Physics-Based Rendering for Improving Robustness to Rain

Shirsendu Sukanta Halder
Inria, Paris, France
shalder@cs.iitr.ac.in

Jean-François Lalonde
Université Laval, Québec, Canada
jflalonde@gel.ulaval.ca

Raoul de Charette
Inria, Paris, France
raoul.de-charette@inria.fr

https://team.inria.fr/rits/computer-vision/weather-augment/

Abstract

To improve the robustness to rain, we present a physically-based rain rendering pipeline for realistically inserting rain into clear weather images. Our rendering relies on a physical particle simulator, an estimation of the scene lighting and an accurate rain photometric modeling to augment images with arbitrary amount of realistic rain or fog. We validate our rendering with a user study, proving our rain is judged 40% *more realistic that state-of-the-art. Using our generated weather augmented Kitti and Cityscapes dataset, we conduct a thorough evaluation of deep object detection and semantic segmentation algorithms and show that their performance decreases in degraded weather, on the order of 15% for object detection and 60% for semantic segmentation. Furthermore, we show refining existing networks with our augmented images improves the robustness of both object detection and semantic segmentation algorithms. We experiment on nuScenes and measure an improvement of 15% for object detection and 35% for semantic segmentation compared to original rainy performance. Augmented databases and code are available on the project page.*

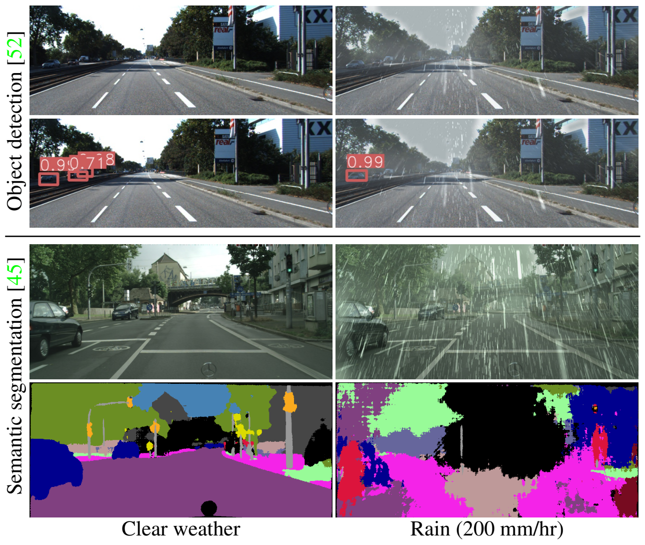


Figure 1. Our synthetic rain rendering framework allows for the evaluation of computer vision algorithms in these challenging bad weather scenarios. We render physically-based, realistic rain on images from the Kitti [20] (rows 1-2) and Cityscapes [8] (rows 3-4) datasets with object detection from mx-RCNN [52] (row 2) and semantic segmentation from ESPNet [45] (last row). Both algorithms are quite significantly affected by rainy conditions.

1. Introduction

A common assumption in computer vision is that light travels, unaltered, from the scene to the camera. In clear weather, this assumption holds: the atmosphere behaves like a transparent medium and transmits light with very little attenuation or scattering. However, inclement weather conditions such as rain fill the atmosphere with particles producing spatio-temporal artifacts such as attenuation or rain streaks. This creates noticeable changes to the appearance of images (see fig. 1), thus creating additional challenges to computer vision algorithms who must be robust to these conditions.

Most, if not all computer vision practitioners know that bad weather affect our algorithms. However, very few of us actually know *how much* it affects them. Indeed, how can one know what the impact of, say, a rainfall rate of 100 mm/hour (a typical autumn shower) will have on the performance of an object detector? Our existing databases all contain images overwhelmingly captured under clear weather conditions. To quantify this effect, one would need a labeled object detection dataset, where all the images have been captured under 100 mm/hour rain! Needless to say, such a "rain-calibrated" dataset does not exist, and capturing one would be prohibitive.

In this paper, we propose a method to realistically augment existing image databases with rainy conditions. Our method relies on well-understood physical models to generate visually convincing results. Our approach is the first to allow controlling the *amount* of rain in order to generate arbitrary amounts, ranging from very light rain (5 mm/hour rainfall) to very heavy storms (300+ mm/hour). This key feature allows us to produce weather-augmented datasets, where the rainfall rate is known and calibrated. Subsequently, we augment two existing datasets (Kitti [20] and Cityscapes [8]) with rain, and evaluate the robustness of popular computer

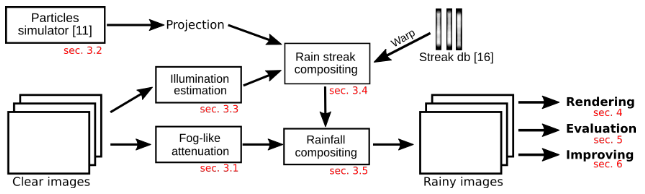


Figure 2. Overview of our weather augmentation pipeline for synthetically adding realistic rain in existing image databases.

vision algorithms on these augmented databases. We also use the latter to refine algorithms using curriculum learning [4] which demonstrate improved robustness on real rainy scenarios.

As opposed to the recent style transfer approaches [27, 18], which have demonstrated the ability to transfer weather [59], we use physical and photometric models to render all raindrops individually. Indeed, while these GAN-based approaches create visually appealing images, there is no guarantee they respect the underlying physics of weather, and thus cannot be used to estimate the performance of vision systems. What is more, controlling the *amount* of rain cannot easily be achieved with these techniques.

In short, we make the following contributions. First, we present a practical, physically-based approach to render realistic rain in images (fig. 2). Second, we augment Kitti [20] and Cityscapes [8] datasets with rain. Third, we present a methodology for systematically evaluating the performance of 12 popular object detection and semantic segmentation algorithms. Our findings indicate that most algorithms decrease on the order of 15% for object detection, and 60% for semantic segmentation. Finally, our augmented database is used to finetune object detection and segmentation architecture leading to significantly better robustness in real-world rainy/clear conditions.

