Raindrop Detection and Removal from Long Range Trajectories

Shaodi You[1(✉)], Robby T. Tan[2], Rei Kawakami[1], Yasuhiro Mukaigawa[3], and Katsushi Ikeuchi[1]

[1] The University of Tokyo, Tokyo, Japan
yousd@cvl.iis.u-tokyo.ac.jp
[2] SIM University, Singapore, Singapore
[3] Nara Institute of Science and Technology, Nara, Japan

Abstract. In rainy scenes, visibility can be degraded by raindrops which have adhered to the windscreen or camera lens. In order to resolve this degradation, we propose a method that automatically detects and removes adherent raindrops. The idea is to use long range trajectories to discover the motion and appearance features of raindrops locally along the trajectories. These motion and appearance features are obtained through our analysis of the trajectory behavior when encountering raindrops. These features are then transformed into a labeling problem, which the cost function can be optimized efficiently. Having detected raindrops, the removal is achieved by utilizing patches indicated, enabling the motion consistency to be preserved. Our trajectory based video completion method not only removes the raindrops but also complete the motion field, which benefits motion estimation algorithms to possibly work in rainy scenes. Experimental results on real videos show the effectiveness of the proposed method.

1 Introduction

The performance of outdoor vision systems can be degraded due to bad weather conditions such as rain, haze, fog and snow. On rainy days, it is inevitable that raindrops will adhere to camera lenses, protecting shields or windscreens, causing failure to many computer vision algorithms that assume clear visibility. One of these algorithms is motion estimation using long range optical flow. In this case the correct correspondence of pixels affected by adherent raindrops will be erroneous, as shown in Fig. 1.b.

In this paper, our goal is to detect and remove adherent raindrops (or just raindrops for simplicity) by employing long range trajectories. To accomplish this goal, our idea is to first generate initial dense trajectories in the presence of raindrops. Surely, these initial trajectories are significantly affected by raindrops, causing them to be terminated and drifted. We analyze the motion and

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-16808-1_38) contains supplementary material, which is available to authorized users. Videos can also be accessed at http://www.springerimages.com/videos/978-3-319-16807-4.

D. Cremers et al. (Eds.): ACCV 2014, Part II, LNCS 9004, pp. 569–585, 2015.
DOI: 10.1007/978-3-319-16808-1_38

Fig. 1. An example of the results of our proposed detection and removal method. (a) Scene with raindrop. (b) Dense long trajectories. (c) Matching of trajectories occluded by raindrop. (d) Trajectory based video completion.

appearance behavior of the affected trajectories, and extract features from them. We formulate these features in a Markov-random-field energy function that can be optimized efficiently. Having detected raindrops, we use trajectory linking to repair the terminated or drifted trajectories. Finally, we remove the raindrops using the trajectory based video completion (Fig. 1.c and d). The overall pipeline is described in Fig. 2.

Unlike some existing methods, in this work, first we introduce a novel detection method applicable for both thick and thin raindrops as well as raindrops of any size, shape, glare, and level of blurring. We call a raindrop thick when we cannot see the objects behind it, and thin, when it is sufficiently blurred, but still allows us to partially see the objects behind it. Second, we perform a systematic analysis of the behavior of thick and thin raindrops along motion trajectories based on appearance consistency, sharpness, and raindrop mixture level. This analysis is novel, particularly when applied to raindrop detection. Third, we devise a method to detect and remove raindrops that allows us to recover the motion field. In addition, to our knowledge, our method is the first to address the problem of adherent raindrops in the framework of long range motion trajectories.

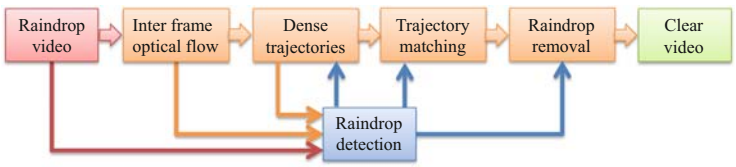

Fig. 2. The pipeline of our method.

2 Related Work

Bad weather has been explored in the past decades including: haze, mist, fog (e.g., [1–4]), falling rain and snow (e.g., [5–7]). For falling rain, Garg and Nayar study the physical model first [8], and later detect and remove it by adjusting camera parameters [6,9]. Barnum *et al.* [5] detect and remove both rain and snow. Recently, single image based methods are proposed by Kang *et al.* [10] and Chen *et al.* [7]. Unfortunately, applying these methods to handle adherent raindrops is infeasible, because of the significant physics and appearance differences between falling raindrops and adherent raindrops.

A number of methods have been proposed to detect thick adherent raindrops caused by sparse rain. Eigen *et al.* [11] and Kurihata *et al.* [12] proposed learning based methods, which are designed to handle raindrops, but not specifically to differentiate raindrops from opaque objects such as dirt. Both of the methods work only with small and clear (non-blurred) raindrops. Yamashita *et al.* utilize specific constraints from stereo and pan-tilt cameras [13,14], and thus is not directly applicable for a single camera. Roser *et al.* propose a ray-tracing based method for raindrops that are close to certain shapes [15,16], and thus can cover only a small portion of possible raindrops. You *et al.* [17] propose a video based detection method by using intensity change and optical flow. The method is generally useful to detect raindrops with arbitrary shapes, however the detection of thin raindrops are not addressed, and it requires about 100 frames to have good results. In comparison, our method only needs 24 frames, assuming the video frame rate is 24 fps.

