Available online at www.sciencedirect.com
SCIENCE DIRECT®
J. Vis. Commun. Image R. 17 (2006) 659–673
JOURNAL OF VISUAL Communication & IMAGE Representation
www.elsevier.com/locate/jvci

A robust real-time video stabilization algorithm

Hung-Chang Chang [a], Shang-Hong Lai [a,*], Kuang-Rong Lu [b]

[a] *Department of Computer Science, National Tsing Hua University, Hsinchu 30043, Taiwan, ROC*
[b] *Ulead Systems, Inc., Taipei, Taiwan*

Received 17 May 2004; accepted 15 October 2005
Available online 4 January 2006

Abstract

The acquisition of digital video usually suffers from undesirable camera jitters due to unstable camera motions. In this paper, we propose a robust real-time video stabilization algorithm that alleviates the undesirable jitter motions from the unstable video to produce a stabilized video. In the proposed algorithm, we first compute the sparse optical flow vectors between successive frames, followed by estimating the camera motion by fitting the computed optical flow vectors to a simplified affine motion model with a robust trimmed least squares method. Then the computed camera motion parameters are smoothed temporally to reduce the motion fluctuations by using a regularization method. Finally, we transform all frames in the video sequence based on the original and smoothed camera motions to obtain a stabilized video. Experimental results are given to demonstrate the stabilization performance and the efficiency of the proposed algorithm.

Keywords: Video stabilization; Optical flow computation; Camera motion estimation; Robust estimation; Motion smoothing

1. Introduction

With the rapid growth of digital home video, video stabilization technique has been strongly demanded to alleviate the handshaking problem when taking video, which may cause unpleasant viewing experience. This problem is very common in non-professional home videos. The same problem occurs when the camera is mounted on a vehicle with unstable motion, especially moving on a rough surface. A number of methods have been proposed to reduce the undesirable camera motion perturbations for such videos [1–4]. Hansen et al. presented a video stabilization system that construct a stable view of the scene through aligning images of an incoming video stream and dynamically constructing an image mosaic. This system used a coarse-to-fine image registration method and was implemented on a video processing unit to achieve real-time performance. Morimoto and Chellappa [2] proposed to estimate the camera motion with a 2D rigid motion model in a pyramid structure based on tracking a set of feature points. Then they combined the sequential camera motion estimates between consecutive frames to obtain the global motion with

* Corresponding author. Fax: +886 3 5723694.
E-mail address: lai@cs.nthu.edu.tw (S.-H. Lai).

1047-3203/$ - see front matter
doi:10.1016/j.jvcir.2005.10.004

respect to a reference frame and to warp the current frame, thus producing a stabilized video. In addition, Duric and Rosenfeld [3] proposed an algorithm to stabilize image sequences acquired by a camera in a moving ground vehicle. They showed that only the rotational components of the non-smooth motion have significant perturbing effects in the video. Then they identified the images points with dominant rotational image flow and used such points to estimate the vehicle's rotation. Finally, they obtained smooth rotational motions from these estimates and the residual rotational motion can be used to stabilize the video. Litvin et al. [4] employed a probabilistic model for the camera motions and applied the Kalman filter to reduce the motion noises to obtain stabilized camera motions. Ratakonda [9] presented an integral projection matching method to convert the two-dimensional block matching problem into two one-dimensional vector matching problems to extract the global motion between adjacent frames. Then, they translated each image with respect to the previous image such that they followed the integrated motion trajectory to achieve video stabilization. Jin et al. [10] applied a pyramid-based matching algorithm to the original gray-scale images to extract the motion vectors from the chosen blocks in a frame. Subsequently, they used these motion vectors to determine an appropriate motion model from the three common camera motion models, i.e., dolly, H-tracking, and V-tracking. Finally, they estimate the camera motion of a specific motion type to remove the undesirable motion fluctuations from a dynamic video sequences. In addition, Chang et al. [11] first applied a zero-order optical flow technique by taking both change of brightness and departure from smoothness into consideration to obtain a moving flow field. By using the least-squares method, they derived a new scheme to determine the angular frequency, rotational center and translational motion. Then, the estimated motion parameters are used to compensate the undesirable motion. Etrurk [12,13] separated the full frame into four sub-images and made use of the peak amplitude value of phase correlation surfaces for local motion vectors of sub-images, thus the global motion vector could be computed with effective elimination of unreliable local motion vectors. Then the Kalman filter is employed to alleviate motion fluctuations in the original video.

Note that outliers are inevitable in the motion vector estimation. The outliers could greatly influence the accuracy of the camera motion estimation. Most of the above previous video stabilization methods are either not robust against outliers or computational inefficient. In this paper, we propose a real-time video stabilization algorithm that is robust against outliers in motion vector estimation and computationally efficient. The flow chart of our algorithm is depicted in Fig. 1. We first compute the optical flow vectors between consecutive frames based on a modified Lucas and Kanade's method. Then, the camera motion is estimated by fitting a simplified affine motion model to the computed optical flow vectors with a trimmed least squares algorithm. Subsequently, we smooth the camera motion parameters temporally from the computed camera motions in a regularization framework with the smoothness parameter determined recursively from the area percentage of undefined regions after the stabilization. Since the original and stabilized camera motions are available, we transform every frame of the original video with a corresponding affine transformation to produce a stabilized video.

The rest of this paper is organized as follows. In the next section, we describe a modified Lucas–Kanade algorithm for computing the motion vectors. In Section 3, the proposed robust camera motion estimation

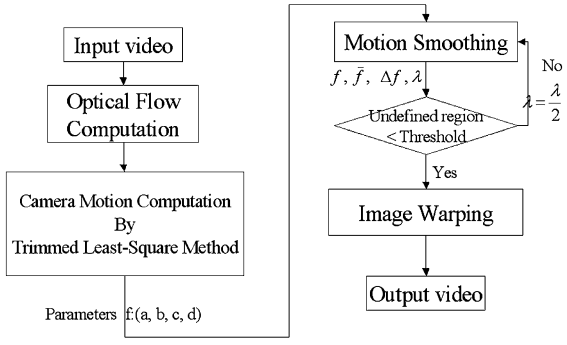

Fig. 1. Flow chart of the proposed video stabilization algorithm.

algorithm is presented. Subsequently, the smoothing of motion parameters is discussed. Then, we describe the motion compensation and image warping procedure in the proposed algorithm in Section 5. Some experimental video stabilization results by using the proposed algorithm are shown in Section 6. Finally, we conclude this paper in the last section.

