GAN-BASED RAIN NOISE REMOVAL FROM SINGLE-IMAGE CONSIDERING RAIN COMPOSITE MODELS

Takuro Matsui, Masaaki Ikehara

EEE Dept., Keio Univ., Yokohama, Kanagawa 223-8522, Japan
{matsui,ikehara}@tkhm.elec.keio.ac.jp

ABSTRACT

Under severe weather conditions, outdoor images or videos captured by cameras can be affected by heavy rain and fog. In this paper, we address a single-image rain removal problem (de-raining). As compared to video-based methods, single-image based methods are challenging because of the lack of temporal information. Although many existing methods have tackled these challenges, they suffer from overfitting, over-smoothing, and unnatural hue change. To solve these problems, we propose a GAN-based de-raining method. The optimal generator is determined by experimental comparisons. To train the generator, we learn the mapping between rainy and residual images from the training dataset. Besides, we synthesize a variety of rainy images to train our network. In particular, we focus on not only the orientations and scales of rain streaks but also the rainy image composite models. Our method also achieves better performance on both synthetic and real-world images than state-of-the-art methods in terms of quantitative and visual performances.

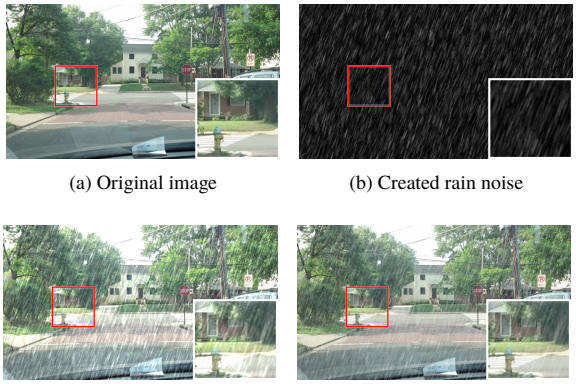
Index Terms — Generative adversarial network, single-image de-raining, deep learning, image restoration, residual learning.

1. INTRODUCTION

Image restoration and enhancement are of considerable practical concern. The number of outdoor vision systems, such as surveillance cameras and dashboard cameras has been increasing in the past years. One major issue for autonomous navigation systems is to drive under bad-weather conditions. Effective rain removal (de-raining) methods are required in multiple types of practical situations. Different from common image de-noising tasks, the de-raining task is more challenging because rain have no fixed shape and orientation.

For the last few decades, several investigations have been conducted on removing rain noises from a rainy observation. De-raining methods can be categorized into two types: one is video-based methods [1–3] and the other is single-image based methods. As compared to the video-based methods, the single-image based methods are more ill-pose and difficult problem for the sake of lack of temporal data.

In this paper, we only address the single-image de-raining such as [4–11]. To our knowledge, Kang *et al.* [4] first tackle single-image de-raining based on morphological component analysis (MCA). Luo *et al.* [5] introduce a discriminative sparse coding approach. However, when the input image has structures similar with the rain streaks, this method tends to leave rain noises (under de-rain). Although Huang *et al.* [6] propose a self-learning-based image decomposition method, it results in over-smoothing some regions (over de-rain). Recently, deep learning-based approaches have been proposed [12]. In [9] by Fu *et al.*, it is observed that deep learning based methods are valid for de-raining. Although this approach is valid for de-raining, the rain structure in low-frequency component causes blur and haziness. Fu *et al.* extended the network structure using Res-block [13] in [10]. Yang *et al.* proposed a CNN structure that can jointly detect and remove rain streaks. Although their single CNN-based methods success in de-raining, they suffer from over de-rain and under de-rain. Zhang *et al.* [11] propose density aware GAN (Generative Adversarial Network) based method. Their method can clearly remove even heavy rain streaks. However, they tend to over smooth and lose important details. Thus, trade-off between removing rain streaks and preserving textures is one of the most challenging problems.



(a) Original image (b) Created rain noise

(c) Additive composite model (d) Screen blend model

Fig. 1. Comparison of two composite models.

In order to address these problems, we propose a GAN-based residual deep network. Residual learning can save the computational cost and success in several image tasks, such as image recognition task [13] and an image de-noising task [14]. Our results show that residual learning achieves better performance than the image decomposition approach.

