SIFT Features Tracking for Video Stabilization

Sebastiano Battiato, Giovanni Gallo, Giovanni Puglisi
University of Catania
(battiato,gallo,puglisi)@dmi.unict.it

Salvatore Scellato
Scuola Superiore di Catania
sascellato@ssc.unict.it

Abstract

This paper presents a video stabilization algorithm based on the extraction and tracking of Scale Invariant Feature Transform features through video frames. Implementation of SIFT operator is analyzed and adapted to be used in a feature-based motion estimation algorithm. SIFT features are extracted from video frames and then their trajectory is evaluated to estimate interframe motion. A modified version of Iterative Least Squares method is adopted to avoid estimation errors and features are tracked as they appear in nearby frames to improve video stability. Intentional camera motion is eventually filtered with Adaptive Motion Vector Integration. Results confirm the effectiveness of the method.

1. Introduction

In the last years hand-held video cameras have increasingly grown in popularity, allowing everyone to easily produce personal video footage, despite of the fact that videos retrieved from such devices is affected by unwanted camera shakes and jitters, resulting in video quality loss. Hence, video stabilization techniques have gained consensus, for they permit to obtain high quality and stable video footages even in non-optimal conditions. The best techniques make use of mechanical tools which physically avoid camera shakes, or exploit optical or electronical devices to influence how the camera sensor receives the input light; these methods are very expensive as they are based on sophisticated sensors that measure camera shakes and then control the jitter acting on the lens or on the CCD sensor [7].

On the other hand, digital video stabilization techniques make use only of information drawn from footage images and do not need any additional knowledge about camera physical motion. Furthermore, these approaches are not expensive, for they may be implemented easily both in real-time and post-processing systems.

Video stabilization systems have been widely investigated and several techniques have been proposed, with different issues and weak points.

A first group of techniques is based on black on block matching: they use different adaptive filters to refine motion estimation from block local vectors[2, 9, 15]. These algorithms do generally provide good results but are more likely to be mislead by video content with moving objects. This happens because they do not associate a unique descriptor to a block and neither track blocks along consecutive frames. This results in an inefficient motion estimation, since such objects will generate uncorrect matching blocks and therefore wrong estimated motion.

Feature-based algorithms extract features from video images and estimate interframe motion using their location. Some authors present techniques [4, 5] combining features computation with other robust filters; these methods have gained larger consensus for their good performances.

Another important stage in any digital video stabilization system is motion filtering, where estimated motion is evaluated to recognize intentional movement: several techniques such as Kalman filtering [6] and Motion Vector Integration [13] have been proposed and recently modified and enhanced to correct translational and rotational jitters [2, 14] according to real systems constraints. A video stabilization system based on SIFT features [11] has been recently presented in [16]. It uses SIFT features to estimate interframe motion: then a Kalman filter recognizes intentional movement and a particle filter lowers error variance. However, this system does not modify SIFT implementation nor adapts it to the video stabilization problem. Consequently, it is not possible to change SIFT algorithm's parameters to improve feature matching performances. Moreover, it is computationally intensive: both Kalman filtering and particle filtering must be performed on each frame and the latter implies intensive computation to update filter's weights for each frame.

Here we present a method which infers interframe motion by tracking SIFT features through consecutive frames: particularly, features are used in a motion estimation algorithm that individuates camera jitter and therefore allows to stabilize the video sequence.

The paper is organized as follows: in Sec. 2 our modifications to original SIFT algorithm are presented, then follows a detailed discussion about feature-based motion estimation. Motion filtering is explained in Sec. 3. Experimental results are shown in Sec. 4 and conclusions are summarized in Sec. 5.

2. SIFT implementation details

Scale Invariant Feature Transform [11] has been designed for extracting highly distinctive invariant features from images, which can be used to perform reliable matching of the same object or scene between different images. Originally these features were proposed for object recognition: features detected in a sample image are matched to a large database of previously extracted features from various object at different viewpoints.

SIFT algorithm is not crisply defined and has several free parameters, so each implementation must figure out some issues and how these issues influence the final result, making further improvements during testing phase. There are many steps where some simplifications or potential improvements can be designed to make features more suitable for video stabilization purposes.

The major stages of computation to generate the set of image features are scale-space extrema detection, keypoint localization, orientation assignment and descriptor computation.

