Video Stabilization for Strict Real-Time Applications

Article *in* IEEE Transactions on Circuits and Systems for Video Technology · January 2016

DOI: 10.1109/TCSVT.2016.2589860

2 authors:

Jing Dong
National University of Defense Technology
15 PUBLICATIONS **69** CITATIONS
SEE PROFILE

HaiBo Liu
Hunan University
60 PUBLICATIONS **476** CITATIONS
SEE PROFILE

Video Stabilization for Strict Real-time Applications

Jing Dong, Haibo Liu

***Abstract*—Offline or deferred solutions are frequently employed for high quality and reliable results in current video stabilization. However, neither of these solutions can be used for strict real-time applications. In this paper, we propose a practical and robust algorithm for real-time video stabilization. To achieve this, a novel and efficient motion model based on inter-frame homography estimation is proposed to represent the video motion. An important feature of the proposed motion model is that it updates at each frame input to reduce the accumulation errors caused by parallax or scene changes. We also propose a novel Kalman filter for the motion smoothing and a unique mosaic algorithm for the video completion. The proposed Kalman filter and mosaic algorithm enable the development of a practical real-time video stabilizer that not only produces steady video but also retains the full resolution of the original video. We verify the proposed algorithm through a broad range of video sequences that demonstrate that the proposed algorithm is computationally efficient while being able to robustly stabilize videos with various challenges.**

***Index Terms*—video stabilization, motion estimation, optical flow, normalized cross correlation, frame orbits, Kalman filter, mosaic.**

I. Introduction

Video filmed on hand-held or vehicle-mounted cameras frequently suffers from annoying jitters owing to the unsteady motion of the platform. Video stabilization is the process of improving video quality by removing the effect of fluctuant motion caused by jittering. The goal can be achieved by employing solutions based on sophisticated sensors and lens systems to modify the manner that the camera receives the input light or using mechanical tools to avoid undesired shakes during recording. Though these approaches are feasible for some specialized applications, the drawback is obvious: they are either overly expensive because of the requirement for sophisticated hardware or are inconvenient owing to the necessity of cumbersome equipment. Conversely, digital video stabilization (DVS) does not require additional hardware nor does it require any knowledge of the capturing device. It provides a convenient and economical solution for various vision tasks such as unmanned aerial vehicle (UAV) exploration [1], robot navigation [2], and video retargeting [3]. In the remaining sections of this paper, we will focus on DVS.

The authors are with the College of Aerospace Science and Engineering, National University of Defense Technology, Changsha 410073, China (e-mail: liuhaibo@nudt.edu.cn).

Offline or deferred solutions are frequently employed by current DVS [4-8]. The commonly used smoothing approaches such as Gaussian low-pass filter [4], smoothing by optimization [5-7], and dual pass filter [8] are used for either offline or deferred DVS. The state-of-the art methods [7, 9-11] typically require sophisticated motion estimation, which is computationally expensive. Consequently, these methods are only applicable to offline DVS because they usually run under 5 fps on an ordinary machine. In fact, offline DVS is used for stabilizing a video only after it has been recorded. Deferred DVS can process online; however, it requires a frame buffer for path planning. Hence, the deferred DVS output frame is always delayed from the input frame. Clearly, neither offline DVS nor deferred DVS can be applied to strict real-time applications. Unquestionably, strict real-time video stabilization is challenging for existing DVS methods.

Motivated by the limits of the current methods and practical demands, we propose a practical and robust algorithm for strict real-time video stabilization. There are three main contributions in our work.

A novel motion model parameterized by frame orbits

Our key observation is that video motion over a short time interval can be approximately modeled by four short frame orbits based on inter-frame homography estimation. A frame orbit is a set of 2-dimensional (2D) points $\{P_j \mid F_s \le j \le F_e\}$ across the frames between the starting frame F_s and ending frame F_e. Each point of the frame orbit can be iteratively estimated as follows,

$$P_j = H_{j-1}P_{j-1} \tag{1}$$

where H_{j-1} is the homography estimated between frame j and $j-1$. An example of a frame orbit is presented in Fig. 1.

A well-known fact is that homographies can model inter-frame motion only when the scene is coplanar or when there is no camera translation. That is, there will be accumulation errors caused by parallax or scene changes if homographies are employed for representing the motion of video that is captured by a camera moving in a non-planar scene. Fortunately, accumulation errors can be substantially reduced if homographies are employed to represent the video motion over a short time interval only. The reason is that we can assume there is virtually no camera translation within a short timeframe. Hence, we update the frame orbits to ensure they account for the motion within a short interval only (The details of the update

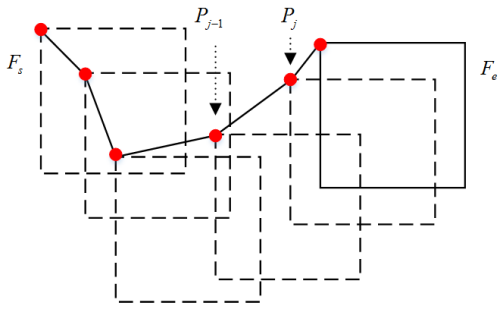

Fig. 1. Frame orbit starting from the top-left corner of the frame rectangle. This frame orbit consists of six points depicted by the six dots. The solid line connecting the dots depicts the trajectory of the frame orbit. The transitions between the dots are estimated based on inter-frame homographies.

process are described in Section III.B). Consequently, the accumulation errors caused by parallax or scene changes are substantially reduced.

Although the proposed method is based on a 2D model in nature, it is superior to traditional 2D methods [1, 12]. Specifically, the traditional 2D methods use 2D transformation matrixes to model the video motion. For decomposing the matrixes into independent components such as rotation, translation, and zoom, they must choose a model with low degrees of freedom (DOF). Consequently, some complicated motions including wobble and shear are ignored by their methods. Conversely, we use four frame orbits, which is a model with eight DOF, to represent the video motion. As the employed motion model has higher DOF and decomposition is avoided, the proposed method is actually more precise and robust than the traditional 2D methods.

The merits of the proposed motion model are further discussed and confirmed in the experimental portion of this document.

A novel Kalman filter for smoothing frame orbits

A Kalman filter is a standard real-time filter. However, the traditional Kalman filter (TKF) cannot smooth a short frame orbit. To decrease the accumulation errors caused by parallax or scene changes, we must use short frame orbits to model the video motion and update them at each frame input. The update causes displacement, which is observed as noise by TKF. Hence, there would be unsteady movements caused by the update in the output of TKF. To solve the problem, we propose an associate Kalman filter (AKF) to smooth the frame orbit. After the frame orbit has been updated, AKF resets its corrected state to compensate for the displacement. Experiments indicate that AKF outperforms TKF when smoothing a short frame orbit.

A highly efficient and robust solution for real-time DVS

The proposed method is highly efficient while being robust to various challenging scenes with light change, motion blur, severe occlusion, or scene change. Furthermore, the proposed method achieves full resolution using a novel backward mosaic algorithm. Lastly, the proposed method does not cause any additional lag (except the time for processing) as it does not require a frame buffer, resulting in a practical solution for telecontrol, which requires that the input image be immediately

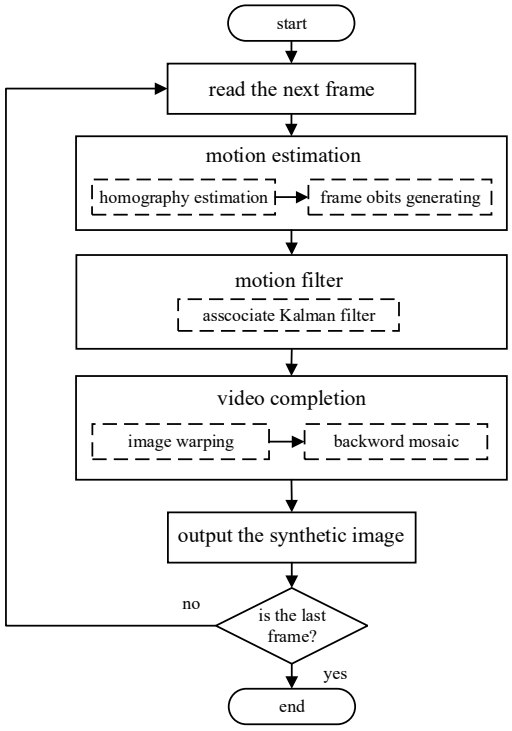

Fig. 2. Flowchart of the proposed algorithm.

stabilized and presented to the operator to request feedback commands.

