CodingFlow: Enable Video Coding for Video Stabilization

Shuaicheng Liu, *Member, IEEE*, Mingyu Li, Shuyuan Zhu, *Member, IEEE*, and Bing Zeng, *Fellow, IEEE*

***Abstract*— Video coding focuses on reducing the data size of videos. Video stabilization targets at removing shaky camera motions. In this paper, we enable video coding for video stabilization by constructing the camera motions based on the motion vectors employed in the video coding. The existing stabilization methods rely heavily on image features for the recovery of camera motions. However, feature tracking is time-consuming and prone to errors. On the other hand, nearly all captured videos have been compressed before any further processing and such a compression has produced a rich set of block-based motion vectors that can be utilized for estimating the camera motion. More specifically, video stabilization requires camera motions between two adjacent frames. However, motion vectors extracted from video coding may refer to non-adjacent frames. We first show that these non-adjacent motions can be transformed into adjacent motions such that each coding block within a frame contains a motion vector referring to its adjacent previous frame. Then, we regularize these motion vectors to yield a spatially-smoothed motion field at each frame, named as *CodingFlow*, which is optimized for a spatially-variant motion compensation. Based on CodingFlow, we finally design a grid-based 2D method to accomplish the video stabilization. Our method is evaluated in terms of efficiency and stabilization quality, both quantitatively and qualitatively, which shows that our method can achieve high-quality results compared with the state-of-the-art methods (feature-based).**

***Index Terms*— CodingFlow, video stabilization, video coding, camera motion, mesh grid, spatial-temporal optimization.**

I. Introduction

VIDEOS captured by moving devices, e.g., hand-held cameras or cameras mounted on vehicles, often suffer from strong shakiness, which severely damages the viewing experiences. Video stabilization [1]–[9] tries to improve the video quality by removing unwanted camera motions, thus rendering videos with smooth transitions in the temporal domain. The existing video stabilization methods heavily rely on image features [10] for the recovery of camera motions. Some methods [1], [5] match image features between neighboring frames while others [2], [3] track them for a certain range of frames (e.g., 30 consecutive frames [11]). Feature matching and tracking are very time-consuming, e.g., they take $70\% \sim 80\%$ of the run-time of the whole stabilization system. Moreover, they are sensitive to the type of camera motions and scene contents. For instance, video frames under quick camera motions (e.g., quick swing or fast zooming) often yield low-quality features so that the number of matched features or the length of tracked motions is limited. Textureless regions is another challenge that disturbs a high-quality feature detection. On the other hand, some methods [12], [13] rely on gyroscopes for the recovery of camera motions. However, gyroscopes can only smooth rotational motions, leaving translational motions uncompensated. Furthermore, gyroscopes are not applicable to all levels of cameras.

Video coding aims at removing spatial (intra coding mode) and temporal (inter coding mode) redundancies within the raw video data. Nearly all captured videos have been compressed before storage and distribution, because an uncompressed video can easily reach dozens of gigabytes. In the inter coding mode [14], [15], motion vectors of image blocks are estimated across multiple frames. We find that these motion vectors not only link image blocks with similar contents, but also provide strong cues for revealing the camera motion. However, these valuable motion cues are ignored completely in all existing feature-based video stabilization methods. In this work, we propose to use the motion vectors extracted from the video coding for video stabilization. Because all processing associated with image features (feature detection, feature matching, and feature tracking) is no longer needed, our proposed method can offer a much lower computational complexity. In the meantime, we will show that our method can provide a competitive performance as compared with the feature-based methods

Early stabilization methods [1], [7], [16] deal with global motions that can be described by a single parametric transformation model (e.g., affine or homography). Nevertheless, videos containing parallax and depth variations often require spatially-variant motion representations, where each part of a frame can be smoothed differently, leading to smaller motion residuals. Therefore, we are more interested in methods that can handle spatially-variant motions. In general, these methods can be classified into two categories, smoothing long feature tracks [2], [3], [11], [17] and smoothing multiple transformation models [5], [6], [8]. If long feature tracks are present, the full 3D or partial 3D structures can be recovered, and

Manuscript received September 27, 2016; revised February 15, 2017 and March 19, 2017; accepted April 14, 2017. Date of publication April 24, 2017; date of current version May 9, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61502079, Grant 61672134, and Grant 61370148, in part the Fundamental Research Funds for Central Universities of China under Grant ZYGX2016J038 and Grant ZYGX2015KYQD054, and in part by the 111 Projects under Grant B17008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zhengguo LI. *(Corresponding authors: Shuyuan Zhu; Bing Zeng.)*

The authors are with the Institute of Image Processing, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: eezsy@uestc.edu.cn; eezeng@uestc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2697759

smoothing of them can lead to a high-quality stabilization. However, long feature tracks are usually hard to obtain in consumer videos. Especially, both the track length and the number of tracks drop quickly when the camera undergoes a quick rotation or zooming.

On the other hand, smoothing of multiple transformation models is more robust to various types of camera motions as it only requires the feature matching between adjacent frames. Here, a single frame is usually divided into several parts and a local transformation model is estimated for each part, thus yielding multiple affines [8], homographies [5], or even non-parametric dense flows [6] that can represent spatially-variant motions. In this work, we follow the multiple transformation approach, but replace the feature matching by the motion vectors extracted from the video coding.

Notably, motion vectors extracted in video coding may not necessarily correspond to the camera motion, as they are not designed for estimating the camera motion. In fact, a motion vector in video coding refers to a place that yields the minimum compensation residue. After extensive experiments, not surprisingly, we have found that the majority of motion vectors do coincide with the camera motion. That is, if an image block moves to a place according to the camera motion (e.g., the optical flow of that block), this place also has a high probability to yield the minimum residue. Of course, we also notice that a small part of "outliers" violates the camera motion, which would be regularized in our method.

In this paper, we propose a new motion model for video stabilization, named as CodingFlow, that is constructed from the motion vectors estimated during the video coding procedure. CodingFlow is a spatially-smoothed and sparse motion field that describes camera motions between adjacent frames. For P frames (predictive frames) and B frames (bidirectional references), we design two recursive algorithms to manipulate the inter motions of image blocks at frame t so that each only refers to its previous frame $t-1$. Then, a regular grid mesh is placed onto each frame. The motion vector of each grid is propagated to its mesh vertices. Notably, a mesh vertex may receive multiple motion vectors. CodingFlow is produced by assigning each vertex an unique motion vector through median filters for spatial smoothness. Next, the path optimization is conducted on the *vertex profiles*, which are motion vectors collected at the same vertex position through CodingFlow over time. Due to the property of strong spatial smoothness, vertex profiles can be optimized independently such that the path optimization can be conducted in parallel for an improved efficiency. Finally, the stabilized frames are rendered according to the smoothed vertex positions. In summary, the main contributions of our work are:

- A method of manipulating inter-frame motion vectors such that these manipulated motion vectors of each frame only refers to its adjacent previous frame.
- CodingFlow, a sparse motion field that is generated from video coding for spatially-variant motion representation and stabilization.

The rest of the paper is organized as follows. Sec. II presents an overview of the related works. Sec. III discusses details on how the motion vectors are extracted and then manipulated with respect to P-frames and B-frames. Sec. IV presents the estimation of the CodingFlow. Sec. V discusses how to stabilize CodingFlow for video stabilization. Sec. VI discusses our approach in several aspects. Experiments are presented in Sec. VII with comparisons to previous approaches. Finally, Sec. VIII concludes the paper.

