Applied Mathematical Modelling 79 (2020) 896–913

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Video deraining via nonlocal low-rank regularization

Yugang Wang[a], Ting-Zhu Huang[a,*], Xi-Le Zhao[a,*], Tai-Xiang Jiang[b]

[a] School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China
[b] FinTech Innovation Center, Financial Intelligence and Financial Engineering Research Key Laboratory of Sichuan province, School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu, Sichuan, China

ARTICLE INFO

Article history:
Received 16 April 2019
Revised 21 October 2019
Accepted 30 October 2019
Available online 9 November 2019

Keywords:
Video rain streaks removal
Nonlocal self-similarity
Unidirectional total variation
Alternating direction method of multipliers

ABSTRACT

Outdoor videos captured in rainy weather may be significantly corrupted by the undesired rain streaks, which severely affect the video processing tasks in outdoor computer vision systems. In this paper, we propose a tensor-based video rain streaks removal method using the nonlocal low-rank regularization. Specifically, we first divide videos into overlapped spatial–temporal patches. Then for each patch, we group its nonlocal similar spatial–temporal patches to form a third-order tensor. To model the clean videos, we characterize the wealth redundancy by adopting the tensor nuclear norm to regularize the low-rankness of the third-order tensors formed by similar spatial–temporal patches of clean videos. We also consider the piecewise smoothness and the temporal continuity of clean videos and utilize the unidirectional total variation to enhance the smoothness and continuity. Moreover, as rain streaks are sparse and smooth along the rain direction, we model the rain streaks by employing an ℓ_1 norm and the unidirectional total variation penalty to boost the sparsity and directional smoothness, respectively. We develop an efficient alternating direction method of multipliers to solve the proposed model. Experimental results on both synthetic and real rainy videos show that our method outperforms the state-of-the-art methods quantitatively and qualitatively.

1. Introduction

Outdoor videos taken under raining conditions are often corrupted by the rain streaks which have the bright intensity and also interfere with the neighbor pixels. These factors will cause several types of visibility degradation in the captured videos, like obstructed objects, distorted colors, and blurred scenes. Such degradation may significantly affect the performance of various tasks in video processing and outdoor vision systems, such as feature extraction, classification [1], object tracking [2,3], recognition [4], and incident detection [5]. Therefore, rain streaks removal is essential for many computer vision applications and has drawn much attention in recent research [6–11].

The key issue for rain streaks removal is to explore the intrinsic properties for both the rain streaks and clean videos and utilize their discriminative structures to separate the rain-free videos. Many existing methods design their algorithms along this line and extract the prior knowledge for rain streaks and clean videos from different aspects, such as the chromatic property of rain streaks [12], the spatiotemporal property of rain and non-rain pixels [13], the similarity and repeatability of rain patterns, [14], and the low-rankness of clean videos [15]. The current state-of-the-art deraining method [16]

* Corresponding author.
E-mail addresses: tzhuang@uestc.edu.cn (T.-Z. Huang), xlzhao122003@163.com (X.-L. Zhao).

https://doi.org/10.1016/j.apm.2019.10.065
0307-904X/

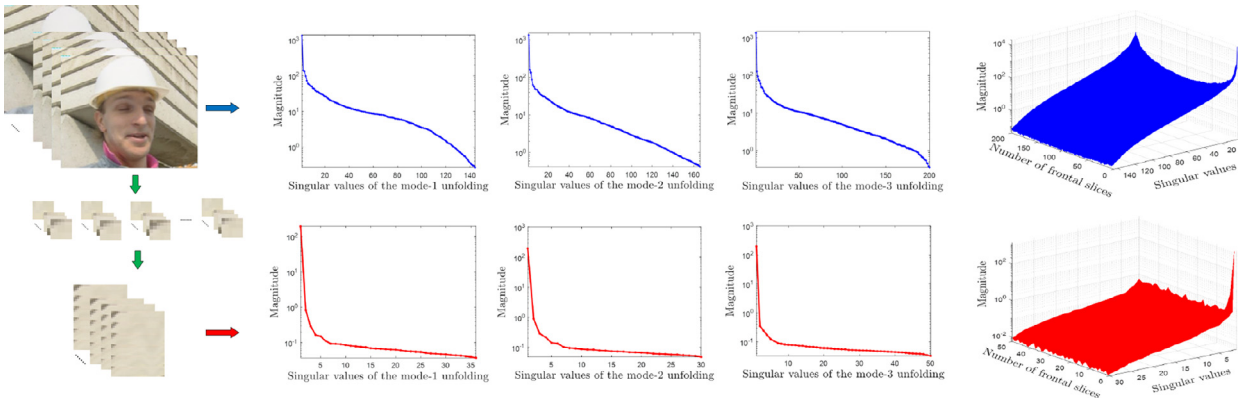

Fig. 1. Comparing the low-rankness of a clean video with dynamic scenes and its similar-patch-formed tensor in the top and bottom rows, respectively. The first column displays the original video, the nonlocal similar patches, and the tensor formed by a nonlocal similar patch group. From the second to fourth column displays: the singular values of the mode-n ($n = 1, 2, 3$) unfolding matrix of the corresponding tensors, respectively. The last column is obtained by first applying discrete Fourier transformation to the tensors along their third modes, then computing the singular values of each frontal slices (see Section 2 for the details).

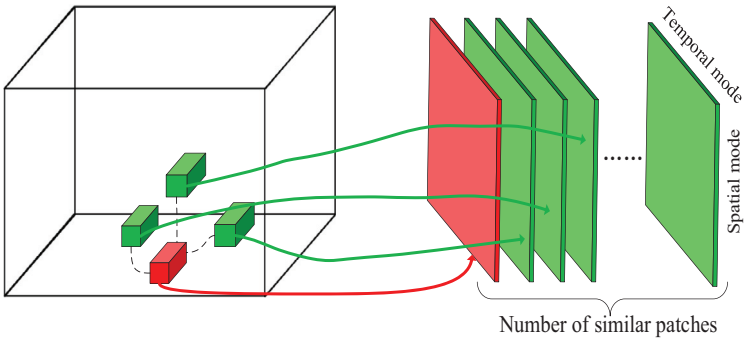

Fig. 2. Illustration for the nonlocal similar spatial–temporal patches in a video. For a red key patch, many green nonlocal similar patches can be found in this video. These similar patches can form a third-order tensor with the spatial mode, temporal mode, and similar-patch-number mode. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

characterized the spatial-temporal correlation of clean videos by assuming the whole clean videos to be low-rank tensors. However, when video frames contain moving objects and highly dynamic scenes, the global low-rankness assumption of clean videos cannot be satisfied (see the first row of Fig. 1). So the detail information of videos may not be well preserved.

To better model the clean videos, in this paper, we adopt the nonlocal self-similarity (NSS) [17] to characterize the spatial-temporal redundancy of clean videos. The NSS prior refers to the fact that for a spatial-temporal local patch, which is the third-order subcube of the video tensor (Fig. 2), many similar spatial–temporal patches can be found across this whole video. To rationally exploit the NSS property, we first divide videos into overlapped spatial–temporal patches, and then for each key patch, we group its nonlocal similar spatial–temporal patches to form a third-order tensor. Such tensors built within clean videos reflect both the NSS and the spatial–temporal correlations (see Fig. 2) and always have low-rank structures (see Fig. 1). To characterize the low-rank property, the sum of nuclear norm (SNN) [18] is usually leveraged for regularizing the low-rankness of tensors. However, unfolding a tensor along one mode will often break the intrinsic structures of tensors. Therefore, we utilize the tensor nuclear norm (TNN) [19] (see Section 2) to enhance the low-rankness so that the structure of tensors can be well preserved. To fully explore the intrinsic priors, we also consider the spatial and temporal continuity of clean videos and use the unidirectional total variation (UTV) regularizers to respectively enhance spatial and temporal smoothness. Moreover, since rain streaks are sparse and smooth along the rain direction, the l_1 norm and the UTV penalty are employed to characterize the sparsity and directional smoothness, respectively. We perform the proposed model on each similar-patch-formed tensor and then aggregate the results to reconstruct the final rain-free video.

In this paper, the proposed method has three main contributions which are summarized as follows:

- We consider the NSS prior for video rain streaks removal, which fully exploits the redundancy of nonlocal similar patches. To the best of our knowledge, this is the first attempt to use NSS prior to remove rain streaks from videos.
- To characterize the redundancy of nonlocal similar patches, we propose to use the TNN, which can efficiently preserve the intrinsic structures, to regularize the similar-patch-formed tensors with their low-rankness, and build a convex model for videos rain streaks removal.

- To tackle the proposed model, we develop an efficient alternating direction method of multipliers (ADMM). The convergence of the proposed algorithm can be theoretically guaranteed under ADMM framework. The extensive experiments on both synthetic and real rainy videos demonstrate that our method performs superiorly over the compared state-of-the-art methods qualitatively and quantitatively.

This paper is organized as follows. The basic notations are introduced in Section 2. In Section 3, some related works are briefly discussed. Our method is proposed in Section 4, Section 5 presents the numerical algorithm. The experimental results are shown in Section 6. At last, the conclusion is given in Section 7.

2. Preliminaries

In this paper, tensors are denoted by bold calligraphic letters, e.g., $\mathcal{X}$. Matrices are denoted by uppercase letters, e.g., X. Vectors are denoted by bold lowercase letters, e.g., $\boldsymbol{x}$, and scalars are denoted by lowercase letters, e.g., x.

The squared Frobenius norm of a third-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as $\|\mathcal{X}\|_F^2 = \sum_{i=1}^{n_1} \sum_{j=2}^{n_2} \sum_{k=3}^{n_3} x_{ijk}^2$, where x_{ijk} is the (i, j, k)-th component of $\mathcal{X}$.

The *mode-k unfolding* of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_l}$ can be denoted as a matrix $X_{(k)} \in \mathbb{R}^{k \times \prod_{i \neq k} n_i}$, where the $(i_1, i_2, \ldots, i_l)$-th entry of $\mathcal{X}$ corresponds to the (i_k, j)th entry of $X_{(k)}$ with $j = \sum_{c=1, c \neq k}^{l} \left[(i_c - 1) \prod_{m=1, m \neq k}^{c-1} n_m \right]$.

Slices are two-dimensional sections of a tensor, defined by fixing all but two indices. For a third-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the frontal slices are the matrix with the third index fixed, denote as $X_{:\,:\,k}$ $(k = 1, 2, \ldots, n_3)$.

For $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we denote $\widehat{\mathcal{X}}$ as the tensor by applying discrete Fourier transformation to $\mathcal{X}$ along its third mode, which is herein denoted by the command in Matlab `fft`, and $\widehat{\mathcal{X}} = \texttt{fft}(\mathcal{X}, [], 3)$. By this way, $\mathcal{X}$ can also be computed from $\widehat{\mathcal{X}}$ by using inverse fast Fourier transformation (FFT), i.e., $\mathcal{X} = \texttt{ifft}(\widehat{\mathcal{X}}, [], 3)$.

In this paper, the **tensor nuclear norm (TNN)** of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is denoted as $\|\mathcal{X}\|_* = \frac{1}{n_3} \sum_{i=1}^{n_3} \|\widehat{X}_{::i}\|_*$, i.e., $\|\mathcal{X}\|_*$ is the average of the nuclear norm of all the frontal slices of $\widehat{\mathcal{X}}$.

For more information about tensors, please refer to [19–21] for the extensive overview.

3. Related works

According to the different data formats, existing algorithms for rain streaks removal can be divided into two categories. The first group of methods aims to remove rain streaks from a single observed rainy image. Kang et al. [22] proposed their method based on morphological component analysis (MCA) and then decomposed rainy images into clean image and rain streaks. Luo et al. [23] detected and removed rain pattern using the dictionary learning technique. Li et al. [24] removed the rain effects by adopting a patch-based Gaussian mixture model (GMM) prior of both the clean images and rain streaks in a single image. Deng et al. [25] utilized the directional global sparse priors in their deraining model. In recent years, some deep learning based methods [26–29] have drawn much attentions and achieved impressive performances in rain streak removal task. Generally, they first trained large amount of images to learn the deep features of both rain layers and clean images and then removed the static rain streaks through the tailored deep networks.