We developed the proposed algorithm with C++ code. Some basic modules of this algorithm are available from OpenCV (http://opencv.org). To allow our work to be conveniently reproducible, software and testing videos will be released through our website (http://Real-timeDVS.blogspot.com/).

The proposed algorithm consists of three main steps: motion estimation, motion smoothing, and video completion. The flowchart is presented in Fig. 2.

The remainder of this paper is as follows. Section II describes the related works. The motion estimation and motion filter of the proposed algorithm are presented in Section III and Section IV, respectively. Video completion based on image warping and backward mosaic is described in Section V. In Section VI, we conduct a thorough evaluation of the proposed algorithm and compare it with other recent efforts. Finally, the conclusions, limitations, and future work are presented in Section VII.

II. Related Work

According to the motion model employed, DVS can be approximately divided into 2D and three-dimensional (3D) methods. The 2D methods [1, 4, 12, 28] employ 2D transformations to model the video motion and smooth them with a low pass filter. Then, a steady video is generated by warping the image according to the smoothed motion. The 2D methods are robust, efficient, and perform well on videos with distant and static backgrounds. However, they cannot process parallax, which is introduced by capturing a 3D scene using a moving camera.

Recently, Liu *et al*. [7] used a spatially variant 2D model to represent the video motion and smooth it appropriately with an "as-similar-as-possible" regularization constraint. Their method managed parallax well and was tested with a wide range of consumer videos [13]. However, the method is not applicable for strict real-time DVS as it relies on a frame buffer for path optimization.

Early 3D stabilization methods [14, 15, 27] use structure from motion (SFM) [25] to estimate 3D camera motion. After motion smoothing, new images are rendered by homography approximation [15] or content-preserving warp (CWP) [14, 16]. These 3D methods produce superior results to 2D methods when there are significant depth variations in the input video. However, SFM is brittle and time-consuming [9] and cannot manage videos without sufficient depth variations [26].

Feature trajectories is an ideal motion model for video with both 2D and 3D scenes. Lee *et al*. [5] proposed a novel DVS method using robust feature trajectories to model the video motion. However, this method is not full 3D stabilization as the output image is rendered based on 2D global transformation. Liu *et al*. [9] smoothed the feature trajectories with a 9-dimensional subspace constraint based on the observation that a trajectory matrix should have at most rank 9 for video motion within a short time interval [17]. To perform full 3D stabilization, their method also uses CWP to address the image deformation caused by viewpoint change. Consequently, the method achieves superior quality of 3D stabilization without the requirement of 3D reconstruction. However, feature trajectories can be interrupted by motion blur, rapid camera rotation, or severe occlusion, which makes the feature tracking-based methods fragile.

To relieve the requirement of long feature trajectories, Liu *et al*. [6] use a simplified model, a Bézier curve, to represent the feature trajectory and stabilize the input video by performing a spatial-temporal optimization on the curves. This method achieves video stabilization by solving a global optimization that can address short feature trajectories. Most recently, Liu *et al*. [11] proposed a novel DVS method based on SteadyFlow, which is a specific optical flow enforced by strong spatial coherence. Similar to feature trajectories, SteadyFlow can represent spatially variant motion. Hence, the method can stabilize complicate videos without the requirement of long feature trajectories. The methods presented in [6] and [11] are more robust than the methods requiring long feature trajectories. However, both the methods are overly computationally expensive to be implemented for real-time DVS owing to the requirements of enormous feature trajectories or dense optical flow.

Some researches presented online DVS; however, only a small number of them focused on strict real-time DVS. Wang *et al*. [1] used an adaptive Kalman filter to smooth the inter-frame motion. However, their method may introduce significant accumulation errors because they use a 2D similarity model that has only three DOF to represent the video motion. To manage with this, they restart the accumulated motion estimation when detecting a scene change. However, the method works poorly for videos with continuous scene changes because the performance of the Kalman filter is limited if it is restarted frequently. Ryu *et al*. [18] presented real-time video stabilization by filtering the feature trajectories that are directly generated by a KLT tracker [19]. However, their method is not as reliable as the methods presented in [6] and [11] because the feature trajectories are not produced in a reliable manner. Most recently, Dong *et al*. [20] proposed a lightweight DVS for a UAV platform. The method represents video motion using robust motion trajectory that is a longer edition of the frame orbit. As a motion trajectory is smoothed with a linear fitting filter, they must extend the trajectory to 10 - 15 points. Similar to frame orbit, motion trajectory is based on a 2D transformation. Hence, a longer motion trajectory will introduce more severe accumulation errors.

The motion model of the proposed algorithm is partially based on the work presented in [20]. However, we improve their work using considerably shorter orbits to model the video motion and propose a novel motion filter to smooth the orbits appropriately. Further, we develop a backward mosaic algorithm to render a full-frame image in real-time.

III. Frame Orbits

Frame orbit is the motion model employed by the proposed algorithm; this is estimated in two steps: inter-frame homography estimation and frame-orbit generating. In the following subsections, they will be described in detail.

A. Inter-frame Homography Estimation

The state-of-the art methods such as SIFT [22] and SURF [24] can estimate inter-frame homography robustly. However, they are overly computationally expensive for real-time DVS. To address this, we develop a reliable and efficient method for inter-frame homography estimation. The proposed method can manage various challenges including light change, motion blur, and severe occlusion and is considerably faster than other state-of-the art methods.

The proposed homography estimation is based on the point correspondences between two adjacent frames. We first detect the feature points using a highly efficient algorithm introduced by Rosten *et al*. [21]. Typically, the number of detected feature points is enormous and the detected points tend to be converged in highly textured regions, which can increase the estimation time and make the result vulnerable to occlusion. To avoid this, we track only some of the feature points and distribute them evenly on the frame. Specifically, the frame is evenly divided into $N \times N$ blocks. For each block, the feature point with the greatest response is selected as the key point to track, where the response is calculated based on the corner strength, which is introduced in [21].

A KLT tracker is an efficient point-tracking algorithm. Unfortunately, it cannot manage excessive light change because a point is tracked under the assumption that the gray of the same point does not change across the frame. To make the proposed method more reliable to light change, we develop a key point-tracking algorithm as a complement to the KLT tracker and template matching. Specifically, we first track the $N \times N$ key points using the KLT tracker alone. Then, a key point is removed if the gray of its local area changes significantly when

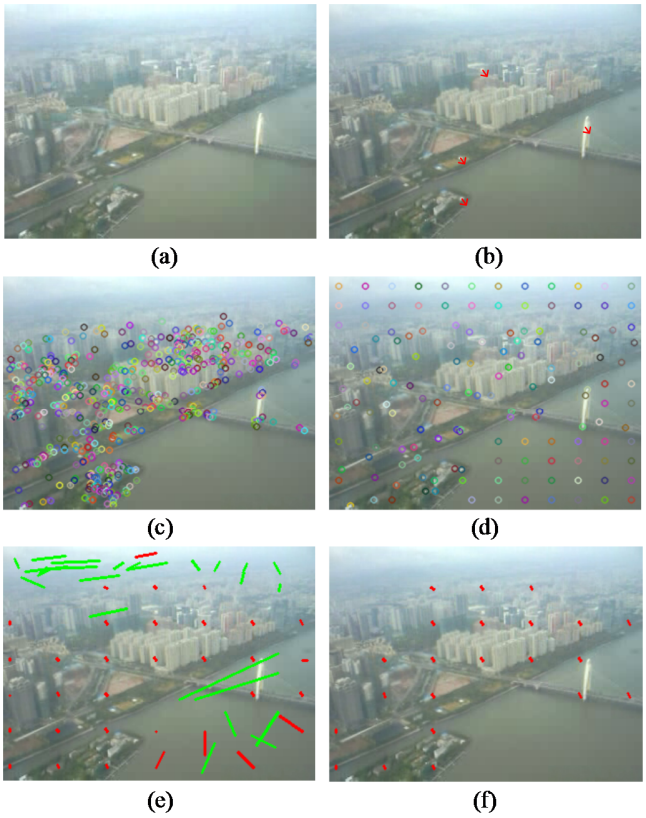

Fig. 3. Process of key-point tracking. Images (a) and (b) are two adjacent frames. The arrows in image (b) indicate the true motion of the inter-frame. The detected feature points and selected key points are depicted in (c) and (d), respectively. In image (e), the green (light color) lines are the results of the KLT tracker; the red (dark color) lines are the results of the NCC template matching. Some result of the KLT tracker are removed because the gray of their local area changes significantly when tracking across frames. Image (f) illustrates that only the results tracked by the NCC template matching are retained after the rejection performed by the RANSAC algorithm.

