Received March 12, 2018, accepted April 8, 2018, date of publication April 19, 2018, date of current version May 24, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2828653

High-Quality Real-Time Video Stabilization Using Trajectory Smoothing and Mesh-Based Warping

ZHONGQIANG WANG[1], LEI ZHANG[2], (Member, IEEE), AND HUA HUANG[2],(Member, IEEE)
[1]School of Electronics and Information Engineering, Xian Jiaotong University, Xi'an 710049, China
[2]Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Hua Huang (huahuang@bit.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61425013 and Grant 61472035.

ABSTRACT Some state-of-the-art video stabilization methods can achieve quite good visual effect, but they always cost a lot of time. On the other hand, current real-time video stabilization methods cannot generate satisfactory results. In this paper, we propose a novel trajectory-based video stabilization method which can generate high-quality results in real time. Our method runs very fast, because many techniques are proposed for acceleration. In the trajectory smoothing step, trajectories are extracted, pre-processed, and smoothed. A video splitting algorithm is proposed for pre-processing, and binomial filtering is used for smoothing. Both of them are simple and fast. In the frame warping step, we calculate a spatially varying warp that is directed by a grid mesh for each frame. Instead of solving time consuming global optimization problems, the transformation matrix of each grid is calculated using nearby trajectories in our method, leading to very high speed. We implement our method and run it on a variety of videos. Experiments show that while the stabilization effect is comparable with state-of-the-art methods, our algorithm can run in real time.

INDEX TERMS Video stabilization, real-time, trajectory smoothing, mesh-based warping.

I. INTRODUCTION

For most amateur photographers, hand-held video camera is a good choice for recording dynamic scenes because of its low price and convenience. However, videos captured by hand-held cameras are usually of low quality because of irregularly manual operation. One significant defect is the jitter introduced by hand shaking. Without the help of professional tools, such as camera dollies, it is almost impossible to keep the camera path stabilized even for a short while. In order to make these jittered videos stabilized and easy to watch, video stabilization techniques are developed for post-processing.

Early methods usually stabilize a video by estimating and smoothing a motion model. According to the used model, they can be divided into two categories: 2D methods and 3D methods. In 2D methods such as [1] and [2], the motion between two frames is modelled using one transformation matrix (usually affine or perspective). This kind of methods are very robust and fast because 2D model is very simple. However, they only work well on scenes of single plane or scenes far away from camera because 2D model cannot well handle the parallax introduced by depth change. In 3D methods such as [3] and [4], the full 3D structure of the scene is recovered and motion is modelled directly using the recovered position of the camera. Theoretically, 3D methods can generate the best results, but they are very fragile and slow. This is because structure from motion (SFM) algorithm (which is used to recover 3D structure from a video) is ill-posed and very sensitive to noise.

Some recent methods, such as [5] and [6], try to generate high quality results while keeping robust enough. They use trajectories to represent the motion in the video, so we call them trajectory based methods. Trajectory based methods can be divided into two steps: trajectory smoothing step and frame warping step. In the first step, trajectories are extracted from the video and then smoothed directly, without setting up any motion model. In the second step, a spatially-varying warp from each input frame to corresponding output frame is generated guiding by the smoothed trajectories. Trajectory based methods can handle parallax quite well, so they can generate better results than 2D methods. Meanwhile, the spatially-varying warping algorithm makes the results very perceptual plausible even in the absence of 3D motion model.

Till now, the visual effect of state-of-art trajectory based methods is so good that there is hardly any space for further improvement. Meanwhile, their execution speed becomes a big problem. Due to their high computation complexity, the time costs of these methods are very large. In their trajectory smoothing steps, either additional operation [5] or additional constraint [6] is introduced, leading to extra time cost. In their frame warping steps, optimization, which costs a lot of time, is used to find the warping destination of grids. Motivated by these problems, we aim to propose a trajectory based video stabilization method with real-time execution speed. Currently, there are only a few methods which can reach real-time. Most of these methods use hardware such as Field Programmable Gate Array (FPGA) [7], [8] and Boundary Signal Computation (BSC) chip [9]. However, all these methods are based on 2D motion model and single homography warping. Obviously, current real-time methods cannot generate high quality results.

In this paper we propose a real-time video stabilization method which generates high quality results that is comparable to state-of-art trajectory based methods. Similar to other trajectory based methods, our method also consists of two steps: trajectory smoothing step and frame warping step. In both steps, novel techniques are proposed to reduce time cost. The key acceleration strategy of our method is to decompose a big global problem into small local problems and solve them respectively. In the proposed trajectory smoothing algorithm, we decompose and solve the smoothing problem in temporal dimension. We split long videos into clips which contain enough continuous frames and then smooth each clip respectively. In the proposed frame warping algorithm, we decompose and solve the warping problem in the spatial dimension. We first divide every input frame into grids uniformly and then warp each grid ruled by its four vertices. For every vertex, we only use the trajectories near it to estimate a transformation matrix to warp it. Since the computation complexity of the decomposed small local problems is much lower than the original problem, the execution speed of our method can reach real-time.

II. RELATED WORKS

A. 2D METHODS

The basic assumption of 2D methods is that, the motion between two frames can be represented by a single 2D transformation matrix. Early methods such as [10] and [11] use a four or six parameter transformation to model the motion. Matsushita *et al.* [1] proposed a full frame stabilization method which uses eight parameter perspective transformation to further improve the flexibility of the model. This method stabilizes each frame locally by filtering the transformation matrices between this frame and several neighboring frames, to avoid cumulative error. Rather than smoothing the camera path, some methods, such as [2] and [12], speculate the users' intention by analysing the jittered video and carefully design the stabilized camera path based on this intention, to generate cinematographic and meaningful results. Kim *et al.* [13] handle stabilization and rolling shutter together. They propose a complex model in which the motion is modelled using similarity transformation. Zhou *et al.* [14] propose a special stabilization and completion method for surveillance videos using two videos. This method first estimates a rough affine model and then use pixel-based alignment to refine it. Since 2D motion models cannot handle parallax well, 2D methods do not work well on videos with complex scene.

