Rain rendering for evaluating and improving robustness to bad weather

Maxime Tremblay · Shirsendu Sukanta Halder · Raoul de Charette · Jean-François Lalonde

Abstract Rain fills the atmosphere with water particles, which breaks the common assumption that light travels unaltered from the scene to the camera. While it is well-known that rain affects computer vision algorithms, quantifying its impact is difficult. In this context, we present a rain rendering pipeline that enables the systematic evaluation of common computer vision algorithms to controlled amounts of rain. We present three different ways to add synthetic rain to existing images datasets: completely physic-based; completely data-driven; and a combination of both. The physic-based rain augmentation combines a physical particle simulator and accurate rain photometric modeling. We validate our rendering methods with a user study, demonstrating our rain is judged as much as 73% more realistic than the state-of-the-art. Using our generated rain-augmented KITTI, Cityscapes, and nuScenes datasets, we conduct a thorough evaluation of object detection, semantic segmentation, and depth estimation algorithms and show that their performance decreases in degraded weather, on the order of 15% for object detection, 60% for semantic segmentation, and 6-fold increase in depth estimation error. Finetuning on our augmented synthetic data results in improvements of 21% on object detection, 37% on semantic segmentation, and 8% on depth estimation.

Keywords Adverse weather · vision and rain · physics-based rendering · image to image translation · GAN

1 Introduction

A common assumption in computer vision is that light travels unaltered from the scene to the camera. In clear weather, this assumption is reasonable: the atmosphere behaves like a transparent medium and transmits light with very little

Fig. 1 Vision tasks in clear and rain-augmented images. Our synthetic rain rendering framework allows for the evaluation of computer vision algorithms in challenging bad weather scenarios. We render physically-based, realistic rain on images from the KITTI [23] (rows 1-2) and Cityscapes [13] (rows 3-4) datasets with object detection from mx-RCNN [73] (row 2), semantic segmentation from ESPNet [65] (row 4). We also present a combined data-driven and physic-based rain rendering approach which we apply to the nuScenes [9] (rows 5-6) dataset with depth estimation from Monodepth2 [25] (row 6). All algorithms are quite significantly affected by rainy conditions.
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 which must be robust to these conditions.

While the influence of rain on image appearance is well-known and understood [19], its impact on the performance of computer vision tasks is not. Indeed, how can one evaluate what the impact of, say, a rainfall rate of 100 mm/hour (a typical autumn shower) is on the performance of an object detector, when our existing databases all contain images overwhelmingly captured under clear weather conditions? To measure 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 is prohibitive. Indeed, datasets with bad weather information are few and sparse, and typically include only high-level tags (rain or not) without mentioning how much rain is falling. While they can be used to improve vision algorithms under adverse conditions by including rainy images in the training set, they cannot help us in systematically evaluating performance degradation under increasing amounts of rain.

Alternatively, one can attempt to remove its effects from images—i.e., create a “clear weather” version of the image—prior to applying subsequent algorithms. For example, rain can be detected and attenuated from images [21,3,74,79,50]. We experiment with this approach in sec. 7.4. An alternative approach is to employ programmable lighting to reduce the rain visibility, by shining light between raindrops [11]. Unfortunately, these solutions either add significant processing times to already constrained time budgets, or require custom hardware. Instead, if we could systematically study the effect of weather on images, we could better understand the robustness of existing algorithms, and, potentially, increase their robustness afterwards.

In this paper, we propose methods to realistically augment existing image databases with rainy conditions. We rely on well-understood physical models as well as on recent image-to-image translations to generate visually convincing results. First, we experiment with our novel physics-based approach, which 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 (200+ 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 [23] and Cityscapes [13]) with rain, and evaluate the robustness of popular object detection and segmentation algorithms on these augmented databases. Second, we experiment with a combination of physics- and learning-based approaches, where a popular unpaired image-to-image translation method [83] is used to convey a sense of “wetness” to the scene, and physics-based rain is subsequently composited on the resulting image. Here, we augment the nuScenes dataset [9], and use it to evaluate the robustness of object detection and depth estimation algorithms. Finally, we also use the latter to refine algorithms using curriculum learning [5], and demonstrate improved robustness on real rainy images.

In short, we make the following contributions. First, we present two different realistic rain rendering approaches: the first is a purely physic-based method and the second is a combination of a GAN-based approach and this physic-based framework. Second, we augment KITTI [23], Cityscapes [13], and nuScenes [9] datasets with rain. Third, we present a methodology for systematically evaluating the performance of 13 popular algorithms—for object detection, semantic segmentation and depth estimation—on rainy images. Our findings indicate that rain affects all algorithms: performance drops of 15% mAP for object detection, 60% AP for semantic segmentation, and a 6-fold increase in depth estimation error. Finally, our augmented database can also be used to finetune these same algorithms in order to improve their performance in real-world rainy conditions.

This paper significantly extends an earlier version of this work published in [27], by combining physics-based rendering with learning-based image-to-image translation methods, conducting a novel, more in-depth user study, evaluating depth estimation algorithms, comparing to a deraining approach, and by providing a more extensive evaluation of the performance improvement on real images. Our framework is readily usable to augment existing image with realistic rainy conditions. Code and data are available at the following URL: https://team.inria.fr/rits/computer-vision/weather-augment/.

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 [21] and synthesizing [20] realistic rain. In particular, they propose an image-based rain streak database [20] modeling the drop oscillations, which we exploit in our physics-based rendering framework. Other streak appearance models were proposed in [71,3] using a frequency model. Realistic rendering was also obtained with ray-tracing [64] or artistic-based techniques [69,14] but on synthetic data as they require complete 3D knowledge of the scene including accurate light estimation. Numerous works also studied generation of raindrops on-screen, with 3D modeling and ray-casting [62,63,28,29] or normal maps [57] some also accounting for focus blur.
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 [18]. For this, several techniques have been proposed, ranging from frequency space analysis [3] to deep networks [74]. Sparse coding and layers priors were also important axes of research [46,51,12] due to their facility to encode streak patches. Recently, dual residual networks have been employed [50]. Alternatively, camera parameters [19] or programmable light sources [11] can also be adjusted to limit the impact of rain on the image formation process. Additional proposals were made for the specific task of raindrops removal on windows [17] or windshields [28,57,29].