As for raindrop removal, Roser and Geiger [15] utilize image registration, while Yamashita *et al.* [13,14] align images using the position and motion constraints from specific cameras. You *et al.* use temporal low-pass filtering and patch based video completion [18]. Generally, there are some artifacts in the repaired video because none of these methods consider motion consistency which is sensitive to human visual perception. Eigen *et al.* [11] replace raindrop image patches with clear patches through a neural-network learning technique, causing the method to be restricted on the raindrop appearance in the training data set. This method can only replace small and clear raindrops.

Sensor dust removal might be related to raindrop detections, [19–21], by considering raindrops as dust. Unlike dust however, raindrops could be large, not as blurred as dust, and affected by the water refraction as well environment reflection, making the sensor dust removal methods unsuitable for detecting raindrops.

For video based motion estimation, dense and temporally smooth motion estimation is desired. Sand *et al.* [22] propose particle video which generates motion denser than sparse tracking and longer than optical flow. Later, this idea is improved by Sundaram *et al.* [23] by utilizing GPU acceleration and large displacement optical flow [24]. Volz *et al.* [25] archive a pixel-level density by a new optical flow objective function, however their latency is limited to several frames. Rubinstein *et al.* [26] extend the temporal latency of methods [22,23] by linking the trajectories occluded by solid objects. This paper uses [23] for initial trajectory estimation but with different termination criteria, and utilizes trajectory linking as in [26] but with the features derived from our trajectory analysis over raindrops. As a result, the motion field estimation of degraded videos by raindrops can be much improved, compared to those that do not consider such degradation.

3 Trajectory Analysis

To find features that differentiate raindrops from other occlusions, as well as to identify thick and thin raindrops, we need to analyze the appearance of patches

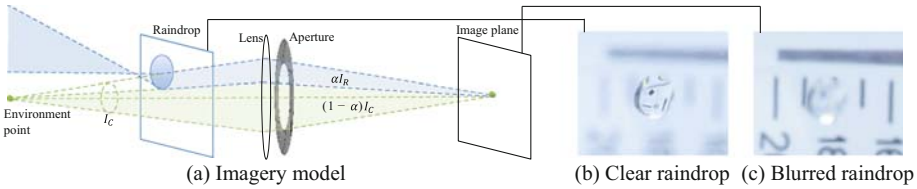

Fig. 3. (a) Raindrop model. (b) Appearance of a clear raindrop. (c) Appearance of blurred raindrop observed on the image plane.

along individual trajectories and the consistency of forward/backward motion. For this, we first need to know the image formation model of raindrops, and the computation of long range trajectories.

Raindrop Model. Unlike opaque objects, raindrops can look different in different environments due to the focus of the camera on the environment. Figure 3 illustrates a raindrop physical model. Given a pixel located at (x, y), the appearance of the clear environment is denoted as $I_c(x, y)$ and the raindrop appearance as $I_r(x, y)$. For raindrops, the following mixture function models the intensity [17]:

$$I(x, y) = (1 - \alpha(x, y))I_c(x, y) + \alpha(x, y)I_r(x, y), \tag{1}$$

where $\alpha(x, y)$ denotes the mixture level, which is dependent on the size and position of the raindrop as well as the camera aperture.

Dense Long Range Trajectories. Given a video sequence, we can form a $3D$ spatio-temporal space as illustrated in Fig. 4.a, where the spatial position of each pixel is indicated by (x, y) and the time of each frame i by t_i. The notation T in Fig. 4.b represents a trajectory consisting of a number of concatenated nodes $N(i)$, shown in Fig. 4.c, and can be expressed as:

$$\begin{aligned} T &= \{N(i)\}, i_{start} \leq i \leq i_{end} \\ N(i) &= (x(t_i), y(t_i)) = (x_i, y_i), t_{start} \leq t_i \leq t_{end}, \end{aligned} \tag{2}$$

where i is the index of the video frame, and (x_i, y_i) is the position of the node. The start and end of a trajectory are denoted by t_{start} and t_{end} respectively. Note

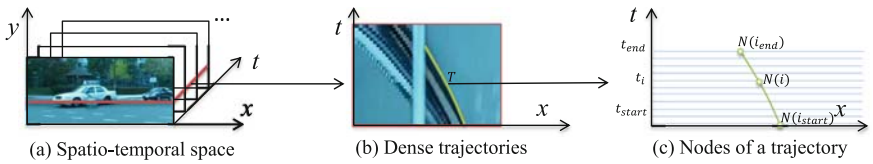

Fig. 4. Spatio-temporal space and dense trajectories. (a) 3D Spatio-temporal space; (b) A 2D slice visualizes the dense trajectories. (c) A trajectory consists of a number of concatenated nodes.

that the nodes are arranged in a temporal ascending order, where a trajectory has only one node at each frame.

We employ GPU-LDOF [23] to generate the initial dense trajectories. However, we ignore its trajectory termination criteria; since, [23] considers only solid occlusions, while in rainy scenes, there are thin raindrops, where the occluded scenes can still be seen. Another reason is that [23] considers occlusion boundaries to be sharp, while in our case, raindrop boundaries are usually soft due to the out-of-focus blur. We generate trajectories in a forward motion, from the first to the last frame. In this case occlusions by raindrops or other objects might cause some trajectories to stop, and consequently some areas in some frames will not have trajectories. To cover these areas, we also generate trajectories in a backward motion.

Figure 5 shows an example of the dense trajectories in a clear day scene and in a scene with a thick and in a scene with a thin raindrop. In our findings, with regard to occlusions, a trajectory can encounter the following events: (A) it is occluded by a solid non-raindrop object and drifted; (B) it is occluded by a thick raindrop and drifted; (C) it is occluded by a thin raindrop and drifted; and (D) it is occluded by a thin raindrop but not drifted.