B. 3D METHODS

3D methods recover the full 3D structure of the scene and smooth the camera motion directly. Zhang *et al.* [3] warp every frame using a single homography after smoothing the camera path. Liu *et al.* [4] propose a warping method using multiple homographies. Zhou *et al.* [15] improve [4] to achieve better performance on videos with large textureless regions by detecting large planar surfaces in the scene. Liu *et al.* [16] simplify the 3D structure reconstruction step by taking advantage of depth cameras. 3D methods usually generate good results as long as the 3D structure is well recovered. However, the structure from motion step always suffers from fragility and low speed.

C. TRAJECTORY BASED METHODS

This kind of method stabilizes videos by smoothing the extracted trajectories without setting up a motion model. Liu *et al.* [17] find that smoothed trajectories should approximately lie in a low-rank subspace, so they enforce subspace constraints on the trajectories matrix while smoothing it. Goldstein and Fattal [5] constrain trajectories using epipolar geometry. Wang *et al.* [6] divide every frame into triangles and force every triangle to be as similar as possible. Motion vectors, which is similar as trajectories, is smoothed for stabilization in some methods [18], [19]. Many state-of-art methods are trajectory based and they can generate comparable results to 3D methods with more robustness.

D. REAL-TIME VIDEO STABILIZATION

There are some real-time video stabilization methods now and some of them even use hardware for acceleration. Araneda and Figueroa [7] and Yong-Jie [8] estimate and smooth motion vectors on a FPGA. Dimov and Nikolov [9] estimate motion as Accurate Vector Modal and smooth it using BSC chip. Aguilar and Angulo [20] represent motion between frames using affine transformation matrix and estimate user's intention for stabilization. Li *et al.* [21] claim their method to be feature based, but in fact features are only used to generate affine transformations, which are smoothed using Kalman filter. Therefore this method is a real-time version of 2D method [22]. Similart to [21], Dong and Liu [23] propose a real-time method using Kalman filter, and the difference is that they use perspective transformations as motion model. After investigation, we find that the motion models used in current real-time methods are all 2D transformation matrices, i.e., they are all 2D methods. Due to the limitation of 2D motion model, they cannot generate as good results as state-of-art methods, such as trajectory based methods. However, it is very difficult to reach real time using complex models.

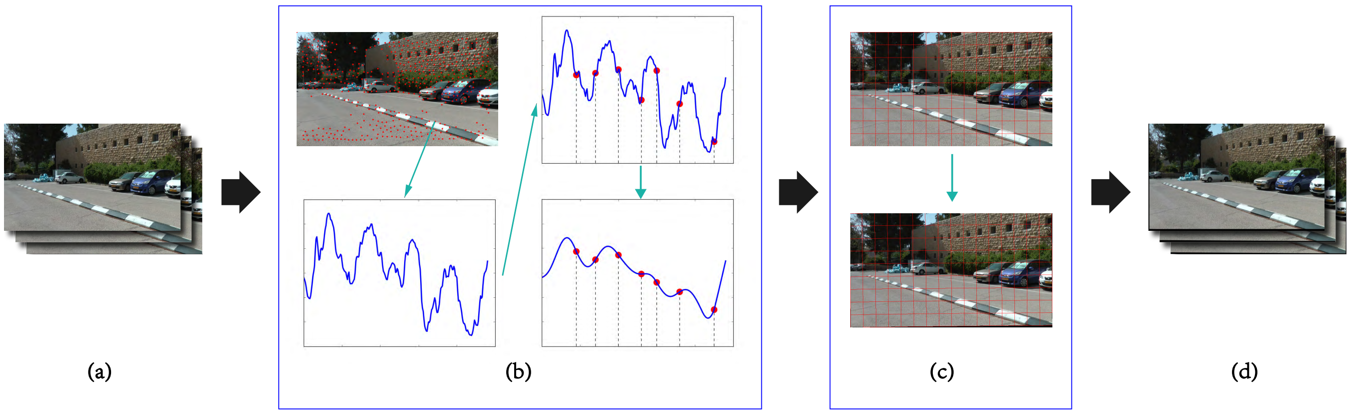


FIGURE 1. Pipeline of our approach. From the input video (a), we first extract trajectories and smooth them (b). Then we perform mesh-based warping on every frame (c) to generate the output video (d).

E. MESH-BASED WARPING ALGORITHMS

Warping algorithm is an important part of video stabilization methods. Early methods, including most 2D methods, use single homography to warp every frame. In order to improve visual effect, Liu *et al.* [4] propose a mesh-based warping algorithm. They divide every frame into several grids and calculate one warping homography for each grid. This algorithm is wildly used by some later methods [5], [17], [24]. Zhou *et al.* [15] improve [4] to make it work better on videos with large textureless region. Mesh-based idea is used not only for warping, but also for motion estimation in [25]. This method divides input frames into grids before smoothing and use the set of homographies of grids between two adjacent frames to model the motion between frames. Wang *et al.* [6] propose another mesh-based warping algorithm with a different energy function. But until now, all the mesh-based warping algorithms use optimization, leading to very large time cost.

III. OUR APPROACH

A. OVERVIEW

The general pipeline of our method is shown in Fig. 1. Usually, a trajectory based video stabilization method consists of two steps: trajectory smoothing step and frame warping step. In the trajectory smoothing step, trajectories are extracted from videos and then smoothed by low-pass filters. In the frame warping step, the smoothed trajectories are used as guidance to generate output frames. While keeping the core procedure unchanged, we use some techniques in both steps to make our method much faster than prior methods. As a result, our method can run in real-time while generating comparable stabilization results to state-of-art methods.

1) TRAJECTORY SMOOTHING

The biggest challenge in trajectory smoothing step is that the extracted trajectories do not share the same starting and ending frames, so they cannot be smoothed directly. To solve this problem, [5] generates 'virtual trajectories', which are lengthen from original trajectories and go through the entire video, and smooth them instead of smoothing original trajectories. However, calculating man-made 'virtual trajectories' not only costs a lot of extra time, but also suffers from accumulative error.

Inspired by [5], our algorithm also generates trajectories with the same starting and ending frames. In order to reach high speed, we do not use the lengthening strategy. On the contrary, we split a long video into several clips, ensuring that there are enough trajectories going through every clip. Since the splitting operation is much simpler and faster than the lengthening operation, our method is greatly accelerated. Meanwhile, all the used trajectories are in fact the original trajectories, which makes our algorithm high reliable. Moreover, some other acceleration techniques are also used. The detail of our trajectory smoothing step will be explained in Section III-B.