2. Related Work

Rain modeling In their series of influential papers, Garg and Nayar provided a comprehensive overview of the appearance models required for understanding [17] and synthesizing [16] realistic rain. In particular, they propose an image-based rain streak database [16] modeling the drop oscillations, which we exploit in our rendering framework. Other streak appearance models were proposed in [51, 2] using a frequency model. Realistic rendering was also obtained with ray-tracing [44] or artistic-based techniques [48, 9] but on synthetic data as they requires complete 3D knowledge of the scene including accurate light estimation.

Rain removal Due to the problems it creates on computer vision algorithms, rain removal in images got a lot of attention initially focusing on photometric models [14]. For this, several techniques have been proposed, ranging from frequency space analysis [2] to deep networks [53]. Sparse coding and layers priors were also important axes of research [33, 35, 7] due to their facility to encode streak patches. More recently, Zhang et al. [56] proposed to use conditional GANs for this task. Alternatively, camera parameters [15] or programmable light sources [11] can also be adjusted to limit the impact of rain on the image formation process. Additional proposal were made for the specific of raindrop removals on windows [12] or windshields [22].

Weather databases In computer vision, few images databases have precise labeled weather information. Of note for mobile robotics, the BDD100K [54] and the Oxford dataset [36] provides data recorded in various weathers including rain. Other stationary cameras datasets such as AMOS [25], the transient attributes dataset [30], the Webcam Clip Art dataset [31], or the WILD dataset [38] are sparsely labeled with weather information. As of yet, there exists no dataset with systematically-recorded rainfall rates and object/scene labels. The closest systematic work in spirit [28] evaluated the effect of simulating weather on vision but in a virtual game GTA. Of particular relevance to our work, Sakaridis et al. [46] propose a framework for rendering fog into images from the Cityscapes [8] dataset. Their approach assumes a homogenous fog model, which is rendered from the depth estimated from stereo. Existing scene segmentation models and object detectors are then adapted to fog. In our work, we employ the similar idea of rendering realistic weather to existing images, but focus on rain rather than fog. We emphasize that rendering rain is significantly harder than fog, as it requires the accurate modeling of dynamics of rain drops and the radiometry of the resulting rain streaks as they get imaged by a camera. We also render fog, using a more realistic heterogeneous model.

3. Rendering rain into images

Rain is the result of moisture condensation at high altitude which creates raindrops (0.1-10 mm) falling at high speeds (up to 10 m/s) [37, 50]. The intensity of rain is measured as the rain fall average over an hour. As a reference, moderate rain is roughly 10 mm/hr and heavy rain greater than 50 mm/hr. In non-tropical regions, extreme rain fall rates 200+ mm/hr usually occur during a short period of time, usually on the order of a few minutes only.

Because of their large size relative to the light wavelength, the interaction with the light is complex. As opposed to fog [46], rainy events cannot be modeled as a volume and each drop physics must be modeled separately. In this section, we describe our rain rendering pipeline which requires an image, its associated per-pixel depth, and a desired rainfall rate. From this information, rain is rendered and blended with the original image (see fig. 1).

We make the distinction between two types of raindrops, based on their imaging size. First, when they are too far away, the accumulation of drops image within the pixel cone attenuates the light in a fog-like manner. When they are closer to the camera and thus larger, falling drops produce motion blurred streaks. We render these two effects separately.

3.1. Fog-like rain

We begin by rendering fog-like rain, which are the set of drops that are too far away and thus imaged on less than 1 pixel. In this case, a pixel may even be imaging a large number of drops, which causes optical attenuation [17]. In practice, most drops in a rainfall are actually imaged as fog-like rain[1], though their visual effect is less dominant.

We render volumetric attenuation using the model described in [51] where the per-pixel attenuation I_{att} is expressed as the sum of the extinction L_{ext} caused by the volume of rain and the airlight scattering A_{in} that results of the environmental lighting. Using equations from [51] to model the attenuation image at pixel $\mathbf{x}$ we obtain

$$I_{\text{att}}(\mathbf{x}) = I L_{\text{ext}}(\mathbf{x}) + A_{\text{in}}(\mathbf{x})\,, \text{ where} \tag{1}$$

$$\begin{aligned} L_{\text{ext}}(\mathbf{x}) &= e^{-0.312 R^{0.67} d(\mathbf{x})}\,, \text{ and} \\ A_{\text{in}}(\mathbf{x}) &= \beta_{\text{HG}}(\theta)\bar{E}_{\text{sun}}(1 - L_{\text{ext}}(\mathbf{x}))\,. \end{aligned} \tag{2}$$

Here, R denotes the rain fall rate R (mm/hr), $d(\mathbf{x})$ the pixel depth, β_{HG} the standard Heynyey-Greenstein coefficient, and $\bar{E}_{\text{sun}}$ the average sun irradiance which we estimate from the image-radiance relation [24].

3.2. Simulating the physics of raindrops

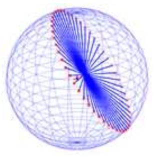
We use the particles simulator of de Charette et al. [11], which computes the position and dynamics of all raindrops for a given fallrate[2]. To reduce the algorithm complexity, only drops of 1 mm or more are individually simulated which is reasonable since smaller ones are unlikely to be visible. In particular, the simulator computes the position and dynamics (start and end points of streaks) of all the rain particles in both world and image space, and accounts for intrinsic and extrinsic calibration for image projection. We calibrate the simulator as to generate particles that cover the entire field of view of the camera.

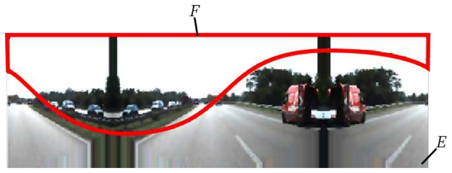
3.3. Rendering the appearance of raindrops

Visually, raindrops act as small spherical lenses imaging a wide portion of the scene. While it is possible to synthesize the exact photometry of a drop with ray casting, this comes at very high processing cost and is virtually only possible in

[1]Assuming a stationary camera with Kitti calibration, we computed that only 1.24% of the drops project on 1+ pixel in 50 mm/hr rain, and 0.7% at 5 mm/hr. This is logical, as the heavier the rain, the higher the probability of having large drops.

