4, pp. 95–101, 2004.

[23] V. Biri and S. Michelin, "Real Time Animation of Realistic Fog," in *Poster Session of 13th Eurographic Workshop on Rendering*, June 2002, p. 9.

[24] S. Raudsepp, "Volumetric Fog Rendering," Bachelor Thesis, University of Tartu, Tartu, 2018.

[25] R. V. Klassen, "Modeling the effect of the atmosphere on light," *ACM Transactions on Graphics*, vol. 6, no. 3, pp. 215–237, July 1987.

[26] E. Dumont, "Semi-Monte Carlo Light Tracing Applied to the Study of Road Visibility in Fog," in *Monte-Carlo and Quasi-Monte Carlo Methods 1998*, H. Niederreiter and J. Spanier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 177–187, http://link.springer.com/10.1007/978-3-642-59657-5_11.

[27] H. W. Jensen and P. H. Christensen, "Efficient simulation of light transport in scenes with participating media using photon maps," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, July 1998, pp. 311–320.

[28] M. Colomb, K. Hirech, P. André, J. Boreux, P. Lacôte, and J. Dufour, "An innovative artificial fog production device improved in the European project "FOG"," *Atmospheric Research*, vol. 87, no. 3-4, pp. 242–251, Mar. 2008.

[29] M. Dupuis and W. Karl, "VTD - VIRES Virtual Test Drive," https://vires.com/vtd-vires-virtual-test-drive/, 2017.

[30] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[31] G. Koh, J. Lacombe, and D. L. Hutt, "Snow mass concentration and precipitation rate," *Cold Regions Science and Technology*, vol. 15, no. 1, pp. 89–92, Feb. 1988.

[32] M. P. Langleben, "Terminal Velocity of Snowflakes," no. January, 1954.

[33] K. L. S. Gunn and J. S. Marshall, "The Distribution With Size of Aggregate Snowflakes," *Journal of Meteorology*, vol. 15, no. 5, pp. 452–461, 1958.

[34] R. S. Sekhon and R. C. Srivastava, "Snow Size Spectra and Radar Reflectivity," *Journal of the Atmospheric Sciences*, vol. 27, pp. 299–307, 1970.

[35] C. F. Bohren and D. R. Huffman, *Absorption and Scattering of Light by Small Particles*. Weinheim: Wiley-VCH, 2004, oCLC: 254937169.

[36] R. M. Rasmussen, J. Vivekanandan, J. Cole, B. Myers, and C. Masters, "The Estimation of Snowfall Rate Using Visibility," *Journal of Applied Meteorology*, vol. 38, no. 10, pp. 1542–1563, Oct. 1999.

[37] S. T. Surjikov, "Mie scattering," Feb. 2011.

[38] J. Pomeroy and D. Male, "Optical Properties of Blowing Snow," *Journal of Glaciology*, vol. 34, no. 116, pp. 3–10, 1988.

[39] L. Wang and S. L. Jacques, "Sphere Mie Scattering Program," University of Texas, Texas, Sept. 1995, https://omlc.org/software/mie/miesphr.c.

[40] J. S. J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate Single Stage Detector Using Recurrent Rolling Convolution," *CoRR*, vol. abs/1704.0, 2017, http://arxiv.org/abs/1704.05776.