2) FRAME WARPING

Mesh-grid based spatially-varying warping algorithms are widely used in recent methods, such as [5], [25]. The visual effect of this kind of algorithm is quite good, but its execution speed is very low. This is due to the use of energy optimization, which constraints the relations of all grids in every frame.

In our method, we propose a novel mesh-based warping algorithm which is much faster than previous similar algorithms.The key insight of our algorithm is to avoid solving time consuming energy optimization problems. It can be obviously seen that, after we divide one frame into grids, each grid is controlled by its four vertices and the whole frame is controlled by the vertices of all grids. For each vertex, we apply a 2D warping on it to calculate its stabilized position. The warping matrices are calculated only using some nearby trajectories, and this operation can be simplified to several matrix multiplications. Furthermore, a matrix operation lib is used for acceleration. Our mesh-based warping algorithm will be explained in Section III-C.

3) NOTATION

We first explain the notation in this paper. Supposing that the input video consists of n frames, we denote these input frames as $\{I_i\}, i \in \{1, n\}$. The extracted m trajectories are denoted as $\{T_j\}, j \in \{1, m\}$. Since each trajectory T_j includes a set of corresponding points in video frames, we denote the position of trajectory T_j in frame I_i as T_j^i.

B. TRAJECTORY SMOOTHING

1) TRAJECTORY EXTRACTION

As the first step, we extract trajectory from input video using KLT [26] feature tracking algorithm. KLT algorithm is widely used in video stabilization methods to extract trajectories. An alternative algorithm is SIFT [27], which can extract and match feature points between arbitrary frames. However, for videos, KLT algorithm is more popular than SIFT algorithm because it is faster and more efficient.

Many free implementations of KLT algorithm can be obtained from Internet, either as source code or as software, but their running speed are different. For example, VOODOO camera tracker [28] is a software which can extract trajectories and reconstruct 3D scene from videos. Some video stabilization methods [4]–[6] directly use the extracted trajectories of VOODOO camera tracker. However, we tested many implementations, including VOODOO camera tracker, but none of them is fast enough for our method. Therefore, we implement our own version based on the OpenCV optical flow lib [29].

Acceleration: OpenCV optical flow lib is an open source library which provides functions of many optical flow algorithms, including KLT. It is written in C/C++, so it runs very fast. Using OpenCV, we implemented our trajectory extraction algorithm. During the implementation, we use the following two strategies for acceleration:

1) Only extracting high reliable trajectories. Some KLT implementations track all available trajectories and leave further choice to users, which wastes a lot of time tracking unnecessary trajectories. In fact, the quality of a feature point in KLT is determined by the minimal eigenvalue of gradient matrices of that pixel. Since we do not need all trajectories, especially the low reliable ones, we set a threshold for eigenvalue to only track high reliable trajectories. The time cost then can be reduced.
2) Excluding the consistency check step. Some KLT implementations have an optional consistency check step, in which excessive trajectories are extracted before outliers are removed. For example, in this online implementation [30], translation, similarity or affine transformations are used to check the consistency between frames. This step is originally optional and we find that it is totally unnecessary for our method, so we do not include this step to save some time. Notice that this step is in fact estimating a 2D motion model between frames, same as the motion estimation step in 2D video stabilization methods, and remove trajectories which not fit this model. However, we are going to use these trajectories to estimate a more complex motion model in Section III-C, so the consistency check step using a 2D motion model is not appropriate for our method. On the other hand, we only track high reliable trajectories, so we do not need this step to remove low reliable trajectories.

2) VIDEO SPLITTING

As mentioned in Section III-A, a set of trajectories cannot be smoothed at the same time if they do not share the same starting and ending frames. Unfortunately, the extracted trajectories are randomly distributed, so they need to be pre-processed before being smoothed. The trajectory preprocessing algorithms in previous similar methods usually cost a lot of time, so we propose a fast trajectory preprocessing algorithm in this section.

a: ACCELERATION

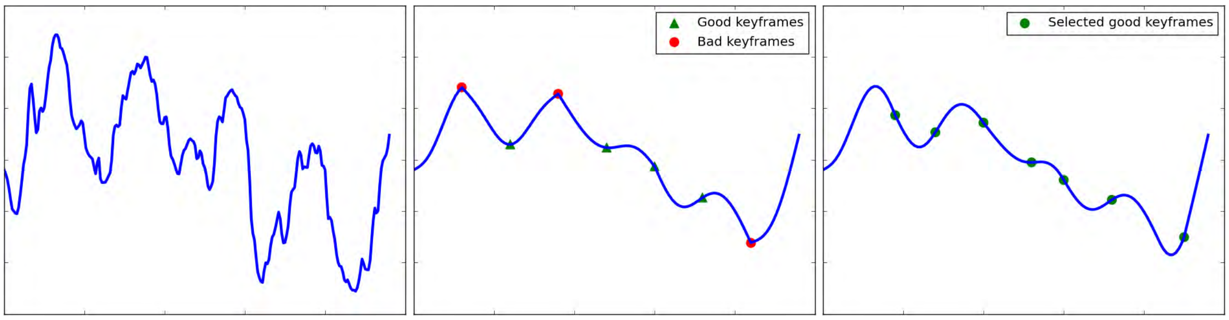
In order to reduce time cost, our key idea is not to introduce extra computation. Therefore, we do not lengthen extracted trajectories like [5] does. Instead, we split the input video into several clips. Trajectories are split at the same time and only trajectories going through arbitrary clip are kept. During this precedure, it is ensured that in each clip, there are enough trajectories going though all frames of that clip. These trajectories can used to represent this clip and then be smoothed. Since the trajectories are all originally extracted from the video, little extra computation is introduced. Therefore the algorithm's speed is very fast.

We can simply smooth trajectories in each clip using any low-pass filter, but it is difficult to smooth trajectories in the keyframes. Since one keyframe belongs to two adjacent clips (except for the first and the last keyframe), if it is smoothed independently in both clips, the two results may be different and thus discontinuity is introduced. We use a simple but effective way to solve this problem: keeping the keyframes fixed and only smoothing other frames. The continuity of adjacent clips then can be kept.

b: KEYFRAME SELECTION

After a video is split into several clips, each two adjacent clips share one *keyframe*. Once the keyframes are selected, the clips are determined. Obviously, the first and the last frame of the input video must be fixed as the first and the last keyframe. Thus the video splitting problem turns into a keyframe selection problem.

