Semi-Supervised Video Deraining with Dynamical Rain Generator

Zongsheng Yue\textsuperscript{1}, Jianwen Xie\textsuperscript{2}, Qian Zhao\textsuperscript{1}, Deyu Meng\textsuperscript{1,3,*}
\textsuperscript{1}Xi’an Jiaotong University, Xi’an, China
\textsuperscript{2}Cognitive Computing Lab, Baidu Research, Bellevue, USA
\textsuperscript{3}The Macau University of Science and Technology, Macau, China

zsyzam@gmail.com, jianwen@ucla.edu, timmy.zhoa@gmail.com, dymeng@mail.xjtu.edu.cn

https://github.com/zsyOAOA/S2VD

Abstract

While deep learning (DL)-based video deraining methods have achieved significant successes in recent years, they still have two major drawbacks. Firstly, most of them are insufficient to model the characteristics of rain layers contained in rainy videos. In fact, the rain layers exhibit strong visual properties (e.g., direction, scale, and thickness) in spatial dimension and causal properties (e.g., velocity and acceleration) in temporal dimension, and thus can be modeled by the spatial-temporal process in statistics. Secondly, current DL-based methods rely heavily on the labeled training data, whose rain layers are synthetic, thus leading to a deviation from real data. Such a gap between synthetic and real data sets results in poor performance when applying them to real scenarios. To address these issues, this paper proposes a new semi-supervised video deraining method, in which a dynamical rain generator is employed to fit the rain layer for the sake of better depicting its intrinsic characteristics. Specifically, the dynamical generator consists of one emission model and one transition model to simultaneously encode the spatial appearance and temporal dynamics of rain streaks, respectively, both of which are parameterized by deep neural networks (DNNs). Furthermore, different prior formats are designed for the labeled synthetic and unlabeled real data so as to fully exploit their underlying common knowledge. Last but not least, we design a Monte Carlo-based EM algorithm to learn the model. Extensive experiments are conducted to verify the superiority of the proposed semi-supervised deraining model.

1. Introduction

Rain is a very common bad weather that exists in many videos. The appearance of rain not only negatively affects the visual quality of the video, but also seriously deterio-rates the performance of subsequent video processing algorithms, e.g., semantic segmentation \cite{38}, object detection \cite{9}, and autonomous driving \cite{7}. Thus, as a necessary video pre-processing step, video deraining has attracted much attentions from the computer vision community.

As an ill-posed inverse problem raised by Garg and Nayar \cite{15}, various methods have been proposed to handle the video deraining task \cite{47}. Most of the traditional methods focus on exploiting rational prior knowledge for the background or rain layers so as to obtain a proper separation between them. For example, low-rankness \cite{23, 24, 53} is widely used to encode the temporal correlations of background video. As for rain streaks, many visual characteristics, such as photometric appearance \cite{16}, geometrical features \cite{41}, chromatic consistency \cite{36}, local structure correlations \cite{8} and multi-scale convolutional sparse coding \cite{31}, have been explored in the past few years. Different from these deterministic assumptions for rain streaks, Wei et al. \cite{53} firstly regard them as random variables, and use Gaussian mixture model (GMM) to fit them. Although substantiated to be effective in some ideal scenarios, these traditional methods are mainly limited by the subjective manually-designed prior knowledge and huge computation burden.

Recently, owning to the powerful nonlinear fitting capability of DNNs, DL-based methods facilitate significant improvements for the video deraining task. The core idea of this methodology is to directly train a derainer parameterized by DNNs based on synthetic rainy/clean video pairs in an end-to-end manner. Most of these methods leverage different technologies, e.g., superpixel alignment \cite{6}, dual-level flow \cite{56} and self-learning \cite{58}, to extract clean backgrounds from rainy videos. In addition, Liu et al. \cite{34, 35} design a recurrent network to jointly perform both the rain degradation classification and rain removal tasks.

Even though these DL-based methods have achieved impressive deraining results on some synthetic benchmarks, there still exists large room to further increase the perfor-
presents several typical frames of synthetic and real rainy images in NTURain data set. (a1)-(c1): synthetic rainy images, (a2)-(c2): real rainy images.