In addition, we focus on a rainy image model. Our proposed method assumes two rainy image models as training dataset to make the network suitable to many real-world data. In [4, 15], the authors use a linear additive composite model (Fig. 1(c)) for synthesizing. Some recent studies, such as those conducted in [5,9,11], do not adopt the additive model but use a non-linear screen blend model (Fig. 1(d)). Because each model has both advantages and disadvantages, either is not always applicable to real-world images. Accordingly, we prepare training dataset which consists of images made by both an additive model and a screen blend model. Our experimental results demonstrate that this two-composite-model dataset is applicable to various real rainy images.

In summary, this paper makes the following four contributions:

1) We explore an optimal deep learning structure for de-raining. Inspired by the success of GANs in other image processing tasks, we introduce GAN for de-raining.
2) We introduce residual learning to remove rain streaks without losing the textures and edges. We learn the relationship between rainy images and residual images.
3) To create synthetic rainy images, we introduce an automatic rain streaks generator. Our proposed method can easily change parameters on Python, which results in saving time and effort to obtain natural rain streaks.
4) We propose combination of two composite models for creating synthetic rainy images. Although most existing methods use only one rain composite model, it is not enough for real-world images. A combination of these models achieves better performance than using either of them.

This paper is organized as follows. Section 2 describes a brief review of de-raining. The details of our proposed method are given in Section 3. In Section 4, we present experimental results. Finally, Section 5 concludes the paper with a concise summary.

2. BACKGROUND AND RELATED WORKS

2.1. Single-image rain removal

Unlike video-based de-raining [1–3], single-image de-raining is an extremely challenging task. That is because of its ill-posed nature and the unavailability of temporal information. In single-image rain removal, prior-based methods have been proposed. These include sparsity-based methods [4, 5], low-rank representation-based methods [7] and Gaussian Mixture Model-based methods [8]. These prior-based methods tend to leave many rain streaks and over-smooth the details.

Recent studies introduce deep learning for single-image de-raining. These include simple CNN-based approaches [9, 10, 12] and GAN based approaches [11, 16]. To train the network, synthetic rainy image dataset and corresponding clean image dataset are used.

2.2. Rainy image composite models

In order to make synthetic rain images, several models have been proposed. The linear additive composite model in (1) is the simplest one. This is based on the hypothesis that rainy image $\boldsymbol{y}_{\text{add}}$ is broken down into background part $\boldsymbol{x}$ and rain streaks part $\boldsymbol{v}$:

$$\boldsymbol{y}_{\text{add}} = \boldsymbol{x} + \boldsymbol{v}. \tag{1}$$

As calculated values of $\boldsymbol{y}_{\text{add}}$ is clipped between 0 and 1 in the image processing, we can rewrite (1) as:

$$\boldsymbol{y}_{\text{add}} = \min\left(\boldsymbol{x} + \boldsymbol{v}, \boldsymbol{1}\right), \tag{2}$$

where $\boldsymbol{1}$ represents an array filled with ones and $\min\left(\boldsymbol{X}, \boldsymbol{Y}\right)$ denotes the operation in which the smallest elements from $\boldsymbol{X}$ or $\boldsymbol{Y}$ are taken. This model tends to generate unnatural results when the original image contains bright region such as clouds or white objects (Fig. 1(c)). Otherwise, it is an optimal composite method for real-world extremely heavy rain.

On the other hand, a screen blend model is also adopted to synthesize images. In that model, to avoid unnatural appearance of the additive model, multiplied $\boldsymbol{x}$ and $\boldsymbol{v}$ are subtracted from the sum of them as:

$$\boldsymbol{y}_{\text{blend}} = \boldsymbol{x} + \boldsymbol{v} - \boldsymbol{x} \circ \boldsymbol{v}, \tag{3}$$

where $\circ$ indicates an element-wise multiplication operator. The images synthesized by the screen blend model look natural when an image contains shining part or dark part (Fig. 1(d)).

