Journal of Electronic Imaging

JElectronicImaging.org

Low cost rototranslational video stabilization algorithm

Giuseppe Spampinato
Arcangelo Ranieri Bruna
Sebastiano Battiato
Giovanni Puglisi

SPIE. IS&T imaging.org

Giuseppe Spampinato, Arcangelo Ranieri Bruna, Sebastiano Battiato, Giovanni Puglisi, "Low cost rototranslational video stabilization algorithm," *J. Electron. Imaging* **27**(5), 051224 (2018), doi: 10.1117/1.JEI.27.5.051224.

Low cost rototranslational video stabilization algorithm

Giuseppe Spampinato,[a,*] Arcangelo Ranieri Bruna,[a] Sebastiano Battiato,[b] and Giovanni Puglisi[c]
[a]STMicroelectronics, Advanced System Technology, Catania, Italy
[b]Università degli Studi di Catania, Dipartimento di Matematica e Informatica, Catania, Italy
[c]Università degli Studi di Cagliari, Dipartimento di Matematica e Informatica, Cagliari, Italy

Abstract. To avoid grabbing the unintentional user motion in a video sequence, video stabilization techniques are used to obtain better-looking video for the final user. We present a low power rototranslational solution, extending our previous work specifically addressed for translational motion only. The proposed technique achieves a high degree of robustness with respect to common difficult conditions like noise perturbations, illumination changes, and motion blurring. Moreover, it is also able to cope with regular patterns, moving objects and it is very precise, reaching about 7% of improvement in jitter attenuation, compared to previous results. Overall performances are competitive also in terms of computational cost: it runs at more than 30 frames/s with VGA sequences, with a CPU ARM926EJ-S at just 100 MHz clock frequency. *© 2018 SPIE and IS&T* [DOI: 10.1117/1.JEI.27.5.051224]

Keywords: video stabilization; integral projection; regular patterns; parameters extraction.

Paper 180043SS received Jan. 12, 2018; accepted for publication Apr. 24, 2018; published online May 15, 2018.

1 Introduction

In video processing, it is really important to enhance frame by frame quality to obtain the best possible output for final user. Different enhancement techniques have been proposed, such as, for example, exposure correction[1] and noise reduction.[2] Apart from the application of these kinds of processing, it is vital to stabilize the video to avoid unpleasant video shaking effects.

Video stabilization techniques can be grouped in two main categories: direct-[3] and feature-based methods.[4] The former approaches estimate the unknown parameters through global minimization criteria based on direct image information usually exploiting assumptions such as brightness constancy as a starting point. Some techniques (block based) split the image into blocks and the relative motion[5] is computed comparing blocks of consecutive frames. Matching is then performed within a search window minimizing a proper error function like the partial distortion elimination,[6] mean absolute difference,[7] or universal image quality index.[8] A compounding algorithm to detect the global motion vector is then applied.

Feature-based methods first compute a sparse set of features in the input image and then estimate the motion parameters from their matching. Recent papers have mainly adopted speeded-up robust features (SURF),[9] scale-invariant feature transform (SIFT),[10,11] and Kanade–Lucas–Tomasi techniques.[12] Even if there is no need to process the whole image, the disadvantage of feature based methods is that they are strictly dependent on feature point extraction step[13] in terms of accuracy and execution time.

In order to achieve a satisfying degree of robustness, algorithms based on features extraction and matching usually exploit reliable features (e.g., SIFT[14] and SURF[15]) and robust estimators (e.g., RANSAC[16]). These design choices, usually, do not allow real-time performances. Also block-based techniques are slow essentially because the whole image should be processed, in a block-by-block fashion, especially if features' extraction and matching are complex.

The goal of this paper is to present a robust rototranslational system, which is very flexible and low-cost, achieving very low power consumption, so suitable also for inexpensive and small video cameras. To reach this goal, the proposed technique is partially inspired by our previous work,[17] based on the usage of characteristic curves. This work has been also extended to cope with regular patterns, moving objects and to achieve subpixel precision. Moreover, the proposed rototranslational algorithm should obtain at least equal quality compared to the reference translational one,[17] without increasing complexity.

In order to estimate rotational movements, both characteristic curves and block based algorithms have been exploited. Typically, video stabilization approaches based on block matching, taking into account the limited block size and the high frame rate, assume only a translational motion for each block. This information, coming from different spatial locations in the frame, is then used to compute (through Least Squares) the global motion vector (e.g., two translations, one zoom factor, and one rotation in the similarity model).[18,19] This methodology, just filtering out outliers, generally works pretty well. Due to the good properties of the integral projection, in our approach, it has been used as a local motion estimator replacing block matching algorithm in Ref. 20. Additional filters have been also designed to cope with moving objects, to increase robustness of the whole system.