Figure 1. The comparison of typical synthetic and real rainy images in NTURain data set. (a1)-(c1): synthetic rainy images, (a2)-(c2): real rainy images.

formance and the generalization capability in real applications. On one hand, most of these methods make efforts to depict the background, but neglect to model the intrinsic characteristics of the rain layers. In fact, the rain layer in video, which is an image sequence of rain streaks, can be represented by a spatial-temporal process. Specifically, the randomly scattered rain streaks in each time frame are characterized with evident visual properties (e.g., direction, scale, and thickness) in the spatial dimension, and the rain layers in different time frames correspond to a continuous time series along the temporal dimension, showing the causal properties (e.g., velocity and acceleration) of the rain dynamics. Therefore, elaborately representing and exploiting these intrinsic physical properties underlying the rain layers in video data is expected to facilitate the rain removal task.

On the other hand, it is well known that the performance of DL-based methods heavily relies on a large amount of pre-collected training data, i.e., rainy/clean video pairs. In fact, due to the high labor cost to obtain such video pairs in real scenes, most of current methods have to use synthetic ones, which are manually simulated based on the photo-realistic rendering technique [17] or professional photography and human supervision [50]. Fig.1 presents several typical frames of synthetic and real rainy images in NTURain [6] data set, which is widely used as a benchmark for current video deraining methods. It can be easily seen that the rain patterns in synthetic and real rainy images are obviously different, and the real ones contain more complex and diverse rain types. Because of such a deviation between synthetic and real data sets, the performances of these DL-based methods deteriorate seriously in the real cases. To deal with the generic video deraining task, it is thus critical to build a reasonable semi-supervised learning framework that sufficiently exploits the common knowledge in the labeled synthetic and unlabeled real data.

To address these issues, in this paper we propose a semi-supervised video deraining method, in which a dynamical rain generator is adopted to mimic the generation process of the rain layers in video, hopefully better capturing the intrinsic knowledge simultaneously from the spatial and temporal dimensions. Besides, the real rainy videos are taken into consideration in our model as unlabeled data, in order to achieve more robust deraining results. In summary, the contributions of this work are as follows:

Firstly, we propose a new probabilistic video deraining method, in which a dynamical rain generator, consisting of a transition model and an emission model, is employed to fit the rain layers in videos. Specifically, the transition model is used to represent the dynamics of rains in a low-dimensional state space, while the emission model seeks to generate the observed rain streaks in the image space from the state space. To increase the capacities of such a dynamical rain generator, both the transition and emission models are parameterized by DNNs. Secondly, a semi-supervised learning mechanism is designed by constructing different prior models for labeled synthetic data and unlabeled real data. Specifically, for the labeled synthetic data, the corresponding ground truth rain-free videos are included into an elaborate prior distribution as a strong constraint. As for the unlabeled real data, we introduce the 3-D Markov Random Field (MRF) to model the temporal consistencies and correlations of the underlying backgrounds. Thirdly, a Monte Carlo-based EM algorithm is designed to learn the model. In the expectation step, the posterior of the latent variables is intractable due to the usages of DNNs to parameterize the generator and derainer, thus the Langevin dynamics is adopted to approximate the expectation.

2. Related Work

In this section, we give a short recap for the developments on the video/image deraining methods.

2.1. Video Deraining Methods

To the best of our knowledge, Garg and Nayar [15] firstly proposed the problem of video deraining, and developed a rain detector based on the photometric appearance of rain. Later, they further explored the relationships between rain effects and some camera parameters [16–18].

Inspired by these seminal works, various video deraining methods have been proposed in the past few years, focusing on seeking more reasonable prior knowledge for the rain or background. For example, both the chromatic properties [36, 64] and shape characteristics [2, 3] of rain in the time domain were employed to identify and remove the rain layers from the captured rainy videos, while the regular visual effects of rain in the global frequency space were also exploited by [1]. Besides, Santhaseelan and Asari [43] employed local phase congruency to detect rain based on chromatic constraints. Notably, Wei et al. [53] firstly regarded rain streaks as random variables and fitted them by GMM. In addition, matrix/tensor factorization technologies were
also very popular in the field of video deraining, which were typically used to encode the correlations of background video along the temporal dimension, e.g., [8, 23, 24, 27, 41].

In recent years, DL-based methods represent a new trend along this research line. In [31], Li et al. employed the multi-scale convolutional sparse coding to encode the repetitive local patterns under different scales of rain streaks. Chen et al. [6] proposed to decompose the scene into superpixels and then align the scene content based on the superpixel segmentation result, and finally a CNN was used to compensate the lost details and add normal textures to the deraining result. In [35], Liu et al. designed a recurrent neural network to jointly perform the rain degradation classification and rain removal tasks. And in [34], a hybrid rain model was proposed to model both rain streaks and occlusions. Besides, Yang et al. [56] also built a two-stage recurrent networks that utilize dual-level regularizations toward video deraining. Very recently, Yang et al. [58] proposed a self-learning manner for this task by taking both temporal correlations and consistencies into consideration.

