Lee *EURASIP Journal on Image and Video Processing* (2019) 2019:64
https://doi.org/10.1186/s13640-019-0463-2

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

A novel camera path planning algorithm for real-time video stabilization

Yun Gu Lee

Abstract

In video stabilization, a steady camera path plan is as important as accurate camera motion prediction. While several camera path planning algorithms have been studied, most algorithms are used for post-processing video stabilization. Since all of the frames are accessible in post-processing video stabilization, a camera path planning method can generate a good camera path, considering the camera movements at all of the frames. Meanwhile, the number of accessible frames in real-time video stabilization is limited. Thus, smooth camera path planning is a challenging issue in real-time video stabilization. For example, a camera path planner does not know in advance the locations of sudden camera motions, so it is not easy to compensate a sudden camera movement. Therefore, this paper proposes a novel camera path planning algorithm for real-time video stabilization. A camera path planning method should fully utilize the given image margin to provide steady camera paths. In contrast, it should retain the certain amount of an unused image margin for use in frames with dynamic or sudden camera movements. To resolve this problem, the proposed algorithm uses two terms related to the steady camera path and the amount of image margin, and it cross-optimizes the two terms to provide a new camera path on-the-fly. Experimental results show that the proposed algorithm provides excellent performance in real-time video stabilization while requiring negligible computation.

Keywords: Video stabilization, Camera path, Camera trajectory

1 Introduction

With the recent advancement in camera technology, the shooting and sharing of videos with consumer cameras has significantly increased. However, the quality of videos made by amateur users differs from that by professional users. One of the most obvious differences between videos shot by amateur and professional users is the quality gap in terms of video stability. In order to stabilize videos, professional users adopt various professional stabilization hardware tools such as gimbals, steadicams, tripods, and camera dollies. In contrast, the amateur user shoots videos using consumer cameras equipped with only an optical image stabilizer [1] without professional stabilization hardware tools. Hence, video stabilization is crucial to reduce the quality gap between videos shot by amateurs and those by professional users.

Correspondence: harmony96@gmail.com
School of Software, Kwangwoon University, Kwangwoon-ro 20, Nowon-Gu, Seoul, 01897 South Korea

1.1 Video stabilization

Video stabilization mainly consists of three steps: camera motion estimation, new camera path planning, and new image synthesis. In camera motion estimation, the geometric relationship between consecutive frames is analyzed based on the adopted motion model, and the original camera path is predicted. The new camera path is planned by smoothing the original camera path. Finally, the stabilized views on the new camera path are synthesized.

1.1.1 Camera motion estimation

The geometric relationship between consecutive frames should be accurately described to completely remove unwanted camera motion. Hence, the performance of video stabilization is directly related to the accuracy of the adopted motion model, and the main concern of conventional video stabilization techniques is to develop new motion models to accurately describe the geometric relationship between consecutive frames.

Video stabilization is categorized into 2D and 3D approaches according to the motion model. The first approach is 2D video stabilization [2] that describes a relationship between two successive frames using a 2D motion model such as a simple translational model, 2D rigid model, and 2D affine model [3–12]. The 2D methods are fast and robust, compared to the 3D approach. If camera rotations are small along the x and y axes in the 3D space, 2D video stabilization provides good performance [13]. However, as the camera motion becomes dynamic, the 2D motion model cannot describe the geometric relationship between successive frames, and the performance is very limited. Lee at el. proposed a method to directly stabilize a video without explicitly estimating camera motion [14]. This method finds a set of transformations to smooth out feature trajectories [15, 16] and stabilize the video.

The second approach is 3D video stabilization introduced by Buehler et al. [17]. Early 3D approaches reconstruct 3D models of the scene and camera motion using structure-from-motion (SFM) techniques [18]. The stabilized views are then rendered at the new 3D camera path [17, 19]. The 3D video stabilization provides better performance than the 2D approach. Since these 3D methods often fail with videos that lack parallax, they are not practical in many cases [20]. Liu proposed subspace video stabilization that combines the advantages of 2D and 3D video stabilization [21]. However, this method often fails for videos that include dynamic motion due to an insufficient number of long feature trajectories [20]. Hence, Wang [20] proposed video stabilization for a video where 3D reconstruction is difficult or where long feature trajectories are not available. Liu et al. [22] proposed a motion model SteadyFlow that is a specific optical flow by enforcing strong spatial coherence. Grundmann et al. proposed calibration-free rolling shutter removal, based on a mixture model of homographies [23]. Dong proposed video stabilization for real-time applications using a motion model based on inter-frame homography estimation [24]. Ringaby et al. proposed a method for rectifying and stabilizing video by parameterizing camera rotation as a continuous curve [25]. Karpendo et al. proposed video stabilization and rolling shutter correction using gyroscopes [26]. Lee proposed video stabilization based on human visual system [27].

1.1.2 *Camera path planning*

The second step of video stabilization is new camera path planning that removes the annoying irregular perturbations. The simplest method for predicting a new camera path is the application of a low-pass filter (or IIR filter [28]) or Gaussian filter, which suppresses high-frequency jitter in the original camera path. Since the low-pass filter efficiently smooths the original camera path, it is widely used in video stabilization [3, 5, 10–12, 12, 27, 29, 30]. Zhu et al. proposed an inertial motion model [8] for the motion filtering. Litvin et al. [31] proposed a Kalman filter to smooth the camera path. Because of camera shake, the actual camera path includes noise in the intended camera path. These methods [4, 31] estimate the intended camera path using the Kalman filter [32]. Another camera path planning is to model the camera trajectory such as linear and parabolic [29].

Video stabilization methods crop the original view by a cropping window positioned on the new camera path, which is one of the simplest and most robust approaches [21]. If input videos include dynamic camera motion, the cropping windows sometimes will not fit the original frame. In this case, out-of-bound areas will not be visible or complex motion-inpainting will be required [10, 33]. However, motion-inpainting does not provide satisfactory performance in some cases and has a high computational cost. Another approach is to forcibly modify the new camera path to fit the original frame, but compulsory modification of the new camera path will degrade the visual quality of the stabilized videos. In other approaches, the size of output videos can be set so that the cropping window always fits the original frame. While this method significantly reduces the field of view (or the size of output views) for videos having dynamic camera motions, it can maximize dramatic cinematographic effect [34]. Video 1, which can be downloadable on website [35], shows examples of a video having dynamic camera motion. Videos 1(a), 1(b), 1(c), and 1(d) depict an original video, a stabilized video including out-of-bound, a video stabilized by forcibly modifying the new camera path, and a video stabilized by reducing the size of the cropping window, respectively. While Video 1(d) provides the best performance in terms of video stability, the field of view is considerably reduced. Hence, Grudmann et al. proposed an algorithm for automatically applying constrainable, L1-optimal camera paths for post-process video stabilization [36]. Under boundary constraints, the method computes camera paths that are composed of constant, linear, and parabolic segments, in order to mimic camera motions employed by professional cinematographers.

1.1.3 *Image synthesis*

Based on the adopted motion model, the 2D methods synthesize output videos using an interpolation technique. The 3D methods need to employ the view interpolation for rendering output videos. However, the output videos from the view interpolation sometimes include image artifacts like the ghost effect [29]. In order to reduce the image artifacts, new image rendering by content-preserving warp [29, 30] was proposed. The content-preserving warp relaxes the constraint for a physically correct reconstruction while preserving the perceptual quality. Some studies employ a purely rotational motion model without considering translational camera motion, in order to avoid image artifacts from view interpolation [26, 27].

Since they consider only the camera rotation, there is no occlusion issue.

1.2 Real-time processing

Most conventional video stabilization methods have been developed for post-processing used in professional video editing tools. After shooting videos, experts can edit their videos using the professional video editing tools. These methods are necessary in generating high-quality videos for advertisement, film, commercial use, etc. Meanwhile, most users using consumer digital cameras are not good at the professional video editing tools. They usually display, upload, and share videos without modification. Therefore, the performance of real-time video stabilization is very crucial in consumer digital cameras. Fast and efficient video stabilization methods for real-time applications have been studied [5, 12, 24, 27, 37, 38]. However, these methods did not consider camera path planning in depth.

On the other hand, camera path planning for real-time video stabilization has several challenging issues as follows. First, the size of output videos should be determined prior to applying real-time video stabilization, regardless of whether an input video includes static or dynamic camera motions. Thus, the real-time video stabilization cannot adaptively determine the size of the cropping window to fit the original frame according to the given video. Second, a camera path planning algorithm should predict the new camera path on-the-fly. Unlike post-processing video stabilization, the new camera path cannot be updated or modified to gradually improve the quality. Third, it cannot consider camera motions at all of the frames to plan the new camera path. For example, since Grundmann's method obtains the L1-optimal camera paths via linear programming using all of the frames, it cannot be used for real-time video stabilization [36]. If the quality of a planned camera path is not acceptable, although accurate camera motions are predicted, the visual quality of stabilized videos will be significantly degraded. Consequently, in real-time video stabilization, camera path planning is as important as accurate camera motion prediction. As Dong pointed out, most conventional video stabilization methods have been developed for post-processing [24]. Moreover, camera path planning algorithms for real-time video stabilization have not yet been studied in detail. Therefore, in this paper, we propose a new camera path planning algorithm for real-time video stabilization by cross-optimizing two terms related to steady camera path and the amount of image margin. One term renders the new camera trajectory smooth, and the other term retains the certain amount of an unused image margin for use in dynamic or sudden camera motions.