*Address all correspondence to: Giuseppe Spampinato, E-mail: giuseppe.spampinato@st.com

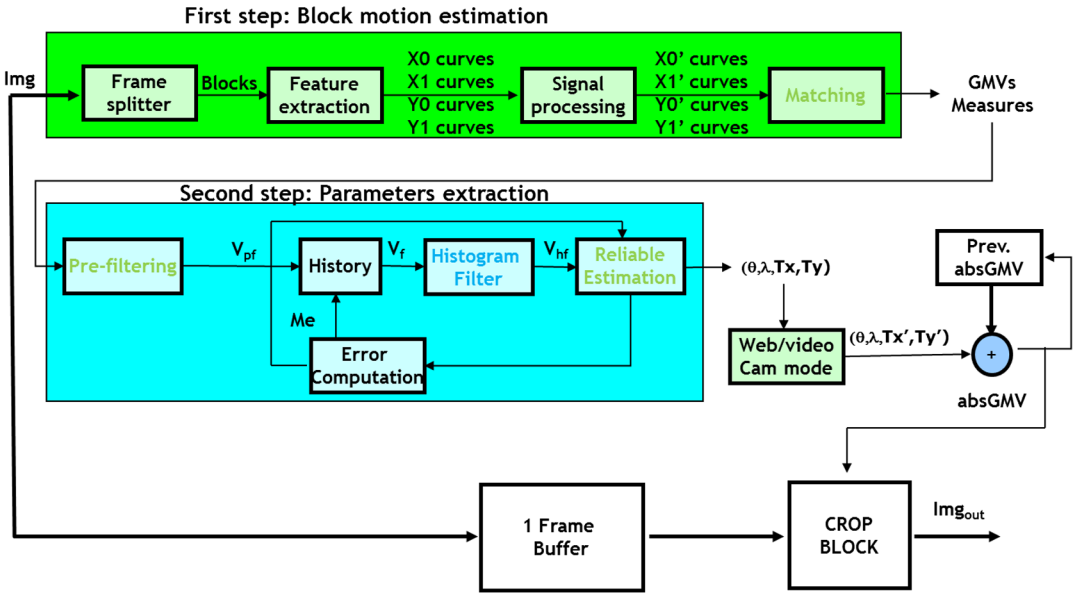


Fig. 1 Proposed algorithm: first step calculates motion vectors block by block; second step computes the global rototranslational transformation parameters.

This paper is structured as follows: in Sec. 2, the proposed method is shown; in Secs. 2.1 and 2.2, the block motion estimation and the parameters extraction are described. Then, the experimental results are reported in Sec. 3, followed by simulation results in Sec. 4. Finally, conclusions are sketched in Sec. 5.

2 Proposed Algorithm

The proposed algorithm is fundamentally distributed in two steps, as shown in Fig. 1. The first step, named block motion estimation, splits the frame into blocks, extracts and matches features, to obtain some measures and translational motion vectors blocks by blocks; whereas the second step, denoted parameters extraction, after the application of different filters to the set of motion vectors, paints out the global rototranslational transformation parameters, taking into account the estimated computation error.

The following sections, respectively, Secs. 2.1 and 2.2, will explain these two main steps in detail.

2.1 *Block Motion Estimation*

This step estimates the motion vectors from the incoming frames, dividing them into blocks. Moreover, it calculates some measures used in the following parameter extraction step. In the next subsections, four substeps will be explained: frame division into blocks, feature extraction, signal processing, and matching.

2.1.1 *Frame division into blocks*

Before the application of the proposed schema, the first thing to do is to split the incoming frames into blocks. It is a very delicate and important aspect, which will heavily impact the whole work. The number of blocks obtained is really important because the block should be big enough to allow effective motion estimation (in the first step) and small enough to have a considerable number of motion vectors to allow a good parameter extraction (in the second step).

For what concerns the estimation, the dimension of the block depends also from the searching window we want to use (that is the maximum allowed movement in the two directions). In fact, in the optimized matching phase (see Sec. 2.1.4), we can divide each dimension of the block in three parts, as indicated in Fig. 2.

The first and last parts of the block dimension are not used in the motion estimation, so only some pixels are used. Subjective experiments with different pixels used for matching (8, 16, 32, and 64) have been made. In these experiments, we noted that the "pixel used for matching" should be at least 32. Moreover, we considered, from subjective experiments, that at least 5% of the image dimension is acceptable for "searching window," so we obtained the data shown in Table 1.

If these parameters are acceptable for a reliable estimation, it is not the case of the parameter extraction. The number of blocks obtained (30 to 35) is too low for good functionality. For this reason, it is necessary to have more blocks, so more motion vectors are estimated. In our implementation, we use overlapped blocks with two parameters: Shift_X and Shift_Y, which represent the number of pixel to shift starting from the origin of the frame to obtain the new block, respectively, in horizontal and vertical direction. The total number of blocks "BlockNumber" can be computed using the following formula:

$$\begin{aligned}\text{BlockNumber}_X &= [(\text{Width} - \text{BlockSize}_X)/\text{Shift}_X] + 1,\\ \text{BlockNumber}_Y &= [(\text{Height} - \text{BlockSize}_Y)/\text{Shift}_Y] + 1,\\ \text{BlockNumber} &= \text{BlockNumber}_X \cdot \text{BlockNumber}_Y, \end{aligned} \quad (1)$$

where Width and Height are the horizontal and vertical dimensions of the frame, while BlockSize_X and BlockSize_Y

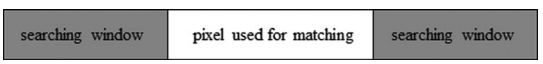


Fig. 2 Ideal vertical division of a dimension of a block.

Table 1 Ideal block dimensions to allow a good motion estimation.