II. Related Works

A. Video Coding

Video coding focuses on reducing the data size by removing redundancies in the raw video data. There are two types of redundancies in the raw video data: the spatial redundancy and the temporal redundancy. The former one refers to the structural similarity between neighboring pixels within a single frame and the later one stands for the content similarity across several adjacent frames. In the classical video coding, two coding modes, i.e., the intra coding and the inter coding, are designed to remove these two types of redundancies accordingly. More specifically, removing of redundancies in video frames is performed on the block-by-block basis. In both coding modes, the discrete cosine transform (DCT) [18], quantization, and entropy coding are often utilized to compress each image block.

In the redundency-removing procedure, DCT plays a rather important role as it transforms an image block in the pixel domain into the frequency domain so that the block's energy is much more compacted into only a few coefficients. Recently, several new transforms, such as the directional DCT (DDCT) [19], [20] and some novel unitary transforms [21], have been proposed to further improve the performance. Meanwhile, the integer DCT [22] has been developed to support the hardware-based implementation. Furthermore, the rate-distortion optimization (RDO) [23] principle has been introduced in video coding to balance the bit-consumption and coding distortion during the compression.

Meanwhile, it has been demonstrated for a long time that removing of temporal redundancies would become much more efficient through the so-called motion-compensated prediction (MCP) [24], [25]. Because of this, MCP has been adopted in all video codecs to construct the inter coding mode. In MCP, a motion vector (MV) is used to identify a predictive block from the previously-coded frames for the current block. The procedure of searching for the best MV is called motion estimation (ME). The most straightforward way to find the optimal MV is to do an exhaustive search. However, such a searching strategy leads to a high cost. To solve this problem, a large number of fast search algorithms have been proposed. Several widely-recognized algorithms are the three-step search (TSS) [26], the new three-step search (NTSS) [27], the four-step search (4SS) [28], and the diamond-shaped search [29].

B. Video Stabilization

Video stabilization methods can be roughly categorized into 3D [2], [9], [30], [31], 2D [1], [5]–[7], [16] and 2.5D [3], [11] methods according to their different motion models.

Liu *et al.* proposed a 3D method that relies on the full 3D reconstruction [2]. Liu *et al.* [30] and Smith *et al.* [31] adopted a depth camera and light-field camera for the robust 3D recovery. Moreover, plane constraints can be introduced for the better regularization [9], [32]. Meanwhile, gyroscopes were also adopted for the efficient recovery of 3D orientations [12], [13]. Jia and Evans constrained the 3D rotations during the path smoothing [33]. In general, these full 3D reconstruction is fragile because of tracking failures and scene degenerations.

To solve this problem, some 2.5D methods can relax the requirement of full 3D reconstruction to some partial 3D information. For instance, Liu *et al.* proposed to smooth feature tracks in a subspace to maintain 3D relationship during stabilization [3], and Goldstein and Fattal used epipolar geometry for 3D coherence maintaining [11]. Such partial 3D information is embedded in the long feature trajectory. However, long feature tracks are hard to obtain in many consumer videos due to quick camera motions.

In contrast, the 2D methods only match features between neighboring frames. Thus, they are more robust to different types of camera motions. Early 2D methods estimated a single homography between each pair of two neighboring frames and then smoothed all estimated homographies after cancatenating them, referred to as the 2D camera path, for stabilization [1], [4]. More recently, Grundmann *et al.* estimated a single homography between two adjacent frames and employed cinematographic rules for the camera path design [7]. Then, the single homography is divided into a homography array for the rolling shutter correction [8]. Meanwhile, Liu *et al.* adopted "as-similar-as-possible" image warping for spatially-variant motion estimation [5] in which the bundled camera paths were approximated and smoothed for the stabilization. User interactions could further improve the performance [34]. Liu *et al.* estimated a dense and spatially-smoothed optical flow between two adjacent frames and upgraded the raw flow into spatially smooth steadflow for stabilization [6]. Recently, the dense flow is simplified into a sparse flow according to image feature matches, aiming at online processing [35]. In this work, we adopt a similar idea, but estimate a spatially-smoothed sparse flow using the motion vectors extracted from the video coding.

III. Inter Motion Manipulation

Although the newest video coding standard is H.265/HEVC, we choose to describe our inter motion manipulation based on the H.264/AVC framework because its structure is not as complicated as H.265/HEVC. Note that exactly the same processing applies to H.265/HEVC. In H.264/AVC, each frame is processed in units of macroblocks of size 16×16. Each macroblock can be encoded in **intra** or **inter** mode, where the former aims at reconstructing the frame using samples in the current frame while the latter uses motion-compensated prediction from one or more reference frames. We focus on the inter mode in this work.

In H.264/AVC, *a tree-structured motion compensation strategy* is supported, where each macroblock can be further divided into smaller sizes, including 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4. The division is performed by the encoder according to a rate-distortion optimization (RDO). For example, some image regions in one frame are divided into 16×16 due to their homogeneities, while others with rich image details may correspond to a smaller size.

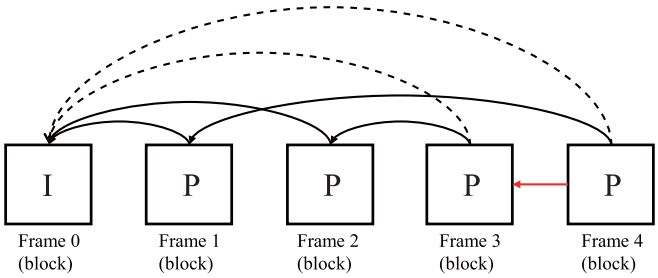


Fig. 1. An example of inter motion manipulation on P-frames. All blocks are with the same spatial location at different frames. Each block refers to a previous frame. The motion vectors are indicated by solid arrows. The red arrow indicates the desired inter motion at the fourth frame, which will be derived by our inter-motion manipulation algorithm. The intermediate motion vectors are shown in dashed arrows.

Upon receiving the compressed bit-stream, we can easily extract all motion vectors. More importantly, we know to which block at the current frame a motion vector corresponds, what is the block size, and to which reference frame the motion vector indicates. In our inter motion manipulation, we fix our processing unit with the size 4×4. That is, if a block has a bigger size, its motion vector is duplicated and assigned to all processing units (of size 4×4) in the block. For example, a 8×16 block is divided into eight 4×4 processing units and all units share the same motion vector.

In general, a video encoder uses the 4:2:0 YUV format for motion prediction, where the luminance component Y is twice of height and width as two chrominance components U and V. In this work, we only extract motion vectors from the luminance component for the motion manipulation. The resulted CodingFlow is resized accordingly for the chrominance components.

There are three types of frames in video coding, i.e., I, P, and B frames. An I-frame is regarded as the key frame. It is the base frame that is referred by P-frames and B-frames. In our case, the first frame is set to be I. A **P-frame** can refer to its previously-coded frames (not necessarily the neighboring frame) for motion prediction. A **B-frame** can refer to one past and one future coded frames (again, not necessarily the neighboring frames).

A. Manipulation of P-Frame

Figure 1 shows an example of inter motion manipulation of P-frames. In this example, we show five blocks with the same spatial location in five consecutive frames, in which the first frame is an I-frame and the rest are P-frames. Except for the first one, each of the other 4 blocks refers to one previous frame for motion prediction, where the motion vectors are indicated by solid arrows. Specifically, the first and second blocks refer to the I-frame, the third block refers to the second frame, and the fourth block refers to the first frame. Notice that these referrings are generated automatically by the video encoder because they offer the best coding performance.