While DL-based methods have achieved impressive performance on some synthetic benchmarks, they are still very hard to be applied to the real applications due to the large gap between the used synthetic data and the real data. Therefore, in order to increase the generalization capacity of the deraining model in the real tasks, it is crucial to design a semi-supervised learning framework that makes use of the information in both the labeled synthetic data and the unlabeled real one. This paper mainly focuses on this issue.

2.2. Single Image Deraining Methods

For literature comprehensiveness, we also briefly review the single image deraining methods. The single image deraining method can be roughly divided into two categories, i.e., model-based methods and DL-based methods. Most of the model-based methods formulate the deraining task as a decomposition problem of the rain and background layers, and various technologies have been employed to deal with it, such as morphological component analysis [25], non-local means filter [26], and sparse coding [5, 37]. Besides, methods built on prior knowledge of rain and background are also explored in this field, mainly including sparsity and low-rankness [4, 19, 60], narrow directions of rain and the similarities of rain patches [66], and GMM [33].

The earliest DL-based method was proposed by Fu et al. [12, 13], in which CNNs were adopted to remove rains from the high frequency parts of rainy images. Led by these two works, DL-based methods began to dominate the research in this field. Many effective and advanced network architectures [14, 21, 30, 32, 40, 50] were put forward in recent years. And some works attempted to jointly handle the rain removal task with other related tasks, e.g., rain detection [57], rain density estimation [61], so as to obtain better deraining performance. Besides, some useful priors, e.g., multi-scale [22, 59, 65], convolutional sparse coding [48] and bilevel layer prior [39], were also embedded into the DL-based methods to sufficiently mine the potentials of DNNs. Different from the above methods, Zhang et al. [62] and Wang et al. [46] both introduced the adversarial learning scheme to enhance the fidelity of the derained images, and Wei et al. [52] proposed a semi-supervised deraining model that can be better generalized to real tasks.

In general, single image deraining methods can be directly used in the video deraining task by treating each video as a bunch of independent images. However, ignoring the abundant temporal information contained in videos will lead to an unsatisfied performance. Thus it is necessary to design a reasonable deraining model specific for video data.

3. Semi-Supervised Video Deraining Model

Given a labeled data set \( D = \{ Y^k, X^k \}_{k=1}^{N_1} \) and an unlabeled data set \( U = \{ Y^k \}_{k=1}^{N_2} \), where \( Y^k \) and \( X^k \) denote the \( k \)-th rainy and clean videos, respectively, we aim to construct a semi-supervised probabilistic model based on them and then design an EM algorithm to learn the model.

3.1. Model Formulation

Let \( Y = \{ Y_t \}_{t=1}^{N} \) denote any rainy video in \( D \) or \( U \), where \( Y_t \in \mathbb{R}^{H \times W} \) is the \( t \)-th image frame. Similar to [31, 32], we decompose the rainy video \( Y \) into three parts, i.e.,

\[
Y = f(Y; W) + R + E, \quad E_{ijt} \sim N(0, \sigma^2),
\]

where \( f(Y; W) \), \( R \) and \( E \) are the recovered rain-free background, rain layer and residual term, respectively, and \( E_{ijt} \) is the element of \( E \) at location \((i, j, t)\). The residual term is assumed to follow a zero-mean Gaussian distribution with variance \( \sigma^2 \). \( f(\cdot; W) \), which is parameterized by DNNs, denotes a function that maps the observed rainy video to the underlying rain-free background, and is called the “de-rainer” in this paper. Next, we consider how to model the derainer parameter \( W \) and rain layer \( R \):

**Modeling of background layer:** As is well known, one general prior knowledge for video data is that the rain-free background video has strong correlations and similarities along spatial and temporal dimensions. Therefore, for any rainy video \( Y \in U \), we encode such a knowledge through the following MRF prior distribution for \( W \):

\[
p(W) \propto \exp \left( -\rho \sum_{i,j,t} \psi^T \gamma \right),
\]

where \( \psi = \begin{bmatrix} f_{i+1,j,t} - f_{ijt} \\ f_{i,j+1,t} - f_{ijt} \\ f_{i,j,t+1} - f_{ijt} \end{bmatrix} \), \( \gamma = [\gamma_1 \gamma_2 \gamma_3]^T \), and \( f_{ijt} \) denotes the element of \( f(Y; W) \) at location \((i, j, t)\). \( \rho \) and \( \gamma \) are both manual hyper-parameters, and the latter represents the
strength of smoothness constraint on the spatial and temporal dimensions. As for the rainy video \( \mathcal{Y} \in \mathcal{D} \), the known rain-free background \( \mathcal{X} \) can be further embedded into Eq. (2) as another strong prior, i.e.,