Sequence type	Sequence dims	Searching window	Block dims	Block number
VGA	640×480	32×32	96×96	$6 \times 5 = 30$
>VGA <SVGA	$>640 \times 480$ $<800 \times 600$	40×40	112×112	≥ 30 ≤ 35
SVGA	800×600	40×40	112×112	$7 \times 5 = 35$
>SVGA <HDTV	$>800 \times 600$ $<1280 \times 720$	64×64	160×160	≥ 32 ≤ 35
HDTV	1280×720	64×64	160×160	$8 \times 4 = 32$

are the horizontal and vertical dimensions of the block. After various experiments, we empirically found that the number of blocks to allow parameter extraction step works well is at least 200. So, considering this additional constraint and Eq. (1), we obtained final data listed in Table 2.

2.1.2 *Feature extraction*

This block computes the chosen integral projection features[17] along the horizontal and vertical dimensions. For the sake of simplicity, let us assume that we have two gray-scale frames, captured with two successive captures, where M and N are the horizontal and vertical dimensions and p_{ij} is the pixel value in position (i, j). The characteristics curves, also called integral projections, along the horizontal and vertical dimensions (for each of the two consecutive frames), are, respectively, defined as in Ref. 17:

$$C_h(j) = \frac{1}{N}\sum_{i}^{N} p_{ij}, \quad C_v(i) = \frac{1}{M}\sum_{j}^{M} p_{ij}. \tag{2}$$

2.1.3 *Feature filtering*

This block deals with the problems to reduce the scene change illumination and motion blur effects. A simple global normalization not always handles these problems correctly. In the characteristics curves, the effect of the illumination change is a shift, which can be removed by filtering them with a high-pass filter. On the other hand, to avoid mismatch when motion blur is present, we should apply a low-pass filter to remove highest frequencies. Combining the two effects (illumination changes and motion blur), a band-pass filter (BPF) is a good choice. In this way, a second-order IIR filter has been chosen, to obtain a good tradeoff between implementation cost and results. The cutoff frequencies of the filter were fixed to $w_1 = 0.01$ Hz and $w_2 = 0.20$ Hz as in Ref. 17.

2.1.4 *Matching*

In order to properly evaluate the motion occurring between consecutive frames F_1 and F_2, the shift along the axes $(\text{off}_h, \text{off}_v)$ of both C_h and C_v curves can be calculated as follows:

$$P_h(s) = \frac{1}{M - |s|}\sum_{j=\max(1,-s)}^{\min(M-s,M)} |C_h^{F_1}(j) - C_h^{F_2}(j+s)|,$$
$$\text{off}_h = \{s\prime : P_h(s\prime) = \min\ P_h(s)\}, \tag{3}$$

$$P_v(s) = \frac{1}{N - |s|}\sum_{i=\max(1,-s)}^{\min(N-s,N)} |C_v^{F_1}(i) - C_v^{F_2}(i+s)|,$$
$$\text{off}_v = \{s\prime : P_v(s\prime) = \min\ P_v(s)\}. \tag{4}$$

The term s is the search window size and represents the maximum retrievable displacement (see Ref. 17).

Two innovative steps have been added at this step: regular patterns feature and subpixel estimation. In the case of regular patterns, the chosen features can have regular behavior and in the related matching curve, we can obtain many local minima, as shown in Fig. 3. A good choice is to take the lower minimum (in absolute value), to avoid too much movement following the patterns.

A better pattern handling is obtained considering two successive minimums on the matching curve (min_1 and min_2)

Table 2 Ideal block dimensions to allow good motion estimation and a good parameter extraction.

Sequence type	Sequence dims	Search window	Block dims	Shift	Block number
VGA	640×480	32×32	96×96	32×32	$17 \times 12 = 204$
>VGA <SVGA	$>640 \times 480$ $<800 \times 600$	40×40	112×112	$\geq 32 \times 32$ $\leq 40 \times 40$	≥ 204
SVGA	800×600	40×40	112×112	40×40	$18 \times 13 = 234$
>SVGA <HDTV	$>800 \times 600$ $<1280 \times 720$	64×64	160×160	$\geq 40 \times 40$ $\leq 48 \times 48$	≥ 234
HDTV	1280×720	64×64	160×160	48×48	$23 \times 12 = 276$

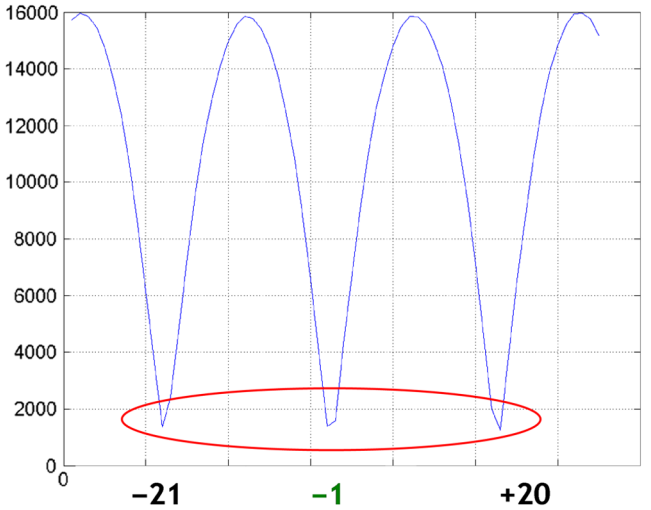


Fig. 3 Matching in the case of a frame with regular patterns ($s = [-32, 32]$).

and the maximum ($\max_1$) composed between these two minimums. The percentage P is calculated as follows:

$$P = (|\min_1 - \min_2|)/\max_1 - \min(\min_1, \min_2). \quad (5)$$

If the percentage P is less than a threshold T, the selected minimum will be the one with the lower position (in absolute value), otherwise, the selected new minimum will be normally selected (the minimum between $\min_1$ and $\min_2$). This way avoids wrong local minima selections. Experimental results demonstrate that a good value for T is 0.15.