Algorithm 1 Calculate $V_{i,i-1}$ for P Frames

Ensure: $V_{i,ref}$ and $V_{i-1,ref'}$
  while $ref \neq ref'$ **do**
    if $ref > ref'$ **then**
      $V_{i,ref} = V_{i,ref} + V_{ref,ref->ref}$
    else
      $V_{i-1,ref'} = V_{i-1,ref'} + V_{ref',ref'->ref}$
    end if
  end while
$V_{i,i-1} = V_{i,ref} - V_{i-1,ref'}$

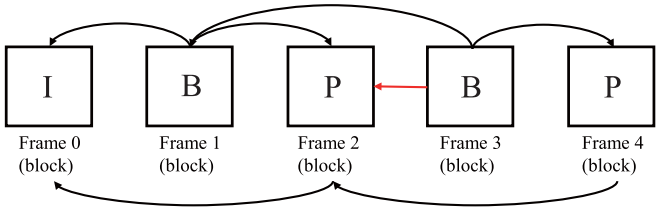


Fig. 2. An example of inter motion manipulation on B-frames. A B-frame can refer to two directions for motion predictions. In the hierarchical coding, P-frames are coded before B-frames. The red arrow indicates the desired inter motion at the third frame.

Our job here is to derive the desired motion vector for the fourth block that needs to refer to the third frame (as indicated by a red arrow).

Let us denote a motion vector as $V_{i,j}$, where i and j are frame indexes, pointing from i to j. For the example shown in Fig. 1, the motion vectors of the 2nd, 3rd, and 4th blocks are denoted as $V_{2,0}$, $V_{3,2}$, and $V_{4,1}$, respectively. In general, two motion vectors can be added up:

$$V_{i,k} = V_{i,j} + V_{j,k}, \tag{1}$$

where k is another frame index ($i \neq j \neq k$). Similarly, we can also perform the substraction as:

$$V_{i,j} = V_{i,k} - V_{j,k}. \tag{2}$$

In the example of Fig. 1, we want to obtain the motion vector $V_{4,3}$, which can be obtained as $V_{4,3} = V_{4,0} - V_{3,0}$, where $V_{4,0}$ and $V_{3,0}$ are dashed arrows. Although $V_{4,0}$ and $V_{3,0}$ do not exist in Fig. 1, they can be obtained as $V_{4,0} = V_{4,1} + V_{1,0}$ and $V_{3,0} = V_{3,2} + V_{2,0}$. Finally, $V_{4,3}$ is obtained as $V_{4,3} = V_{4,1} + V_{1,0} - V_{3,2} - V_{2,0}$.

In fact, the desired neighboring motion vector $V_{i,i-1}$ can always be obtained by manipulations of the available motion vectors. The manipulation algorithm is summarized as Algorithm 1.

B. Manipulation of B-Frame

We follow the hierarchical coding structure where P-frames are encoded before B-frames. A P-frame can refer to a previous P-frame or I-frame, but not B-frame. A B-frame can refer to two directions or either of these two directions for motion predictions. Fig. 2 shows an example of five blocks of the same spatial location at two B-frames, two P-frames and one I-frame. Again, all directional referrings as well as the frame types are determined by the video encoder because they provide the best coding performance. Here, our job is to find the motion vector $V_{3,2}$ indicated by the red arrow at the third frame.

Algorithm 2 Calculate $V_{i,i-1}$ for B Frames

$paths = find\_all\_routes(V_{i,ref}, V_{i-1,ref'})$
$shortest = find\_shortest\_path(paths)$
Ensure: $V_{i,shortest\_ref}$ and $V_{i-1,shortest\_ref'}$
Apply the Algorithm 1

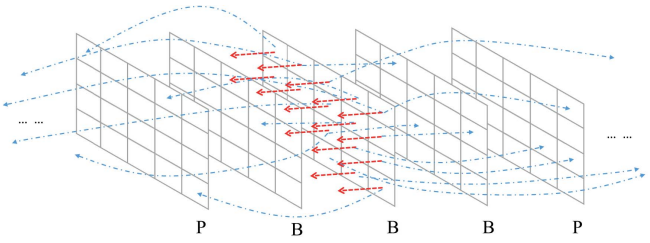


Fig. 3. Blocks in a B-frame can refer to different previous and future frames (the light blue arrows). After the motion manipulation, all blocks point to the previous neighboring frame (the red arrows).

In this example, two candidates are available: $V_{3,2} = V_{3,1} + V_{1,0} - V_{2,0}$ and $V_{3,2} = V_{3,1} + V_{1,0} - (V_{3,4} + V_{4,2} + V_{2,0})$. We can choose the one with the shortest path or the one with the minimum residual. The former provides the smallest shift while the latter evaluates the quality of motion vectors. To calculate the residual, we need to extract the residual error from the coded residual layer and compute the sum of squared difference (SSD) error, which is time consuming. Therefore, we take the candidate with the shortest path. In this example, the first candidate will be chosen.

Note that P-frames are usually separated by B-frames. Therefore, after calculating neighboring motions of all B-frames, bridges are built for P-frames so that all frames can point to their previous adjacent frames. The algorithm for manipulating B-frames is summarized as Algorithm 2.

Figure 3 shows an example of our motion manipulations. Initially, each block in a B-frame can refer to either direction with a different frame interval. After the manipulation, all blocks refer to their adjacent previous frame.

C. Motion Tracking vs. Motion Profile

Before we move on to the estimation of the camera motion, we would like to clarify the difference between motion tracking and motion accumulation. Motion tracking originates for image features, in which features are detected on one frame and tracked to the subsequent frames [10]. For image blocks, tracking will lead a block to different places at different frames. Fig. 4 (top) shows an example in which blocks move according to their motions. Note that some blocks may move outside of a frame, thus terminating the tracking. On the other hand, motion accumulation always collects motions at the same spatial location, leading to a dense coverage of the whole frame, both spatially and temporally. Motion accumulation is first introduced in SteadyFlow [6], where optical flows are

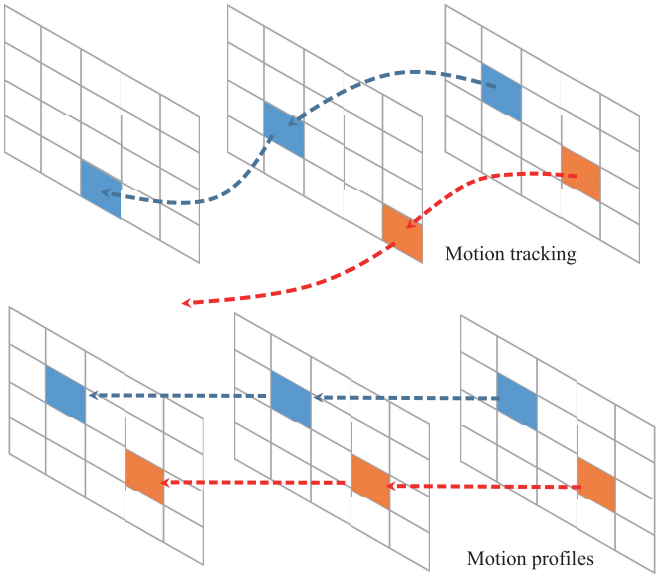


Fig. 4. Motion tracking vs. motion profile. The top row shows an example of the motion tracking where blocks follow their motions to different places. Some of the blocks may move outside the frame boundary. The bottom row shows an example of motion profile where the motion from the same location is collected over time.

estimated between two neighboring frames and a *pixel profile* is produced by collecting motions at a pixel location over time. It has been shown in [6] that the motions recorded in the pixel profile are quite similar to the motions obtained by the corresponding feature tracking, assuming that the tracking starts at the same position with the corresponding motion within the pixel profile.

