Fast 3D Stabilization and Mosaic Construction

Carlos Morimoto Rama Chellappa
Computer Vision Laboratory, Center for Automation Research
University of Maryland, College Park, MD 20742-3275
carlos@cfar.umd.edu

Abstract

We present a fast electronic image stabilization system that compensates for 3D rotation. The extended Kalman filter framework is employed to estimate the rotation between frames, which is represented using unit quaternions. A small set of automatically selected and tracked feature points are used as measurements. The effectiveness of this technique is also demonstrated by constructing mosaic images from the motion estimates, and comparing them to mosaics built from 2D stabilization algorithms. Two different stabilization schemes are presented. The first, implemented in a real-time platform based on a Datacube MV200 board, estimates the motion between two consecutive frames and is able to process gray level images of resolution 128x120 at 10 Hz. The second scheme estimates the motion between the current frame and an inverse mosaic; this allows better estimation without the need for indexing the new image frames. Experimental results for both schemes using real and synthetic image sequences are presented.

1. Introduction

Camera motion estimation is an integral part of any computer vision or robotic system that has to navigate in a dynamic environment. Whenever part of the camera motion is not necessary (unwanted or unintended), image stabilization can be applied as a useful preprocessing step before further analysis of the image sequence. In this paper we use an iterative extended Kalman filter (IEKF) to estimate the 3D motion of the camera. Stabilization is achieved by derotating the input sequence [2, 3, 9].

We present two different stabilization algorithms. The first estimates the motion between two consecutive frames and was implemented in a real-time platform (a Datacube MV200 connected to a Sun Sparc 20). The second algorithm computes the motion of the current frame relative to an inverse mosaic. Computing the frame-to-mosaic motion requires indexing (finding the approximate position of the new image in the mosaic, in order to facilitate the motion estimation process). We show in Section 4.1 how the use of inverse mosaics can solve the indexing problem. For an image sequence with n frames, a mosaic image can be constructed by placing the initial frame f_0 at the center of the mosaic and then warping every new frame to its correct position up to frame f_n. To build an inverse mosaic one can first warp frame f_n using the inverse motion parameters, and then warp every previous frame down to frame f_0. The inverse mosaic can also be computed directly from the mosaic image, using a single global inverse transformation.

Most of the current image stabilization algorithms implemented in real-time use 2D models due to their simplicity [4, 8]. Feature-based motion estimation or image registration algorithms are used by these methods in order to bring all the images in the sequence onto alignment. For 3D models under perspective projection, the displacement of each image pixel also depends on the structure of the scene, or more precisely, on the depth of the corresponding 3D point. It is possible to parameterize these models so that only the translational components carry structural information, while the rotational components are depth-independent. In this paper, stabilization is defined as the process of compensating for 3D rotation; this definition is also used in [2, 3, 9]. Compensating for the camera rotation makes the image sequence look mechanically stabilized, as if the camera were mounted on a gyroscopic platform. The fast implementation of the 3D stabilization system can process about 10 frames per second for 8-bit gray level images of resolution 128×120 (with maximum interframe displacement of 15 pixels), demonstrating that IEKFs are efficient tools for 3D motion estimation and real-time image analysis applications.

The rest of this paper is organized as follows: Section 2 describes the camera motion model used for parameter estimation, which is the subject of Section 3. The details of the two stabilization algorithms are presented in Section

[0]The support of ARPA and ARO under contract DAAH04-93-G-0419, and of the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), are gratefully acknowledged.

4, and experimental results from both algorithms using real and synthetic image sequences are shown in Section 5. Section 6 concludes the paper.

2. Camera Motion Model

Let $\mathbf{P} = (X, Y, Z)^T$ be a 3D point in a Cartesian coordinate system fixed on the camera and let $\mathbf{p} = (x, y)^T$ be the position of the image of $\mathbf{P}$. The image plane defined by the x and y axes is perpendicular to the Z axis and contains the principal point $(0, 0, f)$. Thus the perspective projection of a point $\mathbf{P} = (X, Y, Z)^T$ onto the image plane is

$$\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix} \tag{1}$$

where f is the camera focal length.