3. PROPOSED METHOD

We propose a GAN-based model in which the generator detects rain streaks and the discriminator judges whether the input clean image is a de-rained output (fake) or a clean image (true). The overall framework is illustrated in Fig. 2. While common CNN-based methods directly output de-rained images, the output of our generator is the rain streaks. This residual learning enhances the de-raining performance. In the training process, several number of rainy images are required. Since we do not have ground truth of real-world rainy images, we create pairs of synthetic rainy and clean image datasets. To be robust against many rain conditions, our synthetic rainy images are generated by two composite models.

3.1. The network structure of proposed GAN

During a test phase, in the input of our proposed network, the input is a rainy image $\boldsymbol{y}$ and the final output is a de-rained image $\tilde{\boldsymbol{x}}$ in Fig.2. Whereas, during the training phase, there are mainly four differences with the conventional methods

First, we do not use an image decomposition method. Many conventional methods divide the image into high frequency domain and low frequency domain, and each domain is processed independently. The problem is that they leave the rain noise in low frequency domain, which makes the restored image whitish.

Second, we use residual learning. In most CNN-based conventional methods, they directly output the de-rained image. The problem is that a tremendous amount of training dataset and training time are required to optimize the network. This is because it is difficult to recognize edges of the image and the rain streaks. Besides, the restoration of the contaminated image needs high computational cost. To solve these problems, we do not directly output a de-rained image $\tilde{\boldsymbol{x}}$ but output a residual image $\boldsymbol{y} - \tilde{\boldsymbol{x}}$.

Third, we introduce UNet structure [17] to detect rain streaks. In order to distinguish rain streaks and background, both global features and local features have to be considered. Inspired by the success of UNet for image segmentation tasks [17], we adopt UNet architecture.

Fourth, we propose a GAN structure. The discriminator identifies whether the input is a real clean image or a generated clean image. To enhance the performance of the discriminator, we combine rainy image $\boldsymbol{y}$ with the de-rained image $\tilde{\boldsymbol{x}}$ as a fake data. Similarly, the real image is the rainy image $\boldsymbol{y}$ and the ground truth clean image $\boldsymbol{x}$.

3.2. Loss function

Our goal is to optimize parameters of the generator that minimize the following loss function:

$$\mathcal{L}_G = \mu_1 \mathcal{L}_{\text{MSE}} + \mu_2 \mathcal{L}_{\text{adv}}, \tag{4}$$

where μ_1 and μ_2 are the coefficients which are empirically determined. The first term, $\mathcal{L}_{\text{MSE}}$ is a mean square error between residual images $\boldsymbol{y_n} - \boldsymbol{x_n}$ and output images $G(\boldsymbol{y}_n; \boldsymbol{\Theta_G})$. The loss function is expressed as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2N} \sum_{n=1}^{N} \|(\boldsymbol{y}_n - \boldsymbol{x}_n) - G(\boldsymbol{y}_n; \boldsymbol{\Theta_G})\|_2^2, \tag{5}$$

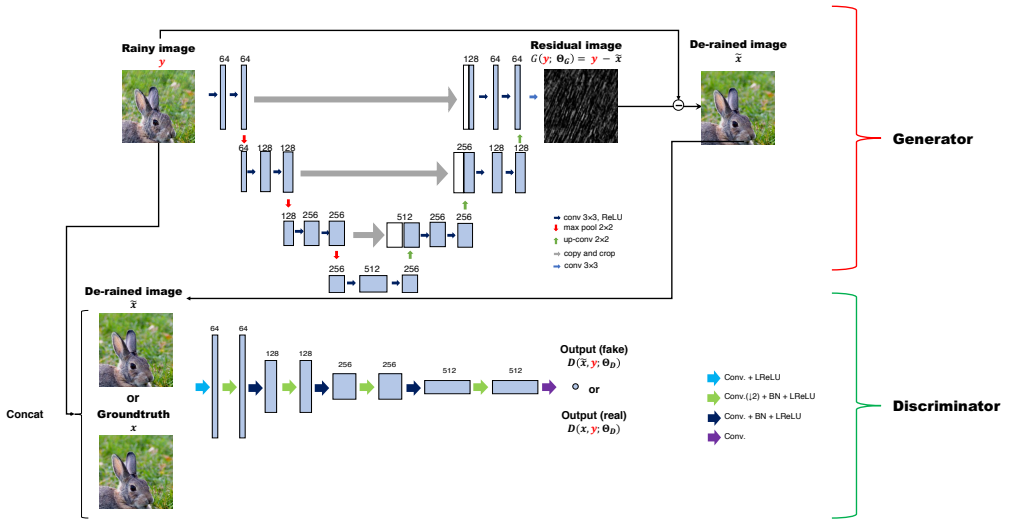


Fig. 2. The proposed framework.