2 Background

For convenience of explanation, this paper describes the proposed algorithm of the camera path planning under simple video stabilization with a 2D translational motion model as depicted in Fig. 1. However, the proposed method can be modified to plan the camera path for other stabilization using a different motion model such as homography.

The original camera path of videos shot by a hand-held camera is usually shaky, as depicted in Fig. 2. To provide steady output videos, it is necessary to predict a smooth camera path from the original camera path. The simplest method for estimating a smooth camera path is to apply a low-pass filter to the original camera path as follows.

$$P_N(t) = \sum_k w(k) P_O(k) \tag{1}$$

Here, $P_O(t)$ and $P_N(t)$ denote the original and new camera positions, respectively. $w(k)$ is the kth coefficient of the low-pass filter. Since the above ideal filter requires an infinite number of signals, it is impossible to realize. In

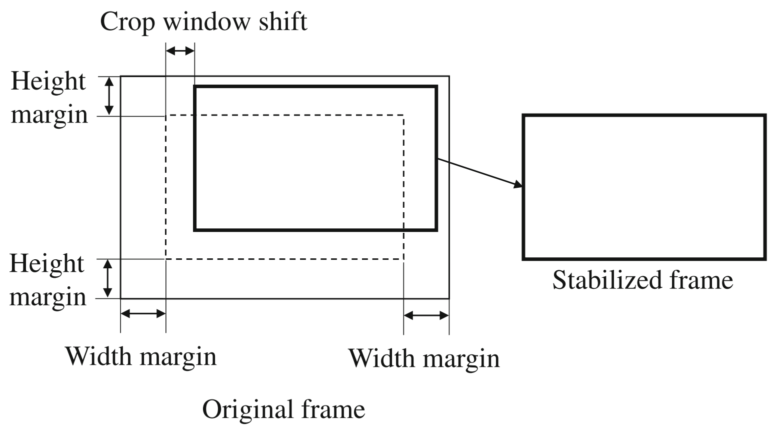

Fig. 1 Simple video stabilization with 2D translational motion model

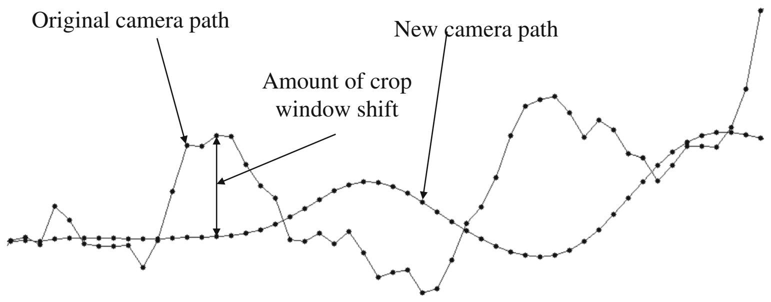

Fig. 2 Original and new camera path

practice, the low-pass filter should consider a finite number of signals. Specifically, a casual filter that depends on camera positions of past and present frames is considered in real-time applications as follows.

$$P_N(t) = \sum_{k=0}^{k=t} w(k) P_O(k) \tag{2}$$

Figure 2 shows the result of applying the casual low-pass filter to a camera path along the x direction. The low-pass filter significantly smoothens the camera path. Here, the difference between the original and new camera paths in Fig. 2 can be interpreted as the amount of the cropping window shift shown in Fig. 1. If the amount of the shift is more than the width or height of the margin, the cropping window will not fit within the original frame. The width and height margins (M_W and M_H) are defined as follows.

$$M_W = \frac{W_I - W_O}{2}, \quad M_H = \frac{H_I - H_O}{2} \tag{3}$$

where W_I, H_I, and W_O, H_O are the width and height of the original and output frames, respectively. Out-of-bound areas will be invisible or require motion-inpainting [10, 33]. The boundaries are depicted in Fig. 3. In the figure, a gray region illustrates an example of the out-of-bound area. For real-time applications, the motion-inpainting is not a feasible solution due to complexity. Hence, the new camera path should be determined so that the cropping window is located inside the original frame. However, the amount of cropping window shift in the method based on the low-pass filter is uncontrollable, and the method does not guarantee that the cropping window will always be located within the original frame. Therefore, this simple method cannot be applied to real-time video stabilization where the size of the output video is fixed.

In post-process video stabilization, the entire original camera path is given before planning the new camera path. Hence, the smooth camera path can be optimally planned so that an out-of-bound area does not occur.

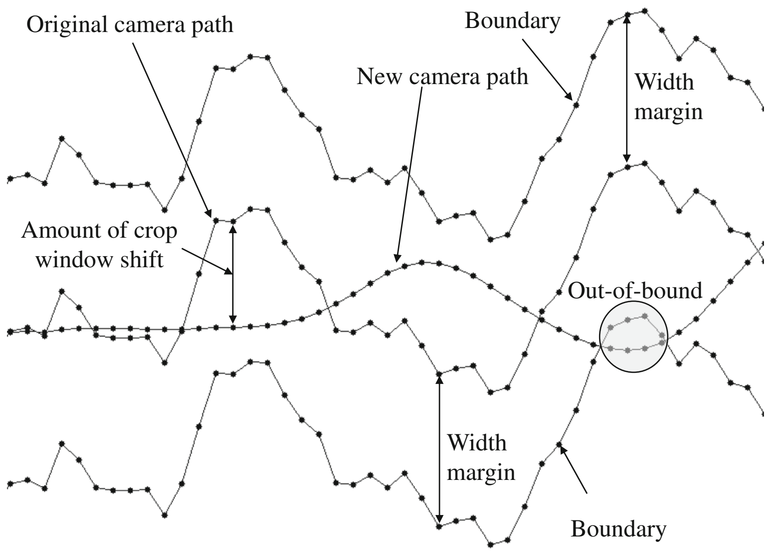

Fig. 3 Width and height margin

However, since camera positions at the subsequent frames are unknown in the real-time video stabilization, it is sometimes difficult to correctly determine the camera position at the current frame. Figure 4 illustrates an example of the difficulty in planning a camera path for real-time video stabilization. In Fig. 4a, it could not be predicted whether the camera path goes up or down at the subsequent frames. If the camera path travels downward at the subsequent frames, the new camera path can be straight, as shown in Fig. 4b. If the camera path travels upward at the subsequent frames, the direction of the new camera path should change, as seen in Fig. 4c. This sudden movement will degrade the visual quality of the stabilized video. On the other hand, since the entire original camera path is given in post-process video stabilization, the new camera path can be planned as shown in Fig. 4d. As shown in this example, the real-time video stabilization is difficult without the camera positions at the subsequent frames. The performance of the camera path planning in real-time video stabilization will inevitably be limited. Therefore, camera path planning for real-time video stabilization is a challenging problem.

3 Method—camera path planning

3.1 Observation

Simple tests were performed to observe how camera paths affect visual perception. People who did not have related knowledge were selected as participants. Ten participants evaluated the visual perception of synthetic videos with different types of camera paths. Each video has one type of camera path among various types including static view, horizontally moving views with zero acceleration, and randomly moving views. If the synthetic video includes local motion, it will disturb the evaluation of the visual perception caused only by the camera path. In order to exclude the local motion, the test videos are synthesized by shifting a single image along the camera path. The participants were asked to sort the videos in a high-quality order according to their visual perception. Table 1 shows the results. Interestingly, the scores made by the participants were in the same order. Obviously, the static camera path obtained a high score. Hence, the constant path is the first property to consider when planning a camera path. The next property is the linearity of a camera path. Consistency in camera movements is an important feature.

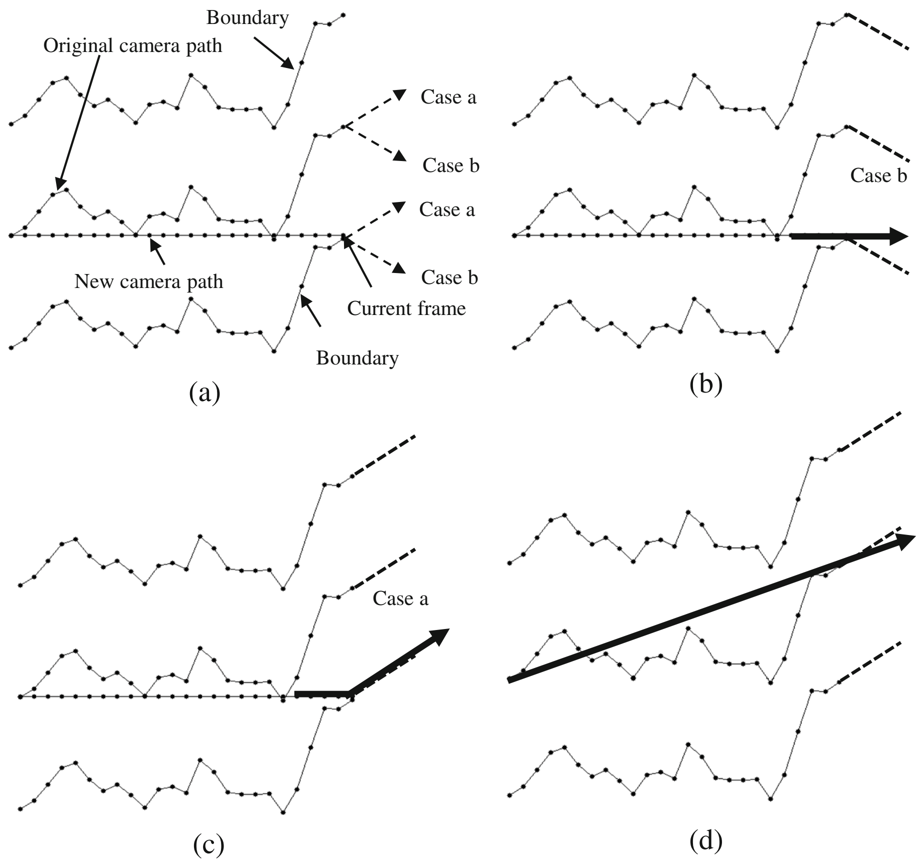

Fig. 4 Difficulty in planning a camera path for real-time video stabilization **a** current status, **b** the new camera path in "case b," **c** the new camera path in "case a," and **d** the ideal camera path in "case a"

Table 1 Simple test showing how camera paths affect visual perception

Order	Motion type	Amount of motion (pixel)
1	Static	0
2	Horizontal	1
3	Horizontal	2
4	Horizontal	3
5	Horizontal	4
6	Random	± 1
7	Random	± 2

Results are sorted in high-quality order

Even a small random motion gives a poor visual impression. A camera path with a 4 pixel horizontal movement is better than a camera path with a ± 1 random camera movement. The amount of slope of the linear camera path also affects the visual quality. A linear camera path with a gentle slope gives better visual quality than that with a steep slope.