Under a rigid motion assumption, an arbitrary motion of the camera can be described by a translation $\mathbf{T}$ and a rotation $\mathbf{R}$, so that a point $\mathbf{P}_0$ at the camera coordinate system at time t_0 is described in the new camera position at t_1 by

$$\mathbf{P}_1 = \mathbf{R}\mathbf{P}_0 + \mathbf{T} \Rightarrow$$
$$\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + \begin{pmatrix} T_X \\ T_Y \\ T_Z \end{pmatrix} \tag{2}$$

where $\mathbf{R}$ is a 3×3 orthonormal matrix.

For distant points ($Z_0 \gg T_X, T_Y$, and T_Z), the displacement is primarily due to rotation, and the projection of $\mathbf{P}_1$ can be obtained by

$$\mathbf{p}_1 = \begin{pmatrix} f\frac{(r_{11}x_0 + r_{12}y_0 + r_{13}f)}{(r_{31}x_0 + r_{32}y_0 + r_{33}f)} \\ f\frac{(r_{21}x_0 + r_{22}y_0 + r_{23}f)}{(r_{31}x_0 + r_{32}y_0 + r_{33}f)} \end{pmatrix} \tag{3}$$

The use of distant features for motion estimation and image stabilization has been addressed in [2, 3, 9]. Such features constitute very strong visual cues that are present in almost all outdoor scenes, although it may be hard to guarantee that all the features are distant. In this paper we estimate the three parameters that describe the rotation of the camera using an iterated extended Kalman filter (IEKF).

2.1. Quaternions and Unit Quaternions

In this section we introduce a few basic properties of quaternions. More detailed descriptions can be found in [5, 6]. Quaternions are commonly used to represent 3D rotation, and are composed of a scalar and a vector part, as in $\breve{\mathbf{q}} = q_0 + \mathbf{q}$, where $\mathbf{q} = (q_x, q_y, q_z)^T$. The *dot product* operator for quaternions is defined as $\breve{\mathbf{p}}.\breve{\mathbf{q}} = p_0 q_0 + \mathbf{p}.\mathbf{q}$, and the *norm* of a quaternion is given by $|\breve{\mathbf{q}}| = \breve{\mathbf{q}}.\breve{\mathbf{q}} = q_0^2 + \mathbf{q}.\mathbf{q}$. A unit quaternion is defined as a quaternion with unit norm.

A unit quaternion can be interpreted as a rotation θ around a unit vector $\mathbf{w}$ by the following equation: $\breve{\mathbf{q}} = \sin(\theta/2) + \mathbf{w}\cos(\theta/2)$. Note that $\breve{\mathbf{q}}$ and $-\breve{\mathbf{q}}$ correspond to the same rotation since a rotation of θ around a vector $\mathbf{q}$ is equivalent to a rotation of $-\theta$ around the vector $-\mathbf{q}$.

Let the conjugate of a quaternion be defined as $\breve{\mathbf{q}}^* = q_0 - \mathbf{q}$, and the product of two quaternions as

$$\breve{\mathbf{r}} = \breve{\mathbf{p}}\breve{\mathbf{q}} = \begin{cases} r_0 = p_0 q_0 - \mathbf{p}.\mathbf{q}; \\ \mathbf{r} = p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}. \end{cases} \tag{4}$$

Thus, the conjugate of $\breve{\mathbf{q}}$ is also its inverse since $\breve{\mathbf{q}}^*\breve{\mathbf{q}} = \breve{\mathbf{q}}\breve{\mathbf{q}}^* = 1$. It is useful to have the product of quaternions expanded in matrix form as follows:

$$\breve{\mathbf{p}}\breve{\mathbf{q}} = \begin{pmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & -p_z & p_y \\ p_y & p_z & p_0 & -p_x \\ p_z & -p_y & p_x & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} \tag{5}$$

Note that the multiplication of quaternions is associative but it is not commutative, i.e., in general $\breve{\mathbf{p}}\breve{\mathbf{q}}$ is not the same as $\breve{\mathbf{q}}\breve{\mathbf{p}}$.