where n and N indicate an image index and the total number of images, respectively. $\boldsymbol{\Theta_G}$ denotes the learned parameters of the generator. The second term of the Eq. 4, $\mathcal{L}_{\text{adv}}$ is an adversarial loss. The loss function is to optimize the discriminator to evaluate whether the input is a true clean image or a fake clean image. The following equation is the adversarial loss.

$$\mathcal{L}_{\text{adv}} = \frac{1}{2N}\sum_{n=1}^{N} \|1 - D(\boldsymbol{x}_n, \boldsymbol{y}_n; \boldsymbol{\Theta_D})\|_2^2. \tag{6}$$

On the other hand, the discriminator parameters are optimized by minimizing the following loss function:

$$\mathcal{L}_D = \frac{1}{2N}\sum_{n=1}^{N} \left(\|D(\tilde{\boldsymbol{x}}, \boldsymbol{y}; \boldsymbol{\Theta_D})\|_2^2 + \|V - D(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta_D})\|_2^2\right), \tag{7}$$

where V indicates a random value matrix which follows Gaussian distribution with an average of 1. To enable the discriminator trained effectively, we use random values instead of constant values.

3.3. Generating rain streaks

To train the mapping between rainy and clean images, we are required to create many natural rainy images from clean images. However, we cannot simultaneously get the same location images on a rainy day and a sunny day. Instead, we composite a clean image $\boldsymbol{x}$ and rain streaks $\boldsymbol{v}$ for training. Unlike the previous method as [9, 11] in which the authors synthesize rain streaks using Photoshop, we can easily generate rain noise $\boldsymbol{v}$ and control the densities and angles of rain on Python. As shown in Fig. 3, generating rain noise process follows three steps.

First, uniformly distributed random numbers $\boldsymbol{u} \in \mathcal{U}(0, 1)$ are generated with the same size as the clean image. To shift the average value and normalize the number, we adjust the noise amount σ_a and clip them between 0 and 1. One element of the random noise $v_i \in \boldsymbol{v}$ is calculated as:

$$v_i \leftarrow \max\left(\min\left(\sigma_a(u_i - \lambda) + \lambda, 1\right), 0\right), \tag{8}$$

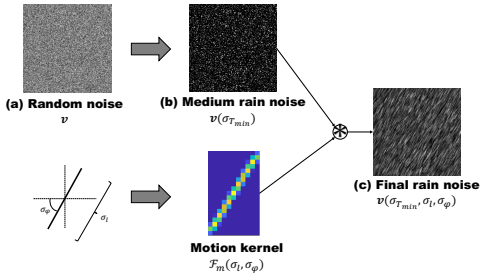


Fig. 3. The flow of generating rain noises.

where $\lambda = 0.5$. Generated random noise is show in Fig. 3 (a). Next, the generated noise needs to be blurred by a Gaussian filter $\boldsymbol{\mathcal{F}}_g$. Filtered output is normalized using values of $\sigma_{T_{\text{min}}}$ and $\sigma_{T_{\text{max}}}$. These two thresholds are used to reduce the amount of noise and boost its contrast. The calculated values are clipped between 0 and 1 as follows:

$$v_i \leftarrow \max\left(\min\left(\frac{\hat{v}_i - \sigma_{T_{\text{min}}}}{\sigma_{T_{\text{max}}} - \sigma_{T_{\text{min}}}}, 1\right), 0\right), \tag{9}$$

where $\hat{\boldsymbol{v}} = \boldsymbol{\mathcal{F}}_g \boldsymbol{v}$. The medium rain noise looks like Fig. 3 (b). Lastly, natural rain streaks are reproduced by giving the movement with a certain direction. After applying a motion filter $\boldsymbol{\mathcal{F}}_m(\sigma_l, \sigma_\varphi)$, we adjust the rain scale σ_s. The filter $\boldsymbol{\mathcal{F}_m}$ is a function of the length σ_l and angle σ_φ.

$$\boldsymbol{v} \leftarrow \sigma_s \boldsymbol{\mathcal{F}}_m(\sigma_l, \sigma_\varphi)\boldsymbol{v}. \tag{10}$$

The final rain noise is show in Fig. 3 (c). To simplify the noise model, we fix some parameters except for $\boldsymbol{v}$ as a function of $\sigma_{T_{\text{min}}}$, σ_l and σ_φ. In the above procedure, we can obtain natural rain noise $\boldsymbol{v}(\sigma_{T_{\text{min}}}, \sigma_l, \sigma_\varphi)$.