These events encountered by trajectories allow us to identify the presence of raindrops. We consider that occlusions by thick raindrops or opaque objects will

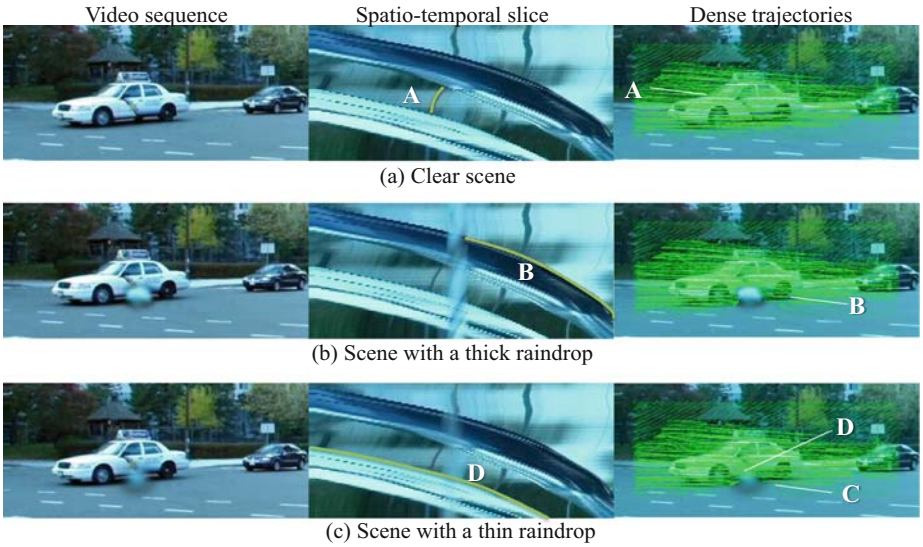

Fig. 5. Video in rainy scenes and events on the trajectories. (a) A clear day scene. (b) A scene with a thick raindrop. (c) A scene with a thin raindrop. The clear scene data is from [22]. Four trajectory events are labeled as, A: Occluded by a solid non-raindrop object and drifted. B: Occluded by a thick raindrop and drifted. C: Occluded by a thin raindrop and drifted. D: Occluded by a thin raindrop but not drifted. The trajectory appearance of each event is shown in Fig. 6.

cause abrupt changes in both the appearance and the motion along trajectories, while occlusions by thin raindrops will mainly cause changes in the appearance, particularly the sharpness. The details of the analysis are as follows.

3.1 Motion Consistency Analysis

For a trajectory T generated by forward tracking, we consider a node $N(i)$ on frame t_i. Its succeeding $N(i+1)$ is found by referring to the forward optical flow $\boldsymbol{f}_i^+ = (u^+, v^+)_i$ from frame i to frame $i+1$:

$$N(i+1) = (x_i, y_i) + (u^+(x_i, y_i), v^+(x_i, y_i))_i = N(i) + \boldsymbol{f}_i^+(N(i)). \tag{3}$$

Similarly, given a trajectory T' generated from backward tracking, nodes are related by the backward motion:

$$N'(i) = N'(i+1) + \boldsymbol{f}_{i+1}^-(N'(i+1)), \tag{4}$$

where $\boldsymbol{f}_{i+1}^- = (u^-, v^-)_{i+1}$ is the backward optical flow from frame t_{i+1} to frame t_i.

If nodes along a trajectory are not occluded and the optical flow is correctly estimated, the following equation stands with negligible (sub-pixel) error:

$$\begin{aligned} m^+(N(i)) &= \|\boldsymbol{f}_i^+(N(i)) + \boldsymbol{f}_{i+1}^-(N(i) + \boldsymbol{f}_i^+(N(i)))\|_2 = 0 \\ m^-(N'(i)) &= \|\boldsymbol{f}_i^-(N'(i)) + \boldsymbol{f}_{i-1}^+(N'(i) + \boldsymbol{f}_i^-(N'(i)))\|_2 = 0 \end{aligned} \tag{5}$$

where $m^+(N(i))$ and $m^-(N(i))$ are the forward motion consistency and the backward motion consistency of node $N(i)$, respectively.

Motion Inconsistency Caused by Occlusions. Given a trajectory from the forward tracking (or the backward tracking), the motion consistency $m^+(N(i))$ might not be zero if $N(i+1)$ is occluded. In events A and B, $N(i+1)$ is completely occluded by an opaque object or a thick raindrop. In this case, $N(i)$ does not have a corresponding node in the next frame. However, the inter-frame optical flow $\boldsymbol{f}_i^+$ still gives correspondence for $N(i)$. This is because the optical flow regulation forces every pixel to have correspondence. Thus, corresponding node $N(i) + \boldsymbol{f}_i^+(N(i))$ is wrong, resulting in a non-zero motion consistency.

In event C, $N(i+1)$ is occluded by a thin raindrop, which according to Eq. (1), can generate a partial occlusion. As illustrated in Fig. 6, in this event, the consistency is likely to be non-zero, since the pixel at $N(i+1)$ is the mixture of both the tracked node and the raindrop, where each of them has correspondence in the previous frame; causing both the forward and backward optical flow to likely generate wrong correspondence. Here, the mixture level α plays an important role for the wrong correspondence. Overall, the thicker the raindrop, the more likely the consistency is to be non-zero.

In event D, $N(i+1)$ is occluded by a considerably thin raindrop, where $N(i+1)$ is sufficiently visible such that both the forward and backward optical

Fig. 6. When a point is covered by a thin raindrop, it has two correspondences in other frames: the raindrop and the covered object. The causes incorrect tracking for optical flow that assumes only one correspondence.

flow correctly match $N(i)$ with $N(i+1)$. In this event, the mixture level is close to zero, usually less than 0.2.

