A Rain Pixel Recovery Algorithm for Videos with Highly Dynamic Scenes

Jie Chen, Lap-Pui Chau

Abstract—Rain removal is a very useful and important technique in applications such as surveillance, and movie editing. Several rain removal algorithms have been proposed these years, where photometric, chromatic, and probabilistic properties of the rain have been exploited to detect and remove the rainy effect. Current methods generally work well with light rain and relatively static scenes, when dealing with heavier rain fall in dynamic scenes, these methods give very poor visual results. The proposed algorithm is based on motion segmentation of dynamic scene. After applying photometric and chromatic constraints for rain detection, rain removal filters are applied on pixels such that their dynamic property as well as motion occlusion clue are considered; both spatial and temporal information are then adaptively exploited during rain pixel recovery. Results show the proposed algorithm has a much better performance for rainy scenes with large motion than existing algorithms.

Index Terms—Motion segmentation, Motion occlusion, Dynamic scene, Motion buffering, Adaptive filters, Rain removal.

I. INTRODUCTION

RAIN removal is a complex task. In rainy videos pixels exhibit small but frequent intensity fluctuations, and this fluctuation could be caused by several other reasons besides rain fall, namely, global illumination change, camera move, and object motion etc. In order to remove the rainy effect, it is necessary to detect the fluctuations that are caused by rain, and then replace them with their original value. Some good algorithms have been proposed for this purpose.

Garg and Nayar first analyzed the physical and photometric properties of the rain [1] [2], and they used their observation data to apply both intensity and temporal constraints to detect and then remove the rain. However, their assumptions such as the uniform velocities and directions of the rain drops limited its performance.

Zhang proposed a method based on chromatic constraint [3], where they assume that intensity changes in the R, G, B channels caused by rain are approximately the same, while those caused by object motion are asymmetric. This method is only applicable with static background, and it gives out false result for particular foreground colors.

Tripathi et al. proposed a probabilistic spatial-temporal model [4] [5], where they proposed to use statistical features, i.e. intensity fluctuation range, and spread asymmetry, which are collected from a spatial-temporal neighborhood to classify the rain affected pixels. This method is more robust dealing with dynamic scenes, however some statistical feature it proposes (i.e. spread asymmetry) works poorly in many occasions, and it gives a lot of false detections.

Except from rain detection, one shared shortcoming of the existing methods is in the prediction of the rain covered pixel’s original value, where they simply compute its temporal mean.

This causes serious corruptions in areas where obvious motion is present; important information are erased; and an undesired ghost effect is often observed.

The proposed algorithm is based on motion segmentation of dynamic scene. After applying photometric and chromatic constraints for rain detection, rain removal filters are applied on pixels such that their dynamic property as well as motion occlusion clue are considered; both spatial and temporal information are then adaptively exploited during rain pixel recovery. Experiment results show that our algorithm outperforms existing ones in highly dynamic scenarios.

II. MOTION SEGMENTATION

The pixel intensity fluctuation of a rainy scene is caused by rain and object motion. The fluctuations caused by rain need to be removed, and the ones caused by object motion need to be retained. Thus motion field segmentation naturally becomes a fundamental procedure of our algorithm.

A. Estimation of Motion Field

Motion field is a 2-D vector field projected from the 3-D velocity field of a dynamic scene [6]. Independent moving objects could be detected using motion segmentation [7] [8].

Optical flow is used to evaluate the existence of motion. With the constraint of intensity conservation, apply the chain rule for differentiation, we have [8]:

\[
\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{dI}{dt} = 0,
\]

where \( I(x, y, t) \) is the image brightness at pixel \( P(x, y) \) at time \( t \). Let \( u = \frac{dx}{dt}, v = \frac{dy}{dt}, E_x = \frac{\partial I}{\partial x}, E_y = \frac{\partial I}{\partial y}, E_t = \frac{dI}{dt} \), this equation can be rewritten as

\[
E_x u + E_y v + E_t = 0,
\]

Horn’s method [9] is used for the estimation of \( u, v \), which are the optical flow velocities.