Here, we borrow the similar idea to construct a *motion profile* by collecting all motion vectors obtained after the motion manipulation described above.

IV. CodingFlow Estimation

CodingFlow is a sparse but regular grid of motions with strong spatial smoothness, generated by manipulating the motion vectors that are extracted from a compressed video bitstream. In this section, we discuss how CodingFlow is estimated and regularized.

The manipulated inter motion vectors are raw motion vectors (red arrows in Fig. 3). These raw motion vectors are usually not applicable for video stabilization because they may contain "errors" that are inconsistent with the camera motion. Note that these "erroneous" motions are actually good motions in the context of video coding, as they generate the minimum residues. In the context of video stabilization, we consider them as errors because they do not reveal the camera motion.

There are several reasons that these errors may occur. First, motions extracted on the dynamic objects are certainly inconsistent with the camera motion. Second, for textureless regions, such as blue sky, sea, white wall, and road, no differentiable textures exist. The block-matching algorithm for motion estimation in these regions is not well constrained so that a random location may provide the best match. Third, the presence of repetitive structures might confuse the

Fig. 5. Left: the raw motion vectors are noisy (red arrows). Right: after a global motion regularization.

blocking-matching. Repetitive structures have been utilized in 3D reconstruction [36], image super-resolution [37], and patch match for image editing [38]. They are preferred in video coding, as they provide additional candidates during the motion prediction. However, motions at these regions are usually inconsistent with the camera motion. The erroneous motions should be excluded in the video stabilization: smoothing them otherwise would introduce severe artifacts.

It is important to point out that our extensive experiments reveal that a majority of matches does follow the camera motion, while only a small amount of them suffers from the inconsistency. Therefore, we can correct these erroneous motions by the correct ones. This regularization procedure consists of two steps, a global pre-process to reject outliers and a local refinement to provide further regularization during the CodingFlow estimation.

A. Global Motion Correction

We use a global homography to identify outliers. Referring to Fig. 5 (left), outliers in the textureless regions can be identified and corrected using a global homography. In particular, a global homography $F(t)$ is estimated with RANSAC outlier removal [39] using raw motions. Then, global motion vectors at all blocks can be computed using the estimated global homography. Specifically, we transform the center coordinates of each block by the homography, and the global motion vector of a block is the difference between the transformed center and the original center. Notably, the whole frame shares a global homography, but their global motion vectors are not the same. For example, if a frame undergoes a clockwise in-plane rotation, a block on the top-left moves towards the up-right direction, but the block on the bottom-right moves towards the bottom-left direction. As a result, each vertex has two motion vectors, the global one and the local one. The angle between them is calculated. If the angle is small, we retain the local motion; otherwise we replace the local motion with the global motion. Fig. 5 (right) shows the result after the motion correction.

Notably, it is of particular importance to overwrite the problematic motions before conducting the stabilization. Fig. 6 shows two examples: with and without the global motion correction. The first example contains some repetitive structures (windows on the background). For a given window, the block-matching algorithm may pick a similar but a different window, thus yielding the motions that could not reveal the camera motion. In the second example, the sky has fewer textures to constrain the block-matching. Local motions

(a) Without motion correction (b) With motion correction

Fig. 6. With and without global motion correction. (a) Without motion correction, the stabilization results suffer from content distortions (highlighted by the arrows). (b) Motion correction can remove such distortions effectively.

at these regions do not coincide with the camera motion. Consequently, smoothing these erroneous motions would lead to severe content distortions as verified by the stabilized results. Specifically, we extracted 14080 motion vectors within a frame for each example shown in Fig. 6. We corrected 23% and 34% of the motions for the first and the second example. Empirically, the percentage of the correction is scene dependent, ranging from 10% $\sim$ 40%. More discussions will be given in the experiment section.

B. CodingFlow Motion Model Estimation

CodingFlow is produced at each frame. Fig. 7 shows an example of producing the CodingFlow at one frame. Specifically, a regular grid mesh is placed onto each frame. The motion of a block (Fig. 7(a)) is propagated to the nearby mesh vertices (Fig. 7(b)). Each block can contribute motion vectors to its surrounding mesh vertices. As a result, a mesh vertex receives multiple motion vectors from the nearby blocks. However, an unique motion vector is required for each vertex, which is achieved by applying a median filter to all candidate motions (Fig. 7(c)). The median filter is frequently used in the dense optical flow estimation. It has been considered as the secret of a high-quality optical flow estimation [40]. It not only rejects errors but also enforces the spatial smoothness. Here, we borrow the similar idea for our sparse motion regularization. The final CodingFlow is generated by applying median filters to all vertices, where each vertex is assigned with one motion vector (Fig. 7(d)).

CodingFlow describes the camera motion between adjacent frames. It is important to note that all motions of CodingFlow come from the video coding. The estimation process of CodingFlow includes a global homography estimation, some motion assignments, and median filtering, all of them are very efficient. No computationally expensive operations are involved, such as feature detection and matching [41], dense optical flow [6], or "as-similar-as-possible" mesh warping [5].

V. Video Stabilization

In this section, we describe how to stabilize a video after CodingFlow is obtained for each frame. For the sake of completeness, we first describe the method of smoothing a single camera path - similar approaches are also reported in methods [1], [7]. Then, we describe how to smooth multiple paths for the spatially-variant motion compensation.

A. Stabilizing Global Camera Path

In the previous section, a homography $F(t)$ between two adjacent frames has been estimated for the global motion correction. In fact, $F(t)$ encodes the global camera motion of neighboring frames. It is estimated by image features in traditional 2D stabilization approaches [1], [5], [7]. Here, it is obtained by manipulating motion vectors obtained from the video coding. The camera path at the t-th frame is defined as a concatenation of all these neighboring homographies:

$$C(t) = F(0)F(1)\cdots F(t-1), F(0) = I. \tag{3}$$

Given the original camera path $\{C(t)\}$, the stabilized path $\{P(t)\}$ is obtained by minimizing the energy as follows:

$$\mathcal{O}(\{P(t)\}) = \sum_t \|P(t) - C(t)\|^2 + \sum_t (\lambda_t \sum_{r\in\Omega_t} \omega_{t,r} \|P(t) - P(r)\|^2), \tag{4}$$

where Ω_t denotes a temporal smoothing radius, $w_{t,r}$ is a Gaussian weight that is set to $exp\big(-\|r-t\|^2/(\Omega_t/3)^2\big)$, and λ_t balances two terms.

The first term enforces the stabilized video staying close to the original camera path so as to avoid some artifacts, such as excessive cropping and wobbling. The second term encourages the smoothness of the path. The stabilized frames are generated by warping the input video frames using a transform $B(t)$, defined as $B(t) = C^{-1}(t)P(t)$ [5].

The weighting coefficient λ_t for each frame plays an important role for artifact suppression. If $\lambda_t = 0$ for all t, the optimized path is equal to the original path. As a result, the output video is equal to the input video, leading to no crops and no wobbles. The method of [5] adopts an *iterative refinement* approach to search for the optimal values of λ_t for every frame. Initially, all λ_t's are set to 1 and Eq. (4) is minimized. Then, the cropping of each frame is evaluated. If some frames do not satisfy the pre-requirement, i.e., empty region no more than 20% of a frame, the corresponding λ_t is reduced by a step, e.g., 0.1, and the optimization is re-run for another time. The process terminates until all frames satisfy the pre-requirement.

The drawback of employing a global camera path is that it can not handle spatially-variant motions, which is a much desired property for high-quality stabilization, especially for scenes with depth variations. An improved solution is presented below.