Unpaired image translation An interesting solution to weather augmentation is the use of data-driven unpaired image translation frameworks. Zhang et al. [79] proposed to use conditional GANs for rain removal. By proposing to add a cyclic loss to the learning process, CycleGAN [83] became a significant paper for unpaired image translation; they produced interesting results in weather and season translation. DualGAN [75] uses similar ideas with differences in the network models. The UNIT [47], MUNIT [33], and FUNIT [48] frameworks all, in one way or another, propose to perform image translation with the common idea that data from different sets have a shared latent space. They showed interesting results on adding and removing weather effect to images. Since the information in clear and rainy images is symmetrical, many unsupervised image translation approaches could produce decent visual results. In [56], Pizzati et al. learn to disentangle the scene from lens occlusions such as raindrops, which improves both realism and physical accuracy of the translations. Another strategy for better qualitative translations is to rely on semantic consistency [44,68].

Weather databases In computer vision, few images databases have precise labeled weather information. Of note for mobile robotics, the BDD100K [76], the Oxford dataset [52], and Wilddash [77] provide data recorded in various weather conditions, including rain. Other stationary camera datasets such as AMOS [35], the transient attributes dataset [40], the Webcam Clip Art dataset [41], or the WILD dataset [55] are sparsely labeled with weather information. The relatively new nuScenes dataset [9] have multiple labeled scenes containing rainy images, but variation in rain intensities are not indicated. Gruber et al. [26] recently released a dataset with dense depth labels under a variety of real weather conditions produced by a controlled weather chamber, which inherently limits the variety of scenes (limited to four common scenarios) in the dataset. Note that [6] also announced—but at the time of writing, not yet fully available—a promising dataset including heavy snow and rain events. Still, datasets with rainy data are too small to train algorithms and there exists no dataset with systematically recorded rainfall rates and object/scene labels. The closest systematic works in spirit [38,39] evaluated the effect of simulating weather on vision, but did so in purely virtual environments (GTA and CARLA, respectively) whereas we augment real images. Of particular relevance to our work, Sakaridis et al. [66] propose a framework for rendering fog into images from the Cityscapes [13] dataset. Their approach assumes a homogeneous 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 on top of existing images, but we focus on rain rather than fog.

3 Rain Augmentation

Broadly speaking, synthesizing rain on images can be achieved using two seemingly antagonistic methods: 1) physics-based rendering (PBR) methods [28,20], which explicitly model the dynamics and the radiometry of rain drops in images; or 2) learning-based image-to-image translation approaches [83,33], which train deep neural networks to “translate” an image into its rainy version. While completely different, we argue both these methods offer complementary advantages. On one hand, physics-based approaches are accurate, controllable, can simulate a wide variety of imaging conditions, and do not require any training data. On the other hand, learning-based approaches can realistically simulate important visual cues such as wetness, cloud cover, and overall gloominess typically associated with rainy images.

In this paper, we propose to first explore the use of both techniques independently, then to combine them into a hybrid approach. This section thus first describes our PBR approach (sec. 3.1), followed by the image-to-image translation with a GAN (sec. 3.2), and concludes with the hybrid combination of the two GAN+PBR (sec. 3.3).
3.1 Physics-Based Rendering (PBR)

Taking inspiration from the vast literature on rain physics, [53,7,21,54] we simulate the rain appearance in an arbitrary image with the approach summarized in fig. 2. Based on the estimated scene depth, a fog-like attenuation layer is first generated. Individual rain streaks are subsequently generated, and composited with the fog-like layer. The final result is blended in the original image to create a realistic, physics-based and controllable rainfall rate.

3.1.1 Fog-like rain

Following the definition of [21], fog-like rain is the set of drops that are too far away and that project on an area smaller than 1 pixel. In this case, a pixel may even be imaging a large number of drops, which causes optical attenuation [21]. In practice, most drops in a rainfall are actually imaged as fog-like rain\(^1\), though their visual effect is less dominant.

We render the volumetric attenuation using the model described in [71] where the per-pixel attenuation \( I_{\text{att}} \) is expressed as the sum of the extinction \( L_{\text{ext}} \) caused by the volume of rain and the airlight scattering \( A_{\text{air}} \), that results of the environmental lighting. Using equations from [71] to model the attenuation image at pixel \( x \) we get

\[
I_{\text{att}}(x) = I_{\text{ext}}(x) + A_{\text{air}}(x),
\]

where

\[
L_{\text{ext}}(x) = e^{-0.312R^0.67d(x)},
A_{\text{air}}(x) = \beta_{\text{HG}}(\theta)\hat{E}_{\text{sun}}(1 - L_{\text{ext}}(x)).
\]

Here, \( R \) denotes the rainfall rate \( R \) (in mm/hr), \( d(x) \) the pixel depth, \( \beta_{\text{HG}} \) the standard Heyney-Greenstein coefficient, and \( \hat{E}_{\text{sun}} \) the average sun irradiance which we estimate from the image-radiance relation \[32\].

\(^1\) Assuming a stationary camera with KITTI calibration [22], we computed that only 1.24% of the drops project on 1+ pixel in 50 mm/hr rain, and 0.7% at 3 mm/hr. This follows logic: the heavier the rain, the higher the probability of having large drops.

3.1.2 Simulating the physics of raindrops

We use the particles simulator of [11] to compute the position and dynamics of all raindrops greater than 1 mm for a given fall rate\(^2\). The simulator outputs the position and dynamics (start and end points of streaks) of all visible rain drops in both world and image space, and accounts for intrinsic and extrinsic camera calibration.

3.1.3 Rendering the appearance of a rain streak

While ray casting allows exact modeling of drops photometry, this comes at very high processing cost and is virtually only possible in synthetic scenes where the geometry and surface materials are perfectly known [64,28]. 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 [21], 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 [3].

To render a raindrop, we first select a rain streak \( S \in S \) from the streak database \( S \) of [20], which contains 20 different streaks (each with 10 different oscillations) stored in an image format. We select the streak that best matches the final drop dimensions (computed from the output of the physical simulator), and randomly select an oscillation.