Subpixel estimation is really important to improve the final results of the algorithm. To increase the precision of the matching, obtaining a float precision instead of integer precision, we consider three points in the matching: the chosen minimum and the previous and following integer value. With these three points, we perform a parabolic interpolation and the new minimum is computed with subpixel accuracy from the obtained parabola.

Considering the three points $A = (x_1, y_1)$, $B = (x_2, y_2)$, and $C = (x_3, y_3)$, the problem is to determine the coefficients a, b, c of the equation of the parabola $y = ax^2 + bx + c$ passing for the points A, B, C. Considering that we are only interested in calculation of V_x, that is the vertex X of the parabola, with some optimizations, we obtain:

$$\begin{aligned} V_x &= -b/(2 \cdot a) \\ &= (x_1 \cdot t_1 + x_2 \cdot t_2 + x_3 \cdot t_3)/[2 \cdot (t_1 + t_2 + t_3)]; \\ t_1 &= x_1 \cdot (y_3 - y_2); \quad t_2 = x_2 \cdot (y_1 - y_3); \\ t_3 &= x_3 \cdot (y_2 - y_1). \end{aligned} \quad (6)$$

So, the total operations to calculate V_x are further reduced: 7 sums, 1 shift, 6 multiplications, and 1 division.

Just to show an example, if we consider the three points: $A = (x_1, y_1) = (-5, 5111)$, $B = (x_2, y_2) = (-4, 4259)$, and $C = (x_3, y_3) = (-3, 4259)$, we obtain $V = (V_x, V_y) = (-3.5, 4152.5)$, that is exactly on the middle of the points B and C, since they have the same y-coordinate, as indicated in Fig. 4.

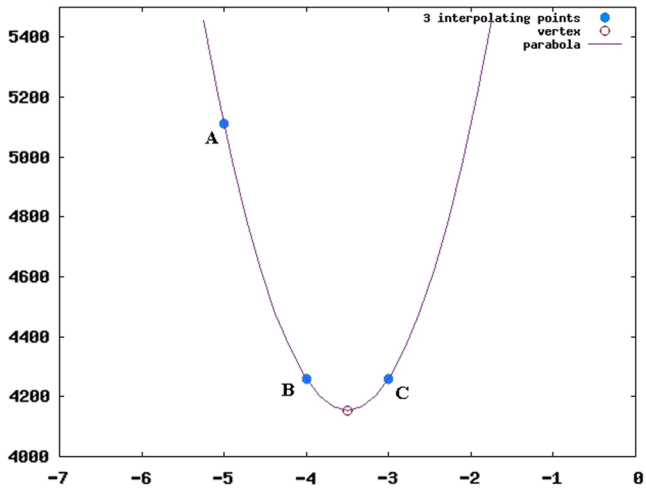


Fig. 4 Vertex of the parabola, where points B and C have the same y-coordinate.

2.2 *Parameters Estimation*

Starting from local motion vectors, through simple and fast rejection rules, our algorithm computes interframe transformation parameters. Global motion between consecutive frames is estimated considering a two-dimensional (2-D) similarity model (two shifts, a rotation angle, and a zoom factor), usually a good trade-off between effectiveness and complexity. In the following subsections, five substeps will be explained: prefiltering, history filter, histogram filter, reliable estimation, and error computation.

2.2.1 *Prefiltering*

Local motion estimator typically computes many wrong motion vectors. To filter out these vectors, not useful for global motion estimation, we make use of the following considerations:

- Local motion vectors usually have similar values in their neighborhood (motion continuity). It is the neighborhood similarity index already used in Ref. 20.
- The matching should be reliable. IP_Error_X, IP_Error_Y values, calculated using the matching error value provided by the local estimator based on integral projections, have to be low (effective match).
- Local motion vectors referred to homogeneous blocks are not reliable. The Disp_X and Disp_Y values, calculated as the difference between maximum and minimum value of the integral projection curves, have to be low.

The aforementioned rules have been derived after an exhaustive experimental phase devoted to achieve a suitable trade-off between overall complexity and real-time constraints. Both IP_Error_X, IP_Error_Y, and the inhomogeneity indexes (Disp_X and Disp_Y) have been already computed during the curve matching phase.