2. Optical flow computation

Optical flow is the instantaneous motion vector at each pixel in the image frame at a time instant. This information is fundamental to many video analysis systems; including motion segmentation, image mosaic and video stabilization. Many approaches for optical flow computation have been proposed in the past. Among them, the gradient-based approach has been quite popular. The optical flow computation algorithm employed in this paper belongs to this camp.

For the sake of efficiency in the computation of optical flow, we first partition an image into smaller $n \times n$ non-overlapped blocks. If the block size is not too large, we can assume that the motion vectors in all pixels in the block are the same and the motion vector at the center is sufficient to approximate the motion of all pixels in the whole block. This assumption was originally used in the Lucas and Kanade's method for computing optical flow [7].

To account for temporal brightness variations, we used the generalized optical flow constraint under non-uniform illumination changes [5,6], instead of using the traditional image flow constraint that was derived from the brightness constancy assumption [7]. This generalized optical flow constraint can be written as

$$\frac{\partial I_0(x,y)}{\partial x}u + \frac{\partial I_0(x,y)}{\partial y}v + I_0(x,y)\cdot w + I_0(x,y) - I_1(x,y) = 0, \tag{1}$$

where w is a constant used for compensating the intensity variation between two corresponding points at consecutive frames.

Similar to the assumption made in Lucas and Kanade's optical flow computation, we assume the three unknowns $\hat{u}$, $\hat{v}$, and $\hat{w}$ to be constant in a local window. Then, we combine all the generalized optical flow constraints in this local window to estimate the three unknowns with the linear least squares method. The least-square estimation is to minimize the following energy function.

$$E(u,v,w) = \sum_{(x,y)\in W_{i,j}} \left(\frac{\partial I_0(x,y)}{\partial x}u + \frac{\partial I_0(x,y)}{\partial y}v + I_0(x,y)\cdot w + I_0(x,y) - I_1(x,y) \right)^2, \tag{2}$$

where $W_{i,j}$ is the local neighborhood window centered at the location (i,j). Note that the minimization of the above quadratic energy function leads to the least-square solution for the optical flow vector (u,v) and the illumination factor w. The least-square minimization is accomplished by solving an associated linear system.

$$\begin{bmatrix} \sum_{(x,y)\in W}\left(\frac{\partial I_0}{\partial x}\right)^2 & \sum_{(x,y)\in W}\frac{\partial I_0}{\partial x}\frac{\partial I_0}{\partial y} & \sum_{(x,y)\in W}\frac{\partial I_0}{\partial x}I_0 \\ \sum_{(x,y)\in W}\frac{\partial I_0}{\partial x}\frac{\partial I_0}{\partial y} & \sum_{(x,y)\in W}\left(\frac{\partial I_0}{\partial y}\right)^2 & \sum_{(x,y)\in W}\frac{\partial I_0}{\partial y}I_0 \\ \sum_{(x,y)\in W}\frac{\partial I_0}{\partial x}I_0 & \sum_{(x,y)\in W}\frac{\partial I_0}{\partial y}I_0 & \sum_{(x,y)\in W}I_0^2 \end{bmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \sum_{(x,y)\in W}\frac{\partial I_0}{\partial x}(I_1 - I_0) \\ \sum_{(x,y)\in W}\frac{\partial I_0}{\partial y}(I_1 - I_0) \\ \sum_{(x,y)\in W} I_0(I_1 - I_0) \end{pmatrix}. \tag{3}$$

Note that all partial derivatives, I_0 and I_1 in the above equation are functions of location (x,y). This least-square estimation is applied for each block in the partitioned image as mentioned before.

To alleviate the Taylor approximation error in the above optical flow constraint equation and get more accurate motion vectors, we have developed an iterative least-square optical flow estimation algorithm that refines the generalized optical flow constraints step by step. The main idea behind this iterative refinement process is to move the block in the image with the newly estimated motion vector and compute the updated flow constraints in a recursive manner. The updated optical flow constraint is given by

Fig. 2. Example of optical flow computation: (A) and (B) are two adjacent frames, (C) is the optical flow field between these two frames computed by the proposed algorithm.

$$\frac{\partial I_0(x+u^{(k)},y+v^{(k)})}{\partial x}\Delta u^{(k)}+\frac{\partial I_0(x+u^{(k)},y+v^{(k)})}{\partial y}\Delta v^{(k)}+(1+w^{(k)})I_0(x+u^{(k)},y+v^{(k)})-I_1(x,y)=0. \tag{4}$$

Thus, the residual optical flow vector $(\Delta u^{(k)},\Delta v^{(k)})$ and the illumination factor $w^{(k)}$ are computed by the above least-square estimation procedure with the above updated flow constraint.

Note that the Lucas and Kanade's optical flow computation method cannot provide reliable motion estimation in nearly homogeneous regions. This problem still exists in our optical flow computation method. To avoid this problem, we detect these homogeneous regions and skip the optical flow computation in these regions. The homogeneous regions can be detected by checking the following inequality

$$\left[\sum\left(\left|\frac{\partial I_0(x,y)}{\partial x}\right|+\left|\frac{\partial I_0(x,y)}{\partial y}\right|\right)<T\right],$$

where T is a user-defined threshold. If the sum of the absolute gradients in a block is small than a threshold T, then this block is declared as a homogeneous region and we will skip this block in the optical flow computation. More detail is described in a previous paper [8].

Fig. 2 depicts an example of optical flow computation. Figs. 2A and B show two successive frames and Fig. 2C is the corresponding result of optical flow computation by using the proposed algorithm. The upper-left parts of the two frames tally with the homogeneous region criterion, therefore no motion vectors are available in this region. In contrast, we can compute quite some optical flow vectors from the lower region.

This optical flow computational algorithm may not work well when the video motion is too large, thus leading to inaccurate camera motion estimation and unsatisfactory video stabilization. To alleviate such problem, we can incorporate this algorithm into a multiresolutional coarse-to-fine search strategy to increase the range of image registration between consecutive frames. With this coarse-to-fine strategy, this problem can be relieved to allow larger video motions.

3. Robust camera motion estimation

After the optical flow vectors are computed in the previous step, we estimate the camera motion from these optical flow vectors. Considering complexity and robustness, we employ a simplified affine motion model for the camera motion since it can represent scaling, rotation and translation all together. Then, we use the trimmed least squares algorithm to robustly fit the simplified affine motion from the computed optical flow vectors and simultaneously discard the outliers in the computed optical flow field. The outliers may be due to object motions or large errors in the optical flow computation. In is paper, we assume that the camera motion is the most dominant motion at every frame in the video.