The selected rain streak \( S \) is subsequently warped to match the drop dynamics from the physical simulator:

\[
S' = H(S),
\]

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

3.1.4 Computing the photometry of a rain streak

Computing the photometry of a rain streak from a single image is impractical because drops have a much larger field of view than common cameras (165° vs approx. 70–100°). To render a drop accurately, we must therefore estimate the environment map (spherical lighting representation) around that drop. Sophisticated methods could be used [31,30,80] but we employ [10] which approximates the environment map through a series of simple operations on the image.

From each camera relative 3D drop position, we compute the intersection \( F \) of the drop field of view with the environment map \( E \), assuming a 10m constant scene distance. The process is depicted in fig. 3, and geometrical details are

\(^2\) The distribution and dynamics of drops vary on earth due to gravity and atmospheric conditions. We selected here the broadly used physical models recorded in Ottawa, Canada [53, 1].
where we multiply the streak appearance with a per-channel weight:

### 3.1.6 Compositing rainfall on the image

We then blend the rendered drop with the attenuated background image \( I_{\text{att}} \) using the photometric blending model from [21]. Because the streak database and the image \( I \) are likely to be imaged with different exposures, we need to correct the exposure to match the imaging system used in \( I \). Suppose a pixel \( x \) of the image \( I \) and \( x' \) the overlapping coordinates in streak \( S' \), the composite is obtained with

\[
I_{\text{rain}}(x) = \frac{T - S'_0(x')\tau_1}{T} I_{\text{att}}(x) + S'(x')\frac{\tau_1}{\tau_0},
\]

where \( S'_0 \) 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 \( \tau_1 \) the same measure according to our physical simulator. We refer to appendix B for details.

### 3.1.5 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 [58], convolving the streak image \( S' \) with the circle of confusion\(^4\) \( C \), that is: \( S' = S' \ast C \).

Provided in appendix A. 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\(^3\).

Since a drop refracts 94% of its field of view radiance and reflects 6% of the entire environment map radiance [21], we multiply the streak appearance with a per-channel weight:

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

where \( F \) is the mean of the intersection region \( F \), and \( E \) is the mean of the environment map \( E \).

### 3.1.4 Compositing rain 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 \( I_{\text{att}} \) is rendered (eq. 1), 2) the drops outputted by the physical simulator are rendered individually on the image (eq. 5), 3) the global luminosity average of the rainy image denoted \( I_{\text{rain}} \) 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 \( I = I_{\text{rain}} \), where the overbar denotes the intensity average.

### Photometric validation

A limitation of our physical pipeline is the lighting estimation which impacts the photometry of the rain. To measure its effect, in fig. 4 we compare the same rain rendered with our estimated environment map or ground truth illumination obtained from high dynamic range panoramas [30]. Overall, our estimation differs from ground truth when the scene is not radially symmetric but we observe that it produces visually similar rain in images.

### 3.2 Image-to-image translation (GAN)

While our physic-based rendering generates realistic rain streaks and fog-like rain effect, it ignores major rainy characteristics such as wetness, reflections, clouds and thus may fail at conveying the overall look of a rainy scene. Conversely, generative adversarial networks (GAN) excel at learning such visual characteristics as they constitute strong signals for the discriminator in the learning process.

\( ^3 \) 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:

\[
C = \frac{(d-f_p)f_N}{\sqrt{f_p}}\]

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 [20] does not provide an alpha channel, the latter is easily computed since drops were rendered on black background in a white ambient lighting.
We further learn the clear $\mapsto$ rain mapping with Cycle-GAN [83] from a set of unpaired clear/rain images. We train our model with the $256 \times 256$ architecture from [83] on images of input size $448 \times 256$. The generator is similar to [37] with 2 downsampling blocks followed by 9 ResNet blocks and 2 upsampling blocks. The discriminator is a simple 3 hidden layers ConvNet similar to the one used in PatchGAN [34]. The model is optimized for 40 epochs with Adam, using batch size of 1, a learning rate of 0.0002, and $\beta = \{0.5, 0.999\}$.

3.3 Combining GAN and PBR

We combine both the PBR- and the GAN-based rain generation methods together, simply by first translating the image to its rainy version with the GAN, then compositing the rain layer onto the resulting image using PBR (see fig. 5). The sun irradiance estimation $\bar{E}_{\text{sun}}$ (sec. 3.1.1) of the “translated” image is typically darker, which, in turn, makes the fog-like rain more realistic. The estimated environment map is also darker and, consequently, so is its mean value $\bar{E}$. The appearance of rain streaks will thus remain coherent with their environment.

Since rain streaks smaller than 1px in diameter are ignored before the rendering and instead generated as fog-like rain (sec. 3.1.1), we need to apply the PBR renderer at full resolution by upsampling the output of the GAN to the image original size. Once the rain rendering is complete, we downsample the augmented image at $448 \times 256$. We further refer to this hybrid rendering as GAN+PBR.

4 Validating rain appearance

We now validate the appearance of our synthetic rainy images when using either of our rain augmentation pipelines. We observe visual results and quantify their perceptual realism by comparing them to existing rain augmentation approaches.

4.1 Qualitative evaluation

Fig. 6 presents real photographs captured under heavy rain, qualitative results of our rain renderings on images from nuScenes [9] and representative results from 3 recent synthetic rain augmented approaches [78, 74, 79]. From the real rain photographs, our rain rendering shows volume attenuation and sparse visible streaks which correctly vary with the scene background. As opposed to the other rain rendering methods, our pipeline simulates physical rainfall (here, 100mm/hr and 200mm/hr) and valid particles photometry.
rain photographs, it is noticeable that rainy scenes have complex visual appearance, and that the streaks visibility is greatly affected by the imaging device and background.

Our PBR approach is able to reproduce the complex pattern of streaks with orientation consistent with camera motion and photometry consistent with background and depth. As in the real photographs, the streaks are sparse and only visible on darker backgrounds. The veiling effect caused by the rain volume (i.e., fog-like rain) is visible where the scene depth is larger (i.e., image center, sky) and nearby streaks are accurately defocused. Still, the absence of visible wetness arguably affects the rainy feeling.