The strategy of keyframe selection is very important because the performance of stabilization is strongly affected by the positions of keyframes. Fig. 2 shows one original trajectory and its smoothed results using two keyframe selection strategies. In Fig. 2 (b), the keyframes are selected uniformly, i.e.,all clips have the same length. This is the simplest strategy, but the positions of keyframes are not well planed. We can see that every clip is well smoothed, but there is sudden



(a) (b) (c)

FIGURE 2. Comparison of video splitting strategy. (a) Initial trajectory. (b) Result using constant interval strategy. There is sudden motion change at some keyframes (red circles). (c) Result using our video splitting strategy. All keyframes are carefully selected to avoid sudden motion change.

motion change near some bad keyframes. The red circles indicate bad keyframes, we can see that they are all peaks and valleys of the original trajectories. Since the keyframes are fixed, sudden motion changes keeps unsmoothed near the bad keyframes.

In order to achieve better performance, our splitting algorithm carefully chooses the keyframes. First we describe what a good keyframe should be. From Fig. 2 (b) we know that bad keyframes appear at the peaks and valleys of the trajectories. Therefore, we select frames in the middle of peaks and valleys as keyframes. We use the following equation to measure how much a frame I_i is 'in the middle of' a set of continuous frames from I_{begin} to I_{end}:

$$E(I_i) = \sum_{j=begin}^{end} \sum_{k} ||T_k^i - T_k^j||_2 \tag{1}$$

where k enumerates all the trajectories going through the frames from I_{begin} to I_{end}. It can be easily seen that a good keyframe has a smaller value and vise-versa.

In (1), I_{begin} and I_{end} define the searching range, so they need to be determined before searching for the next keyframe. If a clip is too short, it cannot be well smoothed because the filtering kernel is small. If a clip is too long, there may not be enough trajectories going through it. Supposing the last keyframe is I_{last} (for the first clip, it is set to the first frame I_1), we set $I_{begin} = I_{last+100}$ and $I_{end} = I_{last+200}$ in our experiments. Since the frame rate of videos we used are all 50fps, this means the lengths of clips are 2-4 seconds. After a keyframe is determined, we set I_{last} to the new keyframe and repeat this procedure till the end of the video. The last frame of the video is set to be the last keyframe. The smoothing result using our splitting strategy is shown in Fig. 2 (c), it can be seen that bad keyframes and sudden motion changes are all well avoided.

3) TRAJECTORY FILTERING

In this section, we smooth the pre-processed trajectories using low-pass filters, while keeping trajectories in keyframes fixed. Each video clip is smoothed independently, and only trajectories going through a whole clip are used and other trajectories are ignored. It should be mentioned that, the trajectory sets used in different clips are different, even in adjacent clips.

```
n=0                 1
n=1               1   1
n=2             1   2   1
n=3           1   3   3   1
n=4         1   4   6   4   1
n=5       1   5  10  10   5   1
n=6     1   6  15  20  15   6   1
                    ⋮
```

FIGURE 3. Pascal's triangle.

Gaussian filter is the most widely used linear low-pass filter and they are also widely used in video stabilization methods, but they are not suitable for our method due to the existence of keyframes. When filtering one frame using Gaussian filters, some neighboring frames within a filtering window is involved. However in our method, for a frame near a keyframe, some of its neighboring frames are in another clip. Since the trajectory sets of different video clips are different, these neighboring frames are not available. Another reason why we do not use Gaussian filter is that its coefficients are a little complex and hard to calculate.

Instead of Gaussian filters, binomial filters are used in our method. The kernels of binomial filters are same as the coefficients of binomial expansions. Fig. 3 shows the Pascal's triangle. It determines the coefficients which arise in binomial expansions. The n**th** line of Pascal's triangle indicates the coefficient of the expansion $(x + y)^n$. For example, if $n = 2$, the binomial expansion is $(x + y)^2 = x^2 + 2xy + y^2$. The binomial coefficients (1 2 1) can be used as filter kernel when the windows size is 3. For the reason of symmetry, only windows with odd number size are used in our algorithm.

In order to smooth trajectories near keyframes, we gradually reduce the filtering window for frames near keyframes to ensure that no frame in other clips is used. The distance between a frame I_i and the nearest keyframe I_j is $d = ||i-j||$. If $d > t$, we use the default filter window size t for I_i. If $d < t$, we use $2d + 1$ as the filter window size. We can see that the filter window gets smaller while approaching keyframes and always keep in its own clip. For every keyframe, its window size is 0 in both clips it belongs to, meaning that this keyframe is fixed.

Acceleration: Binomial filters are approximations of Gaussian filters but are more efficient. Multiplications are not needed when calculating coefficients of binomial filters [31], leading to very little time cost. Furthermore, we can calculate all needed coefficients and store them before stabilization. During stabilization, coefficients of filters are read from memory, rather than calculated at runtime. This costs some memory, but the time of calculating coefficients can be saved.

C. FRAME WARPING

In this section, we propose a fast mesh-based frame warping algorithm to generate output frames, using the smoothed trajectories. Mesh-based warping algorithms are frequently used in recent video stabilization method. In a mesh-based warping algorithms, every frame is divided into many grids and one warping homography is calculated for each grid. Compared to traditional warping algorithms which warp every whole frame using only one homography, the visual effect is greatly improved. However, the disadvantage of mesh-based warping algorithms is that they cost much time because they use time consuming global optimization.