\[
p(W) \propto \exp \left( -\frac{\|f(\mathcal{Y}; W) - \mathcal{X}\|_2^2}{\varepsilon_0^2} - \rho \sum_{i,j,t} \mathbf{v}_i^T \mathbf{r}_j \right),
\]

where \( \varepsilon_0 \) is a very small hyper-parameter close to zero.

As for the derainer \( f(\cdot; W) \), we adopt a simple network architecture as shown in Fig. 2. Without any special designs, it only contains several 3-D convolution layers and residual blocks [20]. To accelerate the computation, the pixel-unshuffle [63] and pixel-shuffle [45] layers are added to the head and the tail of the network, respectively.

**Modeling of rain layer:** Intuitively, the rain layer is a dynamical sequence, thus we naturally employ the spatio-temporal process [11, 54, 55] in statistics to characterize it. Let’s use \( \mathcal{R}_t \) to denote the \( t \)-th image frame of rain layer sequence \( \mathcal{R} \), and then our dynamical rain generator can be formulated as follows,

\[
s_t = F(s_{t-1}, z_t; \alpha), \quad \mathcal{R}_t = H(s_t; \mathbf{\beta}),
\]

where

\[
z_t \sim \mathcal{N}(0, \mathbf{I}), \quad s_0 \sim \mathcal{N}(0, \mathbf{I}),
\]

\( s_t \) represents the hidden state variable in \( t \)-th frame, and \( z_t \) the noise vector. Specifically, Eq. (4) is the transition model with parameters \( \alpha \) expecting to depict the dynamics of rains over time, and Eq. (5) is the emission model with parameters \( \mathbf{\beta} \) that maps the hidden state space to the space of rain layer. Note that the noise vectors \( \{z_t\}_{t=1}^n \) are independent of each other, and each \( z_t \) encodes the random factors that affect the rains (e.g., wind, camera motion, etc) at time \( t \) in the transition from \( s_{t-1} \) to \( s_t \).

Furthermore, following [55], we can extend the generator to an advanced version for multiple rain videos. Specifically, for the \( i \)-th rain video \( \mathcal{R}^i = \{\mathcal{R}_t^i\}_{t=1}^n \), another vector \( \mathbf{m}^i \sim \mathcal{N}(0, \mathbf{I}) \) is introduced to account for the variation of rain appearances or patterns over different videos, and thus the transition model of Eq. (4) can be reformulated as:

\[
s_t^i = F(s_{t-1}^i, z_t^i; \mathbf{m}^i; \alpha),
\]

where \( \mathbf{m}^i \) is fixed for the \( i \)-th rain video. For notation convenience, we write Eqs. (7) and (5) together as follows:

\[
\mathcal{R}_t^i = G(s_0^i, z^i, \mathbf{m}^i; \mathbf{\theta}),
\]

where \( z^i = \{z_t^i\}_{t=1}^n, \mathbf{\theta} = \{\alpha, \mathbf{\beta}\} \). In practice, we use the extended version of Eq. (8) to simultaneously fit the rain layers in each mini-batch of video data.

To increase the capacity of such a dynamical rain generator, we parameterize both of the transition model and the emission model by DNNs. Following [55], we use a two-layers multi-layer perceptron (MLP) in Fig. 3 (a) for the transition model. As to the emission model, we elaborately design a CNN architecture that takes the state variable \( s_t \) as input and outputs the rain image as shown in Fig. 3 (b), which is mainly inspired by a recent work [49] that uses CNN as a latent variable model to generate rain streaks.

---

**Figure 2.** The network architecture for the derainer \( f(\cdot; W) \). In this figure, all “Conv”s denote the 3-D convolution layer.

**Figure 3.** An illustration of network architectures of the transition model and the emission model in the dynamical rain generator. In this figure, “FC”, “Conv” and “Tanh” denote fully connected, 2-D convolution and hyperbolic tangent layers, respectively. “Pixel-Shuffle” is the sub-pixel layer [45] with a scale factor \( r \).
Remark: The employment of such a dynamical generator to fit the rain layers is one of the main contributions of this work, which directly affects the deraining performance of the entire model. Therefore, it is necessary to validate the capability of the dynamical generator on simulating the rain layers. To prove this point, we pre-collected some rain layer videos synthesized by commercial Adobe After Effects software from YouTube as source videos, and trained the dynamical generator on them. Empirically, we found that the proposed dynamical rain generator is able to sufficiently mimic the given rain layer videos. Due to the page limitation, we put the experiments to the supplementary materials.