Another group of approaches focuses on the video rain streaks removal problem. Garg and Nayar [4,7] first proposed to alternate the camera settings so that the obtained videos could avoid the degradation from rain drops. In the last two decades, various efficient prior knowledge has been explored for video deraining task. Barnum et al. [30] detected and separated the rain streaks in Fourier domain where the rain had united visual effects. Bossu et al. [31] assumed that the histogram of directions of rain streaks could be modeled as the Gaussian distribution, and then they adopted the GMM to characterize rain streaks. Santhaseelan and Asari [32] utilized the phase congruency features to detect the rain streaks, then reconstructed the rain-free videos based on the intensity information. Ren et al. [33] divided the rain streaks into sparse and dense categories and then proposed their method using the matrix decomposition technique. Chen et al. [34] proposed their method based on deep convolutional neural networks. They decomposed the scene into depth consistent units by superpixel segmentation (SP) and applied the alignment of clean contents at the SP level. Liu et al. [35] built a joint deep recurrent convolutional networks to remove rain effects based on the assumption that rain streaks has low light transmittance. To characterize the wealth redundancy in videos, current state-of-the-art methods often dealt with the temporal correlation via a low-rank assumption on the whole clean video. Kim et al. [9] computed an initial rain map from the difference of the adjacent frames and refined the rain map by a pre-trained classifier. Then the rain-free videos were recovered via low-rank matrix completion. Wei et al. [36] adopted the Mixture of Gaussians (MOG) distribution to model the rain streaks, and decomposed the clean video into the low-rank background and smooth foreground. Li et al. [37] removed the rain streaks using the similar decomposition with [36], while they depicted the rain streaks as the combination of multiscale features. Jiang et al. [16] proposed a tensor-based deraining method which considered the discriminative smoothness priors. They characterized the spatial–temporal correlation by assuming whole clean videos to be low-rank tensors. It is worth noting that the above methods almost assume the whole clean videos is low-rank. However, when the video captured with moving objects and dynamic scenes, the global low-rank assumption of the clean video will not be satisfied (see Fig. 1), thus the diverse structures in videos may not be well recovered.

To better model the clean videos and characterize the redundancy of videos in both spatial and temporal domains, in this paper, we propose a tensor-based deraining method exploiting the NSS prior of clean videos. Instead of assuming the whole

rain-free video to be low-rank, we consider the low-rankness of tensors formed by similar patch groups within clean videos. In addition, the UTV penalties are also employed to enhance the directional smoothness of rain streaks and the piecewise smoothness and temporal continuity of clean videos. The proposed method will be presented in the next section.

4. The proposed method

Generally, an observed rainy video can be formulated as a linear combination of the rain-free video and the rain streaks part

$$\mathcal{O} = \mathcal{V} + \mathcal{R}, \tag{1}$$

where $\mathcal{O}$, $\mathcal{V}$, $\mathcal{R} \in \mathbb{R}^{n_h \times n_w \times n_t}$ are three tensors corresponding to the rainy video, clean video, and rain streaks, respectively. The rain streaks removal task is an ill-posed problem which aims to recover the unknown clean video $\mathcal{V}$ and rain streaks $\mathcal{R}$ from a given $\mathcal{O}$. To solve this problem, we propose a tensor-based video deraining method combining the UTV and nonlocal low-rank regularization.

For an observed rainy video $\mathcal{O} \in \mathbb{R}^{n_h \times n_w \times n_t}$ with the spatial resolution $n_h \times n_w$ and the frame number n_t, we firstly sweep across the whole video tensor along three directions with overlaps, and collect a group of spatial–temporal patches $\{\mathcal{C}_{ijk}\} \subset \mathbb{R}^{\omega \times \omega \times t}$, where $1 \le i \le n_h - \omega + 1, 1 \le j \le n_w - \omega + 1, 1 \le k \le n_t - t + 1$, and ω and t are the spatial size and the frame number, respectively, of each spatial–temporal patch. Moreover, A subset of $\{\mathcal{C}_{ijk}\}$ is treated as key patches which are selected by sweeping across the tensor along three directions with the size of overlaps being 1. We further lexicographically rearrange the frontal slices of each $\mathcal{C}_{ijk}$ as column vectors, thus each $\mathcal{C}_{ijk}$ can be represented by a 2D patch, denoted as $C_{ijk} \in \mathbb{R}^{\omega^2 \times t}$, and all the $\mathcal{C}_{ijk}$ can be rearranged to form a set of 2D patches. Due to the NSS property, we can search the most similar $m-1$ ($m > 1$) patches for each key patch and stack these patches (including this key patch) to build a third-order tensor denoted as $\mathcal{O}_i \in \mathbb{R}^{\omega^2 \times t \times m}$, where i is the index of key patches. By this way, the nonlocal similar patches of all key patches can construct a set of tensors $\{\mathcal{O}_i\}$.

To estimate the clean video and rain streaks, we can first recover the corresponding patch groups of clean videos $\{\mathcal{V}_i\}$ and rain streaks $\{\mathcal{R}_i\}$ from the rainy observation $\{\mathcal{O}_i\}$, then aggregate the results to reconstruct the $\mathcal{V}$ and $\mathcal{R}$. For the ith group, we address the rain streaks removal problem as finding the decomposition of $\mathcal{O}_i$ by solving the following optimization problem

$$\min_{\mathcal{V}_i, \mathcal{R}_i} L_1(\mathcal{V}_i) + L_2(\mathcal{R}_i), \quad s.t. \quad \mathcal{O}_i = \mathcal{V}_i + \mathcal{R}_i, \qquad \mathcal{O}_i \ge \mathcal{R}_i \ge 0, \tag{2}$$

where $L_1(\cdot)$ and $L_2(\cdot)$ are the regularizations for $\mathcal{V}_i$ and $\mathcal{R}_i$, respectively. We would further explore and analyze the priors to determine the appropriate L_1 and L_2 regularizations so that $\mathcal{V}_i$ and $\mathcal{R}_i$ can be well distinguished.

Prior knowledge for $\mathcal{V}_i$. $\mathcal{V}_i$ reflects both the NSS property and the spatial–temporal correlations of clean videos and always possesses the low-rank structure. We thus consider the low-rankness of the nonlocal similar patch group $\mathcal{V}_i$, and regularize it with TNN. TNN has been proved to be more effective than the SNN for characterizing the low-rankness of tensors [19]. And only one parameter is needed when minimizing TNN. Moreover, since the clean videos are piecewise smooth in the spatial domain and consist of continuous frames, we additionally regularize the rain-free videos with UTV penalties for the horizontal smoothness and the temporal continuity. Note that raindrops often fall down from top to the bottom due to the gravity, we assume the rain directions are uniform and nearly vertical in this paper. Actually, when rain directions are highly oblique, our method can still remove the rain streaks, which will be further discussed in our experiments in Section 6. In addition, we do not consider the piecewise smoothness of clean videos along rain direction, because the rain steaks maintain a stronger smooth property along this direction [16]. Therefore, under the assumption of vertical rain streaks, we define L_1 as

$$L_1(\mathcal{V}_i) = \|\mathcal{V}_i\|_* + \alpha_1 \|D_x \mathcal{V}_i\|_1 + \alpha_2 \|D_t \mathcal{V}_i\|_1, \tag{3}$$

where $\|\cdot\|_*$ is the TNN defined in Section 2, $\|\cdot\|_1$ is the ℓ_1 norm which calculates the sum of absolute values of all entries, $\alpha_1, \alpha_2 > 0$ are two positive parameters, and $D_x, D_t : \mathbb{R}^{\omega^2 \times t \times m} \to \mathbb{R}^{\omega^2 \times t \times m}$ are defined as the operators that compute the differential values of horizontal and temporal directions in the original spatial–temporal patches. Specifically, we define $\{\mathcal{V}_i^{(j)}\}_{j=1}^m \in \mathbb{R}^{\omega \times \omega \times t}$ as the spatial–temporal patches reshaped from the frontal slices of tensor $\mathcal{V}_i$. We further respectively define $\nabla_x, \nabla_t : \mathbb{R}^{\omega \times \omega \times t} \to \mathbb{R}^{\omega \times \omega \times t}$ as the unidirectional first-order difference operators of horizontal and temporal directions for tensors. Then by reshaping the tensors in the sets $\{\nabla_x \mathcal{V}_i^{(j)}\}_{j=1}^m$ and $\{\nabla_t \mathcal{V}_i^{(j)}\}_{j=1}^m$ as 2D matrices with size $\omega^2 \times t$ and stacking them, we can construct $D_x \mathcal{V}_i$ and $D_t \mathcal{V}_i$, respectively.

Prior knowledge for $\mathcal{R}_i$. Since the number of rain pixels is often less than that of clean pixels and the rain streaks often have bright intensity, we assume $\mathcal{R}$ to be a sparse tensor. Obviously, $\mathcal{R}_i$ preserves the sparsity of rain streaks, thus we use the l_1 norm to regularize $\mathcal{R}_i$ in our model. Additionally, we assume the directions of rain streaks are nearly uniform and vertical, thus the rain streaks naturally have strong smoothness along the vertical direction, and we regularize the rain streaks with the UTV penalty for the directional smoothness. Therefore, the regularization L_2 then can be defined as

$$L_2(\mathcal{R}_i) = \alpha_3 \|D_y \mathcal{R}_i\|_1 + \alpha_4 \|\mathcal{R}_i\|_1, \tag{4}$$

where $\alpha_3, \alpha_4 > 0$ are the positive parameters, $D_y : \mathbb{R}^{\omega^2 \times t \times m} \to \mathbb{R}^{\omega^2 \times t \times m}$ is the operator to compute the differential values of vertical direction in the original spatial–temporal patches. Here D_y is defined similarly to D_x and D_t, i.e., $D_y \mathcal{R}_i$ is constructed

by reshaping the tensors $\{\nabla_y \mathcal{R}_i^{(j)}\}_{j=1}^m$ as matrices and stacking them into a 3-order tensor, where ∇_y and $\mathcal{R}_i^{(j)}$ are the first-order difference operator of vertical direction and the spatial–temporal patch reshaped from the jth frontal slice of $\mathcal{R}_i$, respectively.

By incorporating (3) and (4) into (2), for the ith group of nonlocal similar patches, we have the following optimization model

$$\begin{aligned} &\min_{\mathcal{V}_i,\mathcal{R}_i} \|\mathcal{V}_i\|_* + \alpha_1\|D_x\mathcal{V}_i\|_1 + \alpha_2\|D_t\mathcal{V}_i\|_1 + \alpha_3\|D_y\mathcal{R}_i\|_1 + \alpha_4\|\mathcal{R}_i\|_1, \\ &\quad s.t. \quad \mathcal{O}_i = \mathcal{V}_i + \mathcal{R}_i, \qquad \mathcal{O}_i \geq \mathcal{R}_i \geq 0. \end{aligned} \tag{5}$$

It is worth mentioning that all the variables in our model are the tensors built by the similar patch groups instead of the original videos, which is much different between the work of [16]. The work [16] considered holistic low-rankness of the clean videos, that are not always satisfied especially when there are moving objects and dynamic background in videos. Our work considers the NSS [38] prior and fully exploits the redundancy of nonlocal similar patches of the clean videos. The NSS is a more general prior which is satisfied even when there are moving objects and dynamic background in clean videos. For example, in Fig. 1, we can observe that the low-rankness of the holistic clean video is not significant while the low-rankness of nonlocal similar patches is significant. Moreover, Our method adopts the TNN that is more effective than the SNN employed in [16].

When all $\mathcal{V}_i$'s are recovered by solving (5), we can reconstruct the final clean $\mathcal{V}$ by aggregating all $\mathcal{V}_i$s to compute the average intensity values for each pixel. The efficient numerical algorithm for the proposed model will be introduced in the next section.

5. Numerical algorithms

The proposed model (5) is a convex optimization problem and easy to be solved by many efficient solvers, such as Bregman method [39], proximal splitting method, primal-dual methods [40,41], and ADMM [42]. In this paper, we develop an efficient ADMM algorithm to tackle the proposed model. The other methods can also be applied and implemented effectively.

We first introduce some auxiliary variables $\mathcal{U}, \mathcal{W}, \mathcal{M}, \mathcal{N}$, and $\mathcal{P}$ into (5). For the ith group of nonlocal similar patches, we can then rewrite the problem as the following equivalent optimization problem

$$\begin{aligned} \min_{\substack{\mathcal{V}_i,\mathcal{R}_i,\mathcal{U},\\ \mathcal{W},\mathcal{M},\mathcal{N},\mathcal{P}}} \quad & \|\mathcal{U}\|_* + \alpha_1\|\mathcal{W}\|_1 + \alpha_2\|\mathcal{M}\|_1 + \alpha_3\|\mathcal{N}\|_1 + \alpha_4\|\mathcal{P}\|_1, \\ s.t. \quad & \mathcal{U} = \mathcal{V}_i, \quad \mathcal{W} = D_x\mathcal{V}_i, \quad \mathcal{M} = D_t\mathcal{V}_i, \quad \mathcal{N} = D_y\mathcal{R}_i, \\ & \mathcal{P} = \mathcal{R}_i, \quad \mathcal{O}_i = \mathcal{V}_i + \mathcal{R}_i, \quad \mathcal{O}_i \geq \mathcal{R}_i \geq 0. \end{aligned} \tag{6}$$