Conversely, the GAN believably renders the wetness appearance of rainy scenes. While some reflections are geometrically incorrect (e.g., a pole is reflected in the middle of the street in the left column of fig. 6 yet no pole is present), the overall appearance is visually pleasant and the global illumination matches that of real photographs. A noticeable artifact caused by GAN is the blurry appearance of images, whereas real rain images are only blurred in the distance. This is explained by the inability of the GAN to disentangle the scene from the lens drops present in the “rainy” training images. This leads to blurring the whole image being an easy learning optimum, as highlighted in [56]. Another limitation we already mentioned is that GAN does not allow to control the amount of rain in the image.

This limitation is circumvented by our GAN+PBR approach which renders controllable rain streaks while preserving the global wetness appearance learned with image translation. Despite the naive GAN and PBR compositing strategy, the drops naturally blend in the scene.

4.2 User study

To evaluate the perceptual quality of our rain renderings, we conducted two user studies. In the first, users were shown one image at a time, and asked to rate if rain looks realistic on a 5-point Likert scale. A total of 42 images were shown, that is 6 for each of the following: real rainy photographs, ours (PBR, GAN, GAN+PBR), and previous approaches [74, 78, 79]. Answers were obtained for a total of 67 participants, aged from 22 to 75 (avg 37.0, std 14.2), with 32.8% females.

From the Mean Opinion Score (MOS) in fig. 7, all our rain augmentation approaches are judged to be more realistic than any of the previous approaches. Specifically, when converting ratings to the [0, 1] interval, the mean rain realism is 0.77 for real photos, 0.44 for PBR, 0.68 for GAN, 0.52 for GAN+PBR, and 0.30/0.23/0.08 for [78]/[79]/[74] respectively. Despite physical and geometrical inconsistencies, the users consistently judged GAN images to be more realistic. This is in favor of using image-to-image translation rather than physics-based rendering for realism purposes. However, for benchmarking or physical accuracy purposes, GAN+PBR allows us to have an arbitrary control on the rain amount at the cost of slightly lower realism.

In the second study, we asked respondents who participated in the first study to determine, for each of the same images as before, which visual characteristics influenced their decisions. 52 of the original participants responded, aged from 22 to 72 (avg 36.6, std 13.9) including 28.9% females. Results are reported in fig. 8. We note that while falling rain and wetness are the main characteristics in real rain, GAN—judged the most realistic approach—fails to convey the falling rain appearance but excels at rendering wetness. The opposite is observed with PBR, though few users indicated wetness (despite its absence). The GAN+PBR offers a trade-off balancing all characteristics.

5 Evaluating the impact of real rain

We now aim at quantifying the impact of rain on three computer vision tasks: object detection, semantic segmentation, and depth estimation. These tasks are critical for any outdoor vision systems such as mobile robotics, autonomous driving, or surveillance. Before we experiment on rain-augmented images with our PBR, GAN, and GAN+PBR approaches, we first experiment on real rainy photographs.

For this, we use the nuScenes dataset [9]. Benefiting from coarse frame-wise weather annotations in the dataset, we split
nuScenes images\textsuperscript{6} in two subsets: “nuScenes-clear” (images without rain) and “nuScenes-rain” (rainy images). Due to the noisy weather labels, we cross-validated each frame with a historic weather database\textsuperscript{7} using GPS location and time, and only kept frames where the nuScenes label agreed with the weather database. This resulted in sets of 24134 images for nuScenes-clear, and 6028 for nuScenes-rain. In this dataset, rain images are dark, gloomy, the sky is heavily covered, and no falling rain is visible but there are unfocused raindrops occlusions on the lens.

We experiment on these sets of real images with algorithms from each task: YOLOv2 \cite{redmon2018yolov2} for object detection, PSPNet \cite{zhao2017pyramid} for semantic segmentation, and Monodepth2 \cite{godard2019digging} for depth estimation. The nuScenes-clear set is split into train/test subsets of 19685/4449 images from 491/110 scenes. The nuScenes-rain is also split into train/test subsets of 5419/609 images from 134/15 scenes. Here, 1000 images from the nuScenes-clear(test) and all images from the nuScenes-rain(test) subset are used for evaluation (train will be used for the GAN in sec. 6.1). These training and testing sets and subsets are all displayed in table 3 in appendix C.

Note that a large number of images were needed to train the CycleGAN for image translation and, to avoid any overlap in image sequences, it unfortunately left a relatively small subset of nuScenes for the evaluation on real rainy images.

\textsuperscript{6} Note that only front camera and annotated key frames are used for consistency and ground truth accuracy.

\textsuperscript{7} Weather database at \url{https://openweathermap.org}.

For object detection and depth estimation, we first pre-train each algorithm on ImageNet (Darknet53, 448 × 448) and KITTI (monocular, 1024 × 320) respectively, and further finetune them on the nuScenes-clear(train) subset to limit the domain gap between datasets. We then evaluate their performance on the aforementioned test subsets. For segmentation, since semantic labels are not provided on nuScenes, we carefully annotated 25 images from both nuScenes-clear(test) and nuScenes-rain(test). Since we do not have enough labeled data for finetuning, we use our model pretrained on Cityscapes \cite{cordts2016cityscapes}, with the caveat that there may be a significant domain gap between training and evaluation.

Table 1 reports the results of this experiment. The performance on real rainy images compared to clear images for all three tasks are a mAP of 16.30% instead of 32.53% for object detection, an AP of 18.7% instead of 40.8% for semantic segmentation and a square relative error of 3.53% instead of 2.96% for depth estimation. Corresponding qualitative results are displayed in fig. 9. As expected, real rain deteriorates the performance of all algorithms on all tasks. However, we cannot evaluate how rain \textit{intensity} affects these algorithms since it would require the accurate measurement of the rainfall rate at the time of capture.

\section*{6 Evaluating the impact of synthetic rain}

To study how vision algorithms perform under increasing amounts of rain, we leverage our rain synthesis pipeline and
augment popular clear-weather datasets. Specifically, our PBR and GAN+PBR frameworks allow us to measure the performance of these algorithms in controlled rain settings.