The aim of our frame warping algorithm is to reduce the computation complexity of mesh-based warping algorithms while keeping good visual effect. Therefore, instead of using a global optimization, we calculate the homography for every grid separately using nearby trajectories. The perspective transformation matrix of every grid can be represented by the four vertices of the grid. The original positions of these vertices are known when dividing frames into grids, so we need to know the positions of them after stabilization. Since the vertices are very unlikely to be among the extracted trajectories, we need to compute the stabilized positions of the vertices using the extracted trajectories. Considering that using local information rather than global information can reduce the time cost, we use trajectories near every vertex to calculate the stabilized position of it. The detailed procedures of our algorithm are:

1) Divide every frame into rectangle grids uniformly. The size of grids can be decided adaptively according to the frame size. For videos with the frame size of 1280×720 in our experiments, we choose the grid size to be 40×40. For simplicity, we use square grids.
2) Calculate the stabilized position for every vertex. We collect the trajectories near every vertex in the input frame and these trajectories have been smoothed in Section III-B. Using the corresponding positions of these trajectories before and after smoothing, we can estimate a perspective transformation matrix. Then we warp the vertex using this matrix to get the stabilized position.
3) After the stabilized positions of all vertices are got, we can calculate a perspective transformation matrix using the four vertices for every grid.
4) Warp all pixels in input frames by the transformation matrix of the corresponding grid to get output frames.

1) ACCELERATION

The most time consuming step is step 2). For each vertex, we must calculate a transformation matrix using neighboring trajectories. From [32] we know that this problem can be simplified as

$$Ax = b \tag{2}$$

where A and b are know and x are unknown homography (as a vector). Equation (2) becomes an over-determined problem if the trajectories number is larger than 4 and can be solved using least square estimation. In order to reduce time consuming, we use the direct solution:

$$x = (A^T A)^{-1} A^T b \tag{3}$$

Since (3) costs a lot of time for every frame, we use Basic Linear Algebra Subprograms (BLAS) for the matrix multiplication to reach as high speed as possible.

2) COMPARISONS

Since only local informations are used in our warping algorithm, the results may not be global-optimized, but that does not mean the visual effect is not good. Human eyes are sensitive to local inconsistency, so we focus on optimizing it. Even some global inconsistency exists, people can hardly notice it. On the contrary, forcing global optimized may cause some local inconsistency which can be easily noticed. Reference [4] also mentioned that the aim of content-preserving warping algorithm is perceptual plausibility, rather than accurate reconstruction.

In Fig. 4, we show the comparison of our frame warping algorithm and two prior warping algorithm. Since we only want to compare the frame warping algorithms, the inputs for the three algorithms are kept the same using the smoothed trajectories of our trajectory smoothing algorithm.

Early methods warp every frame using only one global homography. The simple single homography motion model leads to very high speed. However, this algorithm only minimizes the overall deviation of one frame, and every local area is not considered independently, leading to local wobble among successive frames. These wobbles do not exist in every single frame and can only be seen when watching videos.

Content-preserving warps [4] is the first mesh-based frame warping algorithm for video stabilization. It generates visual plausible results because relation between neighboring grids is considered. Our frame warping algorithm is also mesh-based and generates very close results to [4] (see Fig. 4). Moreover, the time cost of our algorithm is greatly reduced.

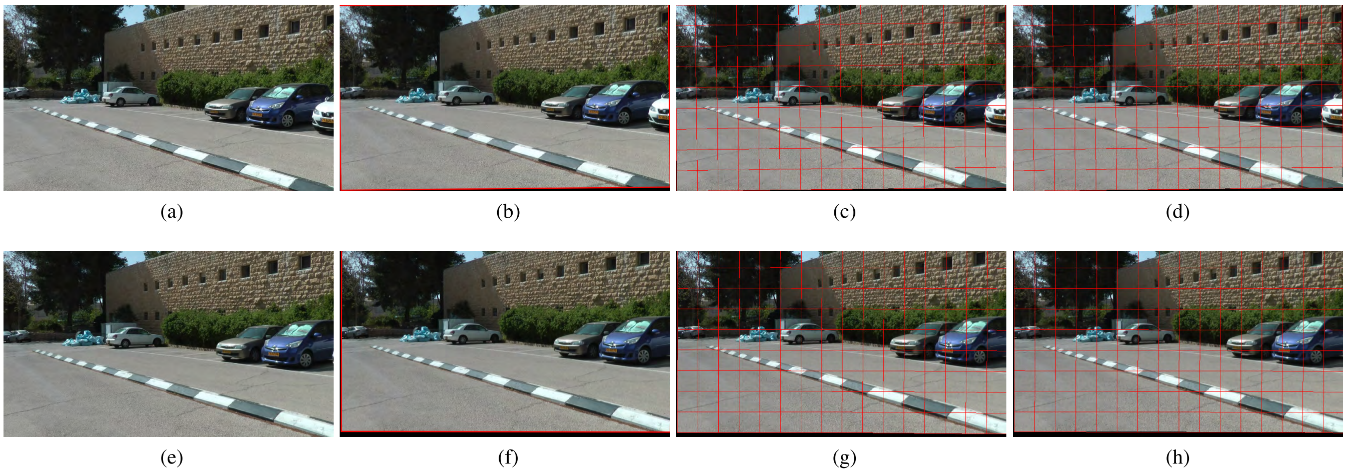


FIGURE 4. Comparison of warping algorithms. (a) Input frame. (b) Result of single homography warping algorithm. (c) Result of content-preserving warps [4]. (d) Result of our mesh-based warping algorithm. (e)-(h) Another set of input and results.

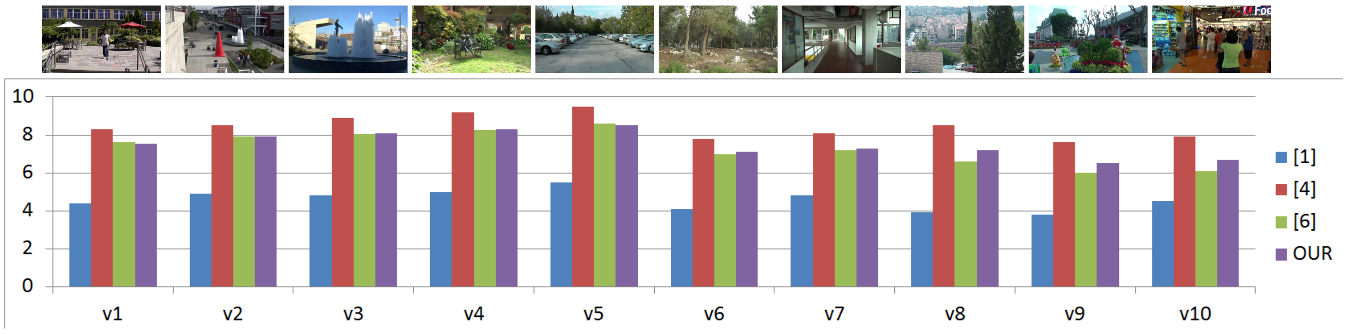


FIGURE 5. User study. Top: Ten videos used in the user study. Bottom: Average scores of the ten videos using methods [1], [4], [6] and our method respectively.