Motion Consistency Feature. Since events A, B and C might result in a non-zero motion consistency value, we can use the consistency, $m^+(N(i))$ and $m^-(N(i))$, as features to indicate the presence of occlusion, which in some cases, can be raindrops.

We calculate the motion consistency feature for each frame at t_i by collecting m^+ and m^- of all the nodes in the frame, denoted as M_i. Assuming the video frame rate is 24 fps[1] and raindrops are static in a short time period (one second), we sum up the features over 24 frames:

$$\mathcal{M}_i = \sum_{i-24<j\leq i} M_j. \tag{6}$$

Some pixels might not have consistency values due to the failure of optical flow to track. In this case, we obtain the values from linear interpolation. Figure 8.a shows an example of $\mathcal{M}_i$.

As for event D, since possible occlusion can not be detected by the motion consistency, we detect it based on the appearance analysis, discussed in the subsequent section.

3.2 Appearance Analysis

Given a trajectory T, we crop a small image patch, denoted as $P(i)$, centered at each node $N(i)$ with length r, where r is set to 21 pixels by default (based on the resolution of our videos). Figure 7 shows an example of patches sequenced along trajectories for events A, B, C, and D.

[1] The 24fps framerate only for reference on how we can deal with raindrop dynamics since our method assumes static raindrops during the detection process, while in fact in the real world raindrops can move. Hence, assuming the widely adopted framerate, it means we assume raindrops at least do not move in 1-s period of time. Obviously, a higher framerate does not pose any problem (except for the computation time), however a much lower framerate will create a large displacement problem, which can affect the optical flow accuracy.

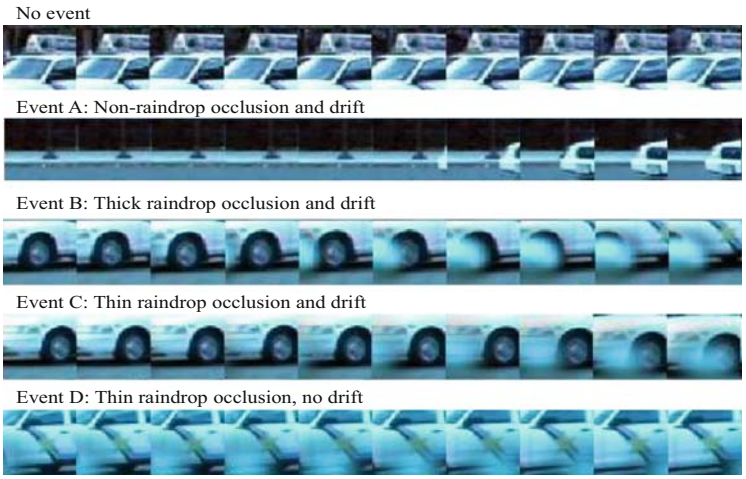

Fig. 7. Appearance of trajectories in Fig. 5. The patch size (21×21 pixels by default) is set to 41×41 pixels for better visualization.

Appearance Consistency. As can be seen in Fig. 7, all four events might generate appearance changes, particularly for events A, B and C. We calculate the appearance consistency for node $N(i)$ using:

$$a(N(i)) = \|SIFT(P(i+1)) - SIFT(P(i))\|_2, \tag{7}$$

where $SIFT(\,)$ is the SIFT descriptor [27], converts patch P to one feature array. For color images, RGB channels are converted separately and, later combined.

The reason of choosing SIFT is to achieve robustness against some degrees of affine deformation. Since even without occlusions, the appearance of an image patch might change. Note that within a few frames (i.e., fewer than 24 frames), these changes should be within the degrees where SIFT can still work, since they represent less than 1 s in real time.

Similar to the motion consistency, we compute the appearance consistency for frame t_i, denoted as A_i, by collecting the appearance consistency of all of the nodes in the frame. The integration of A_i over 24 frames is denoted as $\mathcal{A}_i$. Figure 8.b shows an example of $\mathcal{A}_i$.

The appearance consistency is able to detect all of the occlusion events (A, B, C, and D), however, it lacks the ability to distinguish a non-raindrop occlusion from a solid raindrop occlusion.

Sharpness Analysis. We define the sharpness of patch $P(i)$ as:

$$s(P(i)) = \sum_{(x,y)\in P(i)} \left\| \frac{\partial}{\partial x} I(x,y), \frac{\partial}{\partial y} I(x,y) \right\|_2 \tag{8}$$

where $I(x, y)$ is the intensity value of pixel (x, y). For color images, RGB channels are calculated separately and added up afterward.

Unlike blurred raindrops that have low sharpness in the area including the boundary, non-raindrop objects will have large sharpness at their boundary, or inside their area when they are textured. Therefore, by evaluating the sharpness, we can differentiate non-raindrop objects (Event A) from raindrops (Events B, C and D). The sharpness for frame t_i, denoted as S_i is the collection of the sharpness of all nodes in the frame. The integration of S_i over 24 frames is denoted as $\mathcal{S}_i$, Fig. 8.c shows an example of $\mathcal{S}_i$.

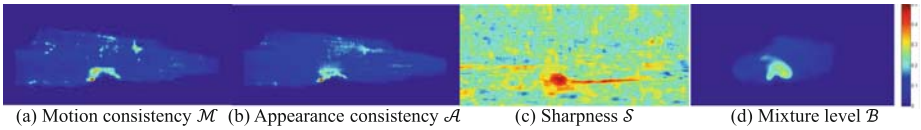

Fig. 8. Raindrop features. (a) Accumulated motion consistency $\mathcal{M}$. (b) Accumulated appearance consistency $\mathcal{A}$. (c) Accumulated sharpness $\mathcal{S}$, colormap is inversed for visualization. (d) Mixture level estimation $\mathcal{B}$.