6.1 Rain generation setup

We augment all three of the KITTI, Cityscapes, and nuScenes-clear datasets with PBR. We generate rainfall rates ranging from light to heavy storm \( R = \{0, 5, 25, 50, 100, 200\} \) mm/hr. Only the nuScenes-clear dataset is augmented with GAN and GAN+PBR, since neither KITTI nor Cityscapes contain rainy images to train the GAN. Our PBR and GAN+PBR rain augmentation require some preparation as they rely on calibration, depth and camera motion. Our GAN and GAN+PBR rain augmentation require the training of a CycleGAN. These preparations are described below.

**Calibration.** For the realistic physical simulator (sec. 3.1.2) and the rain streaks photometric simulation (sec. 3.1.4), intrinsic and extrinsic calibration are used to replicate the imaging sensor. We used frame-wise or sequence-wise calibration for KITTI and nuScenes. In addition, we use 6mm focal and 2ms exposure for KITTI [23, 22] and assumed 5ms exposure for nuScenes. As Cityscapes does not provide calibration, we use intrinsic from the camera manufacturer with 5ms exposure and extrinsic is assumed similar to KITTI.

**Depth.** The scene geometry (pixel depth) is also required to model accurately the light-particle interaction and the fog optical extinction. We estimate KITTI depth maps from RGB+Lidar with [36], and Cityscapes/nuScenes from monocular RGB with Monodepth [24, 25]. While absolute depth is not required, we aim to avoid the critical artifacts along edges, and thus further align RGB with depth using guided filter [4].

**Camera motion.** We mimic the camera ego motion in the physical simulator to ensure realistic rain streak orientation on still images and preserve temporal consistency in sequences. Ego speed is extracted from GPS data when provided (KITTI and nuScenes), 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.

### Table 1 Vision tasks on real clear/rain images from nuScenes [9].

| Tasks                     | Metric       | Clear (%) | Rain (%) |
|---------------------------|--------------|-----------|----------|
| Object detection [59]     | mAP (%) ↑    | 24.70     | 11.09    |
| Semantic segmentation [82]| AP (%) ↑     | 40.8      | 18.7     |
| Depth estimation [23]     | Sq. err. (%) ↓ | 2.96     | 3.53     |

Figure 10: Performance using our PBR rain augmentation. (a) Object detection performance on our weather augmented KITTI dataset, (b) pixel-semantic segmentation performance on our weather augmented Cityscapes dataset, and (c) depth estimation performance on our weather augmented nuScenes dataset, all of them as a function of rainfall rate. The object detection plot shows the Coco mAP@[.1:.1:.9] (%) across cars and pedestrians, the semantic segmentation plot shows the AP (%), and the depth estimation shows the squared relative error (%). As opposed to object detection which exhibits some robustness, the segmentation and depth estimation tasks are strongly affected by the rain.

**CycleGAN.** A CycleGAN is trained for image-to-image rain translation on the train subsets of nuScenes-clear and nuScenes-rain (sec. 5). In order to make sure that no image is used to both train and evaluate the GAN simultaneously, we use the 4449 images from the nuScenes-clear(test) subset, resize them to \(448 \times 256\), and perform image-to-image translation to generate GAN-augmented rain images. We dub this new set of images “nuScenes-augment” for clarity, and will also use this for the GAN+PBR rain augmentation.

6.2 Evaluating PBR rain augmentation

We compare the performance on PBR-augmented images for 6 object detection algorithms on KITTI [23] (7481 images), 6 segmentation algorithms on Cityscapes [13] (2995 images) and 2 depth estimation algorithm on nuScenes-augment (4449 images). For all of the algorithms, the clear version always serves as a baseline to which we compare the performance on synthetic rain translations.

**Object detection.** We evaluate the 6 PBR augmented weather on KITTI for 6 car/pedestrian pre-trained detection algorithms (with IoU ≥ .7): DSOD [67], Faster R-CNN [60],
R-FCN [15], SSD [49], MX-RCNN [73], and YOLOv2 [59]. Quantitative results for the Coco mAP@[.1:.1:.9] metric across classes are shown in fig. 10a. 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 on PBR images are shown in fig. 11 for 4 out of 6 algorithms to preserve space. All algorithms are strongly affected by the rain; it has a chaotic effect on object detection results because there can be large variance of occlusion level for objects populating the image. Also, as in real-life, far away objects (which are generally small objects) are more likely to disappear behind fog-like rain.

Semantic segmentation. For semantic segmentation, the PBR augmented Cityscapes is evaluated for: AdaptSegNet [70], ERFNet [61], ESPNet [65], ICNet [81], PSPNet [82] and PSPNet(50) [82]. Quantitative results are reported in fig. 10b. 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 differently under rain. ESPNet for example, ranks among the top 3 in clear weather but drops relatively by a staggering 85% and ranks last in stormy conditions (200mm/hr). Corresponding qualitative results are shown in fig. 12 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.

Depth estimation. We evaluate the performance of the recent Monodepth2 [25] and BTS [43] on the nuScenes-augment subset augmented with our PBR method. We report the standard squared relative error in fig. 10c and note that the error seems to increase linearly with rain. In the extreme 200mm/hr rain conditions, we measure an error of 3x that of clear images. Qualitative results are shown in fig. 13. It can be observed that performance drops when raindrops block the view or fog-like limits the visibility in the image.

6.3 Evaluating GAN and GAN+PBR rain augmentations

Next, we ascertain the effect of the rain augmentation with our GAN and GAN+PBR augmentation strategies. As mentioned above, semantic segmentation algorithms are not eval-
Rain rendering for evaluating and improving robustness to bad weather

**Fig. 12** Qualitative evaluation of semantic segmentation on our PBR rain augmentation of Cityscapes. From left to right, the original image (clear) and three PBR augmentations with varying rainfall rates.