IV. EXPERIMENTS

We tested our method on a wide range of videos to show the effectiveness and robustness of our method. Since videos can not be presented in the paper, please refer to the accompanying demo to see the exact results. In order to show that our method is widely applicable, we use many challenging examples from previous methods, including a video database [33] assembled by [25], which contains six categories of videos.

In this section, we evaluate our method and make comparisons from two aspect: visual effect and execution speed. First we show the visual effect of our method and make comparisons with several previous methods. Both a subjective user study and objective metrics are used for evaluation. Then we make comparisons on execution speed to show the main achievement of our method. At last we discuss the limitation of our method and future works.

A. VISUAL EFFECT

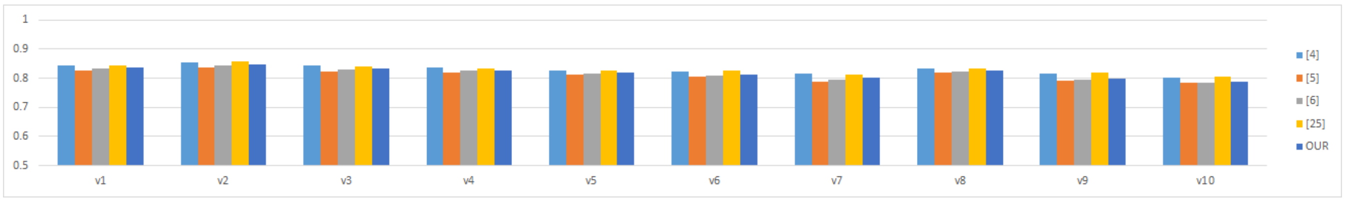
1) USER STUDY

First of all, we conduct a user study to make comparisons between our method and three previous methods. These methods are chosen to represent three categories of stabilization methods: 2D methods [1], 3D methods [4] and trajectory based methods [6]. 10 jittered videos are used as input. Each set of the four output videos are offered to 72 users and they are asked to give a score between 1 and 10 for every output video to represent its stability. A higher score indicates a stabler result.

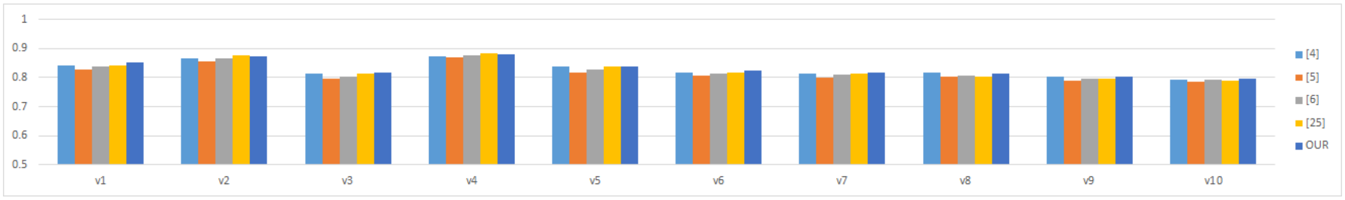
The average scores of the user study are shown in Fig. 5. It can be seen that the score of 2D method [1] is obviously lower than other three methods because its motion model is too simple. The scores of [4], which is a 3D method, are generally the highest because 3D model is the most exact model for motion estimation. The results of [6] and our method look very close to the results of [4]. In fact the differences are very small and hard to distinguish. This is attributed to the techniques used in trajectory based methods. According to the users, the difference between our method and 3D method [4] is hard to notice at the first glance. To summarize, the results of our method are quite comparable to state-of-art methods.

2) QUANTITATIVE EVALUATION

Since user study is very subjective and ineffective, objective metrics are needed to evaluate the quality of stabilization results quantitatively. In this section, we evaluate the results from three aspects: stability, cropping and distortion.



(a)



(b)

FIGURE 6. Quantitative comparisons using (a) bundled camera paths stability and (b) trajectory stability.

a: STABILITY

This metric is used to measure the stability of stabilization results. An objective metric (*bundled camera paths stability*) is proposed in [25]. It uses the energy percentage of low frequency components of bundled camera paths (translation and rotation only) to measure the stability of a video. We use this metric to evaluate the results of our method and some previous methods. From the user study above we know that the visual effect of 2D method [1] is too poor. Therefore we do not compare to it any more in this quantitative evaluation. Instead we use two other methods [5], [25] which generate better results. The comparisons are shown in Fig. 6 (a).

Additionally, we propose a *trajectory stability* metric in this section. Bundled camera paths model is a high-level abstraction of videos. It is inconvenient for methods other than [25] and it may introduce information loss. On the contrary, trajectories are simple and reliable abstraction of videos, so the trajectory stability is accurate and can be easily used by all kinds of methods. The comparisons using this metric are shown in Fig. 6 (b).

Calculating our trajectory stability metric is quite simple. We evaluate the energy percentage of the extracted trajectories similar to [25]. Every trajectory includes x and y components. Each component is treated as 1D signal and the percentage of the lowest frequencies (from the 2nd to the 6th) is calculated. The smaller one between x and y component is used to represent a trajectory and the average value of all trajectories is used to represent a video.

Similar to the results from user study, 3D method [4] usually generates the best results, no matter which metric is used. But the difference between it and other methods are very small. When bundled camera paths stability is used, [25] is the best method apart from [4]. Trajectory based methods [5], [6] are slightly behind it. When trajectory stability is used, the difference between [25] and trajectory based methods get smaller. Our method gets close scores to [4] and [25] with both metrics, and get even high scores in some cases in Fig. 6 (b). This indicates that the visual effect of our method is comparable with state-of-art methods.