Optical flow is accurate for estimating the relative displacement between two adjacent frames for most objects. However for rain, which is an ensemble of rain drops falling at high
 velocities [10], its contribution to the motion field can be 

eliminated by setting a preset threshold value ($\rho m$ in Eqn. (3))
to $f(x, y)$ (in Eqn. (4)). Here $m$ is the mean of the motion 
field (Eqn. (5)). $\rho$ is a parameter set according to how bad 
the rainy effect is. In typical applications, $\rho = 0.01 \sim 0.1$. By 
applying thresholding it can also eliminate the bad influences 
caused by slow global camera motion.

From Eqn. (3) a binary motion pixel map $I_m(x, y)$ is 
retrieved based on the thresholded optical flow field.

$$I_m(x, y) = \begin{cases} 
0 & f(x, y) \geq \rho m \\
1 & f(x, y) < \rho m, 
\end{cases}$$

$$f(x, y) = \sqrt{u(x, y)^2 + v(x, y)^2},$$

$$m = \frac{1}{MN} \sum_x \sum_y f(x, y).$$

Using optical flow for motion target segmentation has its 
intrinsic drawbacks [8], as it can only detect obvious intensity 
changes, which are usually at the boundary of the target. Areas 
that belong to the target without obvious intensity change 
cannot be effectively recognized, which causes the so called 
“Aperture Problem” [9].

In an effort to tackle with this problem, a parametric 
Gaussian Mixture Model is used (GMM) [11] to simulate 
the distribution of $I_m(x, y)$. First, $K$ GMM components 
are presumed to exist in the optical flow field. Then, Expectation 
Maximization (EM) algorithm is applied to calculate the 
optimal mean and variance for each component. The likelihood 
function for the motion target(s) can thus be expressed as [11]:

$$p(F_1(i, j)|C_1) = \sum_{k=1}^K \pi_k N((i, j)|\mu_k, \Sigma_k),$$

where $\pi_k$ is the mixing coefficient for each Gaussian com-
ponent, its values are determined in the EM iteration, with 
the constraint of $\sum_{k=1}^K \pi_k = 1$. Here $F_1$ represents 
the motion cue, which is combined later with locality cue $F_2$ for 
foreground/background classification. $C_1$ represents the pixel 
class for foreground object, and $C_2$ represents the pixel class 
for background. Fig. 1 shows the simulation result of motion 
target estimation on two video sequences.

Regarding the value of parameter $K$, the proposed algorithm 
does not rely on the assumption that each GMM component 
represents one single motion object, and it works well in 
situations where the number of GMM components $K$ is larger 
than the number of motion object in the frame. The GMM 
models the motion field of the whole frame, each pixel’s 
motion cue is determined by the combination of all Gaussian 
components. Therefore, each motion object can be represented 
by one or multiple GMM components; the algorithm works 
well when 'K' is chosen large enough. For most applications, 
$K$-the maximum number of motion object, can be estimated 
beforehand. For example, for a street traffic surveillance 
camera with resolution 240x320, 20 motion object number is 
already sufficient, considering the possible number of vehicles 
that can appear in such limited space.

Considering the similarity between adjacent video frames, 
GMM parameters from the current frame can be used as initial 
parameters for EM iteration of the next frame, which will make 
the algorithm converge very fast and highly efficient.

B. Including Local properties

In an effort to include the local properties (pixel location, 
and chromatic values) into the segmentation, a feature vector is 
formed consisting of each pixel’s spatial and color information 
[12] [13]:

$$F_{ij} = (R_{ij}, G_{ij}, B_{ij}, w \cdot i, w \cdot j)$$

In Eqn. (7) $R_{ij}, G_{ij}, B_{ij}$ are the intensity values of the 
$R, G, B$ channel at $P(i, j)$, $w$ is the weighting factor between 
the color and position space. For different frame size and scene 
complexity, $w$ should be adjusted as different values.

After generating the feature vectors, k-means clustering is 
applied on the vector space. The number of clusters is pre-
defined according to the frame size and scene complexity as 
well. Fig. 2 shows the result of frame segmentation based on 
the proposed feature vector.

