# Joint Bi-layer Optimization for Single-image Rain Streak Removal

Lei Zhu[1,3], Chi-Wing Fu[1,3], Dani Lischinski[2], and Pheng-Ann Heng[1,3]
[1]The Chinese University of Hong Kong, [2] The Hebrew University of Jerusalem,
[3] Shenzhen Key Laboratory of Virtual Reality and Human Interaction Technology,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
{lzhu, cwfu, pheng}@cse.cuhk.edu.hk, danix@mail.huji.ac.il

## Abstract

*We present a novel method for removing rain streaks from a single input image by decomposing it into a rain-free background layer $B$ and a rain-streak layer $R$. A joint optimization process is used that alternates between removing rain-streak details from $B$ and removing non-streak details from $R$. The process is assisted by three novel image priors. Observing that rain streaks typically span a narrow range of directions, we first analyze the local gradient statistics in the rain image to identify image regions that are dominated by rain streaks. From these regions, we estimate the dominant rain streak direction and extract a collection of rain-dominated patches. Next, we define two priors on the background layer $B$, one based on a centralized sparse representation and another based on the estimated rain direction. A third prior is defined on the rain-streak layer $R$, based on similarity of patches to the extracted rain patches. Both visual and quantitative comparisons demonstrate that our method outperforms the state-of-the-art.*

## 1. Introduction

Computer vision methods generally assume a clear image as input for processing and scene understanding. However, rain streaks, when present, tend to obstruct and blur the scene [19], thereby distorting the image content and degrading the accuracy of visual analysis [5].

Several methods have been proposed to remove rain streaks in images. They are typically classified by their input type. *Video-based* methods [11, 34, 1, 2, 24, 33] leverage the rich temporal information across frames to locate and remove rain streaks, while *single-image* methods [15, 16, 5, 26, 14, 21, 19] require the use of image priors to recover the underlying background scene, e.g., dictionary-based sparse prior [15, 26, 14, 21], low-rank prior [5], nonlocal self-similarity prior [16], and GMM-based layer prior [19].

**(a)** *Input rain image* **(b)** *Result of Luo et al. [21]*
**(c)** *Result of Li et al. [19]* **(d)** *Our result*

**Figure 1:** *Rain streak removal on a real photo.*

Compared with video-based methods, it is much more challenging to remove rain streaks given only a single image. Although existing single-image methods can improve the overall scene visibility, they tend to either oversmooth the background image details, or retain a significant amount of rain streaks in the results. This is demonstrated in Fig. 1, which shows de-rained results produced using two state-of-the-art methods [21, 19] next to our result.

The goal of single-image rain streak removal is to decompose an input image $\mathbf{I}$ into two layers: a background image $\mathbf{B}$, which is rain-free, and a rain image $\mathbf{R}$, which contains only rain streaks. To achieve this, we develop a *joint bi-layer optimization method* that alternatively processes the two layers, where we alternatively smooth the background layer to push rain streaks from $\mathbf{B}$ to $\mathbf{R}$, and smooth the rain layer to push background details from $\mathbf{R}$ back to $\mathbf{B}$. Then, we can improve the preservation of background details in $\mathbf{B}$, which is the optimization output. Although several priors on $\mathbf{B}$ and $\mathbf{R}$ have been proposed in previous work, they usually have limited capability to differentiate background details from rain streaks, especially for heavy rain conditions. For example, the Gaussian mixture models (GMM) of Li et al. [19],

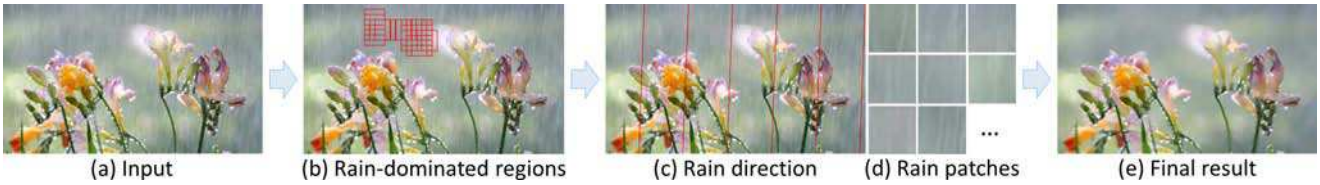


**Figure 2:** *Overview: given a rain image (a), we first locate rain-dominated regions (b), and use them to estimate the rain direction (c) and extract rain patches (d); then, we use our joint bi-layer optimization method to iteratively create the result (e).*

trained from a set of manually-collected rain and non-rain patches, are ineffective on patches with a significant amount of background detail corrupted with many rain streaks.

In summary, this work has two major novel components. First, we introduce an automatic method to locate rain-dominated regions from the input image and estimate the dominant direction of the rain streaks. Observing that rain streaks usually fall within a narrow band of directions, even for heavy rain, we analyze the statistics of gradient vectors over image blocks in $\mathbf{I}$ and identify scene regions dominated by rain streaks, see Fig. 2(b). From such regions, we can then estimate the rain direction in $\mathbf{I}$ and extract rain patches to model the rain pattern, see Fig. 2(c&d). The other major novel component in this work is the joint bi-layer optimization model, which iteratively separates rain ($\mathbf{R}$) and background ($\mathbf{B}$) through the following three priors:

- First, we introduce a centralized sparse representation (CSR) [7] to improve the performance in removing rain streaks, while preserving the background details. This prior integrates both local and nonlocal sparsity constraints and adapts the CSR for de-raining by constructing a guidance image with the window inherent variation metric [30]. Note that previous methods employ only the local sparsity [15, 26, 14, 21] or the nonlocal prior [28].
- Second, we construct a rain direction prior by considering the angular deviation of pixel gradients from the rain direction. This prior is built upon the rain direction information we automatically extracted by detecting rain-dominated scene regions. Previous works like [11] estimate rain directions by tracking how rain streaks move over successive video frames, so they cannot be used for single-image rain streak removal. To the best of our knowledge, ours is the first work to make use of a rain direction prior for single-image rain streak removal.
- Lastly, we introduce a rain layer prior specifically for the rain layer $\mathbf{R}$, designed to smooth out non-rain-streak background details in $\mathbf{R}$ by using the rain patches we automatically extracted from the rain-dominated regions.