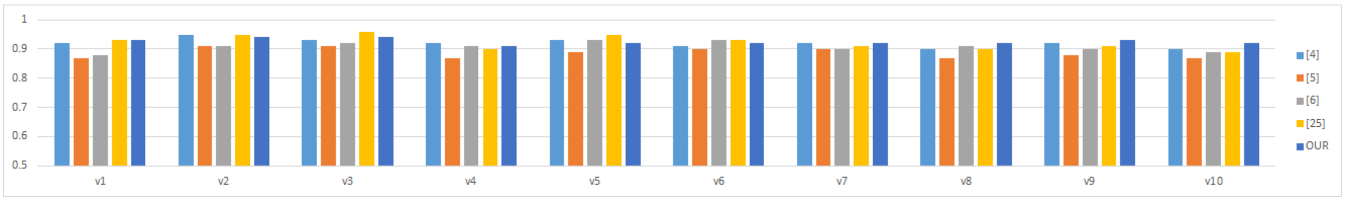
b: CROPPING AND DISTORTION

These two metrics are used to measure the quality of result videos. For each pair of input frame and output frame, we calculate a set of homographies to warp the grid mesh between them, just like what we do in the frame warping step. With this set of homographies known, the overlapping area can be easily calculated. The cropping metric of one frame is defined as the ratio of this overlapping area and the whole input frame. The average ratio of all frames is used to represent the cropping of the entire video. The distortion degree of each homography can be represented by its scaling component. Therefore the distortion degree of one frame can be defined as the average distortion degree of all homographies. Finally, we use the minimum value of all frames to represent the distortion of the entire video.

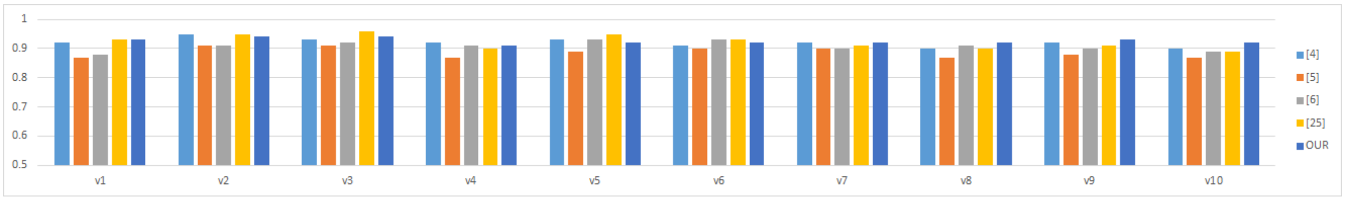
The comparisons of cropping ratio and distortion degree are shown in Fig. 7. We compare our method to the same methods on the same videos as the stability metric. Although our method does not explicitly constraint the cropping ratio, the values of our results are still large enough because every frame is smoothed within a neighboring window, in which all frames overlap a lot. The distortion metric proposed in this section measures the distortion of every grid. Since every grid is warped only using nearby trajectories in the frame warping step, no large distortion is introduced in our method.

B. EXECUTION SPEED

While our method generates comparable results to some state-of-the-art methods, the main contribution of our method is the high execution speed. In the trajectory smoothing step, we split input videos into clips and use binomial filters to smooth the pre-processed trajectories. The time cost is very little and can even be ignored. In the frame warping step, computation complexity is reduced by estimating homography for



(a)



(b)

FIGURE 7. Comparisons of (a) cropping ratio and (b) distortion degree.

TABLE 1. Execution speed comparison.

method	video size	motion estimation (ms)	motion compensation (ms)	image reconstruction (ms)	total (ms)
[1]	1280 × 720	50	20	10	80
[25]	1280 × 720	300	62	30	392
[4]		VOODOO		500	
[5]	1280 × 720	VOODOO	500	500([4])	1200
[6]		200(VOODOO)			real-time[a]
[17]	640 × 360	140	2	100	242
OUR	1280 × 720	11	1	16	28

[a]Ignoring trajectory extraction time.

each grid directly using nearby trajectories, rather than using a global energy optimization.

We run our method on a desktop PC with Intel i5 3.1GHz quad-core CPU and 4G RAM. We use KLT [26] to extract trajectories from videos. For a 10*s* video of 1280 × 720 resolution, typically we extract 300-500 trajectories. The default grid size of our mesh-based warping algorithms is 40 × 40. Therefore, we get a 32 × 18 grid for the above video. In our experiment, the speed of our program is averagely 30fps, which is faster than most state-of-art methods.

In TABLE 1, we show the execution speed of our method and six related methods. Here, we divide every method into three modules: motion estimation, motion compensation and image composition, according to [34]. Execution time of each module and the total time is presented in the table. For trajectory based methods, including our method, these three modules correspond to trajectory extraction step, trajectory smoothing step and frame warping step respectively. Some methods even use other programs, such as Voodoo camera tracker [28], to track high quality KLT features, which costs a lot of time. It should be mentioned that the statistics of related methods come from corresponding papers and there are some unknown values which are left blank in the table. Additionally,

From TABLE 1 we can see the general time cost of different categories of methods. 2D methods [1] are not slow but they are seldom used now because of poor visual effect. 3D methods [4] costs lots of time due to their high complexity. [6] is a trajectory based method that is claimed to be real-time, but the time cost of feature tracking step is ignored. Other state-of-art methods are faster than 3D methods but still quite slow. Our method is real-time. Since execution speed is not the main concern of these methods, precise time cost is not provided in corresponding paper. But without detailed statistics, we can still see that our method is at least 30 times faster than [4], and about 10 times faster than other methods. This proves that our method makes a great improvement on execution speed.

C. LIMITATION AND FUTURE WORKS

Although our method reaches real time, its execution speed is still lower than previous real-time methods. This is because the motion model used in our method is more complex than previous methods. Besides, we have not implement our method using GPU or any hardware, such as FPGAs, on which our method can be further accelerated.

Since there is much matrix computation in our method, we plan to implement it using GPU, to achieve better execution speed. Furthermore, we want to implement our method on mobile devices, such as mobile phones, so that users can use it while or right after capturing videos. In this case we can fully utilize the advantage of real-time methods.

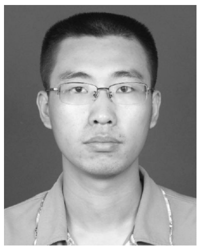
V. CONCLUSION

In this paper, we propose a high quality real-time video stabilization method. Unlike previous real-time methods which only use simple 2D motion models, our method use trajectory based motion model and thus can generate high quality results. Our method contains a trajectory smoothing step and a mesh-based frame warping step. Many acceleration techniques are used in our method to make it much faster than prior trajectory based methods. Meanwhile, the visual effect of our method is still comparable to state-of-art video stabilization methods. Experiments on many different kinds of videos show the good visual effect of our method and the speed data show its high speed.

