VIDEO STABILIZATION FOR A HAND-HELD CAMERA BASED ON 3D MOTION MODEL

J. M. Wang[1], H. P. Chou[3], S. W. Chen[2], and C. S. Fuh[1]
[1] Computer Science and Information Engineering, National Taiwan University, Taiwan
[2] Computer Science and Information Engineering, National Taiwan Normal University, Taiwan
[3]Information Management, Chung-Hua University, Taiwan

ABSTRACT

In this paper, a video stabilization technique is presented. There are four steps in the proposed approach. We begin with extracting feature points from the input image using the Lowe SIFT (Scale Invariant Feature Transform) point detection technique. This set of feature points is then matched against the set of feature points detected in the previous image using the Wyk et al. RKHS (Reproducing Kernel Hilbert Space) graph matching technique. We can calculate the camera motion between the two images with the aid of a 3D motion model. Expected and unexpected components are separated using a motion taxonomy method. Finally, a full-frame technique to fill up blank image areas is applied to the transformed image.

Index Terms—SIFT detection, RKHS graph matching, 3D motion, Motion taxonomy, Full-frame process.

1. INTRODUCTION

Vision systems play important roles in many intelligent applications, such as transportation systems, security systems, and monitoring systems. Cameras may be installed on building or held by a people. Hand held cameras often suffer from image instability [1]. In this paper, a video stabilization technique for a video camera held by a person is presented. This technique can be considered as a solution for the general video instability problems.

There are typically three major stages constituting a video stabilization process, *motion estimation*, *motion taxonomy*, and *image compensation*. Motion estimation is an ill-conditioned problem because of camcorder motion, which is three-dimensional (3-D), is estimated on the basis of input images that are two-dimensional (2-D). This stage actually dominates the entire stabilization process in both accurancy and time complexity. Many techniques, both hardware and software, have been proposed for improving the performance of motion estimation.

Since 3-D motion estimation from 2-D images is ill-conditioned, additional information needs be included in order to make the problem well-conditioned. Several motion models, which serve as an important source of a priori information, have been introduced, including a 2-D rigid model with four parameters [2], a 2-D affine model with six parameters [3], a 2.5-D model with seven parameters [4], and a 3-D model with nine parameters. Intuitively, the higher the dimension of the model is used, the better the accuracy is achieved. However, this may not be always the case because high dimensional models involve more complicated computations, which themselves may incur numerical instability.

There are two important tasks when using motion models: interframe motion calculation and model fitting. The interframe motions between successive images are calculated and then fitted to the preselected motion model, from which an overdetermined system of equations of motion parameters is obtained. By finding an optimal solution to this system, motion parameters are determined.

The techniques for calculating interframe motions can be classified into two categories, differential and matching (or correlation) approaches. The differential approaches, primarily for computing optical flow [5] and based on the assumption of image brightness constancy, are known to be sensitive to both high-order derivative errors and the aperture problem. Matching techniques can be divided into two classes, block matching [6] and feature-based matching [7]. Block matching is intrinsically weak at handling motions involving rotation and scale change. On the other hand, feature-based matching can cope with motions involving translation, rotation, and scaling. However, the effectiveness of the feature-based method depends heavily on the features employed.

The rest of this paper is organized as follows. In Section 2, we address the problem under consideration and give the solution process. The critical techniques for implementing the process are given in Sections 3-5. These include global feature extraction and matching in Section 3, camcorder motion estimation in Section 4, image compensation in Section 5. Experimental results are presented in Section 6. Finally, Section 7 presents our conclusions and gives suggestions for future work.

2. SYSTEM WORK FLOW

Our stabilization method consists of four steps (Fig. 1): feature point matching, camera motion estimation, motion taxonomy, and image compensation. To extract feature points, we apply SIFT (Scale Invariant Feature Transform) proposed by Lowe [8] to detect the feature points in an image. These feature points will be invariant to changes in scale, rotation, and illumination, and so are suitable for our application. A graph matching method modified from the method proposed in [9] and discussed in Section 3 is applied here for matching feature points between two successive images.

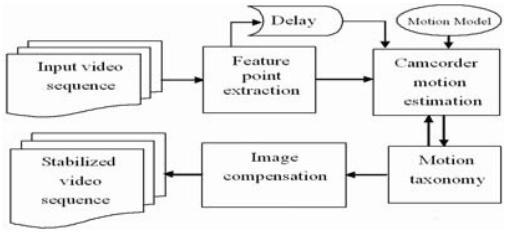


Fig. 1. Image stabilization flowchart.

