Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Real-time video smoothing for small RC helicopters

Marynel Vazquez and Carolina Chang
Grupo de Inteligencia Artificial
Universidad Simón Bolívar
Caracas 1080, Venezuela
{marynel,cchang}@gia.usb.ve

***Abstract*—We present a real–time smoothing methodology for the stabilization of videos captured from small robotic helicopter platforms. We suppress supposedly unintended vibrations considering relative rotation and displacements between successive frames. We propose a similar, an affine or a bilinear transformation to model global motion assuming that camera movement dominates the motion field of aerial footage. A similar model gave worst results in comparison to the others, however all can effectively be employed to stabilize video and should be used depending on particular circumstances. In our implementation all transformations can be estimated by iterative least squares, and an affine model can also be adjusted by a proposed iterative total least squares procedure. Field experiments were carried out with a tele–operated helicopter that transmits wireless video to a receiver in ground where digital smoothing is done. With this configuration we stabilized video at an average speed between 20 and 28 fps, while surpassing problems generated because of the presence of high–levels of noise. We improved our system performance by predicting camera motion and got satisfactory results with even just a small delay of 3 frames. Extending our smoother with more complex vision understanding processes seems straightforward given its flexibility and robustness.**

***Index Terms*—Video Smoothing, Vision, Robotics.**

I. Introduction

Radio controlled (RC) helicopters have become widely used as robotic aerial platforms, thus enhancing image sequences captured from them can be advantageous. In particular, these sequences tend to present high–frequency motions that degrade their quality. As a simple illustration, consider a tele–operated helicopter with vision capabilities used for search and rescue operations. It would be useful for an emergency responder to retrieve visual information from the vehicle without distractions caused by irregular motion perturbations.

Digital video smoothing comprises the stabilization of image sequences by reducing perceived vibrations. These high–frequency motions are taken as a consequence of unwanted camera movements, such as those generated by flying dynamics. Even though video stabilization can be accomplished with sensors that give information about how the camera moves or mechanical dampers, these strategies require devices that may not always be available.

We present a video smoother that relies in computer vision technology for motion estimation, under the assumption that motion field between consecutive frames is dominated by camera movement. This common supposition works specially well for robotic helicopters [1], since independent moving objects in the scenes tend to be of small size given the altitude of the camera while the robot flies.

In our stabilization procedure different geometric models can be used to describe global motion induced by the change of position of the camera. Being flexible is important since no model works effectively under all circumstances. In addition, a good estimation of global motion facilitates the implementation of robust vision understanding processes, such as independent motion trackers [2]. Thus we present our smoothing methodology as an initial step towards the construction of a more complex vision application independent of expensive hardware.

Wireless video transmissions were accounted for in our tests. Under these circumstances noise degrades images and alter their content significantly. Since distortions tend to be temporal unless signal is lost, we evaluated the capability of our system to surpass this problem. Recovering from bad global motion estimations is important specially for online applications.

The rest of the paper is organized as follows. In Section II we first describe related work of interest. Afterwards, we explain our smoothing procedure in Section III and mention relevant aspects of our implementation in Section IV. We present our experimental results in Section V and finally conclude in Section VI.

II. Related Work

A smoothing stabilization process can be distinguished by the use of a non–static frame of reference to which register successive images of a sequence [3], [4]. Smoothing systems concentrate in reducing vibrations by updating this reference, while they avoid loosing the correspondence between images being registered. These systems don't need to be reinitialized because images can't be compared, which tends to happen for motionless stabilizers that rely on a static reference [5], [6], [7].

Motion vectors from which derive global motion are usually estimated by a block–matching algorithm or by employing Lucas–Kanade feature tracker [8]. The former looks for a correspondence of a block from one image to the other by reducing an error measure, such as the minimum mean absolute difference. Since this requires an expensive evaluation of all possible matching blocks, Estalayo et al. [9] limit the searching area to a predefined window. In addition, they use a median filter to improve point correspondences between images given noise in FLIR sequences.

For small inter–frame displacements, Lucas–Kanade feature tracker looks for a correspondence of selected blocks between

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

two images in a Newton–Raphson style minimization. Tomasi and Kanade [10] and Irani et al. [11] later answered the question of which features were good to track with a similar reasoning. Blocks that are not characterized by a constant intensity profile or a unidirectional pattern seem reliable in practice [1], [2], such as corners or salt–and–pepper textures.

Seeking to stabilize an image sequence, some global motion estimation procedures are constrained by the use of a certain (sometimes really simple) geometric model. Only translations along the coordinate axes were approximated by Ratakonda [12] and a rigid model is estimated sequentially by Guestrin et al. [6] and Chen and Lovell [13]. Further, Johansen [4] incorporates isotropic scaling into the rigid model and Matsushita et al. [3] go for an affine transformation.