The rotation of a vector or point $\mathbf{P}$ to a vector or point $\mathbf{P}'$ can be represented by a quaternion $\breve{\mathbf{q}}$ according to

$$(0 + \mathbf{P}') = \breve{\mathbf{q}}(0 + \mathbf{P})\breve{\mathbf{q}}^* \tag{6}$$

Composition of rotations can be performed by multiplication of quaternions since

$$\begin{aligned} (0 + \mathbf{P}'') &= \breve{\mathbf{q}}(0 + \mathbf{P}')\breve{\mathbf{q}}^* = \\ \breve{\mathbf{q}}(\breve{\mathbf{r}}(0 + \mathbf{P})\breve{\mathbf{r}}^*)\breve{\mathbf{q}}^* &= (\breve{\mathbf{q}}\breve{\mathbf{r}})(0 + \mathbf{P})(\breve{\mathbf{q}}\breve{\mathbf{r}})^* \end{aligned} \tag{7}$$

where it is easy to verify that $(\breve{\mathbf{r}}^*\breve{\mathbf{q}}^*)$ is equivalent to $(\breve{\mathbf{q}}\breve{\mathbf{r}})^*$.

The nine components of the orthonormal rotation matrix $\mathbf{R}$ in (2) can be represented by the parameters of a unit quaternion simply by expanding (6) using (5), so that

$$\mathbf{R} = \begin{pmatrix} 1-2q_y^2-2q_z^2 & 2(-q_0q_z+q_xq_y) & 2(q_0q_y+q_xq_z) \\ 2(q_0q_z+q_xq_y) & 1-2q_x^2-2q_z^2 & 2(-q_0q_x+q_yq_z) \\ 2(-q_0q_y+q_xq_z) & 2(q_0q_x+q_yq_z) & 1-2q_x^2-2q_y^2 \end{pmatrix} \tag{8}$$

3. Motion Estimation

The dynamics of the camera is described as the evolution of a unit quaternion. An IEKF is used to estimate the interframe rotation $\breve{\mathbf{q}}$. EKFs have been extensively used for the problem of motion estimation from a sequence of images [1, 9, 10]. In order to achieve real-time performance, this framework is simplified here to compute only the rotational parameters from distant feature points.

A unit quaternion has only three degrees of freedom due to its unit norm constraint, so that it can be represented using only the vector parameters. The remaining scalar parameter

is computed from $q_0 = (1 - q_x^2 - q_y^2 - q_z^2)^{\frac{1}{2}}$. Only nonnegative values of q_0 are considered, so that (8) can be rewritten using (q_x, q_y, q_z) only.

The state vector $\mathbf{x}$ and plant equations are defined as follows:

$$\left.\begin{array}{c} \mathbf{x} \stackrel{def}{=} \breve{\mathbf{q}} + \mathbf{n} \\ \dot{\mathbf{x}} = \mathbf{0} \end{array}\right\} \Rightarrow \mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) \qquad (9)$$

where $\breve{\mathbf{q}}$ is a quaternion represented by its vector part only, and $\mathbf{n}$ is the associated plant noise.

The following measurement equations are derived from (3):

$$\mathbf{z}(t_i) = \mathbf{h}_{i|i-1}[\mathbf{x}(t_i)] + \eta(t_i) \qquad (10)$$

where $\mathbf{h}$ is a nonlinear function which relates the current state to the measurement vector $\mathbf{z}(t_i)$, and η is the measurement noise. After tracking a set of N feature points, a two step EKF algorithm is used to estimate the total rotation. The first step is to compute the state and covariance predictions at time t_{i-1} before incorporating the information from $\mathbf{z}(t_i)$ by

$$\begin{array}{c} \hat{\mathbf{x}}(t_i^-) = \hat{\mathbf{x}}(t_{i-1}^+) \\ \boldsymbol{\Sigma}(t_i^-) = \boldsymbol{\Sigma}(t_{i-1}^+) + \boldsymbol{\Sigma}_{\mathbf{n}}(t_{i-1}) \end{array} \qquad (11)$$

where $\hat{\mathbf{x}}(t_{i-1}^+)$ is the estimate of $\mathbf{x}(t_{i-1})$ and $\boldsymbol{\Sigma}(t_{i-1}^+)$ is its associated covariance obtained based on the information up to time t_{i-1}; $\hat{\mathbf{x}}(t_i^-)$ and $\boldsymbol{\Sigma}(t_i^-)$ are the predicted estimates before the incorporation of the i^{th} measurements; and $\boldsymbol{\Sigma}_{\mathbf{n}}(t_{i-1})$ is the covariance of the plant noise $\mathbf{n}(t_{i-1})$.