Given a motion model, the 3-D motion of the camera can be calculated from the 2-D image. However, among these motions, some are unexpected because of camera motion, while some are expected because of object motion. In motion taxonomy, we can ignore the expected motions to calculate the camera motion. Camera motions are then smoothed in time and applied to transform each image to obtain a stabilized video sequence. Some missing parts of one image will be compensated at the same time to have a full-frame sequence.

3. FEATURE-POINT EXTRACTION

Feature points in one image are first detected by the SIFT method [8]. They are represented as a full connected graph $G = (V, E)$, where V is the set of the nodes denoting the feature points, and E is the set of the edges showing the relations between points. Using the graphs for two successive images, the corresponding points in each image can be found by using a graph matching method. From the corresponding feature points in successive images, the geometric relationship between the images can be estimated.

To represent the feature points in an image I, we construct matrices A_k, $k = 1,2,..,m$, where m is the number of kinds of features. The elements of A_k are the feature values calculated according to the k-th kind of feature for all feature points. Unary feature points are represented by a diagonal matrix, and binary features values by a symmetric matrix whose (i, j)-th element is the value for the relationship between the i-th and j-th feature points.

Feature points in the following image I' in the sequence can also be represented by matrices A'_k. The correspondence between the feature points in successive images can be obtained by solving the following equation for the permutation matrix P:

$$P = \min_P \left(\sum_{k=1}^{m} \left\| A_k - P A'_k P^T \right\| \right),$$

where P can be found by the method based on RKHS proposed in [9], and $\|.\|$ is some norm and it is Frobenius norm in our application.

RKHS graph matching, however, can be applied to A_k whose elements have been normalized. Here we modify this method by applying a new measurement function to adapt to the various types of feature values. Matrix P is found in two steps, weight matrix construction and optimal assignment. Each element $W(i, j)$ of the weight matrix is computed by:

$$W(i,j) = \frac{\sum_{k=1}^{m} \max_s \left[\min_t |a_k(i,s) - a'_k(j,t)| \right] + \max_s \left[\min_t |a_k(s,i) - a'_k(t,j)| \right]}{\max[a_k(i,.), a'_k(j,.), a_k(.,i), a'_k(.,j)]},$$

where $a_k(.)$ and $a'_k(.)$ are the element in A_k and A'_k, respectively. The calculated value gives the degree of the correspondence between node i and node j.

The graph with fewer nodes is padded with null nodes to give an equal number of nodes in each graph. In the matrices, feature values of the null nodes and their corresponding edges are set to null value, and are ignored in constructing the weight matrix. After constructing the weight matrix, we assign the optimal value for each element of P. Hopfield neural network is a well known assigning method. In this application, we use Hungarian algorithm [11] because of its polynomial processing time.

In our work, the unary feature values are location (x, y), color values, magnitude m and orientation θ:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - (L(x,y-1))^2},$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1)/(L(x+1,y) - L(x-1,y)),$$

where L is the corresponding scale image after convoluting with the Gaussian filter. The binary feature value is the Euclidean distance between the feature points in the same image. After these computations, each node will match a node in the other graph. Redundant nodes will match either a null node or another redundant node in the other graph.

4. GEOMETRIC CAMERA CALIBRATION

In the camera model shown in Fig. 2, $\boldsymbol{O}$ is the optical center of the lens; $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ are three orthogonal unit vectors and $\boldsymbol{k}$ points in the viewing direction. The image plane is located at $z = f$, where f is the focal length. In the image plane, the perspective projection point $\boldsymbol{p}$ of a scene point $\boldsymbol{P} = (x, y, z)^T$ ($x = \overrightarrow{\boldsymbol{OP}} \cdot \boldsymbol{i}$, $y = \overrightarrow{\boldsymbol{OP}} \cdot \boldsymbol{j}$, $z = \overrightarrow{\boldsymbol{OP}} \cdot \boldsymbol{k}$) can be defined as the intersection of $\overrightarrow{\boldsymbol{OP}}$ and the image plane. If $(u, v, f)^T$ is the coordinate vector of $\boldsymbol{p}$, u and v can be calculated by $u = \frac{f}{z}x$ and $v = \frac{f}{z}y$. We may say (u, v) are the image coordinates of $\boldsymbol{p}$. This process is known as geometric camera calibration.

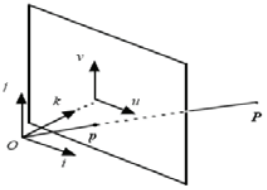


Fig. 2. Camera model.