After deriving global motion and its intended component, the amount of compensation needed to smooth a sequence can be calculated. Ratakonda [12] employs a derivative procedure with the most recent images captured to estimate intended motion, meanwhile others propose digital signal processing filters [6], [13] or adjusting a parabola to a group of estimated states [4]. More similar to our approach, Matsushita et al. [3] apply a convolution operation that returns the amount of compensation needed at a certain moment. This strategy uses a predetermined amount of global motion transformations relative to the actual frame being smoothed for the convolution. The smoothed sequence is delayed as in all previous cases where a group of frames is relevant to compute intended motion.

Some special considerations have been taken by few authors. Ratakonda [12] assumes motion to be zero if there is a large error between compensated frames. Besides, Morimoto and Chellappa [5] select good features every two frames seeking to reduce computations.

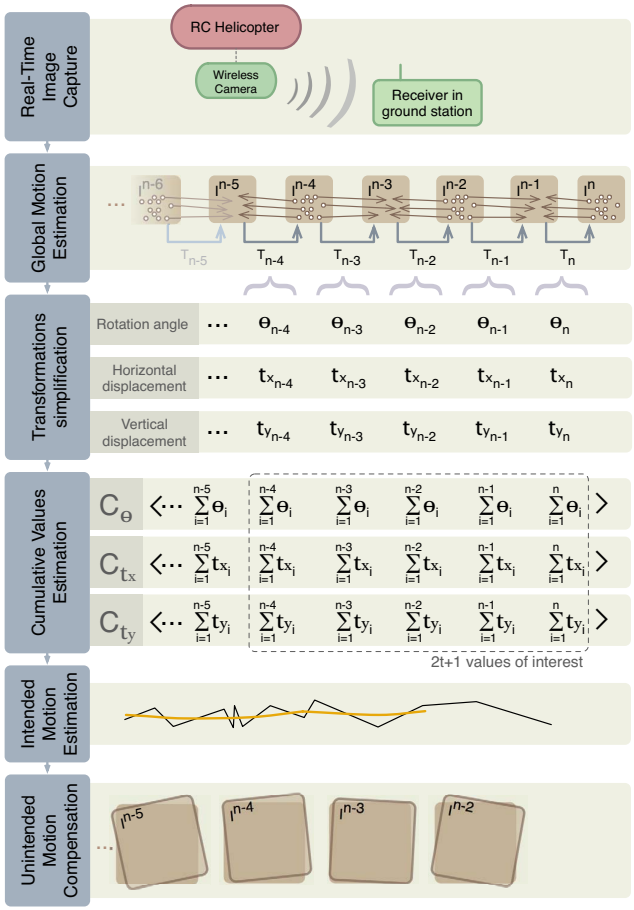


Fig. 1. Real–time smoothing methodology for helicopter platforms (exemplified with wireless video transmissions as tested in our experiments). Global motion is estimated between consecutive frames and represented by a geometric transformation T that is simplified for further processing. Intentional global motion is approximated from the cumulative rotation angle and displacements chains of values. Given global motion and its intentional component, unintended motions are finally detected and compensated for.

III. Video Smoothing

Our video smoothing procedure consists in estimating global motion, decomposing this global motion in order to obtain its unintentional component, and finally compensating for irregular movements. Figure 1 briefly describes our methodology and exemplifies the process as carried out in our field experiments.

We **estimate global motion** between each last pair of captured frames by approximating optic flow. Suppose we select "good" features to track from image I^{k-1} to I^k [14], we look for their correspondence $(x_i, y_i) \rightarrow (\dot{x}_i, \dot{y}_i)$ using an iterative implementation of the Lucas–Kanade feature tracker algorithm [15]. To accelerate the process, we select features every two frames and look for their correspondence in both forward and backward directions. The former case is trivial, while the latter requires inverting feature relations [5].

