A robust and efficient video stabilization algorithm

Conference Paper · July 2004

DOI: 10.1109/ICME.2004.1394117 · Source: IEEE Xplore

3 authors, including:

Shang-Hong Lai
National Tsing Hua University
276 PUBLICATIONS **2,723** CITATIONS

SEE PROFILE

A Robust and Efficient Video Stabilization Algorithm

Hung-Chang Chang[1], Shang-Hong Lai[1], and Kuang-Rong Lu[2]
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[2]Ulead Systems, Inc., Taipei, Taiwan

Abstract

The acquisition of digital video usually suffers from undesirable camera jitters due to unstable random camera motions, which are produced by a hand-held camera or a camera in a vehicle moving on a non-smooth road or terrain. In this paper, we propose a real-time robust video stabilization algorithm to remove undesirable jitter motions and produce a stabilized video. In our algorithm, we first compute the optical flows between successive frames, followed by estimating the camera motion by fitting the computed optical flow field to a simplified affine motion model with a trimmed least squares method. Then the computed camera motions are smoothed temporally to reduce the motion vibrations by using a regularization method. Finally, we transform all frames of the video based on the original and smoothed 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.

1. Introduction

In the past decade, the video stabilization has been widely demanded to remove the uncomfortable video motion vibrations, which are common in non-professional home video taking. A number of methods have been proposed to reduce the irregular motion perturbations from the video. Morimoto and Chellappa [1, 2] proposed to use a 2D rigid motion model in combination with a pyramid structure to compute the motion vectors with sub-pixel accuracy. Then they used a global motion model such as the affine motion model to represent rotational and translational camera motions. In addition, Chen [3] used a block-based motion estimation in conjunction with an affine motion model to compute the camera motion model from consecutive frames, followed by smoothing the sets of affine motion parameters along time. 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.

In this paper, we propose a robust and efficient video stabilization algorithm. The flow chart of our algorithm is shown in Figure 1. The optical flow between consecutive frames is computed based on modification of the Lucas and Kanade's method. Then, the camera motion is estimated by fitting a simplified affine motion model to the computed optical flow vectors by using a trimmed least squares algorithm. Subsequently, we smooth the camera motion parameters from the computed camera motion 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. For the rest of this paper, we describe the components in the flow diagram in details in the following sections. In addition, we show some experimental results of using the proposed video stabilization algorithm. Finally, we give a conclusion for this paper.

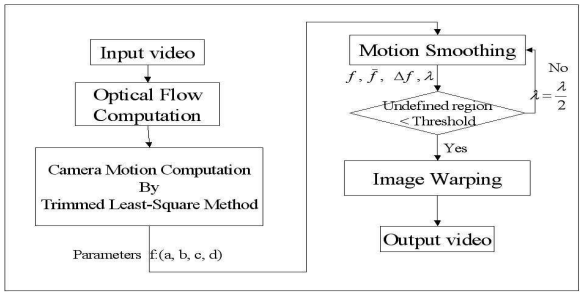


Figure 1. Flow Chart of Video Stabilization Algorithm

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-by-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

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

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 $\hat{w}$ be constants 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.

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

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.

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

$$\frac{\partial I_0(x+u^{(k)},y+v^{(k)})}{\partial x}\Delta u^{(k)}+\frac{\partial I_0(x+u^{(k)},y+v^{(k)})}{\partial y}\Delta v^{(k)} + (1+w^{(k)})I_0(x+u^{(k)},y+v^{(k)})-I_1(x,y)=0 \quad (3)$$

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 can't 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 condition $\left[\sum\left(\left|\frac{\partial I_0(x,y)}{\partial x}\right|+\left|\frac{\partial I_0(x,y)}{\partial y}\right|\right)<T\right]$ 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].

3. Robust Camera Motion Estimation

After the motion vectors are computed from the previous step, we estimate the camera motion from these motion vectors. Considering complexity and accuracy, we employ a simplified affine motion model for the camera motion since it can handle scaling, rotation and translation all together. Then, we use the trimmed least squares algorithm to robustly fit the simplified affine motion from the computed motion vectors by discarding the outliers of the motion vectors. In our algorithm, we assume that the most dominant motion in every frame is the camera motion.