3.1. Simplified affine motion model

The simplified affine motion is modeled by four parameters, i.e., a, b, c, and d, and it can be written as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix},$$

where the parameters a and b control the scaling and rotation, and the parameters c and d correspond to the translation. Now, assume we have N optical flow vectors with corresponding image positions (x_i, y_i) and (x_i', y_i'), for $i = 1, 2, \ldots, N$, at two adjacent frames, then we can estimate the simplified affine motion between these two frames from the N optical flow vectors by solving the following over-constrained linear system.

$$\begin{pmatrix} x_1 & -y_1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & -y_n & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ y_n & x_n & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} x_1' \\ \vdots \\ x_n' \\ y_1' \\ \vdots \\ y_n' \end{pmatrix}. \tag{5}$$

The solution to this over-constrained linear system, of the form $\mathbf{Ax} = \mathbf{b}$, can be obtained by using the standard least squares method, which is to solve the associated normal equation $(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$.

This simplified affine motion model is employed in both the camera motion estimation and image mosaic in the proposed video stabilization algorithm. Note that the use of the simplified affine motion model implicitly assumes the weak perspective camera projection the out-of-image-plane camera rotation is negligible. Although this camera motion model is not valid for videos with large 3D perspective effect, the proposed algorithm can still provide reasonable video stabilization results. Although the synthesis of the video stabilization may not be valid from the perspective of 3D projective geometry, the stabilized videos are suited for viewing purpose. In fact, almost all video stabilization methods made assumptions on camera motion and used some approximate camera projection models to estimate the camera motion and synthesize the stabilized videos.

The affine motion model has been frequently employed for video stabilization. The problem with the standard affine motion model is that the resulting shearing artifact for the image mosaic by using the estimated affine motion model could be undesirable. The shearing factor in the estimated affine motion is mainly due to two reasons. Firstly, the approximation of 3D camera motion by an affine model may lead to non-trivial and undesirable shearing component. Second, the computed optical flow vectors available for affine motion estimation may be sparse or unevenly distributed, as shown in Fig. 2C, thus leading to inaccurate affine motion estimation accompanied with shearing effect. Therefore, we propose to use the simplified model to remove the shearing artifact in the synthesis of stabilized image frames. Although some previous stabilization methods employed more accurate models for camera motion estimation, they require more computational efforts in motion estimation and the results are more sensitive to errors in motion vector estimation. When the computed motion vectors available for camera motion estimation are sparse and non-uniformly distributed, the resulting camera motion estimation tend to cause undesirable synthesized image frames. In practice, the simplified affine motion model turns out to provide more reliable camera motion estimation for image mosaic.

3.2. Trimmed least squares method

Note that the standard least-squares solution to the above over-constrained system is sensitive to outliers. The computed optical flow vectors may be due to camera motion or object motions. If we use all the computed optical flow vectors to estimate the global motion, this may lead to inaccurate camera motion estimation and thus the final stabilization result will be degraded. To discard the less dominant object motions from the fitting of the simplified affine motion, we employ a trimmed least squares method to achieve robust camera motion estimation. Most previous works on video stabilization did not apply any robust estimation technique to com-

pute the camera motion. A representative robust camera motion estimation algorithm was presented by Bouthemy et al. [14]. In their method, the camera motion was estimated by fitting a global motion model to the computed optical flow vectors by using robust M-estimation. In this paper, we propose a new, efficient and robust camera motion estimation algorithm by using trimmed least square fitting of the simplified affine motion model.

In the trimmed least squares algorithm as shown in the flow chart in Fig. 3, we first apply the standard least squares method to estimate the initial simplified affine motion parameters (a, b, c, and d) and then compute the fitting errors

$$EX_i = ax_i - by_i + c - x'_i,$$
$$EY_n = ay_i + bx_i + d - y'_i$$

for every computed optical flow vector. By collecting the error statistics from all the data points, we use the estimated standard derivation to identify the data points with large errors as the outliers and discard them from the data set for model fitting. Then we repeat the least squares fitting of the simplified affine motion model with the trimmed data point set and reject outliers with the updated motion parameters. This process is repeated for several iterations until there is no new outlier being identified. In the flow chart, if the remaining motion vectors in iteration n, denoted by RMVn, is equal to the remaining motion vectors in iteration $n-1$, i.e., RMV$n-1$, then there is no new outlier founded. Thus, the final converged simplified affine motion parameters can provide a robust estimation for the camera motion.

Fig. 4 depicts an example of the camera motion estimation with and without trimmed least-squares method. Figs. 4A and B show two consecutive frames in a video sequence. In this example, there exist four different global motions in the video. The camera motion is the most dominant motion, and the multiple moving objects include the person with black suit, the person with brown suit, and the static caption in the lower region. Fig. 4C depicts the computed optical flow vectors, Fig. 4D shows the trimmed optical flow vectors for estimating the camera motion, and the estimated simplified affine motion field is displayed in Fig. 4E. It is evident from this example that the camera motion estimation with trimmed least squares method can successfully remove foreground optical flow vectors. Note that the optical flow vectors cannot be computed in the homogeneous regions, thus leaving them blank in Figs. 4C and D.

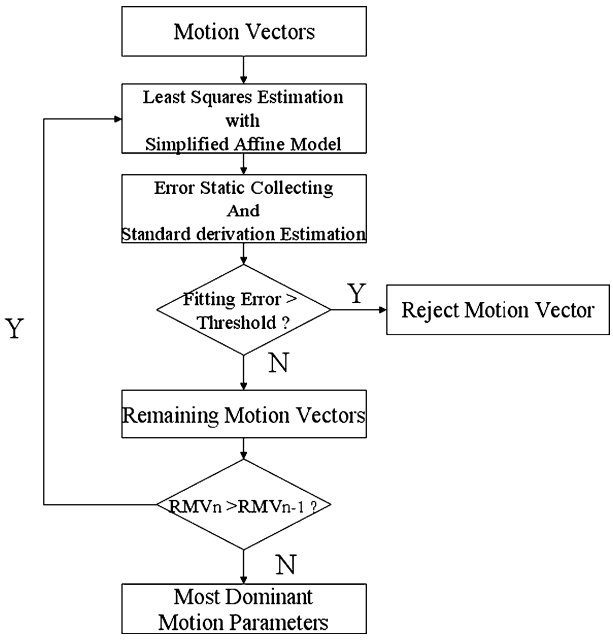

Fig. 3. Flow chart of the trimmed least squares camera motion estimation method.

