

# How to Train Neural Networks for Flare Removal

Yicheng Wu<sup>1†</sup> Qiurui He<sup>2</sup> Tianfan Xue<sup>2</sup> Rahul Garg<sup>2</sup> Jiawen Chen<sup>3</sup>

Ashok Veeraraghavan<sup>1</sup> Jonathan T. Barron<sup>2</sup>

<sup>1</sup>Rice University <sup>2</sup>Google Research <sup>3</sup>Adobe Inc.

## Abstract

*When a camera is pointed at a strong light source, the resulting photograph may contain lens flare artifacts. Flares appear in a wide variety of patterns (halos, streaks, color bleeding, haze, etc.) and this diversity in appearance makes flare removal challenging. Existing analytical solutions make strong assumptions about the artifact’s geometry or brightness, and therefore only work well on a small subset of flares. Machine learning techniques have shown success in removing other types of artifacts, like reflections, but have not been widely applied to flare removal due to the lack of training data. To solve this problem, we explicitly model the optical causes of flare either empirically or using wave optics, and generate semi-synthetic pairs of flare-corrupted and clean images. This enables us to train neural networks to remove lens flare for the first time. Experiments show our data synthesis approach is critical for accurate flare removal, and that models trained with our technique generalize well to real lens flares across different scenes, lighting conditions, and cameras.*

## 1. Introduction

Photographs of scenes with a strong light source often exhibit lens flare—a salient visual artifact caused by unintended reflections and scattering within the camera. Flare artifacts can be distracting, reduce detail, and occlude image content. Despite significant efforts in optical design to minimize lens flare, even small light sources can still produce substantial artifacts when imaged by consumer cameras.

Flare patterns depend on the optics of the lens, the location of the light source, manufacturing imperfections, and scratches and dust accumulated through everyday use. The diversity in the underlying cause of lens flare leads to the diversity in its presentation. As Fig. 1 shows, typical artifacts include halos, streaks, bright lines, saturated blobs, color bleeding, haze, and many others. This diversity makes the problem of flare removal exceedingly challenging.

<sup>†</sup>This work was done while Yicheng Wu was an intern at Google Research. He is currently a Research Scientist at Snap Research.



Figure 1. Lens flare artifacts frequently occur in photographs with strong light sources. They exhibit a wide range of shapes and colors, which makes them difficult to remove with existing methods.

Most existing methods for lens flare removal [1, 3, 24] do not account for the physics of flare formation, but rather naïvely rely on template matching or intensity thresholding to identify and localize the artifact. As such, they can only detect and potentially remove limited types of flares, such as saturated blobs, and do not work well in more complex real-world scenarios.

Despite the proliferation of deep neural networks, there seems to be no successful attempt at learning-based flare removal. What is it that makes this problem so hard?

The main challenge is the lack of training data. Collecting a large number of perfectly-aligned image pairs with and without lens flare would be tedious at best and impossible at worst: the camera and the scene would need to be static (a particularly difficult requirement given most lens flare occurs outdoors and involves the sun), and one would need some mechanism to “switch” the artifacts on and off without also changing the illumination of the scene. With significant effort this can be accomplished by collecting pairs of images taken on a tripod where the photographer manually places an occluder between the illuminant and the camera in one image. But this approach is too labor-intensive to produce the thousands or millions of image pairs usually required to train a neural network. Furthermore, this approach only works when the flare-causing illuminant lies outside of the camera’s field of view (e.g., real scenes in Fig. 7), which limits its utility.

To overcome this challenge, we propose to generate *semi-synthetic* data grounded on the principles of physics. We make the key observation that lens flare is an additive layer on top of the underlying image, and that it is induced by either scattering or internal reflection. For the scattering case (e.g., scratches, dust, other defects), we construct a wave optics model that we demonstrate closely approximates reality. For the unintended reflections between lens elements, we adopt a rigorous data-driven approach, as an accurate optical model for a commercial camera is often unavailable. With this formulation, we are able to generate a large and diverse dataset of semi-synthetic flare-corrupted images, paired with ground-truth flare-free images.

Another challenge is removing flare while keeping the visible light source intact. This is hard even with our semi-synthetic data, as we cannot separate the light source from the flare-only layer without affecting the flare it induces. Hence, if trained naïvely, the network will try to remove the light source along with the flare, leading to unrealistic outputs. To this end, we propose a loss function that ignores the light source region, and a post-processing step to preserve the light source in the output.

To show the effectiveness of our dataset and procedures, we train two distinct convolutional neural networks originally designed for other tasks. During training, we minimize a loss function on both the predicted flare-free image and the residual (i.e., inferred flare). At test time, the networks require only a single RGB image taken with a standard camera and are able to remove different types of flare across a variety of scenes. Although trained exclusively on semi-synthetic data, both models generalize well to real-world images. To the best of our knowledge, this is the *first* general-purpose method for removing lens flare from a single image.

Our code and datasets are publicly available at <https://yichengwu.github.io/flare-removal/>.

## 2. Related work

Existing solutions for flare removal fall into three categories: (a) optical design intended to mitigate the presence of flare, (b) software-only methods that attempt post-capture enhancement, and (c) hardware–software solutions that capture additional information.

**Hardware solutions** The lenses of high-end cameras often employ sophisticated optical designs and materials to reduce flare. As each glass element is added to a compound lens to improve image quality, there is also an increased probability that light is reflected from its surface to create flare. One widely used technique is to apply anti-reflective (AR) coating to lens elements, which reduces internal reflection by destructive interference. However, the thickness of this coating can only be optimized for particular wave-

lengths and angles of incidence and therefore cannot be perfect. Additionally, adding an AR coating to all optical surfaces is expensive, and may preclude or interfere with other coatings (e.g., anti-scratch and anti-fingerprint).

**Computational methods** Many post-processing techniques have been proposed to remove flare. Deconvolution has been used to remove flare in X-ray imaging [7, 21] or HDR photography [19]. These approaches depend critically on the assumption that the point spread function of the flare does not vary spatially, which is generally not true. Other solutions [1, 3, 24] adopt a two-stage process: detecting lens flare based on their unique shape, location, or intensity (i.e., by identifying a saturated region), and then recovering the scene behind the flare region using inpainting [5]. These solutions only work on limited types of flare (e.g., bright spots), and are vulnerable to the misclassification of all bright regions as flare. Additionally, these techniques classify each pixel as either “flare” or “not flare”, ignoring the fact that most lens flares are better modeled as a semi-transparent layer on top of the underlying scene.

**Hardware–software co-design** Researchers have used computational imaging for flare removal, where the camera hardware and post-processing algorithms are designed in conjunction. Talvala et al. [23] and Raskar et al. [18] attempt to selectively block flare-causing light using structured occlusion masks and recover the flare-free scene using either direct–indirect separation or a light field-based algorithm. Though elegant, they require special hardware and are therefore limited in their practicality.

**Learning-based image decomposition** While no learning-based flare removal techniques exist, a number of recent works apply learning to similar applications such as reflection removal [6, 15, 27], rain removal [17, 25], and denoising [2]. These methods attempt to decompose an image into “clean” and “corrupt” components by training a neural network. Their success relies heavily on high-quality domain-specific training datasets, which this work tries to address for the first time.

## 3. Physics of lens flare

An ideal camera, when in focus, is supposed to refract and converge all rays from a point light source to a single point on the sensor. In practice, real lenses scatter and reflect light along unintended paths, resulting in flare artifacts [12], as shown in Fig. 2(a). The scattered and reflected parts only constitute a small fraction of each incident light ray. So although flare is omnipresent, it is imperceptible in most photographs. But, when a strong light source is many orders of magnitude brighter than the rest of the scene (e.g., the sun), the small fraction of scattered and reflected rays from this bright light will lead to visible artifacts at other

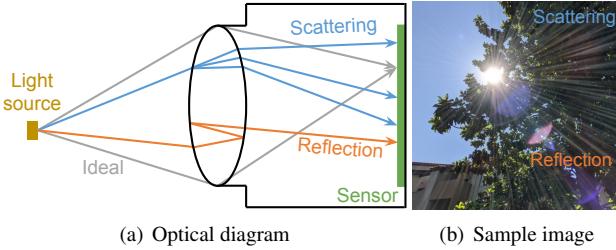


Figure 2. Cameras are intended to focus light from a point source at exactly one point on the sensor (gray rays). However, dust and scratches may cause scattering (blur rays), leading to haze and bright streaks “emitting” radially. Additionally, internal reflections (orange rays) can occur between the optical surfaces of lens elements (only one element is shown), resulting in disk- or arc-shaped bright spots. This reflective flare may also have a color tint, as the anti-reflective coating only blocks certain wavelengths.

pixels on the image. The geometry of the scattering from dust and scratches, and that of the multiple reflections, result in characteristic visual patterns. At a high level, flare can be classified into two principal categories: scattering-induced and reflection-induced.