Furthermore, we adopt the alternative direction method of multipliers (ADMM) and iteratively re-weighted least squares (IRLS) methods to efficiently solve the resulting optimization problem, and evaluate it using both synthetic and real images. Both quantitative comparisons and visual results demonstrate that our method can efficiently remove rain streaks and better preserve background details, compared to the state-of-the-art single-image methods.

## 2. Related Work

**Video-based rain streak removal methods** leverage the temporal information in videos to remove rain streaks, typically by analyzing the difference between adjacent frames. Garg and Nayar [11, 12, 13] proposed an appearance model based on photometric properties and temporal dynamics to describe rain streaks, while Zhang et al. [34] exploited the temporal and chromatic properties of rain in videos. Other methods include [1, 2, 24]; see [28] for a detailed review.

**Single-image rain streak removal methods** is a very challenging problem, since we observe only one color value per pixel in the input image. To address this problem, various image priors have been explored. Dictionary-based sparse prior [23] describes an image patch as a linear combination of a few atoms from a pre-specified dictionary. Kang et al. [15] decomposed an input image into a low-frequency and a high-frequency component, and then separated rain streaks from the background using sparse-coding-based dictionary learning. Sun et al. [26] proposed an incremental dictionary learning strategy to represent the high-frequency layer and used the structural similarity metric to identify the dictionary atoms associated with rain patterns. Huang et al. [14] introduced a self-learning mechanism to identify a subset of atoms in the dictionary of high-frequency components, which correspond to rain patterns. Luo et al. [21] took a discriminative sparse-coding approach to rain removal by separating sparse codes associated with background patches from codes associated with the patches of the rain image. While these sparse-coding methods help improve the overall visibility, they tend to oversmooth the background image details, or retain excessive rain streaks in the results [19].

Apart from the local sparse prior, Kim et al. [16] exploited the nonlocal self-similarity prior; they detected rain streaks by assuming elliptical shape and vertical orientations of the rain streaks. Observing that a rainy scene contains similar rain streak patterns, Chen at al. [5] used a low-rank structure to model the rain streaks. Most recently, Li et al. [19] proposed using two Gaussian mixture models (GMMs), one for the background and another for the rain streaks layer. How-

ever, the rain streak GMM is trained on rain-only patches in the input image, so its discriminative power is reduced in images full of background details. Fig. 1(c) shows that in such images many rain streaks remain in the result. Most recently, Fu et al. [10] and Yang et al. [32] independently presented two different deep models for single-image rain streak removal. From the results we obtained from Yang et al. [32] and by running the code of Fu et al. [10], we find that these methods generally improve upon previous works, but still retain some rain streaks in the results; see Fig. 8 and Fig. 9. This is possibly due to the synthetic training data that includes limited number of rain streak directions.

This paper presents a new optimization method for single-image rain streak removal with several novel components: automatic estimation of rain streak direction, new regularization terms to model and separate rain streaks from background, and a new appearance model that pushes the non-rain details from the rain layer back to the background. Experimental results confirm the superiority of our method over previous works, both visually and quantitatively.

Some methods [18, 22, 8, 33] remove artifacts caused by raindrops adhering to the lens or to the windshield in front of cameras; however, this is a different problem, since raindrop patterns differ substantially from rain streak patterns [19].

## 3. Our Approach

A rain image $\mathbf{I}$ is often considered as a linear combination of a rain-free background ($\mathbf{B}$) and a rain streak layer ($\mathbf{R}$) [19]:

$$\mathbf{I} = \mathbf{B} + \mathbf{R}\,, \tag{1}$$

To compute the above decomposition, we take an approach that jointly optimizes $\mathbf{B}$ and $\mathbf{R}$, by alternating between reducing the rain streaks in $\mathbf{B}$ and keeping only rain streaks in $\mathbf{R}$. Specifically, we consider the following objective:

$$\min_{\mathbf{B},\mathbf{R}} ||\mathbf{I}-\mathbf{B}-\mathbf{R}||_F^2 + \lambda_1 \mathbf{\Psi}(\mathbf{B}) + \lambda_2 \mathbf{\Phi}(\mathbf{B}) + \lambda_3 \mathbf{\Omega}(\mathbf{R}), \tag{2}$$

where $||.||_F$ is the Frobenius norm; $||\mathbf{I}-\mathbf{B}-\mathbf{R}||_F^2$ is the fidelity term to enforce the decomposition of Eq. (1); the other three regularization terms $\mathbf{\Psi}(\mathbf{B})$, $\mathbf{\Phi}(\mathbf{B})$, and $\mathbf{\Omega}(\mathbf{R})$ are proposed to model our image priors on $\mathbf{B}$ and $\mathbf{R}$ for rain streak removal. In short, $\mathbf{\Psi}(\mathbf{B})$ aims to smooth out rain streaks in $\mathbf{B}$, $\mathbf{\Phi}(\mathbf{B})$ aims to preserve more background details in $\mathbf{B}$, while $\mathbf{\Omega}(\mathbf{R})$ aims to suppress non-rain-streak details in $\mathbf{R}$, thereby pushing them back to $\mathbf{B}$. Their exact formulations are provided in Section 5, which also describes the iterative optimization scheme that we use to solve Eq. (2). Before these details, we first present how we estimate the rain direction, and extract rain patches in Section 4, since the regularization terms are built upon these rain information.

