Real-Time Low-Complexity Digital Video Stabilization in the Compressed Domain

Vasileios Avramelos (*Author*), Glenn Van Wallendael and Peter Lambert
Department of Electronics and Information Systems
Ghent University - imec - IDLab
Technologiepark - Zwijnaarde 19, 9052 Ghent, Belgium
vasileios.avramelos[at]ugent.be

***Abstract*—Undesired vibration in videos is more and more common due to the rise of hand-held or vehicle mounted cameras. Non-indented motion of a video capturing device causes an unpleasant vibrating effect for the consumer. Therefore, video stabilization has been an extremely active field of research in the past years. Motion estimation is the most computationally expensive step of the video stabilization process. Our goal is to circumvent this expensive step in order to achieve real-time performance. We do so by using the precomputed motion vectors from the encoded video streams and thus operate in the compressed domain. These vectors already contain an approximation of the needed motion information. A low-level motion model is used for mitigating complexity, and a low-pass filter performs motion smoothing before the final motion compensation step which is used for correcting the video. In many real-time applications where the video vibration is moderate, the proposed framework can reach online video stabilization at 30 frames per second for high definition video and 60 frames per second for lower resolutions, while retaining satisfactory performance in video correction, comparable to pixel-based equivalent algorithms.**

***Index Terms*—Digital video stabilization, compressed domain processing, motion vectors, real-time systems, low-complexity algorithms.**

I. Introduction

Cameras keep getting smaller and lighter and video capturing has become easier than ever. Some examples are camcorders, mobile phone cameras, cameras on unmanned aerial vehicles or mobile robots, and front car cameras. Using live video over the network instead of being physically present has broaden the opportunities for remote operations. In many real life scenarios corrupted video is transmitted over the network and video correction is necessary (or desired) at real-time either for entertaining reasons e.g., live streaming from a sports action camera, or for more crucial reasons such as remote surgery or remote heavy machinery operation. For discarding unindented camera movement from videos in consumer electronics end products, many video stabilization approaches have been proposed and commercialized throughout the last years. Solutions reported in literature are either hardware or software solutions. Hardware-based stabilizers such as motorized gimbals require the acquisition of additional equipment which is often designed for a certain camera type, e.g., hand-held cameras. Software solutions make use of computer algorithms to align misaligned frames due to camera vibrations in real-time or on prerecorded video.

The three major steps within a digital video stabilization pipeline is motion estimation, motion smoothing and motion compensation, with motion estimation being the most computationally expensive step. Digital solutions perform stabilization mainly in the pixel domain by using techniques such as feature matching [1], computer vision optical flow fields [2], edge pattern matching [3], SIFT point matching [4], [5], etc. Less research has been performed on video correction in the compressed domain [6], [7]. In the compressed domain, the encoded motion information is basically block-based motion vectors which are a result of an optimization of block size costs, motion vector signaling costs, residual signaling costs, etc. Therefore, even if the intention of the motion vectors is to describe the motion in a scene, what they actually do is minimizing the prediction error for maximizing the compression efficiency [8]. However, the motion information in those motion vectors is adequate enough to describe decently the movement in a scene. As such, during video stabilization the motion estimation step can be avoided and the whole correction process can be simplified.

Conventionally, for estimating the global camera movement in order to correct unwanted shakiness, motion models are utilized for representing the 3D motion of the background in a scene as it is being projected in two dimensions [9]. The higher the order of the chosen motion model, the more different types of motion can be controlled in a scene, i.e., translation, rotation, sheering, scaling, etc. However, depending on the application, the motion model can be of a lower order. For instance, in applications where the camera focuses mainly in one direction for multiple frames (constant background), correcting translation and/or rotation is adequate for yielding a stable video sequence.