2.2.2 *History filter*

Due to moving objects in the scene, there are vectors that must be deleted in order to have a good interframe parameters

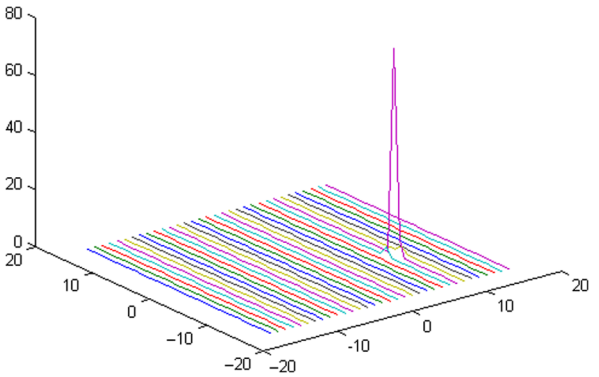


Fig. 5 An example of 2-D histogram computed from motion vector components.

estimation, so we need the "history filter" module to use previous frames information (see Ref. 20).

2.2.3 *Histogram filter*

This step improves the robustness of the proposed approach in presence of moving objects in the scene. Due to the limited amount of rotation and zoom factor involved in the mobile applications, motion vector components, and translational parameters should have close values. Considering a 2-D histogram of motion vector components, the vectors typically have the behavior depicted in Fig. 5: motion vectors are very close together.

In presence of big moving objects, the 2-D histogram contains multiple peaks, as depicted in Fig. 6. However, only one peak corresponds to the camera motion, the other one is related to a moving object entering in the scene. The histogram filter finds the highest peak in the 2-D histogram and filters out all the vectors too far from it (a threshold based on the maximum rotation and scale allowed sets the closeness). The remaining vectors (V_{hf}) are then propagated to the robust estimator module.

2.2.4 *Reliable estimation*

Global motion between adjacent frames can be estimated with a 2-D similarity model. Outliers can be still present also at this step, so we need a simple way to remove them. 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.

In order to obtain real-time performances, we have implemented a fast rejection technique, as indicated in Fig. 7:

- Starting from V_{hf} values, it computes a first least squares estimation of the interframe transformation parameters $(\lambda_1, \theta_1, T_{x1}, T_{y1})$.
- Distance filter. Motion vectors of which components along x and y axes are too far from the translational values (T_{x1}, T_{y1}) previously computed are filtered out. Due to the limited amount of rotation and zoom factor involved in the mobile applications, motion vector components, and translational parameters should have close values.
- Error filter based on the difference between the estimated and measured motion vectors considering both Euclidean (E_1) and angle-based distance (E_2) (see Ref. 20 for details).

According to the error E_1 and E_2, all V_d elements are sorted in increasing order and filter out a percentage with high error value. On the remaining V_s elements, another least squares estimation is then performed to obtain transformation parameters $(\lambda, \theta, T_x, T_y)$.

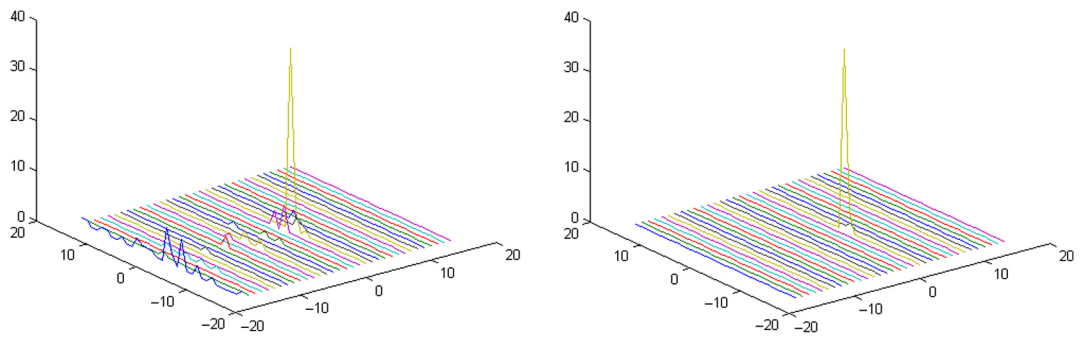
2.2.5 *Error computation*

Euclidean distance is used to fill the error matrix. This metric is simple to compute and able to efficiently distinguish between vectors belonging to objects entering in the scene and vectors describing the scene movements (see Ref. 20 for details).

3 Experimental Results

About 100 video sequences were tested in subjective manner. Videos have been taken at different resolutions (QVGA, VGA, 720p), with different source sensors (smartphones, digital cameras) and in different conditions (artificial and natural light). Moreover, critical conditions have been also taken into consideration like noise distortions, sudden illumination changes, motion blur, moving objects, and regular patterns.

Objective measures have been applied to some of these sequences. The objective quality parameters used (jitter and divergence) have been suggested in Ref. 21. The jitter measure evaluates the ability to reduce the high frequency



(a) (b)

Fig. 6 Motion vectors histogram (a) before and (b) after histogram filter application.

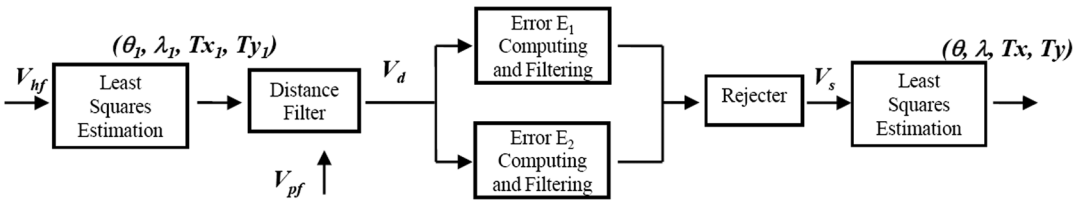


Fig. 7 Robust estimation module schema. Starting from V_f vector, it produces the final interframe parameters $(\lambda, \theta, T_x, T_y)$.

vibrations of the sequence and it should be as bigger as possible; while the divergence measure evaluates the ability of the system to follow the user intentional motion and it should be as lower as possible.