3.2. Maximum A Posteriori Estimation

Combining Eqs. (1)-(6), a full probabilistic model is obtained for video deraining. Then our goal turns to maximize the posteriors w.r.t. the model parameters $W$ and $\theta$, i.e.,

$$\max_{W,\theta} \log p(W,\theta | Y) = \log p(\mathcal{Y}|W,\theta) + \log p(W) + \text{const}$$

where $p(\mathcal{Y}|W,\theta)$ is the likelihood of the rainy video $Y$. According to Eqs. (1) and (8), it can be written as:

$$p(\mathcal{Y}|W,\theta) = \int p(\mathcal{Y}|W,\theta,z)p(z)\,dz = \int \mathcal{N}(f(\mathcal{Y};W) + G(s_0, z; \theta), \sigma^2 \mathcal{I}) p(z)\,dz.$$

Finally, we directly optimize the problem of Eq. (9) on the whole labeled and unlabeled data sets, i.e.,

$$\max_{W,\theta} \sum_{\mathcal{Y}^l \in \mathcal{D}} \mathcal{L}(\mathcal{Y}^l; W, \theta) + \sum_{\mathcal{Y}^u \in \mathcal{U}} \mathcal{L}(\mathcal{Y}^u; W, \theta).$$

The insight behind Eq. (10) is to learn a general mapping from rainy videos to clean ones, based on large amount of data samples in $\mathcal{D}$ and $\mathcal{U}$, which is expected to obtain a more efficient and robust derainer than that in traditional inference paradigm implementing on single video.

Most notably, if only considering labeled data set, our method naturally degenerates into a supervised deraining model. However, involving the unlabeled real data can increase the generalization capacity of the model such that it can be applied to the real cases as shown in the ablation studies in Sec. 4.2.2.

3.3. Inference and Learning Algorithm

For notation brevity, we only consider one data sample $\mathcal{Y}$ in this part. Inspired by the technology of alternative back-propagation through time [55], a Monte Carlo-based EM [10] algorithm is designed to maximize $\mathcal{L}(\mathcal{Y}; W, \theta)$, in which the expectation step samples the latent variable $z$

$$\mathcal{Q}(W, \theta) = \int p_{\text{old}}(z|\mathcal{Y}) \log p(\mathcal{Y}, z|W, \theta)\,dz + \log p(W) \approx \log p(\mathcal{Y}, z|W, \theta) + \log p(W).$$

Equivalently, Eq. (13) can be further rewritten as the follow-

### Algorithm 1 Inference and learning procedure for S2VD

**Input:** training data $\mathcal{D} = \{\mathcal{Y}^l_j, \mathcal{X}^l_j\}_{j=1}^{B_l}$ and $\mathcal{U} = \{\mathcal{Y}^u_j\}_{j=B_l+1}^{B_l+u}$, where $\mathcal{Y}^l_j$ denotes the $j$-th mini-batch data, number of Langevin steps $l$.

**Output:** the derainer parameters $W$.

1. Initialize $W$ and $\theta^{0}$, $z^{0}$, $j = 1, 2, \cdots, B_l + B_u$.
2. while not converged do
3. for $j = 1, 2, \cdots, B_l + B_u$ do
4. Sample the mini-batch data $(\mathcal{Y}^l_j, \mathcal{X}^l_j)$ or $\mathcal{Y}^u_j$.
5. E-Step: For each data example $\mathcal{Y}^l_j$ in current mini-batch $\mathcal{Y}^l_j$, run $l$ steps of Langevin dynamics to sample $z^l$ following Eq. (12).
6. M-Step: Update $W$ and $\theta^{j}$ by Eq. (15).
7. end for
8. end while

from the posterior distribution $p(z|\mathcal{Y})$, and the maximization step updates the model parameters $W$ and $\theta$ based on the inferred latent variable $z$.