More complex experiments are needed to more profoundly understand the relationship between the camera path and visual perception in terms of stability. However, it may require deep psychovisual understanding and extensive experiments with a greater number of participants. This paper considers only the two properties obtained from the above simple experiments in planning the new camera path.

3.2 Assumption

It is challenging to plan the steady camera path within the upper and lower boundaries. As discussed in Section 2, the difficulty is that the real-time video stabilization determines the new camera position in the current frame without the original camera positions in the subsequent frames. Even if the original camera positions in several subsequent frames are available before determining the new camera position in the current frame, the quality of the new camera path will be significantly increased. Fortunately, recent camera systems have extraordinarily large memory to temporally store multiple frames. Some frames can be buffered to improve the quality of the new camera path. Figure 5 shows the assumptions for solving a problem in the proposed algorithm. In the figure, the camera captures up to the $(n + K)$th frame and stores them to the buffer. The original camera positions for the buffered frames are predicted. The proposed algorithm then determines the new camera position in the (n)th frame (or the current frame). Here, K represents the number of additional buffered frames.

The quality of the new camera path will depend on the amount of K. This paper proposes a new method to estimate a steady camera path for the given K with the lower and upper boundary constraints.

3.3 Framework of the proposed algorithm

The first property to consider in planning camera path is providing constant camera path. The constant camera path represents a line with a zero slope. The next property is the linear camera path where the camera has a constant velocity. While the slope of the linear camera path is not zero, the linear path is also a line. Although the physical meanings of the two properties differ in terms of camera movement, the shape of the constant and linear camera paths is basically a line. Hence, the proposed algorithm considers a line as a basic feature to pursue in the camera path.

The proposed algorithm first finds all possible candidate lines that pass through the last camera position on the new camera path, as shown in Fig. 6. Since the new camera path should be continuous, the candidate lines should start from the camera position at the $(n - 1)$th frame, where it is the last camera position on the new camera path. They then end between low and upper boundaries at the $(n + K)$th frame. All possible candidate lines are generated in a way that the interval between adjacent candidate lines is 1 at the $(n + K)$th frame. In the figure, the camera position at the current frame (or the nth frame) is not yet determined. The buffered frames range from the $(n+1)$th frame to the $(n+K)$th frame. Camera motions for the buffered frames can be predicted in advance, so that the original camera positions and boundaries at the buffer frames can be determined. Hence, the end points of the candidate lines are simply located between the two boundaries at the $(n+K)$th frame. Then, the proposed algorithm examines all the possible candidate lines within the search range using the proposed cost function that will be described in the next subsection. The candidate line having the minimum cost is chosen as the best line. Finally, the camera position at the current frame is calculated from the best line. Note that the camera positions at the buffered frame do not belong to the new camera path.

If a candidate line passes through an out-of-bound area, it should not be chosen as the best line. Figure 7 depicts examples of the candidate lines that should not be chosen. In Fig. 7a, candidate lines in region R pass through an out-of-bound area. Hence, no candidate line in region R should be chosen as the best line. Sometimes, all the candidate lines within the search range pass through the out-of-bound area as shown in Fig. 7b. In this case, when no best line can be found, the value of K is decreased by 1 until the best line is found.

3.4 Cost function

The proposed algorithm chooses a candidate line having the minimum cost as the best line. The cost function of the ith candidate line at the nth frame is defined as follows.

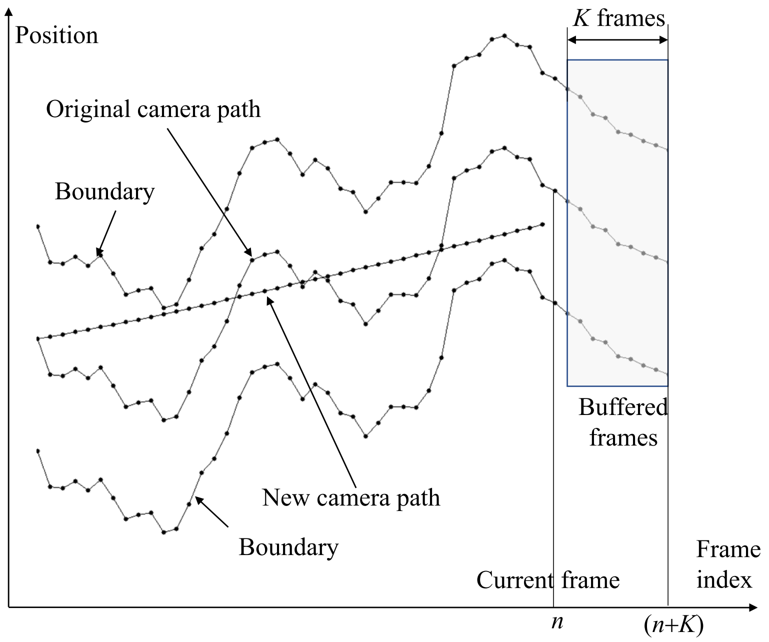

Fig. 5 The goal of the proposed algorithm is to provide a novel camera path under that the original camera positions for the next available *K* frames

$$C(n,i) = w_1 C_S(n,i) + w_2 C_L(n,i) + w_3 C_M(n,i) \tag{4}$$

Here, w_1, w_2, and w_3 are weighting factors. $C_S(n,i)$ is defined as follows.

$$C_S(n,i) = |S(n,i)| \tag{5}$$

$S(n,i)$ is a slope of the ith candidate line at the nth frame. The zero slope of a candidate line represents no camera movement. Therefore, $C_S(n,i)$ can be interpreted as a term to pursue the static camera path, which is the first property. If the slope

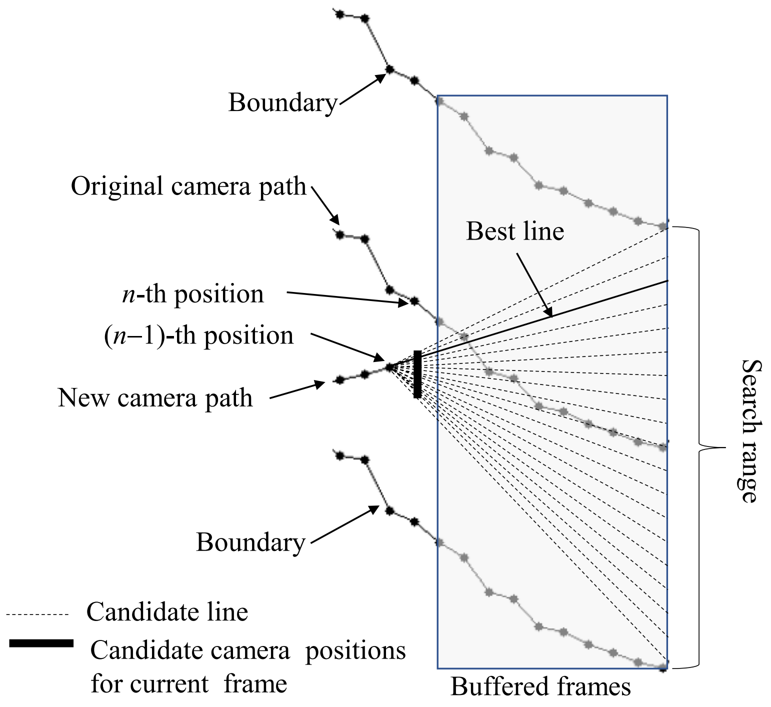

Fig. 6 Possible candidate lines

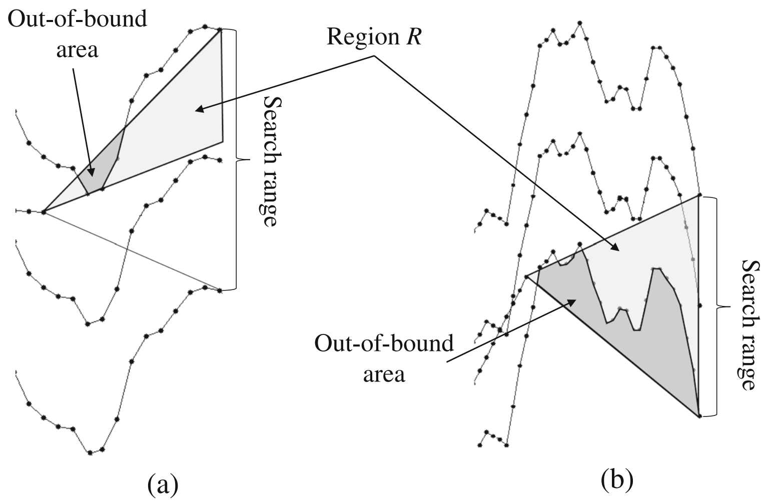

Fig. 7 a Some and **b** all candidate lines pass through an out-of-bound area. None of these candidate lines should be chosen as the best line

of a candidate line is steep, this term becomes large.