[2]The distribution and dynamics of drops varies on earth due to gravity and atmospheric conditions. We selected here the broadly used physical models recorded in Ottawa, Canada [37, 1].


(a) Drop FOV



(b) Projection on the environment map

Figure 3. To estimate the photometric radiance of each drop, we must integrate the lighting environment map over the 165° drop field of view (a). For this, we first estimate the environment map E from the current image using [6], and compute F the intersection of the drop field of view with the environment map (right).

synthetic scenes where the geometry and surface materials are perfectly known [44]. What is more, drops oscillate as they fall, which creates further complications in modeling the light interaction.

Instead, we rely on the raindrop appearance database of Garg and Nayar [17], which contains the individual rain streaks radiance when imaged by a stationary camera. For each drop the streak database also models 10 oscillations due to the airflow, which accounts for much greater realism than Gaussian modeling [2].

3.3.1 Projecting rain streak in the image

To render a raindrop, we first select a rain streak $S_j^k \in \mathcal{S}$ from the streak database $\mathcal{S}$ of Garg and Nayar [16], which contains $j = 20$ different streaks with $k = 10$ different oscillations stored in image format. To select the best rain streak for a particular drop, we pick the model j that best matches the final drop dimensions (computed from the output of the physical simulator), and randomly select an oscillation k.

The selected rain streak S (indices are dropped for clarity in notation) is subsequently warped to make it match the drop dynamics from the physical simulator:

$$S' = \mathcal{H}(S)\,, \tag{3}$$

where $\mathcal{H}(\cdot)$ is the homography computed from the start and end points in image space given by the physical simulator and the corresponding points in the trimmed streak image.

3.3.2 Computing the photometry of rain streak

Computing the photometry of a rain streak from a single image is challenging because drops have a much larger field of view than common cameras (165° vs approx. 70–100°). In other words, a drop refracts light that is not imaged by the camera, which means that, if we are to render a drop accurately, we must estimate the environment map (spherical lighting representation) around that drop. While this is physically infeasible to perform from a single image, we employ the method of [6] which approximates the environment map through a series of simple operations performed on the image itself.

From the estimated environment and the 3D drop position provided by the physical simulator, we compute the intersection F of the drop field of view with the environment map E, assuming a 10 m constant scene distance and accounting for the camera-to-drop direction. The process is depicted in fig. 3, and geometrical details are provided in supplementary.

Note that geometrically exact drop field of view estimation requires location-dependent environment maps, centered on each drop. However, we consider the impact negligible since drops are relatively close to the camera center compared to the sphere radius used[3].

Following [17] which states that a drop refracts 94% of its field of view radiance and reflects 6% of the entire environment map radiance, we multiply the streak appearance with a per-channel weight:

$$S' = S'(0.94\bar{F} + 0.06\bar{E})\,, \tag{4}$$

where $\bar{F}$ is the mean of the intersection region F, and $\bar{E}$ is the mean of the environment map E. Here, solid angles ω of the latitude-longitude representation are taken into account when computing the mean, i.e.: $\bar{F} = \sum_{i \in F} F(i)\omega(i)$.

3.4. Compositing a single rain streak on the image

Now that the streak position and photometry were determined from the physical simulator and the environment map respectively, we can composite it onto the original image. First, to account for the camera depth of field, we apply a defocus effect following [39], convolving the streak image S' with the circle of confusion C[4], that is: $S' = S' * C$.

We then blend the rendered drop with the attenuated background image I_{att}, using the photometric blending model from [17]. Because the streak database and the image I are likely to be imaged with different exposure, we need to correct the exposure so as it reflects the imaging system used in I. Suppose $\mathbf{x}$ a pixel of the image I and $\mathbf{x}'$ the overlapping coordinates in streak S', the result of the blending is obtained with

$$I_{\text{rain}}(\mathbf{x}) = \frac{T - S'_{\alpha}(\mathbf{x}')\tau_1}{T} I_{\text{att}}(\mathbf{x}) + S'(\mathbf{x}')\frac{\tau_1}{\tau_0}\,, \tag{5}$$

where S'_{α} is the alpha channel[5] of the rendered streak, $\tau_0 = \sqrt{10^{-3}}/50$ is the time for which the drop remained on one pixel in the streak database, and τ_1 the same measure according to our physical simulator. We refer to the supplementary materials for details.

[3]We computed that, for Kitti, 98.7% of the drops are within 4 m from the camera center in a 50 mm/hr rainfall rate. Therefore, computing location-dependent environment maps would not be significantly more accurate, while being of very high processing cost.

[4]The circle of confusion C of an object at distance d, is defined as: $C = \frac{(d-f_{\text{p}})f^2}{d(f_{\text{p}}-f)f_{\text{N}}}$ with f_{p} the focus plane, f the focal and f_{N} the lens f-number. f and f_{N} are from intrinsic calibration, and f_{p} is set to 6 m.

[5]While [16] does not provide an alpha channel, the latter is easily computed since drops were rendered on black background in a white ambient lighting.

3.5. Compositing rainfall on the image

The rendering of rainfall of arbitrary rates in an image is done in three main steps: 1) the fog-like attenuated image I_{att} is rendered (eq. 1), 2) the drops output by the physical simulator are rendered individually on the image (eq. 5), 3) the global luminosity average of the rainy image denoted I_{rain} is adjusted. While rainy events usually occur in cloudy weather which consequently decreases the scene radiance, a typical camera imaging system adjusts its exposure to restore the luminosity. Consequently, we adjust the global luminosity factor so as to restore the mean radiance, and preserves the relation $\bar{I} = \bar{I}_{\text{rain}}$, where the overbar denotes the intensity average.