Consider another camera model in the same space, where its optical center is located at $\boldsymbol{O}'$. We can define another coordinate frame F' by the origin $\boldsymbol{O}'$ and another three orthogonal unit vectors $\boldsymbol{i}'$, $\boldsymbol{j}'$, and $\boldsymbol{k}'$, where $\boldsymbol{k}'$ points in the viewing direction of this camera. The point $\boldsymbol{P}$ can be represented using a new coordinate vector $\boldsymbol{P}'$ in F' by a rotation and translation given by:

$$\boldsymbol{P}' = R\boldsymbol{P} + \boldsymbol{O}' \qquad (1)$$

where R is a rotation matrix, and $\boldsymbol{O}'$ is the coordinate vector of origin $\boldsymbol{O}$ in F'.

There is a projection point $\boldsymbol{p}'$ of $\boldsymbol{P}$ in the image plane, which is denoted by (u', v'); it can be calculated by the same function G mentioned above. When we obtain an image sequence from a moving camera, a scene point may change position from image to image. We want to estimate R and $\boldsymbol{O}'$ in two successive images given corresponding points $\boldsymbol{p}_i$ and $\boldsymbol{p}_i'$, i=1,…,n, where n is the number of point pairs.

The matrix R is a combination of the rotation matrices R_α, R_β, and R_γ about the $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ axes of the coordinate frame F, respectively

$$R = R_\alpha R_\beta R_\gamma ,$$

where α, β, and γ, are the rotating angles and

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad R_\beta = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad R_\gamma = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

Since the translation and rotation of the camera should be very small in a small time interval, we can assume that the rotating angles are very small. Under this assumption, R_α, R_β, and R_γ can be simplified as

$$R_\alpha^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\alpha \\ 0 & \alpha & 1 \end{bmatrix} \quad R_\beta^* = \begin{bmatrix} 1 & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & 1 \end{bmatrix} \quad R_\gamma^* = \begin{bmatrix} 1 & -\gamma & 0 \\ \gamma & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and Eq. (1) can be written as

$$\boldsymbol{P}' = R_\alpha^* R_\beta^* R_\gamma^* \boldsymbol{P} + \boldsymbol{O}' . \qquad (2)$$

Expending Eq. (2) under $\boldsymbol{O}' = (\Delta x, \Delta y, \Delta z)^{\mathrm{T}}$ gives

$$\begin{cases} x' = x - \gamma y + \beta z + \Delta x \\ y' = \gamma x + y + \alpha\beta x - \alpha\beta\gamma y - \alpha z + \Delta y \\ z' = \alpha\gamma x + \alpha y - \beta x + \beta\gamma y + z + \Delta z \end{cases} \qquad (3)$$

where $(x', y', z')^{\mathrm{T}}$ is the coordinate vector of $\boldsymbol{P}'$. Then we can calculate the image coordinate (u', v') of $\boldsymbol{p}'$ by $u' = \frac{f'}{z'}x'$ and $v' = \frac{f'}{z'}y'$. If $\alpha\beta x$ and $\alpha\beta\gamma y$ in Eq. (3) are very small relative to y, we will have

$$\begin{cases} u' = \frac{f'}{z'}(x - \gamma y + \beta z + \Delta x) = \frac{f'}{z'}\frac{z}{f}(\frac{f}{z}x - \gamma\frac{f}{z}y + \frac{f}{z}\beta z + \frac{f}{z}\Delta x) \\ v' = \frac{f'}{z'}(y + \gamma x - \alpha z + \Delta z) = \frac{f'}{z'}\frac{z}{f}(\frac{f}{z}y + \gamma\frac{f}{z}x - \frac{f}{z}\alpha z + \frac{f}{z}\Delta y) \end{cases} .$$

Replacing $f\alpha$, $f\beta$, and $\frac{f'}{z'}\frac{z}{f}$ with three constants, m, n, and $\frac{1}{\pi}$, and eliminating small values, $\frac{f}{z}\Delta x$ and $\frac{f}{z}\Delta y$, these functions can be rearranged as:

$$\begin{cases} u' = \frac{1}{\pi}(u - \gamma v + m) \\ v' = \frac{1}{\pi}(v + \gamma u - n) \end{cases} \Rightarrow \begin{cases} \pi u' + \gamma v - m = u \\ \pi v' - \gamma u + n = v \end{cases} . \qquad (4)$$

Given some corresponding point pairs (u_i, v_i) and (u_i', v_i'), i=1,…,n, we can solve for γ, m, and n by the least squares method. Let the solution be $\boldsymbol{x} = (\pi, \gamma, m, n)^{\mathrm{T}}$. This solution is not the actual value because of the expected motions which we will discuss later.