tracking across frames. Then, the image is evenly divided into $M \times M$ blocks. If all the key points included in one block are removed, the center of the block is selected as the key point and then re-tracked by the template matching. The template matching searches the key point in the next frame using normalized cross correlation (NCC), which is invariant to linear gray change [29]. It is worth noting that NCC template matching is more computationally expensive than the KLT tracker. Therefore, we set M smaller than N if the re-tracking process causes a significant increase in the time cost. In practice, we set $N = 12$ and $M = 7$ for all our experimental results. Following the key-point tracking, we use the RANSAC algorithm [23] to estimate the inter-frame homography with the motion vectors generated by the tracking process. An example of key-point tracking under light change is exhibited in Fig. 3. We can see that the results of the KLT tracker are incorrect owing to the light change. However, many results of NCC template matching are correct and are retained after the rejection performed by the RANSAC algorithm.

In addition to light change, key-point tracking can also fail because of a textureless object or motion blur. Fortunately, increasing the size of the tracking patches can avoid these issues. In all our experimental results, the size of the patches tracked by KLT was set to 13×13 and the size of the patches tracked by the template matching was set to16×16.

TABLE I
PSEUDO CODES FOR FRAME-ORBIT GENERATING

F: *current frame.*
L: *last frame.*
P: *the frame orbit* $\{p_j | \ 0 \le j \le l-1\}$.
l: *the lenth of the frame orbit.*
Q: *the queue keeping the inter-frame homographies.*
$A' - A$: *the displacement for AKF resetting*

```
p_0 ← the top left corner of frame rectangle;
while F is not null {
    A ← p_{l-1} ;
    H_n ← homography from L to F;
    Delete the first homography H_0 in Q;
    Add H_n to the Q ;
    Generate P by transform p_0 with Q ;
    A' ← p_{l-1};
    L ← F;
    F ← next frame;
}
```

B. *Frame-orbits Generating*

After the homography estimation, four frame orbits are generated by transforming the four corners of the frame rectangle with inter-frame homographies. The definition of the frame orbit was presented in Section I.A. For real-time stabilization, the ending frame is the current input frame and the starting frame is $l - 1$ frames before, where l is the length of four frame orbits.

To reduce accumulation errors, we set l to a small constant number and update the four frame orbits at each frame input. An example of the update process is displayed in Fig. 4. We can see that the frame orbit is regenerated at each frame input and always starts from the top left corner of the frame rectangle, as indicated in Figs. 4 (a) and (b). When generating the current frame orbit, the old homography H_0, which is used to transform the first point to the second point in the last frame orbit, is deleted, whereas the remaining homographies and the new homography H_n, which is estimated between the current frame and the last frame, are used to generate the current frame orbit. If we place the two frame orbits in the same coordinates, as in Fig. 4 (c), we can observe that the update process introduces a displacement (from A to A') between the two frame orbits. Table 1 displays the pseudo code for frame orbit generating from the top left corner of the frame rectangle.

IV. ASSOCIATE KALMAN FILTER

As mentioned previously, TKF is not a correct solution for filtering a short frame orbit because the displacement caused by the update will be treated as observed noise, which it actually is not. To correct this, we propose AKF, which compensates for

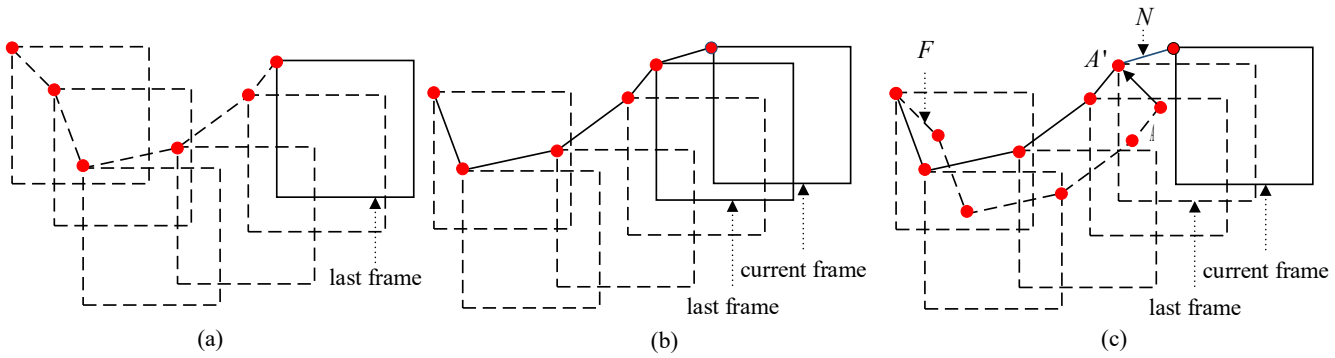

Fig. 4. Frame-orbit generating. For easy viewing, only the orbit starting from the top left corner is depicted. The dashed line indicates the frame orbit generated at the last frame input. The solid line indicates the frame orbit generated at the current frame input. Image (a) depicts the frame-orbit generating at the last frame input. Image (b) is the frame-orbit generating at the current frame input. Image (c) is when the two frame orbits are placed in the same coordinates.

the displacement by resetting the corrected state. It is initiated as follows, which is not different from the standard method.

A. *Initiation of AKF*

Given the increment of the variable at time t, the process model of AKF can be expressed as follows,

$$\begin{pmatrix} \hat{x} \\ \hat{x}^v \\ \hat{y} \\ \hat{y}^v \end{pmatrix}^t = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \hat{x} \\ \hat{x}^v \\ \hat{y} \\ \hat{y}^v \end{pmatrix}^{t-1} + \begin{pmatrix} 0 \\ N(0,\sigma_x) \\ 0 \\ N(0,\sigma_y) \end{pmatrix} \quad (2)$$

where the translation of the frame orbit is described by the variables $\hat{x}$ and $\hat{y}$; the velocity variables $\hat{x}^v$ and $\hat{y}^v$ are auxiliary variables that are not observed. The observation model for each parameter is independent, leading to the observation model as follows,

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} + \begin{pmatrix} N(0,\tilde{\sigma}_x) \\ N(0,\tilde{\sigma}_y) \end{pmatrix} \quad (3)$$

The variances of the processing noise σ_x *and* σ_y are initialized by a small constant value to limit the randomness of the motion estimation. We assume the vibration amplitude of a video is proportional to the frame size owing to the fact that for two videos captured by a camera moving in the same unsteady manner, the video with the higher resolution exhibits greater shake compared with the video with lower resolution. Therefore, we initialize the variances of measurement noise $\tilde{\sigma}_x$, $\tilde{\sigma}_y$ as follows,

$$\tilde{\sigma}_x^2 = (w - c \times w)^2/4, \quad \tilde{\sigma}_y^2 = (h - c \times h)^2/4 \quad (4)$$

where the size of the frame is $w \times h$ and the variable c is a constant value in the range (0, 1). In practice, we set $c = 0.9$ and $\sigma_x^2 = \sigma_y^2 = 0.1$ in all our experimental results.

B. *Corrected state resetting*

The difference between TKF and AKF is that the corrected state of AKF will be reset after the frame orbit has been updated.