4. Validating rain appearance

We now validate the appearance of our synthetic rain photometrically, visually and quantify the perceptual realism by comparing it to existing rain augmentation databases.

Photometric validation. We first evaluate the impact of warping the background image to model the lighting environment around a drop. For that matter, in fig. 4 we compare rain rendered with our pipeline using the estimated environment map (sec. 3.3.2) with the ground truth illumination obtained from high dynamic range panoramas [23]. Overall, while this is clearly an approximation, we observe that rendering rain with this approximation closely match the results obtained with the ground truth lighting conditions. This is especially true when the scene is symmetrical (top image).

Qualitative validation. Fig. 5 presents real photographs of heavy rain, and qualitative results of our rain rendering and representative results from 3 recent synthetic rain augmented databases [55, 53, 56]. From our rendering, the streaks have consistent orientation with the camera motion and consistent photometry with background and depth (sec. 5.1.1 for details). As in the real photographs our streaks are sparse and only visible on darker background. Note also that the attenuation caused by the rain volume (i.e. fog-like rain) is visible where the scene depth is large (i.e. image center, sky) and that close streaks are accurately defocused. As opposed to other rain rendering, our pipeline simulates rain given real rainfall (mm/hr) whereas existing methods use arbitrary amount of rain that has no physical correspondence. This is important as we intend to study the effect of rain on computer vision.

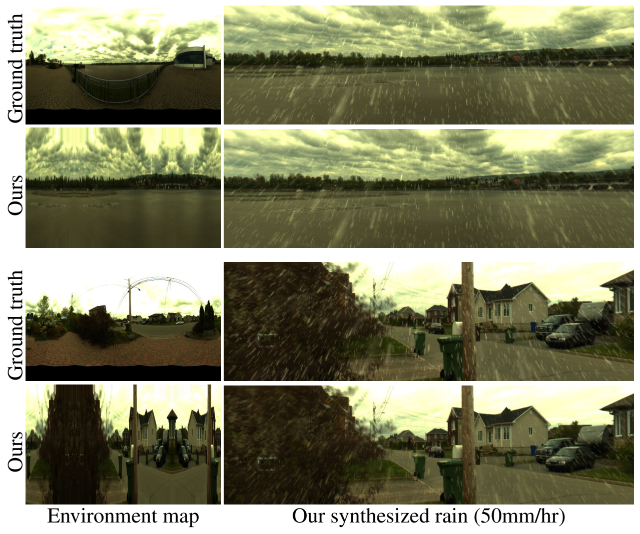


Figure 4. Comparison between rain rendering using ground truth illumination or our approximated environment map. From HDR panoramas [23], we first extract limited field of view crops to simulate the point of view of a regular camera. Then, 50mm/hr rain is rendered using either (rows 1, 3) the ground truth HDR environment map or (rows 2, 4) our environment estimation. The environments are shown as reference on the left. While our approximated environment maps differ from the ground truth, they are sufficient to generate visually similar rain in images.

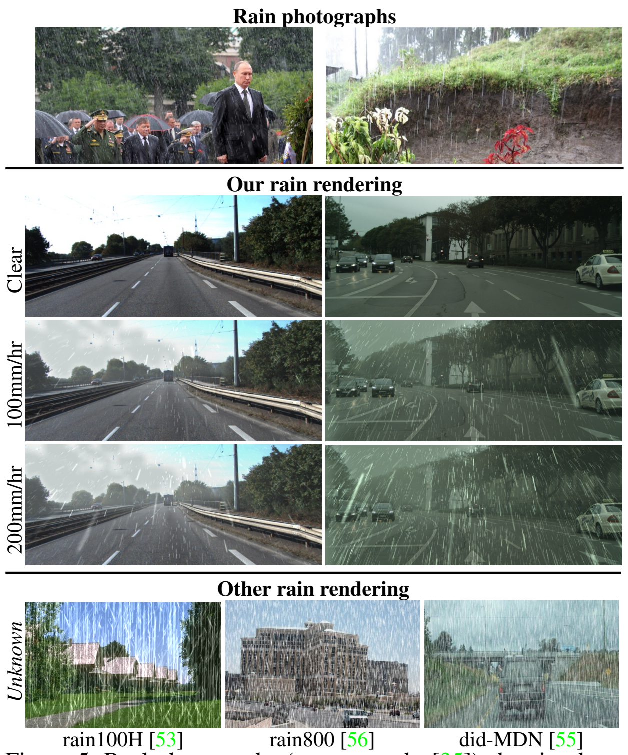


Figure 5. Real photographs (source: web, [35]) showing heavy rain, sample output of our rain rendering and other recent rain rendering methods. Although rain appearance is highly camera-dependent [15], results show that both real photographs and our rain generation share volume attenuation and sparse visible streaks which correctly vary with the scene background. Opposed to the other rain renderings, our pipeline simulates physical rainfall (here, 100mm/hr and 200mm/hr) and valid particles photometry.

User study. We validate the perceptual quality of our synthesized rain through a Mean Opinion Score (MOS) user study with 35 participants which ages range from 19 to 46 (avg 25.9, std 6.7), with 40% females. Users were asked to rate if *rain looks realistic* on 30 randomly-selected images, using a 5-points Likert scale. Results are reported in Fig. 6 against best images from [53, 55, 56] and real rain photography (6 images were shown for each method, in randomized order). Our rain is judged to be significantly more realistic than the state-of-the-art. Converting ratings to the [0, 1] interval, the mean rain realism is 0.78 for real photos, 0.57 for ours and 0.41/0.31/0.12 for [55]/[[56]/[53], respectively.

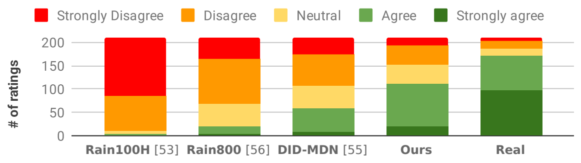


Figure 6. User study of rain realism. The y-axis displays ratings to the statement *Rain in this image looks realistic*. Our rain is closer to real rain ratings method and outperforms all other methods.