The target application in this work is video stabilization under real-time constraints. Video correction is mainly performed on prerecorded video due to the fact that it is an expensive process to be operated in real-time, i.e, online video stabilization. However, a number of research efforts presented

The work in this paper was performed as part of the imec *HELP-Video!* (Hardware platforms for Embedded, Efficient, Low latency, and Portable video Processing) national industrial research project.

real-time video stabilization [10]–[15]. All of these efforts are pixel-based and many of them are of high-complexity with numerous preprocessing steps which are necessary in order to mitigate the expensive step of motion estimation. As long as the stabilization performance remains acceptable, the algorithm's complexity can be lowered in order to minimize the overall computational time. That is the main interest of experimentation in this work. Assume a scenario where corrupted/shaky video is captured, encoded into a bitstream, and transmitted through a network to be decoded and visualized. Examples are remote surgery, remote machinery operation, network-based endoscopy, live streaming, etc. In that case, conventionally the video is being corrected right after the decoding step and before the actual playback. Then, a pixel-based computer vision approach is typically used to correct the video as a post-processing step. In our proposed method, we use the encoded motion vectors readily exist in the bitstream and we perform the correction during decoding and without the need of any costly preprocessing step. According to our knowledge, there is no other work on real-time video stabilization without the need of additional expensive preprocessing steps.

In this paper, we propose and evaluate a low-complexity video stabilization algorithm, which is performed in the compressed domain by only using the readily available motion vectors. After extracting the motion information during decoding, the global motion vector is calculated by performing a small number of non-complex preprocessing actions on the motion vector flow field. Finally, a motion compensation step for the correction of the translational misalignment of subsequent frames is following a motion smoothing technique using a low-pass filter in the frequency domain. Experiments showed that the algorithm is very robust for moderate vibration and constant background video correction scenarios such as front car camera view, video surveillance, endoscopy, etc., while it operates in real-time during video decoding.

The rest of the paper is organized as follows. In Section 2 the proposed method is presented, in Section 3 the experimental results are being discussed and finally, in Section 4 we conclude this work, discuss potential applications, and we recommend improvements which are planned as future work.

II. Proposed method

Video stabilization consists of global motion estimation, motion smoothing, and motion compensation. During the global motion estimation local (object movement) and global (camera movement) motion vectors are separated for each frame, and a global motion vector is calculated, which describes the undesired camera movement or camera trajectory. Filtering high frequencies out of the camera trajectory signal, is the second step of the video stabilization process, while motion compensation is the step where misaligned frames are correctly aligned accordingly.

As specified in [17]–[20], perspective or affine transformation is not always the best suitable model for stabilizing video. Considering our targeted application (real-time) and the nature of the videos we focus (no panning present), we verified that non-linear similarity transformation is the best suited motion model for sufficiently correcting videos while remaining within real-time constraints.

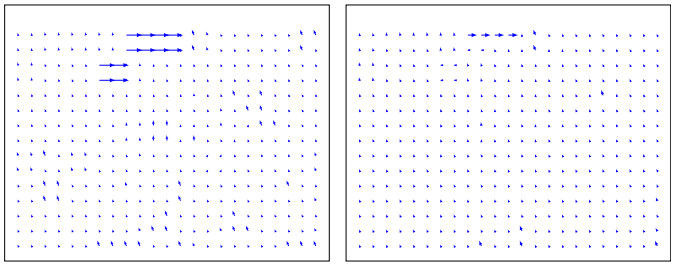

Fig. 1. Example of motion vector field before (left) and after (right) outlier removal using the Hampel identifier [16]. On the right side it can be seen the smoothened motion vector flow field which makes the camera movement approximation easier by mainly discarding motion vectors corresponding to moving objects.

We envision real-time video stabilization at the decoding level reaching at least 30 corrected/stabilized frames per second (fps). To achieve this during decoding, we let a number of frames, e.g., 25 frames, to be decoded and visualized without correction and the subsequent frames are being corrected by taking under consideration only prior frames. As such, the stabilized video is displayed to the end user with only less than one second corrupted video at the beginning of the playback.