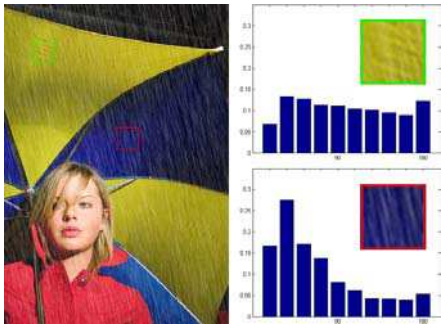

**Figure 3:** *Gradients in rain-dominated regions (red) come mainly from the rain streaks, as compared to regions that contain more background details (green). Thus, the distributions of gradient angles look different in these two cases.*

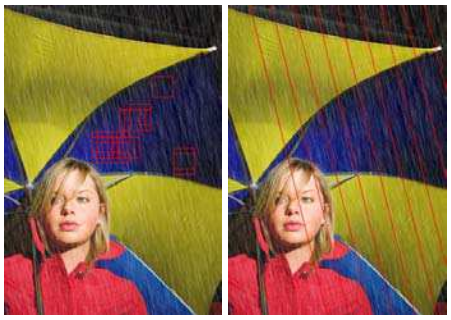

**Figure 4:** *Rain-dominated regions (red boxes, left) and rain direction (red lines, right) estimated by our method.*

## 4. Rain Modeling

In typical rain images, one can observe that the rain streak directions usually fall within a narrow range, while the gradients of the background scene details are not restricted in such a manner. We use this observation to identify rain-dominated image regions, from which we proceed to estimate rain streak directions and extract rain patches in a robust fashion. This is done automatically in the following three steps:

*i) Locate rain-dominated regions in* $\mathbf{I}$. These regions have relatively little background details, such as the red box in Fig. 3. We identify them by examining the local distribution of gradient directions over $\mathbf{I}$. First, we compute the angle of the gradient of each pixel in $\mathbf{I}$ (range: $[0, \pi)$). Then, we use a $W_r \times W_r$ sliding window with a stride of $W_s$ to compute local gradient angle histograms using a histogram with 10 bins. Denoting by $\delta$ the proportion of pixels contained in the most populated bin (as well as its two adjacent bins), we expect rain-dominated regions to have a higher $\delta$ value than ones that also contain a significant amount of background details, see Fig. 3. Thus, we designate the $N$ windows with the highest $\delta$ as the rain-dominated regions; see Fig. 2(b) and Fig. 4 (left) for some example detected windows.

*ii) Estimate rain streak direction.* Next, we use a Canny edge detector to find edges in each of the rain-dominated windows, and employ the Hough transform to detect the

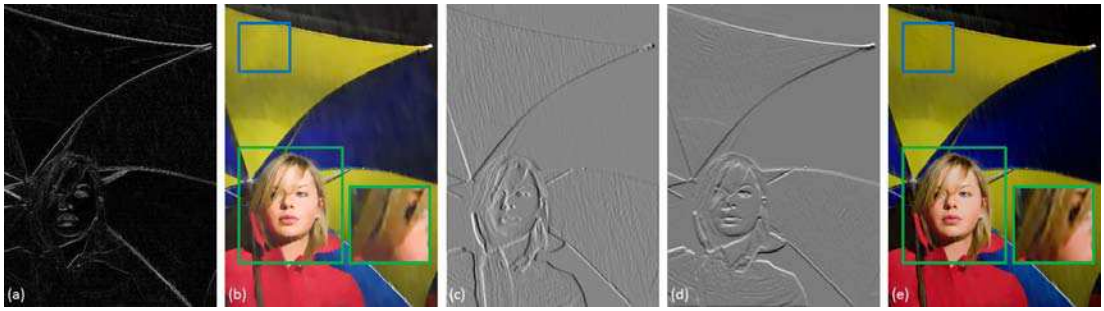

**Figure 5:** *(a) rain-direction map computed by Eq. (5); (b) a result produced by using $\mathbf{\Psi}(\mathbf{B})$ and $\mathbf{\Phi}(\mathbf{B})$; (c&d) weights $w_x$ and $w_y$ computed by Eq. (9); and (e) a result produced by using all three regularization terms: $\mathbf{\Psi}(\mathbf{B})$, $\mathbf{\Phi}(\mathbf{B})$, and $\mathbf{\Omega}(\mathbf{R})$.*

longest line in each window. The median among the slopes of these lines is chosen as the slope of the dominant rain streak direction. See Fig. 4 (right) for a result.

*iii) Extract rain patches.* Lastly, we extract rain patches (or rain-dominated patches) by randomly picking $10N$ patches inside the rain-dominated windows. The size of the rain patches is set to be $7 \times 7$ in our implementation.

## 5. Joint Bi-layer Optimization

Next, we employ the estimated rain information to formulate our regularization terms in Eq. (2): $\Psi(\mathbf{B})$, $\Phi(\mathbf{B})$, and $\Omega(\mathbf{R})$.