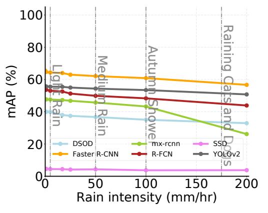
5. Studying the effects of rain

In this section, we present an evaluation of the robustness of popular computer vision algorithms on Kitti [20] and Cityscapes [8] to demonstrate the usefulness of our rain rendering methodology. In doing so, we benefit from available ground truth labels to quantify the impact of rain and fog on two important tasks: object detection and semantic segmentation. Both tasks are critical for outdoor computer vision systems such as mobile robotics. For a comprehensive study, we also render synthetic fog as in [29, 46] which we describe in the supplementary. We use the maximum visibility distance $V_{\max}$ to measure the fog intensity[6]. Unlike rain, fog is a steady weather that produces only a contrast attenuation function of the scene distance.

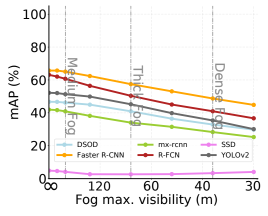
5.1. Methodology

We use the Kitti object benchmark [20] (7480 images) for object detection and Cityscapes [8] (2995 images) for segmentation, and evaluate 12 algorithms (6 per task) in 15 weather conditions. The original (clear) serves as a base-

[6]The Koschmieder law defines the maximum visibility in fog as: $V_{\max} = -\ln(C_T)/\beta$, where β is the fog optical extinction, and $C_T = 0.05$ is the minimum identifiable contrast for humans [40]. I.e., moderate/dense fog has $V_{\max}$ of 375 m ($\beta = .008$) and 37.5 m ($\beta = .08$), respectively.

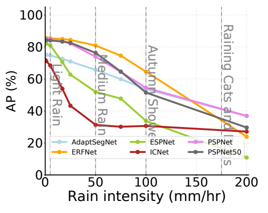


(a) Rain

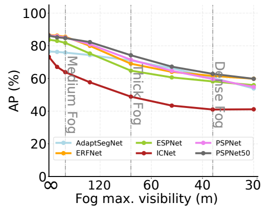


(b) Fog

Figure 7. Object detection performance on our weather augmented Kitti dataset as a function of rain fall rate (left) or fog visibility (right). The plots show the Coco mAP@[.1:.1:.9] accross car and pedestrians. While both fog and rain affect object detection, the effect of effect of fog is linear while rain is more chaotic.



(a) Rain



(b) Fog

Figure 8. Average Precision (AP) of the pixel-semantic prediction for clear and weather augmented Cityscapes dataset as a function of rain intensity (left) and fog extinction (right). This task is clearly more affected by the rain weather rather than the fog. This might be explained with the saliency patterns cause by rain streaks falling.

line to which we compare the performance of 14 additional weather augmentation (7 types of rain, and 7 type of fog). For rain, we render from light rain to heavy storm corresponding to rates $R = \{0, 1, 5, 17, 25, 50, 100, 200\}$ mm/hr, and for fog, $V_{\text{max}} = \{\infty, 750, 375, 150, 75, 50, 40, 30\}$ m. Kitti sequences are also used to demonstrate temporal performance in the supplementary video.

5.1.1 Datasets preparation

For realistic physical simulator (sec. 3.2) and rainstreak photometric (sec. 3.3.2), intrinsic and extrinsic calibration are used to replicate the imaging sensor. For Kitti, we use sequence-wise or frame-wise calibration [20, 19] with 6mm focal and 2ms exposure. As Cityscapes does not provide calibration, we use intrinsic from camera manufacturer with 5ms exposure and extrinsic is assumed similar to Kitti.
Our method also requires the scene geometry (pixel depth) to model accurately the light-particle interaction and the fog optical extinction. We estimate Kitti depth maps from RGB+Lidar with [26], and Cityscapes depth from RGB only with MonoDepth [21]. While perfect absolute depth is not required, a correct RGB-depth alignment is critical to avoid artifacts along geometrical edges. Thus depths are post-processed with a guided-filter [3] for better RGB-depth alignment.

5.1.2 Bad weather simulation

We mimick the camera ego motion in the physical simulator to ensure realistic rainstreak orientation on still images and preserve temporal consistency in sequences. Ego speed is extracted from GPS data when provided (Kitti sequences), or drawn uniformly in the $[0, 50]$ km/hr interval for Cityscapes semantics and in the $[0, 100]$ km/hr interval for Kitti object to reflect the urban and semi-urban scenarios, respectively.

5.2. Object detection

We evaluate the 15 augmented weathers on Kitti for 6 car/pedestrian pre-trained detection algorithms (with IoU $\geq$ .7): DSOD [47], Faster R-CNN [42], R-FCN [10], SSD [34], MX-RCNN [52], and YOLOv2 [41]. Quantitative results for the Coco mAP@[.1:.1:.9] accross classes are shown in fig. 7. Relative to their clear-weather performance the 200 mm/hr rain is always at least 12% worse and even drops to 25-30% for R-FCN, SSD, and MX-RCNN, whereas Faster R-CNN and DSOD are the most robust to changes in fog and rain. Representative qualitative results are shown in fig. 9 for 4 out of 6 algorithms to preserve space. We observe that, unlike fog, rain has a chaotic effect on object detection results whereas fog seems to affect the performance linearly with the depth. While all algorithms get affected by rain, when objects are large and facing the camera, most algorithms can still detect them.