To obtain the jitter and divergence measures, we need to have the motion trajectories of the original (N) and stabilized (S) video sequences. The motion trajectory is defined as the one dimensional sequence comprising the values of a single motion parameter (e.g., horizontal translation or vertical translation) for each video frame with respect to the first frame. At this point, we can calculate the low and high frequency components of each motion trajectory, by a digital filtering. For this purpose, we used the following MATLAB code sequence, where X stands either for original (N) or stabilized (S) video sequences, while L and H represent, respectively, the low and high frequency components of the two motion trajectories:

$$\begin{aligned} &XL = \text{filtfilt}(h, 1, X); && \%\,\text{Low frequency component;}\\ &XH = X - XL; && \%\text{High frequency component;}\\ &h = \text{fir1}[2 * \text{round}(F), 2/F, \text{'low'}]; && \%F = \text{video frame rate.} \end{aligned} \tag{7}$$

The jitter attenuation and the divergence are calculated, respectively, by the following formulas:

$$\text{Jitter} = 10 \log 10 \frac{\sum_i (\text{NH}[i])^2}{\sum_i (\text{SH}[i])^2}, \tag{8}$$

$$\text{Divergence} = \frac{1}{\#\text{frames}} \sum_i |\text{SL}[i] - \text{NL}[i]|. \tag{9}$$

Table 3 shows a comparison between the translational method[17] and the proposed rototranslational method. We can note that, apart to handle the rotational movement, we obtain a mean improvement of 7.14% in jitter and a mean improvement of divergence of 1.27% for the test sequences. It is important to underline that in this objective comparison just translation is considered, so it is not obvious to obtain better results with the rototranslational approach.

Table 3 Jitter and divergence results of translational and rototranslational approaches.

	JitterX	JitterY	Div.X	Div.Y	Jitter%	Div.%
Transl.	5.9260	5.8602	2.6591	2.0050	—	—
Rototransl.	6.4388	6.1890	2.6048	1.9566	+7.14%	−1.27%

Just to show a graphical comparison between results obtained with translational and rototranslational algorithm, let us consider a sequence with slow horizontal panning.

In Figs. 8 and 9, the original curves represent the real original MVs, the Original Y low curves represent the ideal panning filter response, and the BPF curves represent the translational and rototranslational solutions. The BPF curves should be as close as possible to the ideal Original Y low curves. In these two figures, the X-axes represent the frame numbers of the sequence, while Y-axes represent the residual motion, calculated as difference between real and obtained motions.

In particular, in Fig. 8, it is evident that no good performance is reached. The rototranslational version is shown in Fig. 9. The numerical results obtained with rototranslational algorithm are the following: Jitter$X = 3.58$; Divergence$X = 1.06$; Jitter$Y = 5.18$; Divergence$Y = 0.09$. This indicates that graphically and numerically, we obtain enough good ability of the system to reduce the high frequency vibrations and good ability to follow the user intentional motion. Even if, in Fig. 9, still some fluctuations remain, they are in a limited pixel range [−0.5, 1.5] compared to Fig. 8, of which pixel range is [−2.5, 1.5]. The improvement for the proposed rototranslational algorithm is considerable and its Divergence$Y$ value is really low (0.09).

To show the image stabilization quality obtained, just consider Fig. 10 as an example. In this video, there is an evident unintentional movement. Figure 11 shows that both the outputs obtained with the translational method,[17] on the left, and the proposed method, on the right, performs a really

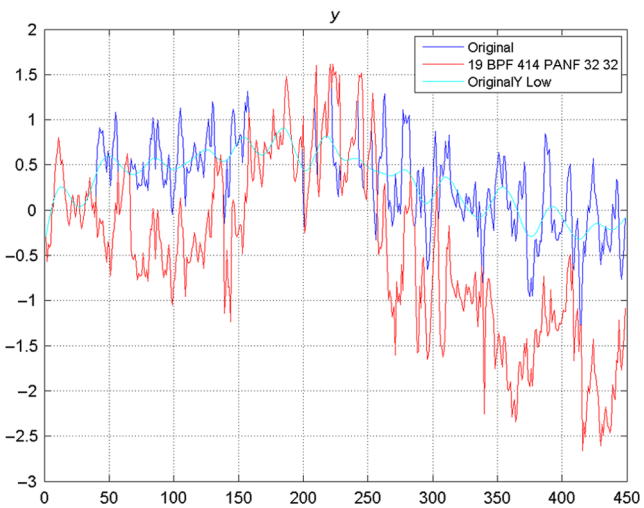


Fig. 8 Graphical representation of the translational algorithm curve compared to the original and ideal one, in the case of a sequence with slow panning.