B. Stabilizing CodingFlow

Figure 8 shows an example of CodingFlows at several video frames. A pixel profile is produced by collecting motions at the same pixel location over time [6]. Here, we generate pixel profiles at mesh vertex locations. Fig. 8 shows four profiles at

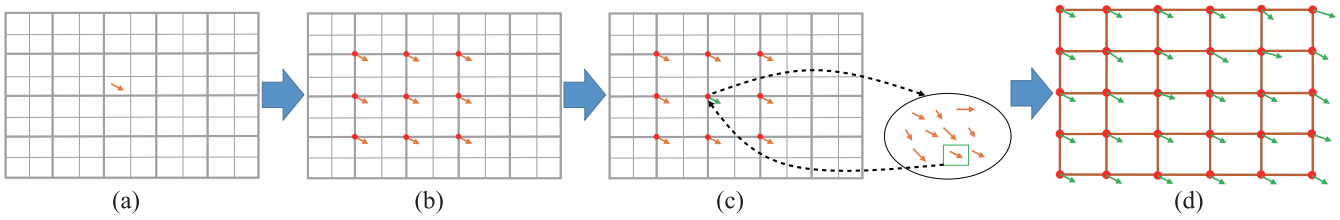


Fig. 7. The pipeline of CodingFlow estimation. A regular grid mesh is placed onto the frame. A motion vector of a block (a) is assigned to the nearby mesh vertices (b). As a result, a mesh vertex will receive multiple motion vectors coming from its nearby blocks. (c) For each vertex, all collected motions are sent to a median filter. The filtered result is assigned back to the vertex. (d) Repeating the same procedure for all vertices yields the final CodingFlow.

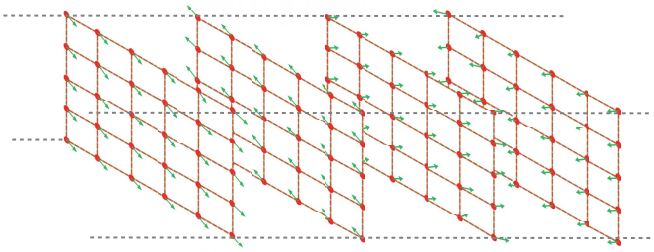


Fig. 8. CodingFlow and pixel profiles extracted at mesh vertex positions.

four image corner. Profiles of the inner vertices are not shown in the example. Camera paths are defined for each profile as:

$$c_i(t) = f_i(t) + f_i(t-1) \ldots + f_i(0),\ f_i(0) = 0 \qquad (5)$$

where $f_i(t)$ is the motion of i-th vertex at the t-th frame. The camera path $c_i(t)$ is the concatenated additions of all adjacent motions. The path at i-th profile $\{c_i(t)\}$ is optimized as:

$$\mathcal{O}(\{p_i(t)\}) = \sum_t \|p_i(t) - c_i(t)\|^2 + \sum_t (\lambda_t \sum_{r \in \Omega_t} \omega_{t,r} \|p_i(t) - p_i(r)\|^2) \qquad (6)$$

where $\{p_i(t)\}$ is the optimized profile of vertex position i. Optimizing profiles over all vertices $\sum_i \mathcal{O}(\{p_i(t)\})$ yields the optimized CodingFlows. The updating motion vector $b_i(t)$ for each vertex is calculated as: $b_i(t) = p_i(t) - c_i(t)$, applying which to all vertices yields new meshes at each frame. The stabilized video can be obtained by image warping guided by mesh grids.

Here, all profiles are optimized independently. Therefore, they can be processed in parallel for an improved efficiency. In the bundled paths approach [5], the local camera paths are optimized with a spatial temporal optimization, where similarity constraints between neighboring local paths are enforced. It is easy to understand that if the estimated flow is smooth spatially, the neighboring local profiles are quite similar. As such, it becomes unnecessary to enforce additional spatial similarities. More discussions will be given in the next section.

All profiles share the same λ_t for each frame. Similarly, the iterative refinement approach is adopted. Previously, the homography-based camera path is used for the evaluation of image crops. Here, four pixel profiles at four corners are used for evaluations. Specifically, after the optimization of Eq. (6), we take the profiles at four corners and calculate their updating vectors $b_i(t)$. At each frame, a new rectangle is formed after the transformation of four corners, which allows us to approximate the crops. If the crops are not satisfied, the corresponding λ_t is relaxed.

VI. Discussions

A. Interference of Intermediate I-Frames

The first frame is an I-frame and will always be encoded with the intra mode. However, other intermediate I-frames would cause some troubles because each I-frame is isolated from its previous frames so that no motion vectors can be extracted from the coded bitstream. In this case, we go back to employ the traditional method, i.e., to detect image features between an intermediate I-frame and its previous adjacent frame. Then, its CodingFlow is estimated by motions that are induced by a global homography.

B. Interference of Intra Mode

In video coding, some blocks of a P or B frame may choose the intra coding mode, i.e., no motion compensation is needed, which means that no motion vector is available. The corresponding blocks are referred to as intra blocks. Because of these intra blocks, some vertices do not receive any motion vectors. In this case, the global motion will be assigned. If the majority of blocks in a frame (e.g., over 50%) are intra blocks, the CodingFlow estimation would become inaccurate. Then, this frame will be treated as an intermediate I-frame. From our experiments, the percentage of intra blocks of one frame is normally within 10%.

C. Flow Comparisions

We compare our CodingFlow with some existing methods to validate its effectiveness. We show two examples in the Fig. 9: the first example is dynamic and the second example is static. We show the Bundled-paths [5] (Fig. 9(b)), the SteadyFlow [6] (Fig. 9(c)), the MeshFlow [35] (Fig. 9(d)), and our CodingFlow (Fig. 9(e)), where the Bundled-paths, the MeshFlow, and the CodingFlow have been interpolated into dense motion fields for visual comparisons. We also show the warping errors in terms of SSD, labeled on the top-left of each flow. Directly smoothing raw optical flow can cause distortions, especially at the discontinuous motion boundaries. The video stabilization methods prefer a spatially-smooth flow field instead of flow fields with rich motion

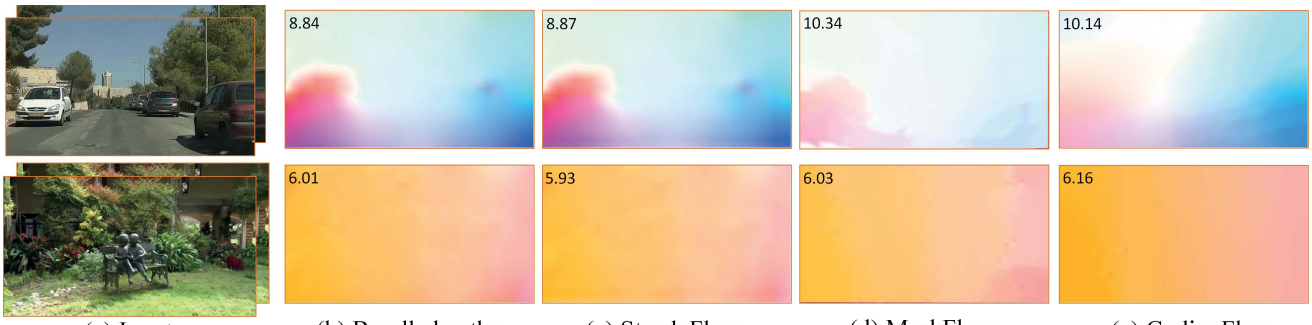


Fig. 9. Comparison of motion fields. (a) are two consecutive frames. (b) the interpolated dense motion field produced by the Bundled-paths [5]. (c) the SteadyFlow [6]. (d) the interpolated dense motion field by MeshFlow [35]. (e) our interpolated CodingFlow for the visual comparisons. The numbers on the top-left show the warping errors.