A. Global motion estimation

For estimating the global motion of the camera in a scene, the motion from frame to frame, i.e., the pixel movement in the 2D space, is modeled. The higher the order of the motion model, the more complex motion can be represented. For example, an 8-parameter model such as the perspective projection motion model with four reference points, can accurately describe a 2D projection of a 3D rigid motion of a point to be projected [9]. In that case, motion such as translation, rotation, scaling and sheering can be extracted from the corresponding transformation matrix. Moving towards lower order models we observe that they are special cases of the perspective model. A 4-parameter model for instance, would correspond to a geometric transformation describing only translation, rotation and scale. However, in cases where panning is absent (e.g., when the camera is mounted), unwanted scaling is rare and most of the vibration to be corrected is due to image translation, i.e., pixel shifting by a number of pixels in either x- or y-direction, or both. The corresponding 2-parameter motion model, also known as non-reflective similarity transformation, can accurately estimate the translational distortions in images. The step-by-step global motion estimation used in this work, is described as follows.

In the first step, we rearrange the motion vector flow field acquired from the encoded bitstream in the 2D space for better visualization. Then, the motion vectors are divided into $m \times n$ blocks or grid cells. The smaller the block size the more accurate separation of local vs. global motion can be

reached and therefore better stabilization results. However, processing the flow field by using larger blocks can accelerate the block processing procedure (due to the sequential nature of the algorithm) which is useful for reducing computational time. Therefore, we mainly used blocks of sizes up to 64×64 depending on the video resolution and the desired frame rate we wanted to reach. The median of all x and y components of the flow field is calculated for every block. Outlier removal techniques were tested for removing vectors indicating too large translation. At the presence of camera shakiness, every decent outlier removal algorithm will mainly differentiate local movement and global movement. We tested a Hampel identifier or Hampel filter [16] to detect and remove outliers. For each x and y component of the motion vectors, the Hampel filter computes the median of a window composed of the actual vector and its six surrounding vectors, three per side. It also estimates the standard deviation of each vector about its window median using the Median Absolute Deviation (MAD). If a vector differs from the median by more than a given number of MADs, it is replaced with the median (see Fig. 1). More specifically, given a signal with samples $x_1, x_2, ..., x_n$ and a sliding window of length k, the median absolute deviation is calculated as in the following [16]:

$$\text{MAD} = \text{median}(|x_i - m_i|, ..., |x_n - m_n|), \tag{1}$$

with m_i being the local median

$$m_i = \text{median}(x_{i-k}, ..., x_i, ..., x_{i+k}). \tag{2}$$

If a sample x_i is such that

$$|x_i - m_i| > n_\sigma \, \text{MAD}, \tag{3}$$

for a given threshold n_σ (default is $n_\sigma = 3$), then the Hampel identifier declares x_i an outlier and replaces it with m_i.

Finally, from the resulted downscaled global motion vector field we calculate the global motion vector. For the remaining inlier x and y translational values, the median is calculated and it represents a single motion vector, namely the global motion vector for the corresponding frame.

B. Motion smoothing

For motion smoothing, we transform our global motion signal into the frequency domain by using a fast Fourier transform, where we apply a low-pass filter for removing high frequencies from the camera trajectory both in the x- and y-direction. The cut-off frequency is set to 1 Hz and only frequencies lower than that are not filtered. In that way high-frequency jitter is also being attenuated for avoiding so-called "jumps" that are frequently found in stabilized videos due to powerful vibrations.

C. Motion compensation

Motion compensation is being used for correcting the frames according to the smoothened curve (see Fig. 2). The corrected frame is aligned accordingly in the x and y direction:

$$\hat{GMV}_{Xi} = GMV_{Xi} - Diff_{Xi} \tag{4}$$

$$\hat{GMV}_{Yi} = GMV_{Yi} - Diff_{Yi} \tag{5}$$

where $\hat{GMV}$ and GMV is the corrected and actual global motion vector respectively, and $Diff$ is the difference between the raw camera trajectory and the smoothened camera trajectory in the x and y directions for the corresponding i^{th} frame.