**Scattering flare** While an ideal lens is 100% refractive, real lenses have many imperfections that cause light to scatter. The scattering (or diffraction) could be due to manufacturing defects (e.g., dents), or normal wear (e.g., dust and scratches). As a consequence, apart from the primary rays that are refracted, a secondary set of rays are scattered or diffracted instead of following their intended paths. While dust adds a rainbow-like effect, scratches introduce streaks that appear to “emit” radially from the light source. Scattering may also reduce contrast in the region around the light source, leading to a hazy appearance.

**Reflective flare** In a practical lens system, each air–glass interface poses an opportunity for a small amount of reflection (typically about 4%). After an even number of reflections, the rays may hit the sensor at an unintended location, forming a reflection pattern. Even if we assume the light is reflected exactly twice, for a lens module containing  $n$  optical elements ( $n \approx 5$  for a modern camera), there are  $2n$  optical surfaces, and thus  $n(2n - 1)$  potential flare-causing combinations. On the image, these reflective flares typically lie on the straight line joining the light source and the principal point. They are sensitive to the light source’s angle of incidence, as demonstrated by Fig. 4(b), but not to rotation about the optical axis. The flare’s shape depends on the geometry, size, and location of the aperture, which may partially block the reflection more than once, resulting in arc-shaped artifacts. As mentioned in Sec. 2, AR coating may be used to reduce reflection—typically reducing reflections at the air–glass interface to below 1%. However, the effectiveness of this coating also depends on wavelength, so lens

flare may exhibit a variety of color tints (often blue, purple, or pink). It is important to note that reflective flare depends on lens design, and therefore cameras with the same design (e.g., all iPhone 12 main camera modules) are expected to produce similar reflective flares when imaging the same scene.

**Challenges in flare removal** The different types of flare are often difficult to visibly distinguish or separate. The appearance of the observed flare may vary significantly based on the properties of the light source (e.g., location, size, intensity, and spectrum), and of the lens (e.g., design and defects). For these reasons, it is impractical to build a completely physics-based algorithm to analytically identify and remove each type of flare, especially when multiple artifacts are present in the same image. Therefore, we propose a data-driven approach.

## 4. Physics-based data generation

Unlike many vision problems in a supervised setting, it is hard to obtain a dataset of flare-corrupted and flare-free image pairs. As a result, we build a physically-realistic, semi-synthetic pipeline.

The additive nature of light implies we can model flare as an additive artifact on top of the “ideal” image—the reduction in the intensity of the “ideal” rays is negligible. We will first explain how the scattering and reflective flares can be modeled and generated, and then use them to synthesize flare-corrupted and flare-free image pairs.

### 4.1. Scattering flare

**Formulation** Under the thin-lens approximation, an optical imaging system can be characterized by the complex-valued *pupil function*  $P(u, v)$ : a 2D field describing, for each point  $(u, v)$  on the aperture plane, the lens’s effect on the amplitude and phase of an incident wave with wavelength  $\lambda$ :

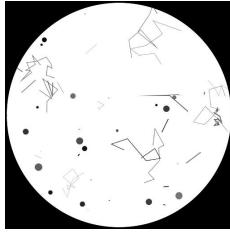
$$P_\lambda(u, v) = A(u, v) \cdot \exp(i\phi_\lambda(u, v)). \quad (1)$$

Here,  $A$  is an *aperture function*, a property of the optics that represents its attenuation of the incident wave’s amplitude.<sup>1</sup> In its simplest form, a camera with an aperture of a finite radius  $r$  has an aperture function of

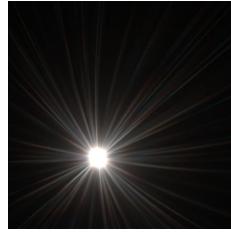
$$A(u, v) = \begin{cases} 1 & \text{if } u^2 + v^2 < r^2 \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

$\phi_\lambda$  in Eq. 1 describes the phase shift, which depends on the wavelength as well as the 3D location of the light source

<sup>1</sup>Strictly speaking, lens optics can also introduce a phase shift, in which case  $A$  becomes a complex-valued function. However, this has shown little difference in our simulation results, so we assume  $A$  is real-valued.



(a) Aperture of a dirty lens



(b) Simulated flare

Figure 3. To simulate a scattering flare component, we generate a number of apertures (a) with random dots and lines that resemble defects. Wave optics then allows us to compute the image (b) of any light source imaged by that synthetic aperture.

$(x, y, z)$ . Omitting the aperture coordinates  $(u, v)$ ,  $\phi_\lambda$  can be written as:

$$\phi_\lambda(x, y, z) = \phi_\lambda^S(x/z, y/z) + \phi_\lambda^{DF}(z) \quad (3)$$

where the linear term  $\phi^S$  is determined by the angle of incidence, and the defocus term  $\phi^{DF}$  depends on the depth  $z$  of the point light source. Once fully specified, the pupil function  $P$  in Eq. 1 can be used to calculate the point spread function (PSF) by a Fourier transform [8]:

$$PSF_\lambda = |\mathcal{F}\{P_\lambda\}|^2 \quad (4)$$

which, by definition, is the image of a point light source at  $(x, y, z)$  formed by a camera with aperture function  $A$ . This is the flare image that we desire.

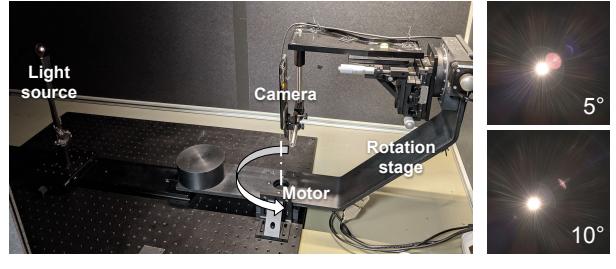
**Sampling PSFs** To mimic dust and scratches on the lens, we add dots and streaks of random size and transparency to the simple aperture function  $A$  in Eq. 2. An example synthetic aperture produced by our procedure is shown in Fig. 3(a), and details are included in the appendix. We generate a total of 125 different apertures.

For a given point light source at location  $(x, y, z)$  with a single wavelength  $\lambda$ , the two phase shift terms in Eq. 3 can be computed deterministically. The PSF for this light source,  $PSF_\lambda$ , can thus be determined by substituting  $A$  and  $\phi_\lambda$  into Eq. 4.

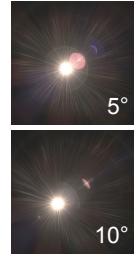
To simulate a light source across the full visible spectrum, we sample  $PSF_\lambda$  for all wavelengths  $\lambda$  from 380nm to 740nm with a spacing of 5nm, resulting in a 73-vector at each pixel of the PSF. The full-spectrum PSF is then left-multiplied by a spectral response function  $SRF$  (a  $3 \times 73$  matrix) to derive the PSF measured by the RGB sensor:

$$\begin{bmatrix} PSF_R(s, t) \\ PSF_G(s, t) \\ PSF_B(s, t) \end{bmatrix} = SRF \begin{bmatrix} PSF_{\lambda=380\text{nm}}(s, t) \\ \vdots \\ PSF_{\lambda=740\text{nm}}(s, t) \end{bmatrix} \quad (5)$$

where  $(s, t)$  are the image coordinates. This produces a flare image for a light source located at  $(x, y, z)$ .