TABLE I
THE PROCESSING SPEED (PER FRAME) OF VARIOUS METHODS

	Epipolar [11]	Bundled [5]	Steady [6]	Mesh [35]	Ours
Speed	950ms	392ms	1500ms	20ms	18ms

details and motion edges. A detailed discussion regarding the motion field for stabilization can be found in [6]. As can be seen, our CodingFlow is quite similar when compared with other approaches. Similar results have also been obtained nearly in all our experiments. While our approach produces motion fields whose spatial variations have been smoothed to a certain extent, it neither relies on image features nor builds upon expensive dense optical flows, thus facilitating the implementation by various levels of portable hardware.

VII. EXPERIMENTS

We run our method on a labtop with an Intel i7 2.5GHz CPU and 16G RAM. We record the speed of the components in stabilization. For a frame with resolution 720×480 and mesh resolution 16×16, our un-optimized C++ implementation can process a frame in around 18ms. Specifically, we spend 3ms, 2ms, 2ms, 8ms, and 3ms for the motion manipulation, global motion correction, coding flow estimation, smoothing, and frame rendering, respectively.

Table I shows the speed of some other approaches for a comparison. The bottle neck of the Epipolar and Bundled approaches is the tracking of image features. The estimation of a dense optical flow required by SteadyFlow is also very expensive computationally. On the other hand, MeshFlow and our approach can achieve a similar speed, which is much faster. While MeshFlow implements the feature tracking very efficiently on a PC, many functions/processing involved in MeshFlow are not easy to be transferred into other platforms (such as embedded ones) to maintain the same quality and efficiency. On the contrary, video decoding has become compulsory in nearly all video-related applications. So far, a lot more efforts have been put on implementing various functions/processing of video decoding on different platforms, which can thus be utilized directly in our CodingFlow. Finally, note that we do not count the run time of video decoding in Table I as all approaches compared here require the same decoding if the input video is given in a coded format.

TABLE II
THE AVERAGED NUMBER OF JUMPS DURING MOTION ACCUMULATIONS

	#1	#2	#3	#4	#5	#6	#7
Number jumps	2.02	2.21	2.20	1.77	2.65	2.19	1.94
	#8	#9	#10	#11	#12	#13	#14
Number jumps	2.03	2.01	1.93	2.14	1.98	2.42	1.96
	#15	#16	#17	#18	#19	#20	#21
Number jumps	1.99	2.00	2.21	2.07	1.83	2.08	1.94
	#22	#23	#24	#25	#26	#27	#28
Number jumps	1.96	1.99	2.08	1.93	1.87	1.88	1.92

We perform various experiments to demonstrate the effectiveness and strength of our method. The video examples (totally 28 video clips) are collected from public datasets that are gathered from [5]–[7], [11]. Figure 10 shows the thumbnails of these video clips. We conduct two types of experiments to evaluate the intrinsic properties of our CodingFlow, following which the quality of stabilization is evaluated in terms of some objective metrics.

A. Number of Motion Accumulations

Our motion manipulation presented earlier requires to accumulate inter motions, including motion additions and subtractions, such that all motions of blocks in one frame only refer to its adjacent previous frame. It is worth to note that the number of accumulations is critical. A large number may introduce some drifting errors, leading to the inaccuracy. Without any manipulations, the number is 1. One addition or subtraction will increase the number by 1. Table II summarizes the averaged numbers of all blocks in all frames of each example. As can be seen, the number is quite small, indicating that the encoder tends to search adjacent neighboring frames for inter motion prediction, which is a very nice property for our motion manipulation.

B. Valid Inter Motions

An inter motion is considered as a valid motion as long as it passes the global motion correction. In other words, if a motion vector contributes itself to the estimation of the Codingflow, it is treated as a valid motion, thought it might be discarded by the median filter. The global motion correction rejects errors

Fig. 10. Video thumbnails. Please refer to the project page for the videos and stabilized results. http://www.liushuaicheng.org/TIP/CodingFlow/index.html.

TABLE III

THE PERCENTAGE OF VALID INTER MOTION OF THE 28 EXAMPLES

	#1	#2	#3	#4	#5	#6	#7
Valid motions	73%	76%	71%	83%	75%	75%	63%
	#8	#9	#10	#11	#12	#13	#14
Valid motions	62%	74%	82%	68%	79%	85%	66%
	#15	#16	#17	#18	#19	#20	#21
Valid motions	66%	77%	79%	74%	67%	81%	85%
	#22	#23	#24	#25	#26	#27	#28
Valid motions	65%	76%	78%	63%	76%	79%	74%

that are totally inconsistent with the camera motion while the median filters smooth out noises for an improved accuracy.

We summarize the percentage of valid inter motions of the 28 examples in Table III. In each example, we record the validity of each frame and take the average of all frames for the final values. The validity varies from 60% to 85%, with the lowest value 62% at Example 8 and the highest value 85% at Examples 13 and 21. Note that Example 8 contains a large portion of stairs - the repetitive structure that confuses the block-matching algorithm; whereas Examples 13 and 21 are full of differentiable textures that facilitate the block-matching. In contrast, the lack of rich textures can lead to a low percentage, such as Examples 22 (ice ground) and 7 (blue sky). In Example 22, apart from poor textures, the background that moves dramatically further decreases the motion validity. Note that thumbnails in Fig. 10 may not retain sufficient information of the video. Please refer to the project page for these videos with a more thorough impression.

C. Warping Errors

We evaluate the quality of the motion estimation in terms of warping errors. Here, we warp the frame at t towards the frame at $t-1$ according to the CodingFlow of the t-th frame. The averaged SSD errors are calculated between two frames. We skip the frame boundaries to exclude any black regions. We average all SSD errors from all frames for each example. Table IV summarizes the averaged SSD errors. We also

TABLE IV

THE AVERAGED WARPING SSD ERRORS OF THE 28 EXAMPLES BY BUNDLED-PATHS [5], STEADYFLOW [6], MESHFLOW [35], AND OUR METHOD

	#1	#2	#3	#4	#5	#6	#7
Bundled-paths [5]	5.60	2.72	2.57	3.64	4.25	2.75	8.00
SteadyFlow [6]	5.41	2.39	2.41	3.18	4.18	2.33	7.06
MeshFlow [35]	6.09	2.68	2.62	3.76	4.27	2.55	8.30
Ours	6.00	2.76	2.77	3.43	4.65	2.67	9.39
	#8	#9	#10	#11	#12	#13	#14
Bundled-paths [5]	5.53	3.32	2.52	1.97	1.29	4.21	7.99
SteadyFlow [6]	5.15	3.16	2.27	1.87	1.30	4.08	7.64
MeshFlow [35]	5.41	3.61	2.37	1.98	1.28	4.25	8.62
Ours	5.40	3.78	2.55	2.44	1.72	4.63	9.50
	#15	#16	#17	#18	#19	#20	#21
Bundled-paths [5]	9.17	10.01	6.35	5.38	12.07	6.62	11.21
SteadyFlow [6]	9.16	9.45	5.86	5.35	11.62	6.49	10.92
MeshFlow [35]	9.05	11.10	6.13	5.54	13.46	6.68	12.06
Ours	9.64	10.23	6.18	6.16	12.52	6.75	11.33
	#22	#23	#24	#25	#26	#27	#28
Bundled-paths [5]	3.87	4.67	5.52	2.62	3.01	2.49	6.99
SteadyFlow [6]	4.10	4.57	5.09	2.35	2.83	2.21	6.91
MeshFlow [35]	5.38	4.77	5.61	2.90	4.04	2.47	7.16
Ours	4.54	4.55	5.47	2.83	3.17	2.41	8.03

compute the warping errors of Bundled-paths [5], SteadyFlow [6], and MeshFlow [35] for comparisons. As shown by Table IV, our performance is comparable with these existing approaches. The performance of SteadyFlow is slightly better than the others. The Bundled-paths approach adopts SURF features for the as-similar-as-possible mesh warps, while SteadyFlow calculates dense optical flow as the motion sources and MeshFlow tracks image features. Our CodingFlow is light-weight, neither requires external motion calculations nor minimizes the least squares for mesh rigidity regularization, yet produces comparable results in terms of the registration quality.