We adjust a transformation T to the set of correspondences by reducing an error measure. Given a point $\mathbf{p} = [x\ y]^T$, our implementation considers the following models:

$$\text{- Similar:} \quad T_s(\mathbf{p}) = \begin{bmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_4 \end{bmatrix} \tag{1}$$

$$\text{- Affine:} \quad T_a(\mathbf{p}) = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix} \tag{2}$$

$$\text{- Bilinear:} \quad T_b(\mathbf{p}) = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_5 \\ b_6 \end{bmatrix} + \begin{bmatrix} b_7 * x * y \\ b_8 * x * y \end{bmatrix} \tag{3}$$

where a_i, $1 \leq i \leq 6$, and b_j, $1 \leq j \leq 8$, are the parameters that compose each one of them. For a similar, an affine or a bilinear model, an iterative least squares (LS) procedure was implemented in a manner similar to that of Jung and Sukhatme [2]. In addition, iterative total least squares (TLS) can be used for an affine transformation. Our TLS implementation consists in finding two 3D planes that adjust as best as possible to the data. We look for $a_1X + a_2Y + Z + a_5 = 0$ and $a_3X + a_4Y + Z + a_6 = 0$, with X taking all x_i values, Y referring to y_i and Z being $\dot{x}_i$ or $\dot{y}_i$, respectively.

Every time a global motion transformation T is estimated,

the model is refined iteratively and outlier motion vectors are detected. We define a threshold ε over the Euclidean distance $e = \|(\dot{x}_i, \dot{y}_i) - T(x_i, y_i)\|_2$, so outliers are those vectors for which $e > \varepsilon$. Both iterative LS and TLS stop if a maximum number of iterations is reached or if the number of correspondences left are not enough to continue the estimation.

We **simplify global motion transformations** under a rigid geometric model by deriving the displacements and rotation angle that best describe them. This step is important since flexibility given to approximate as best as possible global motion complicates the estimation of its intended component. Horizontal and vertical displacements can be derived from equations (1), (2) and (3) with their respective parameters a_3 and a_4, a_5 and a_6 and b_5 and b_6. If the transformation adjusted doesn't include a reflection operation, a polar factorization of its 2×2 matrix easily gives the rotation angle that best describes it [16]. Otherwise, the transformation is taken as a "bad" representation of real camera movement and we assume constant motion. According to whether there is previous information or not, the last simplified transformation or a zero rotation angle and displacements are used to continue the stabilization process.

Three chains of **cumulative values** used to estimate intended motion are calculated from the simplified transformations, as illustrated mathematically in Figure 1. C_θ, C_{t_x} and C_{t_y} represent respectively the chains of cumulative rotation angles, horizontal displacements and vertical displacements calculated with respect to the first frame of the sequence.

Consider the neighborhood of transformations of T_{n-t} for an unbiased **intended motion estimation**. This vicinity is defined as $\{T_j \,|\, i - t \leq j \leq i + t\}$, with t an integer that establishes the size of the set. The $2t + 1$ cumulative rotation and displacements calculated from the simplification of these transformations allow us to estimate $\check{\theta}_{n-t}$, $\check{t}_{x_{n-t}}$ and $\check{t}_{y_{n-t}}$. These values represent intended rotation angle and displacements at instant of time $n - t$ and are calculated applying a convolution operation. For generality suppose ρ represents any of these parameters and C_ρ is its cumulative–values chain. We estimate intended $\check{\rho}_{n-t}$ as

$$\check{\rho}_{n-t} = \frac{\sum_{i=n-2t}^{n} C_\rho(i) G((n-t)-i)}{\sum_{j=-t}^{t} G(j)} \tag{4}$$

where G is a zero–mean Gaussian function with standard deviation σ.

Unintended motion compensation is accomplished by adjusting for extra rotation and displacements that generate vibrations. Extra $\tilde{\theta}_{n-t}$, $\tilde{t}_{x_{n-t}}$ and $\tilde{t}_{y_{n-t}}$ are obtained from the difference between global motion and its intentional component. Using again ρ without loss of generality,

$$\tilde{\rho}_{n-t} = C_\rho(n-t) - \check{\rho}_{n-t} \tag{5}$$

Finally, we calculate the value of a pixel $\mathbf{p}$ in the compensated image of I^{n-t+1} from $I^{n-t+1}(\tilde{T}_{n-t}(\mathbf{p}))$, where

$$\tilde{T}_{n-t}(\mathbf{p}) = \begin{bmatrix} \cos(\tilde{\theta}_{n-t}) & -\sin(\tilde{\theta}_{n-t}) \\ \sin(\tilde{\theta}_{n-t}) & \cos(\tilde{\theta}_{n-t}) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \tilde{t}_{x_{n-t}} \\ \tilde{t}_{y_{n-t}} \end{bmatrix} \tag{6}$$

IV. Application Details

Our smoothing methodology was implemented on a regular laptop computer as a stand alone application that processes video offline or in real–time. When images are grabbed, their dimensions are not necessarily the same of the raw video. Smaller frames can be smoothed depending on the processing power of the computer being used and the frame–rate that needs to be reached while stabilizing. Given our speed test results and the flexibility of our methodology, our application can easily be implemented on embedded computers. This further promotes its extension with other vision understanding processes for robot localization, perception and planning.