This approach generates large numbers of features over the full range of scales and location. The resulting descriptors are highly distinctive since they contain large amount of information and this improves correct matching between features in different images.

The first stage of our implementation is building the scale-space pyramid, where input frames are repeatedly filtered and downsampled. In our implementation two successive Gaussians are separated by a constant factor of $k = \sqrt{2}$ and 5 images per octave are computed: this results in 4 Difference-of-Gaussian images and therefore extrema can be searched on two different images. The total number of octaves is always less than 4, since every additional octave requires the input image to be downsampled by a factor of 2 and repeatedly halving the image may end with useless few pixels.

When keypoints have been located, each one of them must be interpolated, evaluated for contrast and then passed through a Harris edge detector, as SIFT algorithm describes. Particularly, both contrast threshold and edge detector sensitivity affect the final number of keypoints, as they succeed in discarding poorly localized features or noisy and unstable keypoints.

Avoiding unstable features is very important in video stabilization, as highly distinctive features are required for motion estimation algorithms and uncorrect matches may easily mislead the algorithm. So, in order to obtain better features, original thresholds used in the SIFT algorithm for keypoints validation have been hardened to discard a larger number of features. Particularly, Harris detector and test for contrast resulted pivotal for discarding unstable or noisy features and by making harder these tests we obtain less features, much more stable and distinctive.

After orientation assignment has been fulfilled, a 128-element feature vector is derived and bilinear interpolation is successfully used to increase descriptor robustness and invariance. The final output is therefore a set of keypoints with their descriptors.

For further details the reader is referred to [11].

3. Motion estimation and filtering

Our approach uses SIFT keypoints as features to be tracked between frames and a feature-based matching algorithm that estimates interframe motion. SIFT keypoints are extracted from two consecutive frames and then these two sets of features are matched to obtain a *Local Motion Vector* for each keypoint. Not all local motion vectors give correct information about how the frame has moved relatively to the previous, so it is necessary to discard wrong matches that do not fit into the estimate trasformation and may mislead the process.

3.1. Feature matching

The first problem to address is relative to keypoint matching. In the original paper [11] it is performed using Euclidean distance between descriptors' vectors and a *distance ratio*, namely ratio of closest neighbour distance to that of the second-closest one, that can be checked against a threshold to discard false matches. In fact correct matches should have lower ratios while wrong ones should have ratios closer to one.

We have tested the SIFT operator on consecutive stabilized frames. In this case keypoints are likely to appear in the same location on both frames, so feature matching should match point quite in the same position on nearby images.

Correlation between these two variables is then easily investigated: when two keypoints are matched, it is easy to compute the Euclidean distance between the pixel position of the first keypoint and the pixel position of the second one.

While SIFT original implementation employs a threshold of 0.8, we noticed that this is not enough for our purposes: we investigated correlation between distance ratio and correctness of matches and found that using a value of 0.6 as threshold performs better in discarding wrong matches: actually, only few keypoint couples show such a low distance ratio, but they are more likely to be correct matchings than the many more that present higher distance ratios, as shown in Fig. 1. It is important to notice that a medium-size image (352×288 pixels) may reveal even thousands keypoints, but interframe matching performed as described results only in few hundreds couples.

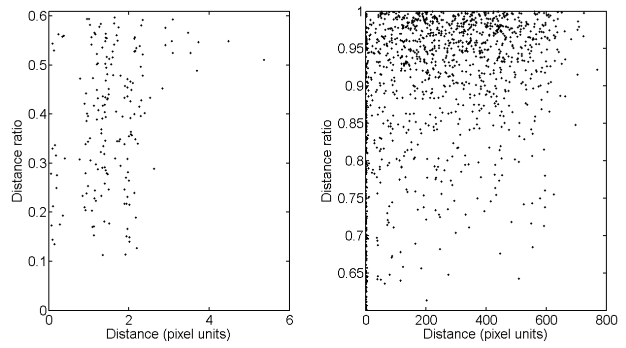

Figure 1. Correlation between pixel distance (X axis) and distance ratio (Y axis) for a medium size image: on the left features with a distance ratio below 0.6 and on the right remaining features

The result of this matching process is a list of keypoints pairs that can be easily used as input of the feature-based motion estimation algorithm. More formally, a Local Motion Vector is associated with each couple of matched features: since absolute positions (x_k, y_k) and $(\hat{x_k}, \hat{y_k})$ of both first and second keypoint in both images are known, the local motion vector $\mathbf{v_k}$ of the feature k can be easily derived and represents how the feature has supposedly moved from the previous frame to the next one.