By attaching the Lagrangian multipliers $\{\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3, \mathbf{\Lambda}_4, \mathbf{\Lambda}_5, \mathbf{\Lambda}_6\}$, the augmented Lagrangian function for (6) can be formulated as

$$\begin{aligned} &\mathcal{L}(\mathcal{V}_i, \mathcal{R}_i, \mathcal{U}, \mathcal{W}, \mathcal{M}, \mathcal{N}, \mathcal{P}, \mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3, \mathbf{\Lambda}_4, \mathbf{\Lambda}_5, \mathbf{\Lambda}_6) \\ &= \|\mathcal{U}\|_* + \alpha_1\|\mathcal{W}\|_1 + \alpha_2\|\mathcal{M}\|_1 + \alpha_3\|\mathcal{N}\|_1 + \alpha_4\|\mathcal{P}\|_1 \\ &\quad + \frac{\beta}{2}\|\mathcal{U} - \mathcal{V}_i + \frac{\mathbf{\Lambda}_1}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{W} - D_x\mathcal{V}_i + \frac{\mathbf{\Lambda}_2}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{M} - D_t\mathcal{V}_i + \frac{\mathbf{\Lambda}_3}{\beta}\|_F^2 \\ &\quad + \frac{\beta}{2}\|\mathcal{N} - D_y\mathcal{R}_i + \frac{\mathbf{\Lambda}_4}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{P} - \mathcal{R}_i + \frac{\mathbf{\Lambda}_5}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{O}_i - \mathcal{V}_i - \mathcal{R}_i + \frac{\mathbf{\Lambda}_6}{\beta}\|_F^2, \end{aligned} \tag{7}$$

where β is a positive scalar. In (7), the unknown variables can be divided into two groups $(\mathcal{U}, \mathcal{W}, \mathcal{M}, \mathcal{N}, \mathcal{P})$ and $(\mathcal{V}_i, \mathcal{R}_i)$. We can solve them alternatively within ADMM framework. Firstly, we keep the variables in $(\mathcal{V}_i, \mathcal{R}_i)$ fixed and solve the variables in another group. To begin with, we can update $\mathcal{U}$ by solving the following sub-problem

$$\min_{\mathcal{U}} \|\mathcal{U}\|_* + \frac{\beta}{2}\|\mathcal{U} - \mathcal{V}_i + \frac{\mathbf{\Lambda}_1}{\beta}\|_F^2. \tag{8}$$

Here $\|\cdot\|_*$ is the tensor nuclear norm (TNN) defined in Section 2. Problem (8) can be efficiently solved by using the discrete Fourier transform [19] and the singular value thresholding (SVT). Specifically, we obtain each frontal slice of $\mathcal{U}^{k+1}$ via

$$\widehat{U}_{::j}^{k+1} = \mathrm{SVT}\left([\widehat{V}_i^k]_{::j} - \frac{\widehat{[\Lambda_1^k]}_{::j}}{\beta}, \frac{1}{\beta}\right), \qquad j = 1, \ldots, m, \tag{9}$$

where SVT is the soft-thresholding operation of singular values. Then we can apply inverse FFT to obtain $\mathcal{U}$. Please refer to [19,43] for more details. Here the computational cost of updating $\mathcal{U}^{k+1}$ is $O(\omega^2 tm \log m + m\min(\omega^2 t^2, \omega^4 t))$.

The sub-problems for $\mathcal{W}, \mathcal{M}, \mathcal{N}$, and $\mathcal{P}$ can be solved by the same optimization problem, which can be written as the following formulation

$$\min_{\mathcal{X}} \rho\|\mathcal{X}\|_1 + \frac{\mu}{2}\|\mathcal{X} - \mathcal{Y}\|_F^2, \tag{10}$$

where ρ and μ are positive scalars. This problem can be solved by applying the operation as follows

$$\mathcal{X} = S_{\frac{\rho}{\mu}}(\mathcal{Y}), \tag{11}$$

where $S(\cdot)$ is performed componentwisely and defined as

$$S_{\frac{\rho}{\mu}}(y) = \begin{cases} \left(|y| - \dfrac{\rho}{\mu}\right) \cdot \mathtt{sign}(y), & |y| > \dfrac{\rho}{\mu}; \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

where $\mathtt{sign}(a)$ is 1 if $a > 0$ and -1 otherwise. Then $\mathcal{W}, \mathcal{M}, \mathcal{N}$, and $\mathcal{P}$ can be updated by the following closed-form solutions

$$\begin{aligned} &\mathcal{W}^{k+1} = S_{\frac{\alpha_1}{\beta}}\left(D_x\mathcal{V}_i^k - \frac{\mathbf{\Lambda}_2^k}{\beta}\right), \ \mathcal{M}^{k+1} = S_{\frac{\alpha_2}{\beta}}\left(D_t\mathcal{V}_i^k - \frac{\mathbf{\Lambda}_3^k}{\beta}\right), \\ &\mathcal{N}^{k+1} = S_{\frac{\alpha_3}{\beta}}\left(D_y\mathcal{R}_i^k - \frac{\mathbf{\Lambda}_4^k}{\beta}\right), \ \mathcal{P}^{k+1} = S_{\frac{\alpha_4}{\beta}}\left(\mathcal{R}_i^k - \frac{\mathbf{\Lambda}_5^k}{\beta}\right). \end{aligned} \tag{13}$$

The computational cost of updating $\mathcal{W}, \mathcal{M}, \mathcal{N}$, and $\mathcal{P}$ is $O(\omega^2 tm)$.

To solve $\mathcal{V}_i$ and $\mathcal{R}_i$, we keep the variables $(\mathcal{U}, \mathcal{W}, \mathcal{M}, \mathcal{N}, \mathcal{P})$ fixed and minimizing (7) with respect to $\mathcal{V}_i$ and $\mathcal{R}_i$. Obviously, they can be jointly solved by the following least squares problem

$$\begin{aligned} \min_{\mathcal{V}_i, \mathcal{R}_i} \ & \frac{\beta}{2}\|\mathcal{U} - \mathcal{V}_i + \frac{\mathbf{\Lambda}_1}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{W} - D_x\mathcal{V}_i + \frac{\mathbf{\Lambda}_2}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{M} - D_t\mathcal{V}_i + \frac{\mathbf{\Lambda}_3}{\beta}\|_F^2 \\ & + \frac{\beta}{2}\|\mathcal{N} - D_y\mathcal{R}_i + \frac{\mathbf{\Lambda}_4}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{P} - \mathcal{R}_i + \frac{\mathbf{\Lambda}_5}{\beta}\|_F^2 + \frac{\beta}{2}\|\mathcal{O}_i - \mathcal{V}_i - \mathcal{R}_i + \frac{\mathbf{\Lambda}_6}{\beta}\|_F^2. \end{aligned} \tag{14}$$

The problem (14) is equivalent to the following vectorized formulation

$$\begin{aligned} \min_{\mathcal{V}_i, \mathcal{R}_i} \ & \frac{\beta}{2}\|\mathtt{vec}(\mathcal{U}) - \mathtt{vec}(\mathcal{V}_i) + \mathtt{vec}\left(\frac{\mathbf{\Lambda}_1}{\beta}\right)\|_2^2 + \frac{\beta}{2}\|\mathtt{vec}(\mathcal{W}) - \mathtt{vec}(D_x\mathcal{V}_i) + \mathtt{vec}\left(\frac{\mathbf{\Lambda}_2}{\beta}\right)\|_2^2 \\ & + \frac{\beta}{2}\|\mathtt{vec}(\mathcal{M}) - \mathtt{vec}(D_t\mathcal{V}_i) + \mathtt{vec}\left(\frac{\mathbf{\Lambda}_3}{\beta}\right)\|_2^2 + \frac{\beta}{2}\|\mathtt{vec}(\mathcal{N}) - \mathtt{vec}(D_y\mathcal{R}_i) + \mathtt{vec}\left(\frac{\mathbf{\Lambda}_4}{\beta}\right)\|_2^2 \\ & + \frac{\beta}{2}\|\mathtt{vec}(\mathcal{P}) - \mathtt{vec}(\mathcal{R}_i) + \mathtt{vec}\left(\frac{\mathbf{\Lambda}_5}{\beta}\right)\|_2^2 + \frac{\beta}{2}\|\mathtt{vec}(\mathcal{O}_i) - \mathtt{vec}(\mathcal{V}_i) - \mathtt{vec}(\mathcal{R}_i) + \mathtt{vec}\left(\frac{\mathbf{\Lambda}_6}{\beta}\right)\|_2^2, \end{aligned} \tag{15}$$

where $\mathtt{vec}(\cdot)$ is the vectorization operation. By reshaping tensors into the vector form, we can further represent the terms $\mathtt{vec}(D_x\mathcal{V}_i)$, $\mathtt{vec}(D_t\mathcal{V}_i)$, and $\mathtt{vec}(D_y\mathcal{R}_i)$ by the form of first-order difference matrices multiplying vectors. In specific, we can construct the first-order difference matrices $F_x, F_y, F_t \in \mathbb{R}^{\omega^2 tm \times \omega^2 tm}$ that satisfy the following equations

$$\mathtt{vec}(D_x\mathcal{V}_i) = F_x \cdot \mathtt{vec}(\mathcal{V}_i), \quad \mathtt{vec}(D_t\mathcal{V}_i) = F_t \cdot \mathtt{vec}(\mathcal{V}_i), \quad \mathtt{vec}(D_y\mathcal{R}_i) = F_y \cdot \mathtt{vec}(\mathcal{R}_i). \tag{16}$$

It can be easily derived that F_x, F_y, and F_t are the matrices with circulant blocks. In (15), through replacing $\mathtt{vec}(D_x\mathcal{V}_i)$, $\mathtt{vec}(D_t\mathcal{V}_i)$, and $\mathtt{vec}(D_y\mathcal{R}_i)$ by the corresponding terms in (16), the objective function of (15) becomes derivable. We can then minimize it to get the following normal equation

$$\begin{pmatrix} F_x^\top F_x + F_t^\top F_t + 2I & I \\ I & F_y^\top F_y + 2I \end{pmatrix} \begin{pmatrix} \mathtt{vec}(\mathcal{V}_i) \\ \mathtt{vec}(\mathcal{R}_i) \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{17}$$

where

$$\begin{aligned} b_1 &= \mathtt{vec}(\mathcal{O}_i + \mathcal{U} + \frac{1}{\beta}(\mathbf{\Lambda}_1 + \mathbf{\Lambda}_6)) + F_x^\top\left(\mathtt{vec}\left(\mathcal{W} + \frac{1}{\beta}\mathbf{\Lambda}_2\right)\right) + F_t^\top\left(\mathtt{vec}\left(\mathcal{M} + \frac{1}{\beta}\mathbf{\Lambda}_3\right)\right), \\ b_2 &= \mathtt{vec}(\mathcal{O}_i + \mathcal{P} + \frac{1}{\beta}(\mathbf{\Lambda}_5 + \mathbf{\Lambda}_6)) + F_y^\top\left(\mathtt{vec}\left(\mathcal{N} + \frac{1}{\beta}\mathbf{\Lambda}_4\right)\right), \end{aligned} \tag{18}$$

and $I \in \mathbb{R}^{\omega^2 tm \times \omega^2 tm}$ is the identity matrix. Due to the circulant structure of $F_x^\top F_x, F_y^\top F_y$ and $F_t^\top F_t$, the coefficient matrices $(F_x^\top F_x + F_t^\top F_t + 2I)$ and $(F_y^\top F_y + 2I)$ can be diagonalized by FFTs. Therefore, problem (17) can be efficiently solved in

$O(\omega^2 tm\log(\omega^2 tm))$ operations. Then $\mathcal{R}_i^{k+1}$ and $\mathcal{V}_i^{k+1}$ are elementwisely projected between 0 and $\mathcal{O}_i$. At last, the Lagrangian multipliers can be updated by

$$\begin{cases} \mathbf{\Lambda}_1^{k+1} = \mathbf{\Lambda}_1^k + \beta(\mathcal{U}_i^{k+1} - \mathcal{V}_i^{k+1}), \\ \mathbf{\Lambda}_2^{k+1} = \mathbf{\Lambda}_2^k + \beta(\mathcal{W}^{k+1} - D_x\mathcal{V}_i^{k+1}), \\ \mathbf{\Lambda}_3^{k+1} = \mathbf{\Lambda}_3^k + \beta(\mathcal{M}^{k+1} - D_t\mathcal{V}_i^{k+1}), \\ \mathbf{\Lambda}_4^{k+1} = \mathbf{\Lambda}_4^k + \beta(\mathcal{N}^{k+1} - D_y\mathcal{R}_i^{k+1}), \\ \mathbf{\Lambda}_5^{k+1} = \mathbf{\Lambda}_5^k + \beta(\mathcal{P}^{k+1} - \mathcal{R}_i^{k+1}), \\ \mathbf{\Lambda}_6^{k+1} = \mathbf{\Lambda}_6^k + \beta(\mathcal{O}_i^{k+1} - \mathcal{V}_i^{k+1} - \mathcal{R}_i^{k+1}). \end{cases} \tag{19}$$

The cost of updating these multipliers is $O(\omega^2 tm)$.

We perform our algorithm on all the similar patch groups. The final result $\mathcal{V}$ can be obtained by aggregating all the result $\mathcal{V}_i$'s and computing the average values of each pixel. In the next section, we will show our experimental results.

The analysis of computational and storage complexity:

In our method, the block-matching operation, which aims to search the similar patches for every key patch, is the most time-consuming procedure besides the proposed ADMM algorithm. To analyze the complexity of the block-matching procedure, we use n_a and n_b to denote the total number of spatial–temporal patches and key patches, respectively. In this way, the cost of computing the distances between each key patch and all patches is $O(\omega^2 tn_a)$. Moreover, the cost of finding similar patches for each key patch is $n_a \log n_a$. Thus, the total computational complexity of block matching is $O(n_a n_b(\log n_a + \omega^2 t))$. Meanwhile, the storage complexity can also be calculated as $O(n_b(n_a + 1 + m) + n_a\omega^2 t)$.