$C_L(n,i)$ is defined as follows.

$$C_L(n,i) = |S(n,i) - S(n-1, i^{\min}_{n-1})| \quad (6)$$

Here, $i^{\min}_{n-1}$ is the index of the best line at the $(n-1)$th frame. Therefore, $S(n-1, i^{\min}_{n-1})$ denotes the slope of the best line at the $(n-1)$th frame. $C_L(n,i)$ will be zero if the slope of a candidate line at the current frame is the same as the slope of the best line at the previous frame. It can be interpreted that this term seeks to keep the linearity of the camera path, which is the second property.

If the camera path plan method utilizes the upper and lower boundaries as simple constraints, although the constraints will prevent the new camera path exceeding the available image margin, the new camera position can sometimes be very close to the lower boundary such as the new camera position for the current frame in Fig. 4a. In this case, if the original camera path travels upwards (case a), since there is no image margin in the down direction, the new camera path should suddenly change the direction to upward. This sudden movement on the new camera path will degrade the visual quality of stabilized videos. To avoid this situation, this paper considers a new term of $C_M(n,i)$ to retain the available image margin in the upper and lower directions.

$$C_M(n,i) = \sum_{k=n}^{n+K} (F_{\mathrm{LO}}(k,i) + F_{\mathrm{UP}}(k,i))$$

$$F_{\mathrm{UP}}(n,i) = \frac{1}{\max(0.01, P_O(n) + M_{\mathrm{UP}} - P^i_N(n))^2} \quad (7)$$

$$F_{\mathrm{LO}}(n,i) = \frac{1}{\max(0.01, P^i_N(n) - P_O(n) + M_{\mathrm{LO}})^2}$$

where $P^i_N(n)$ is a camera position at the nth frame on the ith candidate line. M_{LO} and M_{UP} are the sizes of the lower and upper margins, respectively. These margins are usually set at the same value. $P_O(n)+M_{\mathrm{UP}}-P^i_N(n)$ (or $P^i_N(n)+M_{\mathrm{LO}}-P_O(n)$) is the Euclidean distance between $P^i_N(n)$ and the upper boundary (or lower boundary), as given in Fig. 8. Here, 0.01 is considered to prevent division by zero. If $P^i_N(n)$ is close to the upper boundary (or lower boundary), the value of $F_{\mathrm{UP}}(n,i)$ (or $F_{\mathrm{LO}}(n,i)$) will become significantly large. If $P^i_N(n)$ is remote from the upper boundary (or lower boundary), the value of $F_{\mathrm{UP}}(n,i)$ (or $F_{\mathrm{LO}}(n,i)$) will become significantly small. When $P^i_N(n)$ is located at a position equally far from both of the two boundaries, $(F_{\mathrm{UP}}(n,i)+F_{\mathrm{LO}}(n,i))$ will obtain the minimum value. Accordingly, $F_{\mathrm{UP}}(n,i)$ and $F_{\mathrm{LO}}(n,i)$ can be interpreted as forces to push a camera position from boundaries to retain the image margin. $C_M(n,i)$ accumulates all of $F_{\mathrm{UP}}(n,i)$ and $F_{\mathrm{LO}}(n,i)$ at camera positions on the ith candidate line. The camera position cannot be located outside the boundaries. If the K value decreases, $C_M(n,i)$ becomes an important term to achieve a new camera path with high quality in real-time video stabilization. The details are given in Section 4 with the experimental results.

Let us briefly consider the computational process of the proposed method. The first step of the proposed algorithm is to find candidate lines. This can be simply achieved by obtaining the lines connecting from the $(n-1)$th camera position to the points within the search range, as shown in Fig. 6. The next step involves excluding candidate lines that include at an out-of-bound area, as shown in Fig. 7. If any point on the ith candidate line (or $P^i_N(n)$) is not located from $P_O(n) - M_{\mathrm{LO}}$ to $P_O(n) + M_{\mathrm{UP}}$, the corresponding candidate line should be excluded. In the final step, the method evaluates each candidate line

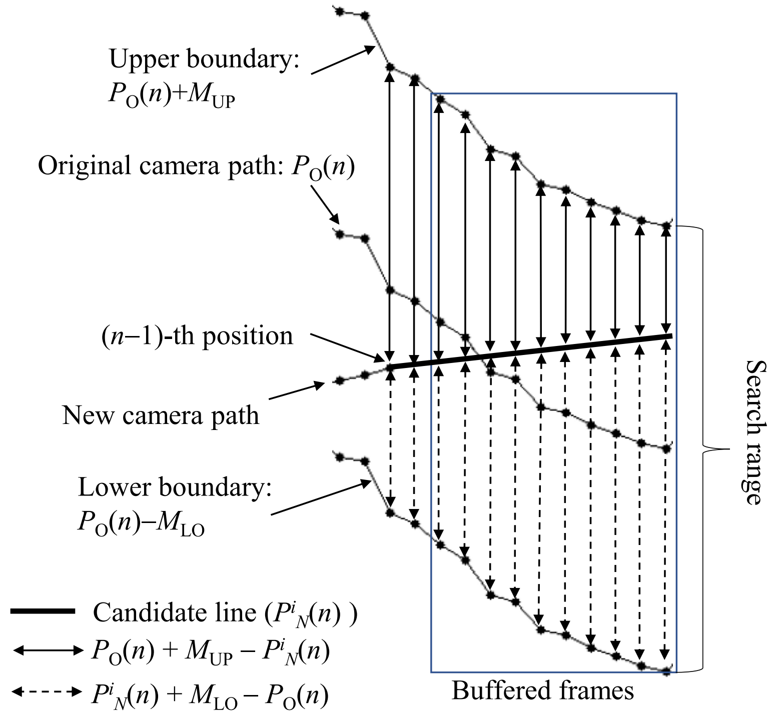
Upper boundary: $P_O(n)+M_{UP}$
Original camera path: $P_O(n)$
$(n-1)$-th position
New camera path
Lower boundary: $P_O(n)-M_{LO}$
Candidate line ($P^i_N(n)$)
$P_O(n) + M_{UP} - P^i_N(n)$
$P^i_N(n) + M_{LO} - P_O(n)$
Buffered frames
Search range

Fig. 8 Distance between the candidate line and boundaries

according to Eq. (4) and finds the candidate line having the minimum cost. As shown above, the proposed algorithm requires several arithmetic operations and comparisons to plan the new camera position of the new camera path at the current frame. Compared to the computational burden of video stabilization, its computational process is extremely low, as will be demonstrated in the Section 4.

4 Experimental results and discussion

We show simulation results of four representative videos among the extensive test videos to evaluate the proposed algorithms. Two videos were made by a man running, so that the videos include very dynamic camera motion. The other two videos were shot while walking, and the camera motion is normal. In the videos, the region of interest is set on the human face. For each video, the original camera path is extracted by tracking the human face. In this paper, not all simulation results can be illustrated due to the limited space. Only some of the simulation results are given in this section. The original and new camera paths for all the frames in four representative test videos are downloadable on the Web [35]. Output videos are also downloadable on the Web. High-quality Figs. 9, 10, 11, 12, 13, and 14 were uploaded on the Web.

Figure 9 depicts some part of the simulation results, in order to show the effect of $C_S(n,i)$ in Eq. (4). Here, (w_1, w_2, w_3) for Fig. 9 a and b are set at (0, 0.01, 10) and (0.01, 0.01, 10), respectively. Without $C_S(n,i)$, the first property mentioned in Section 3.1 is not satisfied. Therefore, the new camera path is not constant as shown in Fig. 9 a. $C_S(n,i)$ renders the new camera path constant.

Figure 10 shows the effect of $C_L(n,i)$ in Eq. (4). Weights for Fig. 10 a and b are set at (0.01, 0, 10) and (0.01, 0.01, 10), respectively. The method without $C_L(n,i)$ considers only a constant camera path ($C_S(n,i)$) and a force pushing from boundaries ($C_M(n,i)$). When the current position is located remote from the boundaries, $C_S(n,i)$ is dominant and the method renders the new camera path constant. If the original camera path globally descends as in Fig. 10, the camera position along the constant camera path eventually approaches close to the boundary. Then, the force pushing from the boundary becomes large so that the method will change the direction of the new camera path downward. The camera position again becomes remote from the boundary. Then, $C_S(n,i)$ becomes dominant and the method renders the new camera path constant again. Accordingly, the slope of the new camera path frequently changes as shown in Fig. 10 a. $C_L(n,i)$ improves the performance as shown Fig. 10 b.