D. Objective Evaluations

To evaluate the quality of stabilization, we adopt three objective metrics: *cropping ratio*, *distortion*, and *stability*, which are proposed in [5]. For a good result, these metrics

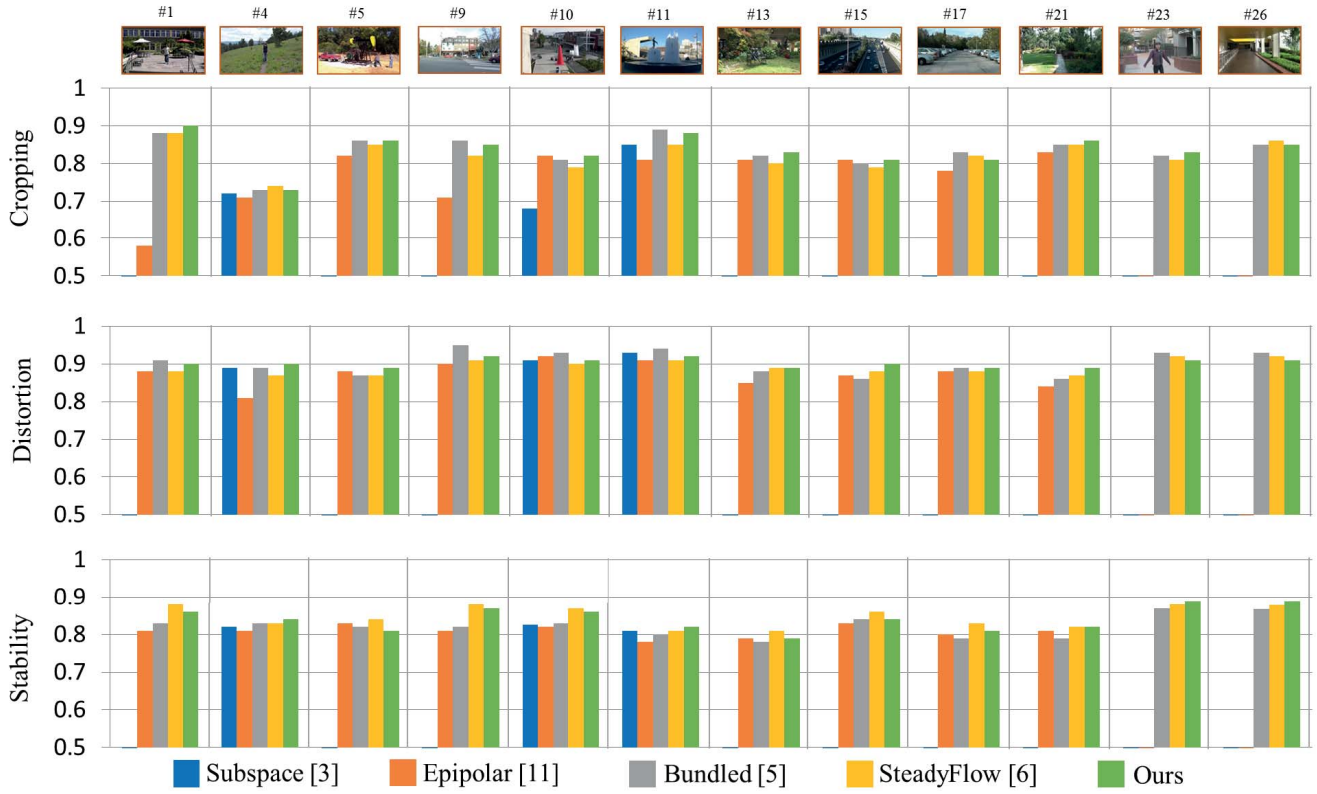


Fig. 11. Quantitative evaluations of 12 examples using objective metrics.

should be close to 1. For the sake of completeness, we first describe these metrics briefly in the following.

1) Cropping Ratio: Evaluates the effective area after cropping away black boundaries. A larger ratio means less cropping. It is estimated by finding a homography between every input and output frames. The cropping ratio of each frame is estimated by the scale component of the homography. The smallest ratio is adopted as the final cropping ratio.

2) Distortion Score: Evaluates the wobble distortion of stabilized frames. Similarly, a homography is estimated between the input and output frames. Then, the distortion score is estimated from the anisotropic scaling of the homography. In particular, it is computed by the ratio of the two largest eigenvalues of the affine part from the homography. Similarly, the worst score is adopted as the distortion score.

3) Stability Score: Evaluates the smoothness of the stabilized video. The vertex profiles are extracted from the stabilized video for the evaluation. Specifically, we analyze each vertex profile in the frequency domain. We take a few of the lowest frequencies (2nd to 6th) and calculate the energy percentage over full frequencies (without DC component). The final score is obtained by averaging from all profiles.

We choose 12 out of all 28 examples (Fig. 10) and compare our method with the methods of [3], [5], [6], and [11]. The results of other approaches are either collected from their project pages or produced by ourselves. For videos that do not provide the results, we leave them blank. As illustrated by Fig. 11, our method can produce comparable, sometimes superior, results when compared with these state-of-the-art stabilization methods. Most importantly, except for our method, all other methods relay on image features for motion estimation. On the other hand, our method has effectively utilized the involved video coding that automatically provides necessary motion vectors. Therefore, it has avoided the time-consuming process for feature detecting, matching, and tracking nearly completely for all frames (except intermediate I-frames and some bad P or B frames).

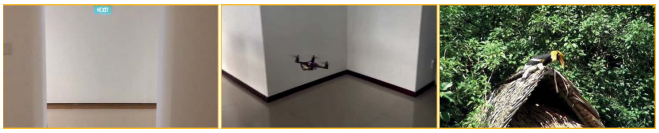


Fig. 12. Some failure cases. (Lack of the valid inter motions).

E. Failure Cases

When the number of inter motions corresponding to the camera motion is insufficient, our system may fail. In general, the inter motion validity percentage needs to stay above 50% for a successful stabilization. Though a very low validity happens rarely, it does exist in some extreme cases, such as filming a purely white wall or shaking dramatically. Fig. 12 shows three failure cases. The first one was captured in a corridor with three white walls. The second one was shot at a flying drone with textureless backgrounds. The third one contains periods with sudden dramatic shakes. All these examples yield poor inter motions, stabilizing which would lead to wobble distortions (the first two cases) and unstable motions (the third case).

VIII. Conclusion

We have presented a video stabilization approach that utilizes the involved video coding for the recovery of camera motions. In our method, block-based motion vectors are extracted directly from the coded video bitstream, and then manipulated to refer only to its previous adjacent frames. A novel CodingFlow is estimated from the manipulated motion vectors such that a sparse motion field is obtained to describe spatially-variant motions for a high-quality video stabilization. Numerious experiments as well as quantitative evaluations are conducted, which shows the effectiveness of our proposed method with comparisons to several state-of-the-art methods.