The *update step* follows the previous *prediction step*. When $\mathbf{z}(t_i)$ becomes available, the state and covariance estimates are updated by

$$\begin{array}{c} \mathbf{K}(t_i) = \frac{\boldsymbol{\Sigma}(t_i^-)\mathbf{H}_{i|i-1}^T}{\mathbf{H}_{i|i-1}\boldsymbol{\Sigma}(t_i^-)\mathbf{H}_{i|i-1}^T + \boldsymbol{\Sigma}_\eta(t_i)} \\ \hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\{\mathbf{z}(t_i) - \mathbf{h}_{i|i-1}[\hat{\mathbf{x}}(t_i^-)]\} \\ \boldsymbol{\Sigma}(t_i^+) = [\mathbf{I} - \mathbf{K}(t_i)\mathbf{H}_{i|i-1}]\boldsymbol{\Sigma}(t_i^-) \end{array} \qquad (12)$$

where $\mathbf{K}(t_i)$ is a $3 \times N$ matrix which corresponds to the Kalman gain, $\boldsymbol{\Sigma}_\eta(t_i)$ is the covariance of $\eta(t_i)$ and $\mathbf{I}$ is a 3×3 identity matrix. $\mathbf{H}_{i|i-1}$ is the linearized approximation of $\mathbf{h}_{i|i-1}$, defined as

$$\mathbf{H}_{i|i-1} = \left.\frac{\delta \mathbf{h}_{i|i-1}}{\delta \mathbf{x}(i)}\right|_{\hat{\mathbf{x}}(t_i^-)} \qquad (13)$$

A batch process using a least-square estimate of the rotational parameters can be used to initialize the algorithm, as in [9], but for the experiments shown in Section 5 we simply assume zero rotation as the initial estimate.

To speed up the process and reduce the amount of computation that is required to achieve real-time performance, the measurement vectors are sorted according to their fits to the previous estimate, so that only the best M points ($M < N$) are actually used by the EKF. The solution is refined iteratively by using the new estimate $\hat{\mathbf{x}}_i$ to evaluate $\mathbf{H}$ and $\mathbf{h}$ for a few iterations. More detailed derivations of the Kalman filter equations can be found in [7].

4. Image Stabilization

Real-time image stabilization systems have been described in [4, 8]. They are based on 2D similarity or affine motion models, and use multi-resolution to compensate for large image displacements. Each algorithm was optimized for a different image processing platform. Part of the system described in this paper is based on the system presented in [8]. As in [2, 3, 9], image stabilization is defined as the process of compensating for the 3D camera rotation, i.e., stabilization is achieved by derotating the image sequence. Rotation is estimated as described above, using the tracked feature positions as measurements.

Figure 1 shows the block diagram of the image stabilization system, which was implemented in a real-time image processing platform based on a Datacube MV200 board. The Datacube board is based on a parallel architecture composed of several image processing elements which can be connected into pipelines in order to accomplish image processing tasks. Image acquisition, pyramid construction, and image warping are performed by the Datacube. The host computer performs tracking, estimates the rotation based on the algorithm presented in Section 3, and finally combines the interframe estimates to determine the global transformation used to stabilize the current image frame. The host is also responsible for the user interface module, which is not shown in the block diagram.