**E-Step:** Let $(W^{\text{old}}, \theta^{\text{old}})$ and $p_{\text{old}}(z|\mathcal{Y})$ denote the current model parameters and the posterior under them, we can sample $z$ from $p_{\text{old}}(z|\mathcal{Y})$ using the Langevin dynamics [29]:

$$z^{(r+1)} = z^{(r)} + \frac{\delta^2}{2} \left[ \frac{\partial}{\partial z} \log p_{\text{old}}(z|\mathcal{Y}) \right]_{z=z^{(r)}} + \delta \xi^{(r)}$$

$$= z^{(r)} - \frac{\delta^2}{2} \left[ \frac{\partial}{\partial z} g(z) \right]_{z=z^{(r)}} + \delta \xi^{(r)}$$

(11)

where we define

$$g(z) = \frac{1}{2\sigma^2} \left\| \mathcal{Y} - f(\mathcal{Y}; W^{\text{old}}) - G(s_0, z; \theta^{\text{old}}) \right\|^2_2 + \frac{1}{2} \|z\|_2,$$

(12)

$\tau$ indexes the time step for Langevin dynamics, $\delta$ denotes the step size, and $\xi^{(r)}$ is the Gaussian white noise, which is used to avoid falling into local modes. A key point to compute Eq. (11) is $\frac{\partial}{\partial z} \log p_{\text{old}}(z|\mathcal{Y}) = \frac{\partial}{\partial z} \log p_{\text{old}}(\mathcal{Y}, z)$, and the right term can be easily calculated.

In practice, for the purpose of avoiding the high computational cost of MCMC, at each learning iteration, Eq. (11) starts from the previous updated results of $z$. As for the initial state vector $s_0$ and the rain variation vector $m$ in Eq. (8), because they are also latent variables in our model, we sample them together with $z$ using the Langevin dynamics.

**M-Step:** Denote the sampled latent variable in E-Step as $\tilde{z}$. M-Step aims to maximize the approximate upper bound w.r.t. $W$ and $\theta$ as follows:

$$\max_{W,\theta} \mathcal{Q}(W, \theta) = \int p_{\text{old}}(z|\mathcal{Y}) \log p(\mathcal{Y}, z|W, \theta)\,dz + \log p(W) \approx \log p(\mathcal{Y}, \tilde{z}|W, \theta) + \log p(W).$$


Table 1. PSNR/SSIM results of different methods on the synthetic testing data set of NTURain. The best and second best results are highlighted in red and blue, respectively.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>29.71</td>
<td>0.9149</td>
<td>27.13</td>
<td>0.9007</td>
<td>29.29</td>
<td>0.9159</td>
<td>31.79</td>
<td>0.9481</td>
<td>32.13</td>
<td>0.9511</td>
</tr>
<tr>
<td>a2</td>
<td>29.30</td>
<td>0.9284</td>
<td>28.84</td>
<td>0.9224</td>
<td>30.21</td>
<td>0.9245</td>
<td>30.34</td>
<td>0.9360</td>
<td>30.41</td>
<td>0.9375</td>
</tr>
<tr>
<td>a3</td>
<td>29.68</td>
<td>0.9294</td>
<td>26.73</td>
<td>0.8942</td>
<td>29.94</td>
<td>0.9039</td>
<td>30.70</td>
<td>0.9301</td>
<td>30.73</td>
<td>0.9316</td>
</tr>
<tr>
<td>a4</td>
<td>32.62</td>
<td>0.9381</td>
<td>30.58</td>
<td>0.9381</td>
<td>34.69</td>
<td>0.9707</td>
<td>35.77</td>
<td>0.9689</td>
<td>35.77</td>
<td>0.9700</td>
</tr>
<tr>
<td>b1</td>
<td>30.03</td>
<td>0.8956</td>
<td>30.06</td>
<td>0.9015</td>
<td>29.35</td>
<td>0.9199</td>
<td>32.53</td>
<td>0.9465</td>
<td>32.66</td>
<td>0.9491</td>
</tr>
<tr>
<td>b2</td>
<td>30.69</td>
<td>0.8874</td>
<td>30.85</td>
<td>0.9017</td>
<td>31.90</td>
<td>0.9520</td>
<td>33.89</td>
<td>0.9559</td>
<td>33.74</td>
<td>0.9557</td>
</tr>
<tr>
<td>b3</td>
<td>32.31</td>
<td>0.9299</td>
<td>31.30</td>
<td>0.9295</td>
<td>29.28</td>
<td>0.9287</td>
<td>35.38</td>
<td>0.9663</td>
<td>35.34</td>
<td>0.9681</td>
</tr>
<tr>
<td>b4</td>
<td>29.41</td>
<td>0.8933</td>
<td>30.61</td>
<td>0.9089</td>
<td>27.70</td>
<td>0.9095</td>
<td>32.62</td>
<td>0.9462</td>
<td>33.17</td>
<td>0.9526</td>
</tr>
<tr>
<td>avg</td>
<td>30.41</td>
<td>0.9108</td>
<td>29.52</td>
<td>0.9130</td>
<td>30.54</td>
<td>0.9255</td>
<td>32.87</td>
<td>0.9497</td>
<td>32.99</td>
<td>0.9519</td>
</tr>
</tbody>
</table>

Figure 4. Qualitative results of different methods on one typical image in NTURain synthetic testing data set. From left to right: (a) rainy image, (b) ground truth image, (c)-(g) deraining results by DDN, PReNet, SpacCNN, SLDNet and our S2VD.