We assume that the update process does not change the velocity variables because the moving pattern of the frame orbit is not changed by the update, as indicated in Fig. 4 (c). Thus, we must only reset the translation parameters of the corrected state according to the displacement from A to A':

$$\begin{pmatrix} \hat{x}' \\ \hat{y}' \end{pmatrix} = \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} + (A' - A) \quad (5)$$

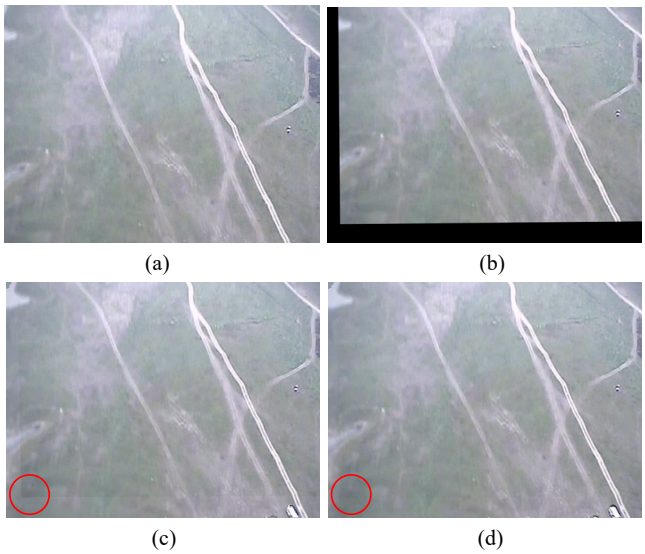

Fig. 5. Comparison of the two mosaic algorithms: (a) is the input image, (b) is the result of image warping, (c) is the result of Matsushita *et al.* [4], and (d) is the result of the proposed method. The circle highlights the mosaicking boundary. We can observe the result achieved by the proposed method is more seamless.

where $\hat{x}$ and $\hat{y}$ are the translation parameters of the corrected state.

C. *Predicting and updating*

After the resetting, AKF will predict the motion with the newest observed value Z_t

$$\hat{X}_t = \Phi \hat{X}_{t-1} + K_t\left[Z_t - H_t \Phi \hat{X}_{t-1}\right] \quad (6)$$

where K_t is the matrix of the Kalman gain, Φ is the transition matrix of states in equation (2), and H_t is given as the following,

$$\mathrm{H}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (7)$$

A more specific description of the updating of K_t can be found in [30], which does not differ from the standard Kalman filter.

V. Video Completion

To generate the stabilized video, the image must be warped

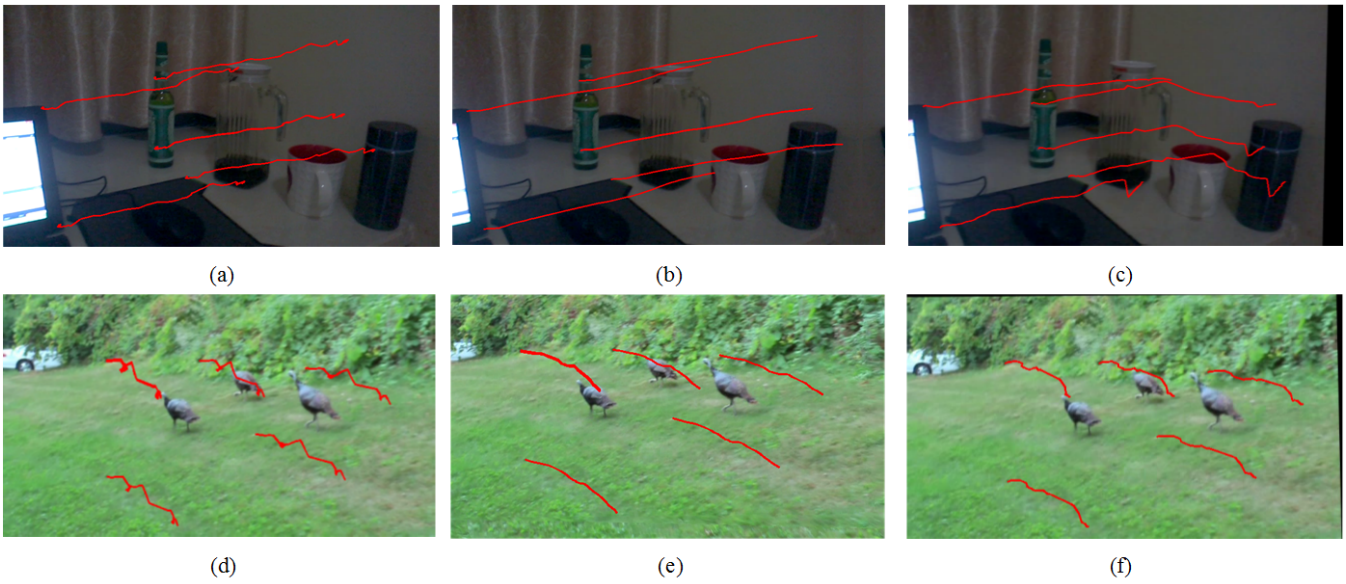

Fig. 6. Two examples of stabilizing videos with challenging cases. The first row illustrates the images with feature point trajectories from a video with significant light change. The second row displays the images with feature point trajectories from a video with excessive motion blur. (a) and (d) are the feature point trajectories of the original videos. (b) and (c) are the feature point trajectories from the results of the proposed algorithm. (c) and (f) are the feature point trajectories from the results of the algorithm introduced by Ryu *et al*. [18]. We can see that the results achieved with the proposed algorithm are smoother. More information regarding the algorithm introduced by Ryu *et al*. is provided in Section VI-D.

Fig. 7. Example of filtering frame orbits and feature trajectories. The red (dark color) lines are the frame orbits and the green (light color) lines are the feature trajectories. The video is provided by Liu *et al*. [26]

according to the smoothed motion, which will leave unfilled areas in the result. An example is displayed in Fig. 5 (b). The consistency mosaic presented by Matsushita *et al*. [4] is an effective method for removing the unfilled areas. However, it is not a real-time solution because it synthesizes the output image by sweeping the video in both a forward and backward direction. Rather, we develop a real-time mosaic algorithm that renders the output image by sweeping the video in a backward direction only. However, the proposed method leaves more unfilled areas in the synthetic image compared with the consistency mosaic. To relieve the visual unpleasantness, we interpolate the remaining unfilled areas with a reflected copy. The motion inpainting introduced in [4] can address the residue caused by a moving object or parallax; however, it is not employed in the proposed algorithm as it requires dense optical flow, which is considerably time consuming.

The consistency mosaic presented in [4] can cause a seam-line in the mosaicking boundary if the estimation of the global transformation is not sufficiently precise. To address this seam-line, we generate the value of a pixel p_t in the missing area using a weighted average of the warped pixels $\{H_t^{t'} p_t\}$ that are included in the neighboring frames:

$$I(p_t) = \frac{\sum_{t'} w(|t-t'|) I(H_t^{t'} p_t)}{\sum_{t'} w(|t-t'|)} \quad (8)$$

where t and t' represent the indexes of the frame and $H_t^{t'}$ is the homography transform from frame t to frame t'.

It is reasonable to assume that the accumulation error increases as the homographies cascade. Therefore, we calculate the weight $w(|t-t'|)$ as follows,

$$w(|t-t'|) = e^{-|t-t'|} \quad (9)$$

According to (8) and (9), the warped pixel in the frame closer to the current frame contributes more to rendering the unfilled pixel.

A comparison of the two mosaic algorithms is presented in Fig. 5. We can observe that the result of the proposed approach (in the right image) is more seamless compared to the other method.

VI. Results and Discussion

We tested the proposed algorithm on a desktop machine with a Core i5 3.6 GHz CPU and GeForce GTX760 GPU. Our implementation did not exploit the multicore processing. Table 2 presents the timing statistics for the proposed algorithm without GPU support. A major portion of the time cost is due to the global transformation estimation. This time cost can be reduced by utilizing fewer key points to track or by decreasing the size of the tracking patches. For high-resolution video, the image warping and mosaic consume the majority of the time. Fortunately, image warping and mosaic are per-pixel rendering solutions that can be easily parallelized with a GPU. In fact, GPU-accelerated modules for feature-point detection, KLT tracker, template matching, and image warping are conveniently available from OpenCV. Except that, a GPU-accelerated module is realized for the mosaic of the proposed

algorithm. All those modules are employed for parallelizing the proposed algorithm with GPU. Table 3 presents the timing statistics for the proposed algorithm with GPU support.