Raindrop Mixture Level. Analyzing the sharpness along trajectories does not only enable us to distinguish raindrops from non-raindrop objects, but it also allows us to estimate the raindrop mixture level, α. For a given patch $P(i)$, Eq. (1) can be rewritten as:

$$\begin{aligned} P(i) &= (1-\alpha(i))P_c + \alpha(i)P_r(i) \\ \alpha(i) &= \alpha(N(i)) = \alpha(x(t_i), y(t_i)), \end{aligned} \tag{9}$$

where P_c is the clear patch component and $P_r(i)$ is the raindrop component. $\alpha(i)$ is the mixture level of the patch. In the equation, we have made two approximations: First, the mixture level α inside a patch is constant. Second, the change of clear patch component P_c along the trajectory is negligible in a short time period (i.e., within 24 frames for a video with 24 fps).

From Eqs. (8) and (9), we can write the following:

$$\begin{aligned} s(P(i)) &= s[(1-\alpha(i))P_c + \alpha(i)P_r(i)] \\ &\leq s[(1-\alpha(i))P_c] + s[(\alpha(i)P_r(i)] \\ &= (1-\alpha(i))s(P_c) + \alpha(i)s(P_r(i)). \end{aligned} \tag{10}$$

$$s((1-\alpha(i))P_c) = s[P(i) - \alpha(i)P_r(i)] \leq s(P_c) + \alpha(i)s(P_r(i)). \tag{11}$$

Assuming the raindrop is sufficiently blurred, we have: $s(P_r(i)) = 0$. Substituting this in Eqs. (10) and (11) and comparing them, we have: $s(P(i)) = (1-\alpha(i))s(P_c)$. Thus, we can estimate the mixture level of patch $P(i)$ by comparing the sharpness with a clear patch in the same trajectory as:

$$\alpha(i) = 1 - s(P(i))/s(P_c). \tag{12}$$

For a given patch $N(i)$, sharpness of a clear patch $sh(P_c)$ is obtained by evaluating the patch sharpness for m neighbor patches along the trajectory:

$$s(P_c) = \max s(P(i \pm j)), j \leq m, \tag{13}$$

where $m = 10$ as default. When the clear patch has less texture, $s(P_c)$ is small and will result in a large error in Eq. (13). Hence, we only use textured patches to estimate the mixture level. Note that, if m is too small, the trajectory interval is too short, making us unable to have clear patches. On the contrary, if m is too large, the tracking drift will accumulate, causing the trajectories to be incorrect. In our observation for our test videos, $m = 10$ could avoid the problem.

Similarly, we can collect the mixture level for frame t_i, denoted as B_i. The integration of B_i is denoted as $\mathcal{B}_i$. Figure 8.d is an example of $\mathcal{B}_i$.

4 Raindrop Detection

The detection of raindrops can be described as a binary labeling problem, where for given a frame, the labels are raindrop and non-raindrop. Similarly, the mixture level can be described as a multiple labeling problem. The labeling can be done in the framework of Markov random fields (MRFs).

Raindrop Labeling. In the previous section, three features are shown for raindrop detection: motion consistency $\mathcal{M}$, appearance consistency $\mathcal{A}$ and sharpness $\mathcal{S}$. Thus, to detect raindrops, we combine these three features, after normalizing them, to form the following data term:

$$\begin{aligned} E_{data}(\boldsymbol{x}) &= \|\mathcal{F}(\boldsymbol{x}) - (w_m + w_a)L(\boldsymbol{x})\|_1 \\ \mathcal{F}(\boldsymbol{x}) &= (w_m\mathcal{M}(\boldsymbol{x}) + w_a\mathcal{A}(\boldsymbol{x}))\max(0, 1 - w_s\mathcal{S}(\boldsymbol{x})) \end{aligned} \tag{14}$$

where w_m, w_a and w_s are the weight coefficients for the three features. And $\mathcal{F}(\boldsymbol{x})$ is the combined feature. The weights were chosen empirically by considering the precision-recall curve, where a larger weight enabled more sensitive detection. We set $w_m = 16$, $w_a = 16$ and $w_s = 1$ by default. $L(\boldsymbol{x}) \in \{0, 1\}$ is the binary label, with 0 being non-raindrop. The normalization of the three features is done by setting the mean value to 0.5 and the variance to 0.5.

Since the boundaries of raindrops are significantly blurred, we can use a smoothness prior term for labeling neighboring pixels:

$$E_{prior}(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in V(\boldsymbol{x})} |L(\boldsymbol{x}_j) - L(\boldsymbol{x})|, \tag{15}$$

where $V(\boldsymbol{x})$ is the neighbor of $\boldsymbol{x}$. We use graphcuts [28–31] to solve the optimization. Figure 9.a is an example of the labeling result.

Mixture Level Labeling. Having obtained the binary labeling of the raindrop areas, we further label the raindrop mixture level $\alpha(\boldsymbol{x})$ through multi-level labeling. We use the estimated mixture level $\mathcal{B}$ (Eq. (12)) as a clue. The data term is expressed as:

$$E'_{data}(\boldsymbol{x}) = w_b\|\mathcal{B}(\boldsymbol{x}) - \alpha(\boldsymbol{x})\|_1 + w_L\|\tilde{L}(\boldsymbol{x}) - \alpha(\boldsymbol{x})\|_1, \tag{16}$$

Fig. 9. Raindrop detection via labeling. (a) Binary labeling of the raindrop area. (b) Multiple labeling of the mixture level.

where $\tilde{L}(\boldsymbol{x})$ is the binary labeling result, w_b and w_L are the weight coefficients which are set to $w_b = 8$, $w_L = 2$ by default. $\alpha(\boldsymbol{x})$ has 21 uniform levels from 0 to 1. The prior term is set in a similar way to that of the binary labeling. Figure 9.b shows our estimated mixture level for all pixels.