Our application includes the option of framing smoothed video. This capability intends to improve even more the quality of a processed sequence, so empty regions around compensated images being displayed won't be visible.

To overcome wrong feature correspondences we detect global motion transformations that possibly doesn't describe real optic flow. These include those with substantial scaling, displacements, rotation or stretching, as well as transformations that include a reflection operation. Undesired global motion estimations are common when processing noisy sequences (typical of wireless video transmissions), in which case it is usual for the Lucas–Kanade feature tracker to behave poorly.

V. Experimental Results

This section presents our experimental results when testing our digital smoother with video captured from a RC helicopter. We collected aerial footage recorded from regular cameras rigidly positioned in helicopters for an offline evaluation of our stabilization system. In addition, we used a TRex 600 helicopter that transmits 2.4 Ghz wireless video from a cheap micro–camera (also rigidly positioned in its frame) for field experiments. All tests were run on a MacBook laptop with a 2.16 GHz Intel Core 2 Duo processor and 2 GB of RAM. Images were processed with a resolution of 320×240 pixels.

A. Global Motion Estimation

We use *peak signal–to–noise ratio* (PSNR) [17] to define the "gain" as our evaluation metric for global motion estimation. Given consecutive frames f and g (of $w \times h$ pixels) and the latter's compensated image g_c, we calculate

$$\text{PSNR}(g_c, f) - \text{PSNR}(g, f) = 10 \log \frac{\sum_{i=0}^{w-1} \sum_{j=0}^{h-1} (g(i,j) - f(i,j))^2}{\sum_{i=0}^{w-1} \sum_{j=0}^{h-1} (g_c(i,j) - f(i,j))^2}$$

leaving out of the computation the pixels left empty in g_c. Hence, the "gain" measures how different two frames remain after compensating completely for global motion.

In this experiment 1881 images transmitted wirelessly (with substantial noise) and 1785 frames recorded directly from a helicopter were processed. We tested a threshold ε for outlier vectors detection equal to 0.25, 1.0, 3.0, 5.0, 8.0 and 15.0 and a maximum number of iterations to refine the model from 1

TABLE I
MODELS AND ESTIMATION METHODS EVALUATION: AVERAGE GAINS OBTAINED FOR THE BEST ε TESTED.

(a) Results for videos recorded directly using 7 pixels wide features

Model	1 It. Max.	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	10.35	10.70 ($\varepsilon = 8$)	10.72 ($\varepsilon = 8$)	10.73 ($\varepsilon = 8$)
Affine/TLS	10.33	10.71 ($\varepsilon = 5$)	10.72 ($\varepsilon = 8$)	10.72 ($\varepsilon = 5$)
Similar/LS	8.51	8.74 ($\varepsilon = 15$)	8.76 ($\varepsilon = 15$)	8.76 ($\varepsilon = 15$)
Bilinear/LS	10.47	10.82 ($\varepsilon = 5$)	10.86 ($\varepsilon = 8$)	10.86 ($\varepsilon = 8$)

(b) Results for videos recorded directly using 13 pixels wide features

Model	1 It. Max.	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	10.59	10.72 ($\varepsilon = 3$)	10.73 ($\varepsilon = 5$)	10.73 ($\varepsilon = 5$)
Affine/TLS	10.59	10.72 ($\varepsilon = 3$)	10.73 ($\varepsilon = 3$)	10.73 ($\varepsilon = 5$)
Similar/LS	8.74	8.78 ($\varepsilon = 15$)	8.79 ($\varepsilon = 15$)	8.79 ($\varepsilon = 15$)
Bilinear/LS	10.71	10.82 ($\varepsilon = 3$)	10.83 ($\varepsilon = 3$)	10.84 ($\varepsilon = 3$)

(c) Results for videos transmitted wirelessly using 7 pixels wide features

Model	1 It. Max.	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	4.83	4.92 ($\varepsilon = 3$)	4.92 ($\varepsilon = 3$)	4.92 ($\varepsilon = 3$)
Affine/TLS	4.80	4.90 ($\varepsilon = 3$)	4.90 ($\varepsilon = 3$)	4.91 ($\varepsilon = 3$)
Similar/LS	4.73	4.79 ($\varepsilon = 8$)	4.79 ($\varepsilon = 8$)	4.79 ($\varepsilon = 8$)
Bilinear/LS	4.60	4.70 ($\varepsilon = 5$)	4.71 ($\varepsilon = 3$)	4.71 ($\varepsilon = 5$)