3.2. Model fitting

The set of features local motion vectors retrieved during features matching are used to estimate the motion needed to overlay the current frame on the previous frame to minimize the visible motion. In order to estimate this Global Motion Vector, local motion vectors must be fit to a frame motion model. Even though the motion in the video is three-dimensional, global motion between frames can be estimated with a two-dimensional linear conformal model, usually the best trade-off between effectiveness and complexity. This model describes interframe motion using four different parameters, namely two shifts, one rotation angle and a zoom factor, and it associates feature (x_i, y_i) in frame I_n with feature (x_f, y_f) in frame I_{n+1} with this transformation:

$$\begin{cases} x_f = x_i\lambda\cos\theta - y_i\lambda\sin\theta + T_x \\ y_f = x_i\lambda\sin\theta + y_i\lambda\cos\theta + T_y \end{cases} \tag{1}$$

where λ is the zoom parameter, θ the rotation angle, T_x and T_y respectively X-axis and Y-axis shifts. In order to estimate four transformation parameters four different linear equations are required, so with only two couples of features the system has a solution. However features may be affected by noise so it is useful to apply a linear Least Squares Method on a set of redundant equations.

The whole set of features local motion vectors does not contains useful information for effective motion compensation, as probably it includes wrong matches or correct matches that indeed belong to self-moving objects in the filmed scene. Obviously there are some correct pairs that do represent real camera shakes but several points simply do not relate to such information.

Least Squares Method does not perform well when there is a large portion of outliers in the total number of features, as in this case. However, outliers can be identified, filtered out of the estimation process, resulting in better accuracy. Many robust exstimation techniques have been developed and applied in computer vision [1, 8], but in order to obtain real-time performance we have implemented a simplified version of iterative least squares. Iterative least squares refinement [3] can be employed to easily avoid outliers and refine solution. It is performed by determining the least squares solution with the whole set of features, calculating the error statistics for the data set compared to the computed motion and then removing any keypoint that present an error greater than a given adaptive threshold. This technique performs well, but for greater accuracy it must be combined with other filters which operate on the set of local motion vector.

At first, features which present a very large local motion vector are not likely to be correct, so they are immediately discarded using a fixed threshold on the Euclidean norm of the local vector. Then all local motion vectors are used to get a first estimation using Least Squares Method. At the same time, features are tracked during their motion through adjacent frames, as some features are likely to appear in several consecutive and therefore their Local Motion Vectors guarantee better accuracy.

After this first motion estimation step, each input keypoint is checked against the found parameters and its error is evaluated. Since each feature is related to a keypoint in the first image and another in the second, the first point is transformed using obtained parameters, deriving an expected point that may be compared to the real second point.

There are two different error measures that can be adopted to fulfill this task:

- *Euclidean distance* between expected and real point: does perform well rejecting matches that do not agree with the found translational components but may resulting misleading for border point when a rotation occurs;

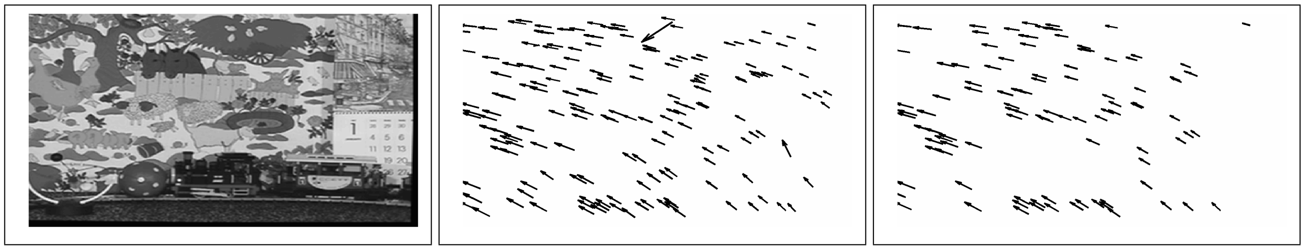
(a) (b) (c)

Figure 2. De-stabilized frame (a) with Local Motion Vectors after first rough estimation (b) and after filtering according to cumulative error (c): wrong feature matchings are discarded.