5 Raindrop Removal

Having detected the raindrops, the next step is to remove them. The idea is that given a detected area of a raindrop, we collect the patches along the corresponding trajectories, and use these patches as a source of information to fill in the detected raindrop area.

Based on the binary labeling result, we first remove nodes in the trajectories that are labeled as raindrops, since these trajectories are likely to be incorrect or drifted. By this operation, some of the trajectories will be shortened, and the others will be broken into two trajectories.

To replace the removed nodes of trajectories, we match the corresponding existing trajectories based on [26], where the data term is based on SIFT, temporal order, and inter-frame motion. Figure 1.c is an example of matched trajectories. After matching, we interpolate the missing nodes. Given a matched trajectory pair T_i and T_j, the last node of T_i, denoted as $N^i(end) = (x(t^i_{end}), y(t^i_{end}))$, is matched to the first node of T_j, denoted as $N^j(1) = (x(t^j_{start}), y(t^j_{start}))$. Here, $t^i_{end} < t^j_{start}$ means for all matched pairs. We linearly interpolate the missing nodes between frames t^i_{end} and t^j_{start} based on:

$$N(k) = \frac{t^j_{start} - t_k}{t^j_{start} - t^i_{end}} N^i(end) + \frac{t_k - t^i_{end}}{t^j_{start} - t^i_{end}} N^j(1), \quad t^i_{end} < t_k < t^j_{start}. \tag{17}$$

5.1 Trajectory-Based Video Completion

Having obtained the trajectories for the raindrop areas, the raindrop completion is done by propagating the clear background pixels along a trajectory towards the raindrop area. Using the guidance of trajectories, we propose a removal strategy which preserves both spatial and temporal consistency.

The completion is done frame by frame. First, we start from the first frame and move forward until we find a frame t which contains interpolated nodes. For the frame t, inside a raindrop area, we denote the interpolated nodes as

$\{N_i(t)\}$, where i is the trajectory index. According to the trajectory, we find the corresponding nodes in the previous frame: $\{N_i(t-1)\}$. A transformation can be determined between the two sets of nodes. Depending on the number of nodes in the set, we use affine transformation for three and more matches, translation and rotation for two matches, and translation for one match. Then, the image patch from $t-1$ is transformed and placed at the raindrop area in t. By utilizing information from groups of nodes, we preserve both spatial consistency and temporal consistency. This process continues until it reaches the last frame. For the repaired patch, we denote its confidence as: $C(t) = C(t-1) - 1$. The confidence degrades by 1 every time it is propagated. And the non-interpolated patches have a confidence of 0.

Similarly, we do the backward process starting from the last frame. As a result, for each repaired area, there are two solutions: one from the forward process, and one from the backward process. We chose the one with the higher confidence. As for static or quasi-static areas where no linked trajectory exists, we use the video inpainting method by Wexler *et al.* [18] for repair. An example of the repaired video is shown in Fig. 1.d.

Thin Raindrops. For thin raindrops (event D, generally $\alpha < 0.2$), the trajectories inside the raindrop areas are already correct, therefore we do not need to propagate the appearance from other frames, since we can directly enhance the appearance. As discussed in Sec. 3.2, thin raindrops can be relatively blurred, hence to enhance them, for a node N with appearance P, we convert P to $\mathcal{P}$ using 2D-DCT and set the constant component $\mathcal{P}(0,0) = 0$. Then, we enhance the sharpness according to the mixture level: $\mathcal{P}' = \frac{1}{1-\alpha}\mathcal{P}$. We replace the constant component which is the one with a non-raindrop node along the trajectory. Finally, the enhanced patch P' is obtained using inverse-DCT.

6 Experiments

We conducted both quantitative and qualitative evaluation to measure the accuracy of our detection and removal method. Our video results are included in the supplementary material.

6.1 Raindrop Detection

Dataset. In our experiments, the video data were taken from different sources to avoid data bias and to demonstrate the general applicability of our method. Data 1 was from Sundarum *et al.* [23], data 3 was from KITTI Benchmark [32], data 5 and 7 were from You *et al.* [17] and the rest were downloaded from the Internet. In these data, the camera setups vary from a car mounted camera, a hand held camera to a surveillance camera.

Comparison with State-of-the-art. We used both synthetic and real raindrops, and compared our method with three state-of-the-art methods, Eigen *et al.*'s [11], You *et al.*'s [17] and Roser *et al.*'s [15]. The results are shown in