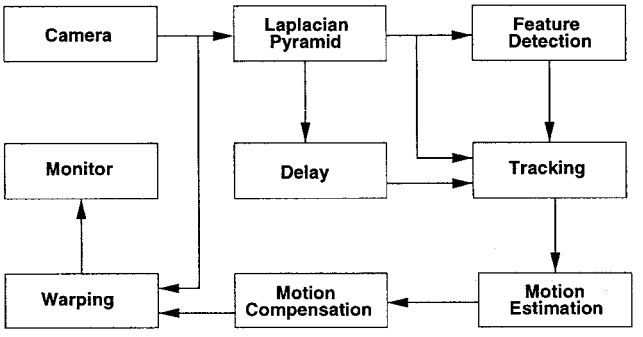

Figure 1. Block diagram of the real-time image stabilization system.

A small set of feature points are detected and hierarchically tracked with subpixel accuracy between two consecutive frames, using the method described in [8]. The feature displacements are then used by the IEKF to update the rotation estimate. The interframe estimates must be combined

in order to obtain the global transformation which stabilizes the current frame relative to the reference frame. If $\breve{\mathbf{p}}_{i-1}$ is the previous global rotation, and $\breve{\mathbf{q}}_i$ is the interframe motion between frames i and $i-1$, the new global rotation $\breve{\mathbf{p}}_i$ is obtained by multiplying two quaternions: $\breve{\mathbf{p}}_i = \breve{\mathbf{q}}_i \breve{\mathbf{p}}_{i-1}$. Finally, the new global estimate is used by the image warper to generate the stabilized sequence.

The simplicity of this method allows its implementation on our real-time platform. This system can be modified to construct panoramic views of the scene (mosaics), which can also be used to facilitate the motion estimation process. The construction and registration of the current frame to a mosaic image can reduce the estimation errors when the camera motion predominantly consists of lateral translation or pan and tilt.

4.1. Registration to an Inverse Mosaic

Building a mosaic image from 2D affine motion parameters or 3D rotation parameters can be done by appropriately warping each pixel from the new frame into its corresponding mosaic position, but in general, better results are achieved when the target image (i.e. the mosaic image) is scanned and the closest pixel value from the new frame (or the result of a small-neighborhood interpolation around that pixel) is inserted into the mosaic.

To register a new frame to the mosaic, it is desirable to have a rough estimate of its correct placement. Finding this initial estimate is known as *indexing*. Hansen *et al.* [4] solve the indexing problem by correlating small "representative landmark" regions of the new frame with the mosaic. Depending on the warping of the current frame, though, this technique could fail. To increase the robustness of this scheme, it is possible to pre-warp the current frame based on the previous estimate; this brings the current frame close to its true position in most cases, but it still may be difficult to register the warped image to the mosaic. The use of inverse mosaics avoids the indexing problem by keeping the previous frame always at a fixed position, e.g., at the center of the mosaic, and without any distortions due to warping, which increases the probability of high correlation of overlapping areas.

Figure 2 shows the block diagram of the stabilization system based on inverse mosaics. In order to track features of different image resolutions, a mosaic pyramid is built from the Gaussian pyramid of the current frame and the global motion estimate. The inverse mosaic pyramid is obtained by warping the mosaic pyramid using the inverse global motion estimate. Features detected in the new frame are hierarchically tracked in the inverse mosaic pyramid, and the estimation procedure used in the frame-to-frame algorithm is applied. The new estimate is then used to update the mosaic and inverse mosaic pyramids for the next iteration. The process has to be reinitialized with the creation of a new mosaic pyramid (or new tiles [4]) whenever a frame is warped outside the mosaic.

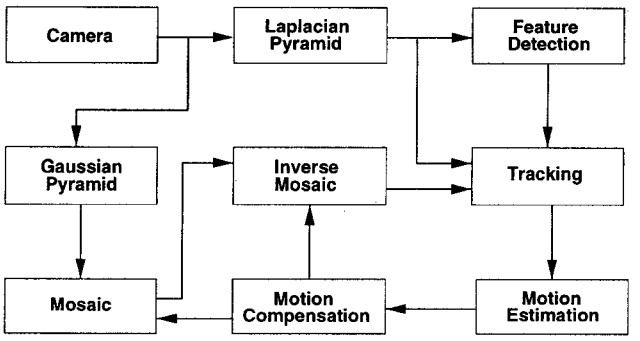

Figure 2. Block diagram of the stabilization system using inverse mosaics.