4. Experimental Results

In this section, we conduct some experiments to evaluate the effectiveness of the proposed semi-supervised video deraining model on synthetic and real data sets. And we briefly denote our Semi-Supervised Video Deraining model by S2VD in the following presentation.

4.1. Evaluation on Rain Removal Task

Training Details: To train S2VD, we employ the synthesized training data of NTURain [6] as labeled data set, which contains 8 rain-free video clips of various scenes. For each rain-free video, 3 or 4 rain layers are synthesized by the Adobe After Effects with different settings, and then added to the videos as rainy ones. As for unlabeled data, 7 real rainy videos without ground truths in the testing data of NTURain are employed. To relieve the burden of GPU memory, we use truncated back-propagation through time in training, meaning that the whole training sequence is divided into different non-overlapped chunks for forward and backward propagation. The length of each chunk is 20.

The Adam [28] algorithm is used to optimize the model parameters in the M-Step of our algorithm. All the network parameters are initialized by [46]. The initialized learning rates for the transition model, emission model and the derainer are set to be 1e-3, 1e-4 and 2e-4, respectively, and decayed by half after 30 epochs. The mini-batch size is set as 12, and each video is clipped into small blocks with spatial size 64 × 64 pixels. Note that we only update the parameter \( W \) for the first 5 epochs to pretrain the derainer, which makes the training more stable. As for the hyper-parameters, \( \varepsilon_0^2 = 1e-6, \rho = 0.5, \gamma = [1,1,2]^T \), and more analysis experiments on them are presented in Sec. 4.2.
average PSNR and SSIM [51] are used as quantitative metrics, which are evaluated only in the luminance channel since we are sensitive to the luminance information.

Table 1 lists the average PSNR/SSIM results on 8 testing videos. Evidently, S2VD attains the best or at least second best performance in all cases. Comparing with current SOTAs (SpacCNN or SLDNet), our method achieves at least 2.5dB PSNR and 0.01 SSIM gains. The qualitative results are shown in Fig. 4. Note that we only display the results of DL-based methods due to the page limitation. We observe that: 1) The derained results of PReNet still contain some rain streaks. 2) Both DDN and SpacCNN lose some image contents. 3) SLDNet can not preserve the original color maps very well. However, our S2VD evidently alleviate these deficiencies and obtains the closest results to the ground truths, which verifies the effectiveness of our model.

### 4.1.2 Evaluation on Real Data

To further test the generalization capability of S2VD in real tasks, we evaluate it on two kinds of real rainy videos, i.e., the real testing data set in NTUTrain and several other real rainy videos in [31]. Note that the former is included in our training set as unlabeled data, but the latter is not. Fig. 5 presents some typical deraining results by different method-

---

**Table 1.** Average PSNR/SSIM results on 8 testing videos.

| Metrics | Methods | NTURain
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>36.11</td>
<td>37.12</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.9602</td>
<td>0.9673</td>
</tr>
</tbody>
</table>

---

**Table 2.** Average PSNR/SSIM results of S2VD on the synthetic testing dataset of NTURain under different ρ values.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>ρ</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>0</td>
<td>38.18</td>
<td>0.9719</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>38.05</td>
<td>0.9713</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>37.37</td>
<td>0.9683</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>35.50</td>
<td>0.9519</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>31.55</td>
<td>0.8947</td>
</tr>
</tbody>
</table>

---

**Table 3.** Average PSNR/SSIM results of three baselines and S2VD on the synthetic testing dataset of NTURain.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Methods</th>
<th>Baseline1</th>
<th>Baseline2</th>
<th>Baseline3</th>
<th>S2VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>36.11</td>
<td>37.12</td>
<td>37.96</td>
<td>37.37</td>
<td></td>
</tr>
<tr>
<td>SSIM</td>
<td>0.9602</td>
<td>0.9673</td>
<td>0.9717</td>
<td>0.9683</td>
<td></td>
</tr>
</tbody>
</table>
ods on these two kinds of data sets. It can be seen that S2VD achieves the best visual results comparing with other methods. Especially, the superiority of our model shown in the second data set substantiates that S2VD is able to handle the real rainy videos even though they do not appear in the unlabeled data set. This generalization capability would be potentially useful in real deraining tasks.