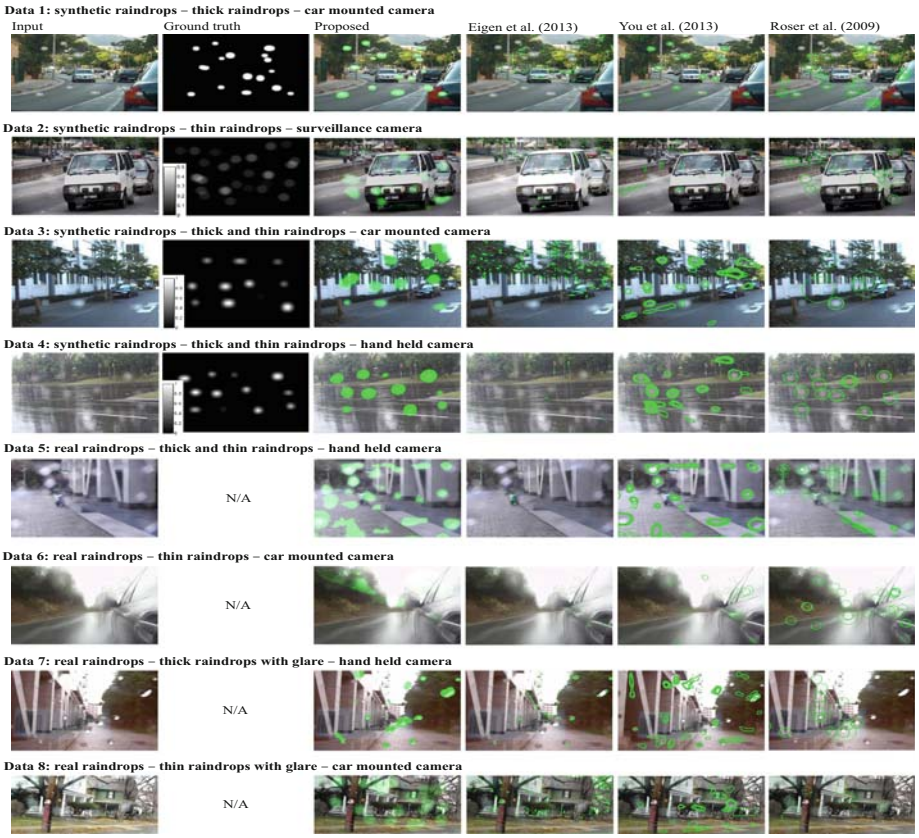

Fig. 10. The raindrop detection results using our method and the existing methods.

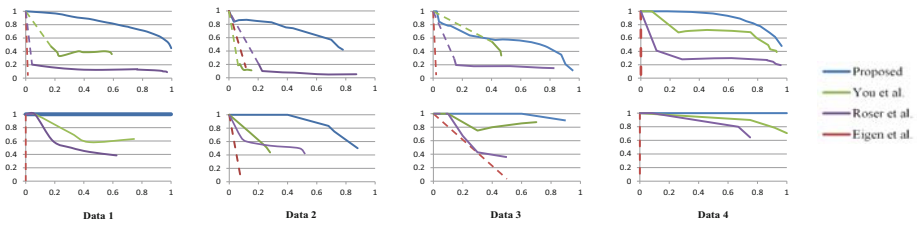

Fig. 11. Precision-recall curve on detection for the methods shown in Fig. 10. First row: evaluation at a pixel level. Second row: evaluation at number of raindrops level. Dashlines indicates the range where no data is available.

Fig. 10. As can be seen, Eigen *et al.*'s method failed to detect large and blurred raindrops, and mislabeled textured areas (such as trees) as raindrops. As for You *et al.*'s method, although it correctly detected thick raindrops, thin raindrops were simply neglected. Roser *et al.*'s method detected round raindrops and thin raindrops only when the background was textuerless.

Quantitative Evaluation. For the synthetic raindrops, data 1–4 in Fig. 10, we quantitatively evaluated using the precision-recall curve. In addition of number of raindrop level evalutation, we also performed pixel-level evaluation. The precision is defined as the number of the correctly labeled pixels divided by the number of all pixels labeled as raindrops. The recall is defined as the number of the correctly labeled pixels divided by the number of the actual raindrop pixels. The result is shown in Fig. 11. As can be seen, our proposed method outperformed some existing methods for both accuracy and recall. Our method have a low false alarm rate for both thick and thin raindrops. As for the real raindrops, data 5–8, our method successfully labeled thin raindrops as well as thick raindrops and achieved better precision.

False Alarm rate Evaluation. To test the robustness of our method, we ran our algorithm on the first four data shown in Fig. 10 with all the synthetic raindrop removed. Table 1 shows the number of raindrop spots detected, although there is no raindrop in the input videos. Our method shows a significantly low false alarm rate compared to the other methods.

Table 1. False alarms on Data 1–4 (Fig. 10) with all synthetic raindrops removed. Evaluated by number of spots erroneuously detected as raindrops.

	Proposed	Eigen et al.	You et al.	Roser et al.
Data 1	0	67	8	17
Data 2	1	48	16	12
Data 3	1	140	6	5
Data 4	0	12	4	2

Speed. On a 1.4 GHz notebook with Matlab and no parallelization, the inter-frame optical flow was about one minute per frame. The tracking and feature collecting together was about 0.2 s per frame. Graphcut was about 5 s for one detection phase. While our algorithm is not real time, we consider it to be still useful for offline applications, such as road accident analysis, Google-like street data collection, etc.

6.2 Raindrop Removal

Figure 12 shows the results of raindrop removal of a few methods, along with the groundtruth. The results include those of Eigen *et al.*'s [11] and You *et al.*'s [17]. Roser *et al.*'s method does not provide the implementation details for raindrop

Fig. 12. The raindrop removal results.

removal, and thus it was not included. As can observed in the figure, our method removed both thin and thick raindrops. Eigen *et al.*'s method failed to remove large raindrops and it erroneously smoothed textured area. You *et al.*'s method failed to remove thin raindrops, and the quality is affected by the detection accuracy.

Repaired Motion Field. Figure 13 shows the results of the motion field estimation, before and after the raindrop removal. As shown in the figure, our method can improve the dense motion estimation, by removing the raindrops, and then repairing the motion fields.