(a) Capture setup



(b) Real flare

Figure 4. We capture images of real lens flare using a strong light source in a dark room. The camera is mounted on a motorized rotation stage that reproduces a wide range of incident angles. Sample images captured at different angles are shown in 4(b).

To construct our dataset of scattering flares, we randomly sample the aperture function  $A$ , the light source's 3D location  $(x, y, z)$ , and the spectral response function  $SRF$  (details can be found in the appendix). We further apply optical distortion (e.g., barrel and pincushion) to augment the  $PSF_{RGB}$  images. One example of final output is shown in Fig. 3(b). We generate a total of 2,000 such images.

## 4.2. Reflective flare

The reflective flare component is difficult to simulate via rendering techniques [10, 14], as they require an accurate characterization of the optics, which is often unavailable. However, lenses of similar design share similar internal reflection paths, so data collected from one instance of the camera often generalizes well to other similar instances.

We capture images of reflective flare in a laboratory setting consisting of a bright light source, a programmable rotation stage, and a fixed-aperture smartphone camera with a  $f = 13\text{mm}$  lens (35mm equivalent), as shown in Fig. 4(a). The setup is insulated from ambient light during capture.

The camera is rotated programmatically so that the light source traces (and extends beyond) the diagonal field of view, from  $-75^\circ$  to  $75^\circ$ . We capture one HDR image every 0.15°, resulting in 1,000 samples. Adjacent captures are then interpolated by 2x using the frame interpolation algorithm of [16], giving us 2,000 images. During training, images are further augmented by random rotation, as reflective flare is rotationally symmetric about the optical axis.

## 4.3. Synthesizing flare-corrupted images

A flare-corrupted image  $I_F$  is generated by adding a flare-only image  $F$  to a flare-free natural image  $I_0$  in linear space (pre-tonemapping raw space where pixel intensities are additive), as illustrated in Fig. 5. We also add random Gaussian noise whose variance is sampled once per image from a scaled chi-square distribution  $\sigma^2 \sim 0.01\chi^2$ , to cover the large range of noise levels we expect to see:

$$I_F = I_0 + F + N(0, \sigma^2). \quad (6)$$

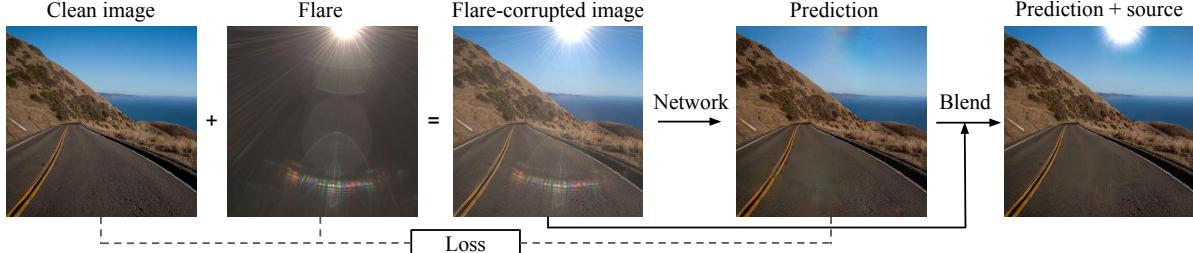


Figure 5. Our approach consists of three steps: 1) We generate training input by randomly compositing a flare-free natural image and a flare image. 2) A convolutional neural network is trained to recover the flare-free scene (in which the light source may also have been removed, which is undesirable). 3) After prediction, we blend the input light source back into the output image.

The flare-free image  $I_0$  is sampled from the 24k Flickr images in [27], and is augmented by random flips and brightness adjustments. Since the Flickr images are already gamma-encoded, we approximately linearize them by applying an inverse gamma curve where  $\gamma$  is sampled uniformly from [1.8, 2.2] to account for the fact that its exact value is unknown.

The flare-only image  $F$  is sampled from both captured and simulated datasets. In Sec. 6.2, we show that both are necessary. Random affine transformations (e.g., scaling, translation, rotation, and shear) and white balance are applied as additional augmentations. Randomization details are in the appendix.

## 5. Reconstruction algorithm

Given a flare-corrupted image  $I_F \in [0, 1]^{512 \times 512 \times 3}$ , our goal is to train a neural network  $f(I_F, \Theta)$  to predict a flare-free image  $I_0$ , where  $\Theta$  is the trainable network weights. Many network architectures may be suitable for our task and we evaluate two popular ones. Refer to the appendix for additional details about the network architectures and our training procedure.

### 5.1. Losses

We want to remove *only* the flare caused by a bright light source. But the flare-only images  $F$  in our datasets contain *both* the flare and the light source, as it is impractical to physically separate the two during capture or simulation. If we trained the network naïvely, it would attempt to hallucinate an image with the light source removed, which is not the intended use of this model and a waste of model capacity. To prevent the model from inpainting the scene behind the light source, we ignore the saturated pixels when computing the loss and assume they are due to the light source. In general, saturated pixels are unrecoverable and contain little information about the scene.

Specifically, we modify the raw network output  $f(I_F, \Theta)$  with a binary saturation mask  $M$  before computing losses. The mask  $M$  corresponds to pixels where the luminance of input  $I_F$  is greater than a threshold (0.99 in

our experiments). We then apply morphological operations so that small saturated regions are excluded from  $M$ , as they are likely part of the scene or the flare. For pixels inside  $M$ , we replace it with the ground truth  $I_0$  so that the loss is zero for such regions<sup>2</sup>:

$$\hat{I}_0 = I_0 \odot M + f(I_F, \Theta) \odot (1 - M), \quad (7)$$

where  $\odot$  denotes element-wise multiplication.

During training, we minimize the sum of two losses: an image loss and a *flare loss*:

$$\mathcal{L} = \mathcal{L}_I + \mathcal{L}_F. \quad (8)$$

The image loss  $\mathcal{L}_I$  encourages the predicted flare-free image  $\hat{I}_0$  to be close to the ground truth  $I_0$  both photometrically and perceptually. The data term is an L1 loss on the RGB values between  $\hat{I}_0$  and  $I_0$ . The perceptual term is computed by feeding  $\hat{I}_0$  and  $I_0$  through a pre-trained VGG-19 network [22]. Like [27], we penalize the absolute difference between  $\Phi_\ell(\hat{I}_0)$  and  $\Phi_\ell(I_0)$  at selected feature layers conv1\_2, conv2\_2, conv3\_2, conv4\_2, and conv5\_2. In summary, the image loss can be expressed as:

$$\mathcal{L}_I = \left\| \hat{I}_0 - I_0 \right\|_1 + \sum_{\ell} \lambda_{\ell} \left\| \Phi_{\ell}(\hat{I}_0) - \Phi_{\ell}(I_0) \right\|_1. \quad (9)$$

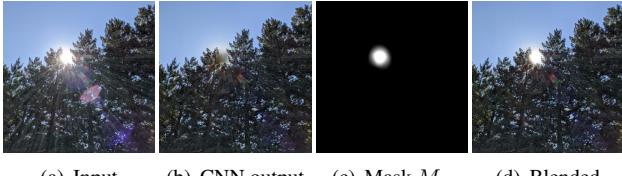
The flare loss  $\mathcal{L}_F$  encourages the predicted flare to be similar to the ground-truth flare  $F$ , and serves to reduce artifacts in the predicted flare-free image (Sec. 6.2). The expression for  $\mathcal{L}_F$  is the same as Eq. 9, but with  $\hat{I}_0$  and  $I_0$  replaced by  $\hat{F}$  and  $F$ , respectively. Here, the predicted flare  $\hat{F}$  is calculated as the difference between the network input and the masked network output:

$$\hat{F} = I_F - f(I_F, \Theta) \odot (1 - M). \quad (10)$$

### 5.2. Post-processing for light source blending

Our losses explicitly prevent the network from “learning to inpaint” anything in the saturated regions, so its output

<sup>2</sup>We replace rather than exclude masked pixels because the perceptual loss in Eq. 9 requires a complete image.