4.2. Additional Analysis

4.2.1 Sensitiveness of Hyper-paramerter $\rho$

The hyper-paramerter $\rho$ in Eq. (2) or (3) controls the relative importance of MRF prior in S2VD. The quantitative performances on the synthetic testing data set and the qualitative performances on the real testing data set of NTURain under different $\rho$ values are presented in Table 2 and Fig. 6, respectively. On one hand, when $\rho$ increases, the performance on the synthetic testing set tends to decrease as shown in Table 2, since the relative importance of the constraint built on the ground truth in Eq. (3) decreases. On the other hand, the MRF prior is able to prevent the de-rainer from overfitting the synthetic data, thus improving the generalization capability in real cases, which has been sufficiently verified by the visual comparisons in Fig. 6. By taking into account these two aspects, we set $\rho$ as 0.5.

4.2.2 Ablation Studies

As shown in Eq. (14), our S2VD degenerates into the Mean Squire Error (MSE) loss when $\varepsilon_0 \rightarrow 0$. Comparing with such special case, our model introduces one more likelihood term, one more MRF regularizer and the semi-supervised learning paradigm. To clarify the effect of each part, we compare S2VD with three baselines as follows: 1) Baseline1: We only train the derainer with the MSE loss on labeled data set. 2) Baseline2: We train S2VD with $\varepsilon_0 = 1e-6$ and $\rho = 0$ only on labeled data set so that we can justify the marginal gain from the likelihood term by comparing with Baseline1 using the MSE loss. 3) Baseline3: On the basis of Baseline2, we introduce the MRF regularizer with $\rho = 0.5$.

The quantitative comparisons on synthetic testing data set of NTURain are listed in Table 3, and the visual results on real testing data set are also displayed in Fig. 6. In summary, we can see that: 1) The performance improvement (1.01dB PSNR and 0.0071 SSIM) of Baseline2 beyond Baseline1 substantiates that the likelihood term plays an important role in our model. 2) Under the supervised learning manner, the MRF prior is beneficial to our model in both synthetic and real cases according to the performance of Baseline3. 3) The addition of unlabeled data in S2VD increases the generalization capability in real tasks as shown in Fig. 6 (d) and (i). However, it leads to a little deterioration of the performance on synthetic data, mainly because there is a gap between the rain types contained in the labeled synthetic and unlabeled real data sets.

4.2.3 Limitation and Future Direction

Although our method can achieve impressive deraining results as shown above, it may still fails in some real scenarios, e.g., a case with a large camera motion between adjacent time frames and a case with heavy rain streaks. Two failure examples are shown in Fig. 7. The reason is because the adopted MRF prior for unlabeled real data is not strong enough to guarantee satisfactory deraining results in these complicated cases. Therefore, it is urgent and necessary to explore better prior knowledge in order to handle more complicated real deraining tasks in the future.

5. Conclusion

In this paper, we design a dynamical rain generator based on the spatial-temporal process in statistics. With such a dynamical generator, a semi-supervised video deraining method is proposed. Specifically, we represent the sequence of rain layers in rain videos using the dynamical rain generator, which is able to facilitate the rain removal task. To handle the generalization issue for real cases, we propose a semi-supervise learning manner to exploit the common knowledge underlying the synthetic labeled and real unlabeled data sets. Besides, a Monte Carlo-based EM algorithm is designed to learn the model parameters. Extensive experimental results demonstrate the effectiveness of the proposed video deraining method.

Acknowledgement: This research was supported by the National Key R&D Program of China (2020YFA0713900), the China NSFC projects under contracts 11690011, 61721002, U1811461, 62076196.
References


[14] Xueyang Fu, Borong Liang, Yue Huang, Xinghao Ding, and John Paisley. Lightweight pyramid networks for image deraining. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. 3


[27] Jin-Hwan Kim, Jae-Young Sim, and Chang-Su Kim. Video deraining and desnowing using temporal correlation and


[53] Wei Wei, Lixuan Yi, Qi Xie, Qian Zhao, Deyu Meng, and Zongben Xu. Should we encode rain streaks in video as deterministic or stochastic? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2535–2544, 2017. 1, 2