(d) Results for videos transmitted wirelessly using 13 pixels wide features

Model	1 It. Max.	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	4.92	4.95 ($\varepsilon = 5$)	4.95 ($\varepsilon = 5$)	4.95 ($\varepsilon = 5$)
Affine/TLS	4.92	4.94 ($\varepsilon = 5$)	4.95 ($\varepsilon = 5$)	4.95 ($\varepsilon = 5$)
Similar/LS	4.79	4.81 ($\varepsilon = 8$)	4.81 ($\varepsilon = 8$)	4.81 ($\varepsilon = 8$)
Bilinear/LS	4.77	4.82 ($\varepsilon = 3$)	4.83 ($\varepsilon = 3$)	4.83 ($\varepsilon = 3$)

TABLE II
CONVERGENCE SPEED OF ITERATIVE LS AND TLS: AVERAGE NUMBER OF ITERATIONS OBTAINED FOR THE BEST ε TESTED.

(a) Results for videos recorded directly using 7 pixels wide features

Model	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	1.166 ($\varepsilon = 8.0$)	1.218 ($\varepsilon = 8.0$)	1.231 ($\varepsilon = 8.0$)
Affine/TLS	1.188 ($\varepsilon = 5.0$)	1.212 ($\varepsilon = 8.0$)	1.262 ($\varepsilon = 5.0$)
Similar/LS	1.188 ($\varepsilon = 15.0$)	1.256 ($\varepsilon = 15.0$)	1.275 ($\varepsilon = 15.0$)
Bilinear/LS	1.187 ($\varepsilon = 5.0$)	1.220 ($\varepsilon = 8.0$)	1.233 ($\varepsilon = 8.0$)

(b) Results for videos recorded directly using 13 pixels wide features

Model	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	1.147 ($\varepsilon = 3.0$)	1.172 ($\varepsilon = 5.0$)	1.191 ($\varepsilon = 5.0$)
Affine/TLS	1.147 ($\varepsilon = 3.0$)	1.182 ($\varepsilon = 3.0$)	1.192 ($\varepsilon = 5.0$)
Similar/LS	1.131 ($\varepsilon = 15.0$)	1.164 ($\varepsilon = 15.0$)	1.173 ($\varepsilon = 15.0$)
Bilinear/LS	1.143 ($\varepsilon = 3.0$)	1.192 ($\varepsilon = 3.0$)	1.210 ($\varepsilon = 3.0$)

(c) Results for videos transmitted wirelessly using 7 pixels wide features

Model	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	1.139 ($\varepsilon = 3.0$)	1.169 ($\varepsilon = 3.0$)	1.174 ($\varepsilon = 3.0$)
Affine/TLS	1.138 ($\varepsilon = 3.0$)	1.167 ($\varepsilon = 3.0$)	1.173 ($\varepsilon = 3.0$)
Similar/LS	1.093 ($\varepsilon = 8.0$)	1.113 ($\varepsilon = 8.0$)	1.116 ($\varepsilon = 8.0$)
Bilinear/LS	1.102 ($\varepsilon = 5.0$)	1.162 ($\varepsilon = 3.0$)	1.137 ($\varepsilon = 5.0$)

(d) Results for videos transmitted wirelessly using 13 pixels wide features

Model	2 Its. Max.	3 Its. Max.	4 Its. Max.
Affine/LS	1.057 ($\varepsilon = 5.0$)	1.074 ($\varepsilon = 5.0$)	1.079 ($\varepsilon = 5.0$)
Affine/TLS	1.056 ($\varepsilon = 5.0$)	1.072 ($\varepsilon = 5.0$)	1.077 ($\varepsilon = 5.0$)
Similar/LS	1.056 ($\varepsilon = 8.0$)	1.070 ($\varepsilon = 8.0$)	1.072 ($\varepsilon = 8.0$)
Bilinear/LS	1.076 ($\varepsilon = 3.0$)	1.093 ($\varepsilon = 3.0$)	1.097 ($\varepsilon = 3.0$)

to 4. 100 "good" features were selected at most with a size of either 7×7 or 13×13 pixels for optic flow estimation [10].

Any of the geometric models implemented gave positive average gains after compensating for global motion, as described in Table I. Thus any of these models can be used effectively to represent camera movement under the assumption that it dominates motion field. Since real camera trajectory is unknown, our results don't allow us to decide which model is better for aerial footage global motion estimation.

Best average gains for a similar transformation were obtained for $\varepsilon = 8.0$ and $\varepsilon = 15.0$ in particular. Yet they were notably lower than using an affine or a bilinear model for images recorded directly from a flying vehicle. For this reason it seems that the values of ε tested weren't the best for a similar transformation applied to a sequence without substantial noise. The difference between results for wireless videos wasn't as relevant, possibly because of important image distortions generated by the transmission.