(a) Input (b) CNN output (c) Mask  $M_f$  (d) Blended

Figure 6. We deliberately prevent the network from learning to inpaint the saturated regions (illuminants), so its output (b) is undefined in these regions. To preserve highlights, we compute a mask (c) for the saturated regions of the input. The masked area in the network output is then replaced by the input pixels, producing a more realistic final result (d) in which the flare is removed, but not the illuminant that produced it.

there can be arbitrary. In practice, it tends to remove the light source so that it is more similar to the surrounding pixels, as shown in Fig. 6(b). Since the goal of this work is to remove the flare, and not the light source, we post-process the network output to add back the light source.

A key observation is that the flare-causing light source is likely saturated in the input image (otherwise it would not result in a visible flare). Hence, it can be identified easily based on intensity. To create a gradual transition, we feather the mask  $M$  defined in Sec. 5.1 at its boundary to construct  $M_f$  (details and parameters are in the supplement). We blend the input and output images using the feathered mask in linear space (e.g., Fig. 6(d)):

$$I_B = I_F \odot M_f + f(I_F, \Theta) \odot (1 - M_f). \quad (11)$$

## 6. Results

To evaluate how the models trained on semi-synthetic data generalizes, we use three types of test data: (a) synthetic images with ground truth (Sec. 4), (b) real images without ground truth, and (c) real images with ground truth. To obtain (c), we capture a pair of images on a tripod with a bright illuminant just outside the field of view. In one image, bright flare-causing rays can enter the lens, producing artifacts. In the other image, we carefully place an occluder, also out of the field of view (e.g., a lens hood), between the illuminant and the camera, blocking the same rays.

### 6.1. Comparison with prior work

We provide quantitative and visual comparisons in Table 1 and Fig. 7. To eliminate the influence of the light source when computing metrics, the masked region is replaced by ground truth pixels following Eq. 7.

We evaluate all recent work in flare removal [1, 3, 24]. Notably, none of them attempt the general flare removal task. Instead, they use hand-crafted heuristics to remove one particular subset of flare (e.g., glare spots). As such, they have little effect on other artifacts such as reflections

and streaks and cause the PSNR and SSIM to be close to or even identical to the input. Since haze and reflections are two common flare artifacts, we also compare with dehazing [9] and dereflection [27] algorithms on our data.

For our method, we trained two variants, one using the architecture from [27], and the other using the popular U-Net [20]. Our method significantly outperforms existing methods and demonstrates the importance of our pipeline and dataset. We use the U-Net variant for the rest of the paper since it performs better.

Finally we also conducted a user study with 20 participants where each user is presented with a real image with lens flare alongside two predicted flare-free images: one from the U-Net and the other from one of the 5 baselines. We then ask the user to identify which of the two did better at removing lens flare. We use 52 images from 3 different sets: images captured by the same type of lens as in Sec. 4.2, images captured using five other lenses with different focal lengths, and images taken from [3]. To avoid bias, we shuffle the images in each instance of the study. As Table 2 shows, our method outperforms all others by a significant margin on all 3 datasets. Even on the dataset by Chabert [3], users strongly preferred our method to theirs (85% vs. 15%). Unsurprisingly, it performs slightly worse when tested on lenses not present in our training set.

Method	Synthetic		Real	
	PSNR	SSIM	PSNR	SSIM
Input image	21.13	0.843	18.57	0.787
Flare spot removal [3]	21.01	0.840	18.53	0.782
Flare spot removal [24]	21.13	0.843	18.57	0.787
Flare spot removal [1]	21.13	0.843	18.57	0.787
Dehaze [9]	18.32	0.829	17.47	0.745
Dereflection [27]	20.71	0.767	22.28	0.822
Ours + network [27]	28.49	0.920	24.21	0.834
Ours + U-Net [20]	<b>30.37</b>	<b>0.944</b>	<b>25.55</b>	<b>0.850</b>

Table 1. Quantitative comparison with related methods on synthetic and real data.

Comparison	Dataset 1	Dataset 2	Dataset 3
Ours: Flare spot removal [3]	98%: 2%	97%: 3%	85%:15%
Ours: Flare spot removal [24]	98%: 2%	93%: 7%	89%:11%
Ours: Flare spot removal [1]	100%: 0%	99%: 1%	88%:12%
Ours: Dehaze [9]	96%: 4%	91%: 9%	92%: 8%
Ours: Derelection [27]	83%:17%	78%:22%	64%:36%
Average	95%: 5%	92%: 8%	84%:16%

Table 2. Percent of images where the users favor our results (ours + U-Net) vs. prior work. Dataset 1 is captured using the same lens design as in Sec. 4.2. Dataset 2 is captured using five other lens types with different focal lengths. Dataset 3 contains images from [3]. We outperform existing methods in all categories, even on Chabert’s own dataset [3].

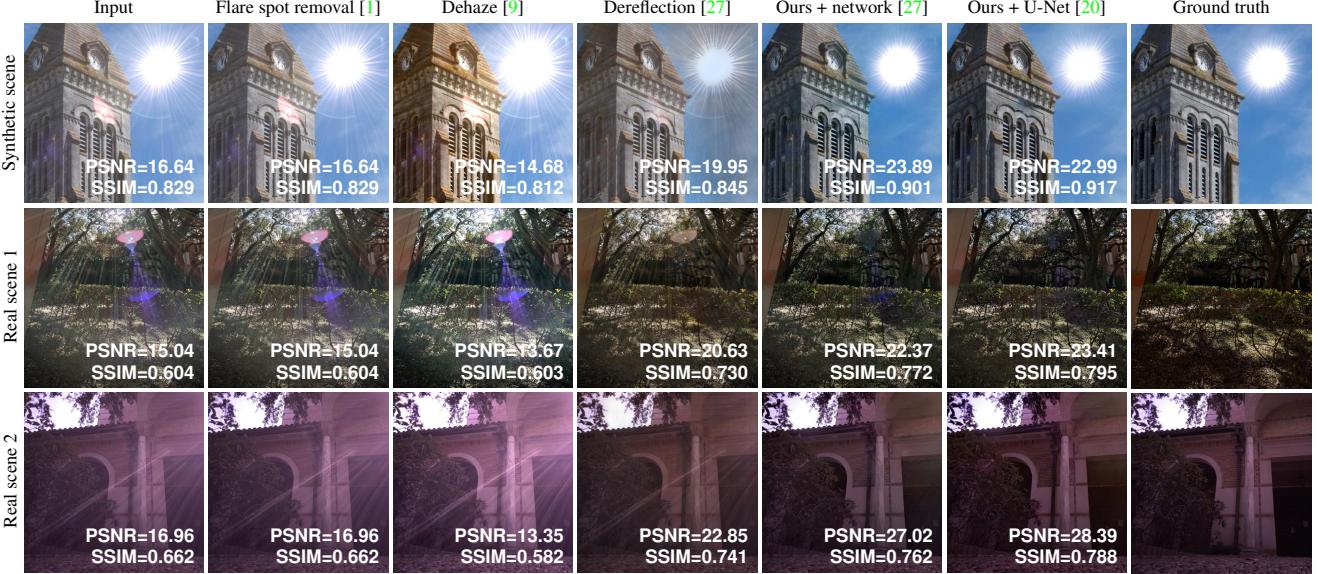


Figure 7. Visual comparison between three related methods and ours, evaluated on both synthetic and real scenes. Networks trained using our method remove lens flare more accurately and produce cleaner outputs.



Figure 8. Our method robustly removes lens flare of various shapes, colors, and locations on diverse real-world images. It generalizes reasonably to multiple light sources (column 2). When there is no significant flare (last column), the input is kept intact.

## 6.2. Ablation study

In this section, we study two key components in our procedure to demonstrate their impact on the output.

No $\mathcal{L}_F$	No sim. data	No captured data	<b>Full</b>
PSNR	24.84	24.44	<b>25.55</b>
SSIM	0.841	0.843	<b>0.850</b>

Table 3. Ablation study on the flare loss and different flare data.

**Flare loss** Since most flares are brighter than the underlying scene, we need to ensure that the network does not simply learn to darken all bright regions. We explicitly model this in our flare loss  $\mathcal{L}_F$ . In Fig. 9, we show test set results from the models trained with and without  $\mathcal{L}_F$ . Without  $\mathcal{L}_F$ , the network tends to remove some parts of bright objects even if they are not part of the flare.