Corrected frames are cropped according to the maximum difference between the actual and smoothened camera trajectory in both axes. In that way, the stabilized video is absent of black borders which are a result of moving frames due to translation correction. An adaptive cropping technique decides the size of the border to be cropped depending on the maximum global motion vector of a corresponding group of pictures.

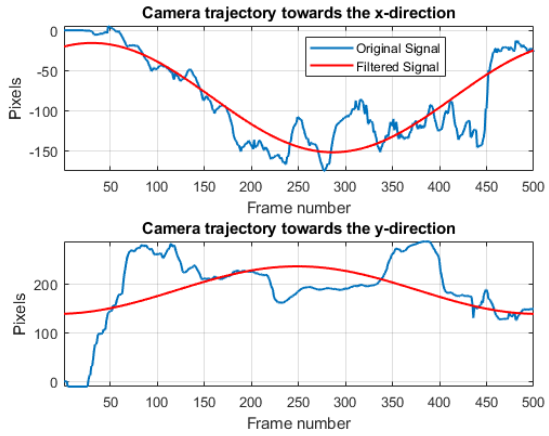

Fig. 2. Example of camera trajectories in the $x-$ and $y-$direction and the corresponding smoothened trajectories after reaching the end of the video. Smoothing filter used for this example is a fast Fourier transform filter with cutoff frequency of 1 Hz.

III. Experimental results

For the experimental part of this work we used three datasets. The fisrt video sequence is *shaky_car* found in [1] and in the MATLAB's Computer Vision Toolbox [21], [22]. It shows distorted material taken from a car front camera with resolution 320x240 at 30 fps. A second test sequence is a YouTube video [23] which suffers from heavy shakiness and it has been a test subject for video stabilization. It shows material from a mounted, but not stable camera filming a moving train. Framerate here is 25 fps with a resolution of 720x480 (SD). For this work we name the sequence *train*. Lastly, we used a more application specific endoscopic video which was explicitly recorded within the context of the *HELP Video!* project and it consists of a video crop of an endoscopic surgery which suffers from common endoscopic camera vibrations. Here, the video is captured at 30 fps with a resolution of 1920x1080 (HD).

The software we used was MATLAB on an Intel Core^{TM}i7-6700K CPU machine. No hardware acceleration techniques were used for a potential parallel implementation of our block-based function. All experiments presented here are serially conducted on the CPU. The motion vectors were extracted from High Efficiency Video Coding (HEVC) [8] bitstreams created by the HEVC reference coder HM-16.5 on a low-delay IPPP setting, which is suitable for real-time applications [24].

Fig. 3. Example of video stabilization performance of the proposed method for the sequence *train* [23]. We compare the mean of the raw (left) vs. the corrected (right) frames for random groups of ca. 10 luminance frames.

We evaluated the performance of the video stabilizer subjectively (see Fig. 3) and compared to a pixel-based approach of higher complexity (see Fig. 4). As it can be seen in Fig. 3 and 4, we calculated the mean of consecutive frames for visualizing the actual correction on the stabilized video in comparison to the unstabilized one, similarly to [1]. We implemented an example of a feature-matching approach for video stabilization in the pixel domain, which uses a higher order affine transformation [1], [22] for correcting scaling, rotation and translation. From Fig. 3, 4 and 5 it can be seen that in cases with moderate vibration and relatively constant background, vibration can be corrected using the proposed low-complexity method with similar results to higher complexity and more sophisticated approaches that operate exclusively in the pixel-domain. After the subjective evaluation we also assessed the stabilizer objectively. Table I also shows a comparison between the proposed method (with and without the use of the Hampel identifier) and the pixel-based method in terms of Peak Signal to Noise Ratio (PSNR) between consecutive frames, which is a common way for measuring video stabilization performance.