5.3. Semantic segmentation

For semantic segmentation, the 15 weather augmented Cityscapes is evaluated for: AdaptSegNet [49], ERFNet [43], ESPNet [45], ICNet [57], PSPNet [58] and PSPNet(50) [58]. Quantitative results are reported in fig. 8 for both rain (a) and fog (b). As opposed to object detection algorithms which demonstrated significant robustness to moderately high rainfall rates, here the algorithms seem to breakdown in similar conditions. Indeed, all techniques see their performance drop by a minimum of 30% under heavy fog, and almost 60% under strong rain. Interestingly, some curves cross, which indicates that different algorithms behave different under rain. ESPNet for example, ranks among the top 3 in clear weather but drops relatively by a staggering 85% and ranks last in when it is raining cats and dogs (200 mm/hr). Corresponding qualitative results are shown in fig. 10 for 4 out of 6 algorithms to preserve space. Although the effect of rain may appear minimal visually, it greatly affects the output of all segmentation algorithms evaluated.

6. Improving robustness to rain

Using our rain-rendered database, we now demonstrate its usefulness for improving robustness to rain through extensive evaluations on synthetic and real rain databases.

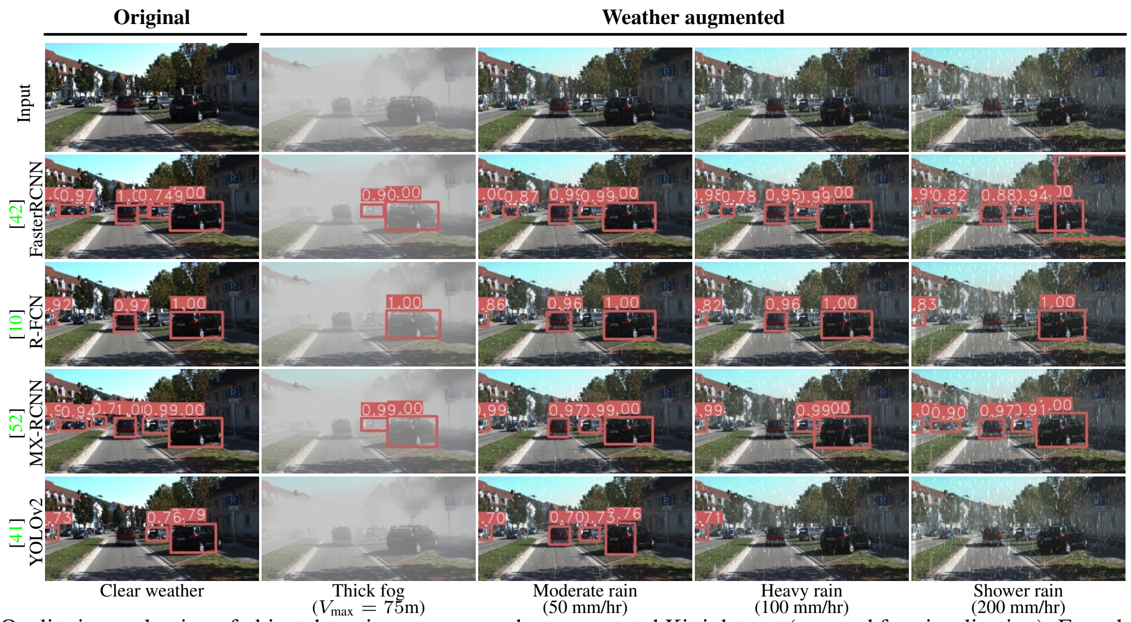


Figure 9. Qualitative evaluation of object detection on our weather augmented Kitti dataset (cropped for visualization). From left to right, the original image (clear) and five weather augmented images.

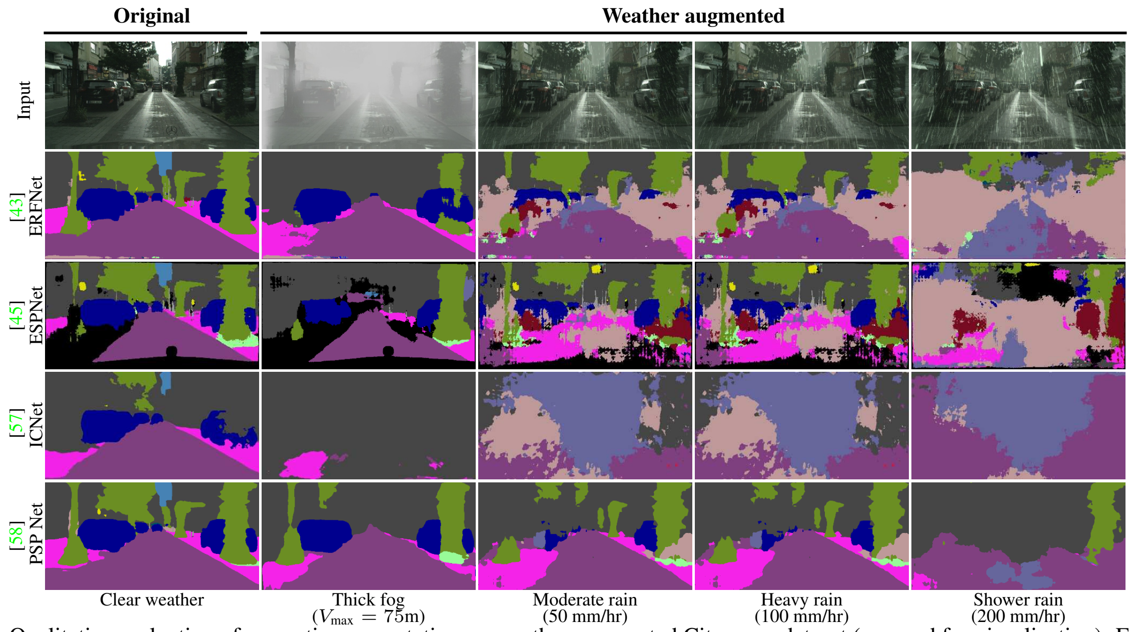
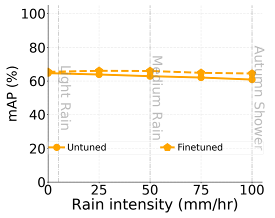


Figure 10. Qualitative evaluation of semantic segmentation on weather augmented Cityscape dataset (cropped for visualization). From left to right, the original image (clear) and five weather augmented images.