**Captured and simulated flare data** In Sec. 4, we mentioned that the captured data mostly accounts for reflective

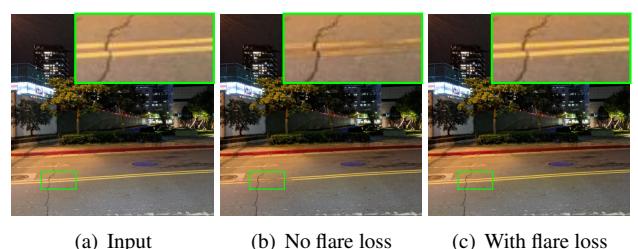


Figure 9. Without our flare loss  $\mathcal{L}_F$ , bright regions in the input (a) are incorrectly removed, especially on images taken at night (b).  $\mathcal{L}_F$  makes the model more robust to such errors (c).

flare, whereas the simulated data covers the scattering case. To show that both components are necessary, we train two ablated models, each with one of the sources excluded. As expected, models trained with flare-only images taken from the captured or simulated dataset alone underperform the model trained with both datasets, as shown in Fig. 10.



(a) Input      (b) Captured      (c) Simulated      (d) Both

Figure 10. The model trained with captured flare only (b) (containing mostly reflective and limited scattering flare) is less effective at removing streak-like scattering artifacts, whereas the simulation-only model (c) is unable to remove reflective flare patterns. Training with both datasets (d) produces superior results.

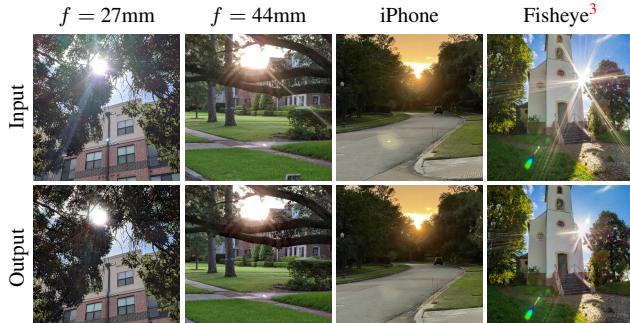


Figure 11. Although our dataset contains real flare patterns from only one camera of an Android smartphone ( $f = 13\text{mm}$ ), the trained model generalizes effectively to the phone’s other lenses ( $f = 27$  and  $44\text{mm}$ ), and another smartphone camera (iPhone). When tested on a drastically different lens design (e.g., a fish-eye mirrorless camera), the model performs less well on reflective flare, as expected, and still manages to remove scattering flare.

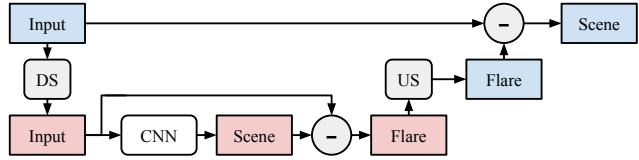
### 6.3. Generalization

**Across scenes** As our semi-synthetic dataset contains diverse flare patterns and scenes, the trained model generalizes well across a wide variety of scene types. As shown in Fig. 8, the input images contain flares with different shapes, colors, and locations, as well as scenes with varying subjects and lighting conditions. The model produces a high-quality output in most scenarios. When there is no flare in the input image, the network correctly performs a no-op.

**Across cameras** As mentioned in Sec. 4.2, all of our reflective flare training images come from one smartphone camera with focal length  $f = 13\text{mm}$ . We also test our model on other camera designs excluded from training. As shown in Fig. 11, the model is still able to reduce lens flare effectively, echoing the findings of the user study in Table 2.

That said, there is a limit on how far the model can generalize. For example, the model performs less well on images taken with an extremely different lens, such as a fisheye (Fig. 11, last column). This is especially true for the lens-dependent reflective component, as discussed in Sec. 4.2. Domain adaptation for drastically different camera designs is an interesting avenue for future work.

<sup>3</sup>Photo by Flickr user barit / CC BY-SA.



(a) Pipeline: blue and red blocks represent high- and low-res respectively.



(b) Input      (c) Low-res output      (d) High-res output

Figure 12. Because lens flares are mostly low-frequency, our algorithm can be trivially extended to high-res inputs. As (a) shows, we downsample (DS) a high-res input (b), predict a flare-free image (c) with our network (CNN), and compute the flare as the difference. This predicted flare is then upsampled (US) and subtracted from the input image to produce a high-res flare-free output (d).

### 6.4. High-resolution images

Our network is trained on  $512 \times 512$  images. The naïve way to apply our method to a higher-resolution input is to train at the desired resolution (e.g.,  $2048 \times 2048$ ), which requires 16x more bandwidth at both training and test time.

Fortunately, we can leverage the fact that lens flare is predominantly a low-frequency artifact. For a high-resolution image, we bilinearly downsample the input, predict a low-resolution flare-only image, bilinearly upsample it back to full resolution, and subtract it from the original input (see Fig. 12). This allows a network trained at a fixed low resolution to process high-resolution images without significant quality loss. On a Xeon E5 CPU, processing time for a  $2048 \times 2048$  input is reduced from 8s to 0.55s when running inference at  $512 \times 512$ .

## 7. Conclusion

We introduced a novel, tractable, and physically-realistic model for lens flare. By building a semi-synthetic data generation pipeline using a principled image formation model, we are able to train convolutional neural networks to recover a clean flare-free image from a single flare-corrupted image without the need for real training data. Our method is shown to achieve accurate results across a range of scenes and cameras. To our knowledge, this is the *first* general-purpose lens flare removal technique.

**Acknowledgments** We thank Sam Huynh, Chung-Po Huang, Xi Chen, and Lu Gao for their help in setting up the lab, and acknowledge the support from NSF grants IIS-1652633 and IIS-1730574.

## A. Simulating random scattering flare

To simulate scattering flare with our wave optics model, we randomly sample the aperture function  $A$ , the light source's 3D location  $(x, y, z)$ , and the spectral response function  $SRF$ .

### Notations

- $N(\mu, \sigma^2)$ : normal distribution with mean  $\mu$  and standard deviation  $\sigma$ .
- $U(a, b)$ : uniform distribution on the interval  $[a, b]$ .
- $k_\lambda$ : the wavenumber corresponding to wavelength  $\lambda$ .  $k_\lambda = 2\pi/\lambda$ .

### A.1. Aperture function

As Fig. 3(a) illustrates, we simulate dust and scratches with random dots and polylines on a clean disk-shaped aperture of radius  $R = 3000$  pixels.

For each aperture function, we randomly generate  $n_d \sim N(30, 5^2)$  dots with maximum radius  $r_d^{\max} \sim N(100, 50^2)$ . Each individual dot's radius  $r_d$  is drawn independently from  $U(0, r_d^{\max})$ .

Additionally, we also generate  $n_p \sim N(30, 5^2)$  polylines, each of which consists  $n_l \sim U(1, 16)$  line segments connected from end to end. The maximum line width for each aperture function is  $w_p^{\max} \sim N(20, 5^2)$ , and the width of each individual line is drawn independently from  $U(0, w_p^{\max})$ .

The opacity of each of the dots and polylines is sampled independently from  $U(0, 1)$ , and its location  $(u, v)$  on the aperture plane is also drawn uniformly.

### A.2. Phase shift

The light source's 3D location  $(x, y, z)$  determines the phase shift  $\phi_\lambda$  of the pupil function  $P_\lambda$ . As shown in Eq. 3, it consists of a linear term  $\phi_\lambda^S$  and a defocus term  $\phi_\lambda^{DF}$ .

The linear phase shift  $\phi_\lambda^S$  on the aperture plane becomes a spatial shift on the sensor plane due to the Fourier transform  $\mathcal{F}\{\cdot\}$  in Eq. 4. For an  $800 \times 800$  image sensor with the center as the origin, the center of the light source  $(x, y)$  is sampled from  $x, y \sim U(-500, 500)$ .