**Fig. 13** Depth estimation on our PBR rain augmentation of nuScenes. From left to right, the original image (clear) and three PBR augmentations with varying rainfall rates.
7.1 Training methodology

While the ultimate goal is to improve robustness to rain, we aim at training a single model which performs well across a wide variety of rainfall rates (including clear weather). Having a single model is beneficial over employing, e.g., intensity-specific encoders [57], since it removes the need for determining rain intensity from the input image. Because rain significantly alters the appearance of the scene, we found that training from scratch with heavy rain or random rainfall rates fails to converge. Instead, we refine our untuned models using curriculum learning [5] on rain intensity in ascending order (25, then 50, and finally 100 mm/hr rain). The final model is referred as finetuned and is evaluated against various weather conditions. Note that, for hybrid augmentation finetuning, the curriculum starts with the refinement on GAN images first and then go through the ascending rain intensities. The same images are used for all steps of the curriculum.

In order to avoid training and testing on the same set of images, in this section, we further divide the nuScenes-augment into train/test subsets of 1000 images each (ensuring they are taken from different scenes). Each algorithm is thus evaluated due to the lack of semantic labels for training. The evaluation is performed on the nuScenes-augment subset.

Object detection. YOLOv2 [59] is evaluated, and the resulting mAP as a function of rainfall rates are reported in fig. 14a. Qualitative results are shown in fig. 15. We note that GAN augmented images have similar performance than PBR 100 mm/hr images and that performance deterioration is stronger and steeper with GAN+PBR images.

Observe that the particles physical simulation is the same in both PBR and GAN+PBR. Still, it is interesting to notice that the decrease is different with GAN+PBR compared to PBR (i.e., curves are shifted but also exhibit different slopes). This may be the result of the two cumulative domain shifts (i.e., wetness + streaks) leading to a non-linear effect.

Depth estimation. We evaluate the performance of the recent Monodepth2 [25] on the nuScenes-augment subset augmented with our GAN and GAN+PBR methods. As reported in fig. 14b, for the same rain intensity between PBR and GAN+PBR images, the error is worse by a factor of 80–100%. The same behavior was observed on other standard depth estimation metrics (absolute square error, RMSE, log RMSE), not reported here. Interestingly, the GAN augmentation affects only slightly the performance on depth estimation which might be because GAN translation keeps occlusion of the image to a minimum. Qualitative results are shown in fig. 16 for GAN and GAN+PBR.

7.2 Improvement on synthetic rain

The synthetic evaluation is conducted on the set of 1000 images from nuScenes-augment(test), with rain up to 200 mm/hr. Note again, for our hybrid GAN+PBR, 0 mm/hr of rain correspond to the GAN-only augmented results.

Fig. 17 shows the performance of our untuned and fine-tuned model for the three vision tasks on different augmented dataset. Fig. 17a and fig. 17b are for object detection (YOLOv2 [59]) and depth estimation (Monodepth2 [25]) on nuScenes-clear augmented data while fig. 17c is for the semantic segmentation (PSPNet [82]) on Cityscapes. We observe a significant improvement in both tasks and additional increase in robustness even in clear weather when refined using our augmented rain. Of interest, we also improve at the unseen 200 mm/hr rain though the network was only trained with rain up to 100 mm/hr. The intuition here is that when facing adverse weather, the network learns to focus on strongest relevant features for all tasks and thus gain robustness.
Rain rendering for evaluating and improving robustness to bad weather

Fig. 15 Object detection on our GAN+PBR augmented nuScenes. From left to right, the original image (clear), the GAN augmented image and three GAN+PBR images.

Fig. 16 Depth estimation on our GAN+PBR augmented nuScenes. From left to right, the original image (clear), the GAN augmented image and three GAN+PBR images.
(b) Depth estimation [25]

Fig. 17 Original (untuned) or finetuned performance on rain-augmented versions of nuScenes (a)-(b) and Cityscapes (c). Not only the finetuned models significantly outperform untuned models, but they exhibit a lower decrease with rain intensity, demonstrating increased robustness to both rain and clear weather.

PBR. For YOLOv2, the finetuned detection performance stays higher than its clear untuned counterpart in the 0-200mm/hr interval. Explicitly, it goes from 34.5% to 31.0% whereas the untuned model starts at 34.6% and finishes at 20.4%. 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. Monodepth2 finetuning helps only for higher rain intensity level (+25mm/hr) and the error differences between 100mm/hr and 200mm/hr stay in the same ballpark (~1.2%). This makes sense considering that since the occlusion created by rain streaks is minimum with a low rain intensity, the untuned model would not be strongly affected.

GAN+PBR. In the case of YOLOv2, we notice a major difference between the hybrid GAN+PBR untuned and finetuned performances. Indeed, the hybrid finetuned performance at 100mm/hr is at 21.7% and only at a measly 7.5% for the untuned model. The same goes for Monodepth2 for hybrid images performance with 5.7% and 8.9% at 100mm/hr for finetuned and untuned respectively. It is interesting to note that, for all tasks, performance evaluated on finetuned hybrid image decrease slower than for untuned models. This demonstrates again that more robust models are learned when finetuning with our rain translations.

7.3 Improvement on real rain

We evaluate the performance on real rain, using our nuScenes-rain(test) subset of images (see sec. 5). Table 2 shows that our finetuning leads to performance increase in real rainy scenes compared to untuned performance in rain. We note for object detection (PBR: +20.7%, GAN: +10.9%, GAN+PBR: +21.0%), for semantic segmentation (PBR: +36.9%), and for depth estimation (PBR: 0.0%, GAN: +3.8%, GAN+PBR: +8.2%) tasks. In clear weather, our finetuned model performs on par with the untuned version, sometimes even better. This boost in performance could be seen as the network learning to rely on more robust features, somehow invariant to rain streaks.

Depth estimation underperformance for PBR finetuning can be explained by the learning loss of Monodepth2 which is, in short, a reprojection error which would not fare well with rain streaks as they do not reproject in consecutive frames. Interestingly, this problem does not seem to affect the GAN or GAN+PBR finetuned model, possibly because the GAN is being trained on split of nuScenes subsequently leading to finetuning images that are more resembling of the test set. These results demonstrate the usefulness of our different rain rendering frameworks for real rain scenarios.

7.4 De-raining comparison

We now compare to the strategy of de-raining images first and then running un-tuned vision algorithms. To this end, we used the state-of-the-art de-raining method DualResNet [50], finetuned using nuScenes-clear augmented with GAN+PBR to accommodate for the domain gap.