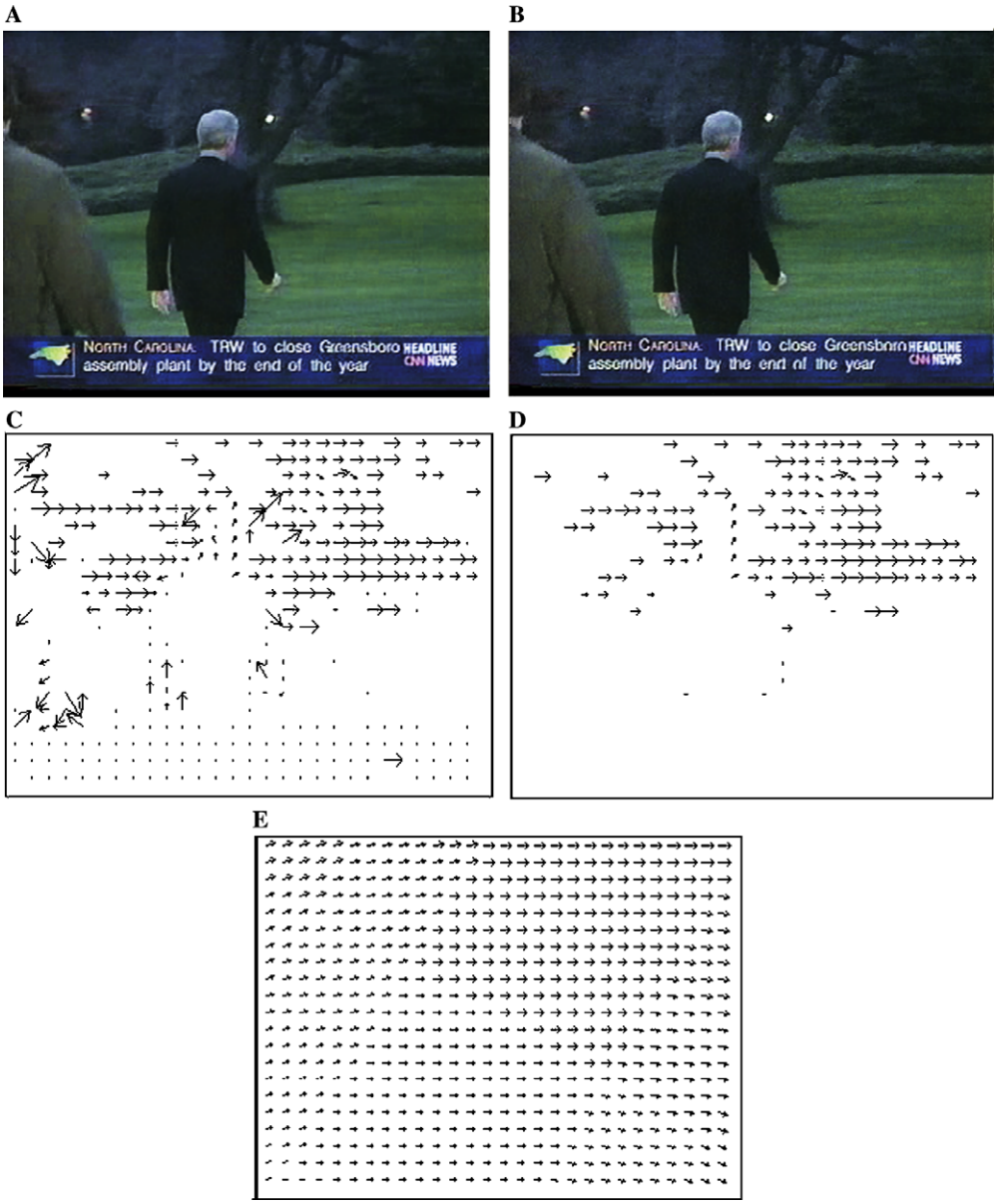

Fig. 4. Optical flow computation result from (A) the first frame, (B) the second frame, (C) the computed optical flow vectors, (D) the trimmed optical flow vectors used for camera motion estimation, and (E) the estimated affine camera motion.

4. Smoothing of camera motion parameters

To alleviate the undesirable effect due to camera motion perturbations, we need to have a motion smoothing step to temporally smooth the camera motions for the whole video. To achieve this goal, we take two issues into consideration. The first issue is about the problem of undefined regions. To reduce the amount of undefined regions in the stabilized video, the difference between the accumulated original simplified affine motion parameters f and the accumulated smoothed affine motion parameters $\bar{f}$ can not be too large. For the second issue, because we want to produce a stable video sequence, the temporal variations of the smoothed affine camera motion parameters cannot be too large. There exists a tradeoff between the amount of undefined regions and the degree of motion smoothing. Therefore, we employ a regularization framework for temporal

smoothing. In this regularization, there is a regularization parameter λ that directly determines the degree of motion smoothness. If λ is large, it will lead to a stabilized video with very smooth camera motion, but the total amount of undefined regions may also be large. Therefore, it is important to select an appropriate value for λ to obtain a compromised solution.

The regularization approach for data smoothing involves minimizing a cost function that consists of two parts. The first part of the cost function is the penalty of data deviations, which can be written as

$$P = \sum_{i=1}^{n} (f_i - \bar{f}_i)^2.$$

The second part of the cost function corresponds to the temporal motion smoothness constraint given by

$$Q = \sum_{i=2}^{n} (\bar{f}_i - \bar{f}_{i-1})^2.$$

So the total cost function is written as

$$E = P + \lambda Q, \tag{6}$$

where $f_i = (a_i, b_i, c_i, d_i)$, and $\bar{f}_i = (a'_i, b'_i, c'_i, d'_i)$ is the numbers of simplified affine motion parameters. Since the unknowns are the $\bar{f}_i$ parameter vectors, the minimization of the cost function leads to solving the following linear system:

$$\begin{bmatrix} \lambda+1 & -\lambda & 0 & \cdots & 0 \\ -\lambda & 2\lambda+1 & -\lambda & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\lambda & 2\lambda+1 & -\lambda \\ 0 & \cdots & 0 & -\lambda & \lambda+1 \end{bmatrix} \begin{bmatrix} \bar{f}_1 \\ \bar{f}_2 \\ \vdots \\ \vdots \\ \bar{f}_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix}. \tag{7}$$

After solving this linear system by LU decomposition, we can obtain all the smoothed affine motion parameters. Then, we can obtain the relation between the original and the stabilized videos. Therefore, we can calculate the percentage of the pixels without reference to the frames in the original video. If any of the stabilized frames have the percentage (P) larger than a user-defined threshold T, we reduce the regularization parameter λ by half and repeat the motion smoothing until all the P values are less than T.