- *angle of rotation* between expected and real point, with respect to the center of the frame: this measure performs well with rotational components, but may result wrong when applied on translational components.

Obviously both measures are fit to discard uncorrect matches, so best choice is to adopt double filtering: error for each keypoint is evaluated using Euclidean distance and angle of rotation, then both errors are assigned to each point.

3.3. Cumulative Error

When features are tracked through consecutive frames the relative error is cumulated in order to use it for iterative least squares refinement. More formally, if feature k appears for the first time on frame n it is inserted in the first set of input points, then its errors are evaluated and stored but the feature is not included in the second data set for least squares refinement; instead if feature k has been previously tracked in frame $n-1$ and has a cumulative error $E_{n-1}^{cum}(k)$, its current error $E_n(k)$ is evaluated and linear interpolation is done to update cumulative errors for both measures:

$$E_n^{cum}(k) = (1-\alpha)E_{n-1}^{cum}(k) + \alpha E_n(k) \qquad (2)$$

With α varying in $[0,1]$ different behaviours can be modeled: as α approaches to one past error values are not considered whereas if α is close to zero actual errors are not. In this case, it is better to keep note of previous errors when discarding keypoints, since large cumulative errors usually corresponds to wrong features, so a weight value of $\alpha = 0.35$ has been employed.

Cumulative error statistics are then used to compute a filtering threshold: the entire set of features that have been previously tracked is sorted according to their two cumulative errors and then only keypoints in the first 50% of the set are inserted in the second data input set. Since there are two error measures, each point must be in the first half of both sets, otherwise it is discarded, as shown in Fig. 2.

Furthermore using cumulative errors rather than only actual errors improves resulting stabilization: in fact, moving objects or moving points that may easily mislead the Least Squares Method cumulate their errors in each frame and therefore are less likely to be inserted in the final data input set. It becomes hence possible to discard correct matches that move in a different way than the frame does, thus enhancing accuracy in motion compensation.

3.4. Motion filtering

Actually, a feature can move from a frame to the next one not only due to camera shakes but even for intentional panning movements or because it belongs to a moving object in the scene. This Local Motion Vector may result in wrong motion estimation, since video stabilization should occur only on the camera jitters and should not compensate wanted movements affecting video quality.

Motion Vector Integration [13] can be successfully used to filter the cumulative motion curve, that is the motion of the current frame respect to the first one. Cumulative translational motion vector is normally obtained by the sum of all previous inteframe motion vectors: in this case it is integrated with a damping coefficient δ and the Integrated Motion Vector of frame n is generated from

$$IMV(n) = \delta IMV(n-1) + GMV(n) \qquad (3)$$

where $GMV(n)$ is the Global Motion Vector between frame n and frame $n-1$ obtained by iterative least squares method. In this way compensation $C(n)$ to be applied on frame n is simply obtained as

$$C(n) = IMV(n) - IMV(n-1) \qquad (4)$$

and intentional motion is therefore smoothed and not compensated. The damping factor δ is usually chosen as a fixed value between 0 and 1, and various values can be found in literature [10, 13] depending on the degree of stabilization required: there is always a trade-off to be met between compensating slight jitter and following the intentional motion with a delay as little as possible.