In terms of speed, we calculated the time needed to apply the whole stabilization process per frame, and we present the results in Table II. For block sizes of 16x16, 32x32, and 64x64 we measured the corrected fps for different video resolutions. Results showed that in every case we exceed with ease the limit of 60 fps for smaller resolutions without significant code optimization or hardware acceleration, while for HD video, 30 fps can be reached with the use of blocks of size 64x64 or larger. Note here, that the higher the resolution the less we suffer from stabilization accuracy decrease due to larger block sizes.

Fig. 4. Example of video stabilization performance of the proposed method (top) vs. a pixel-based method (bottom) [1], [22]. We compare the mean of the raw (left) vs. the corrected (right) frames for a random group of ca. 10 luminance frames.

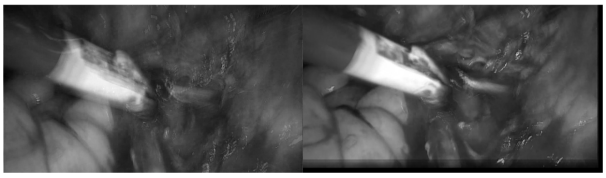

Fig. 5. Example of video stabilization performance of the proposed method on endoscopic video. We compare the mean of the raw (left) vs. the corrected (right) frames for a random group of ca. 10 luminance frames.

	Raw	**Pixel-based**	**Proposed**	
Sequence	-	-	**w/ Hampel filter**	**w/o Hampel filter**
shaky_car	22.01 dB	29.61 dB	29.54 dB	29.41 dB
train	23.95 dB	30.94 dB	31.99 dB	31.82 dB
endoscopic	20.15 dB	27.06 dB	26.77 dB	26.32 dB

TABLE I
MEAN PSNR BETWEEN CONSECUTIVE FRAMES FOR EVALUATING STABILIZATION PERFORMANCE.

IV. CONCLUSIONS

We proposed a low-complexity video stabilization method working exclusively in the compressed domain and aiming

	Block sizes		
Sequence	**16x16**	**32x32**	**64x64**
shaky_car (320x240)	149.78 fps	195.90 fps	215.18 fps
train (720x480)	61.30 fps	106.08 fps	130.24 fps
endoscopic (1920x1080)	13.06 fps	26.57 fps	34.47 fps

TABLE II
TIMINGS OF THE PROPOSED VIDEO STABILIZATION METHOD IN TERMS OF FRAMES PER SECOND (FPS) FOR DIFFERENT BLOCK SIZES.

real-time applications (real-time video correction at the decoder's side) at cases where video is transmitted over the network and only input is the encoded bitstream. The key contribution is the simplicity of the algorithm, which allows implementation of video correction under real-time constraints and with low-computational complexity, during the decoding process. Results showed that satisfactory results in terms of video stabilization can be reached with the proposed method, while in terms of speed, framerates of 60 and 30 fps can be easily reached for SD and HD video respectively without any hardware acceleration. For mounted cameras with relatively constant background and moderate to intense vibration, in the presence of crucial real-time constraints, the proposed method is an excellent solution for video correction applications.

Future work would be a parallel implementation for the proposed method due to the fact that the algorithm is block-based without any dependencies between blocks. Therefore, an attractive speedup factor is expected from a straight-forward GPU acceleration of the algorithm. Additionally, the algorithm can be expanded with the option to use more sophisticated motion models for recognizing more complicated motion present in video and being able to correct more types of shaky video, by only increasing the complexity to an acceptable range.

REFERENCES

[1] L. M. Abdullah, N. Md Tahir, and M. Samad, "Video stabilization based on point feature matching technique," in *IEEE Control and System Graduate Research Colloquium*, July 2012, pp. 303–307.

[2] S. Liu, L. Yuan, P. Tan, and J. Sun, "SteadyFlow: Spatially Smooth Optical Flow for Video Stabilization," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 4209–4216.

[3] C. Tang, X. Yang, L. Chen, and G. Zhai, "A fast video stabilization algorithm based on block matching and edge completion," in *IEEE 13th International Workshop on Multimedia Signal Processing*, October 2011, pp. 1–5.