The term  $\phi_\lambda^{DF}(z)$  is the defocus aberration due to the mismatch between the in-focus depth  $z_0$  of the lens and the actual depth  $z$  of the light source. The analytical expression for  $\phi_\lambda^{DF}(z)$  is

$$\begin{aligned} \phi_\lambda^{DF}(z) &= k_\lambda \frac{u^2 + v^2}{2} \left( \frac{1}{z} - \frac{1}{z_0} \right) \\ &= k_\lambda \cdot r(u, v)^2 \cdot W_m(z) \end{aligned} \quad (12)$$

where  $k_\lambda$  is the wavenumber,  $(u, v)$  are aperture coordinates, and  $r(u, v) = \sqrt{u^2 + v^2}/R$  is the relative displace-

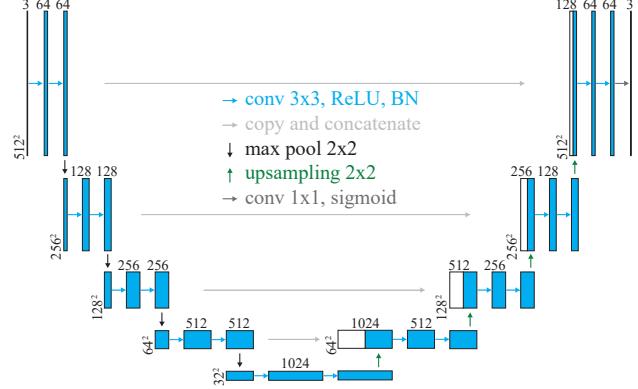


Figure 13. The U-Net architecture [20] we use in the flare removal task.

ment on the aperture plane.  $W_m(z)$  is defined as

$$W_m(z) = \frac{R^2}{2} \left( \frac{1}{z} - \frac{1}{z_0} \right) \quad (13)$$

and quantifies the amount of defocus in terms of the depth  $z$ . Since we do not target any specific camera devices (i.e., specific  $R$  and  $z_0$  values), it suffices to sample the value of  $W_m$  directly from  $N(0, \sigma^2)$ , with  $\sigma = 5/k_{\lambda=550\text{nm}}$ .

### A.3. Spectral response function

The spectral response  $SRF_c(\lambda)$  describes the sensitivity of color channel  $c$  to wavelength  $\lambda$ , where  $c \in \{R, G, B\}$ .

We model  $SRF_c(\lambda)$  as a Gaussian probability density function  $N(\mu_c, \sigma_c^2)$ . The mean  $\mu_c$  is the central wavelength of each channel in nanometers, and is drawn from  $\mu_R \sim U(620, 640)$ ,  $\mu_G \sim U(540, 560)$ , and  $\mu_B \sim U(460, 480)$ . The standard deviation  $\sigma_c$  represents the width of the passband of each color channel, and is drawn independently for each channel from  $U(50, 60)$ .

We then discretize the wavelength  $\lambda$  in  $SRF_c(\lambda)$  for each of the RGB channels at 5nm intervals from  $\lambda = 380\text{nm}$  to  $740\text{nm}$ , resulting in a 73-vector for each channel. The 3 vectors are stacked to produce an instance of  $SRF$  in the form of a  $3 \times 73$  matrix.

## B. Flare-only image augmentation

We augment the captured flare-only images  $F$  by applying random geometric and color transformations. For geometric augmentation, we generate a  $3 \times 3$  affine transform matrix with random rotation ( $\sim U(0, 2\pi)$ ), random translation ( $\sim U(-10, 10)$  pixels), random shear angle ( $\sim U(-\pi/9, \pi/9)$ ), and random scale ( $\sim U(0.9, 1.2)$  for  $x$  and  $y$  independently). For color augmentation, we multiply each color channel with a random weight in  $U(0, 10)$ . Pixel values are clipped to  $[0, 1]$  after the transformed flare image is composited with the clean image.

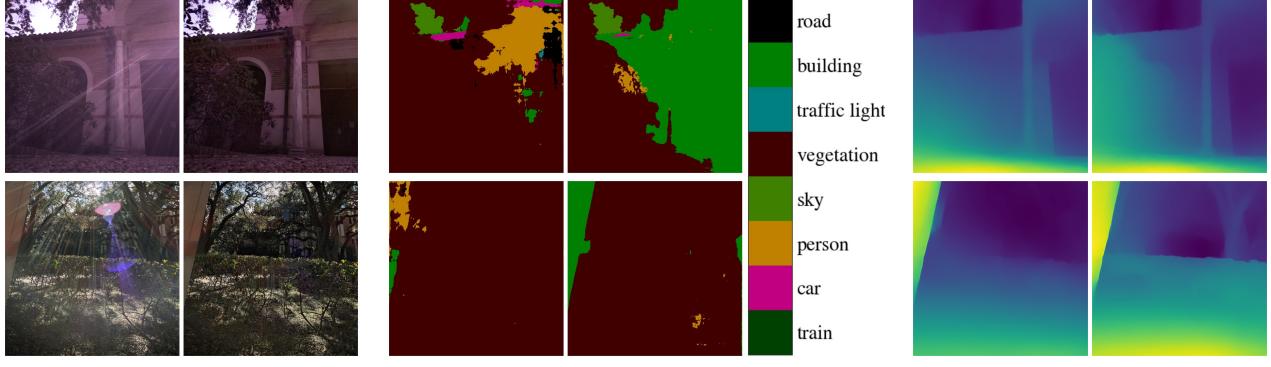


Figure 14. A pair of images with flare and our output (a). Semantic segmentation (b) has much less misclassification once flare is removed using our method. Similarly, monocular depth estimation (c) is more accurate (e.g., the bush on the left in the top image and the tree in the distance in the lower image).

## C. Networks and training details

To show the effectiveness of our method, we train two popular neural networks with distinct architectures. These networks have not previously been used for the task of removing lens flare. Using our method, both networks produce satisfactory results. We detail the network architectures and training configurations below.

### C.1. Context aggregation network (CAN)

We repurpose a network originally designed for reflection removal [27], which is a variant of the original CAN [26]. Starting from the  $512 \times 512 \times 3$  input image, the network first extracts features from a pre-trained VGG-19 network [22] at layers conv1\_2, conv2\_2, conv3\_2, conv4\_2, and conv5\_2. Next, these features are combined with the input image to form a 1475-channel tensor, which is subsequently reduced to 64 channels with a  $1 \times 1$  convolution. It is then passed through eight  $3 \times 3$  convolution layers with 64 output channels at dilation rates of 1 – 64. The last layer is a  $1 \times 1$  convolution with 3 output channels.

### C.2. U-Net

The U-Net [20] is shown in Fig. 13. The input image  $I_F$  is  $512 \times 512 \times 3$ . Each convolution operator consists of a  $3 \times 3$  convolution and ReLU activation. We use  $2 \times 2$  max pooling for downsampling and resize–convolution for upsampling. Concatenation is applied between the encoder and decoder to avoid the vanishing gradient problem. At the final layer, we use a sigmoid function to squeeze the activations to the  $[0, 1]$  range.

### C.3. Training details

We train both CAN and U-Net using our semi-synthetic dataset. The clean images come from the Flickr dataset

of [27]. The flare-only images are generated as described in the previous Sections, and augmented on the fly during training. Training lasted 1.2M iterations (approximately 60 epochs over the 20k images in the Flickr dataset) with batch size 2 on an Nvidia V100 GPU. We use the Adam optimizer [11] with default parameters and a fixed learning rate of  $10^{-4}$ .

## D. Mask feathering

As mentioned in Sec. 5.2 of the main text, we detect the light source and create a feathered mask  $M_f$  to smoothly blend the bright illuminants into the network output.

Starting from a binary mask  $M$  (the pixels where the input luminance is greater than 0.99), we apply morphological opening with a disk kernel of size equal to 0.5% of the image size. To find the primary illuminant, we partition the mask into connected components and find the equivalent diameter  $D$  of the largest region (the diameter of a circle with the same area as the region). We then blur the binary mask  $M$  using a disk kernel of diameter  $D$ . Finally, we scale the blurred mask intensity by 3 in order to guarantee that all the pixels inside the illuminant are saturated after blurring, and clip the mask to  $[0, 1]$ . This is the  $M_f$  used in Eq. 11 which has a feathered edge.