5. IMAGE COMPENSATION

If we calculate the transformation results (u_i^*, v_i^*) from (u_i, v_i) according to $\boldsymbol{x}$, there will be some differences $(\Delta x_i, \Delta y_i)$, where $\Delta x_i = |u_i^* - u_i'|$ and $\Delta y_i = |v_i^* - v_i'|$. These differences should be zero if all of the known points are shifted because of the camera motion. We call such a shift "unexpected motion". However, there will be some "expected motions". The latter are mixed with the object moving and are more complex than the previous one.

Expected motion points cause the initial $\boldsymbol{x}$ to be too imprecise. If we assume there are many more of unexpected motion points than the expected motion points, the initial $\boldsymbol{x}$ can be assumed to be close to the unexpected motion points and those unexpected motion points will have smaller $(\Delta x_i, \Delta y_i)$. To improve on the initial $\boldsymbol{x}$, we can eliminate those points with larger differences, which are assumed to be expected points, and recalculate $\boldsymbol{x}$ again. In our experience, the point with largest difference is eliminated, and we repeat the above process until the value of $\boldsymbol{x}$ approaches that of the previous one.

Suppose that $\boldsymbol{x}(t)$ is the vector of the camera motion from time t-1 to time t, and $\boldsymbol{s}(t)$ is the summation from $\boldsymbol{x}(0)$ to $\boldsymbol{x}(t)$. We may say that a video sequence has been stabilized if the stabilized vector $\boldsymbol{x}'(t)$ does not change significantly from the previous one $\boldsymbol{x}'(t$-1). In other words, the summation of the stabilized motion, $\boldsymbol{s}'(t)$, should smoothly vary. To obtain $\boldsymbol{s}'(t)$ we convolve $\boldsymbol{s}(t)$ with a Gaussian function. The image at time t, I_t, is then transformed using the compensation values, $\Delta\mathbf{x}_{t,t}$, defined as $\Delta\mathbf{x}_{t,t} = \mathbf{s}'(t) - \mathbf{s}(t)$. We denote the transformed image as I_t', and all of the transformed images will constitute a stable image sequence.

The boundary of the transformed image will be blank as shown in Fig. 3(b). To fill up this image, we can extract the lost information from the prior and following images I_k, $k = (t-n)...t...(t+n)$. First, the image I_k is transformed using $\Delta\mathbf{x}_{k,t} = \mathbf{s}'(t) - \mathbf{s}(k)$ to match the stabilized camera model at time t. We denote the new image as I_k'. When the value of the image point at (x, y) in I_t' is missing, we replace this point using the same point in I_k', $k = (t-n)...t...(t+n)$. Fig. 3(c) shows the result.

(a) (b) (c)

Fig. 3. (a) Original image. (b) After transformation for stabilization. (c) Filling up the boundary with the successive frame.

We convolve the Gaussian function with $\boldsymbol{s}(t)$ from time $(t$-$n)$ to $(t$+$n)$, which helps to obtain $\mathbf{s}'(t)$ and I_t' at time t+n. In filling images, I_k' with small $|t-k|$ is checked at first, because we believe that images closer to time t will be more similar to I_t'. Finally, the other points which still have no values can be filled up using the interpolation and extrapolation methods described in [10].

6. EXPERIMENTAL RESULTS

We test our algorithm on three kinds of image sequences. First, we capture images with the camera in a static position. Motions in the video are caused by movements of the object and the camera. And second, we capture the images with a moving camera. In this sequence the camera has a significant change along the z-axis. Final sequence is captured with the camera moved and panned at the same time.

Each image frame is extracted and processed with our algorithm to obtain a stabilized image. The processing time for one image (435x240 pixels) is about two seconds on a 3.0 GHz Pentium IV. The images before processing are shown in the top rows of Fig. 4 and Fig. 5, and the images after processing are shown in the bottom rows. An X in the middle is shown to help see the shift distance of the objects. More experimental results are shown in our website: http://www.csie.ntnu.edu.tw/~ipcv/Research/ulin/

In the first case (Fig. 4), there is a large area of water, where the feature points are few and difficult to match. Our processing result shows that the moving object is more stable than that in the original images (top rows). In addition, some lost information in the stabilized image will be filled up, for example, the plane in the top of frame 1.

Frame 1 Frame 2 Frame 3

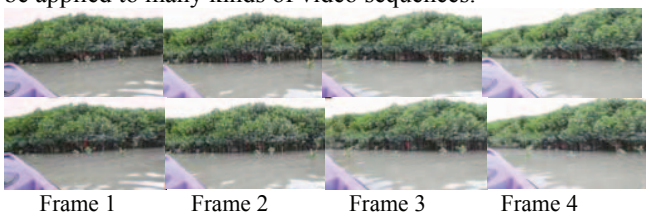
Fig. 4. Top row is original image sequence. Bottom row is image stabilized sequence.