4. EXPERIMENTAL RESULTS

We evaluate the performance of our de-raining method by conducting experiments on both synthetic rainy images and real-world images.

Table 1: Quantitative comparisons with state-of-the-art using PSNR (dB) and SSIM on synthetic test images. In the upper part, results of images from Rain4 are shown. The lower part represents an average of each dataset.

Images	Rainy image	DSC [5]	DerainNet [9]	DID-MDN [11]	Ours
		PSNR(dB)			
Umbrella	26.58	31.68	26.30	29.24	35.93
Bird	18.37	23.77	19.23	26.75	27.79
Rabbit	23.89	26.98	19.16	25.04	29.86
Dock	29.58	29.28	29.16	28.42	34.49
Rain12	28.82	28.71	28.96	26.58	31.38
Rain100	21.15	21.04	21.71	21.08	22.11
BSD100	22.48	26.65	22.89	23.78	30.76
Urban100	22.71	26.21	22.71	21.27	30.38
		SSIM			
Umbrella	0.858	0.910	0.902	0.924	0.976
Bird	0.729	0.815	0.847	00.910	0.924
Rabbit	0.942	0.985	0.908	0.914	0.951
Dock	0.941	0.972	0.960	0.923	0.980
Rain12	0.910	0.916	0.939	0.918	0.942
Rain100	0.768	0.764	0.831	0.886	0.812
BSD100	0.841	0.878	0.898	0.854	0.953
Urban100	0.888	0.891	0.888	0.752	0.955

We compare the de-raining performance with three state-of-the-art methods. We select the prior-based method (DSC [5]), the simple CNN-based method (DerainNet [9]) and the GAN-based method (DID-MDN [11]). All of these methods are implemented with the source codes the authors distribute.

4.1. Training dataset and Learning parameters

Since real rainy images and the corresponding sunny images are not available simultaneously, we have to create synthetic many rainy images. We combine clean images and generated rain streaks to prepare training dataset. Our dataset includes a total of 4900 images including 900 images used in [9] and 4000 images used in [11]. During a training phase, we randomly crop $384 \times 384 \times 3$ patches. In addition, these patches are randomly flipped horizontally or vertically for data augmentation. We create completely random rain noise pattern with different angles, scales and densities. Specifically, we set rain noise parameters $\boldsymbol{v}(\sigma_{T_{\min}}, \sigma_l, \sigma_\varphi, \sigma_s, \sigma_a)$ to $\sigma_{T_{\min}} \in [0.54, 0.62]$, $\sigma_l \in [5, 15]$, $\sigma_\varphi \in [50, 130]$, $\sigma_s \in [1.1, 1.5]$,and $\sigma_a \in [0, 2]$. Note that $\sigma \in [\sigma^a, \sigma^b]$ represents σ is taken the value within a range of σ^a to σ^b. A clean image and the rain noise are synthesized by either an additional model or a screen blend model for each patch.

We start the training with a base learning rate of 0.0002 and use Adam solver to optimize training parameters. In Adam, we set two learning rates as $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The Pytorch framework is used to train our entire network. The models are trained for up to 4900×40 iterations with a batch size of four. During training we set the adversarial loss ratio in 6 to $\mu_1 = 1$, $\mu_2 = 0.001$.

4.2. Results on synthetic images

We compare the performance of three previous methods both qualitatively and quantitatively on several synthetic rainy image datasets. We test on commonly used test datasets "Rain12" [9] and "Rain100" [11]. Furthermore, we newly create synthetic rainy images using clean outdoor image datasets from "Rain4 " [9], "BSD100", and "Urban100".