### 5.1. Sparsity prior $\mathbf{\Psi}(\mathbf{B})$

While a dictionary-based sparse prior has been previously used for rain streak removal [23, 21], Dong et al. [7] show that the centralized sparse representation (CSR) is more effective for image restoration. Thus, our first term $\mathbf{\Psi}(\mathbf{B})$ is based on CSR, which combines local and non-local sparsity:

$$\mathbf{\Psi}(\mathbf{B}) = ||\boldsymbol{\alpha}||_1 + \gamma \sum_{i \in \mathbf{B}} ||\alpha_i - \mu_i||_1, \text{ s.t. } \mathbf{B} = \mathbf{D} \circ \boldsymbol{\alpha}\,, \quad (3)$$

where $\alpha_i$ is the sparse code of the patch centered at pixel $i$ in $\mathbf{B}$ (note that $\mathbf{B}$ is initialized as $\mathbf{I}$ before the first iteration); $\boldsymbol{\alpha}$ is a long vector concatenating all $\alpha_i$; $\gamma$ is a weight; $\mathbf{D}$ is the dictionary created using [7]; $\mathbf{D} \circ \boldsymbol{\alpha}$ is the reconstructed background image; and $\mu_i$ is the weighted average of the sparse codes of the $M$ nonlocal patches that are the most similar to the patch centered at pixel $i$:

$$\mu_i = \frac{1}{\Upsilon} \sum_{m=1}^{M} \tau_{i,m} \alpha_{i,m}, \quad (4)$$

where $\alpha_{i,m}$ is the sparse code of the $m$-th similar patch; $\tau_{i,m}$ is the distance between that patch and the patch centered at pixel $i$; and $\Upsilon$ is the sum of $\tau_{i,m}$ for normalization.

However, finding similar patches in rain images is made difficult by the presence of rain streaks. Regarding the rain streaks as a texture superimposed over the background image, we employ the window inherent variation metric [30], which was designed to separate structure from texture in images. This metric is used to create a guidance image $\mathbf{G}$, which then assists in the estimation of patch similarity, see Fig. 6 (left) for an example. Given $\mathbf{G}$, the similarity $\tau_{i,j}$ of two patches centered at pixels $i$ and $j$ is estimated by multiplying the $L_2$ distance between the two patches in $\mathbf{B}$ with the $L_2$ distance between the two patches in $\mathbf{G}$. See Fig. 6 (right) for a result of using $\mathbf{\Psi}(\mathbf{B})$ alone to remove rain streaks.

**Figure 6:** *Guidance image computed by using [30] (left) and a result produced by using $\mathbf{\Psi}(\mathbf{B})$ alone (right).*

### 5.2. Rain direction prior $\mathbf{\Phi}(\mathbf{B})$

As seen in Fig. 6 (right), $\mathbf{\Psi}(\mathbf{B})$ can effectively suppress the rain streaks, but it also removes many background details, e.g., the hair details (see green box). Having estimated the global rain direction $\vec{d}$, we assume that gradients in $\mathbf{B}$, which are not perpendicular to $\vec{d}$, are unlikely to belong to rain streaks, and thus should not be removed. Hence, we introduce $\mathbf{\Phi}(\mathbf{B})$ to penalize the gradients perpendicular to $\vec{d}$. Specifically, we first construct a rain-direction map by comparing rain direction $\vec{d}$ and gradient $\vec{g}_i$ per pixel $i$ in $\mathbf{B}$:

$$\mathbf{\Theta}_i(\mathbf{B}) = \frac{|\vec{d} \cdot \vec{g}_i|}{||\vec{g}_i|| + \varepsilon_1}\,, \quad (5)$$

where $\varepsilon_1$ is a constant (set to $0.0001$) to avoid division by zero. See Fig. 5(a) for an example. Then, we define

$$\mathbf{\Phi}(\mathbf{B}) = \sum_{i \in \mathbf{B}} \frac{1}{\mathbf{\Theta}_i(\mathbf{B}) + \varepsilon_1}\,. \quad (6)$$


**Figure 7:** *The effect of each regularization term in our method. The PSNR values shown below the resulting images reveal the progressive improvement when we put the terms together for rain streak removal.*

The gradients of most rain-streak pixels are roughly perpendicular to $\vec{d}$. Based on Eq. (6), their small $\mathbf{\Theta}_i$ will contribute more to $\mathbf{\Phi}(\mathbf{B})$ than non-perpendicular gradients. Thus, having $\mathbf{\Phi}(\mathbf{B})$ in the objective in Eq. (2) helps reduce over-smoothing of non-rain-streak details in $\mathbf{B}$, as demonstrated in Fig. 5(b): comparing the same green box in Fig. 5(b) and Fig. 6 (right), we can see that $\mathbf{\Psi}(\mathbf{B})$ and $\mathbf{\Phi}(\mathbf{B})$ together can improve the preservation of background details in result.

### 5.3. Rain layer prior $\mathbf{\Omega}(\mathbf{R})$

Regularizing only $\mathbf{B}$ may still smooth out certain background details, e.g., compare the details in the green box in Fig. 3 against the blue box in Fig. 5(b). Hence, we propose to also regularize the rain layer $\mathbf{R}$ to push scene details, which have been erroneously put into $\mathbf{R}$, back to $\mathbf{B}$.

Inspired by [9], we use a weighted Laplacian term to formulate $\mathbf{\Omega}(\mathbf{R})$ with spatially-varying smoothing capability:

$$\mathbf{\Omega}(\mathbf{R}) = \sum_{i \in \mathbf{R}} \left\{ w_x(i) \left(\partial_x \mathbf{R}_i\right)^2 + w_y(i) \left(\partial_y \mathbf{R}_i\right)^2 \right\} , \quad (7)$$

where $w_x(i)$ and $w_y(i)$ are the smoothing weights on pixel $i$ in $\mathbf{R}$ (note: $\mathbf{R}$ is initialized as $\mathbf{I} - \mathbf{B}$). By assigning larger weights to pixels that are believed to contain background detail rather than rain streaks, the corresponding areas of $\mathbf{R}$ will be smoothed more aggressively, as we minimize Eq. (2). Note also that the smoothing weights are determined using a similarity map $\mathbf{\Gamma}_i(\mathbf{I})$, which measures the similarity between each patch in $\mathbf{I}$ and the rain-dominant patches (extracted as described in Section 4):