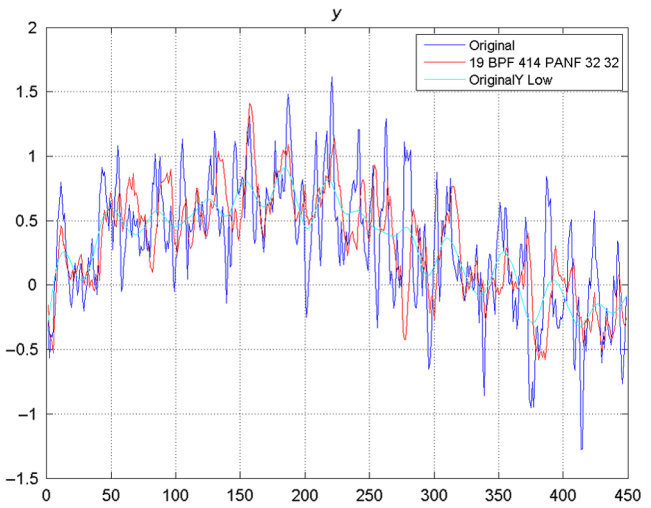


Fig. 9 Graphical representation of the proposed algorithm curve compared to the original and ideal one, in the case of a sequence with slow panning.

Fig. 10 Input video sequences with evident unintentional movement (Video 1, MPEG-4, 668 KB [URL: https://doi.org/10.1117/1.JEI.27.5.051224.1]).

good video stabilization, but on the right, we can note less video hand shaking.

More input video and video stabilization results obtained with translational method[17] and with the proposed method on different sequences are available in Ref. 22.

4 Simulation Results

To demonstrate that the proposed algorithm is a low power consumption solution, suitable for inexpensive and small video cameras, we executed it with a CPU ARM926EJ-S at just 100-MHz clock frequency. Simulation results are listed in Table 4. It shows simulation results obtained running the proposed algorithm with VGA (640×480) sequences, comparing them with the translational method.[17]

It is important to note that the features added to the proposed algorithm, which improve a lot the quality of the final video stabilization, are really light compared to the reference translational algorithm and, with VGA sequences, we can reach more than 30 frames/s.

Moreover, it is to consider that the algorithm was just written in ANSI-C, without any specific platform optimization, obtaining high frames per second with just 100-MHz clock frequency. Since "idle and wait states," that is, the number of cycles where bus is idle and busy, are big enough for sequential and nonsequential cycles, that is the number of sequential and nonsequential access to memory, there is a lot of margin of improvement to speed up the proposed algorithm for a specific platform.

5 Conclusion

A low-cost rototranslational algorithm for video stabilization has been developed, so easily used for real-time processing with the following characteristics, in common with the related translational algorithm: robust to noise distortions (IP used); robust to illumination changes (signal processing block), and not distracted by motion blur (signal processing block). In addition, apart from the fundamental role to correct rotations in the scene, we also added the following

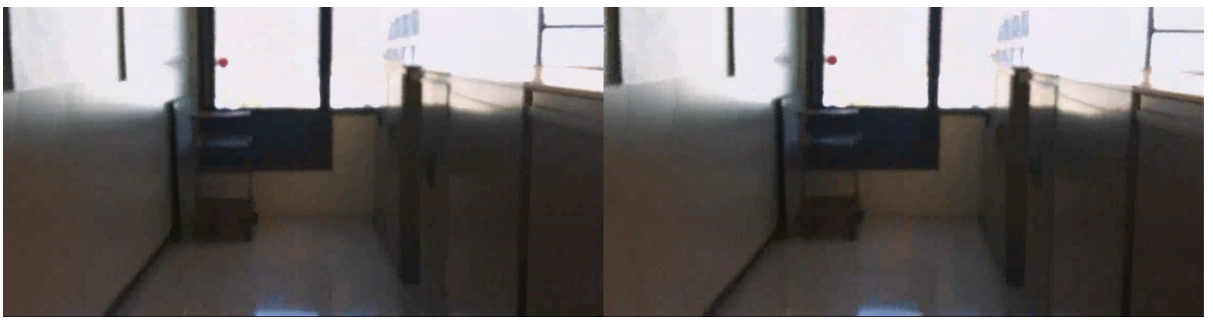


Fig. 11 Output videos obtained with translational (left) and proposed algorithm (right) (Video 2, MPEG-4, 816 KB [URL: https://doi.org/10.1117/1.JEI.27.5.051224.2]).

Table 4 ARM ARM926EJ-S simulation results.

Algorithm	Seq	NonSeq	Idle	Wait states	Total	Time (ms)	FPS
Translational	81,366	12,763	2,952,970	106,892	3,153,991	31.54	31.7
Proposed	83,938	15,205	3,026,308	114,347	3,239,798	32.39	30.9

characteristics: not distracted by regular patterns (regular patterns feature added) and very precise (subpixel estimation feature added).

It achieves significant improvements subjectively and objectively compared with state of translational algorithm, reaching about 7% of improvement in jitter attenuation and about 1% of improvement in divergence.

Moreover, simulation results show that the proposed, not-optimized algorithm runs at more than 30 frames/s with VGA sequences, with a simple ARM926EJ-S, at just 100-MHz clock frequency, obtaining similar performance compared to the reference translational algorithm.

Future work will be devoted to explore other transformations (i.e., homography) to handle more complex scenes and different kinds of distortions, trying not to increase performance.