Jung and Sukhatme prefer a bilinear transformation over an affine for global motion estimation from a mobile robot [2], however we didn't get meaningful results that in average made us favor the more complex model. A selection of ε equal to 3.0, 5.0 or 8.0 seems to be a good option for both transformations with respect to the other values considered.

Outlier motion vectors were detected quickly. Mean number of iterations done to refine global motion transformations (with the best ε found in this experiment for each case) are given in Table II. The average number of iterations is below 2 under all tested configurations and since both LS and TLS gave similar results, we can't prefer one over the other.

It's worth noting that $\varepsilon = 0.25$ or $\varepsilon = 1$ is not recommended for any model and estimation method implemented. We didn't always get better gains as more iterations were done to refine global motion with the use of these values.

B. Real–time Performance

This experiment evaluates the processing speed of our digital video smoother under real–time robotic field tests. We tele–operated our RC helicopter and transmitted wireless video from it to a receiver in ground, as depicted in Figure 1. This analog signal was digitized and stabilized online using a 15–transformations neighborhood. All geometric models were estimated with a maximum of 4 iterations for iterative LS and TLS and ε according to prior recommendations. Tests were run over 7 and a half minutes of video (13500 frames) with diverse content. Aerial footage included a car, a mobile robot or people moving around a big open yard, as shown in Fig. 2.

(a) Automobile (b) Mobile Robot (c) Pedestrians

Fig. 2. Footage content for real–time performance evaluation.

Average frame–rates obtained under different configurations are presented in Figure 3, where it can be seen that our digital smoother runs faster as the size and number of selected "good" features are reduced. Stabilized sequences were visualized slowly with some temporary interruptions for a set of maximum 500 features of 13×13 pixels. However, with a maximum of 100 features of 7×7 pixels no interruptions disturbed visually and our average frame–rate was between 20 and 28 frames per second.

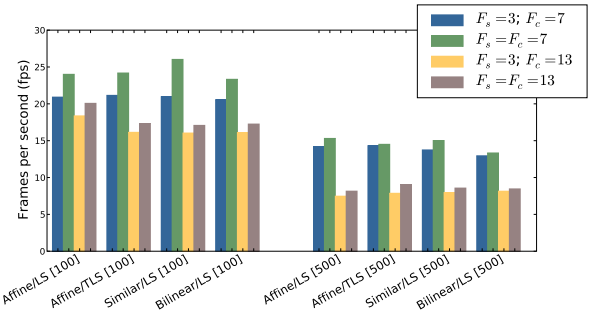


Fig. 3. Mean fps obtained during field tests. Results with a maximum of 100 or 500 "good" features (indicated between square brackets) are presented for each model and estimation method implemented. F_s and F_c denote respectively the size of the quadrangles used for features selection and their correspondence estimation.

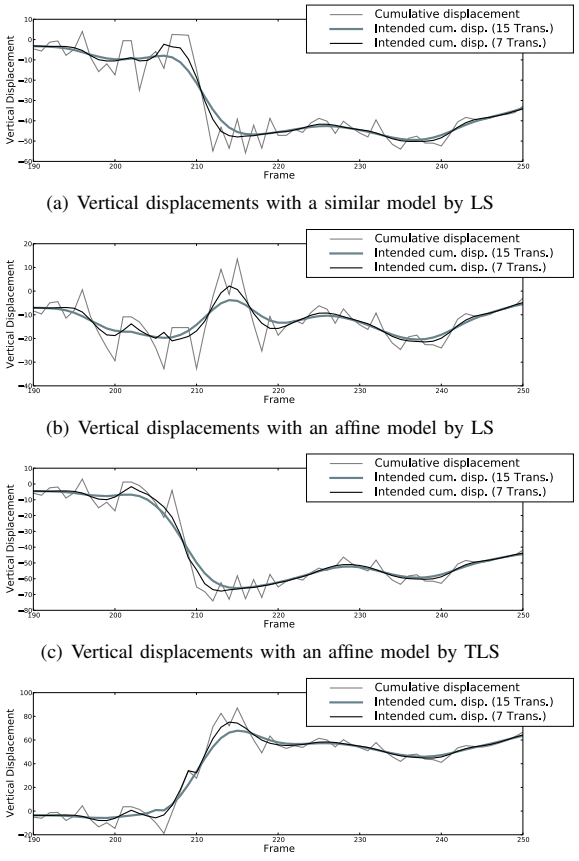
C. Visual Video Enhancement

The purpose of this experiment is to evaluate whether or not the smoothing methodology proposed improves perceived video quality. We chose ε to detect outliers given best results previously obtained and set a maximum of 4 iterations to estimate global motion transformations. At most 100 features were selected for optic flow approximation with a size of 7×7 pixels. The standard deviation σ of the Gaussian employed for intended motion estimation was $\sqrt{t}$, depending on the neighborhood of $2t + 1$ transformations considered, as explained in Section III.