If the value of K is large, the proposed method without a term of $C_M(n,i)$ can easily plan a steady camera path. Figure 11 a shows an example where K is set at 60. Since the method can consider 60 frames in the future, it can provide a steady camera path by coping with sudden camera movements in advance. However, when the

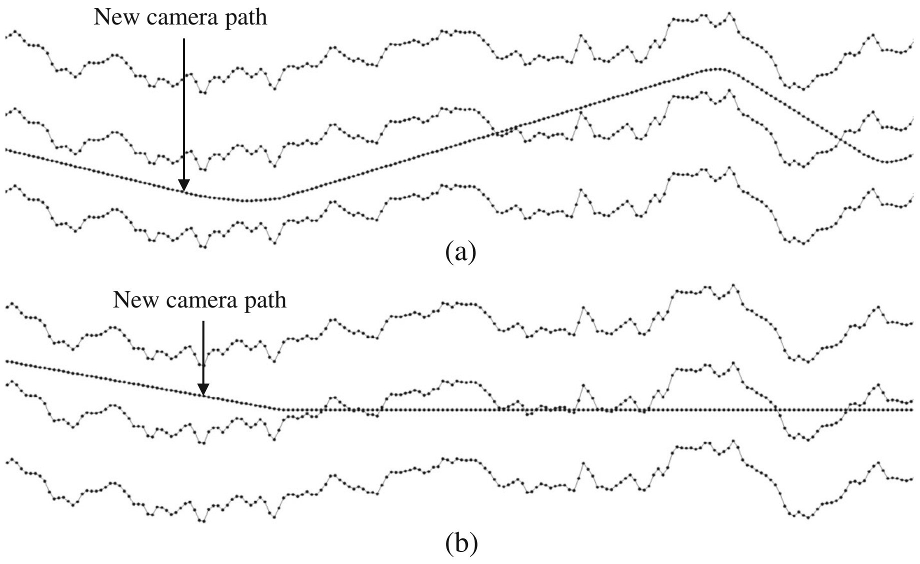

Fig. 9 a Without $C_S(n,i)$ and **b** with $C_S(n,i)$ in Eq. (4). K is set at 10

value of K decreases, it is difficult to cope with the sudden camera movements, as shown in Fig. 11 b. Here, K is set at 2. The proposed method without $C_M(n,i)$ utilizes the upper and lower boundaries as only boundary constraints. Hence, even if the amount of available image margin is very small, the method attempts to make the new camera path constant. Only if there is no image margin, the method changes the new camera path upward or downward. Accordingly, even for small fluctuations of the original camera path, the direction of the new camera path frequently changes, which is undesirable. On the other hand, $C_M(n,i)$ pushes the new camera path from the boundaries to maintain the image margin. In the details, if the weights for $C_L(n,i)$ and $C_S(n,i)$ are set at the same value, $C_L(n,i)$ and $C_S(n,i)$ show almost the same dominance. Then, $C_M(n,i)$ becomes a dominant term.

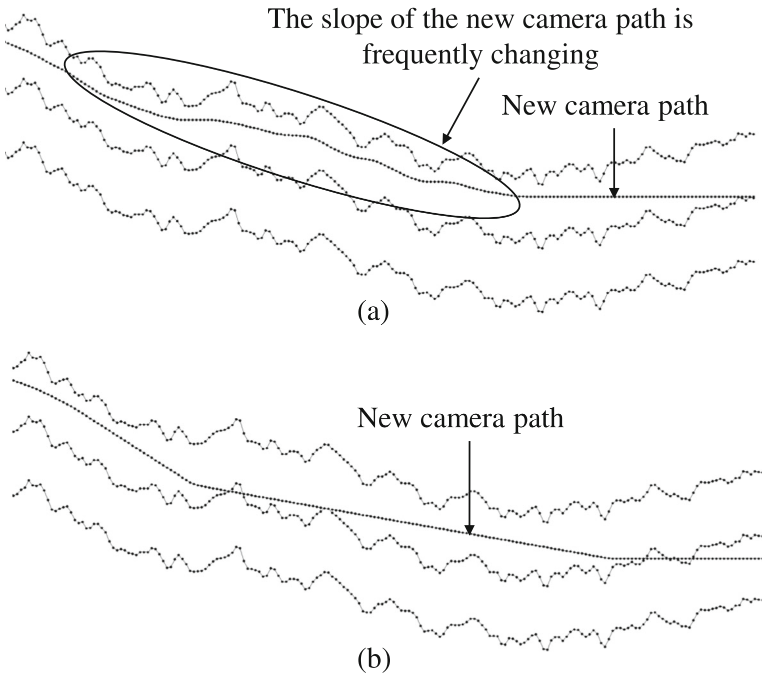

Fig. 10 a Without $C_L(n,i)$ and **b** with $C_L(n,i)$ in Eq. (4). K is set at 10

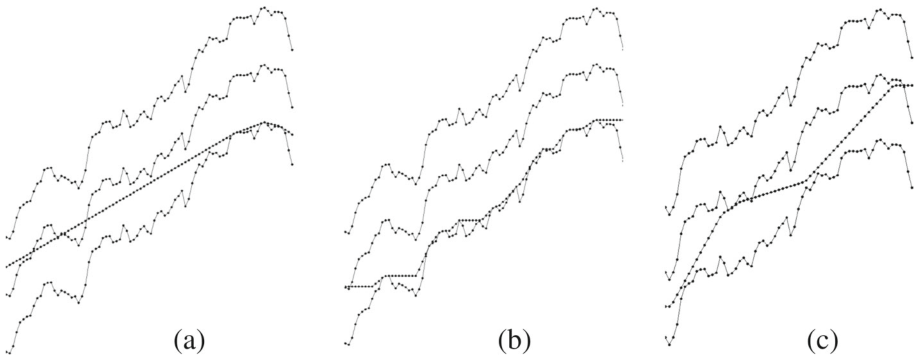

Fig. 11 **a** $K = 60$, **b** $K = 2$ without $C_M(n, i)$, and **c** $K = 2$ with $C_M(n, i)$. Weights are set at **a** (0.01, 0.01, 0), **b** (0.01, 0.01, 0), and **c** (0.01, 0.01, 1), respectively

Therefore, the new camera path travels to the center or the middle of the upper and lower boundaries where the pushing forces from the two boundaries are the same. After the path reaches the center, $C_M(n, i)$ becomes a weak term and the method can set the new direction of the new camera path as shown in Fig. 11c.

Figure 12 depicts the simulation results for a video, including usual camera motion, in order to show the performance of the proposed algorithm. The image margins for Fig. 12 a, b, c, and d are set at 10%. The values of K are set at 2, 5, 10, and 60, respectively. The proposed algorithm significantly reduces the high-frequency jitter of camera motion, and the new camera path does not cross the upper and lower boundaries. Although the number of buffer frames in the proposed method of $P_{2,10\%}$ is only 2, it provides outstanding performance. For reference, the new camera paths based on the following simple low-pass filters, $P_N^{\mathrm{LPF}}(t)$, are illustrated in Fig. 12 e and f.

$$P_N^{\mathrm{LPF}}(t) = \sum_{k=t-K}^{k=t+K} w(k) P_O(k) \tag{8}$$

The values of K for LPF$_{10}$ and LPF$_{60}$ in Fig. 12 e and f are set at 10 and 60, respectively. While LPF$_{10}$ significantly reduces high-frequency jitter of the camera motion, the

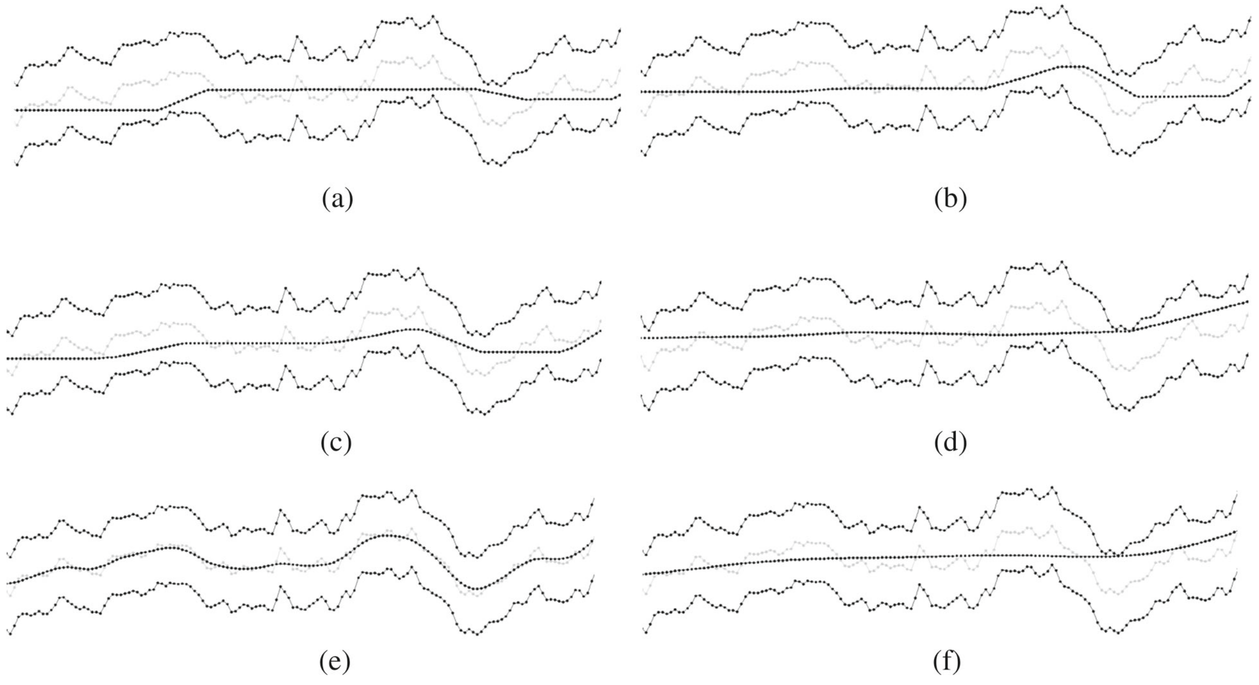

Fig. 12 Simulation results for a video including usual camera motion. **a** $P_{2,10\%}$ ($K = 2$, proposed), **b** $P_{5,10\%}$ ($K = 5$, proposed), **c** $P_{10,10\%}$ ($K = 10$, proposed), **d** $P_{60,10\%}$($K = 60$, proposed), **e** LPF$_{10}$ ($K = 10$, low-pass filter), and **f** LPF$_{60}$ ($K = 60$, low-pass filter). Weights for the proposed algorithm are set at (0.01, 0.01, 10). Image margin is set at 10%