$$\mathbf{\Gamma}_i(\mathbf{I}) = \min_r \|P_i - \tilde{P}_r\|_2^2 , \quad (8)$$

where $\tilde{P}_r$ is the $r$-th extracted rain patch, and $P_i$ is the patch centered at pixel $i$ in $\mathbf{I}$; note that we use $\mathbf{I}$ here, since the rain patches are also extracted from $\mathbf{I}$. Using $\mathbf{\Gamma}_i(\mathbf{I})$, we define

$$w_x(i) = |\partial_x \mathbf{\Gamma}_i(\mathbf{I})|^\eta \text{ and } w_y(i) = |\partial_y \mathbf{\Gamma}_i(\mathbf{I})|^\eta , \quad (9)$$

where $\eta$ is a sensitivity parameter. Fig. 5(c&d) shows an example of $w_x$ and $w_y$, while Fig. 5(e) shows a result produced with $\mathbf{\Psi}(\mathbf{B})$, $\mathbf{\Phi}(\mathbf{B})$, and $\mathbf{\Omega}(\mathbf{R})$. Comparing Fig. 5(b) and (e), we can see that more background details are pushed back to $\mathbf{B}$ by constraining $\mathbf{R}$ to look like the extracted rain patches.

Each prior in the joint bi-layer optimization has its own contribution to the success of the method. First, we define $\mathbf{\Psi}(\mathbf{B})$ as the base model to separate rain streaks from background using both local and nonlocal sparsity. Then, we define $\mathbf{\Phi}(\mathbf{B})$ to improve the background details preservation by taking in rain semantics, i.e., rain streaks in images usually following a certain direction. Next, we find that only regularizing layer $\mathbf{B}$ tends to unavoidably oversmooth or remove some background details, when eliminating the rain streaks during the optimization procedure on $B$; hence, we define $\mathbf{\Omega}(\mathbf{R})$ to regularize and smooth out details on the layer $\mathbf{R}$ using rain patches extracted in rain-dominated regions. In this way, we can further push those non-rain details back to layer $\mathbf{B}$. Fig. 7 demonstrates the effectiveness of the priors by comparing results produced with $\mathbf{\Psi}(\mathbf{B})$, $\mathbf{\Psi}(\mathbf{B})$+$\mathbf{\Phi}(\mathbf{B})$, $\mathbf{\Psi}(\mathbf{B})$+$\mathbf{\Omega}(\mathbf{B})$, and $\mathbf{\Psi}(\mathbf{B})$+$\mathbf{\Phi}(\mathbf{B})$+$\mathbf{\Omega}(\mathbf{R})$. As can be seen, using only $\mathbf{\Psi}(\mathbf{B})$ tends to overblur the details; adding $\mathbf{\Phi}(\mathbf{B})$ helps to alleviate the blurring, while adding $\mathbf{\Omega}(\mathbf{B})$ helps to preserve more background details. Lastly, by putting all three priors together, we can achieve the best results, as revealed by the PSNR values shown at the bottom in Fig. 7.

### 5.4. Solving the Joint Optimization

Next, we describe how we iteratively minimize Eq. (2) with $\mathbf{\Psi}(\mathbf{B})$, $\mathbf{\Phi}(\mathbf{B})$ and $\mathbf{\Omega}(\mathbf{R})$. First, we initialize $\mathbf{B}_0$ as $\mathbf{I}$ and $\mathbf{R}_0$ as $\mathbf{I} - \mathbf{B}_0$ (zero image). Then, we iteratively perform Steps 1 and 2 below $K$ times to compute $\mathbf{B}_k$ and $\mathbf{R}_k$ ($k \in [1, K]$):

**Step 1: Update B.** We compute $\mathbf{B}_k$ in the $k$-th iteration by solving the following augmented Lagrangian function:

$$\mathbf{B}_k = \min_{\mathbf{B},\boldsymbol{\alpha}} ||\mathbf{I} - \mathbf{B} - \mathbf{R}_{k-1}||_F^2 + \lambda_1 \mathbf{\Psi}(\mathbf{B}) + \lambda_2 \mathbf{\Phi}(\mathbf{B}) - \langle \mathbf{H}, \mathbf{B} - \mathbf{D} \circ \boldsymbol{\alpha} \rangle + \frac{\beta}{2} ||\mathbf{B} - \mathbf{D} \circ \boldsymbol{\alpha}||_F^2 , \quad (10)$$