After the clustering, we count the number of motion pixels 
(according to the binary map $I_m(x, y)$) that fall into each 
cluster. The percentage of motion pixel number against the 
cluster’s total pixel number will be used as that cluster’s
likelihood of motion:

\[ p(F_2(i, j)|C_1) = \frac{\text{no. of motion pixels within cluster}}{\text{total no. of pixels of the cluster}}. \]  \hspace{1cm} (8)

Here the symbol \( F_2 \) represents the locality cue.

Since adjacent video frames are closely related to each other, clustering result from the current frame can be used as the initial parameters for the iteration on the next frame, this will help the algorithm converge within several iterations, and makes the computation highly efficient.

C. Combination of Motion and Locality Cues

Assume that the likelihoods of two cues, i.e. motion cue and locality cue \( F_1, F_2 \) are independent, the combined conditional probability is given by [8], [14]:

\[ p(A|C_i) = \prod_{k=1}^{2} p(F_k|C_i), \]  \hspace{1cm} (9)

here \( p(A|C_i) \) is the combined conditional probability for the pixel class \( C_i \). Fig. 3 is a simulation result of the combined cues on two video sequences.

The posterior probability for the foreground motion target can be written (defined as class \( C_1 \)) as:

\[ p(C_1|A) = \frac{p(A|C_1)p(C_1)}{p(C_1)p(A|C_1) + p(C_2)p(A|C_2)}. \]  \hspace{1cm} (10)

Here, \( p(C_1) \) and \( p(C_2) \) are the prior possibilities for each class. \( p(C_1) \) can be predicted as the ratio between the motion target area against the frame size, and \( p(C_2) = 1 - p(C_1) \). The decision boundary can be obtained by setting the likelihood in both classes to be equal, i.e.

\[ p(C_1|A) = p(C_2|A) = 0.5, \]  \hspace{1cm} (11)

and a binary motion decision map is generated to update \( I_{\text{m}} \).

There are several other existing methods for optical flow field segmentation, one famous algorithm is [15], where brightness constancy, gradient constancy, and smoothness assumptions are combined into a variation model. A coarse-to-fine warping strategy is later applied to minimize the total energy. This method provides a better optical flow estimation accuracy, however with an increased computational complexity. As will be explained in the following sections, for rain removal, we need a segmentation that can tell the motion intensive area roughly from the relatively static scene background. The segmentation boundary is not essential, since it is of very low possibility that boundary pixels are covered by rain; accurate optical flow magnitude estimation is also unnecessary, as long as it’s well above the threshold. Under such considerations, the proposed efficient method is preferred over others for the motion field segmentation.

III. RAIN DETECTION

This section describes the rain detection scheme of the proposed algorithm.

A. Differencing and thresholding

First, grey scale intensity differences between two successive frames are calculated and thresholded. The threshold value are set such that all the intensity fluctuations caused by rain can be detected, the typical value is \( D_{\text{th}} = 3 \) (out of intensity scale 255).

\[ I_{\text{diff}} = \begin{cases} 
1 & I_N - I_{N-1} \geq D_{\text{th}} \\
0 & I_N - I_{N-1} < D_{\text{th}} 
\end{cases} \]  \hspace{1cm} (12)

A binary difference map \( I_{\text{diff}} \) is calculated using Eqn.(12).

B. Applying Photometric & Chromatic Constraints

Photometric constraints (according to [1], [16] and [17]) are applied on the candidate rain mask \( I_{\text{diff}} \) calculated in the previous step. First, constraint of intensity fluctuation range \( \epsilon \) is applied on the pixels; and then by considering the speed of rain streak, as well as the comparative camera sensor dimension, the connected area of the rain streak is limited to a certain number of pixels \( \theta \). Based on our experiment, \( \epsilon = 3 \sim 20/255, \theta = 30 \sim 50 / \text{pixels} \), differing on different rain volume, camera frame size and focal length settings [17]. The chromatic constraint [3] is also applied here, the absolute sum of color channel differences caused by rain should be within a bound \( \phi = 15/255 \). After applying the photometric and chromatic constraints, the pixels in \( I_{\text{diff}} \) that fail the constraints are excluded from the final rain mask \( I_{\text{Rain}} \).