Note that the above regularization-based parameter smoothing can be done for a fixed period of time. However, this will cause some time delay to produce the stabilized video output. This delay may be acceptable for video post-processing. For some applications that require immediate output of the stabilized video frame as the current frame is received, we can apply Kalman filter for smoothing the motion parameters to achieve better real-time performance [13].

5. Motion compensation and image warping

In Fig. 5, $I_1, I_2, \ldots$, and I_n represent the original images in the video, and $\bar{I}_1, \bar{I}_2, \ldots,$ and $\bar{I}_n$ denote the corresponding stabilized frames, f_i is the simplified affine transformation parameter vector that associates the camera motion from I_i to I_{i+1}, $\bar{f}_i$ is the corresponding smoothed affine parameter vector and Δf_i indicates the simplified affine parameter vector associated with the affine transformation from the original frame to the stabilized frame. The stabilized frame can be obtained by warping the corresponding original frame with the parameter vector Δf. The simplified affine transformation associated with the parameter vector f_i can be written as

$$A(f_i) = \begin{bmatrix} -b_i & a_i & d_i \\ a_i & b_i & c_i \\ 0 & 0 & 1 \end{bmatrix}.$$

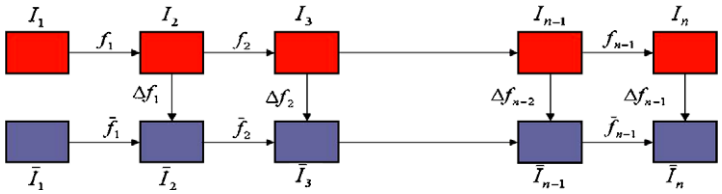

Fig. 5. Relation between the original and stabilized frame.

The above camera transformation between successive frames can be extended to get the accumulated camera transformation between the nth frame and the first frame as follows:

$$\prod_{i=1}^{n-1} A(f_i).$$

Note that this accumulated transformation is still a simplified affine transformation. Similarly, the camera motion transformation between the nth stabilized frame and the first frame can be written as

$$\prod_{i=1}^{n-1} A(\bar{f}_i).$$

Therefore, the camera transformation between the original input frame I_k and the corresponding stabilized frame $\bar{I}_k$ can be derived from the following equation.

$$\prod_{i=1}^{k-1} A(f_i) \cdot A(\Delta f_{k-1}) = \prod_{i=1}^{k-1} A(\bar{f}_i).$$

Thus, it can be determined as follows:

$$A(\Delta f_{k-1}) = \left(\prod_{i=1}^{k-1} A(f_i) \right)^{-1} \prod_{i=1}^{k-1} A(\bar{f}_i).$$

In the image warping of the stabilized frames, we apply the affine transformation with the simplified affine parameters Δf determined above from the original frame to produce the corresponding stabilized frame. The relation between the original frame I_n and the stabilized frame $\bar{I}_n$ can be described as follows:

$$\bar{I}_n(x, y, 1) = I_n(A(\Delta f_{n-1})(x, y, 1)^T).$$

Note that the warping from I_n to $\bar{I}_n$ in the above equation requires image interpolation. We can simply use the nearest-neighbor interpolation, since the result is not much different from that of the bilinear interpolation and its execution time is much less than that of the bilinear interpolation.

6. Experimental results

In this section, we show some experimental results of using the proposed algorithm from the videos Road4b, Road5, and scene_original. The first two videos are used in [10]. We have put all the original and stabilized videos described in this section on the web page http://cv.cs.nthu.edu.tw/research/ICME/. In Fig. 6, we show the original temporal simplified affine motion parameters for the video Road4b and the smoothed motion parameters with $\lambda = 1$, 10, 500, and 1000. We use 30% as the threshold for the undefined region percentage. In our experiments, the smoothing parameter λ is set to 500.

The videos road4_1, road4_10, and road4_1000 show the stabilized videos with the smoothing parameter $\lambda = 1$, 10, and 1000. From these videos, we can see that the global motions of the stabilized videos get smoother with larger values of the smoothing parameter λ while the undefined regions become larger.