We conducted a thorough evaluation using a broad range of video sequences to demonstrate the effectiveness of the proposed algorithm. The results can be reviewed in the supplementary material or on our website. In the following section, we first evaluate the motion model and motion filter of the proposed algorithm and then compare the entire approach with other methods.

A. Robust Global Transformation Estimation

We estimate the global transformation under the assumption that the majority of the selected key points belong to a static planar background. Therefore, the transformation estimation will fail when encountering a severe occlusion. For addressing strong occlusions, we use the inlier ratio and the root mean squared error (RMSE) of the inliers to determine if the transformation estimation failed. If the estimation failed, we allow AKF to predict the current motion with the last observed data. Hence, the prediction will maintain stable video output for a period, even if there is a global occlusion in the video.

Except for severe occlusion, our global transformation estimation can be adapted to the other challenges described in Section III. Fig. 6 displays two examples of still images; the results of stabilizing those videos using the proposed method are provided in *challenges.mp4*.

B. Simplified Motion Model

For a video with non-coplanar scenes, one can use feature trajectories to model the video motion precisely. However, to obtain long feature trajectories requires significant work [26].

Frame orbits may be overly simple to model the motion for a video with non-coplanar scenes; however, we can filter frame orbits to smooth the video motion, though the orbits may not be able to represent the video motion precisely. Fig. 7 helps us to understand how this works. The left image indicates that both the frame orbits and feature trajectories start from the same point; however, they do not move in the same paths because the points are non-coplanar. However, after filtering the frame orbits and compensating the video sequence using global transformation based on homographies accordingly, the right image illustrates that both the frame orbits and feature trajectories have become smooth. This result confirms that we can filter the frame orbits to smooth the feature trajectories implicitly. That is, we can filter frame orbits to smooth the video motion.

Compared with the motion model based on a greater number of feature trajectories, the proposed motion model reduces the computation requirements enormously, which is a significant advantage for real-time DVS.

C. AKF vs. TKF

We compared AKF with TKF using two different video sequences. For the video captured by a static camera, we used long frame orbits (extended from the first frame to the last frame) to estimate the video motion. In this case, the results of AKF and TKF were comparable because neither filter was interrupted by the update of frame orbits. However, for a video with continuous scene changes or parallax, we were required to use short frame orbits (length of three) to estimate the video motion and update the orbits at each frame input. In this case, AKF was considerably more effective because TKF could not process the displacements introduced by the update. The video results are provided in *filter_comparison.mp4*.

TABLE II
TIME STATISTICS FOR THE PROPOSED ALGORITHM WITHOUT GPU SUPPORT

Resolution	Global Transformation Estimation	Image Warping and Mosaic	Total
320 × 240	11.1 ms	10.2 ms	22.1 ms
640 × 480	14.3 ms	24.3 ms	39.7 ms
1280 × 720	19.5 ms	48.9 ms	69.4 ms
1920 × 1080	32.4 ms	111.6 ms	144.9 ms

TABLE III
TIME STATISTICS OF OUR ALGORITHM WITH GPU SUPPORT

Resolution	GLOBAL TRANSFORMATION ESTIMATION	Image Warping and Mosaic	Total
320 × 240	2.6 ms	0.6 ms	4.0 ms
640 × 480	4.3 ms	2.1 ms	7.3 ms
1280 × 720	9.5 ms	5.9 ms	16.2 ms
1920 × 1080	15.2 ms	11.6 ms	27.7 ms

D. Comparison

We defined an objective metric to measure the smoothness of the results produced by the different methods. Specifically, we first used feature trajectories to represent the video motion and then estimated the speed variation of each trajectory as follows,

$$V(x,y) = \sum \sqrt{(x_{j+1} - 2x_j + x_{j-1})^2 + (y_{j+1} - 2y_j + y_{j-1})^2} \tag{10}$$

where (x_j, y_j) is the position of point (x, y) at frame j in the feature trajectory. Then, the average normalized decrease of speed variation was calculated to evaluate the smoothness:

$$smoothness = average\left(\frac{|V_o - V_r|}{V_o}\right) \tag{11}$$

where V_o is the speed variation estimated by the original video and V_r is the speed variation estimated by the output video.

We re-implemented two real-time DVS algorithms that were introduced by Wang *et al*. [1] and Ryu *et al*. [18] and compared them with the proposed algorithm. Fig. 8 presents samples of the tested videos and the smoothness of the output videos stabilized by the three compared algorithms. The algorithm introduced by Wang *et al*. failed to stabilize the "foodcore" video because this algorithm cannot process the continuous scene change presented in the video, as mentioned in Section 2. For the other videos, the smoothness of the result achieved was inferior to the proposed method because the motion model of their algorithm is based on similarity transform, which cannot account for complicated motion such as wobble and shear. The algorithm introduced by Ryu *et al*. is based on smoothing feature trajectories with a Kalman filter. However, the geometric correlation between the feature points can be broken because each trajectory is filtered independently. Hence, their method can create unacceptable output for video with severe depth variation such as the "foodcore" video. Moreover, the

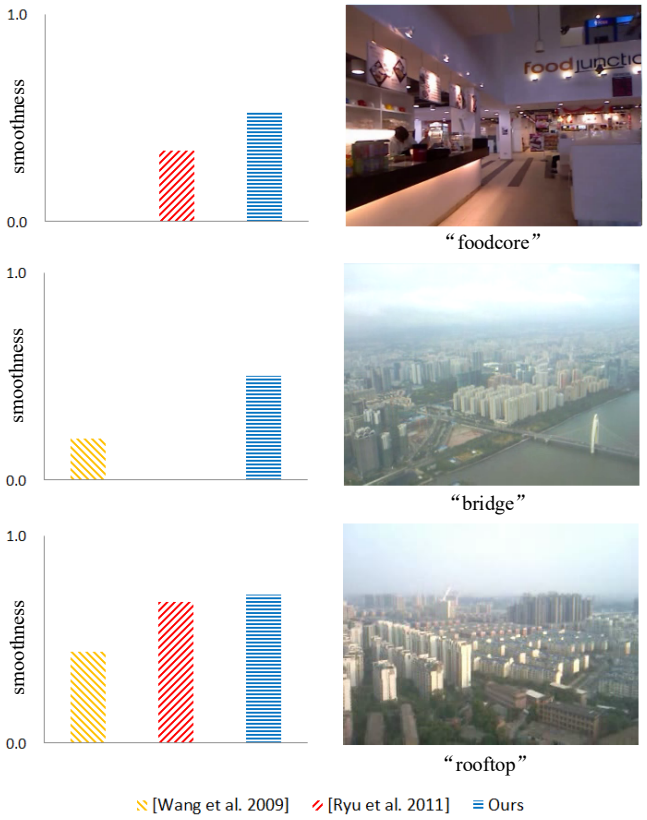

Fig. 8. Quantitative results of three compared real-time DVS algorithms. The result with higher smoothness is superior.

feature trajectories employed by their algorithm are produced by the KLT tracker, which is sensitive to light change. Consequently, their algorithm failed to stabilize the "bridge" video, which has excessive light change.

We also compared the proposed algorithm to some recent efforts [7, 9, 10, 14]. Fig. 9 indicates that all the algorithms significantly decreased the speed variation of the input videos. In general, the proposed method could not achieve results as steady as the other algorithms; however, the proposed algorithm was the only method in the comparison that provided a real-time solution and was computationally efficient.

Finally, we compared our method with the state-of-the art method introduced by Matsushita *et al*. [4], which is based on a deferred solution. The input video was captured by a UAV. The compared result indicated that the deferred solution caused evident lags. For telecontrol, the lags could influence the judgement of the operator and cause inappropriate operations.

All the video results of the comparisons discussed above are provided in *comparisons.mp4*.

VII. Conclusion, Limitations, and Future Work

Real-time video stabilization is a challenging issue for current DVS methods owing to the time limitation and requirement for high reliability. The resolution to this issue is an efficient solution consisting of a simple, yet robust motion model and a novel real-time motion filter. This approach was proposed in this work. The proposed method can offer real-time stabilizing for a wide range of videos, which is rarely achieved by the existing methods.