## E. Flare removal for downstream tasks

In Fig. 14, we show that removing lens flare may benefit downstream tasks such as semantic segmentation and depth estimation. We look forward to thoroughly investigating the effect of reduced flare on a range of computer vision algorithms.

## F. More results

Fig. 15 provides more visual comparisons between our method and prior work. Overall, we have 20 real test images with ground truth. The complete results, including the input images and output images from different methods, can be found at <https://yichengwu.github.io/flare-removal/>.

We also show 24 additional test images captured using the same type of lens as in Sec. 4.2 of the main text (Fig. 16), and 24 test images captured using 7 other lens types with different focal lengths (Fig. 17).

While our results are mostly satisfactory, there are certainly cases where it does not perform well. They typically have strong flare over the entire image. There is a trade-off between flare removal and scene preservation, and we prefer the latter to reduce artifacts.

## References

- [1] CS Asha, Sooraj Kumar Bhat, Deepa Nayak, and Chaithra Bhat. Auto removal of bright spot from images captured against flashing light source. *IEEE DISCOVER*, 2019. [1](#), [2](#), [6](#), [7](#), [12](#)
- [2] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. *CVPR*, 2019. [2](#)
- [3] Floris Chabert. Automated lens flare removal. Technical report, Department of Electrical Engineering, Stanford University, 2015. [1](#), [2](#), [6](#)
- [4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*, 2018. [10](#)
- [5] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE TIP*, 2004. [2](#)
- [6] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. A generic deep architecture for single image reflection removal and image smoothing. *ICCV*, 2017. [2](#)
- [7] K Faulkner, CJ Kotre, and M Louka. Veiling glare deconvolution of images produced by x-ray image intensifiers. *International Conference on Image Processing and its Applications*, 1989. [2](#)
- [8] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company Publishers, 2005. [4](#)
- [9] Kaiming He, Jian Sun, and Xiaou Tang. Single image haze removal using dark channel prior. *IEEE TPAMI*, 2010. [6](#), [7](#), [12](#)
- [10] Matthias Hullin, Elmar Eisemann, Hans-Peter Seidel, and Sungkil Lee. Physically-based real-time lens flare rendering. *SIGGRAPH*, 2011. [4](#)
- [11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2017. [10](#)
- [12] Rudolf Kingslake. *Optics in photography*. SPIE press, 1992. [2](#)
- [13] Katrin Lasinger, René Ranftl, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. [10](#)
- [14] Sungkil Lee and Elmar Eisemann. Practical real-time lens-flare rendering. *Computer Graphics Forum*, 2013. [4](#)
- [15] Chao Li, Yixiao Yang, Kun He, Stephen Lin, and John E Hopcroft. Single image reflection removal through cascaded refinement. *CVPR*, 2020. [2](#)
- [16] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. *CVPR*, 2018. [4](#)
- [17] Rui Qian, Robby T Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. *CVPR*, 2018. [2](#)
- [18] Ramesh Raskar, Amit Agrawal, Cyrus A Wilson, and Ashok Veeraraghavan. Glare aware photography: 4d ray sampling for reducing glare effects of camera lenses. *SIGGRAPH*, 2008. [2](#)
- [19] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010. [2](#)
- [20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015. [6](#), [7](#), [9](#), [10](#), [12](#)
- [21] J Anthony Seibert, O Nalcioglu, and W Roeck. Removal of image intensifier veiling glare by mathematical deconvolution techniques. *Medical physics*, 1985. [2](#)
- [22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. [5](#), [10](#)
- [23] Eino-Ville Talvala, Andrew Adams, Mark Horowitz, and Marc Levoy. Veiling glare in high dynamic range imaging. *ACM TOG*, 2007. [2](#)
- [24] Patricia Vitoria and Coloma Ballester. Automatic flare spot artifact detection and removal in photographs. *Journal of Mathematical Imaging and Vision*, 2019. [1](#), [2](#), [6](#)
- [25] Wei Wei, Deyu Meng, Qian Zhao, Zongben Xu, and Ying Wu. Semi-supervised transfer learning for image rain removal. *CVPR*, 2019. [2](#)
- [26] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. [10](#)
- [27] Xuaner Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. *CVPR*, 2018. [2](#), [5](#), [6](#), [7](#), [10](#), [12](#)

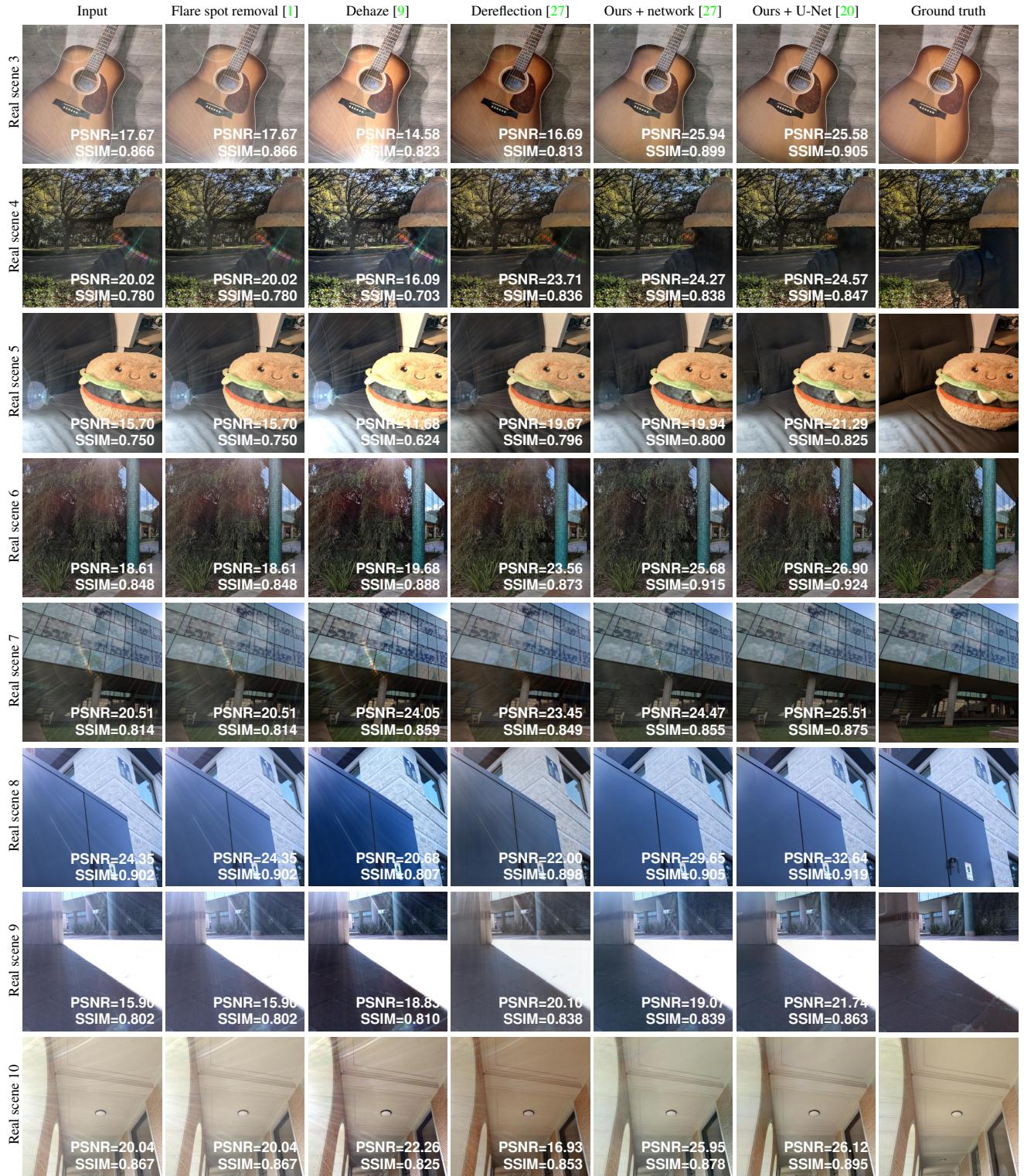


Figure 15. More visual comparison between three related methods and ours on real scenes, with PSNR and SSIM values.