For the proposed ADMM algorithm, by summarizing the computational cost of all variables, the total computational complexity of each patch group is $O(\omega^2 tm\log\omega^2 tm^2 + m\min(\omega^2 t^2, \omega^4 t))$ per iteration. Moreover, the storage complexity can be easily calculated as $O(\omega^2 tm)$.

6. Experiments

In this section, we show and analyze the rain removal performance on both the synthetic data and real data. Our experiments are implemented in MATLAB 2017b on an Intel 3.70 GHz PC with 16GB RAM. As each frame of color videos have RGB three channels, we transform all the test videos into YUV[1] color space and perform our method on the components of brightness. For the parameters, unless otherwise specified, we set $\beta = 2$ and $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ are tuned in the interval [0.1,2] empirically. We set $\omega = 15, t = 10, m = 30$, i.e., the size of spatial–temporal patch is fixed as $15 \times 15 \times 10$, and the number of similar patches is 30. For each key spatial–temporal patch, we search its similar patches in an $80 \times 80 \times 30$ windows centered at this patch. We compare the proposed method (UTV-NLR) with four recent state-of-the-art video deraining methods: the temporal correlation and low-rank matrix completion (TC-LRMC) method [9], the stochastic encoding (SE) method [36], the tensor-based method using the discriminate intrinsic priors (DIP) [16], and the convolutional sparse coding (CSC) method [37]. We test these methods on both the synthetic data and real rainy videos quantitatively and visually.

6.1. Experiments on synthetic data

All the compared methods are first performed on four clean videos with the artificial rain streaks. To generate rain streaks, Garg and Nayar [4] pointed out that the effects of rain streaks are caused by the motion-blurred raindrops. Thus many recent deraining approaches [22,23,27] utilize the Photoshop software and follow the tutorial documents of Photoshop [44] to add the rain streaks on a single image. The main idea of this way is to first generate some noise pixels on a black layer and then apply the motion blur to the noise. In addition, the simulation of rain streaks in videos has been proposed in early research [45], which claims that the rain streaks can be assumed temporal independent in videos. Based on these studies, we simulate the rain streaks $\mathcal{R}$ by first generating the "salt and pepper" noise on each frontal slice of a zero tensor, and then applying a motion blur to it. A small bias angle is allowed between the rain direction and vertical direction (maximum 5 degrees). In this section, we use two scenarios for each video: the video with light rain and the video with heavy rain. The noise density d of light rain is set as $d = 0.01 + 0.001 * p$ and the one of heavy rain is set as $d = 0.04 + 0.001 * p$, where $0 < p < 1$ is a positive scalar taken randomly for each frame. Since the clean videos are known, we make the comparison quantitatively and visually.

Table 1 shows the quantitative comparisons of four synthetic videos. Besides the PSNR and SSIM [46], we additionally use 3 performance metrics: the universal image quality index (UQI) [47], visual information fidelity (VIF) [48], and the feature similarity index (FSIM) [49]. The values of all metrics are the higher, the better. These measures mainly focus on image structure and fidelity so that the comparison from different aspects can be fully considered. In Table 1, we list the mean values of metrics over all frames for each recovered video. From Table 1, we can evidently observe the advantage of the

[1] https://en.wikipedia.org/wiki/YUV

Table 1
Quantitative evaluation on synthetic videos with different metrics.

Video	Scenarios	Methods	MPSNR	MSSIM	MUQI	MVIF	MFSIM
Running car	Light rain	Rainy	36.0193	0.9539	0.8762	0.6987	0.9762
		TC-LRMC	36.6063	0.9799	0.9555	0.8025	0.9833
		SE	37.8375	0.9713	0.9566	0.8556	0.9947
		DIP	43.1908	0.9945	0.9822	0.9103	0.9952
		CSC	43.9136	0.9934	0.9774	0.9415	0.9756
		UTV-NLR	**48.9292**	**0.9976**	**0.9894**	**0.9674**	**0.9985**
	Heavy rain	Rainy	28.7164	0.8556	0.7227	0.5038	0.9290
		TC-LRMC	31.3614	0.9202	0.8047	0.5067	0.9460
		SE	31.8453	0.9575	0.9126	0.7617	0.9840
		DIP	35.3519	0.9664	0.8849	0.7072	0.9796
		CSC	39.4965	0.9869	0.9526	**0.8675**	**0.9923**
		UTV-NLR	**40.0771**	**0.9877**	**0.9549**	0.8219	0.9904
Foreman	Light rain	Rainy	35.3353	0.9710	0.8840	0.6831	0.9698
		TC-LRMC	35.5148	0.9854	0.9372	0.7218	0.9796
		SE	28.9493	0.8966	0.8306	0.6203	0.9705
		DIP	39.7551	0.9923	0.9556	0.8005	0.9877
		CSC	28.2257	0.9249	0.7040	0.4875	0.8978
		UTV-NLR	**41.0047**	**0.9949**	**0.9664**	**0.8339**	**0.9922**
	Heavy rain	Rainy	28.2117	0.9088	0.7498	0.4955	0.9106
		TC-LRMC	29.6899	0.9328	0.7837	0.4452	0.9275
		SE	28.0962	0.8725	0.7531	0.4343	0.9300
		DIP	34.2657	0.9808	0.8993	0.6520	0.9713
		CSC	27.7727	0.9151	0.6833	0.4103	0.8904
		UTV-NLR	**35.0302**	**0.9851**	**0.9167**	**0.6769**	**0.9764**
Highway	Light rain	Rainy	34.0977	0.9141	0.8446	0.6141	0.9580
		TC-LRMC	35.7116	0.9383	0.8515	0.7290	0.9865
		SE	26.1085	0.7944	0.7095	0.5541	0.9738
		DIP	44.6563	0.9934	0.9840	0.9333	0.9955
		CSC	33.8929	0.9500	0.8888	0.6840	0.9648
		Ours	**45.8307**	**0.9943**	**0.9867**	**0.9346**	**0.9966**
	Heavy rain	Rainy	27.6660	0.7576	0.6250	0.4323	0.8798
		TC-LRMC	30.8193	0.8580	0.7065	0.4341	0.9482
		SE	26.6654	0.7308	0.5752	0.3661	0.9491
		DIP	40.8268	0.9824	0.9588	0.8321	0.9896
		CSC	33.5574	0.9399	0.8657	0.6343	0.9643
		UTV-NLR	**42.7556**	**0.9902**	**0.9767**	**0.8915**	**0.9939**
Walking	Light rain	Rainy	30.9037	0.9087	0.7932	0.5512	0.9375
		TC-LRMC	33.4872	0.9647	0.9170	0.7546	0.9660
		SE	29.0450	0.9062	0.9493	0.7661	0.9834
		DIP	37.0518	0.9764	0.9300	0.7463	0.9809
		CSC	39.6051	0.9881	0.9684	**0.8763**	0.9889
		UTV-NLR	**40.1405**	**0.9893**	**0.9699**	0.8703	**0.9901**
	Heavy rain	Rainy	25.8174	0.8106	0.6411	0.4233	0.8817
		TC-LRMC	29.5570	0.8935	0.7404	0.4843	0.9219
		SE	31.4056	0.9474	0.9259	0.7014	0.9722
		DIP	33.3066	0.9565	0.8724	0.7014	0.9644
		CSC	33.6888	0.9715	**0.9273**	0.7284	0.9756
		UTV-NLR	**35.8010**	**0.9726**	0.9213	**0.7482**	**0.9774**

proposed UTV-NLR. It shows that UTV-NLR achieves the highest PSNR and SSIM values in all cases, and highest UQI, VIF, and FSIM values in most cases, which demonstrates the superiority of the proposed UTV-NLR in removing rain streaks and preserving the detailed information of videos.

Fig. 3 shows the visual comparison of rain removal results and the corresponding rain streak maps on the video "running car" with light rain and heavy rain. A positive value (fixed as 0.4 in this paper) is added to each rain streaks map for better visualization. This video is of size $120 \times 160 \times 3 \times 30$. For the case of light rain, we can observe that the rain-free videos recovered by TC-LRMC and SE still have notable rain streaks left, while the results of DIP, CSC, and UTV-NLR achieve better visual performance. In rain streak maps, it shows that the results of other compared methods contain some non-rain pixels more or less, that means some bright pixels of scenes are considered as rain pixels by these methods. For the case of heavy rain, it can be easily observed that the other compared methods cannot remove the heavy rain completely, and many bright pixels of scenes are regarded as the rain pixels in their rain maps. From the quantitative comparison in Table 1, we can see that UTV-NLR achieves the best PSNR results, and at least 5dB and 0.5dB gain beyond the second best performed method for the light rain and heavy rain case, respectively.

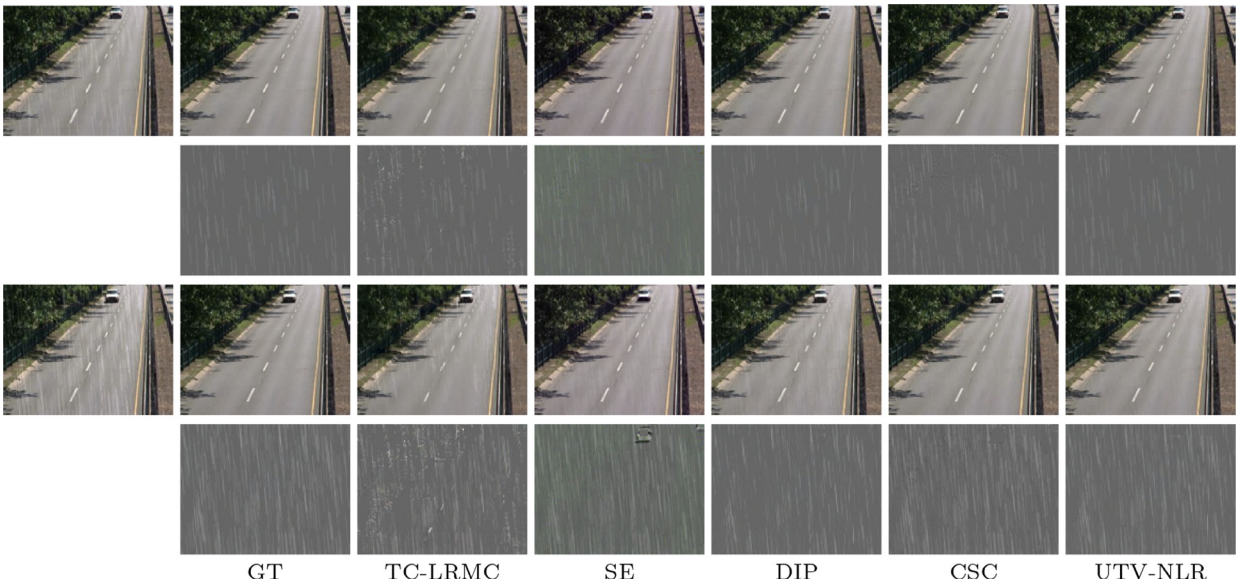

Fig. 3. Rain removal results and the corresponding rain maps on "running car" video with light (the first and second rows) and heavy (the third and fourth rows) rain. The first column shows the rainy frames. From the second to the last column are: ground truth (GT), TC-LRMC [9], SE [36], DIP [16], CSC [37], and the proposed UTV-NLR, respectively.

Fig. 4 shows the visual performance on a video named "foreman", whose size is $144 \times 166 \times 3 \times 30$. In the case of light rain, we can see that SE and CSC fail to remove the rain and their results contain distorted colors of scenes, that is caused by the fact their algorithms assume videos have static backgrounds, but "foreman" is a video taken by the moving camera. The proposed UTV-NLR slightly outperforms the other compared methods and recovers the rain-free video with least rain streaks left. For the case of heavy rain, TC-LRMC, SE, and CSC fail to remove the rain streaks. The rain-free video recovered by DIP still contains rain streaks. Echoing with the visual results, in Table 1 we can see that UTV-NLR outperforms the competing methods by about 1.1dB in light rain case and 0.7dB in heavy rain case. Moreover, UTV-NLR achieves the highest values in other metrics.

Fig. 5 shows the visual comparison on a video named "highway" with the size $144 \times 166 \times 3 \times 30$. In the case of light rain in the first two rows, we can observe that the recovered rain-free videos of SE and DIP still contain some rain streaks in the demarcated areas (the zoomed versions are in the left corner of images). In the zoomed region, the results of TC-LRMC and CSC have over smooth effects. Fig. 5 shows that the proposed UTV-NLR achieves the best visual results. Moreover, from Table 1 we can see that UTV-NLR achieves more than 1.1dB gain in PSNR beyond the other competing methods and the highest values in other metrics. In the case of heavy rain, both the results of TC-LRMC and DIP have clear rain streaks left. Since this video is captured by the moving camera with the dynamic background, the result videos recovered by SE and CSC contain some rain streaks and lose much details of the clean video. Table 1 shows that the proposed UTV-NLR comprehensively removes the rain streaks, and outperforms the other compared methods by more than 1.8 dB in heavy rain cases.