3.1 Simplified Affine Motion Model

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

$$\begin{bmatrix}x'\\y'\end{bmatrix}=\begin{bmatrix}a & -b\\ b & a\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}c\\d\end{bmatrix} \quad (4)$$

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

$$\begin{pmatrix} x_1 & -y_1 & 1 & 0\\ \vdots & \vdots & \vdots & \vdots\\ x_n & -y_n & 1 & 0\\ y_1 & x_1 & 0 & 1\\ \vdots & \vdots & \vdots & \vdots\\ y_n & x_n & 0 & 1\end{pmatrix}\begin{pmatrix}a\\b\\c\\d\end{pmatrix}=\begin{pmatrix}x_1+u_1\\ \vdots\\ x_n+u_n\\ y_1+v_1\\ \vdots\\ y_n+v_n\end{pmatrix} \quad (5)$$

3.2 Trimmed Least Squares Method

Note that the traditional least-squares solution of the above over-constrained system is sensitive to outliers. It is common that the computed optical flow vectors are resulted from camera motion and object motions. To discarding 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. In the trimmed least squares algorithm, we first apply the least squares estimation and then compute the fitting errors $U_n'=ax_n-by_n+c-x_n',V_n'=ay_n+bx_n+d-y_n'$ for every data points. 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 the outliers with updated motion parameters. This process is repeated for some iterations until there is no new outlier being identified. Thus, the final converged motion parameters can well represent the camera motion.

4. Motion Smoothing

To alleviate the uncomfortable effect due to camera motion perturbations, we need to have a motion-smoothing step to temporally smooth out the camera motions for the whole video. To achieve this goal, we take two issues into consideration. The first issue is about the undefined region problem. 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 smooth video sequence, the temporal variations of the smoothed affine motion parameters can't be too large. There usually 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 larger, the stabilized video will have very smooth camera motion, but the total amount undefined regions will also be larger. Therefore, it is important to select a value for λ appropriately.

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, which can be written as

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

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.

5. Motion Compensation and Image Composition

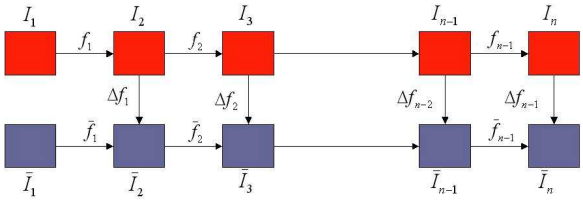


Figure 2. Relation between original & stabilized frame

In Figure 2, I_1, I_2, …, and I_n represent the original images in the video, and $\bar{I}_1$, $\bar{I}_2$, …, and $\bar{I}_n$ represent the corresponding stabilized frames. The stabilized output frame can be obtained by warping the corresponding original frame. Let the affine transformation associated with the simplified affine parameter vector f_i can be written as

$$A(f_i) = \begin{bmatrix} -b_i & a_i & d_i \\ a_i & b_i & c_i \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

and the relation between successive frames can be written as follows:

$$I_n(x', y') = I_{n-1}(A(f_{i-1})(x, y, 1)^T) \tag{9}$$

The relation between the original input frame I_k and corresponding output frame $\bar{I}_k$ can be derived from the following.

$$\prod_{i=1}^{k} A(f_i) \bullet A(\Delta f_k) = \prod_{i=1}^{k} A(\bar{f}_i)$$
$$A(\Delta f_k) = \left(\prod_{i=1}^{k} A(f_i) \right)^{-1} \prod_{i=1}^{k} A(\bar{f}_i) \tag{10}$$

In the image warping of the stabilized frames, we apply the affine transformation with the affine parameters Δf determined above to the original frame to produce the stabilized frame. The backward projection scheme is employed with the nearest-neighbor interpolation for the image warping. The result is not much different from that of the bilinear interpolation but the execution time is much less than the bilinear interpolation.

6. Experimental Result

In this section, we show some experimental results of using the proposed algorithm. In Figure 3, we show the original temporal motion parameters and the smoothed motion parameters with λ=1, 10, 500,

1000. We use 30% as the threshold for the undefined region percentage. In our experiment, the value of λ is chosen to be 500.