Besides the extra computational cost, one drawback of this method is that in general warping requires interpolation, so that the inverse mosaic may be over-smoothed. In the experiments presented in the following section, we compensate for the over-smoothing effect by using larger correlation windows (instead of changing the warping scheme to nearest-neighbor selection, which avoids interpolation but creates some artifacts).

5. Experimental Results

In this section we present experimental results for both of the algorithms described above. We are forced here to show only single frames from the stabilized and mosaic sequences, where video sequences would be much more appropriate. We invite the reader to look at these videos, available in MPEG format, at the following WWW address: **http://www.cfar.umd.edu/~carlos/cvpr97.html**. In the following sections, a file at this address named "mosaic" will be referenced by http:[mosaic].

Figure 3 is an example of the results obtained from the fast 3D derotation system using an off-road sequence provided by NIST. The camera is rigidly mounted on the vehicle and is moving forward. The top row shows the 5th frame of the input sequence (left) and its corresponding stabilized frame (right). The original and stabilized sequences are available at http:[seq-1.mpg]. The difference between the top-left image and the 10th input frame is shown on the bottom left, and the difference between the corresponding stabilized frames on the bottom right. The darkness of a spot on the bottom images is proportional to the difference of intensity between the corresponding spots in each frame. Since stabilization has to compensate for the motion between frames, the difference images can be considered as error measurements which stabilization attempts to

minimize. Observe that the regions around the horizon line are very light due to stabilization. Errors are bigger around objects that are closer to the camera (darker regions), since they have large translational components which are not compensated by our method.

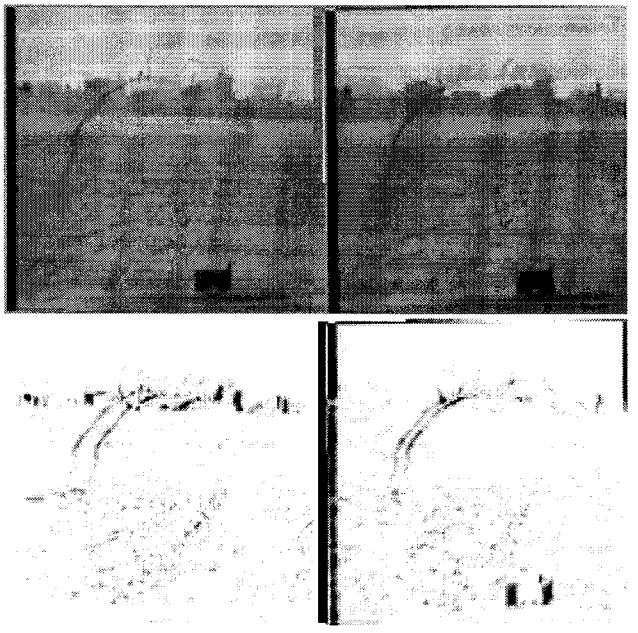

Figure 3. 3D image stabilization results.

The real-time implementation typically detects and tracks 9 feature points between frames, selects the best 7 based on the correlation results, and then uses the 4 points which best approximate the current rotation estimate. The maximum feature displacement tolerated by the tracker is set to 15 pixels. Under these settings, the system is able to process approximately 9.8 frames per second.

Figure 4 shows the result of the frame-to-mosaic algorithm presented in Section 4.1 for 30 frames of a synthetic sequence which simulates dominant lateral translation with small rotations around the optical axis. The sequence was generated by warping and cutting regions (of size 128×128) from a bigger image (of size 512×512). The maximum translation between frames was set to 10 pixels and the maximum rotation to 3 degrees. Figures 4a and 4b show results for the 15th and 26th frames respectively. Each figures shows the inverse mosaic on top; the regular mosaic (left) and current input frame (right) are shown at the bottom. The box in the center of the inverse mosaic corresponds to the last frame put into the regular mosaic and also serves as the initial estimate (index) of the region to be registered to the next frame. After frame 16, the camera starts to move back to its original position, and the overlap of the current frame with the mosaic is almost 100%. For this particular sequence, the real-time frame-to-frame algorithm produced