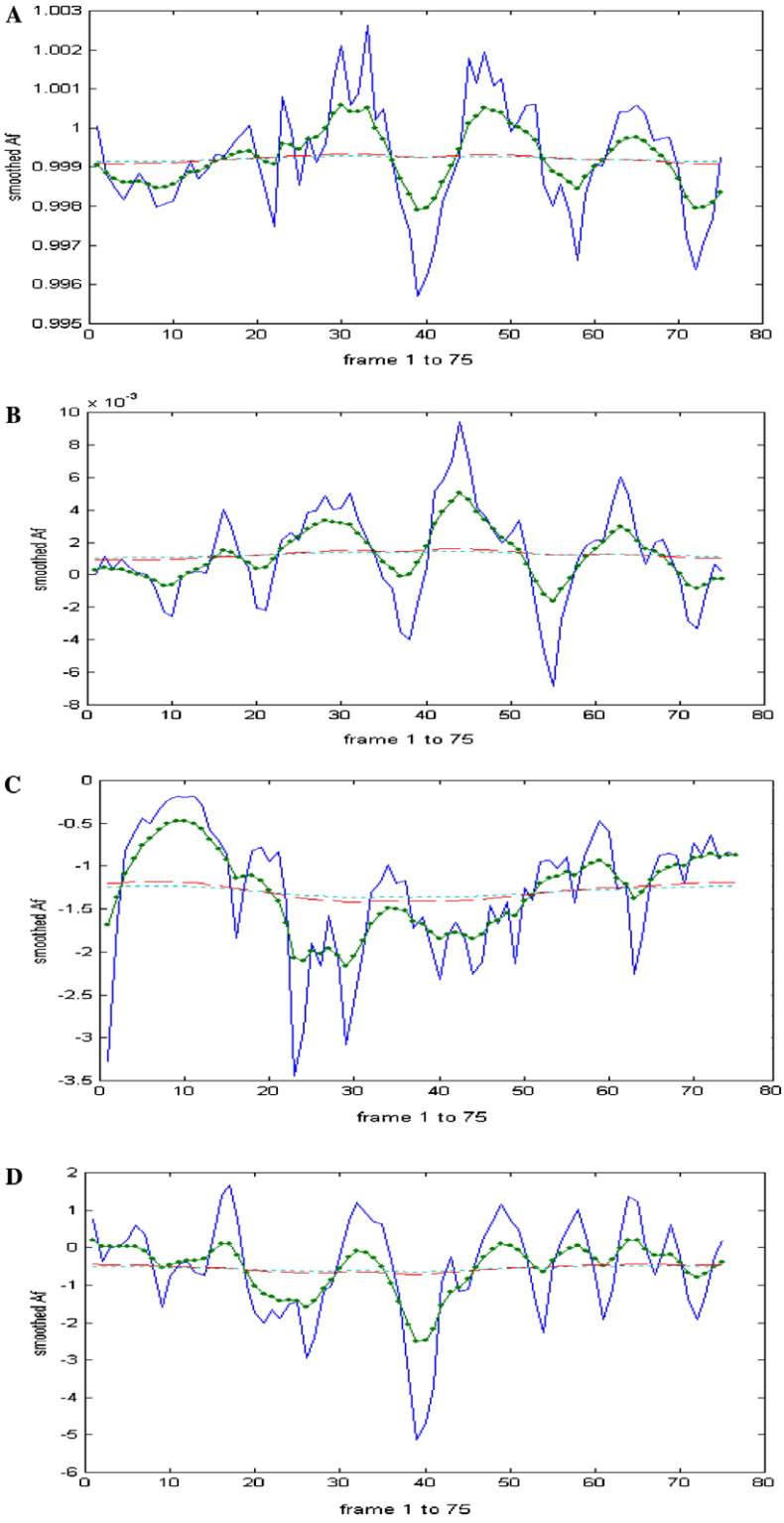

Fig. 6. The original and smoothed temporal curves of the simplified affine motion parameters A, B, C, and D (from top to bottom) for the video Road4b are shown with the smoothing parameter taking the values 1, 10, 500, and 1000. The solid, dashdot, dashed, and dotted curves correspond to the smoothing parameter values 1, 10, 500, and 1000, respectively.

Fig. 7. Frames 15, 42, 55, and 70 from (A) the original video Road4b, (B) the stabilized video by using the proposed algorithm, and (C) the stabilized video with the standard least-square method for global motion estimation.

Fig. 7 depicts some frames from the original video Road4b and the corresponding frames from the stabilized videos Road4b_nt and Road4bs, which are obtained with the standard least square method for the camera motion estimation and the proposed algorithm with the trimmed least squares method for the camera motion estimation, respectively. By checking the fixed reference red lines overlaid on the images in Fig. 7, it is obvious that the stabilized video is more stable than the original video and the proposed algorithm with trimmed least squares camera motion estimation provides better stabilization results than the one with standard least squares estimation.

Fig. 8 shows the temporal variations of the simplified affine parameter d for the three videos depicted in Fig. 7. This affine parameter accounts for the primary camera motion in this example. The blue solid, green dash-dot, and red dashed curves shows the temporal variations of the affine parameter d in the original video, in the stabilized video by using the proposed algorithm and in the stabilized video with the standard least squares motion estimation. It is evident from this figure that the temporal parameter variation with the proposed algorithm with the trimmed least-squares motion estimation is smoother than the others.

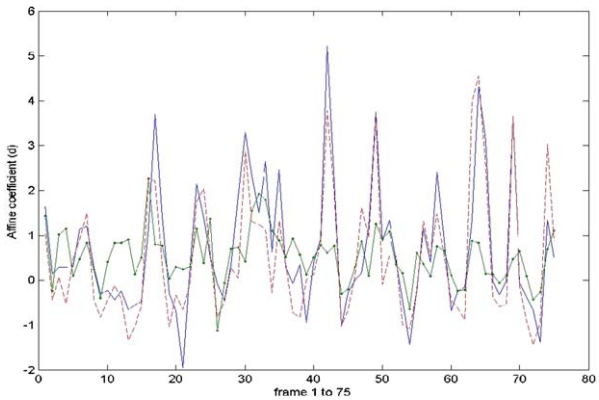

Fig. 8. Comparison of the stabilization experiments on the video road4b: the blue solid, green dash-dot, and red dashed curves shows the temporal variations of the affine parameter d in the original video, in the stabilized video by using the proposed algorithm and in the stabilized video with the standard least squares motion estimation.

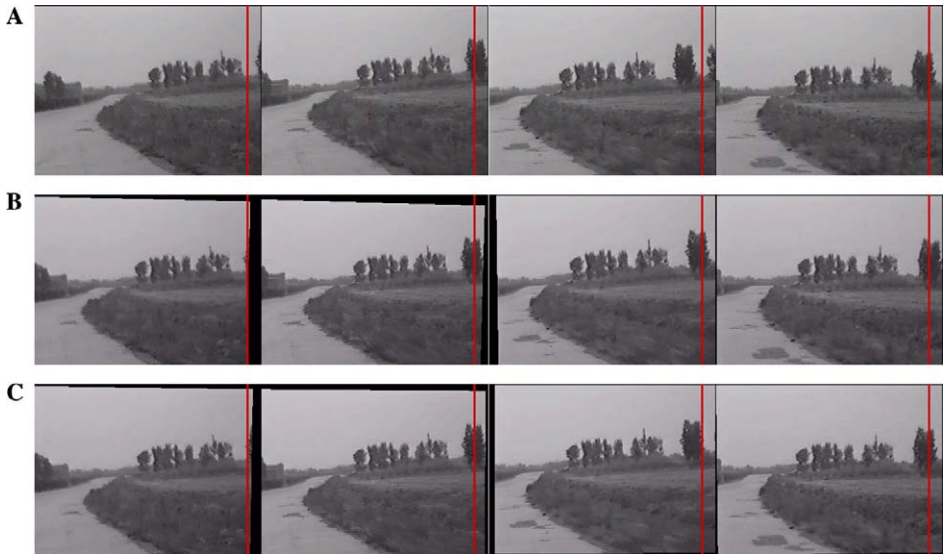

Fig. 9. Frames 20, 28, 57, and 73 from (A) the original video Road5, (B) the stabilized video by using the proposed algorithm, and (C) the stabilized video with the standard least-square camera motion estimation.