In this experiment two aerial image sequences were smoothed: one presents many high–frequency motions that perturb visually, whereas the other is characterized by distorted noisy frames that result from wireless video transmission.

Video smoothing was satisfactorily accomplished using both 7 and 15 cumulative values to convolute with the Gaussian kernel proposed. Naturally, a smoother intended motion is estimated with a bigger neighborhood of transformations. The combination of geometric model and estimation method chosen affects the chains of cumulative values obtained. This fact evidences how errors during the processing modifies substantially the relative rotation angle or displacements of any frame with respect to the first one. As an example, Figure 4 depicts different vertical displacements estimated for a group of frames of the sequence with significant vibrations.

Our strategy to filter transformations was successful to overcome wrong global motion estimations when processing the video with substantial noise. Frames degradation complicates tracking features between them and errors generated at this step end up producing unwanted artifacts in the stabilized sequence. Figure 5(a) shows 8 frames stabilized filtering bilinear transformations that possibly doesn't describe real optic flow. We can appreciate the advantage of discarding certain transformations by comparing this result with the sequence smoothed without filtering, as presented in Fig. 5(b). Notice how motion in this case seems irregular and big empty regions generated by wrong motion compensations tend to appear without reason. More erroneous feature correspondences are



(a) Vertical displacements with a similar model by LS

(b) Vertical displacements with an affine model by LS

(c) Vertical displacements with an affine model by TLS

(d) Vertical displacements with a bilinear model by LS

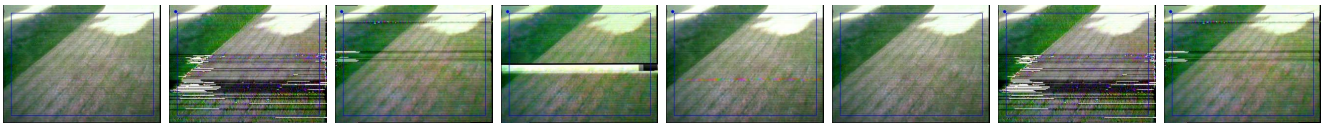
Fig. 4. Vertical displacements estimated with different combinations of geometric models and estimation methods for the same group of frames of a sequence with significant vibrations. Gray thin lines denote cumulative displacements with respect to the first frame. Black thin lines represent estimated intended displacements using a 7–transformations neighborhood. Thicker lines show smoother intended displacements for a bigger vicinity.

calculated with the presence of high–levels of noise and many are discarded by iterative LS and TLS. Even though, significant image distortions still affect global motion estimation and a low–quality smoothed sequence is obtained because of them.

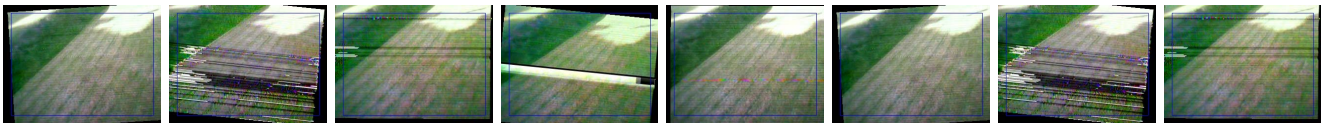
The filtering of transformations for the video with high–magnitude vibrations wasn't as effective as expected. Predicted constant movement didn't describe correctly sudden changes in camera motion when our systems incorrectly classified wrong transformations. We used the same settings as with the other footage and we believe this configuration was not the best.

VI. Conclusions

We described a method for real–time video smoothing intended to stabilize aerial footage captured from helicopter platforms. Our system performance depends on the parameterization of many settings, however it is flexible enough to adapt to particular situations and necessities. Though we smoothed video online from a remote ground station, our methodology can be implemented as well on embedded computers.



(a) Smoothed sequence filtering transformations



(b) Smoothed sequence without filtering transformations

Fig. 5. Smoothed noisy video. Sequence is presented with and without filtering bilinear transformations that probably doesn't describe real optic flow.