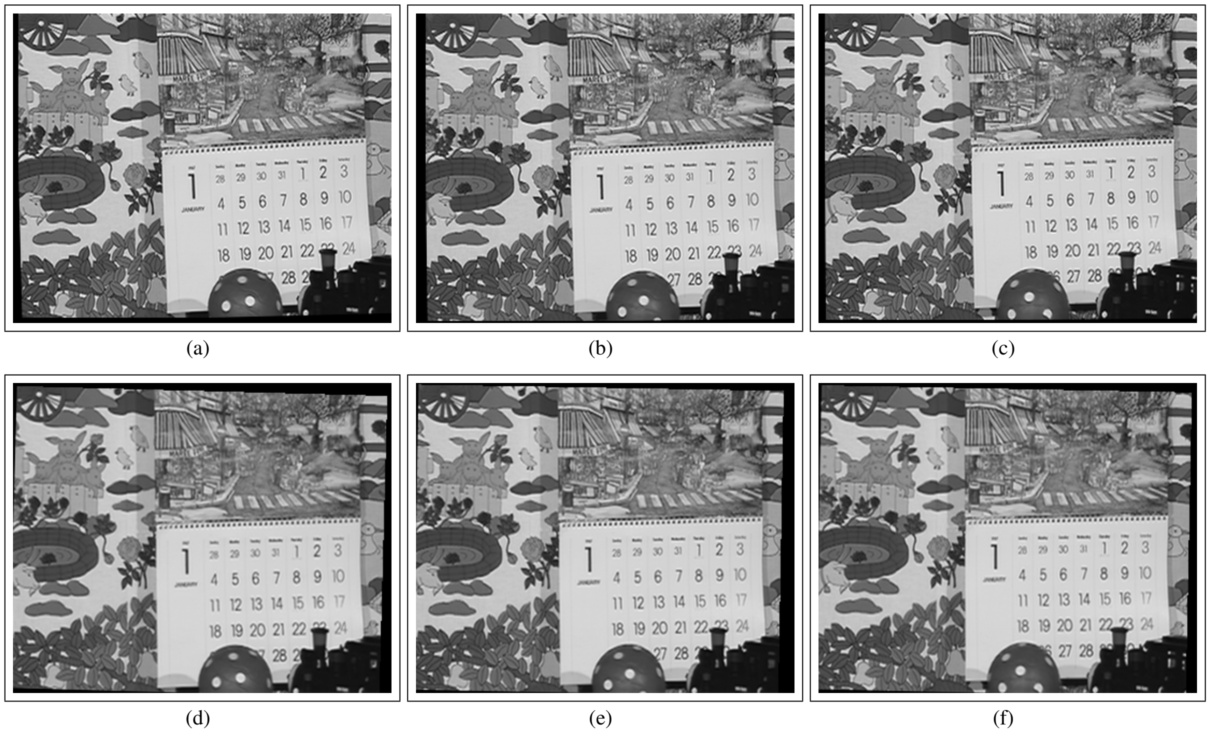

Figure 3. Frame from de-stabilized video (a-c) and relative stabilized frames (d-f)

However, using a fixed value does not provide enough flexibility, while an *adaptive* damping coefficient δ is a better solution [2]. Its value depends on the sum of the last two Global Motion Vectors, for if this sum is low the algorithm assumes an intentional static camera and chooses a high damping factor to strongly stabilize the sequence, whereas a high sum value implies large motion and therefore a lower value of δ is chosen, following more closely the intentional motion. This filtering is applied indipendently on the x and y translational components of the GMV.

4. Experimental results

The performances of our system have been evaluated on both artificially de-stabilized sequences and real jittered ones.

Peak Signal-to-Noise Ratio (PSNR) is an useful error measure to numerically evaluate how good performed stabilization is: PSNR between frame n and frame $n+1$ it is defined as

$$PNSR(n) = 10\log_{10}\frac{I_{MAX}^2}{MSE(n)} \tag{5}$$

where $MSE(n)$ is the Mean-Square-Error between frames and I_{MAX} is the maximum intensity value of a pixel. PNSR measures how an image is similar to another one: in this case it is useful to evaluate how much a sequence is stabilized by the algorithm, as consecutive frames in the resulting sequence should be more continuous than the input sequence and therefore PSNR should increase from the initial sequence to the final one. Inteframe Transformation Fidelity is then used as in [12] to objectively assess the stabilization brought by an algorithm:

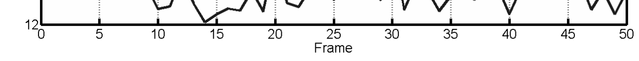

Figure 4. PSNR values for original and de-stabilized sequence

$$ITF = \frac{1}{N_{frame}-1}\sum_{k=1}^{N_{frame}-1} PSNR(k) \tag{6}$$

A stabilized sequence should have a higher ITF than the original sequence. For the tests we have employed some sequences publicly available for research purpose on the website `http://trace.eas.asu.edu/` of the University of Arizona: furthermore, they present broad variety of subjects and shooting conditions, allowing to test the algorithm in different working conditions. Unfortunately, only one sequence (foreman.cif) presents relevant jitter, whereas other videos are steadier and without observable shakes. So we tested our algorithm only on this sequence and on three sequences kindly provided by the authors of [16]: since these sequences present de-stabilization that is for us unknown, they have been processed as original unstable videos and ITF increments are compared with those obtained by aforementioned authors.