where $\mathbf{H}$ is the Lagrange multiplier of linear constraint and

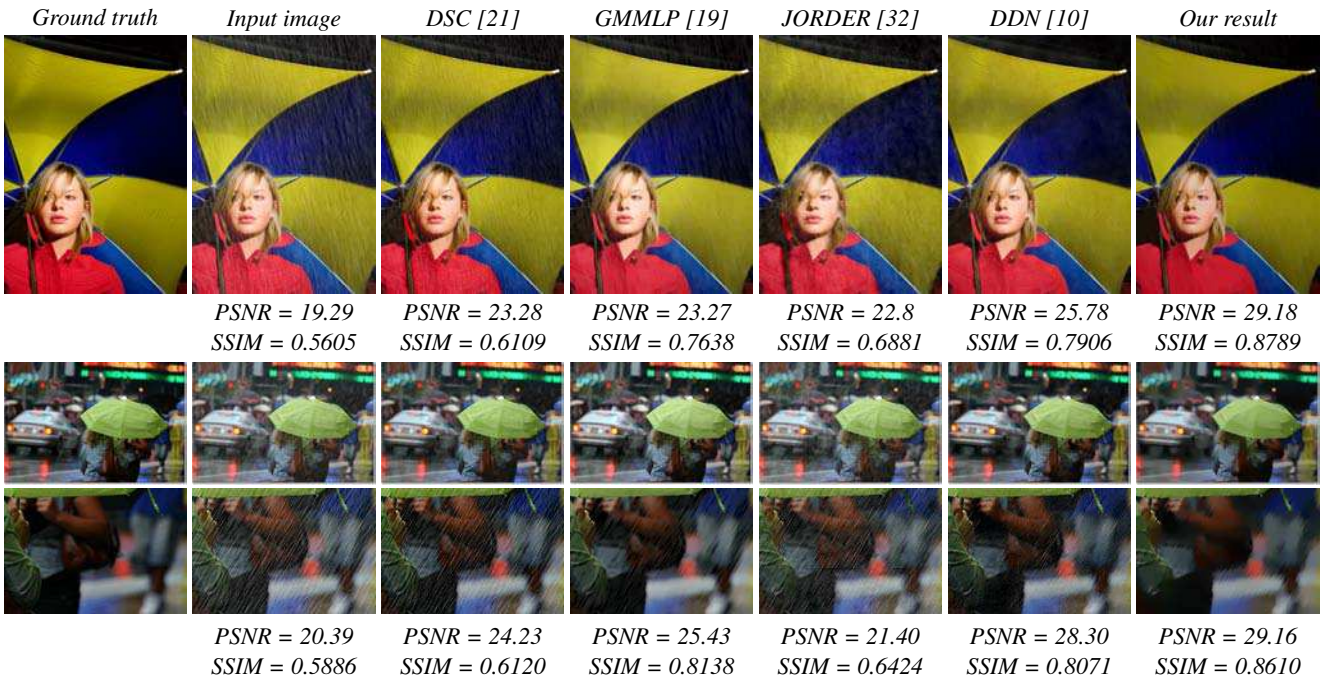


**Figure 8:** *Comparing results produced from our method against state-of-the-art methods on two widely-used images.*

$\beta$ is the penalty parameter. However, it is difficult to simultaneously optimize $\boldsymbol{\alpha}$ in dictionary domain and $\boldsymbol{\Phi}(\mathbf{B})$ in image domain in Eq. (10). Hence, we efficiently solve it by adopting the alternating direction method of multipliers (ADMM) technique [3] by alternatively updating $\mathbf{B}$ and $\boldsymbol{\alpha}$ in the following two subproblems (with $T$ iterations):

**Subproblem 1.1 on B:** By removing $\lambda_1\boldsymbol{\Psi}(\mathbf{B})$ and adding $\frac{\beta}{2}||\beta^{-1}\mathbf{H}^t||_F^2$ to Eq. (10), we can estimate $\mathbf{B}^{t+1}$ as:

$$\min_{\mathbf{B}} ||\mathbf{I}-\mathbf{B}-\mathbf{R}_{k-1}||_F^2+\lambda_2\Phi(\mathbf{B})+\frac{\beta}{2}||\mathbf{B}-\mathbf{D}\circ\boldsymbol{\alpha}^t-\frac{1}{\beta}\mathbf{H}^t||_F^2 . \tag{11}$$

Now, the optimization is quadratic but has a non-linear term $\boldsymbol{\Phi}(\mathbf{B})$. Since a quadratic optimization with a quadratic regularization term can be optimized linearly [27, 20, 17], we solve the problem with the iterative re-weighted least squares (IRLS) technique [30]. The main idea here is to decompose the non-linear $\boldsymbol{\Phi}(\mathbf{B})$ into a quadratic term and another non-linear term (say $\mathbf{S}_{x,i}$), which enable us to solve the optimization in Eq. 11 by computing the new term $\mathbf{S}_{x,i}$ and estimating $\Phi(B)$ with a closed-form solution alternatively; please see the supplementary material for the details.

**Subproblem 1.2 on $\boldsymbol{\alpha}$:** With the estimated $\mathbf{B}^{t+1}$ obtained from subproblem 1.1, we compute $\boldsymbol{\alpha}^{t+1}$ by solving

$$\min_{\boldsymbol{\alpha}} \frac{\beta}{2}||\mathbf{B}^{t+1}-\mathbf{D}\circ\boldsymbol{\alpha}^t-\frac{1}{\beta}\mathbf{H}^t||_F^2+\lambda_1\boldsymbol{\Psi}(\mathbf{B}^{t+1}) . \tag{12}$$

Note that we adopt a similar minimization in [7] to solve the above centralized sparse optimization, and update $\mathbf{H}^{t+1}$ as $\mathbf{H}^t+\beta(\mathbf{B}^{t+1}-\mathbf{D}\circ\boldsymbol{\alpha}^{t+1})$ to complete Step 1.

**Step 2: Update R.** Given $\mathbf{B}_k$ from Step 1, we solve for $\mathbf{R}_k$ by the following sparse linear equation:

$$\left(\boldsymbol{\Gamma}+\lambda_3(\mathbf{E}_x^T V(\mathbf{w}_x)\mathbf{E}_x+\mathbf{E}_y^T V(\mathbf{w}_y)\mathbf{E}_y)\right)V(\mathbf{R})=V(\mathbf{I})-V(\mathbf{B}_k). \tag{13}$$

After updating $\mathbf{R}_k$, we do not need to explicitly push the non-rain details back to $\mathbf{B}$, since this will be done when updating $\mathbf{B}_{k+1}$ in the next iteration.

**Timing.** To save computation time, we follow [19] and run the de-raining only on the luminance (Y) channel by converting $\mathbf{I}$ to YUV space. Moreover, we construct the dictionary only once with the input rain image. Our current Matlab implementation takes about 24 sec. (3 iterations) / 60 sec. (8 iterations) to process a $480\times320$ color image; this compares favorably to the published state-of-the-art (i.e., [19]).