[4] Y. H. Chen, H. Y. S. Lin, and C. W. Su, "Full-Frame Video Stabilization via SIFT Feature Matching," in *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, August 2014, pp. 361–364.

[5] M. Fang and H. Li, "A video stabilization algorithm based on affine sift," in *IEEE International Conference on Computing, Mathematics and Engineering Technologies*. IEEE, April 2018.

[6] M. Okade and P. Biswas, "Fast Video Stabilization in the Compressed Domain," in *IEEE International Conference on Multimedia and Expo*, July 2012, pp. 1015–1020.

[7] M. Okade, G. Patel, and P. K. Biswas, "Robust Learning-Based Camera Motion Characterization Scheme With Applications to Video Stabilization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 3, pp. 453–466, 2016.

[8] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, December 2012.

[9] Y. Su, M. Sun, and V. Hsu, "Global motion estimation from coarsely sampled motion vector field and the applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 2, pp. 232–242, 2005.

[10] H. C. Chang, S. H. Lai, and K. R. Lu, "A robust real-time video stabilization algorithm," *Journal of Visual Communication and Image Representation*, vol. 17, no. 3, pp. 659–673, 2006.

[11] C. H. Chen, Y. L. Kuo, T. Y. Chen, and J. R. Chen, "Real-Time Video Stabilization Based on Motion Compensation," in *Fourth International Conference on Innovative Computing, Information and Control*, December 2009, pp. 1495–1498.

[12] A. J. Amiri and H. Moradi, "Real-time video stabilization and mosaicking for monitoring and surveillance," *4th International Conference on Robotics and Mechatronics*, no. 2, pp. 613–618, 2016.

[13] J. Dong and H. Liu, "Video Stabilization for Strict Real-Time Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 716–724, April 2017.

[14] B. Sultan, J. Ahmed, A. Jalil, H. Nazir, A. Ali, and H. Ali, "Translation and Rotation Invariant Video Stabilization for Real Time Applications," in *IEEE International Conference on Signal and Image Processing Applications*, December 2017, pp. 479–484.

[15] Z. Wang, L. Zhang, and H. Huang, "High quality real-time video stabilization using trajectory smoothing and mesh-based warping," *IEEE Access (Early Access)*, 2018.

[16] H. Liu, S. Shah, and W. Jiang, "On-line outlier detection and data cleaning," *Computers & Chemical Engineering*, vol. 28, no. 9, pp. 1635–1647, August 2004.

[17] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2011, pp. 225–232.

[18] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," in *ACM Trans. Graph.*, vol. 28, 2009.

[19] T. Gross, N. Amin, M. C. Offiah, S. Rosenthal, N. El-sourani, and M. Borschbach, "Optimization of Endoscopic Video Stabilization by Local Motion Exclusion," in *International Conference on Computer Vision Theory and Applications*, January 2014.

[20] N. Amin, T. Gross, M. C. Offiah, S. Rosenthal, N. El-Sourani, and M. Borschbach, "Stabilization of endoscopic videos using camera path from global motion vectors," in *International Conference on Computer Vision Theory and Applications*, October 2014, pp. 130–137.

[21] MATLAB and M. C. V. S. Toolbox, *version 9.3.0.713579 (R2017b)*. Natick, Massachusetts: The MathWorks Inc., 2017.

[22] Mathworks, *Computer Vision System Toolbox: User's Guide (R2017b)*. Natick, Massachusetts: The MathWorks Inc., 2017.

[23] I. Dmitriyevitch, "Extremely shaky video, how to stabilize it," May 2013, retrieved from https://www.youtube.com/watch?v=IBvZD689QR4. [Online]. Available: https://www.youtube.com/watch?v=IBvZD689QR4

[24] C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan, *High Efficiency video coding (HEVC) test model 16 (HM 16) improved encoder description update 2*, February 2015.