References

1. S. Battiato, G. Messina, and A. Castorina, "Exposure correction for imaging devices: an overview," in *Single-Sensor Imaging: Methods and Applications for Digital Cameras*, pp. 323–349, CRC Press, Taylor & Francis Group, Boca Raton, Florida (2008).
2. A. Bosco et al., "Adaptive temporal filtering for CFA video sequences," in *IEEE ACIVS Proc.* (2002).
3. M. Irani and P. Anandan, "About direct methods," in *Proc. of Int. Workshop on Vision Algorithms*, Corfu, Greece, pp. 267–277 (1999).
4. P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Proc. of Int. Workshop on Vision Algorithms*, Corfu, Greece, pp. 278–294 (1999).
5. G. Puglisi and S. Battiato, "A robust image alignment algorithm for video stabilization purposes," *IEEE Trans. Circuits Syst. Video Technol.* **21**(10), 1390–1400 (2011).
6. A. P. Jagtap and P. V. Baviskar, "Review of block based video stabilization," *Int. Adv. Res. J. Sci. Eng. Technol.* **2**(4), 47–50 (2015).
7. D. Bhujbal and B. V. Pawar, "Review of video stabilization techniques using block based motion vectors," *Int. J. Adv. Res. Sci. Eng. Technol.* **3**(3), 1741–1747 (2016).
8. V. KovaazEvic et al., "Block-matching correlation motion estimation for frame-rate up-conversion," *J. Signal Process. Syst.* **84**(2), 283–292 (2016).
9. A. Salunkhe and S. Jagtap, "Robust feature-based digital video stabilization," *Int. J. Adv. Res. Electron. Commun. Eng.* **4**(8), 2163–2167 (2015).
10. M. Patel, N. Parmar, and M. Nilesh, "Comparative analysis on feature descriptor algorithms to aid video stabilization and smooth boundary reconstruction using in-painting and interpolation," *Int. J. Comput. Appl.* **140**(4), 35–39 (2016).
11. S. Battiato et al., "SIFT features tracking for video stabilization," in *Proc. Int. Conf. Image Analysis and Processing (ICIAP)*, pp. 825–830 (2007).
12. G. Spampinato et al., "Advanced feature based digital video stabilization," in *6th Int. Conf. on Consumer Electronics, ICCE Berlin* (2016).
13. P. Rawat and J. Singhai, "Review of motion estimation and video stabilization techniques for hand held mobile video," *Signal Image Process.: Int. J.* **2**(2), 159–168 (2011).
14. D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision* **60**(2), 91–110 (2004).
15. H. Bay et al., "SURF: speeded up robust features," *Comput. Vision Image Understanding* **110**(3), 346–359 (2008).
16. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm model fitting with applications to image analysis and automated cartography," *Commun. ACM* **24**(6), 381–395 (1981).
17. G. Spampinato et al., "Adaptive low cost algorithm for video stabilization," in *IEEE Int. Conf. on Image Analysis and Processing (ICIAP '17)* (2017).
18. S. Soldatov, K. Strelnikov, and D. Vatolin, "Low complexity global motion estimation from block motion vectors," in *Proc. GraphiCom* (2006).
19. W. H. Cho and K. S. Hong, "Affine motion based CMOS distortion analysis and CMOS digital image stabilization," *IEEE Trans. Consum. Electron.* **53**(3), 833–841 (2007).
20. S. Battiato, A. R. Bruna, and G. Puglisi, "A robust block based image/video registration approach for mobile imaging devices," *IEEE Trans. Multimedia* **12**(7), 622–635 (2010).
21. M. Niskanen, O. Silven, and M. Tico, "Video stabilization performance assessment," in *Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME '06)* (2006).
22. G. Spampinato et al., "Low cost rototranslational video stabilization algorithm: examples," http://iplab.dmi.unict.it/JEI2018 (2018).

Giuseppe Spampinato received his degree in computer science from the University of Catania (Italy) in 1996. From 1999, he works in AST (Advanced System Technology) Catania Lab of STMicroelectronics as software senior engineer. He is author of about 50 scientific publications (papers and patents) in the image and video processing field and he serves the reviewer of many international papers and a journal.

Arcangelo Ranieri Bruna is a principal engineer at STMicroelectronics. He received his MS degree in electronic engineering from the University of Palermo and a PhD degree in applied mathematics from the University of Catania. He is the author of more than 70 papers and several book chapters. His current research interests are related to computer vision, machine learning, and AI systems. He is an IEEE senior member.

Sebastiano Battiato is a full professor of computer science at the University of Catania. He is involved in research and directorship of the IPLab research lab (http://iplab.dmi.unict.it). His research interests include computer vision, imaging technology and multimedia forensics. He is also the director (and cofounder) of the International Computer Vision Summer School (ICVSS), Sicily, Italy, and a senior member of the IEEE.

Giovanni Puglisi received his MS degree in computer science engineering (summa cum laude) from Catania University in 2005, and his PhD degree in computer science in 2009. He joined the Department of Mathematics and Computer Science, University of Cagliari, as an associate professor in 2014. His research interests include image/video enhancement and processing, camera imaging technology, and multimedia forensics. He edited one book, coauthored more than 50 papers in international journals, conference proceedings, and book chapters.