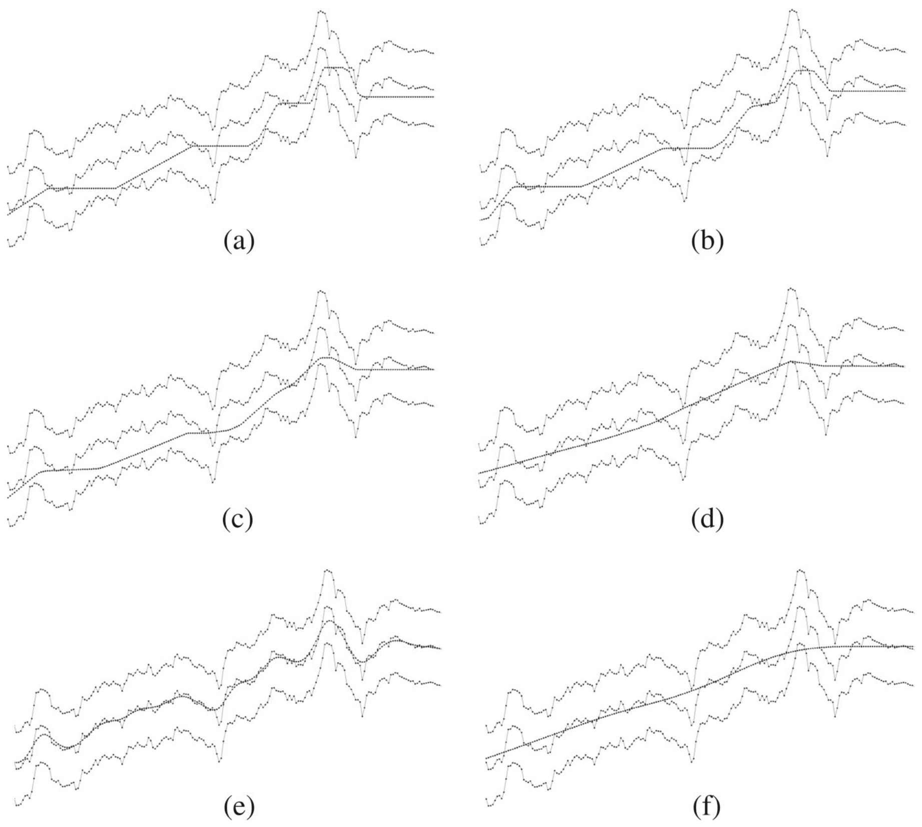

Fig. 13 Simulation results of a video including dynamic camera motion. **a** $P_{2,20\%}$ ($K = 2$, proposed), **b** $P_{5,20\%}$ ($K = 5$, proposed), **c** $P_{10,20\%}$ ($K = 10$, proposed), **d** $P_{60,20\%}$ ($K = 60$, proposed), **e** LPF$_{10}$ ($K = 10$, low-pass filter), and **f** LPF$_{60}$ ($K = 60$, low-pass filter). Weights for the proposed algorithm are set at (0.01, 0.01, 10). Image margin is set at 20%

new camera path is still unstable. Although LPF$_{60}$ provides good performance, the value of K is very large. For full-HD video, the memory size is 178 M bytes ($= 1920 \times 1080 \times 1.5 \times 60$) for a YCbCr format. Moreover, LPF$_{10}$ and LPF$_{60}$ do not guarantee that the cropping window will always be located within the original frame (see Fig. 3.) The performance of the proposed method of $P_{2,10\%}$ provides better performance than LPF$_{10}$. Note that while the number of buffer frames in LPF$_{10}$ is 10, that in $P_{2,10\%}$ is only 2.

Figure 13 depicts the simulation results for a video, including dynamic camera motion depending on the value of K. The image margins for Fig. 13 a, b, c, and d are set at 20%. The values of K are set at 2, 5, 10, and 60, respectively. Figure 13 e and f show simulation results of LPF$_{10}$ and LPF$_{60}$. Although the new camera path predicted from the proposed algorithm of $P_{2,20\%}$ is much better than the original camera path, it does not provide outstanding performance, compared to that shown in Fig. 12 a. Two frame buffers are insufficient to cope with sudden camera movement in the video, including dynamic camera motion. As the value of K increases, the proposed algorithm provides more stable camera path. While LPF$_{60}$ provides good performance, since this method does not consider boundary constraints, the new camera path crosses the boundary.

The threshold values of w_1 and w_2 can be set as the different values. For example, the value of w_1 can be set larger than that of w_2 in order to give priority to $C_S(n, i)$. However, according to our extensive experiments, it is recommended to set the two thresholds to the same value. The performance of the proposed algorithm is not sensitive to the threshold value of w_3.

Figure 14 shows the performance of the proposed algorithm according to the image margin. The image margins for Fig. 14 a, b, c, and d are set at 10%, 10%, 20%, and 20%, respectively. The values of K are set at 10, 20, 10, and 20, respectively. As the size of the image margin decreases, the algorithm performance reduces. The performance gap between $P_{10,10\%}$ and $P_{20,10\%}$ is more than the performance gap between $P_{10,20\%}$ and $P_{20,20\%}$. When the image margin decreases, the value of K needs to be large.

Comparison of performance between the proposed algorithm and Grundmann's work [36] is as illustrated in Fig. 15. The image margins for Fig. 15a, b, and c were set at 20%. The values of K were set at 2, 10, and 60,

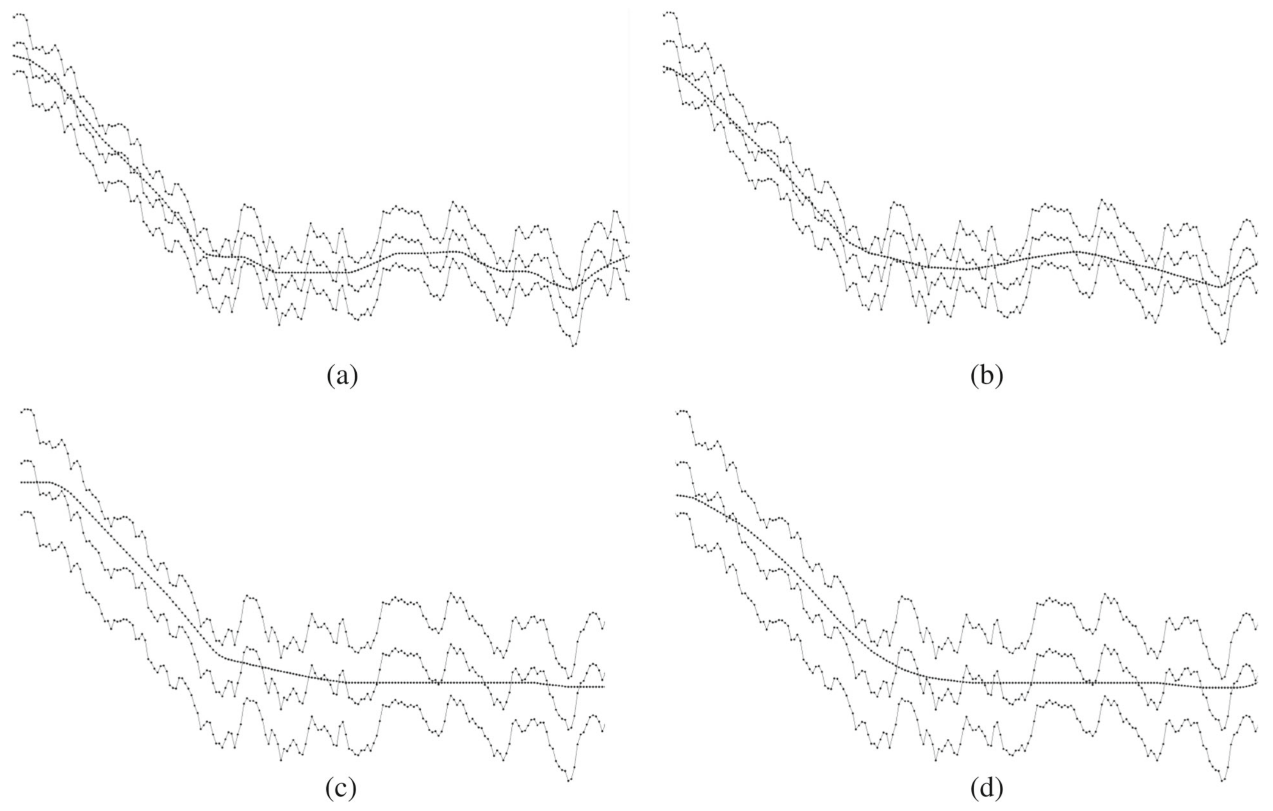

Fig. 14 Simulation results for a video including dynamic camera motion. **a** $P_{10,10\%}$ ($K = 10$, image margin 10%), **b** $P_{20,\%10}$ ($K = 20$, image margin 10%), **c** $P_{10,20\%}$ ($K = 10$, image margin 20%), and **d** $P_{20,20\%}$ ($K = 20$, image margin 20%). Weights for the proposed algorithm are set at (0.01, 0.01, 10)