**Parameters.** Our method, like other rain streak removal works, also has parameters, but most parameters are fixed in the experiments: $\lambda_1$ = 1.0 (Eq. 2), $\lambda_3$ = 0.01 (Eq. 2), $W_r$ = 31 (Sec. 4), $W_s$ = 8 (Sec. 4), $N$ = 20 (Sec. 4), $\gamma$ = 5 (Eq. 3), $M$=20 (Eq. 4), $\eta$ = 1.2 (Eq. 9), $\beta$ = 0.01 (Eq. 10), and $T$ (the ADMM iteration number) = 2. For the canny operator in Sec. 4, we empirically employ the default values provided by the MATLAB canny function. To work with different input images, we only tune two parameters: i) $\lambda_2$ in Eq. (2), and ii) iteration number $K$ in our solver (see Sec. 5.4). Empirically, $\lambda_2$ ranges $[10^{-4}, 10^{-3}]$, while $K$ is set as 4 to 50. For heavy rain condition, we use larger values for $\lambda_2$ and for $K$.

**(a)** *Input image* **(b)** *DSC [21]* **(c)** *GMMLP [19]* **(d)** *JORDER [32]* **(e)** *DDN [10]* **(f)** *Ours*

**Figure 9:** *Comparisons with state-of-the-art methods using various real images, particularly with heavy rain.*

**Table 1:** *Quantitative comparison using two synthetic datasets D1 and D2; note: images in D2 have dense rain.*

| | PSNR (D1) | PSNR (D2) | SSIM (D1) | SSIM (D2) |
|---|---|---|---|---|
| DSC [21] | 27.29 | 24.46 | 0.8215 | 0.7853 |
| GMMLP [19] | 29.36 | 26.23 | 0.8430 | 0.8038 |
| JORDER [32] | 29.90 | 27.36 | 0.8664 | 0.8342 |
| DDN [10] | 28.78 | 25.93 | 0.8408 | 0.8007 |
| Our method | **31.05** | **27.64** | **0.8917** | **0.8567** |

## 6. Results

We compare our method against state-of-the-art single-image de-raining methods, including discriminative sparse coding [21] (denoted as DSC), GMM-based layer prior [19] (denoted as GMMLP), joint rain detection and removal [32] (denoted as JORDER), and removing rain from single images via a deep detail network [10] (denoted as DDN). Many synthetic and real images have been tested, and two measures were used in the evaluation: peak signal to noise ratio (PSNR) and structural similarity index (SSIM) [29].

**Synthetic Images.** We conduct quantitative comparisons using two widely-used synthetic rain images, see Fig. 8. Note that for GMMLP, the results are obtained directly from the published paper [19]; for DSC and DDN [10], we produce the results using the authors' code and tuning its parameters to achieve the best results; while for JORDER [32], we obtain the results directly from the authors. In these results, DSC, GMMLP, JORDER, and DDN tend to retain excessive rain streaks. In contrast, our method is capable of removing rain streaks, while better preserving the background details. The PSNR and SSIM values reported in Fig. 8 provide a quantitative indication of the performance of our method.

Also, we prepare quantitative comparisons between our method and others on two datasets. One is a synthetic dataset (denoted as D1) by randomly selecting 60 images from the BSDS 500 dataset [6], synthesizing rain over the images with different rain streak orientations (with the same procedure using Photoshop as in [10]), and applying different methods to remove rain in the images. Another dataset (denoted as D2) is built intentionally with dense rain by randomly selecting 30 other images from the BSDS 500 dataset [6]. Table 1 reports the resulting averaged PSNR and SSIM values for comparison on D1 and D2, demonstrating that our method achieves the best (highest) PSNR and SSIM values.

**Real Images.** We also verify our method and compare it against others using a number of real rain images, e.g., see Fig. 9. Again, DSC, JORDER and DDN tend to retain excessive rain streaks, while GMMLP tends to keep more rain streaks for images with heavier rain (2nd row in Fig. 9) or oversmooth some background details for lighter rain (4th row in Fig. 9). In contrast, our method achieves better results in terms of effectively removing the rain streaks while preserving the scene details, e.g., see the flowers on the ground in the blown-up views in the last row of Fig. 9, with rain streaks mostly removed. Lastly, Fig. 10 demonstrates the


**Figure 10:** *More results produced by our method under different rain directions. Top row: inputs. Bottom row: our results.*

**Table 2:** *Improvement in on-road vehicle detection.*

| | ODS | OIS | AP |
|---|---|---|---|
| Rain images | 0.617 | 0.632 | 0.675 |
| Using our results | 0.741 | 0.765 | 0.806 |

effectiveness of our method on real images with rain streaks in different directions and over different backgrounds. *Additional results can be found in the supplementary material.*

**Applications.** We conducted preliminary experiments on three applications to demonstrate how our method benefits other computer vision algorithms. The first application is edge detection using a recent edge detector [6]. First, we randomly selected 30 images from the BSDS 500 dataset [6], added rain over the images using the procedure as in [10], and employed our method to remove rain in the images. Then, we applied [6] to generate edge maps from the rained images, as well as from our de-rained images, and compared the results with the ground truths given in the BSDS 500 dataset [6]. To quantitatively evaluate the edge detection accuracy, we followed [6] to use the fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP) metrics. From Table 2, we can see that the edge detection accuracy improves after removing the rain streaks.

Second, we explored saliency detection with and without rain using [31]. Here, we randomly selected 30 images from the CSSD dataset [31], added rain to each of them, and then applied our method to remove the rain. Then, we compare the saliency detection accuracy using the AUC-Judd metric[1]. Results show that the AUC-Judd value increases from 0.9012 to 0.9422 after we remove the rain streaks.