As a subjective evaluation, Fig. 4 shows the visual comparison for two synthetic rainy images. As can be seen in the third column, method [5] leaves a considerable amount of rain streaks. For method [9] in the forth column, rain is almost entirely removed but the overall image turns whitish, specifically feathers of the bird. Method [11] results in over de-raining and losing important details. In contrast, our proposed model can remove rain streaks well with the contrast of the images remained. The combination of residual learning and UNet structure leads to these great performance.

On the other hand, as an objective evaluation, we compare the de-raining performance using PSNR and SSIM in Table 1. PSNR results in Table 1 demonstrate that the whiteness throughout an image in method [9] causes a lower PSNR. Also, SSIM results reveal that under de-raining in [5] and over de-raining in [11] lead to lower SSIM. Meanwhile, since our proposed method can remove rain streaks without losing the important details, we achieve higher PSNR and SSIM than other conventional methods.

4.3. Results on real-world images

In this section, we make sure whether our proposed method is also applicable to real-world rainy images. We collect many real-world rainy images from the Internet and the test dataset used in [9]. While synthetic images have only rain streaks, real-world images include haze as well as rain streaks. For clear appearance, we apply de-hazing method [18] as a post-processing. We compare visually de-raining performance with state-of-the-art methods. Fig. 5 shows the visual comparison. As can be seen, method [5] suffer from under de-raining for all images. Although method [9] can remove rain streaks, they blur textures and edges. Results of method [11] are over de-rained and they lose important details. In contrast, our proposed method can remove rains with the detail textures preserved. One can see that our model works well not only for synthetic images but also for real-world ones.

5. CONCLUSION

We have proposed a GAN-based de-raining network trained with mixture of two rain image composite models. To capture global features and local features of rain streaks, we use UNet structure as the generator. In addition, residual learning improves the performance of training and testing process. Moreover, to generate synthetic rain noise for training, we introduce a completely automatic rain noise generator. We compare the de-raining performance with several state-of-the-art methods for synthetic images and real-world images. The results on synthetic data demonstrate that our proposed model noticeably outperforms other conventional methods both quantitatively and qualitatively. For real-world data, our diverse rainy dataset can remove rain streaks and enhance the hazy images without losing important details.

6. REFERENCES

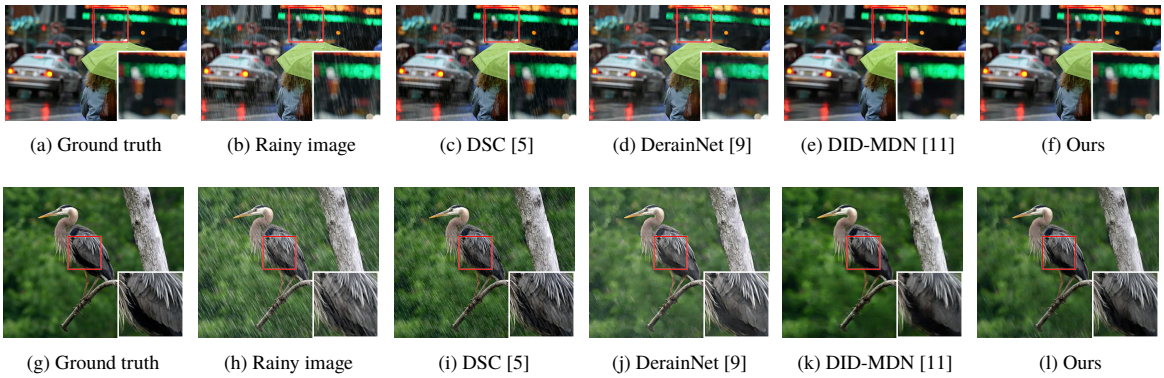
[1] K. Garg and S. K. Nayar, "Detection and removal of rain from videos," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, June 2004, vol. 1, pp. I–528–I–535 Vol.1.

[2] Varun Santhaseelan and Vijayan K Asari, "Utilizing local phase information to remove rain from video," *International Journal of Computer Vision*, vol. 112, no. 1, pp. 71–89, 2015.

[3] Minghan Li, Qi Xie, Qian Zhao, Wei Wei, Shuhang Gu, Jing Tao, and Deyu Meng, "Video rain streak removal by multiscale convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6644–6653.