respectively. Figure 15 d and e represent LPF_{10} and LPF_{60}, respectively. Figure 15 f, g, and h illustrate Grundmann's results. Parameters w_1, w_2, and w_3 for Fig. 15f, g, and h were set to (0, 1, 0), (0, 0, 1), and (10, 1, 100), respectively. $P_{2,\%20}$ and $P_{10,\%20}$ did not outperform Grundmann's work [36]. However, while his work considers all the fames in order to get the optimal paths, the proposed $P_{2,\%20}$ and $P_{10,\%20}$ consider only 2 and 10 buffered frames. As the proposed algorithm uses more buffered frames, its performance becomes comparable to Grundmann's work as shown in Fig. 15c. Since LPF_{60} strongly smoothens the original camera path without considering boundary constraints, it provides performance similar to that of the proposed method and Grundmann's work.

The original camera path was extracted by tracking the human face, and the camera path was generated by using the proposed method, LPF_{10}, and LPF_{60}. Then, simple video stabilization with 2D translational motion model in Fig. 1 was applied to generate the output videos that were uploaded on the web [35]. To evaluate the proposed algorithm, the stability of the human face position along time needs to be examined. Since LPF_{60} sometimes exceeds the available image margin, out-of-bound regions (black regions) are frequently observed in the stabilized videos. Moreover, it requires huge frame delay. Thus, LPF_{60} is not adequate for real-time video stabilization. However, since LPF_{60} provides a very stable camera path, a video rendered by LPF_{60} is used for comparison purpose. As shown in the comparison videos, the proposed algorithm provides performance close to LPF_{60}. A video from LPF_{10} reduces high-frequency jitter of the original video compared to the original video. However, the human face is slightly unstable.

Table 2 shows *mean absolute slopes* of camera paths used to evaluate the performance of algorithms. The smaller the value of the *mean absolute slope*, the more stable the camera path. The *mean absolute slopes* of LPF_{10}, LPF_{60}, $P_{10,10\%}$, $P_{10,20\%}$, $P_{60,10\%}$, and $P_{60,20\%}$ are significantly lower than those of the original camera path. It means that they provide more stable camera path than the original camera path. $P_{10,10\%}$ always provides better performance than LPF_{10}. Since 10 frame buffers are not enough to provide very stable camera path, $P_{10,10\%}$ does not outperform LPF_{60}. If the size of the buffer becomes the same as 60 frames and the margin is enough, $P_{60,20\%}$ outperforms LPF_{60}. It is noted here that LPF_{10} and LPF_{60} do not consider the upper and lower boundary constraints.

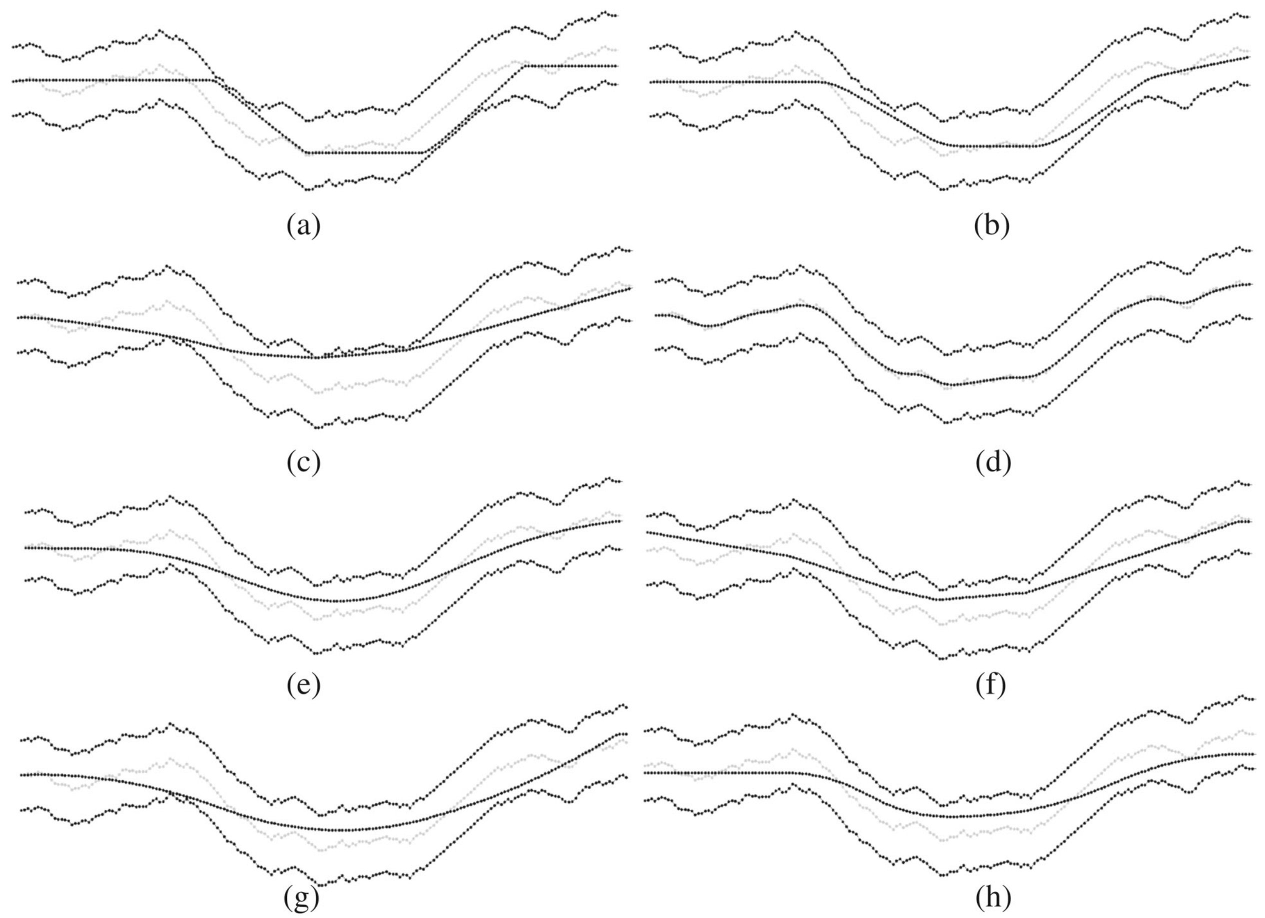

Fig. 15 Performance comparison of the proposed algorithm with [36]. **a** $P_{2,20\%}$ ($K = 2$, image margin 20%), **b** $P_{10,\%20}$ ($K = 10$, image margin 20%), **c** $P_{60,20\%}$ ($K = 60$, image margin 20%), **d** LPF$_{10}$ ($K = 10$, low-pass filter), **e** LPF$_{60}$ ($K = 60$, low-pass filter), **f** [36], $w_2 = 1$, $w_1 = w_3 = 0$, **g** [36], $w_3 = 1$, $w_1 = w_2 = 0$, and **h** [36], $w_1 = 10$, $w_2 = 1$, $w_3 = 100$. Weights for the proposed algorithm are set at (0.01, 0.01, 10)

The proposed algorithm is designed for real-time video stabilization. Hence, its computational burden needs to be dealt with. Table 3 shows the time for processing a single frame in the proposed algorithm under Intel Core i7-7700K CPU (4.2 GHz) according to the value of K. The proposed algorithm was implemented with C language without optimization. As mentioned in Section 1, video stabilization generally requires huge computational complexity; however, the real-time video stabilization is quite challenging. In that sense, tens of microseconds for

Table 2 Comparisons of mean absolute slopes

Method	Normal motion				Dynamic motion			
	A		B		C		D	
	x	y	x	y	x	y	x	y
Original	9.04	6.02	9.85	5.37	18.09	17.28	18.70	17.47
LPF$_{10}$	3.62	1.72	5.34	1.57	7.06	2.95	8.22	2.76
LPF$_{60}$	1.83	0.35	3.45	0.58	3.25	0.80	4.76	0.84
$P_{10,10\%}$	2.18	0.86	4.51	0.85	4.57	2.72	7.01	2.54
$P_{60,10\%}$	1.84	0.40	4.05	0.50	4.10	2.40	6.63	2.25
$P_{10,20\%}$	1.93	0.31	4.04	0.41	3.67	1.33	6.11	1.28
$P_{60,20\%}$	1.66	0.32	3.08	0.42	3.06	0.77	5.06	0.64

Table 3 Processing time of the proposed algorithm under Intel Core i7-7700K CPU (4.2 GHz)

Parameter (K value)	Time (μs)
2	7.5
5	10.6
10	14.8
60	62.2

processing a single frame is extremely small, hence considered negligible, compared to the total processing time of the video stabilization. As the value of K increases, the computational burden also increases. However, the increase of complexity is not proportional to the value of K. Moreover, the processing time for $K = 60$ is only 62.2μs.