\[ I_{\text{Rain}} = I_{\text{diff}} - I_{\text{fail}}. \]  \hspace{1cm} (13)

Quantitative comparison between different methods are carried out for rain detection over static scene background, and the result is listed in Table I. The video we use is 'blue car' (frame size \( 240 \times 320 \)), the rain is rendered using methods proposed in [18], and the rain streak ground truth is calculated as difference between videos before and after rain rendering. The measuring metric is rain mis-detection rate (MD/pels), and false detection rate (FD/pels). We can see our hybrid method

![Fig. 3: Simulation result of two videos showing combined motion likelihood probability using the combined cues. Brighter intensity in the figure signifies higher probability for existence of motion target.](image-url)
TABLE I: Average miss & false detection by competing algorithms over ‘blue car’ video.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>MD/pels</th>
<th>FD/pels</th>
<th>Total Error/pels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al [3]</td>
<td>247</td>
<td>2236</td>
<td>2483</td>
</tr>
<tr>
<td>Tripathi et al [5]</td>
<td>63</td>
<td>1822</td>
<td>1885</td>
</tr>
<tr>
<td>Our method</td>
<td>686</td>
<td>786</td>
<td>1472</td>
</tr>
</tbody>
</table>

produces lower total error as compared to other competing methods.

C. Motion Exclusion

Rain pixels within the motion object and the background need to be treated separately, so here $I_{Rain}$ is divided into two sets: rain candidate pixels in the motion target area will comprise the set $S_m$; rain candidate pixels in the background area will comprise the set $S_b$. Finally, the pixels that are not included in $S_m$ or $S_b$ form the set $S_p$, which are the pixels that are not covered by rain. The definition of $S_m$, $S_b$ and $S_p$ are expressed in Eqn. (14), (15) and (16). $S_C$ is the complete set of frame pixels, $B_M$ is the motion buffer whose definition will be given in the next section.

$$S_m = \{ I(x,y)|I_{Rain}(x,y) = 1 & B_M(x,y,n) = 1 \}$$  (14)

$$S_b = \{ I(x,y)|I_{Rain}(x,y) = 1 & B_M(x,y,n) = 0 \},$$  (15)

$$S_p = S_C - S_m - S_b.$$  (16)

IV. FRAME, RAIN AND MOTION BUFFER

Three buffers are created for the rain removal: video frame buffer $B_I(len,wid,stk)$, rain buffer $B_R(len,wid,stk)$ and motion buffer $B_M(len,wid,stk)$. Here $len \times wid$ is the video frame size, $stk$ is the depth of the buffer, and it is set as $stk = 9$ in the experiment for a better recovering performance.

Each layer of $B_I$ comprises one video frame. The newest frame is pushed on top of the buffer (buffer layer 1), and the oldest frame is moved out from the bottom (buffer layer stk), and the rest of the layers update accordingly.

The rain buffer $B_R(len,wid,stk)$ records the binary rain map $I_{Rain}$ for each corresponding video frame in $B_I$, and motion buffer $B_M(len,wid,stk)$ records the corresponding binary motion decision map $I_m$. Both $B_R, B_M$ update in sync with the video frame buffer $B_I$.

The scene recovery algorithm works on the central frame ($B_I(x,y,n)$, $n = \frac{stk+1}{2}$), such that both historical ($B_I,R,M(x,y,1 \sim \frac{stk-1}{2})$) and future ($B_I,R,M(x,y,\frac{stk+3}{2} \sim stk)$) information in the video, rain and motion buffer could be retrieved for a better scene recovery performance.