Sequence	Original ITF (dB)	Final ITF (dB)
foreman	25.75	27.17
seq1	15.22	19.57
seq2	17.66	19.65
seq3	21.86	23.05

Table 1. ITF on the original and stabilized sequences

Tab. 1 shows a substantial improvement in the ITF confirming the effectiveness of the algorithm: moreover, while videos provided by the authors of [16] presents ITF increments of about 3-5 %, we obtain improvements in the ITF of about 5-10 %.

On the other hand, it is useful to evaluate performances on an artificially de-stabilized sequence: de-stabilization is performed transforming each frame with known translational shifts and a rotation angle. The parameters are estimated by the algorithm and compared to the known values used for de-stabilization.

In Fig. 3 results for sequence mobile.cif are presented: de-stabilization values are recognized quite well, even if this sequence presents a slow panning that may mislead the motion estimation. Furthermore, PSNR increases for each couple of frames in the stabilized video (see Fig. 4).

5. Conclusions and future works

We have presented a novel approach for video stabilization based on the extraction and tracking of SIFT features through video frames. We make use of a feature-based motion estimation algorithm that tracks SIFT features extracted from video frames and then evaluates their trajectory to estimate interframe motion. A modified version of Iterative Least Squares method is adopted to avoid estimation errors and intentional camera motion is filtered with Adaptive Motion Vector Integration. Experiments have confirmed the effectiveness of the method. Further works include improvements in the SIFT extraction and in the motion filtering stage to handle different resolution videos.

References

[1] F. M. A. and B. R. C. Random sample consensus: a paradigm model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, Vol. 24(6):381–395, 1981.

[2] S. Auberger and C. Miro. Digital video stabilization architecture for low cost devices. *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, page 474, 2005.

[3] A. Björck. Numerical methods for least squares problems. *SIAM*, 1996.

[4] A. Bosco, A. Bruna, S. Battiato, and G. D. Bella. Video stabilization through dynamic analysis of frames signatures. *IEEE International Conference on Consumer Electronics*, 2006.

[5] A. Censi, A. Fusiello, and V. Roberto. Image stabilization by features tracking. *International Conference on Image Analysis and Processing*, 1999.

[6] S. Ertürk. Image sequence stabilisation based on kalman filtering of frame positions. *Electronics Letters*, 37(20), 2001.

[7] C. Inc. Canon faq: What is vari-angle prism?, http://www.canon.com/bctv/faq/vari.html.

[8] H. P. J. Robust statistical procedures. *SIAM*, 1996.

[9] S.-W. Jang, M. Pomplun, G.-Y. Kim, and H.-I. Choi. Adaptive robust estimation of affine parameters from block motion vectors. *Image and Vision Computing*, (23):1250–1263, August 2005.

[10] S. Ko, S. Lee, S. Jeon, and E. Kang. Fast digital image stabilizer based on gray-coded bit-plane matching. *IEEE Trans. on Consumer Electronics*, 45(3), 1999.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, Vol. 60(2):91–110, 2004.

[12] L. Mercenaro, G. Vernazza, and C. Regazzoni. Image stabilization algorithms for video-surveillance application. *IEEE Proceedings International Conference of Image Processing*, Vol. 1:p. 349–352, 2001.

[13] Paik, Park, and Kim. An adaptative motion decision system for digital image stabilizer based on edge pattern matching. *Consumer Electronics, Digest of Technical Papers*, pages 318–319, 1992.

[14] M. Tico and M. Vehvilainen. Constraint translational and rotational motion filtering for video stabilization. *Proceedings of the 13th European Signal Processing Conference (EUSIPCO)*, 2005.

[15] F. Vella, A. Castorina, M. Mancuso, and G. Messina. Digital image stabilization by adaptive block motion vectors filtering. *IEEE Trans. on Consumer Electronics*, 48(3):796–801, August 2002.

[16] J. Yang, D. Schonfeld, C. Chen, and M. Mohamed. Online video stabilization based on particle filters. *IEEE International Conference on Image Processing*, 2006.