5 Conclusion

This paper presents a novel camera path planning algorithm for real-time video stabilization by cross-optimizing two terms related to steady camera path and the amount of image margin. Hence, the proposed algorithm attempts to provide a steady camera path while maintaining a sufficient image margin to compensate for dynamic or sudden camera motions.

While all the frames can be used for to plan the new camera path in post-processing video stabilization, only a limited number of frames can be used in real-time video stabilization. Moreover, the real-time camera path planning algorithm should predict the new camera path on-the-fly. Once the new camera path is planned, it cannot be updated or modified to improve the quality, unlike post-processing video stabilization. Hence, camera path planning is a challenging issue in real-time video stabilization. If the quality of a planned camera path is not acceptable, although accurate camera motions are predicted, the quality of stabilized videos will be degraded. Hence, camera path planning is an essential feature in real-time video stabilization, and our work is thus meaningful.

Abbreviations

CPU: Central processing unit; HD: High definition; IIR: Infinite impulse response; LPF: Low-pass filter; SFM: Structure-from-motion

Funding

This present research has been conducted by the Research Grant of Kwangwoon University in 2018. This work was supported by an Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) [2016-0-00144], Moving Free-viewpoint 360VR Immersive Media System Design and Component Technologies.

Availability of data and materials

The data and materials are downloadable on the Web (http://sites.google.com/site/camerapathplanning).

Authors' contributions

Lee works all things about this paper. The author read and approved the final manuscript.

Authors' information

Yun Gu Lee received his B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea, in 2000, 2002, and 2006, respectively. From 2006 to 2013, he was in the Media Processing Laboratory, DMC Research and Development Center, Samsung Electronics, Republic of Korea, where he was a principal engineer. In 2013, he joined the school of software, Kwangwoon University, Seoul, Republic of Korea, where he is currently a professor. His current research interests include the general areas of image and video coding, image and video processing, image stabilization, computer vision, and three-dimensional display systems.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The author declares he has no competing interests.

Received: 23 August 2018 Accepted: 25 April 2019
Published online: 27 May 2019

References

1. F. La Rosa, M. Celvisia Virzì, F. Bonaccorso, M. Branciforte, Optical image stabilization (ois). STMicroelectronics white paper 2015 (2015). https://www.st.com/resource/en/white_paper/ois_white_paper.pdf
2. C. Morimoto, R. Chellappa, in *Proceedings of the DARPA Image Understanding Workshop*. Evaluation of image stabilization algorithms (IEEE, New Jersey, 1998), pp. 295–302
3. W. H. Cho, K. S. Hong, Affine motion based CMOS distortion analysis and CMOS digital image stabilization. IEEE Trans. Consum. Electron. **53**, 833–841 (2007)
4. C. Wang, J. H. Kim, K. Y. Byung, J. Ni, S.-J. Ko, Robust digital image stabilization using the Kalman filter. IEEE Trans. Consum. Electron. **55**, 6–14 (2009)
5. W. H. Cho, D. W. Kim, K. S. Hong, Mos digital image stabilization. IEEE Trans. Consum. Electron. **42**, 979–986 (2007)
6. H.-C. Chang, S.-H. Lai, K.-R. Lu, in *IEEE International Conference on Multimedia and Expo*. A robust and efficient video stabilization algorithm (IEEE, New Jersey, 2004), pp. 16–27
7. K. Ratakonda, in *IEEE International Symposium on Circuits and Systems*. Real-time digital video stabilization for multi-media applications (IEEE, New Jersey, 1998), pp. 69–72
8. Z. Zhu, G. Xu, Y. Yang, J. S. Jin, in *IEEE International Conference on Intelligent Vehicles*. Camera stabilization based on 2.5D motion estimation and inertial motion filtering (IEEE, New Jersey, 1998), pp. 329–334
9. Y. Matsushita, E. Ofek, X. Tang, H.-Y. Shum, in *IEEE CVPR 2005*. Full-frame video stabilization (IEEE, New Jersey, 2005), pp. 50–57
10. Y. Matsushita, E. Ofek, W. Ge, X. Tang, H.-Y. Shum, Full-frame video stabilization with motion inpainting. IEEE Trans. Pattern. Anal. Mach. Intell. **28**, 1150–1163 (2006)
11. S. Kumar, H. Azartash, M. Biswas, T. Nguyen, Real-time affine global motion estimation using phase correlation and its application for digital image stabilization. IEEE Trans. Image Process. **20**, 3406–3419 (2011)
12. Y. G. Lee, G. Kai, Fast rolling shutter compensation based on piecewise quadratic approximation of a camera trajectory. Opt. Eng. **53**, 093101 (2014)
13. J. S. Jin, Z. Zhu, G. Xu, A stable vision system for moving vehicles. IEEE Trans. Intell. Transp. Syst. **1**, 32–39 (2000)
14. K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, M. Ouhyoung, in *IEEE International Conference on Computer Vision*. Video stabilization using robust feature trajectories (IEEE, New Jersey, 2009), pp. 1397–1404
15. C. Liu, J. Yuen, A. Torralba, J. Sivic, W. T. Freeman, in *In Proceedings of European Conference on Computer Vision 2008*. Sift flow: dense correspondence across difference scenes (Springer, New York, 2008), pp. 28–42

16. P. Sand, S. Teller, Particle video: long-range motion estimation using point trajectories. Int. J. Comput. Vis. **80**, 72–91 (2008)
17. C. Buehler, M. Bosse, L. McMillan, in *IEEE Conference on Computer Vision and Pattern Recognition*. Non-metric image based rendering for video stabilization (IEEE, New Jersey, 2001), pp. 609–614
18. R. I. Hartley, A. Zisserman, *Multiple View Geometric in Computer Vision*. (Cambridge University Press, Cambridge, 2004)
19. G. Zhang, W. Hua, X. Qin, Y. Shao, H. Bao, Video stabilization based on a 3D perspective camera model. Vis. Comput. **25**, 997–1008 (2009)
20. Y.-S. Wang, F. Liu, P.-S. Hsu, T.-Y. Lee, Spatially and temporally optimized video stabilization. IEEE Trans. Vis. Comput. Graph. **19**, 1354–1361 (2013)
21. F. Liu, M. Gleicher, J. Wang, H. Jin, A. Agarwala, Subspace video stabilization. ACM Trans. Graph. **30**, 4 (2011)
22. S. Liu, L. Yuan, P. Tan, J. Sun, in *IEEE CVPR*. Steadyflow: spatially smooth optical flow for video stabilization (IEEE, New Jersey, 2014), pp. 4209–4216
23. M. Grundmann, V. Kwatra, D. Castro, I. Essa, in *International Conference on Computational Photography*. Calibration-free rolling shutter removal (IEEE, New Jersey, 2012)
24. J. Dong, H. Liu, Video stabilization for strict real-time applications. IEEE Trans. Circ. Syst. Video Technol. **27**, 716–724 (2017)
25. E. Ringaby, P.-E. Forssen, Efficient video rectification and stabilization of cell-phones. Inernational J. Comput. Vis. **96**, 335–352 (2012)
26. A. Karpenko, D. Jacobs, J. Baek, M. Levoy, in *Stanford Tech Report CTSR*. Digital video stabilization and rolling shutter correction using gyroscopes (Stanford, California, 2011), pp. 1–7
27. Y. G. Lee, Video stabilization based on human visual system. J. Electron. Imaging. **23**, 053009 (2014)
28. A. J. Crawford, H. Denman, F. Kelly, F. Pitie, A. C. Kokaram, in *IEEE International Conference on Image Processing*. Gradient based dominant motion estimation with integral projections for real time video stabilization (IEEE, New Jersey, 2004), pp. 3371–3374
29. F. Liu, M. Gleicher, H. Jin, A. Agarwala, Content-preserving warps for 3D video stabilization. ACM Trans. Graph. **28**, 44 (2009)
30. Z. Zhou, H. Jin, Y. Ma, in *IEEE CVPR*. Plane-based content preserving warps for video stabilization (IEEE, New Jersey, 2013), pp. 2200–2306
31. A. Litvin, J. Konrad, W. Karl, in *IS&TSPIE Symp. Electronic Imaging, Image, and Video Comm*. Probabilistic video stabilization using Kalman filtering and mosaicking (IS&T/SPIE, Springfield, 2003), pp. 663–674
32. D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. (Wiley, New Jersey, 2006)
33. B.-Y. Chen, K.-Y. Lee, W.-T. Huang, J.-S. Lin, Capturing intention-based full-frame video stabilization. Pac. Graph. **27**, 1805–1814 (2008)
34. M. L. Gleicher, F. Liu, Re-cinematography: improving the camerawork of casual video. ACM Trans. Multimed. Comput. Commun. Appl. **5**, 2 (2008)
35. Data . http://sites.google.com/site/camerapathplanning. Accessed 6 Mar 2019
36. M. Grundmann, V. Kwatra, I. Essa, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Auto-directed video stabilization with robust l1 optimal camera paths (IEEE, New Jersey, 2011)
37. H.-C. Chang, S.-H. Lai, K.-R. Lu, A robust real-time video stabilization algorithm. J. Vis. Commun. Image Represent. **17**, 659–673 (2006)
38. J. Song, X. Ma, in *IEEE 7th International Conference on Awareness Science and Technology*. A novel real-time digital video stabilization algorithm based on the improved diamond search and modified Kalman filter (IEEE, New Jersey, 2015)