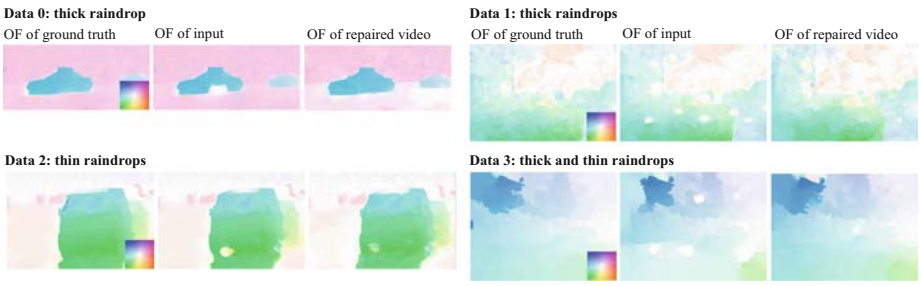

Fig. 13. Comparison on motion field estimation before and after raindrop removal.

7 Conclusion and Future Work

We have introduced a method that automatically detects and removes both thick and thin raindrops using a local operation based on the long trajectory analysis. Our idea is using the motion and appearance features that are extracted from analyzing the trajectories-raindrops encountering events. These features are transformed into a labeling problem which is efficiently optimized in the framework of MRFs. The raindrop removal is performed by utilizing patches indicated by trajectories, enabling the motion consistency to be preserved. We believe our algorithm can be extended to handle other similar occluders, such as dirt or dust. For future work, we consider exploring dense-trajectory analysis of dynamic raindrops and improving the computation time.

Acknowledgement. This work is supported by Next-generation Energies for Tohoku Recovery (NET), MEXT, Japan.

References

1. Tan, R.: Visibility in bad weather from a single image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
2. Fattal, R.: Single image dehazing. In: SIGGRAPH (2008)
3. He, K., Sun, J., Tang, X.: Single image haze removal using dark channel prior. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
4. Meng, G., Wang, Y., Duan, J., Xiang, S., Pan, C.: Efficient image dehazing with boundary constraint and contextual regularization. In: ICCV (2013)
5. Barnum, P., Narasimhan, S., Kanade, T.: Analysis of rain and snow in frequency space. Int. J. Comput. Vis. (IJCV) **86**, 256–274 (2010)
6. Garg, K., Nayar, S.: Vision and rain. Int. J. Comput. Vis. (IJCV) **75**, 3–27 (2007)
7. Chen, Y.L., Hsu, C.T.: A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In: ICCV (2013)
8. Garg, K., Nayar, S.: Photometric model of a raindrop. CMU Technical report (2003)
9. Garg, K., Nayar, S.: Detection and removal of rain from video. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2004)
10. Kang, L., Lin, C., Fu, Y.: Automatic single-image-based rain streaks removal via image decomposition. IEEE Trans. Image Process. (TIP) **21**, 1742–1755 (2012)
11. Eigen, D., Krishnan, D., Fergus, R.: Restoring an image taken through a window covered with dirt or rain. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 633–640 (2013)
12. Kurihata, H., Takahashi, T., Ide, I., Mekada, Y., Murase, H., Tamatsu, Y., Miyahara, T.: Rainy weather recognition from in-vehicle camera images for driver assistance. In: IEEE Intelligent Vehicles Symposium (2005)
13. Yamashita, A., Tanaka, Y., Kaneko, T.: Removal of adherent water-drops from images acquired with stereo camera. In: IROS (2005)
14. Yamashita, A., Fukuchi, I., Kaneko, T.: Noises removal from image sequences acquired with moving camera by estimating camera motion from spatio-temporal information. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2009)

yousd@cvl.iis.u-tokyo.ac.jp

15. Roser, M., Geiger, A.: Video-based raindrop detection for improved image registration. In: Workshops of IEEE International Conference on Computer Vision (2009)
16. Roser, M., Kurz, J., Geiger, A.: Realistic modeling of water droplets for monocular adherent raindrop recognition using bezier curves. In: Asian Conference on Computer Vision (ACCV) (2010)
17. You, S., Tan, R.T., Kawakami, R., Ikeuchi, K.: Adherent raindrop detection and removal in video. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2013)
18. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2004)
19. Willson, R.G., Maimone, M., Johnson, A., Scherr, L.: An Optical Model for Image Artifacts Produced by Dust Particles on Lenses. Jet Propulsion Laboratory, National Aeronautics and Space Administration, Pasadena, CA (2005)
20. Zhou, C., Lin, S.: Removal of image artifacts due to sensor dust. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2007, pp. 1–8. IEEE (2007)
21. Gu, J., Ramamoorthi, R., Belhumeur, P., Nayar, S.: Removing image artifacts due to dirty camera lenses and thin occluders. ACM Trans. Graph. (TOG) **28**, 144 (2009)
22. Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. Int. J. Comput. Vis. (IJCV) **80**, 72–91 (2008)
23. Sundaram, N., Brox, T., Keutzer, K.: Dense point trajectories by GPU-accelerated large displacement optical flow. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 438–451. Springer, Heidelberg (2010)
24. Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **33**, 500–513 (2011)
25. Volz, S., Bruhn, A., Valgaerts, L., Zimmer, H.: Modeling temporal coherence for optical flow. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 1116–1123. IEEE (2011)
26. Rubinstein, M., Liu, C., Freeman, W.T.: Towards longer long-range motion trajectories. In: British Machine Vision Conference (BMVC), pp. 1–11 (2012)
27. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
28. Fulkerson, B., Vedaldi, A., Soatto, S.: Class segmentation and object localization with superpixel neighborhoods. In: IEEE International Conference on Computer Vision (ICCV) (2009)
29. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)s 26, 1124–1137 (2004)
30. Boykov, Y., Veksler, O., Zabih, R.: Efficient approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **20**, 1222–1239 (2001)
31. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **26**, 147–159 (2004)
32. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)