V. SCENE RECOVERY

A 88-spatial-temporal neighborhood $V$ is designed for rain removal, which is illustrated in Fig. 4. Lie in the center of the neighborhood system is the pixel $B_I(x,y,n)$. $V$ consists 8 pixels along the time axis in $B_I$ with location $B_I(x,y,(n-4) \sim (n+4))$, and 80 pixels in the current frame with area $B_I((x-4) \sim (x+4), (y-4) \sim (y+4), n)$. Similar neighborhood set-up could be found in [10], where they use a 26-connected $3 \times 3$ neighborhood. The temporal neighborhood is set to be 8 in our method for a better recovering visual result, and for a better resilience against motion occlusion. The spacial neighborhood is set as 80, for the fact that rain streak breadth is usually around 3 to 8 pixels.

The pixel values in the neighborhood $V$ will later be used to predict the center rain covered pixel by a weighted sum. The definition of the weight is given in Eqn. (17), they will later be assigned to each pixels according to their adjacency to the center $B_I(x,y,n)$:

$$w_{x,y,n}(i,j,t) = B_R(x,y,n) \times$$

$$\exp(-\alpha||v(x,y,n) - v(x,y,t)||^2$$

$$- \beta||v(x,y,n) - v(i,j,n)||^2),$$

here $B_R$ is the binary complement of $B_R$, it is used to filter out the pixels in the neighborhood that are also covered by rain. $v(i,j,t)$ is the vector representing the spatial-temporal location of each pixel in the 88-neighborhood $V$. Within the exponential term, $(-\alpha||v(x,y,n) - v(x,y,t)||^2)$ regulates the filter weights along the time (frame no.) axis, and $(-\beta||v(x,y,n) - v(i,j,n)||^2)$ regulates the filter weights in the pixel’s spatial neighborhood.

In the previous section, for each frame the pixels have been divided into three sets: $S_b$, $S_m$, and $S_p$. Different filter coefficient $\alpha, \beta$ in Eqn. (17) are used on pixels of different sets, based on their different dynamic properties.

A. Rain Covered Pixels in ‘Static’ Scene Background

For the rain covered pixels in static background (pixels in set $S_b$), filter coefficients are set as $\alpha = \frac{2}{stk}$, $\beta = 0$, this makes the filter shaped as gaussian with variance $\frac{stk}{2}$ along the time axis (blue curve in Fig. 4), but no spacial neighbor values are used.

With the gaussian pattern, temporally adjacent records ($B_I(x,y,n-1)$ and $B_I(x,y,n+1)$) are assigned with highest weights, and this is out of two considerations: First, according to [1], when a certain pixel is covered by rain, its temporally adjacent pixels will have a higher possibility that they will not also be covered by rain; Second, if considering camera
motion, or background lighting change, adjacent pixels are certainly more likely to be close to the original value. The rain removal filter kernel for this set of pixels is therefore defined as:

\[
B_I(x, y, n) = \frac{\sum_{(i, j, t) \in V} w_{x,y,n}(i, j, t) B_M(i, j, t) B_I(i, j, t)}{\sum_{(i, j, t) \in V} w_{x,y,n}(i, j, t) B_M(i, j, t)},
\]

where \(B_M\) is the binary complement of the motion buffer, which will filter out the pixels in the time axis that have been classified as motion object. The motion buffer \(B_M\) provides the information which layers of \(B_I\) are usable for the background recovery.

It is possible that a pixel be judged to belong to background, while in the motion buffer, all the historical and future records are judged to be motion object, and vice versa. This happens when motion segmentation fails, it can also happen when object motion is very fast. In both situations, no temporal information could be used for pixel recover. For this specific case, \(\beta\) is set to be \(\sqrt{\frac{1}{4V^2}}\) to exploit the spacial neighborhood information, similar to the recovery for motion objects.