[4] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu, "Automatic single-image-based rain streaks removal via image decomposition," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1742, 2012.

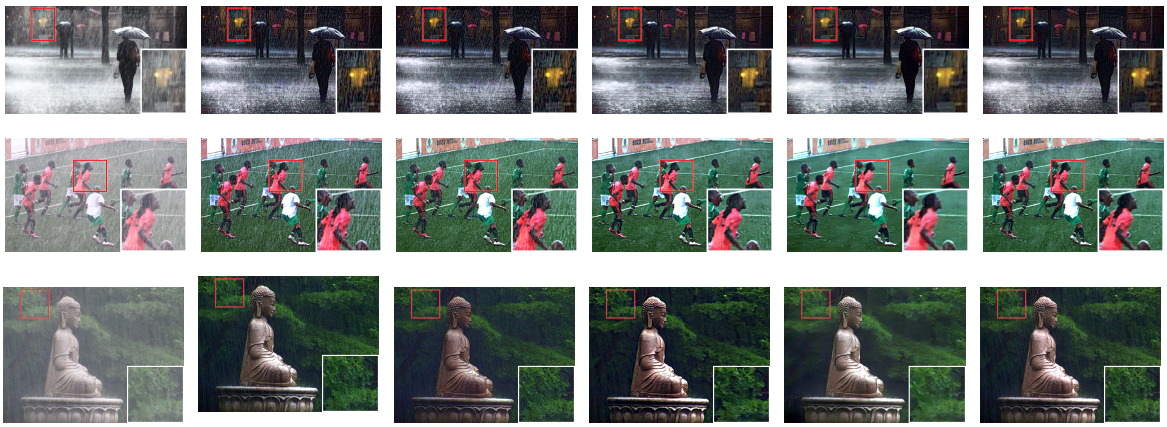
[5] Y. Luo, Y. Xu, and H. Ji, "Removing rain from a single image via discriminative sparse coding," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 3397–3405.



(a) Ground truth (b) Rainy image (c) DSC [5] (d) DerainNet [9] (e) DID-MDN [11] (f) Ours

(g) Ground truth (h) Rainy image (i) DSC [5] (j) DerainNet [9] (k) DID-MDN [11] (l) Ours

Fig. 4. Results on two synthetic test images from Rain4, (first row) "Umbrella" and (third row) "Bird". The second row and the forth one are histogram of the first row and the third one respectively.



(a) Rainy images (b) Dehazed rainy image (c) DSC [5] (d) DerainNet [9] (e) DID-MIN [11] (f) Ours

Fig. 5. Three results on real-world rainy images, "Street" (top), "Soccer" (middle) and "buddha" (bottom). All algorithms use image de-hazing [18] as a post-processing.

[6] D. A. Huang, L. W. Kang, Y. C. F. Wang, and C. W. Lin, "Self-learning based image decomposition with applications to single image denoising," *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 83–93, Jan 2014.

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

[8] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown, "Rain streak removal using layer priors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2736–2744.

[9] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, "Clearing the skies: A deep network architecture for single-image rain removal," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2944–2956, June 2017.

[10] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley, "Removing rain from single images via a deep detail network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3855–3863.

[11] He Zhang and Vishal M Patel, "Density-aware single image de-raining using a multi-stream dense network," in *CVPR*, 2018.

[12] T. Matsui, T. Fujisawa, T. Yamaguchi, and M. Ikehara, "Single-image rain removal using residual deep learning," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, pp. 3928–3932.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, July 2017.

[15] Peter C Barnum, Srinivasa Narasimhan, and Takeo Kanade, "Analysis of rain and snow in frequency space," *International journal of computer vision*, vol. 86, no. 2-3, pp. 256, 2010.

[16] Z. Li, J. Zhang, Z. Fang, B. Huang, X. Jiang, Y. Gao, and J. Hwang, "Single image snow removal via composition generative adversarial networks," *IEEE Access*, vol. 7, pp. 25016–25025, 2019.

[17] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, vol. 9351 of *LNCS*, pp. 234–241, Springer, (available on arXiv:1505.04597 [cs.CV]).

[18] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 12, pp. 2341–2353, Dec 2011.