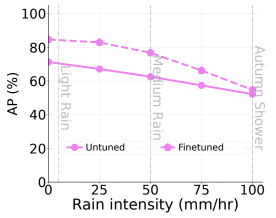
6.1. Methodology

We select Faster-RCNN [42] for object detection and PSP-Net [58] for semantic segmentation due to their public train implementation and good performance. While the ultimate goal is to improve robustness to rain, we aim at learning a model showing robustness to both clear weather and large variety of rains. Because rain significantly alters the appearance of the scene, we found that training from scratch with rain fails to converge. Instead, we refine our *untuned* models using curriculum learning [4] on rain intensity in ascending order (25, then 50, then 75 and finally 100mm/hr rain). The final model is referred as *finetuned* and is evaluated against various weather conditions.

Each of the 4 refinement passes uses 1000 images of the corresponding rain fallrate, and trains for 10 epochs with 0.0004 learning rate and 0.9 momentum.



(a) Object detection mAP (%)



(b) Semantic segmentation AP (%)

Figure 11. Performance on synthetic data for object detection (Faster-RCNN [42]) and semantic segmentation (PSPNet [58]) when finetuned with our rendering pipeline. Both finetuned models show increased robustness to rain events and clear weather.

	Clear	Rainy
Untuned	19.5	10.1
Finetuned (ours)	**20.1**	**11.6**

(a) Object detection mAP (%)

	Clear	Rainy
Untuned	**40.8**	18.7
Finetuned (ours)	39.0	**25.6**

(b) Semantic segmentation AP (%)

Figure 12. Real rain performance on nuScenes [5] datasets for object detection (Faster-RCNN [42]) and semantic segmentation (PSPNet [58]).

6.2. Synthetic performance

The synthetic evaluation is conducted on our augmented databases using 1000 versatile unseen images, with either no-rain (clear) or rain up to 200mm/hr. Fig. 11 shows the performance of our *untuned* and *finetuned* model for object detection (Faster-RCNN [42]), and semantic segmentation (PSPNet [58]). We observe an obvious improvement in both tasks and additional increase in robustness even in the clear weather when refined using our augmented rain. The intuition here is that when facing adverse weather, the network learns to focus on relevant features for both tasks and thus gain robustness. For Faster-RCNN, the finetuned detection performance is nearly constant in the 0-100mm/hr. Explicitly, it drops to 64.5% whereas the untuned model drops to 60.9%. We also tested on stronger (unseen) 200mm/hr fallrate and our finetuned Faster-RCNN got an mAP of 62.4% (versus 55.4% when untuned). For PSPNet, the segmentation exhibits a significative improvement when refined although at 100mm/hr the model is not fully able to compensate the effect of rain and drops to 54.0% versus 52.0% when untuned.

6.3. Real rain performance

We use the recent NuScenes dataset [5] which provides coarse weather meta-data and evaluate object detection and segmentation when *untuned* or *finetuned*. Since meteorological stations only report hour average, precipitation in mm/hr cannot be retrieved frame-wise so we cluster frames into clear and rainy. For objects, we study the mAP on 2000 nuScenes images (1000 clear + 1000 rainy). For segmentation, since semantic labels are not provided we evaluate AP on 50 images (25+25) which we carefully annotated ourselves. There is here a large domain gap between training (Kitti/Cityscapes with our synthetic rain) and evaluation (nuScenes with real rain). Still, Fig. 12 shows that ours *finetuned* leads to performance increase in all real rainy scenes for both object (+14.9%) and semantic tasks (+36.6%). In clear weather ours *finetuned* performs on par with the *untuned* version. This demonstrates the usefulness of our physics-based rain rendering for real rain scenarios.

More comparative and qualitative results are present in the supplementary.

7. Discussion

In this paper, we presented the first intensity-controlled physical framework for augmenting existing image databases with realistic rain.

Limitations. While we demonstrated highly realistic rain rendering results, our approach still has limitations that set the stage for future work. The main limitation is the approximation of the lighting conditions (sec. 3). While it was empirically determined our environment map yielded reasonable results (fig. 4), it may under/over estimate the scene radiance when the sky is not/too visible. This approximation is more visible when streaks are imaged against a darker sky.

Second, we make explicit distinction between fog-like rain and drops imaged on more than 1 pixel, individually rendered as streaks. While this distinction is widely used in the literature [17, 15, 11, 32] it causes an inappropriate sharp distinction between fog-like and streaks. A possible solution would be to render *all* drops as streaks weighting them as a function of their imaging surface. However, our experiments show it comes at a prohibitive computation cost.

Finally, our rendering framework does not model wet surface and the splashes rain makes on surfaces [13]. Properly modeling these effects would require richer illumination and 3D information about the scene, including the location and nature of all surfaces. Nevertheless, it is likely that as scene geometry estimation techniques progress [21], rendering more and more of these effects will be achievable.

Usage by the community. Our framework is readily usable to augment existing image with realistic rainy conditions. The weather augmented Kitti and Cityscapes, as well as the code to generate rain and fog in arbitrary images/sequences is available on the project page.

Acknowledgements

This work was partially supported by the *Service de coopération et d'action culturelle du Consulat général de France à Québec* with the grant Samuel de Champlain. We gratefully thank Pierre Bourré for his priceless technical help, Aitor Gomez Torres for his initial input, and Srinivas Narasimhan for letting us reuse the physical simulator.