In the second example, Fig. 9 depicts representative frames from the original video Road5, the corresponding frames from the stabilized video, Road5_nt, with the standard least square camera motion estimation and the corresponding frames from the stabilized video Road5s with the proposed algorithm. From Fig. 9A, we can see clearly the original video oscillates horizontally compared to the fixed reference red line overlaid on the image. The oscillation is reduced by the stabilization method with the standard least square camera motion estimation as depicted in Fig. 9C. The proposed algorithm can achieve the best stabilization result, which is evident from Fig. 9B.

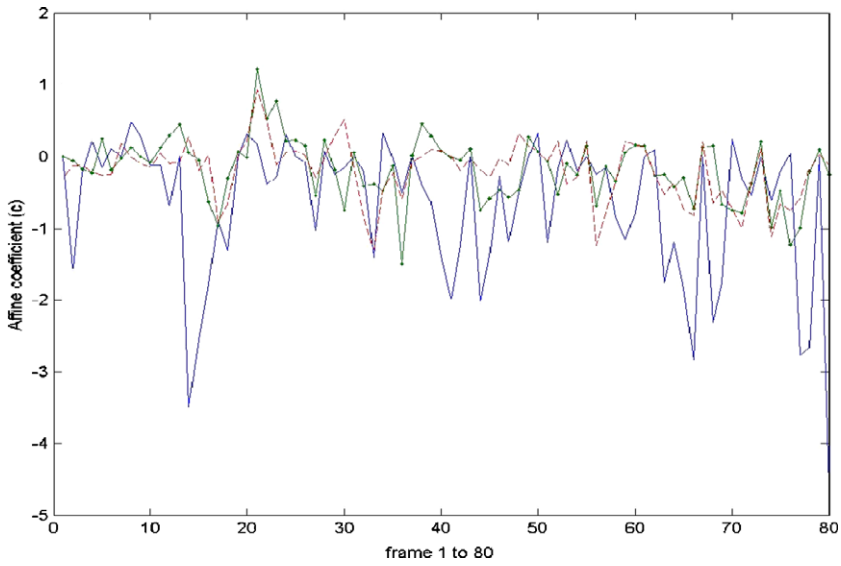

Fig. 10. The temporal curves of the horizontal translational motion parameter from the original video and the stabilized videos. The blue solid curve corresponds to the original video Road5, the red dashed curve is obtained from the stabilized video Road5b_nt with the standard least squares camera motion estimation, and the green curve corresponds to the video Road5s, which is stabilized by the proposed algorithm with the trimmed least squares camera motion estimation.

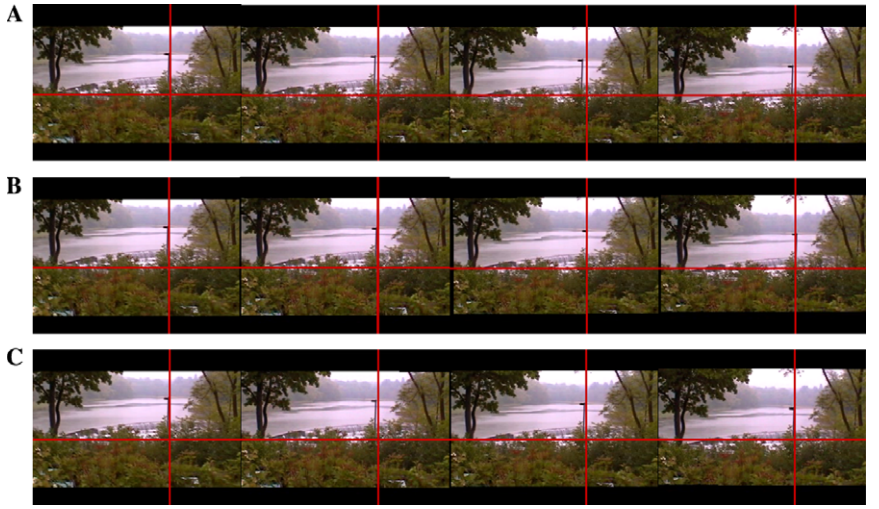

Fig. 11. Frames 1, 10, 16, and 25 from (A) the original video scene_aa, (B) the stabilized video by using the proposed algorithm, and (C) the stabilized video with the standard least-square method for global motion estimation.

In this example, the main camera fluctuations are in the horizontal translational motion, which corresponds to the simplified affine motion parameter c. Fig. 10 shows the temporal variations of this parameter in the original video road5 and the corresponding stabilized videos by using the proposed algorithm with the trimmed least squares camera motion estimation and the standard least squares motion estimation. By comparing the curves in the figure, we can see clearly that the stabilized video obtained by using the proposed algorithm has the smoothness temporal motion parameter variations.

The third example is the experiment on the video scene_original. Some frames of this video are shown in Fig. 11A, and the corresponding stabilized frames by using the proposed algorithm with the trimmed least-squares motion estimation and the standard least-squares motion estimation are given in Figs. 11B and C. From the fixed reference lines, we can see that the proposed algorithm with the trimmed least squares motion estimation provides more stable video than the others.

Fig. 12 gives the temporal variations of the estimated affine camera motion parameters corresponding to horizontal and vertical translational components from the original unstable video scene_aa and the stabilized videos. By comparing these temporal variations of the crucial camera motion parameters, we can see very clearly that the proposed video stabilization algorithm once again provides the most stable and smooth temporal trajectories of camera motion parameters in this example.

Regarding the execution speed of the proposed stabilization algorithm, it took 2.5 s for all the operations required for stabilizing the video read4b, which contains 76 frames of size 320×240, on a Pentium IV 2.4G PC with 248Mb RAM running Windows XP. This execution speed can process 30 frames per second.

7. Conclusion

In this paper, we presented a robust and efficient video stabilization algorithm. Our algorithm consists of an efficient optical flow computational procedure, a robust trimmed least squares camera motion estimation method, a regularization-based motion smoothing scheme, and an affine image warping step. The optical flow computational method is based on a modified optical flow constraint derived from the generalized brightness constancy assumption that allows smooth brightness variations along time. The trimmed least squares camera motion estimation can robustly determine the simplified affine camera motion parameters from the computed optical flow vectors. Then we employ a regularization method to temporally smooth the estimated camera motion parameters to be the stabilized camera motion. Finally, the stabilized video is obtained via affine image warping from the original video. Some experimental results were shown to demonstrate the robust and efficient video stabilization performance of the proposed algorithm.