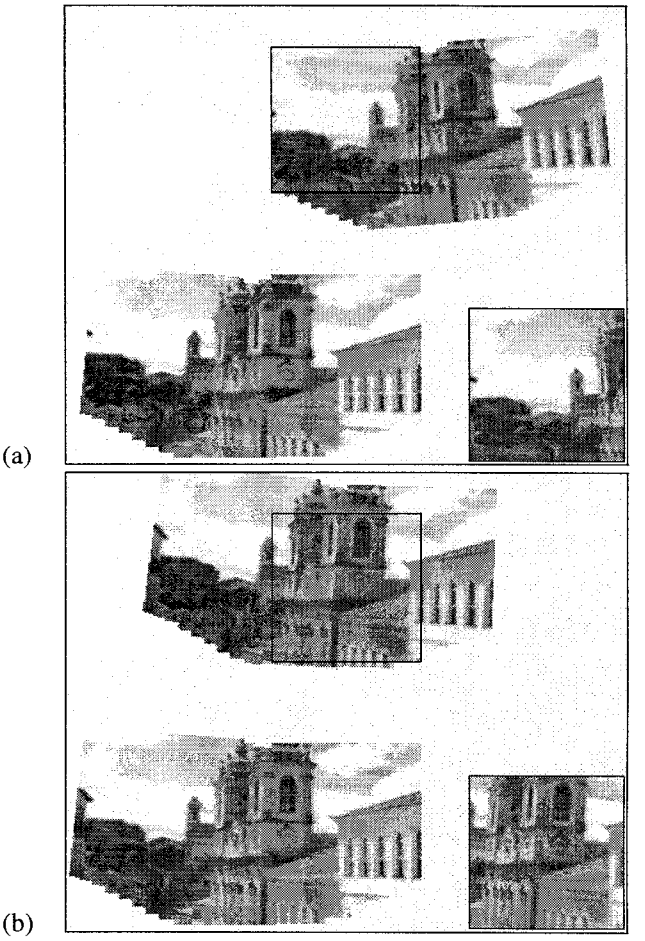

Figure 4.. Registration to an inverse mosaic.

a very similar result, except for a few artifacts due to the accumulation of error.

The last example compares the real-time 3D algorithm with the real-time similarity model-based algorithm presented in [8]. They both use the same set of 11 feature points for motion estimation. The 2D algorithm uses all of them to fit a similarity model using least-squares, and the 3D model uses only the four points which best approximate the current rotation estimate.

The original sequence is composed of 200 frames and the dominant motion is left-to-right panning. To help in comparison and visualization, the reference frame was assumed to be the 100th frame, and appears at the center of the mosaics. The first column shows the 50th, 100th, and 200th frames from top to bottom. The second column shows the corresponding mosaic images constructed from the 2D estimates, and the third column shows the corresponding mosaics constructed using the 3D estimates. Since the camera calibration is unknown, we "guessed" the camera FOV to be 3×4 focal lengths. The mosaic from the 2D estimates (http:[seq-3.mpg]) does a good job locally, but the 3D