**B. Rain Covered Pixels in Motion Objects**

For the rain covered pixels in motion objects (pixels in set \(S_m\)), filter coefficients are set as \(\alpha = 1, \beta = \frac{1}{4\sqrt{2}}\). In the time axis, this gives only considerable weights to the two nearest neighbor in the time axis \(B_I(x, y, n - 1), B_I(x, y, n + 1)\), giving consideration to the fact that the pixel value changes fast when it belongs to a motion object, so when it is covered by rain, buffer layers that are far away from the buffer center provides little relevant information for the motion object, thus their weights are set to be very small. On the contrary, as \(\beta = \frac{1}{4\sqrt{2}}\), more weights are given to the motion pixel’s spacial neighbors, and it has a 2-D gaussian shaped distribution that corresponds to the red curves in Fig. 4. The recovered pixels value is calculated as:

\[
B_I(x, y, n) = \frac{\sum_{(i, j, t) \in V} w_{x,y,n}(i, j, t) B_M(i, j, t) B_I(i, j, t)}{\sum_{(i, j, t) \in V} w_{x,y,n}(i, j, t) B_M(i, j, t)}.
\]

**C. Pixels Uncovered by Rain**

Finally, for the pixels that are not covered by rain (pixels in set \(S_p\)), our algorithm simply keep their values.

**VI. EXPERIMENT RESULT**

Experiments were carried out on two videos of highly dynamic rainy scenes. The results are shown in Fig. 5 and 6. As can be seen from the results, rain streaks are well removed. The cars are not blurred by the rain removal algorithm in spite of its large motion, and no leaving trails (ghost effect) are observable.

**Fig. 8: Comparison on video using different rain removal algorithms.** (a) Garg and Nayar’s [1] result. (b) Zhang’s [3] result. (c) Tripathi’s result [5] result. (d) Result from our proposed method. Motion causes a lot of corruptions on the cars in (a) and (b); in (c) the front wheel of car is removed because of its motion.

The comparison between our rain removal algorithm with other existing ones are shown in Fig. 7. As can be seen, for static scenes (the first row in Fig. 7), all the algorithms exhibit equally satisfying performances. However, when the scene becomes dynamic (car entering), the differences become obvious, and the prominent parts that can show the superiority of our algorithm have been put in red rectangles. Fig. 8 and 9 give a further closer look on the performance differences. From Fig. 8(a), (b) and Fig. 9(a), (b), Garg and Nayar [1], and Zhang’s [3] method added a lot of corruptions to the motion surfaces after rain removal, which is caused by involving background values in the prediction of motion target values. Tripathi’s [5] algorithm performs better with motion than the previous two, as in Fig. 8(c) and 9(c), but a big part of the car’s wheels are wiped out, and a noticeable number of mis-detections of rain streaks often occur after the direction of motion target. In general, the results show that our rain removal algorithm has a much better performance over others for rain removal in highly dynamic scenes.

**VII. CONCLUSION**

Existing rain removal algorithms perform poorly in highly dynamic scenes, serious pixel corruptions often occur in motion intensive areas, which is caused by ignoring motion occlusions during pixel recovery. Based on the proposed motion segmentation scheme, our method recovers the rain pixels such that each pixel’s dynamic property as well as motion occlusion clue is considered; both spatial and temporal information are adaptively exploited during rain pixel recovery. Experiment results show that our algorithm outperforms existing ones in highly dynamic scenarios.
Fig. 5: Experiment results on video sequence 1. The first row are the original videos, the second row are the motion segmentation results, and the third row are the videos after rain removal.

Fig. 6: Experiment results on video sequence 2. Column (a) are the motion segmentation results, (b) are the original videos, and (c) are the rain removed videos.
Fig. 7: Comparison on video sequence 1 using different rain removal algorithms. (a) Original video frames. (b) Garg and Nayar’s results using the photometric model in [1]. (c) Zhang’s results using the chromatic model in [3]. (d) Tripathi’s results using the spatial-temporal model in [5]. (e) Results from our proposed method.

Fig. 9: Comparison on video using different rain removal algorithms. (a) Garg and Nayar’s [1] result. (b) Zhang’s [3] result. (c) Tripathi’s result [5] result. (d) Result from our proposed method. Corruptions are obvious in (a) and (b); the rear wheel of car is removed in (c), and a lot of rain behind the car remain un-removed in (c).

REFERENCES


Jie Chen received the B.S. and M. Eng degree from School of Optical and Electronic Information, Huazhong University of Science and Technology, China. He is currently pursuing Ph.D. degree in School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore.

His research interests are in image processing (low contrast image processing, denoising, interpolation), image sparse representation and applications, and computational photography.