Fig. 6 shows the visual comparison on a video named "walking" with the size of $144 \times 180 \times 3 \times 30$. This video contains a static background and a walking woman. In the case of light rain, all methods succeed to remove the rain and achieve similar visual performances. While the demarcated areas demonstrate that UTV-NLR recovers better videos with less rain streaks and cleaner scenes. Specifically, from Table 1 we can see UTV-NLR outperforms the other compared methods by about 0.5dB in this case. In the case of heavy rain, TC-LRMC cannot remove the rain streaks completely. And the recovered rain-free video is corrupted by dark pixels in some regions. In the zoomed region, SE fails to remove the rain streaks due to the effect of the moving object. DIP removes the rain streaks but burs the video. CSC can not remove the rain streaks completely in this region. The proposed UTV-NLR removes the rain streaks and preserves clean patterns in the original video. In addition, UTV-NLR outperforms the other competing methods by at least 2 dB in heavy rain case.

Fig. 7 demonstrates the PSNR values of recovered clean videos in Figs. 3–6. We calculate the PSNR values of each frame for the recovered videos of different methods. It shows that SE and CSC sometimes fails to remove the rain and achieve low PSNR values. The reason is that their methods rely on low-rank assumption of static backgrounds. TC-LRMC, DIP, and UTV-NLR achieve higher PSNR values, while our method always achieves the highest PSNR values and stabler curves for most cases. Fig. 7 quantitatively demonstrates the superiority and robustness of the proposed method.

Table 2 shows the computing time of experiments from Figs. 3–6. We can see that the proposed UTV-NLR always costs more than 800s because it need to search similar patches for every key patch and compute each pixel value multiple times (because the patches are overlapped). However, UTV-NLR can achieve the best performance because it takes advantage of the

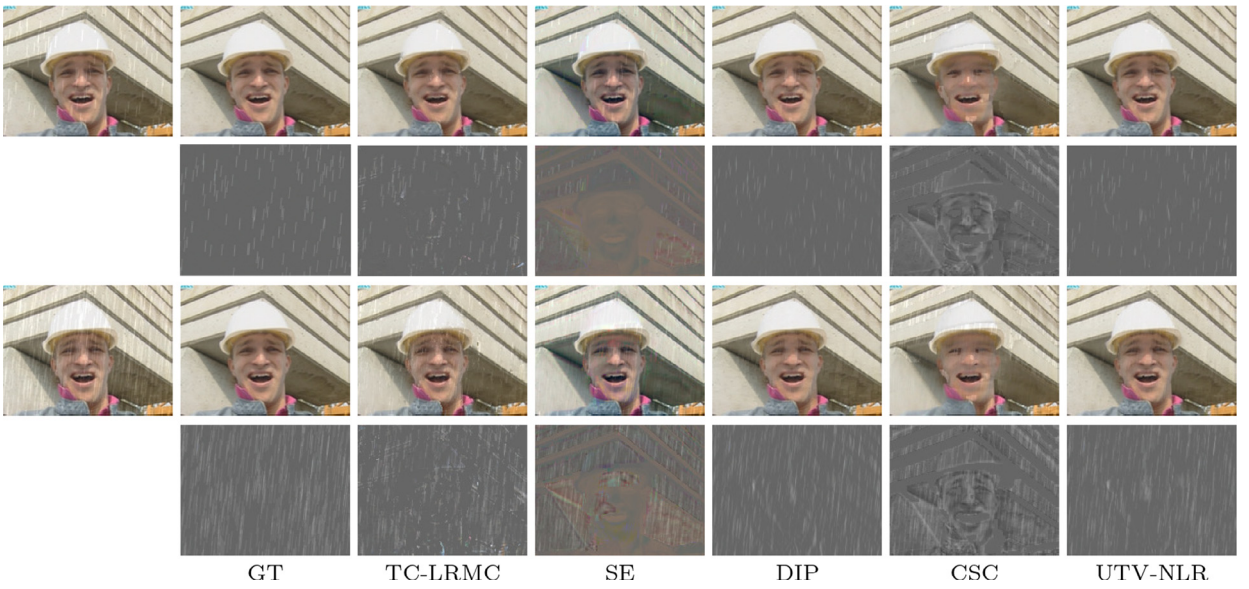

Fig. 4. Rain removal results and the corresponding rain maps on "foreman" video with light (the first and second rows) and heavy (the third and fourth rows) rain. The first column shows the rainy frames.

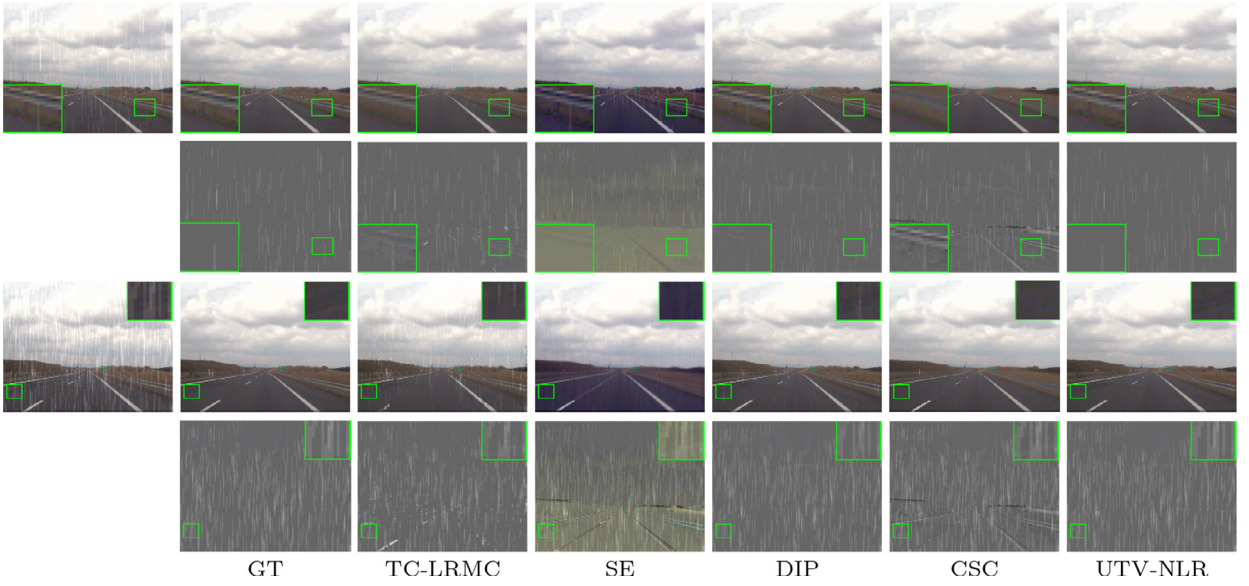

Fig. 5. Rain removal results and the corresponding rain maps on "highway" video with light (the first and second rows) and heavy (the third and fourth rows) rain. The first column shows the rainy frames.

Table 2
The computing time of all compared methods on the images from Figs. 3–6.

Computing time (s)						
Video	Scenarios	TC-LRMC	SE	DIP	CSC	UTV-NLR
Running car	Light rain	248.86	50.18	19.98	22.90	860.31
	Heavy rain	391.14	41.96	28.72	23.77	802.51
Foreman	Light rain	318.75	30.03	47.59	30.93	839.91
	Heavy rain	265.35	31.56	43.54	31.28	879.53
Highway	Light rain	249.71	72.79	43.59	29.40	897.68
	Heavy rain	265.35	31.56	43.54	29.25	879.53
Walking	Light rain	336.90	73.82	41.30	29.24	1043.81
	Heavy rain	549.44	60.42	37.46	29.56	1002.52

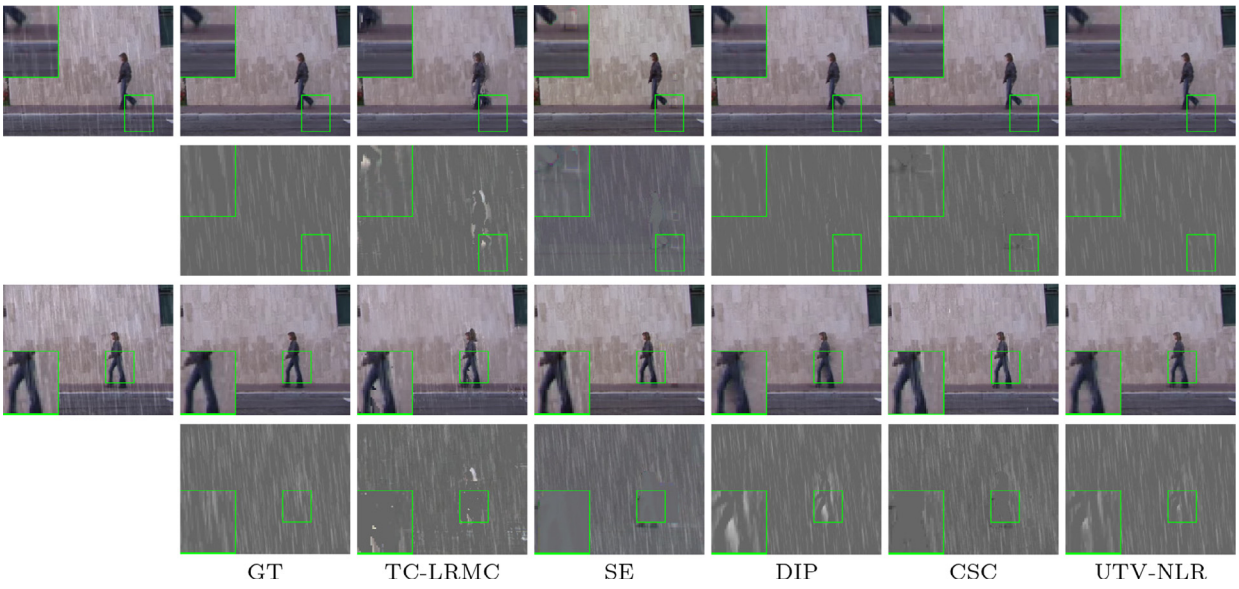

Fig. 6. Rain removal results and the corresponding rain maps on "walking" video with light (the first and second rows) and heavy (the third and fourth rows) rain. The first column shows the rainy frames.

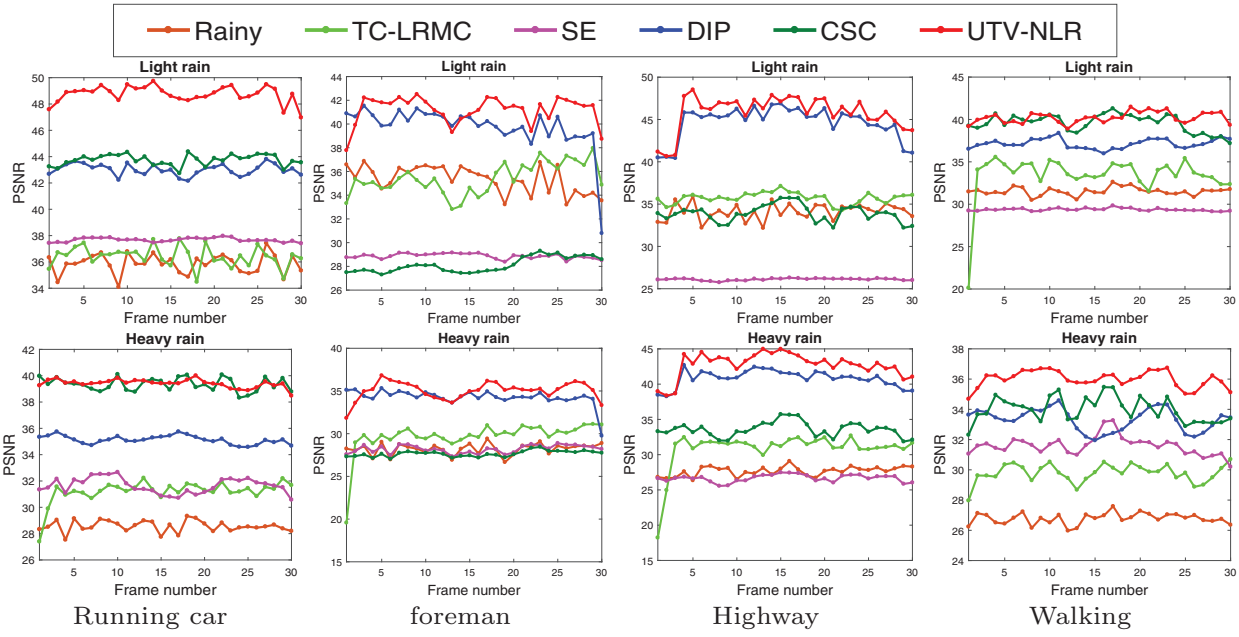

Fig. 7. The PSNR values of each frame in the videos from Figs. 3 to 6.

redundancy of nonlocal similar patches. Table 2 illustrates that our method sacrifices the computing time for high recovery accuracy.

6.2. Oblique rain streaks removal

It should be noted that even though our method is based on the assumption that the directions of rain streaks are vertical in rainy videos, when rain directions are not strictly but nearly vertical, our method can still remove the rain successfully. This can also be verified in Section 6.1, where the rainy videos of Figs. 3, 4, and 6 are all generated with oblique rain streaks with small angles.