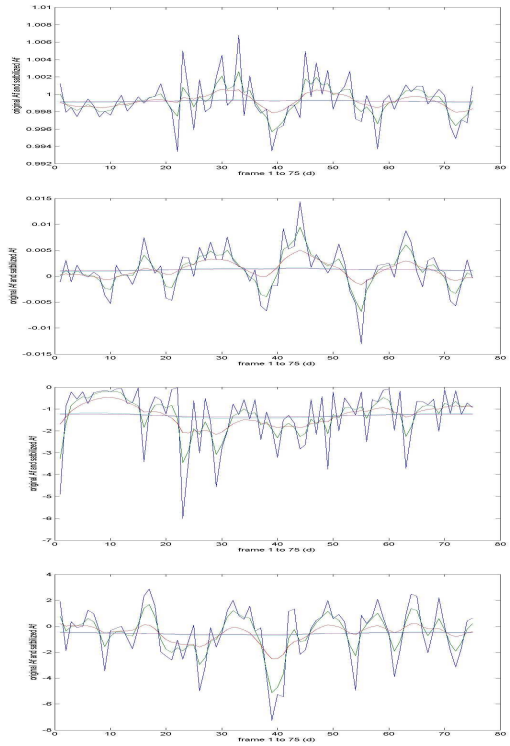
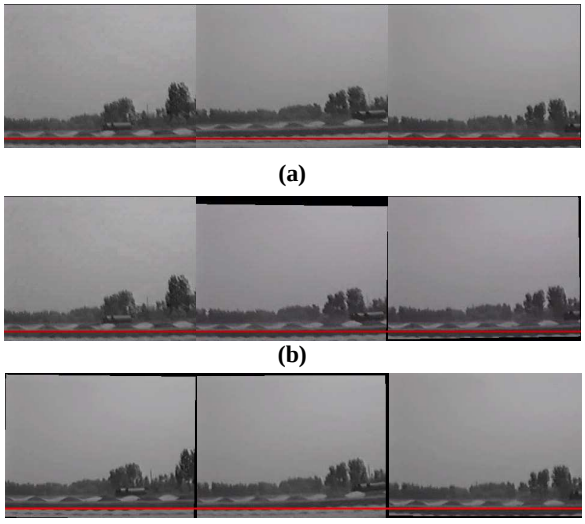


Figure 3. The original and smoothed temporal curves of the motion parameters a, b, c, and d (from top to bottom) are given with smoothing parameters taking the values 1, 10, 500, and 1000.

Figure 4 depicts frames 1, 42, and 70 from the original video Road4b, the corresponding frames from the stabilized video without trimmed least square method and the corresponding frames from the stabilized video Road4bs. By checking the red lines overlaid on the images, the video via stabilization step is smoother than the one without stabilization step. we can see the stabilization performance of our stabilized video.



(a)

(b)

(c)

Figure 4. Frames 1, 42, and 70 from (a) the original video, (b) the stabilized video by using the proposed algorithm, (c) the stabilized video with the standard least-square method for global motion estimation.

7. Conclusion

In this paper, we presented a robust and efficient video stabilization algorithm. Our algorithm consists of using a trimmed least squares algorithm to estimate the camera motions from the computed optical flow vectors and a regularization approach to motion smoothing. Finally, the stabilized video is obtained by affine warping from the original video. Our stabilization algorithm is experimented on Pentium IV 2.4GHz PC and it can achieve real-time performance.

Acknowledgements

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

References

[1] C. Morimoto and R. Chellappa, “Automatic digital image stabilization,” IEEE Intern. Conf. on Pattern Recognition, pp. 660-665, 1997.

[2] C. Morimoto and R. Chellappa, “Fast electronic digital image stabilization for off-road navigation,” Proc. of the 13th Intern. Conf. on Pattern Recognition, vol. 3, pp. 284-288, August 1996.

[3] T. Chen, “Video stabilization algorithm using a block-based parametric motion model”, EE392J Project Report, Winter 2000.

[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., Jan. 20-24, 2003.

[5] A. Nomura, “Spatio-temporal optimization method for determining motion vector fields under non-stationary illumination,” Image and Vision Computing 18**,** pp. 939-950, 2000.

[6] S.-H. Lai and W.-K. Li, “New video shot change detection algorithm based on accurate motion, and illumination estimation,” Proc. SPIE: storage and retrieval for media databases, vol. 4676, pp. 148-157, 2002.

[7] B. D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” Proc. IJCAI81, pp. 674-679, 1981.

[8] J. H. Kuo and 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, vol. 4676, pp.168-176, 2002.