Because of the real-time constraint, the proposed method cannot detect motion change in advance. Therefore, large unexpected motion changes such as fast camera rotation or rapid zoom may cause evident artifacts in the output video. Further, the proposed method is 2D DVS in nature. Although the proposed method can be adaptive to a scene with moderate depth variation, it may cause wobble or distortion for video with strong parallax. Failed examples are illustrated in *limitations.mp4*.

In a future work, we will attempt to improve the proposed method with a spatially variant motion model that can achieve superior results for video with strong parallax.

References

[1] Wang, C., Kim, J. H., Byun, K. Y., Ni, J., and Ko, S. J., "Robust digital image stabilization using the Kalman filter," *IEEE Trans. Consumer Electronics*, vol.55, no. 1, pp. 6–14, Feb. 2009.

[2] R. Kurazume and S. Hirose, "Development of image stabilization system for remote operation of walking robots," in Proc. ICRA, Apr. 2000, vol. 2, pp. 1856–1861.

[3] Grundmann, M., Kwatra, V., and Essa, I., "Auto-directed video stabilization with robust L1 optimal camera paths," in Proc. CVPR, Jun. 2011, pp. 225–232.

[4] Matsushita, Y., Ofek, E., Ge, W., Tang, X., and Shum, H. Y., "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.28, no. 7, pp. 1150–1163, Jul. 2006.

[5] Lee, K. Y., Chuang, Y. Y., Chen, B. Y., and Ouhyoung, M., "Video stabilization using robust feature trajectories," in Proc. ICCV, Sept. 2009, pp. 1397–1404.

[6] Wang, Y. S., Liu, F., Hsu, P. S., and Lee, T. Y., "Spatially and temporally optimized video stabilization," *IEEE Trans. Visualization and Computer Graphics*, vol.19, no. 8, pp. 1354–1361, Aug. 2013.

[7] Liu, S., Yuan, L., Tan, P., and Sun, J., "Bundled camera paths for video stabilization," *ACM Trans. Graph*, vol.32, no. 4, pp. 78, 2013.

[8] Pan, P., Minagawa, A., Sun, J., Hotta, Y., and Naoi, S., "A dual pass video stabilization system using iterative motion estimation and adaptive motion smoothing," in Proc. ICPR, Aug. 2010, pp. 2298–2301.

[9] Liu, F., Gleicher, M., Wang, J., Jin, H., and Agarwala, A., "Subspace video stabilization," *ACM Trans. Graph*, vol.30, no. 1, pp. 4, 2011.

[10] Goldstein, A., Fattal, R., "Video stabilization using epipolar geometry," *ACM Trans. Graph*, vol.31, no. 5, pp. 126, 2012.

[11] Liu, S., Yuan L., Tan P., Sun J., "SteadyFlow: Spatially Smooth Optical Flow for Video Stabilization," in Proc. CVPR, 2014.

[12] Kumar, S., Azartash, H., Biswas, M., and Nguyen, T., "Real-time affine global motion estimation using phase correlation and its application for digital image stabilization," *IEEE Trans. Image Processing*, vol.20, no. 12, pp. 3406–3418, Dec. 2011.

[13] Video database, http://137.132.165.105/SIGGRAPH2013/database.html

[14] Liu, F., Gleicher, M., Jin, H., and Agarwala, A., "Content-Preserving Warps for 3D Video Stabilization," *ACM Trans. Graph*, vol.28, no. 3, pp. 44, 2009.

[15] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao., "Video Stabilization Based on a 3D Perspective Camera Model," *The Visual Computer*, vol.25, no.11, pp. 997–1008, 2009.

[16] Zhou, Z., Jin, H., and Ma, Y., "Plane-Based Content Preserving Warps for Video Stabilization," in Proc. CVPR, Jun. 2013, pp. 2299–2306.

[17] Irani, M. "Multi-frame correspondence estimation using subspace constraints," *International Journal of Computer Vision*, vol.48, no. 3, pp. 173–194, 2002.

[18] Ryu, Y. G., Roh, H. C., and Chung, M. J., "Long-Time Video Stabilization Using Point-Feature Trajectory Smoothing," in Proc. ICCE, Jan. 2011, pp. 189–190.

[19] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Proc. IJCAI, 1981, vol.81, pp. 674–679.

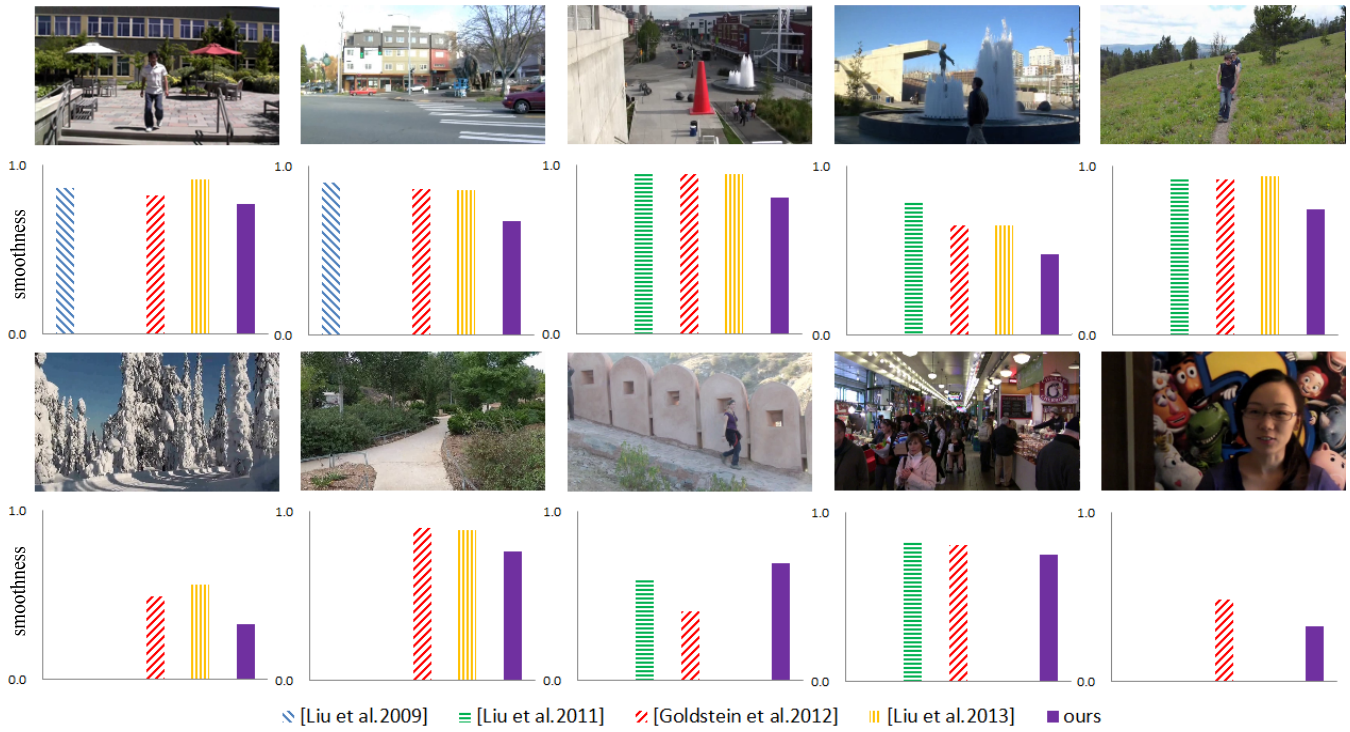

Fig. 9. Quantitative comparisons with other DVS algorithms on publicly available data [10,13]. The result with higher smoothness is superior..

[20] Dong, J., Xia, Y., Yu, Q., Su, A., and Hou, W., "Instantaneous Video Stabilization for Unmanned Aerial Vehicles," *J. Electron. Imaging.,* vol.23, no. 1, 013002, Jan. 2014.

[21] Rosten, E., Porter, R., and Drummond, T., "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.32, no. 1, pp. 105–119, Jan. 2010.