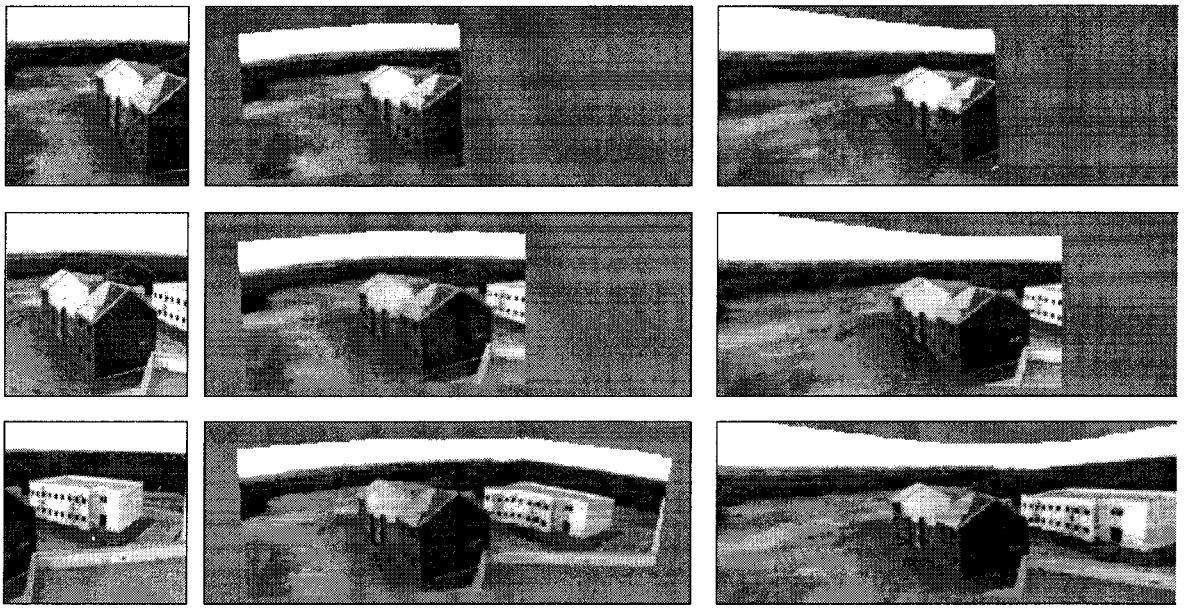

Figure 5. Mosaics from 200 frames of a left-to-right panning sequence.

mosaic (http:[seq-4.mpg]) looks much more natural, as if it were a panoramic picture taken using fish eye lens. The original sequence and the 3D stabilized output can be seen at http:[seq-5.mpg].

6. Conclusion

We have presented a 3D model-based real-time stabilization system that estimates the motion of the camera using an IEKF. Stabilization is achieved by derotating the input camera sequence. Rotations are represented using unit quaternions, whose good numerical properties contribute to the overall performance of the system. The system was implemented in a real-time image processing platform (a Datacube MV200 connected to a Sun Sparc 20) and is able to process 128×120 images at approximately 10 Hz.

The system has been successfully tested under a variety of situations that include dominant forward and lateral translations with small rotations, as well as panning. We have built mosaic images for these last two cases, and for the particular case of panning, we compared the results of the 3D algorithm with those of a 2D similarity model-based algorithm to show that 3D mosaicking provides more natural panoramic views.

We are currently extending the motion models to include translation parameters, which we believe will contribute to make the system more robust and able to generate more realistic mosaics. We are also investigating the applicability of these motion estimation, image stabilization, and mosaicking techniques to video coding.

References

[1] T. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.

[2] L. Davis, R. Bajcsy, R. Nelson, and M. Herman. RSTA on the move. In *Proc. DARPA Image Understanding Workshop*, pages 435–456, Monterey, CA, November 1994.

[3] Z. Durić and A. Rosenfeld. Stabilization of image sequences. Technical Report CAR-TR-778, Center for Automation Research, University of Maryland, College Park, 1995.

[4] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *Proc. DARPA Image Understanding Workshop*, pages 457–465, Monterey, CA, November 1994.

[5] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, April 1987.

[6] K. Kanatani. *Group-Theoretical Methods in Image Understanding*. Springer-Verlag, Berlin, Germany, 1990.

[7] P. Maybeck. *Stochastic Models, Estimation and Control*. Academic Press, New York, NY, 1982.

[8] C. Morimoto and R. Chellappa. Fast electronic digital image stabilization. In *Proc. International Conference on Pattern Recognition*, Vienna, Austria, August 1996.

[9] Y. Yao, P. Burlina, and R. Chellappa. Electronic image stabilization using multiple visual cues. In *Proc. International Conference on Image Processing*, pages 191–194, Washington, D.C., October 1995.

[10] G.-S. J. Young and R. Chellappa. 3D motion estimation using a sequence of noisy stereo images: models, estimation, and uniqueness results. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):735–759, August 1990.