ACKNOWLEDGEMENT
The authors thank all the reviewers for their helpful advice.

REFERENCES

[1] Y. Matsushita, E. Ofek, X. Tang, and H.-Y. Shum, "Full-frame video stabilization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1. Jun. 2005, pp. 50–57.
[2] M. L. Gleicher and F. Liu, "Re-cinematography: Improving the camera dynamics of casual video," in *Proc. 15th ACM Int. Conf. Multimedia*, 2007, pp. 27–36.
[3] G.-F. Zhang, W. Hua, X.-Y. Qin, Y.-L. Shao, and H.-J. Bao, "Video stabilization based on a 3D perspective camera model," *Vis. Comput.*, vol. 25, no. 11, pp. 997–1008, Oct. 2009.
[4] F. Liu, M. Gleicher, H.-L. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 44:1–44:9, Aug. 2009.
[5] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM Trans. Graph.*, vol. 31, no. 5, pp. 126:1–126:10, Sep. 2012.
[6] Y.-S. Wang, F. Liu, P.-S. Hsu, and T.-Y. Lee, "Spatially and temporally optimized video stabilization," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 8, pp. 1354–1361, Aug. 2013.
[7] L. Araneda and M. Figueroa, "Real-time digital video stabilization on an FPGA," in *Proc. IEEE 17th Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2014, pp. 90–97.
[8] F. Yong-Jie, "Real-time digital video stabilization system based on FPGA," in *Proc. IEEE 9th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2014, pp. 436–438.
[9] D. Dimov and A. Nikolov, "Real time video stabilization for handheld devices," in *Proc. ACM 15th Int. Conf. Comput. Syst. Technol.*, 2014, pp. 124–133.
[10] C. Morimoto and R. Chellappa, "Fast electronic digital image stabilization," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Aug. 1996, pp. 284–288.
[11] M. Irani, B. Rousso, and S. Peleg, "Recovery of ego-motion using image stabilization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 1994, pp. 454–460.
[12] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 225–232.
[13] Y.-G. Kim, V. R. Jayanthi, and I.-S. Kweon, "System-on-chip solution of video stabilization for CMOS image sensors in hand-held devices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 10, pp. 1401–1414, Oct. 2011.
[14] J. Zhou, H. Hu, and D. Wan, "Video stabilization and completion using two cameras," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 12, pp. 1879–1889, Dec. 2011.
[15] Z. Zhou, H. Jin, and Y. Ma, "Plane-based content preserving warps for video stabilization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 2299–2306.
[16] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 89–95.
[17] F. Liu, M. Gleicher, J. Wang, H.-L. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Trans. Graph.*, vol. 30, no. 1, pp. 4:1–4:10, Feb. 2011.
[18] T.-H. Tsai, C.-L. Fang, and H.-M. Chuang, "Design and implementation of efficient video stabilization engine using maximum *a posteriori* estimation and motion energy smoothing approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 6, pp. 817–830, Jun. 2012.
[19] C.-T. Lin, C.-T. Hong, and C.-T. Yang, "Real-time digital image stabilization system using modified proportional integrated controller," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 3, pp. 427–431, Mar. 2009.
[20] W. G. Aguilar and C. Angulo, "Real-time video stabilization without phantom movements for micro aerial vehicles," *EURASIP J. Image Video Process.*, vol. 2014, no. 1, pp. 1–13, 2014.
[21] J. Li, T. Xu, and K. Zhang, "Real-time feature-based video stabilization on FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 907–919, Apr. 2017.
[22] A. Litvin, J. Konrad, and W. C. Karl, "Probabilistic video stabilization using Kalman filtering and mosaicing," *Proc. SPIE*, vol. 5022, pp. 663–675, May 2003.
[23] J. Dong and H. Liu, "Video stabilization for strict real-time applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 716–724, Apr. 2017.
[24] Z.-Q. Wang, L. Zhang, and H. Huang, "Multiplane video stabilization," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 265–273, Sep. 2013.
[25] S.-C. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 78:1–78:10, Jul. 2013.
[26] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 1994, pp. 593–600.
[27] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.
[28] T. Thormählen and H. Broszio. (2008). *Voodoo Camera Tracker: A Tool for the Integration of Virtual and Real Scenes*. [Online]. Available: http://viscoda.eu/en/products/non-commercial/voodoo-camera-tracker
[29] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, 2000, Art. no. 2236121. [Online]. Available: https://github.com/opencv/opencv/wiki/CiteOpenCV
[30] *Klt: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker*. Accessed: Mar. 3, 2018. [Online]. Available: http://cecas.clemson.edu/~stb/klt/
[31] M. Aubury and W. Luk, "Binomial filters," *J. VLSI signal Process. Syst. signal, image video Technol.*, vol. 12, no. 1, pp. 35–50, 1996.
[32] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[33] *Video Database*. Accessed: Apr. 10, 2018. [Online]. Available: https://1drv.ms/f/s!AlmVhRElNfJnaUYrc8AOsv9GSAU
[34] C. Morimoto and R. Chellappa, "Evaluation of image stabilization algorithms," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, vol. 5. May 1998, pp. 2789–2792.



ZHONGQIANG WANG received the bachelor's degree from Xi'an Jiaotong University, China, in 2010, where he is currently pursuing the Ph.D. degree with the School of Electronics and Information Engineering. His main research interests include image and video processing.

LEI ZHANG (M'14) received the B.S. and Ph.D. degrees in applied mathematics from Zhejiang University, China, in 2004 and 2009, respectively. He is currently an Associate Professor with the School of Computer Science, Beijing Institute of Technology. His research interests include computer graphics, image, and video processing.

HUA HUANG (M'08) received the B.S. and Ph.D. degrees from Xi'an Jiaotong University, China, in 1996 and 2006, respectively. He is currently a Professor with the School of Computer Science, Beijing Institute of Technology, China. He is also an Adjunct Professor with the School of Electronics and Information Engineering, Xi'an Jiaotong University. His main research interests include image and video processing and computer graphics.