[22] D. G. Lowe. "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol.60, no. 2, pp. 91-110, 2004.

[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol.24, no. 6, pp. 381–395, 1981.

[24] H. Bay, T. Tuytelaars, and G. L. Van. "Surf: Speeded up robust features," in Proc. ECCV, 2006, pp. 404-417.

[25] Hartley, R., and Zisserman, A. "Multiple view geometry in computer vision," Cambridge university press, 2003.

[26] Liu, S., Wang, Y., Yuan, L., Bu, J., Tan, P., and Sun, J. "Video stabilization with a depth camera," in Proc. CVPR, Jun. 2012, pp. 85–95.

[27] Buehler, C., Bosse, M., and McMillan, L., "Non-metric image-based rendering for video stabilization," in Proc. CVPR, 2001, vol.2, pp. 609–614.

[28] Morimoto, C., and Chellappa, R., "Evaluation of image stabilization algorithms," in Proc. ICASSP, 1998, vol.5, pp. 2789–2792.

[29] Lewis, J. P., "Fast normalized cross-correlation," *Vision interface*, vol.10, no. 1, pp. 120–123, 1995.

[30] Kalman R E., "A new approach to linear filtering and prediction problems," Journal of basic Engineering, vol.82, no. 1, pp. 35–45, 1960.

[31] Reilly V, Idrees H, Shah M., "Detection and tracking of large number of targets in wide area surveillance," in Proc. ECCV, 2010.

Jing Dong received his PhD degree from the National University of Defense Technology (NUDT), Changsha, China, in 2015. He is currently a research assistant at College of Aerospace Science and Engineering, NUDT. His research interests include image registration, moving object detection, and video stabilization.

Haibo Liu received his MS and PhD degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2008 and 2012, respectively. He is currently a prelector at College of Aerospace Science and Engineering, NUDT. His research interests include image processing, optics and photoelectricity measurement.

Reply to the comments of the reviewers

Control Number: 9719

Title: Video Stabilization for Strict Real-time Applications

Authors: Jing Dong ; Haibo Liu

I would like to express my sincere thanks to the Associate Editor and the three anonymous reviewers for the comments and suggestions regarding the manuscript. In the following, I will give responses to the issues mentioned by Associate Editor and the reviewers.

Response to Associate Editor:

A table with pseudo codes is added in Chapter III-B to show more details about the frame orbits generating.

Chapter IV-B and Chapter IV-C are revised to explain the proposed Kalman filter in a better way.

We realized our algorithm with GPU support to achieve real-time speed for high resolution and high frame rate sequences, and the results are reported.

The Key references are included.

All Comments of reviewers have been considered and the paper is revised accordingly.

The language of the paper has been improved with professional language editing.

However, it's very hard to make a formalized comparison of time performance with other methods, since there are no available open source for the state-of-the art DVS methods. However, it's worth mention that most state-of-the art DVS methods, such as [5-7, 9-11], employ sophisticated motion estimation methods with complicated feature detection, matching, tracking and optimization process (as we introduced in chapter II). On the contrary, our method only requires a light-weight motion estimation method based on about 150 times KLT Track, 10 times NCC template matching(the pattern size and search space is small) and a RANSAC algorithm performing on about 100 candidates. Therefore, we believe that our algorithm will be much faster than those state-of-the art DVS methods if it realized properly.

Indeed, some DVS software, such as 'Deshaker'（ http://www.guthspot.se/video/deshaker.htm） are available. But we believe that the comparison between our algorithm and the DVS software hardly gives meaningful information. The DVS software, like 'Deshaker', could be implemented very efficiently with powerful optimization techniques, while our method just try decrease the computational complexity of the algorithm. If our algorithm doesn't compete with 'Deshaker' on time performance, that doesn't means our

algorithm is not suitable for real-time DVS. It worth mention that 'real-time' also means low-delay which most current DVS methods can't achieve because they need a frame buffer for read-ahead. On the contrary, read-ahead is not required by our method, which makes our method suitable for strict real-time DVS.

The theoretical contribution of this work may be not as significant as the state-of-the art works, but the work does make breakthrough that it brings the DVS to strict real-time field. A fast DVS algorithm with low-delay and full-frame output is very valuable to strict real-time applications.

Response to reviewer 1:

Thank you very much for your advices. The answers for the problems with your concern are given in the following:

1. "how the method is different from the standard feature based video alignment pipeline."

Answer：

Chapter II gives an introduction about dividing DVS methods in to different types. We can see almost all the DVS methods are based on feature. However, there are some DVS methods based on feature trajectories, unlike other methods, they don't estimate 2D or 3D motion model. The pipeline of the DVS method based on feature trajectories typically consisted of 3 steps: estimating the feature trajectories of original video, smoothing the estimated feature trajectories, and rendering the output video according to the smoothed feature trajectories.

Our method is based on 2D motion model, which means it doesn't estimate feature trajectories, but the estimation of 2D motion model is required. The pipeline is given in Figure 2 of the paper.

The methods based on feature trajectories have a vital shortcoming that robust estimation of long feature trajectories is very computationally expensive. For the purpose of real-time DVS, we make our method based on 2D motion model which can be estimated much more efficiently.

2. "How much are the changes proposed contributing to performance in smoothness and time?"
3. "there is not enough formalized comparison of time performance with other methods."

Answer：

We answer the question 2 and 3 together.

We compare our algorithm with two other real-time DVS algorithms which are introduced by Wang et al. [1] (based on 2D motion model) and Ryu et al. [18] (based on feature trajectories). Figure 8 of the paper and the Supplementary Material gives the results, we can see that the results of our algorithm is more smoothing.

It's very hard to make a formalized comparison of time performance with other methods, since there are no available open source for the state-of-the art DVS methods. However, it's worth mention that most state-of-the art DVS methods, such as [5-7, 9-11], employ sophisticated motion estimation methods with complicated feature detection, matching, tracking and optimization process (as we introduced in chapter II). On the contrary, our method only requires a light-weight motion estimation method based on about 150 times KLT Track, 10 times NCC template matching(the pattern size and search space is small) and a RANSAC algorithm performing on about 100 candidates. Therefore, we believe that our algorithm will be much faster than those state-of-the art DVS methods if it realized properly.

Indeed, some DVS software, such as 'Deshaker'（ http://www.guthspot.se/video/deshaker.htm） are available. But we believe that the comparison between our algorithm and the DVS software hardly gives meaningful information. The DVS software, like 'Deshaker', could be implemented very efficiently with powerful optimization techniques, while our method just try decrease the computational complexity of the algorithm. If our algorithm doesn't compete with 'Deshaker' on time performance, that doesn't means our algorithm is not suitable for real-time DVS. It worth mention that 'real-time' also means low-delay which most current DVS methods can't achieve because they need a frame buffer for read-ahead. On the contrary, read-ahead is not required by our method, which makes our method suitable for strict real-time DVS.

The theoretical contribution of this work may be not as significant as the state-of-the art works, but the work does make breakthrough that it brings the DVS to strict real-time field. A fast DVS algorithm with low-delay and full-frame output is very valuable to strict real-time applications.

Response to reviewer 2:

Thank you very much for your advices. The answers for the problems with your concern are given in the following:

1. "Remarks about Figure 3 : (1) Between (e) and (f) : in (e), results of KLT tracker; in (f) are remaining only the KLT vectors that passed the test of grey level change ? (2) Is there an interest to keep KLT tracker ? All the vectors seem to be wrong. (3) Are the red vectors in (f) really resulting from NCC ? (4) No result with RANSAC algorithm ?"

Answer：

Figure 3 is changed. Image (e) gives the tracking results of KLT tracker (the green lines) and the tracking results of NCC template matching (the red lines, and they are really resulting from NCC template matching). Image (f) shows that only the results tracked by NCC template matching are kept after the rejection performed by RANSAC algorithm.

The results tracked by KLT may be wrong even if they passed the test of gray level. That's a reason why we need the re-track process with NCC template matching.

2. "The descriptions in III-B and IV are heavy and not clear…"

Answer：

Figure 4 is revised, and more complement is add to the text of Figure 4. "The dash line shows the frame orbit generated at last frame input. The solid line shows the frame orbit generated at current frame input."