In this section, we mainly evaluate our method on the videos with highly oblique rain streaks. We denote γ as the angle between the main rain direction and vertical direction, and generate the rain with three different directions: $\gamma = 20^\circ, 30^\circ,$

Table 3
Quantitative evaluation on the videos with oblique rain streaks.

Bridge						
γ	Methods	MPSNR	MSSIM	MUQI	MVIF	MFSIM
20°	Rainy	28.7092	0.8468	0.5503	0.4150	0.8649
	TC-LRMC	33.1017	0.9216	0.6173	0.4798	0.9476
	SE	28.7941	0.9173	0.6269	0.5572	0.9758
	DIP	39.9878	0.9729	0.8616	0.8306	0.9869
	CSC	38.2997	0.9631	0.8118	0.8033	0.9795
	UTV-NLR	**40.8362**	**0.9747**	**0.8718**	**0.8604**	**0.9890**
30°	Rainy	28.7359	0.8488	0.5501	0.4252	0.8641
	TC-LRMC	33.2208	0.9247	0.6277	0.5039	0.9490
	SE	28.9799	0.9250	0.6273	0.5691	0.9757
	DIP	39.2723	0.9695	0.8444	0.8124	0.9839
	CSC	38.0849	0.9629	0.8107	0.8108	0.9790
	UTV-NLR	**40.6011**	**0.9735**	**0.8652**	**0.8551**	**0.9877**
40°	Rainy	28.2611	0.8289	0.5259	0.3776	0.8454
	TC-LRMC	29.5740	0.8912	0.5709	0.4737	0.9060
	SE	29.0138	0.9181	0.6210	0.5713	0.9745
	DIP	39.1306	0.9694	0.8521	0.8286	0.9847
	CSC	38.5752	0.9621	0.8105	0.8261	0.9792
	UTV-NLR	**40.3750**	**0.9722**	**0.8635**	**0.8672**	**0.9880**
Carphone						
γ	Methods	MPSNR	MSSIM	MUQI	MVIF	MFSIM
20°	Rainy	28.0146	0.8705	0.6861	0.5057	0.8887
	TC-LRMC	29.4307	0.9151	0.7332	0.4407	0.9222
	SE	28.9571	0.8681	0.7754	0.5061	0.9502
	DIP	32.8527	0.9592	0.8203	0.5983	0.9599
	CSC	30.2795	0.9477	0.8081	0.5369	0.9422
	UTV-NLR	**33.5989**	**0.9666**	**0.8426**	**0.6160**	**0.9687**
30°	Rainy	27.9244	0.8708	0.6854	0.5191	0.8864
	TC-LRMC	29.6957	0.9254	0.7510	0.4757	0.9294
	SE	28.8585	0.8595	0.7801	0.5155	0.9502
	DIP	32.3282	0.9528	0.8047	0.5978	0.9535
	CSC	30.2510	0.9474	0.8101	0.5451	0.9417
	UTV-NLR	**33.0900**	**0.9636**	**0.8364**	**0.6189**	**0.9658**
40°	Rainy	28.0196	0.8707	0.6858	0.5391	0.8869
	TC-LRMC	28.5237	0.9044	0.7111	0.5335	0.9048
	SE	28.8612	0.8597	0.7802	0.5234	0.9502
	DIP	32.2099	0.9537	**0.8083**	0.6021	0.9538
	CSC	30.2559	0.9469	0.8077	0.5528	0.9418
	UTV-NLR	**32.7298**	**0.9543**	0.8042	**0.6181**	**0.9576**

and 40° on two videos named “bridge” and “carphone”. Fig. 8 and Table 3 respectively shows the visual results and the quantitative evaluation of all compared methods. We can observe that UTV-NLR can remove the rain streaks successfully on video “bridge” with different rain directions. Specifically, Table 3 shows that UTV-NLR outperforms the other methods by more than 1.5 dB and achieves the highest values with other metrics in all cases. For the video “carphone”, UTV-NLR performs competitively against the other compared state-of-the-art methods in visual performance. While in Table 3, we can see that the proposed UTV-NLR outperforms the competing methods in most cases, which indicates that UTV-NLR best keeps the texture details of original clean videos. Fig. 8 and Table 3 demonstrate the proposed UTV-NLR can successfully remove highly oblique rain streaks visually and quantitatively. This can be explained from two aspects: (1) oblique rain streaks always have vertical components,so the UTV regularization of $\boldsymbol{\mathcal{R}}_i$ along vertical direction still promotes the deraining performance. (2) We fully adopt the priors of clean videos and rain streaks. Although the effect of UTV regularization of $\boldsymbol{\mathcal{R}}_i$ is weakened, the others can still distinguish the clean videos and rain streaks successfully.

6.3. Discussion of parameters

In our method, the penalty parameter $\beta = 2$ is fixed for all cases. For the other weighted parameters, α_1, α_2, α_3, and α_4 are first roughly selected in the set $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$. Among this set, we choose the settings for α_1, α_2, α_3, and α_4 with the highest PSNR values. Then we fine-tune the best parameters in a small range. By this way, these parameters are finally tuned empirically in the interval [0.1,2] in this paper. Besides β, α_1, α_2, α_3, and α_4, the parameters of the patch size and number of similar patches are also important in our model. In this section, we analyze how the regularization terms and these parameters influence the performance of rain streaks removal task. We test them on the video “running car” with heavy rain.

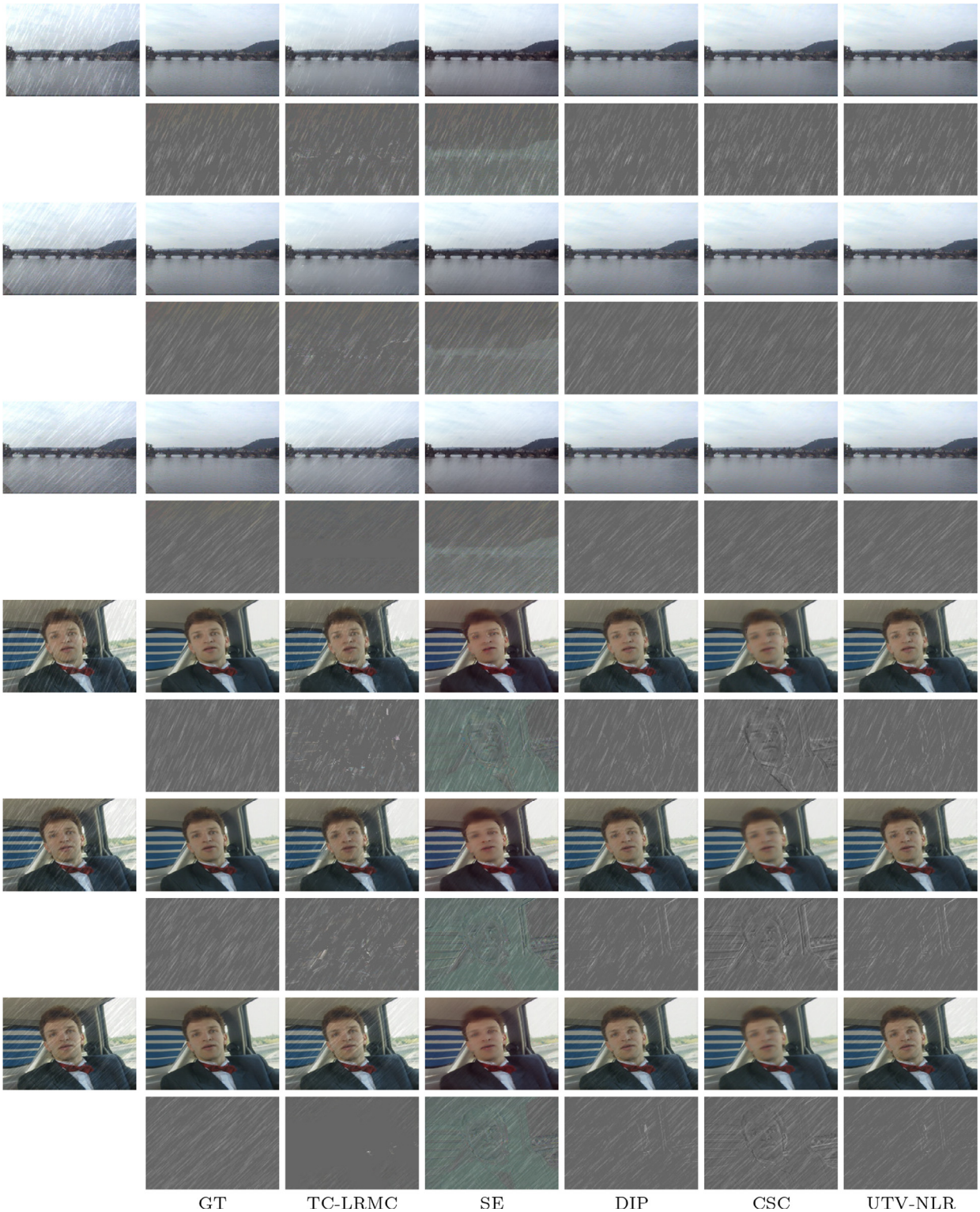

Fig. 8. Rain removal results and the corresponding rain maps with different rain directions. The first column shows the rainy frames. The rain directions are set as $\gamma = 20^\circ$, $\gamma = 30^\circ$, and $\gamma = 40^\circ$ with the corresponding result frames shown from top to bottom for each videos.

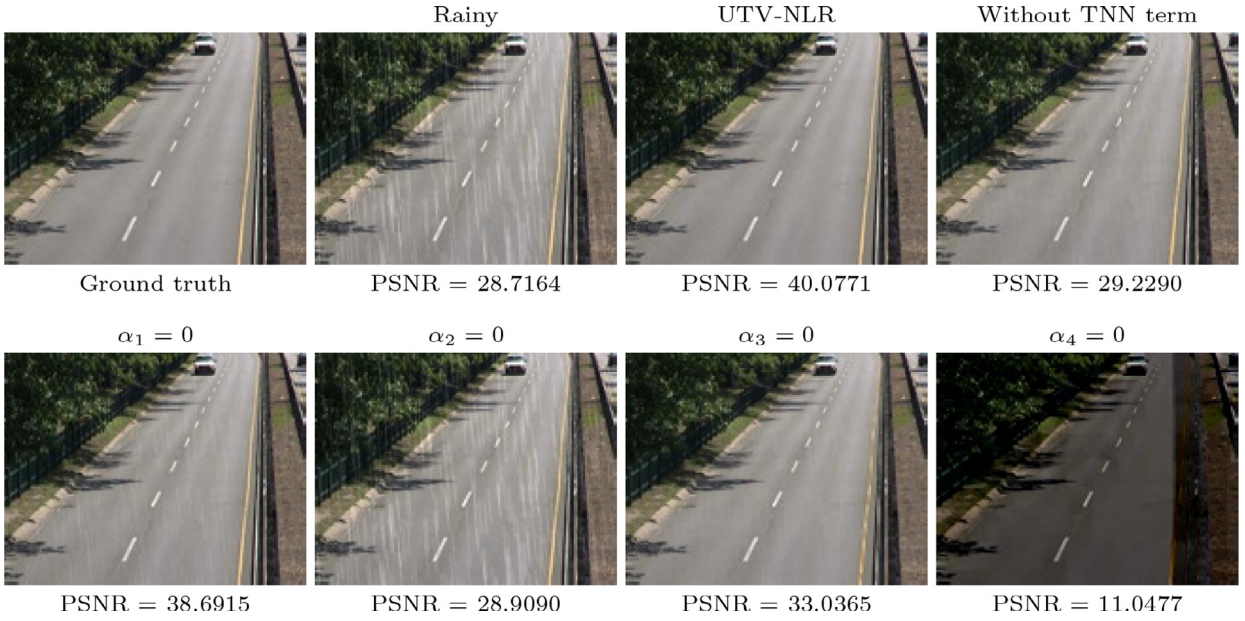

Fig. 9. The results of the proposed UTV-NLR obtained without different regularization terms.

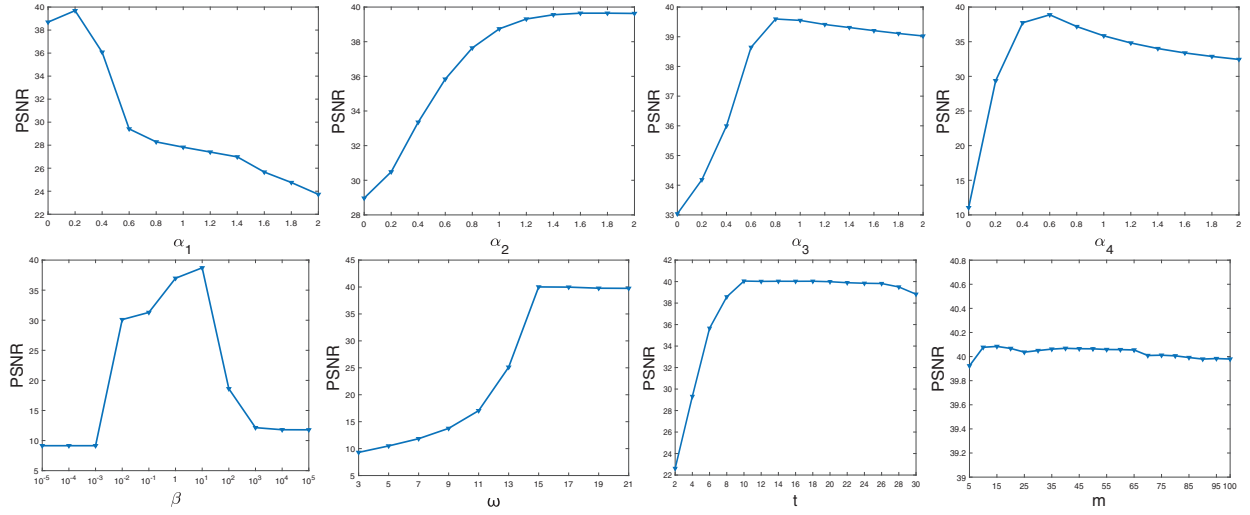

Fig. 10. Analysis of the parameters.