In the second case (Fig. 5), an image sequence is captured with a moving camera. Objects in the scene have a significant change in size. Our processing result shows that the objects in the scene are stable, but there are some errors in the boundary after filling up. We can compute the depth (z value) to correct that, which will be our future works.

Frame 1 Frame 2 Frame 3 Frame 4

Fig. 5. Image sequence captured by a moving camera.

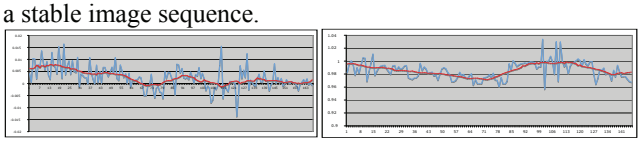
In the final case (Fig. 6), camera is held by a people on the boat. The camera has a significant change along x-axis. Our algorithm needs no motion assumption, so that it could be applied to many kinds of video sequences.



Frame 1 Frame 2 Frame 3 Frame 4

Fig. 6. Image sequence captured with significant pan motion.

Fig. 7 shows the motion values before and after stabilization. In this figure, lighter lines show the values before processing, and the darker lines show the stabilized results. In Fig. 7(a), the γ values (rotation angle) are shown along the vertical axis (time), which should be close to zero during the video acquisition. It shows that the camera motion is more stable. Fig. 7(b) shows the π value (scale) in the second case. Since the camera is moving forward, we will have $\pi < 1$. Our processing stabilizes the value and produces a stable image sequence.



(a) (b)

Fig. 7. (a) γ value (vertical axis) of the first sequence along time (horizontal axis), where lighter lines and darker lines denote the value before and after stabilization respectively. (b) π value of the second sequence.

7. CONCLUSIONS

In this paper, we propose an algorithm for stabilizing an image sequence captured by a hand-held video camera. This algorithm makes the following contributions: First, a 3D camera motion model can be obtained and used to stabilize the image sequence. It is more precise than traditional methods using 2D models. Second, our method can be applied to general case because it need not detect any objects in the scene. Third, by detecting the expected and unexpected motions, the camera motion model can be calculated more precisely and the foreground objects can be located at the same time.

Image stabilization helps to more conveniently extract information from the video. However, our method can not be applied to real-time systems because of the slowness of the SIFT computation. Our next phase of research is to improve the speed so that it can be performed in real time.

REFERENCES

[1] Y. M. Liang, H. R. Tyan, S. L. Chang, H. Y. Liao, and S. W. Wang, "Video Stabilization for a Camcorder Mounted on a Moving Vehicle," *IEEE Trans. on Vehicular Technology*, vol. 53, no. 6, pp. 1636-1648, 2004.

[2] C. Morimoto and R. Chellappa, "Fast Electronic Digital Image Stabilization for Off-Road Navigation," *Real-Time Imaging*, vol. 2, no. 5, pp. 285-296, 1996.

[3] M. Betke, "Recognition, Resolution, and Complexity of Objects Subject to Affine Transformations," *Int. Journal of Computer Vision*, vol. 44, no. 1, pp. 5–40, 2001.

[4] J. S. Jin, Z. Zhu, and G. Xu, "A Stable Vision System for Moving Vehicles," *IEEE Trans. Intell. Transport. Syst.*, vol. 1, pp. 32–39, 2000.

[5] J. Y. Chang, W. F. Hu, M. H. Cheng and B. S. Chang, "Digital Image Translational and Rotational Motion Stabilization Using Optical Flow Technique," *IEEE Trans. on Consumer Electronics*, vol. 48, no. 1, pp. 108–115, 2002.

[6] L. Xu and X. Lin, "Digital Image Stabilization Based on Circular Block Matching," *IEEE Trans. on Consumer Electronics*, vol. 52, no. 2, pp. 566-574, 2006

[7] Z. Duric and A. Rosenfeld, "Image Sequence Stabilization in Real Time," *Real-Time Imaging*, vol. 2, no. 5, pp. 271–284, 1996.

[8] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[9] M. A. van Wyk, T. S. Durrani, and B. J. van Wyj, "A RKHS Interpolator-Base Graph Matching Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 988-995, 2002.

[10] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H. Y. Shum, "Full-Frame Video Stabilization with Motion Inpainting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150-1163, 2006.

[11] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.