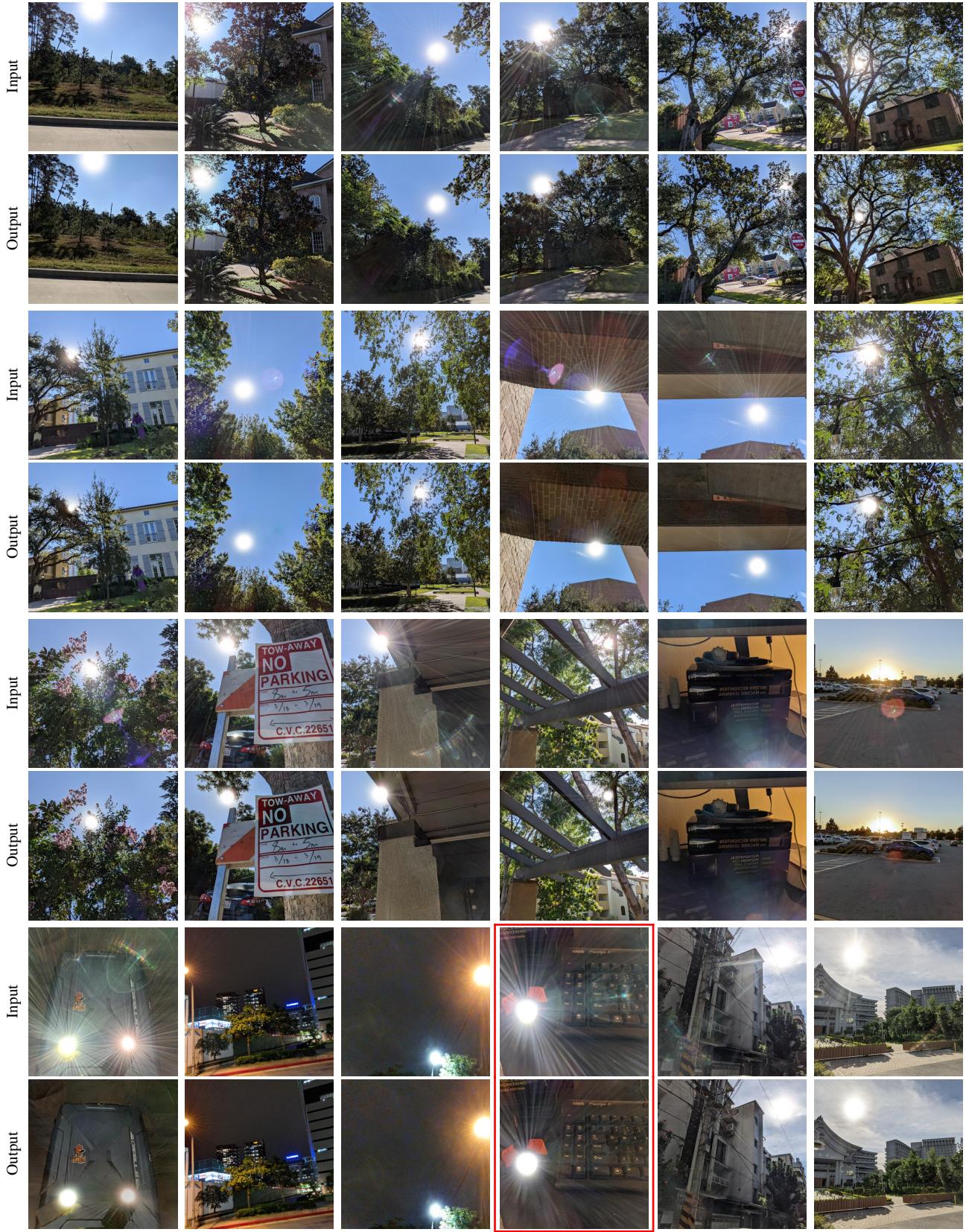


Figure 16. 24 testing images captured by the same type of lens as in Sec. 4.2. Our method is effective in removing most of the lens flare, with occasional failures where it only removes a part of the flare (red box).

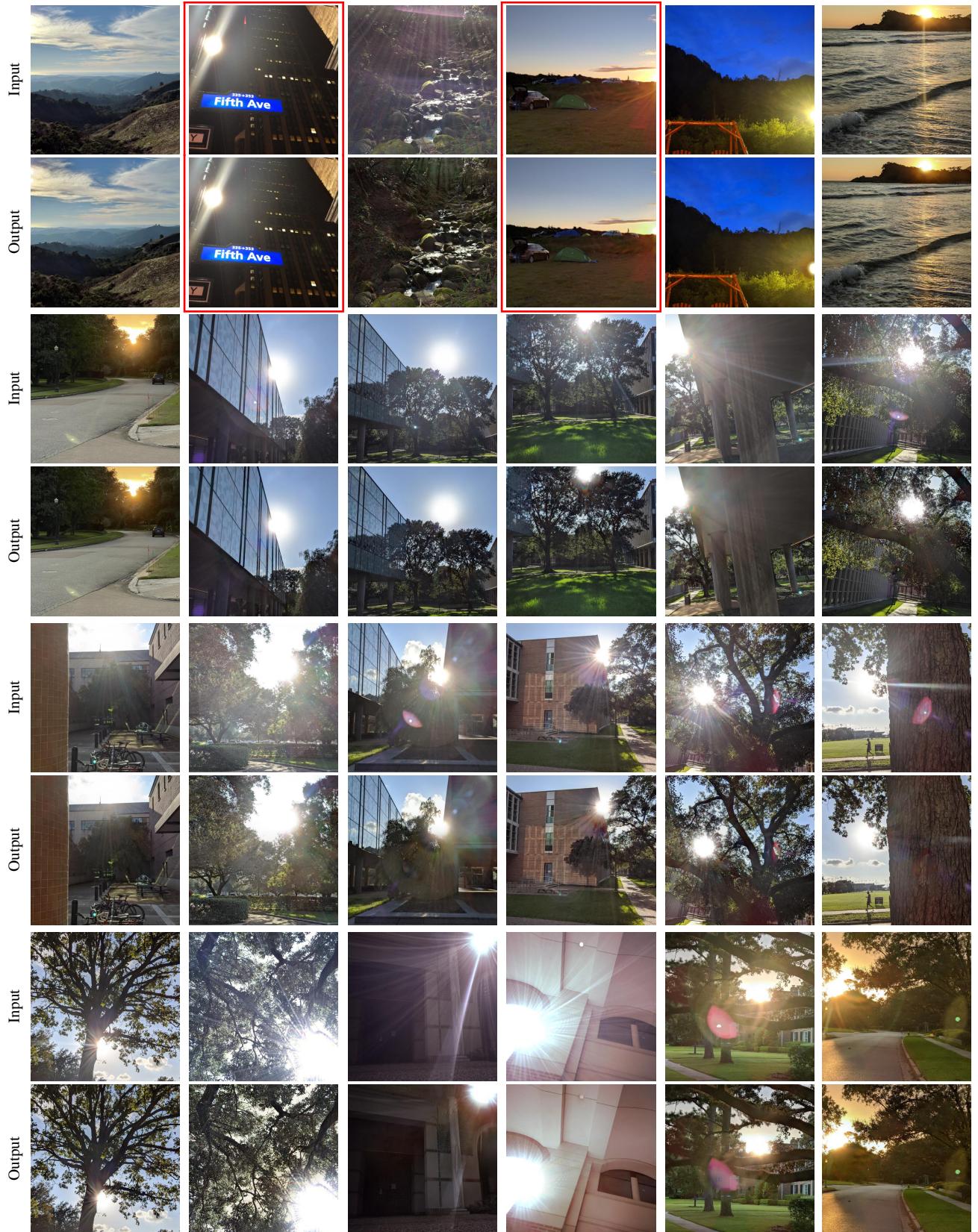


Figure 17. 24 testing images captured by 7 other lens types with different designs and focal lengths. Our method successfully remove most flares, with a few occasional failures that the algorithm either fails to identify flare (e.g., first red box) or incorrectly removes non-flare highlights (e.g., clouds in the second red box).