We first perform our method with discarding each regularization terms in equation (5) to test the effect of these terms. Fig. 9 shows the visual performance and the corresponding PSNR values. We can see that performing UTV-NLR without the TNN term $\|\mathcal{V}\|_*$, $\|D_x\mathcal{V}_i\|_1$ ($\alpha_1 = 0$), or $\|D_y\mathcal{R}_i\|_1$ ($\alpha_3 = 0$) leads to notable rain streaks left in the recovered video, which demonstrates that these terms can well characterize the clean video or rain streaks and benefit the final deraining results. When setting $\alpha_2 = 0$, i.e., without the temporal UTV regularization term $\|D_t\mathcal{V}_i\|_1$, the proposed UTV-NLR fails to remove the rain streaks and achieves about only 28 dB. Performing UTV-NLR with $\alpha_4 = 0$, i.e., without term $\|\mathcal{R}_i\|_1$, we can see that the result video is not well recovered and has very low intensity. Fig. 9 demonstrates that all the regularization terms make great contributions to the final deraining results. Moreover, it can be observed that $\|\mathcal{V}_i\|_*$, $\|D_x\mathcal{V}_i\|_1$, and $\|D_y\mathcal{R}_i\|_1$ significantly contribute to preserving the detailed textures of clean videos. $\|D_t\mathcal{V}_i\|_1$ and $\|\mathcal{R}_i\|_1$ are the most important terms for removing rain streaks and recovering clean videos, respectively.

We then test how the parameters influence the proposed algorithm. Fig. 10 presents the quantitative results with varying parameters. We can observe that when $\alpha_1 \leq 0.4$, UTV-NLR achieves high and stable PSNR values. While for α_2, α_3, and α_4, when they become larger than 0.8, the corresponding PSNR curves tend to be stable. For the penalty parameter β, the stable interval is about $10^{-1} < \beta < 10^1$. In this paper, we set $\beta = 2$ and empirically tune α_1, α_2, α_3, and α_4 in the interval [0.1, 2] for all videos. Note that the size of the spatial–temporal patch is $\omega \times \omega \times t$ and the size of similar-patch-based tensor

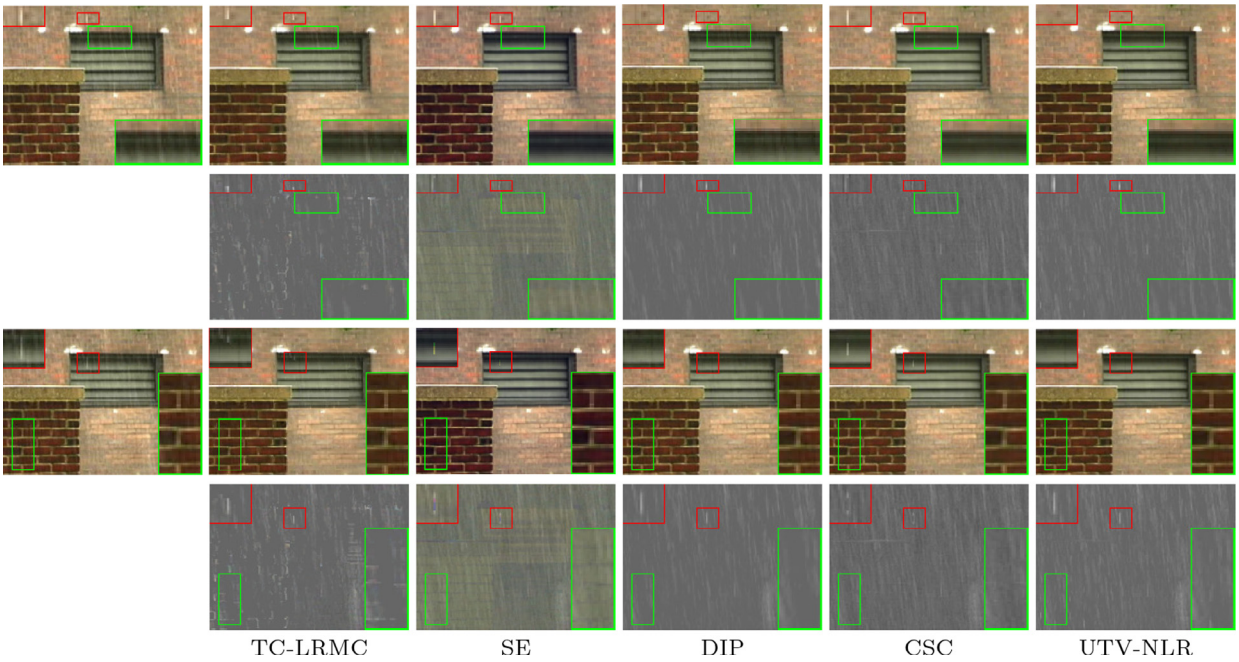

Fig. 11. Rain removal results on a real video "wall" and the corresponding rain streaks. The first column shows the input rainy frames.

$\mathcal{V}_i$ is $\omega^2 \times t \times m$, we also test the parameters of spatial size ω, temporal size t, and the number of similar patches m. Fig. 10 shows the curves of PSNR values with respect to ω, t, and m in the second row. We can see that the PSNR values are very low with small ω or t, but increase sharply when ω or t become larger than 15. The reason is that too small patches can not present the image structures and will further lead to unfaithful matches of similar patches. In the meanwhile, too large ω or t will reduce the self-similarities of the spatial–temporal patches and affect the rain removal performance, which can also be observed from the slight decline of PSNR curves with respect to ω and t in Fig. 10. For the parameter of similar patch number m, the PSNR curve is much stabler than those of ω and t. But the PSNR values slightly decrease when m become bigger. The reason may be that selecting too many similar spatial–temporal patches will reduce the low-rankness of $\mathcal{V}_i$ and then degrade the performance of rain streaks removal task. In this paper, We set $\omega = 15, t = 10, m = 30$, i.e., the size of spatial–temporal patch is fixed as $15 \times 15 \times 10$, Fig. 10 demonstrates that our method is robust to most parameters within a certain range.

6.4. Experiments on real data

In this section, we test the methods on four real rainy videos. Since these videos contain the real rain streaks and no ground truths are known, we mainly evaluate the visual performance of the compared methods.

Fig. 11 shows the results on a real rainy video named "wall" with the size $144 \times 184 \times 3 \times 30$. Two frames are selected and shown from top to bottom. All methods succeed to removal the rain streaks and recover the rain-free video. While in the demarcated areas, we can see that UTV-NLR recovers the best texture and detail information of the rain-free video. Additionally, the rain steak maps also demonstrate that UTV-NLR removes the rain streak most completely.

Fig. 12 shows the rain removal results and the corresponding rain streak maps on a real video extracted from the movie "the Matrix". This video has the size of $83 \times 210 \times 3 \times 30$. Three frames are selected from the results of all the methods and shown from top to bottom. Fig. 12 demonstrates that TC-LRMC cannot remove the rain completely and some recovered rain-free frames are clearly blurred or corrupted. The reason may be that when the rain is very heavy, many clean pixels are severely affected and detected as the rain streaks by TC-LRMC. So they are difficult to be recovered by the subsequent matrix completion processing in TC-LRMC. The recovered clean video of SE and CSC have smooth backgrounds but the dynamic regions still contain a lot of rain streaks. This may be caused by the fact that SE and CSC aim to decompose the clean video into the static background and moving objects, but the background of this video is dynamic. The rain-free frames of DIP still contain clear marks of rain streaks in the demarcated areas and blur the scenes in some frames. This is because the low-rank assumption of the whole clean video makes the pixels of dynamic scenes affected by their neighbors in adjacent frames. Our result achieves the best visual performance. Since we exploit the NSS property to characterize the spatial–temporal redundancy of clean videos, our method handles the dynamic scenes well and recovers the clean video with the best visual effects.

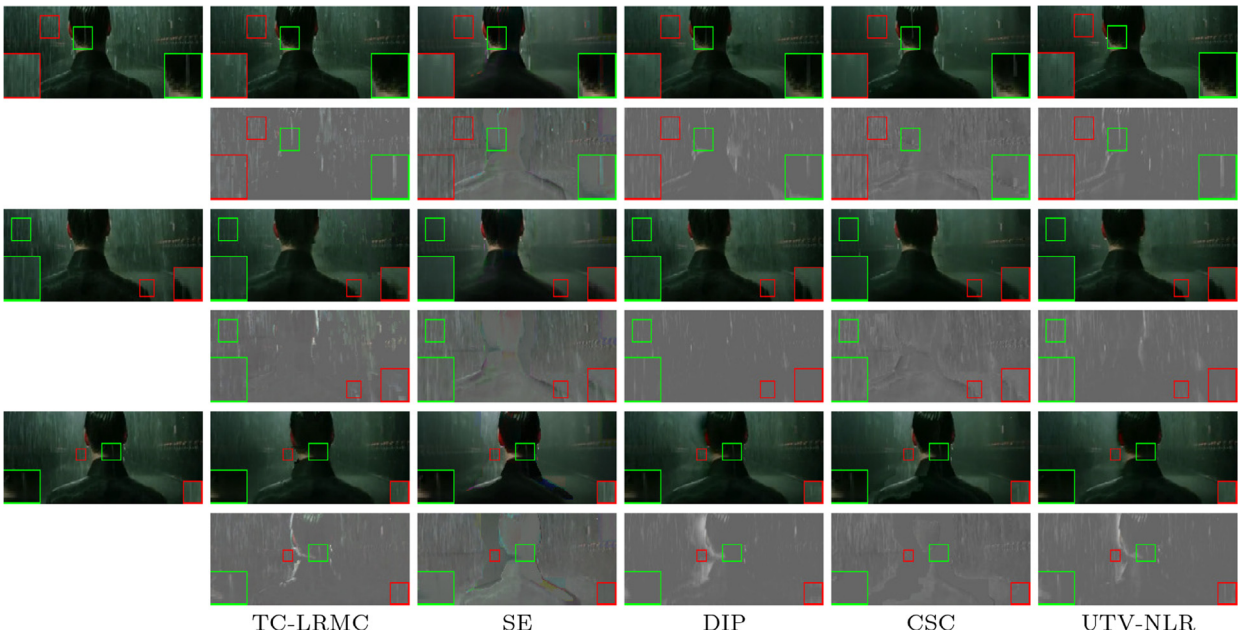

Fig. 12. Rain removal results on the real video "the Matrix". The first column shows the input rainy frames.

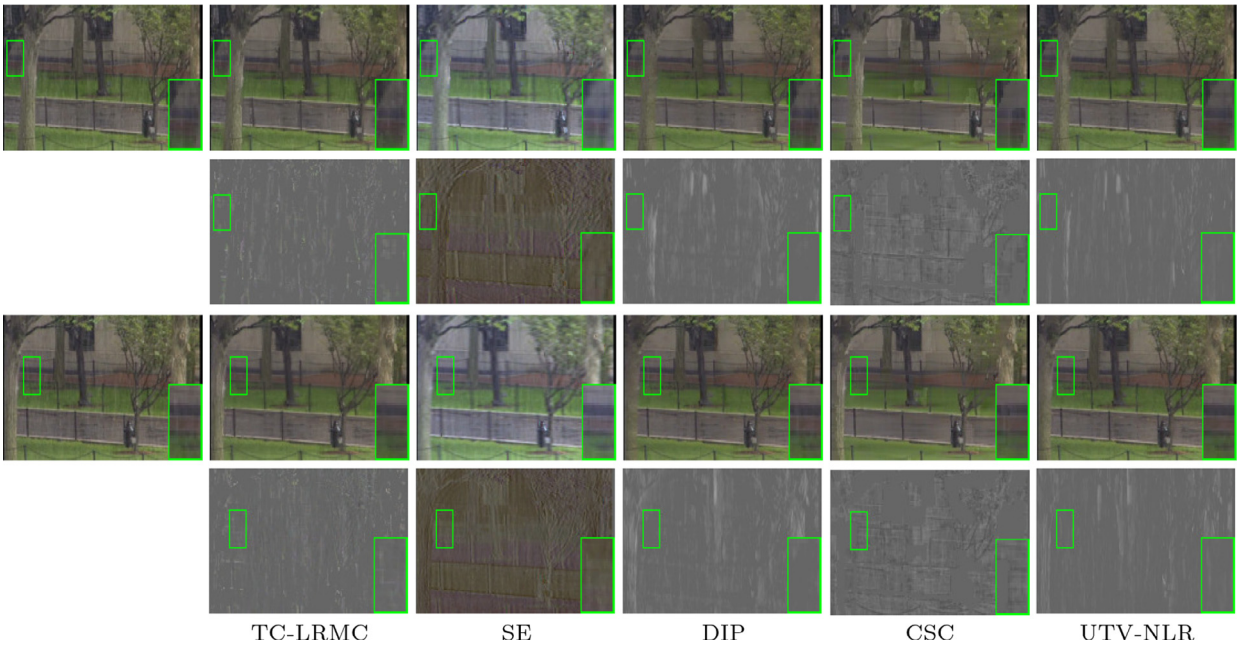

Fig. 13. Rain removal results and the estimated rain streaks with the real video "trees". The first column shows the input rainy frames.