Also, a table with pseudo codes is added in Chapter III-B to show more details about the frame orbits generating.

Four frame orbits are the point trajectories generated from the four corners of the image rectangle.

IV-B and IV-C are revised to explain the proposed Kalman filter in a better way. Indices t and (t-1) are also used in equation (6) and (7), which show the prediction and update process of the proposed filter.

The original equation (5) will cause misunderstanding, so, it's revised.

3. "Sentence : "The consistency mosaic presented in [4] will cause seam-line in the mosaicking boundary if the estimation of global transformation is not precise enough." Why would you have better results as the quality of your method also depends on homography accuracy?"

Answer：

The removing of seam-line is not just depended on the accuracy of image matching, the synthesis process is also vital. Both our method and the mothed proposed in [4] synthesis one unfilled pixel with multiple warped pixels which are included in the neighboring frames. However, the mothed proposed in [4] simply choose the median of warped pixels to synthesis the unfilled pixel, which seems not enough for seam-line removing. To improve that, we changed the synthesis process. We assume that the accumulation error is increasing as the homographies cascade, so we let the warped pixel in the frame closer to the current frame make more contribution for synthesizing the unfilled pixel (as we descripted in chapter V). As a result, we got better results.

4. "First sentence : ... belongs to a static **planar** background"

Answer：

Thank you very much, the phrase is revised.

5. “Little interest of figure 6, it gives no information about the result. In particular the 3rd line is not clear (unlike the 2 first rows, these are 2 images of different videos). Profiles of point trajectories in these cases would be interesting.”

Answer:

We removed original images of figure 6 and added new images to show the Profiles of point trajectories in the original and processed sequences.

6. “In "challenges.mp4" : the sequence with the train is not unsteady. So, the result just shows that the method is not disrupted by the train motion.”

Answer:

Indeed, but the method will produce serious distortion if the predication strategy (descripted in chapter VI-A) is not employed.

7. “In "comparisons.mp4" : the format of the images is not the same and the border of images has been lost in other methods (e.g. Liu and Goldstein methods).”

Answer:

Video completion is employed by our algorithm, but it's not employed by other methods, so the border of images has been lost in other methods. However, the input videos of all the method are the same.

8. “Video completion comparison with Matsushita: the delay of 30 images is annoying and prevents a relevant comparison.”

Answer:

In that video, we just want show the delay caused by Matsushita’s method. The video completion comparison is given by still images as shown in Figure 5.

9. “It would be useful to show results of tracking some points in original and processed sequences, with your result and with other method(s), for example with the sequences shown in Figure 6.”

Answer:

Figure 6 was changed to show the results of tracking some points in original and processed sequences,

with our algorithm and with the algorithm introduces by Ryu et al. [18].

10. “"...is the position of point j in the feature trajectory" > ...is the position of point (x,y) at frame j in the feature trajectory" ?
    The sum in equation (8) is along the trajectory ? V(x,y) ? ”

Answer:

The phrase is revised and equation (8) is revised to equation (10):

$$V(x,y) = \sum \sqrt{(x_{j+1} - 2x_j + x_{j-1})^2 + (y_{j+1} - 2y_j + y_{j-1})^2}$$

11. Thank you very much for you pointing out the grammar mistakes. The language of the paper has been improved with professional language editing.

Response to reviewer 3:

Thank you very much for your advices. The answers for the problems with your concern are given in the following:

1. “Small errors with tenses ("were" versus "are"; "was" versus "is" should be corrected in a number of places).”

Answer:

Thank you. The language of the paper has been improved with professional language editing.

2. In chapter II, “Citations or sufficient reason should be provided to justify the claim "SFM is brittle and time-consuming, and it can't cope with the videos without sufficient depth variations"”

Answer:

Thank you, we added citations for that claim.

3. “The authors describe the ways in which the KLT point tracker may fail if lighting changes. It could be made clear why it was decided to re-track with a different algorithm (NCC) rather than using NCC throughout, or combining estimates somehow. It is worth pointing out that re-tracking imposes a higher

worst-case cost, and more variation in cost, which is less suitable for HW even if SW performance is adequate."

Answer:

NCC template matching is more computational expensive than KLT track (we add this as complement in the third paragraph of chapter III-A). Therefore, we get less track results or lower speed if we use NCC throughout.

In practice, we found the execution time of the estimation with re-tracking strategy varies less. We believe the reason is the following:

For the combining estimation, both tracking methods will be performed certain times. However, some results of KLT tracker will be deleted before the rejection with RANSAC, which makes the total estimation time varied. Actually the execution time varies significantly for light change case.

For the re-tracking strategy, if the re-tracking is performed, the time of tracking will be longer, but less tracking results will be passed to RANSAC; if the re-tracking is not performed, the time of tracking will be shorter but more tracking results will be passed to RANSAC(because we set M smaller than N as we described in chapter III-A). As a result, the whole time of estimation varied less compared with the combining estimation.

4. "This section is the least clear. It is not very clear what is the difference between the update processes for the KLT and the ALT, and how this can be represented as a coordinate change. More detail on the update process steps would be useful."

Answer:

Figure 4 is revised, and more complement is add to the text of Figure 4. "The dash line shows the frame orbit generated at last frame input. The solid line shows the frame orbit generated at current frame input."

Also, a table with pseudo codes is added in Chapter III-B to show more details about the frame orbits generating.

IV-B and IV-C are revised to explain the proposed Kalman filter in a better way. Indices t and (t-1) are also used in equation (6) and (7), which show the prediction and update process of the proposed filter.

The original equation (5) will cause misunderstanding, so, it's revised.

5. "The algorithm is clearly fast, but on the basis of the execution times presented it is only real-time in SW for low resolution or frame rate sequences, although clearly GPU acceleration could greatly increase that and is no doubt feasible. The authors mention multicore but some remarks of how parallelism might be

implemented and what impact it might have on performance (e.g. due to incomplete context being available in a given thread) would be useful."

Answer:

The motion estimation of our algorithm is very fast, but the warping and video completion processes are very costly, especially for the video in high resolution. Good news is that the warping and video completion processes are typical SIMD processes, so we realized our algorithm with GPU support and reported the results (Relevant content is added in the first paragraph of chapter VI).

If multicores are supported, one can use pipeline to enhance the average throughput. For example, one thread for the motion estimation, one thread for the warping, one thread for video completion. Strictly, the pipeline is not a parallelize process, but the execution times are overlapped. A shortcoming of pipeline is that it introduces delays.

We deleted the content about multicore and pipeline from the first paragraph of chapter VI because we want focus on what we have realized.

6. "The smoothness metric presented clearly favours linearization of the motion trajectories, but one criticism could be that it does not produce zero output for smoothly accelerating motion (i.e. constant second derivative). Directly optimising to minimise this metric could possibly over-smooth some real-life video, including rotational motion."

Answer:

Actually, long-time smoothly accelerating motion is rare in real-life video. Usually case is that there are many successive short smoothly accelerating motion with different or opposite directions in a video, which makes the video unstable. If higher derivatives are used in the smoothness metric, the unstable motion may be ignored or misjudged.

We also employed Gaussian high pass filter to estimate the high frequency energy E and evaluate the smoothness:

$$smoothness = average\left(\frac{|E_o - E_r|}{E_o}\right)$$

There are no much difference in the results of Figure 8 and 9 with the two evaluating ways.

The AKF with acceleration (second derivative) and AKF without acceleration were compared, the results show that AKF without acceleration get more stable output video. For avoiding make the paper too long, we directly presented the better choice. Indeed, higher derivative is employed in different filter, like the filter employ in [3]. The optimization they employed try minimize the following objective:

$$\mathcal{O}(P) = w_1|D(P)|_1 + w_2|D^2(P)|_1 + w_3|D^3(P)|_1$$

where $D^3(P)$ is the second derivative, and w3 is chosen to be an order of magnitude larger than both w1 and w2. That means the major part of the minimizing objective is the first two terms.

We believe that it's very hard to find a quantitative smoothness metric to precisely descript the stable felt by human eyes. So we provided video form results.

In the view of above considerations, we decide to not change the relevant content.