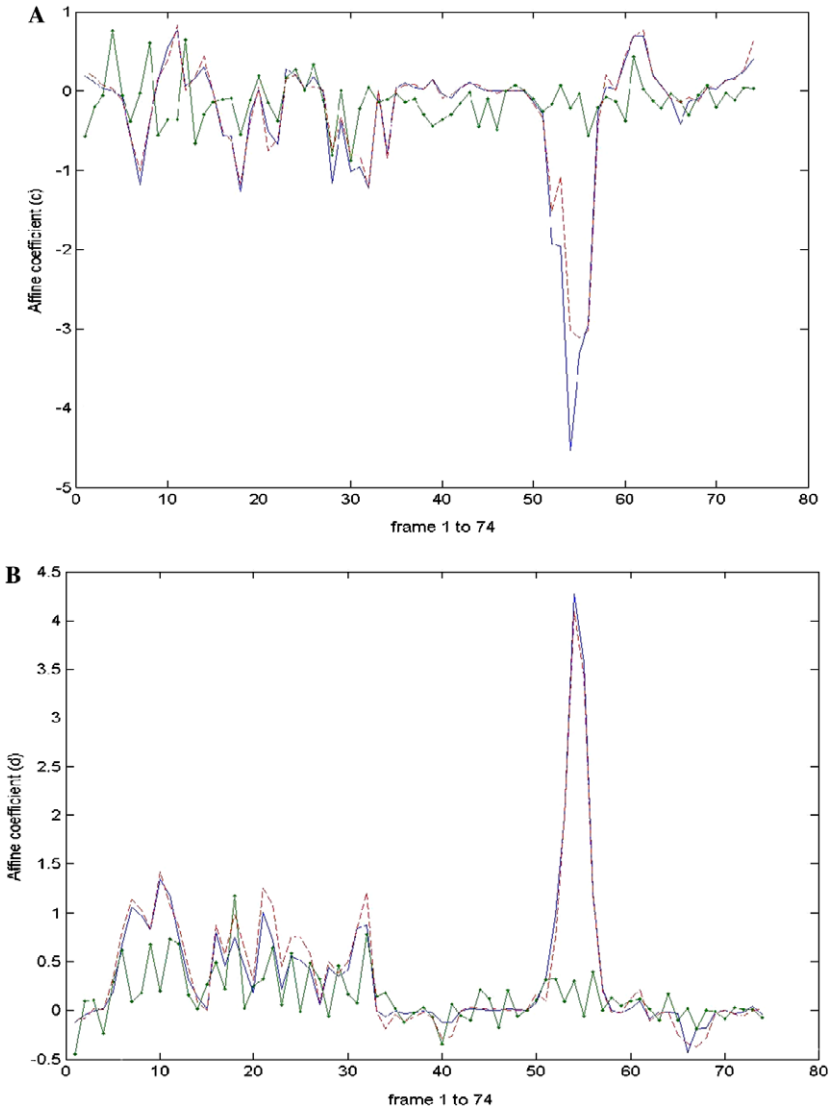

Fig. 12. The temporal variations of the (A) horizontal and (B) vertical translational components in the affine motion estimation from the original video scene_aa, the stabilized videos scene_nt and scene_as are displayed by the blue solid curves, red dashed curves and green solid curves, respectively. Note that the stabilized video scene_as is obtained by using the proposed stabilization algorithm with trimmed least squares motion estimation, while the video scene_nt is obtained by the same stabilization algorithm except using the standard least squares motion estimation.

Acknowledgments

This research was jointly supported by Ulead Systems and the Chinese Multimedia Information Extraction project funded by Academia Sinica, Taipei, Taiwan.

References

[1] M. Hansen, P. Anadan, K. Dana, G. van de Wal, P. Burt, Real-time scene stabilization and mosaic construction, in: Proceedings of the IEEE Conference Computer Vision and Pattern Recognition, 1994, pp. 54–62.

[2] C. Morimoto, R. Chellappa, Fast electronic digital image stabilization for off-road navigation, Real Time Imaging 2 (5) (1996) 285–296.

[3] Z. Duric, A. Rosenfeld, Image sequence stabilization in real time, Real Time Imaging 2 (1996) 271–284.
[4] A. Litvin, J. Konrad, W.C. Karl, Probabilistic video stabilization using Kalman filtering and mosaicking, in: IS&T/SPIE Symposium on Electronic Imaging, Image and Video Communications and Proc., 2003. pp. 20–24.
[5] A. Nomura, Spatio-temporal optimization method for determining motion vector fields under non-stationary illumination, Image Vis. Comput. 18 (2000) 939–950.
[6] S.-H. Lai, W.-K. Li, New video shot change detection algorithm based on accurate motion, and illumination estimation, Proc. SPIE: Storage and Retrieval for Media Databases 4676 (2002) 148–157.
[7] B.D. Lucas, and T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proc. Intern. Joint Conf. on Artificial Intelligence, 1981, pp. 674–679.
[8] J.H. Kuo, J.L. Wu, An efficient algorithm for scene change detection and camera motion characterization using the approach of heterogeneous video transcoding on MPEG compressed videos, Proc. SPIE: Storage and Retrieval for Media Databases 4676 (2002) 168–176.
[9] K. Ratakonda, Real-time digital video stabilization for multimedia applications, in: Proceedings of the 1998 IEEE International Symposium on Circuit and System, vol. 4, 31 May–3 June 1998, pp. 69–72, 31.
[10] J.S. Jin, Z. Zhu, G. Xu, Digital video sequence stabilization based on 2.5D motion estimation and inertial motion filtering, Real Time Imaging 7 (4) (2001) 357–365.
[11] J.-Y. Chang, W.-F. Hu, M.-H. Chen, B.-S. Chang, Digital image translational and rotational motion stabilization using optical flow technique, IEEE Trans. Consum. Electron. 48 (1) (2002) 108–115.
[12] S. Etrurk, Digital image stabilization with sub-image phase correlation based global motion estimation, IEEE Trans. Consum. Electron. 49 (4) (2003) 1320–1325.
[13] S. Etrurk, Real-time digital image stabilization using Kalman filters, Real Time Imaging 8 (2002) 317–328.
[14] P. Bouthemy, M. Gelgon, F. Ganansia, A unified approach to shot change detection and camera motion characterization, IEEE Trans. Circuits Syst. Video Technol. 9 (7) (1999).