The last application is on-road vehicle detection in rain images using a state-of-the-art detector [4]. Here, we randomly selected 30 images from the KITTI dataset [4] and synthesized rain on them. We then removed the rain using our method. Results show that the average precision in terms of detecting vehicles improves from $\sim$59.76% to $\sim$65.71% after using our method to remove the rain in those images.

**Limitations.** First, some background details may still be oversmoothed by our method, as shown in Fig. 8. Second, two parameters of our method are required to be specified by users for optimal performance. Lastly, our method does not consider haze in the rain although dehazing and contrast enhancement may certainly be applied as a separate process.

**Acknowledgments.** We thank reviewers for the valuable comments. This work was supported in part by the 973 Program (2015CB351706), the Shenzhen Science and Technology Program (JCYJ20170413162617606 and JCYJ20170413162256793), the National Natural Science Foundation of China (61233012), the Israel Science Foundation (ISF grant No. 2366/16), and the ISF-NSFC joint research program (grant No. 2217/15).

## 7. Conclusion

We present a joint bi-layer optimization method for single-image rain streak removal with the following novel components. First, we automatically locate rain-dominated regions in the input image, and then estimate the rain direction and extract rain patches by analyzing the gradient statistics. With this information, we model three regularization terms to progressively separate rain streaks from background details in various aspects: integrating local and nonlocal sparsity via a centralized sparse representation, measuring deviation of gradients from the estimated rain direction, and measuring the visual similarity between image patches and rain patches to filter the rain layer. Comparing with the state-of-the-art methods, we do not need to manually prepare rain and non-rain inputs to train a data model, and our method can better separate background details from rain streaks as shown in the various results using real and synthetic rain images.

[1] http://saliency.mit.edu/results_mit300.html

## References


[1] P. C. Barnum, S. Narasimhan, and T. Kanade. Analysis of rain and snow in frequency space. *IJCV*, 86(2-3):256–274, 2010.

[2] J. Bossu, N. Hautière, and J.-P. Tarel. Rain or snow detection in image sequences through use of a histogram of orientation of streaks. *IJCV*, 93(3):348–367, 2011.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[4] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016.

[5] Y.-L. Chen and C.-T. Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In *ICCV*, 2013.

[6] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2015.

[7] W. Dong, L. Zhang, and G. Shi. Centralized sparse representation for image restoration. In *ICCV*, 2011.

[8] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013.

[9] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):67:1–67:10, 2008.

[10] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley. Removing rain from single images via a deep detail network. *CVPR*, 2017.

[11] K. Garg and S. K. Nayar. Detection and removal of rain from videos. In *CVPR*, 2004.

[12] K. Garg and S. K. Nayar. When does a camera see rain? In *ICCV*, 2005.

[13] K. Garg and S. K. Nayar. Vision and rain. *IJCV*, 75(1):3–27, 2007.

[14] 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*, 16(1):83–93, 2014.

[15] L.-W. Kang, C.-W. Lin, and Y.-H. Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE TIP*, 21(4):1742–1755, 2012.

[16] J.-H. Kim, C. Lee, J.-Y. Sim, and C.-S. Kim. Single-image deraining using an adaptive nonlocal means filter. In *2013 IEEE International Conference on Image Processing*, pages 914–917. IEEE, 2013.

[17] D. Krishnan and R. Szeliski. Multigrid and multilevel preconditioners for computational photography. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 30(6):177, 2011.

[18] H. Kurihata, T. Takahashi, I. Ide, Y. Mekada, H. Murase, Y. Tamatsu, and T. Miyahara. Rainy weather recognition from in-vehicle camera images for driver assistance. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 205–210. IEEE, 2005.

[19] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown. Rain streak removal using layer priors. In *CVPR*, 2016.

[20] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local adjustment of tonal values. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):646–653, 2006.

[21] Y. Luo, Y. Xu, and H. Ji. Removing rain from a single image via discriminative sparse coding. In *ICCV*, 2015.

[22] M. Roser, J. Kurz, and A. Geiger. Realistic modeling of water droplets for monocular adherent raindrop recognition using bézier curves. In *ACCV*, 2010.

[23] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[24] V. Santhaseelan and V. K. Asari. Utilizing local phase information to remove rain from video. *IJCV*, 112(1):71–89, 2015.

[25] G. Schaefer and M. Stich. Ucid: an uncompressed color image database. In *Electronic Imaging 2004*, pages 472–480. International Society for Optics and Photonics, 2003.

[26] S.-H. Sun, S.-P. Fan, and Y.-C. F. Wang. Exploiting image structural similarity for single image rain removal. In *IEEE International Conference on Image Processing*, pages 4482–4486. IEEE, 2014.

[27] R. Szeliski. Locally adapted hierarchical basis preconditioning. In *ACM Transactions on Graphics (SIGGRAPH)*, volume 25, pages 1135–1143. ACM, 2006.

[28] A. K. Tripathi and S. Mukhopadhyay. Removal of rain from videos: a review. *Signal, Image and Video Processing*, 8(8):1421–1430, 2014.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004.

[30] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. *ACM Trans. Graph. (SIGGRAPH Asia)*, 31(6):139:1–139:10, 2012.

[31] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013.

[32] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan. Joint rain detection and removal from a single image. *CVPR*, 2017.

[33] S. You, R. Tan, R. Kawakami, Y. Mukaigawa, and K. Ikeuchi. Adherent raindrop modeling, detection and removal in video. *IEEE PAMI*, 2015.

[34] X. Zhang, H. Li, Y. Qi, W. K. Leow, and T. K. Ng. Rain removal in video by combining temporal and chromatic properties. In *2006 IEEE International Conference on Multimedia and Expo*, pages 461–464. IEEE, 2006.