In Fig. 13, we evaluate the methods on a real rainy video named "trees" with the size $116 \times 180 \times 3 \times 30$. This video was taken by the moving camera with dynamic scenes. Two frames are selected for visual comparison in the top and bottom rows in Fig. 13. It demonstrates that the recovered rain-free video of SE has blurred scenes and distorted colors. TC-LRMC, DIP, and CSC succeed to remove most of the rain, however, some shallow marks of rain streaks are still left in demarcated areas. The proposed UTV-NLR slightly outperforms the other compared methods and achieves a cleaner rain-free video.

Fig. 14 shows rain removal results and the estimated rain maps on a video named "sitting man" with the size $350 \times 720 \times 3 \times 100$. This video is taken by a moving camera. We can observe that DIP cannot remove the rain streaks completely in the first frame. TC-LRMC achieves better performance in the first frame but fails to remove the conspicuous rain streaks in the second frame. The result of CSC has notable rains streaks left in both the first and second frames. Due to

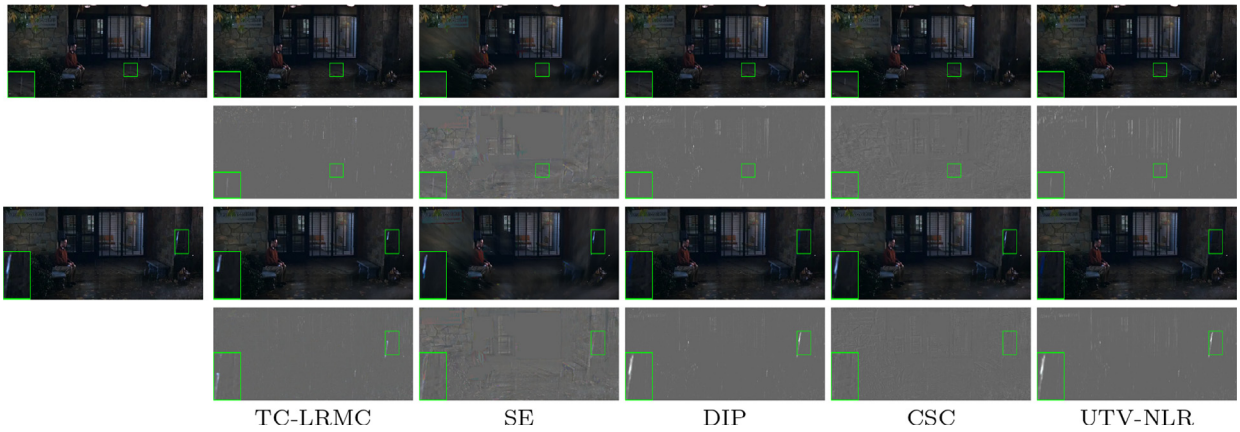

Fig. 14. Rain removal results and the corresponding estimated rain streak maps on the real video "sitting man". The first column shows the input rainy frames.

the moving camera, SE removes the rain streaks but gets a severely ambiguous background. Our method not only removes the rain streaks but also recovers the detailed video information well in both frames.

7. Conclusion

In this paper, we proposed a tensor-based video rain streaks removal method using the UTV and nonlocal low-rank regularization. We exploited the NSS to characterize the spatial–temporal correlation of clean videos and used the TNN to enhance the low-rankness of the tensors constructed by similar spatial–temporal patches of clean videos. We also considered the piecewise smoothness and the temporal continuity of clean videos and employed the UTV regularization to enhance the smoothness and continuity. Since rain streaks are sparse and smooth along rain direction, we utilized the l_1 norm and the UTV term to enhance the sparsity and directional smoothness. An efficient ADMM algorithm was developed to tackle the proposed model. Experimental results demonstrate that our method outperforms the state-of-the-art methods quantitatively and visually.

Even though our method can handle the oblique rain streaks removal task, the vertical UTV terms may still affect the deraining performance when dealing with oblique rain streaks. In [50], Wang et al. proposed to use the directional total variation (DTV) to remove rain streaks with arbitrary directions from a single-image. This motivated us to employ the DTV term into the video deraining task in our future work.

Acknowledgments

This research is supported by NSFC (61772003, 61876203, 61702083), Science Strength Promotion Programme of UESTC, and the Fundamental Research Funds for the Central Universities (ZYGX2016KYQD142).

References

[1] V. Kantorov, I. Laptev, Efficient feature extraction, encoding, and classification for action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2593–2600.
[2] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 564–577.
[3] X. Zhang, C. Zhu, S. Wang, Y. Liu, M. Ye, A Bayesian approach to camouflaged moving object detection, IEEE Trans. Circ. Syst. Video Technol. 27 (9) (2017) 2001–2013.
[4] K. Garg, S.K. Nayar, Vision and rain, Int. J. Comput. Vis. 75 (1) (2007) 3–27.
[5] M.S. Shehata, J. Cai, W.M. Badawy, T.W. Burr, M.S. Pervez, R.J. Johannesson, A. Radmanesh, Video-based automatic incident detection for smart roads: the outdoor environmental challenges regarding false alarms, IEEE Trans. Intell. Transp. Syst. 9 (2) (2008) 349–360.
[6] K. Garg, S.K. Nayar, Detection and removal of rain from videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition., 1, 2004, pp. 528–535.
[7] K. Garg, S.K. Nayar, When does a camera see rain? in: Proceedings of the IEEE International Conference on Computer Vision, 2, 2005, pp. 1067–1074.
[8] A.K. Tripathi, S. Mukhopadhyay, A probabilistic approach for detection and removal of rain from videos, IETE J. Res 57 (1) (2011) 82–91.
[9] J.H. Kim, J.Y. Sim, C.S. Kim, Video deraining and desnowing using temporal correlation and low-rank matrix completion, IEEE Trans. Image Process. 24 (9) (2015) 2658–2670.
[10] T.-X. Jiang, T.-Z. Huang, X.-L. Zhao, L.-J. Deng, Y. Wang, Fastderain: a novel video rain streak removal method using directional gradient priors, IEEE Trans. Image Process. 28 (4) (2019) 2089–2102.
[11] Y.-T. Wang, X.-L. Zhao, T.-X. Jiang, L.-J. Deng, T.-H. Ma, Y.-T. Zhang, T.-Z. Huang, A total variation and group sparsity based tensor optimization model for video rain streak removal, Signal Process. Image Commun. 73 (2019) 96–108.
[12] P. Liu, J. Xu, J. Liu, X. Tang, Pixel based temporal analysis using chromatic property for removing rain from videos, Comput. Inf. Sci. 2 (2009) 53–60.
[13] A. Tripathi, S. Mukhopadhyay, Video post processing: low-latency spatiotemporal approach for detection and removal of rain, IET Image Process. 6 (2012) 181–196.
[14] Y.-L. Chen, C.-T. 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.

[15] A.E. Abdel-Hakim, A novel approach for rain removal from videos using low-rank recovery, in: Proceedings of the International Conference on Intelligent Systems, Modelling and Simulation, 2014, pp. 351–356.
[16] T.-X. Jiang, T.-Z. Huang, X.-L. Zhao, L.-J. Deng, Y. Wang, A novel tensor-based video rain streaks removal approach via utilizing discriminatively intrinsic priors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2818–2827.
[17] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang, B. Zhang, Decomposable nonlocal tensor dictionary learning for multispectral image denoising, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2949–2956.
[18] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 208–220.
[19] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, S. Yan, Tensor robust principal component analysis: exact recovery of corrupted low-rank tensors via convex optimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5249–5257.
[20] T. Kolda, B. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[21] Z. Zhang, S. Aeron, Exact tensor completion using t-SVD, IEEE Trans. Signal Process. 65 (6) (2017) 1511–1526.
[22] L.W. Kang, C.W. Lin, Y.H. Fu, Automatic single-image-based rain streaks removal via image decomposition, IEEE Trans. Image Process. 21 (4) (2012) 1742–1755.
[23] Y. Luo, Y. Xu, H. Ji, Removing rain from a single image via discriminative sparse coding, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3397–3405.
[24] Y. Li, R.T. Tan, X. Guo, J. Lu, M.S. Brown, Rain streak removal using layer priors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2736–2744.
[25] L.-J. Deng, T.-Z. Huang, X.-L. Zhao, T.-X. Jiang, A directional global sparse model for single image rain removal, Appl. Math. Model 59 (2018) 662–679.
[26] D. Eigen, D. Krishnan, R. Fergus, Restoring an image taken through a window covered with dirt or rain, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 633–640.
[27] X. Fu, J. Huang, X. Ding, Y. Liao, J. Paisley, Clearing the skies: a deep network architecture for single-image rain removal, IEEE Trans. Image Process. 26 (6) (2017) 2944–2956.
[28] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, J. 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. 1715–1723.
[29] W. Yang, R.T. Tan, J. Feng, J. Liu, Z. Guo, S. Yan, Deep joint rain detection and removal from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1685–1694.
[30] P.C. Barnum, S. Narasimhan, T. Kanade, Analysis of rain and snow in frequency space, Int. J. Comput. Vis. 86 (2) (2009) 256.
[31] J. Bossu, N. Hautiéré, J.-P. Tarel, Rain or snow detection in image sequences through use of a histogram of orientation of streaks, Int. J. Comput. Vis. 93 (3) (2011) 348–367.
[32] V. Santhaseelan, V.K. Asari, Utilizing local phase information to remove rain from video, Int. J. Comput. Vis. 112 (1) (2015) 71–89.
[33] W. Ren, J. Tian, Z. Han, A. Chan, Y. Tang, Video desnowing and deraining based on matrix decomposition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 00, 2017, pp. 2838–2847.
[34] J. Chen, C.-H. Tan, J. Hou, L.-P. Chau, H. Li, Robust video content alignment and compensation for rain removal in a cnn framework, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[35] J. Liu, W. Yang, S. Yang, Z. Guo, Erase or fill? deep joint recurrent rain removal and reconstruction in videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[36] W. Wei, L. Yi, Q. Xie, Q. Zhao, D. Meng, Z. Xu, Should we encode rain streaks in video as deterministic or stochastic? in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2535–2544.
[37] M. Li, Q. Xie, Q. Zhao, W. Wei, S. Gu, J. Tao, D. Meng, Video rain streak removal by multiscale convolutional sparse coding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[38] X.-T. Li, X.-L. Zhao, T.-X. Jiang, Y.-B. Zheng, T.-Y. Ji, T.-Z. Huang, Low-rank tensor completion via combined non-local self-similarity and low-rank regularization, Neurocomputing 367 (2019) 1–12.
[39] T. Goldstein, S. Osher, The split bregman method for l1-regularized problems, SIAM J. Imaging Sci. 2 (2) (2009) 323–343.
[40] A. Chambolle, T. Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, J. Math. Imaging Vis. 40 (1) (2011) 120–145.
[41] E. Esser, X. Zhang, T. Chan, A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science, SIAM J. Imaging Sci. 3 (4) (2010) 1015–1046.
[42] Y. Wang, J. Yang, W. Yin, Y. Zhang, A new alternating minimization algorithm for total variation image reconstruction, SIAM J. Imaging Sci. 1 (3) (2008) 248–272.
[43] Z. Zhang, G. Ely, S. Aeron, N. Hao, M. Kilmer, Novel methods for multilinear data completion and de-noising based on tensor-SVD, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3842–3849.
[44] Adding rain to a photo with photoshop, (https://www.photoshopessentials.com/photo-effects/rain/).
[45] S. Starik, M. Werman, Simulation of rain in videos, in: Proceedings of the IEEE International Conference on Computer Vision Texture Workshop, 2, 2003, pp. 406–409.
[46] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[47] Z. Wang, A.C. Bovik, A universal image quality index, IEEE Signal Process. Lett. 9 (3) (2002) 81–84.
[48] H.R. Sheikh, A.C. Bovik, Image information and visual quality, IEEE Trans. Image Process. 15 (2) (2006) 430–444.
[49] L. Zhang, L. Zhang, X. Mou, D. Zhang, Fsim: a feature similarity index for image quality assessment, IEEE Trans. Image Process. 20 (8) (2011) 2378–2386.
[50] Y. Wang, T.-Z. Huang, X.-L. Zhao, L.-J. Deng, T.-X. Jiang, Rain streaks removal for single image via directional total variation regularization, in: Proceedings of the IEEE International Conference on Image Processing, 2019, pp. 2801–2805.