During the de-raining fine-tuning process, random batches of \{25, 50, 100, 200\}mm/hr paired with their non-augmented counterpart are generated. Except for a smaller learning rate ($10^{-5}$), we used the DualResNet default hyper-
We see that the performance on both tasks decrease linearly for both (a) object detection with YOLOv2 [59] and (b) depth estimation with Monodepth2 [25]. Performance on both tasks are lower at low rain intensities (and the opposite at high rain intensities) compared to models finetuned with GAN+PBR synthetic images (cf. fig. 17).

With this de-raining finetuned model, we compare the performance of our “untuned” object detection (YOLOv2 [59]) and depth estimation (Monodepth2 [25]) models. Fig. 18 shows the performance of the de-raining strategy compared to our “rain-aware” GAN+PBR finetuned models. Here, we observe that the rain-aware models offer improved performance for object detection over de-raining, while the latter improves depth estimation. This is likely due to the fact that streaks occlude the scene background, while de-raining acts as prior imprinting thus easing depth estimation.

We also applied the same de-raining strategy to the real nuScenes images and report performance in the last row of table 2. Again, for object detection on rainy images our rain-aware models perform better than de-raining. However, for depth estimation, the de-raining strategy is better for both clear and rainy images. This is consistent with the results obtained on synthetic data.

These experiments illustrate that de-raining is also a valid strategy that may even outperform “rain-aware” algorithms. However, this comes at the cost of having to perform two tasks, which may limit practical applications. On the long term, we believe rain-robust algorithms offer an exciting new research paradigm while avoiding the in-filling of occluded areas.

8 Discussion

In this paper, we presented the first intensity-controlled physical framework for augmenting existing image databases with realistic rain. This allows us to systematically study the impact of rain on existing computer vision algorithms on three important tasks: object detection, semantic segmentation, and depth estimation.

Limitations and future work. While we demonstrated highly realistic rain rendering results, our approach still has limitations that set the stage for future work.

For our PBR approach, the approximation of the lighting conditions 3.1.1 yielded reasonable results (fig. 3), but 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. More robust approaches for outdoor lighting estimation could potentially be used [30,80]. Second, we make an 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 [21,19,11,45], 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, rain streaks are added to image irrespective of the scene contents. Here, the depth estimate could be used to mask out streaks that appear behind objects and under the ground plane. Another limitation is the computational cost of PBR. While this has no downside for benchmarking purposes as PBR may be run off-line, simulation requires increasing time with larger rainfall rates. With our current unoptimized implementation, the simulation of 1 / 25 / 50 / 100 mm/hr rainfall rates on Cityscapes requires 0.35 / 5.65 / 16.60 / 20.67 seconds respectively for the rain physics [11] and an additional 6.71 / 34.92 / 62.94 / 104.76 seconds for the rendering (times are per image, on single core, and averaged over 100 frames). This restricts the usage of PBR to off-line processing though significant speed up could be obtained at the cost of additional optimization efforts.

The GAN employed also has limitations. First, while PBR is well-suited for videos since the rain simulator is temporally consistent, this is not the case for CycleGAN which does not guarantee temporal smoothness. Existing approaches such as [2] are alternatives, but GANs are known for their non-realistic physical outcome [72]. Second, CycleGAN imposes a limit on the image resolution. Here, super-resolution networks such as SRRResNet [16] or SRGAN [42] could potentially be used, or large-scale GANs such as BigGAN [8] are also an option. More importantly, GANs tend to have difficulty in generating rain on images of datasets different than which they are trained on since the learning process does not disentangle rain from scene appearance, demonstrating a strong domain dependence.

Finally, while our results demonstrate that fine-tuning on synthetically generated rain does improve performance on real rainy images (cf. sec. 7), the improvements obtained are still quite modest. Further efforts are necessary to develop algorithms that are truly robust to challenging rainy conditions.
A Field of view of a drop in a sphere

We estimate the field of view (FOV) of a drop when projected on a sphere to compute the radiance and chromaticity of each streak, as detailed in sec. 3.3 of the main paper. Despite its motion, we make the assumption of a constant field of view within a given exposure time. This is acceptable due to the short exposure time used here (i.e. 2ms for KITTI, 5ms for Cityscapes). For each drop, the simulator outputs the detailed in sec. 3.3 of the main paper. Despite its motion, we make the donation of the GPU used in this research.

Acknowledgements

This work was partially supported by the Service de coopération et d’action culturelle du Consulat général de France à Québec, as well as the FRQ-NT with the Samuel-de-Champlain grant. 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. We also thank the Nvidia corporation for the donation of the GPU used in this research.

B Compositing a rain streak with different exposure time

In their seminal work, Garg and Nayar [19] demonstrated that the streak appearance is closely related to the amount of time τ a drop stays on a pixel. It is thus important to account for the difference of exposure time in the streak appearance database [20] when adding rain to existing images. Given that the appearance database does not provide enough calibration data to recompute the exact original τ0, we estimate it using observations made in appendix 10.3 of [21]. The latter states that for a constant exposure time τ can be safely approximated with $\sqrt{\pi/50}$ (a the drop diameter, in meters), which we use to compute τ0 according to simulation settings in [20].

Using the notation defined in eq. (6) from the main paper, the radiance of streak $S'$ is corrected with

$$S' \frac{\tau_{\text{rad}}}{\tau_0} \tag{10}$$

where $\tau_{\text{rad}}$ is the time the current drop stays on a pixel, as obtained in a streak-wise fashion by the physical simulator. Noteworthy, [21] also emphasizes that for a given streak the changes of τ across pixels are negligible, so τ can safely be assumed constant.

Finally, after normalization, the alpha of each streak is scaled according to $\tau_{\text{rad}}$ and the targeted exposure time $T$. According to Garg and Nayar equations (cf. eq. (18) from [21]), the composite rainy image is an alpha blending of the background image $I_{\text{bg}}$ and the rain layer $I_r$.