References

[1] David Atlas, RC Srivastava, and Rajinder S Sekhon. Doppler radar characteristics of precipitation at vertical incidence. *Reviews of Geophysics*, 11(1):1–35, 1973. 3

[2] Peter C Barnum, Srinivasa Narasimhan, and Takeo Kanade. Analysis of rain and snow in frequency space. *International Journal of Computer Vision*, 86(2-3):256, 2010. 2, 3

[3] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, 2016. 6

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pages 41–48. ACM, 2009. 2, 7

[5] Holger Caesar and et al. nuscenes: A multimodal dataset for autonomous driving. *preprint arXiv:1903.11027*, 2019. 8

[6] Christopher Cameron. Hallucinating environment maps from single images. Technical report, 2005. 3

[7] Yi-Lei Chen and Chiou-Ting Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In *IEEE International Conference on Computer Vision*, pages 1968–1975, 2013. 2

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 5

[9] Carles Creus and Gustavo A Patow. R4: Realistic rain rendering in realtime. *Computers & Graphics*, 37(1-2):33–40, 2013. 2

[10] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016. 6, 7

[11] Raoul de Charette, Robert Tamburo, Peter C Barnum, Anthony Rowe, Takeo Kanade, and Srinivasa G Narasimhan. Fast reactive control for illumination through rain and snow. In *International Conference on Computational Photography*, 2012. 2, 3, 8

[12] David Eigen, Dilip Krishnan, and Rob Fergus. Restoring an image taken through a window covered with dirt or rain. In *IEEE International Conference on Computer Vision*, pages 633–640, 2013. 2

[13] Kshitiz Garg, Gurunandan Krishnan, and Shree K Nayar. Material based splashing of water drops. In *Eurographics Conference on Rendering Techniques*, 2007. 8

[14] Kshitiz Garg and Shree K Nayar. Detection and removal of rain from videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004. 2

[15] Kshitiz Garg and Shree K Nayar. When does a camera see rain? In *IEEE International Conference on Computer Vision*, volume 2, pages 1067–1074. IEEE, 2005. 2, 5, 8

[16] Kshitiz Garg and Shree K Nayar. Photorealistic rendering of rain streaks. In *ACM Transactions on Graphics (SIGGRAPH)*, volume 25, pages 996–1002. ACM, 2006. 2, 3, 4

[17] Kshitiz Garg and Shree K Nayar. Vision and rain. *International Journal of Computer Vision*, 75(1):3–27, 2007. 2, 3, 4, 8

[18] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013. 6

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1, 2, 5, 6

[21] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 8

[22] Jad C Halimeh and Martin Roser. Raindrop detection on car windshields using geometric-photometric environment construction and intensity-based correlation. In *IEEE Intelligent Vehicles Symposium*, pages 610–615. IEEE, 2009. 2

[23] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 4, 5

[24] Berthold Horn, Berthold Klaus, and Paul Horn. *Robot vision*. MIT press, 1986. 3

[25] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 2

[26] Maximilian Jaritz, Raoul de Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with CNNs: Depth completion and semantic segmentation. In *International Conference on 3D Vision*, 2018. 6

[27] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. 2

[28] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *International Conference on Robotics and Automation*, 2016. 2

[29] Sule Kahraman and Raoul de Charette. Influence of Fog on Computer Vision Algorithms. Research report, Inria Paris, Sept. 2017. 5

[30] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (SIGGRAPH)*, 33(4), 2014. 2

[31] Jean-François Lalonde, Alexei A Efros, and Srinivasa G Narasimhan. Webcam clip art: Appearance and illuminant transfer from time-lapse sequences. In *ACM Transactions on Graphics (SIGGRAPH Asia)*, volume 28, page 131, 2009. 2

[32] Ruoteng Li, Robby T. Tan, and Loong-Fah Cheong. Robust optical flow in rainy scenes. In *European Conference on Computer Vision*, September 2018. 8

[33] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. In *IEEE*

Conference on Computer Vision and Pattern Recognition, pages 2736–2744, 2016. 2

[34] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016. 6

[35] Yu Luo, Yong Xu, and Hui Ji. Removing rain from a single image via discriminative sparse coding. In *IEEE International Conference on Computer Vision*, pages 3397–3405, 2015. 2, 5

[36] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *International Journal of Robotics Research*, 36(1):3–15, 2017. 2

[37] John S Marshall and W Mc K Palmer. The distribution of raindrops with size. *Journal of meteorology*, 5(4):165–166, 1948. 2, 3

[38] Srinivasa G Narasimhan, Chi Wang, and Shree K Nayar. All the images of an outdoor scene. In *European Conference on Computer Vision*, 2002. 2

[39] Michael Potmesil and Indranil Chakravarty. A lens and aperture camera model for synthetic image generation. *ACM SIGGRAPH Computer Graphics*, 15(3):297–305, 1981. 4

[40] Ales Prokes. Atmospheric effects on availability of free space optics systems. *Optical Engineering*, 48(6):066001, 2009. 5

[41] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 7

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. 6, 7, 8

[43] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2018. 6, 7

[44] Pierre Rousseau, Vincent Jolivet, and Djamchid Ghazanfarpour. Realistic real-time rain rendering. *Computers & Graphics*, 30(4):507–518, 2006. 2, 3

[45] Mohammad Rastegari Sachin Mehta, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *European Conference on Computer Vision*, 2018. 1, 6, 7

[46] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, 126(9):973–992, Sep 2018. 2, 5

[47] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *IEEE International Conference on Computer Vision*, 2017. 6

[48] Natalya Tatarchuk. Artist-directable real-time rain rendering in city environments. In *ACM SIGGRAPH Courses*, pages 23–64. ACM, 2006. 2

[49] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 6

[50] John H van Boxel et al. Numerical model for the fall speed of rain drops in a rain fall simulator. In *Workshop on wind and water erosion*, pages 77–85, 1997. 2

[51] Yoann Weber, Vincent Jolivet, Guillaume Gilet, and Djamchid Ghazanfarpour. A multiscale model for rain rendering in real-time. *Computers & Graphics*, 50:61–70, 2015. 2, 3

[52] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 6, 7

[53] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1357–1366, 2017. 2, 4, 5

[54] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018. 2

[55] He Zhang and Vishal M Patel. Density-aware single image de-raining using a multi-stream dense network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 695–704, 2018. 4, 5

[56] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *arXiv preprint arXiv:1701.05957*, 2017. 2, 4, 5

[57] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *European Conference on Computer Vision*, 2017. 6, 7

[58] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 7, 8

[59] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. 2017. 2