References

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

[2] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 44.

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

[4] B.-Y. Chen, K.-Y. Lee, W.-T. Huan, and J.-S. Lin, "Capturing intention-based full-frame video stabilization," *Comput. Graph. Forum*, vol. 27, no. 7, pp. 1805–1814, Oct. 2008.

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

[6] S. Liu, L. Yuan, P. Tan, and J. Sun, "Steadyflow: Spatially smooth optical flow for video stabilization," in *Proc. CVPR*, 2014, pp. 4209–4216.

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

[8] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, "Calibration-free rolling shutter removal," in *Proc. ICCP*, 2012, pp. 1–8.

[9] Z. Zhou, H. Jin, and Y. Ma, "Plane-based content-preserving warps for video stabilization," in *Proc. CVPR*, 2013, pp. 2299–2306.

[10] J. Shi and C. Tomasi, "Good features to track," in *Proc. CVPR*, 1994, pp. 593–600.

[11] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM Trans. Graph.*, vol. 32, no. 5, 2012, Art. no. 126.

[12] S. Bell, A. Troccoli, and K. Pulli, "A non-linear filter for gyroscope-based video stabilization," in *Proc. ECCV*, 2014, pp. 294–308.

[13] A. Karpenko, D. E. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," Stanford Comput. Sci., Stanford, CA, USA, Tech. Rep. CSTR 2011-03, 2011.

[14] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[15] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[16] M. L. Gleicher and F. Liu, "Re-cinematography: Improving the camera dynamics of casual video," in *Proc. ACM Multimedia*, 2007, pp. 27–36.

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

[18] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[19] B. Zeng and J. J. Fu, "Directional discrete cosine transforms—A new framework for image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 305–313, Mar. 2008.

[20] SY. Zhu, S.-K. A. Yeung, and B. Zeng, "R-D performance upper bound of transform coding for 2-D directional sources," *IEEE Signal Process. Lett.*, vol. 16, no. 10, pp. 861–864, Oct. 2009.

[21] S. Y. Zhu, S.-K. A. Yeung, and B. Zeng, "In search of 'better-than-DCT' unitary transforms for encoding of residual signals," *IEEE Signal Process. Lett.*, vol. 17, no. 11, pp. 961–964, Nov. 2010.

[22] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer DCT architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 168–178, Jan. 2014.

[23] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[24] Y. Taki, M. Hatori, and S. Tanaka, "Interframe coding that follows the motion," in *Proc. Inst. Electron. Commun. Eng. Jpn. Annu. Conv. (IECEJ)*, 1974, p. 1263.

[25] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799–1808, Dec. 1981.

[26] T. Koga, K. Linuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. NTC*, 1981, pp. C9.6.1–C9.6.5.

[27] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[28] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[29] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[30] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *Proc. CVPR*, 2012, pp. 89–95.

[31] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala, "Light field video stabilization," in *Proc. ICCV*, 2009, pp. 341–348.

[32] Z.-Q. Wang, L. Zhang, and H. Huang, "Multiplane video stabilization," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 265–273, Sep. 2013.

[33] C. Jia and B. L. Evans, "Constrained 3D rotation smoothing via global manifold regression for video stabilization," *IEEE Trans. Signal Process.*, vol. 62, no. 13, pp. 3293–3304, Jul. 2014.

[34] J. Bai, A. Agarwala, M. Agrawala, and R. Ramamoorthi, "User-assisted video stabilization," *Comput. Graph. Forum*, vol. 33, no. 4, pp. 61–70, 2014.

[35] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, "Meshflow: Minimum latency online video stabilization," in *Proc. ECCV*, 2016, pp. 800–815.

[36] N. Jiang, P. Tan, and L.-F. Cheong, "Multi-view repetitive structure detection," in *Proc. ICCV*, 2011, pp. 535–542.

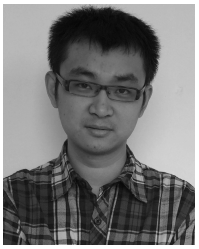
[37] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. ICCV*, 2009, pp. 349–356.

[38] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.

[39] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge Univ. Press, 2003.

[40] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *Proc. CVPR*, 2010, pp. 2392–2399.

[41] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.



Shuaicheng Liu (M'15) received the B.E. degree from Sichuan University, Chengdu, China, in 2008, and the M.S. and Ph.D. degrees from the National University of Singapore, Singapore, in 2010 and 2014, respectively. In 2014, he joined the University of Electronic Science and Technology of China and is currently an Associate Professor with the Institute of Image Processing, School of Electronic Engineering. His current research interests include computer vision and computer graphics.

Mingyu Li received the B.Eng. degree from Zhengzhou University, Zhengzhou, China, in 2013. He is currently pursuing the master's degree with the Institute of Image Processing, School of Electronic Engineering, University of Electronic Science and Technology of China. His current research interests include image/video compression and image processing.

Shuyuan Zhu received the Ph.D. degree from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2010. From 2010 to 2012, he was with HKUST and Hong Kong Applied Science and Technology Research Institute Company Ltd., respectively. In 2013, he joined the University of Electronic Science and Technology of China and is currently an Associate Professor with the School of Electronic Engineering. His research interests include image/video compression, image processing, and compressive sensing. He is a member of the IEEE CAS Society. He has over 40 research publications and received the Top 10% Paper Award at the IEEE ICIP 2014. He was the special Session Chair of image super-resolution at the IEEE DSP 2015. He served as the Committee Member of the IEEE ICME 2014, and serves as the Committee Member of the IEEE VCIP 2016.

Bing Zeng (M'91–SM'13–F'16) received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983 and 1986, respectively, and the Ph.D. degree in electrical engineering from the Tampere University of Technology, Tampere, Finland, in 1991.

He was a Post-Doctoral Fellow with the University of Toronto from 1991 to 1992 and as a Researcher with Concordia University from 1992 to 1993. He then joined The Hong Kong University of Science and Technology (HKUST). After 20 years of service at HKUST, he returned to UESTC in 2013, through China's 1000-Talent-Scheme. At UESTC, he leads the Institute of Image Processing to focus on image and video processing, 3D and multi-view video technology, and visual big data. During his tenure at HKUST and UESTC, he has graduated over 30 master and Ph.D. students, received about 20 research grants, filed eight international patents, and authored or co-authored over 250 papers.

Three representing works are as follows: one paper on fast block motion estimation, published in the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT) in 1994, has so far been SCI-cited over 1000 times (Google-cited over 2100 times) and currently stands at the seventh position among all papers published in this Transactions; one paper on smart padding for arbitrarily-shaped image blocks, published in the IEEE TCSVT in 2001, leads to a patent that has been successfully licensed to companies; and one paper on directional discrete cosine transform, published in IEEE TCSVT in 2008, receives the 2011 IEEE CSVT Transactions Best Paper Award. He also received the best paper award at China-Com three times (2009 Xi'an, 2010 Beijing, and 2012 Kunming).

He served as an Associate Editor of the IEEE TCSVT for eight years and received the Best Associate Editor Award in 2011. He was General Co-Chair of the IEEE VCIP-2016, Chengdu, in 2016. He is currently on the Editorial Board of the *Journal of Visual Communication and Image Representation* and serves as General Co-Chair of PCM-2017. He received the Second Class Natural Science Award (the first recipient) from the Chinese Ministry of Education in 2014 and was elected as an IEEE Fellow in 2016 for contributions to image and video coding.