For pixel $x$ corresponding to $x'$ in the streak coordinates, it leads to:

$$I_{\text{raing}}(x) = I_{\text{bg}}(x) + I_r(x') \frac{\tau_{\text{rad}}}{\tau_0} T,$$

$$= \frac{T - S'(x') \tau_{\text{rad}}}{T} I(x) + S'(x') \frac{\tau_{\text{rad}}}{\tau_0} T. \tag{11}$$
Table 3 Data splits for all experiments. The splits are separated per experimental step of this paper.

| C Experiments data splits |
|---------------------------|
| Table 3 contains the minutiae of the data splits of the various experimental steps of this paper. |

| References |
|-------------|
| Atlas, D., Srivastava, R., Sekhon, R.S.: Doppler radar characteristics of precipitation at vertical incidence. Reviews of Geophysics 11(1), 1–35 (1973) 4 |
| Bansal, A., Ma, S., Ramanan, D., Sheikh, Y.: Recycle-gan: Unsupervised video retargeting. In: European Conference on Computer Vision (2018) 15 |
| Barnum, P.C., Narasimhan, S., Kanade, T.: Analysis of rain and snow in frequency space. International Journal of Computer Vision 86(2-3), 256 (2010) 2, 3, 4 |
| Barron, J.T., Poole, B.: The fast bilateral solver. In: European Conference on Computer Vision (2016) 9 |
| Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: International Conference on Machine Learning (2009) 2, 12 |
| Bijelic, M., Mannan, F., Gruber, T., Ritter, W., Dietmayer, K., Heide, F.: Seeing through fog without seeing fog: Deep sensor fusion in the absence of labeled training data. In: IEEE Conference on Computer Vision and Pattern Recognition (2020) 3 |
| van Boxtel, J.H., et al.: Numerical model for the fall speed of rain drops in a rain fall simulator. In: Workshop on wind and water erosion (1997) 4 |
| Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018) 15 |
| Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: IEEE Conference on Computer Vision and Pattern Recognition (2020) 1, 2, 3, 6, 7, 9, 14, 17 |
| Cameron, C.: Hallucinating environment maps from single images. Tech. rep. (2005) 4 |
| de Charette, R., Tamburo, R., Barnum, P.C., Rowe, A., Kanade, T., Narasimhan, S.G.: Fast reactive control for illumination through rain and snow. In: IEEE International Conference on Computational Photography (2012) 2, 3, 4, 15 |
| Chen, Y.L., Hsu, C.T.: A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In: IEEE International Conference on Computer Vision (2013) 3 |
| Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 1, 2, 3, 8, 9, 17 |
| Creus, C., Patow, G.A.: R4: Realistic rain rendering in realtime. Computers & Graphics 37(1-2), 33–40 (2013) 2 |
| Dai, J., Li, Y., He, K., Sun, J.: R-FCN: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems (2016) 10 |
| Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 38(2), 295–307 (2015) 15 |
| Eigen, D., Krishnan, D., Fergus, R.: Restoring an image taken through a window covered with dirt or rain. In: IEEE International Conference on Computer Vision (2013) 3 |
| Garg, K., Nayar, S.K.: Detection and removal of rain from videos. In: IEEE Conference on Computer Vision and Pattern Recognition (2004) 3 |
| Garg, K., Nayar, S.K.: When does a camera see rain? In: IEEE International Conference on Computer Vision (2005) 2, 3, 6, 15, 16 |
| Garg, K., Nayar, S.K.: Photorealistic rendering of rain streaks. ACM Transactions on Graphics (SIGGRAPH) 25(3), 996–1002 (2006) 2, 3, 4, 5, 16 |
| Garg, K., Nayar, S.K.: Vision and rain. International Journal of Computer Vision 75(1), 3–27 (2007) 2, 4, 5, 15, 16 |
| Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal of Robotics Research 32(11), 1231–1237 (2013) 4, 9 |
| Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: IEEE Conference on Computer Vision and Pattern Recognition (2012) 1, 2, 9, 17 |
| Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 9 |
| Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: IEEE International Conference on Computer Vision (2019) 1, 8, 9, 10, 11, 12, 13, 14, 15 |
| Gruber, T., Bijelic, M., Heide, F., Ritter, W., Dietmayer, K.: Pixel-accurate depth evaluation in realistic driving scenarios. In: International Conference on 3D Vision (2019) 3 |
| Halder, S.S., Lalonde, J.F., Charette, R.d.: Physics-based rendering for improving robustness to rain. In: IEEE International Conference on Computer Vision (2019) 2 |
| Halimeh, J.C., Roser, M.: Raindrop detection on car windshields using geometric-photometric environment construction and intensity-based correlation. In: IEEE Intelligent Vehicles Symposium (2019) 2, 3, 4 |
| Hao, Z., You, S., Li, Y., Li, K., Lu, F.: Learning from synthetic photorealistic raindrop for single image raindrop removal. In: IEEE International Conference on Computer Vision Workshops (2019) 2, 3 |
| Hold-Geoffroy, J., Athawale, A., Lalonde, J.F.: Deep sky modeling for single image outdoor lighting estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 4, 5, 15 |
| Hold-Geoffroy, Y., Sunkavalli, K., Hadap, S., Gambaretto, E., Lalonde, J.F.: Deep outdoor illumination estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 4 |
| Horn, B., Klaau, B., Horn, P.: Robot vision. MIT press (1986) 4 |
| Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: European Conference on Computer Vision (2018) 3 |
76. Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving video database with scalable annotation tooling. arXiv preprint arXiv:1805.04687 (2018)

77. Zendel, O., Honauer, K., Murschitz, M., Steininger, D., Fernandez-Dominguez, G.: Wilddash-creating hazard-aware benchmarks. In: European Conference on Computer Vision (2018)

78. Zhang, H., Patel, V.M.: Density-aware single image de-raining using a multi-stream dense network. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)

79. Zhang, H., Sindagi, V., Patel, V.M.: Image de-raining using a conditional generative adversarial network. IEEE Transactions on Circuits and Systems for Video Technology (2019)

80. Zhang, J., Sunkavalli, K., Hold-Geoffroy, Y., Hadap, S., Eisenman, J., Lalonde, J.F.: All-weather deep outdoor lighting estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)

81. Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: ICNet for real-time semantic segmentation on high-resolution images. In: European Conference on Computer Vision (2018)

82. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)

83. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision (2017)