A unique feature of our stabilizer is the possibility it offers to choose between 4 combinations of transformations and estimation methods to find a model that represents global motion. We didn't find evidence that made us prefer either the proposed TLS procedure or the classic LS method to estimate an affine transformation. Likewise, we couldn't determine which geometric model used is better to estimate global motion. Jung and Sukhatme go for a bilinear transformation [2], yet we can't make it our favorite given our experimental results. The fact that we can't uphold their preference might be due to the higher frame–rate at which we smoothed videos in comparison to their processing speed. In spite of successful results given by all tested transformations and estimation methods, additional combinations can be implemented.

Our experimental outcomes showed empirically that the refinement of global motion transformations tends to be useful. This is because refinement helps detecting outliers in the sets of motion vectors calculated for consecutive images. Since iterative LS and TLS tend to converge fast, they are well suited for real–time robotic applications.

We smoothed video online sufficiently fast for its reproduction. Chen and Lovell's implementation of a digital stabilizer is the fastest known by us that considers rotation and translations to model global motion [13]. They were able to process as many as 17 images of 320×240 pixels per second using specialized hardware. In comparison, we effectively smoothed video at a speed between 20 and 28 fps.

We were able to overcome successfully wrong estimations of optic flow by predicting camera motion. This was done adjusting our application settings according to the video being processed. If no attention is given to this situation, visual artifacts usually appear and degrade video quality under the presence of high–levels of noise.

We observed an effective reduction of high–frequency motions introducing a small delay of 3 frames when displaying enhanced video. Though our method can't stabilize immediately the last captured frame, it doesn't need to be reinitialized because of lack of correspondences when estimating optic flow.

ACKNOWLEDGMENT

Research partially funded by Project LOCTI 5217 from "Diario El Universal" (http://www.eluniversal.com) to USB.

REFERENCES

[1] A. Lookingbill, D. Lieb, D. Stavens, and S. Thrun, "Learning activity-based ground models from a moving helicopter platform," in *ICRA'05*, 2005, pp. 3948–3953.

[2] B. Jung and G. S. Sukhatme, "Real-time motion tracking from a mobile robot," Center for Robotics and Embedded Systems, University of Southern California, CRES-05-008, 2005.

[3] Y. Matsushita, E. Ofek, X. Tang, and H.-Y. Shum, "Full-frame video stabilization," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 50–57.

[4] D. L. Johansen, "Video stabilization and target localization using feature tracking with small UAV video," Master's thesis, Brigham Young University, December 2006.

[5] C. Morimoto and R. Chellappa, "Automatic digital image stabilization," in *IEEE International Conference on Pattern Recognition*, 1996.

[6] C. Guestrin, F. Cozman, and E. Krotkov, "Image stabilization for feature tracking and generation of stable video overlays," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-97-42, November 1997.

[7] J. Yang, D. Schonfeld, C. Chen, and M. Mohamed, "Online video stabilization based on particle filters," in *ICIP06*, 2006, pp. 1545–1548.

[8] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI81*, 1981, pp. 674–679. [Online]. Available: citeseer.ist.psu.edu/lucas81iterative.html

[9] E. Estalayo, L. Salgado, F. Jaureguizar, and N. García, "Efficient image stabilization and automatic target detection in aerial FLIR sequences," in *Automatic Target Recognition XVI. Edited by Sadjadi, Firooz A.. Proceedings of the SPIE, Volume 6234, pp. 62340N (2006).*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6234, Jun. 2006.

[10] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, Jun. 1994. [Online]. Available: http://citeseer.ist.psu.edu/shi94good.html

[11] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," *Int. J. Comput. Vision*, vol. 12, no. 1, pp. 5–16, 1994.

[12] K. Ratakonda, "Real-time digital video stabilization for multi-media applications," in *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS'98)*, vol. 4, 1998, pp. 69–72.

[13] S. Chen and B. C. Lovell, "Real-Time MMX-Accelerated Image Stabilization System," in *Proceedings of Image and Vision Computing '01 New Zealand*, 2001, pp. 163–168.

[14] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University, Tech. Rep. CMU-CS-91-132, April 1991. [Online]. Available: http://citeseer.ist.psu.edu/tomasi91detection.html

[15] J.-Y. Bouget, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm," Microprocessor Research Labs, Intel Corporation, Tech. Rep., 1999.

[16] K. Shoemake and T. Duff, "Matrix animation and polar decomposition," in *Proc. of the Graphics Interface*, Vancouver, CA, 1992, pp. 258–264.

[17] L. Marcenaro, G. Vernazza, and C. Regazzoni, "Image stabilization algorithms for video-surveillance applications," in *ICIP01*, 2001, pp. I: 349–352.