Fast Image Stabilization and Mosaicking

Carlos Morimoto **Rama Chellappa**
Center for Automation Research
University of Maryland
College Park, MD 20742

Steve Balakirsky
U.S. Army Research Laboratory
Adelphi, MD 20783

Abstract

We present two fast implementations of electronic image stabilization and mosaicking systems. The first one is based on a 2D similarity model and is targeted to process PREDATOR video data. The second system uses a 3D model and compensates for 3D rotation. Both systems have been implemented on parallel pipeline image-processing hardware (a Datacube Max-Video 200) connected to a Themis 10MP. Both algorithms use a feature-based multi-resolution technique which tracks a small set of features to estimate the motion of the camera. The extended Kalman filter framework is employed by the 3D de-rotation system. The inter-frame motion estimates relative to a reference frame are used to warp the current frame in order to achieve stabilization. The estimates are also used to construct mosaics by aligning the frames. A fast mosaicking implementation is presented for the 2D system. Experimental results demonstrate the robustness of both systems at frame rates above 10 frames/second.

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 or "unwanted", image stabilization can be applied as a preprocessing step before further analysis of the image sequence. It can be used as a front-end system in a variety of dynamic image analysis applications or simply as a visualization tool. Image stabilization has been used for the computation of egomotion [Viéville *et al.*, 1993; Irani *et al.*, 1994], video compression [Kwon *et al.*, 1995; Morimoto *et al.*, 1996], and detection and tracking of independently moving objects [Balakirsky, 1995; Burt and Anandan, 1994; Morimoto *et al.*, 1995]. For more natural visualization, vehicle models are used to filter high-frequency or oscillatory motion components due to irregularities of the terrain [Durić and Rosenfeld, 1995; Yao *et al.*, 1995].

Methods proposed for electronic image stabilization can be distinguished by the models adopted to estimate the camera motion. Several 2D and 3D stabilization schemes are described by Davis *et al.* [Davis *et al.*, 1994]. For 2D models, all the estimated motion parameters are in general compensated for, i.e., all motion is removed from the input sequence [Burt and Anandan, 1994; Irani *et al.*, 1995; Sawhney *et al.*, 1995].

For 3D models under perspective projection, the displacement of each image pixel will also depend 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. Stabilization in 3D is achieved by derotating the frames, generating a translation-only sequence, or a sequence containing only translation and low-frequency rotation (smoothed rotation) [Durić and Rosenfeld, 1995; Yao *et al.*, 1995]. By compensating for the camera rotation, the resulting image sequence looks mechanically stabilized, as if the camera were mounted on a gyroscopic platform.

Most of the current image stabilization algorithms that have been implemented in real time use 2D models due to their simplicity [Hansen *et al.*, 1994; Morimoto and Chellappa, 1996]. Feature-based motion estimation or image registration algorithms are used by these methods

The support of the Defense Advanced Research Projects Agency (ARPA Order No. A422) and the U.S. Army Research Office under Contract DAAH04-93-G-0419 is gratefully acknowledged.

in order to bring all the images in the sequence into alignment. These algorithms are targeted to specific real-time image processing platforms. The systems operate with images of resolution 128×128 at above 10 frames per second, and are robust to large image displacements. The system developed by Hansen *et al.* [Hansen *et al.*, 1994] uses a mosaic-based registration technique implemented on pyramidal hardware (VFE-100). The system uses a multi-resolution, iterative process to estimate the affine motion parameters between levels of Laplacian pyramid images. From coarse to fine levels, the optical flow of local patches of the image is computed using a cross-correlation scheme. The motion parameters are then computed by fitting an affine motion model to the flow. These parameters are used to warp the previous image of the next finer pyramid level to the current image, and the refinement process continues until the desired precision is achieved. This scheme, combined with the construction of a mosaic image, allows the system to cope with large image displacements.

In this paper we present two image stabilization systems based on 2D and 3D models. The 2D system includes several modifications to the system presented in [Morimoto *et al.*, 1995], in order to process data from PREDATOR video, which are sequences taken from an airborne platform and are characterized by low quality and relatively low inter-frame displacement (less than 10% of the image size). Most of the modifications were necessary because of the low quality of the video sequences, due to lossy compression. The second system uses an extended Kalman filter (EKF) to estimate the 3D motion of the camera, and stabilization is achieved by derotating the input sequence.

Both systems were implemented on a Datacube MaxVideo 200 card plugged into the same VME backplane as a Themis 10MP. The MV200 is a parallel pipeline image processing board very commonly used for real-time image processing, and the Themis is a dual 100MHz hyperSPARC board which is running Solaris 2.4. They are able to process about 10 frames per second for 8-bit gray level images of resolution 128×120. The 2D system is also able to construct mosaic images in real time, directly onto a window on the host computer.

This paper is organized as follows. Section 2 introduces the 2D model-based image stabilization algorithm; the 3D algorithm is described in Section 3. Section 4 describes the implementation of the real-time 2D mosaicking display and how 3D mosaics can also be computed. Section 5 shows experimental results of the performance of both systems, and Section 6 concludes the paper.

2 2D Image Stabilization Algorithm

The 2D similarity model-based image stabilization system is based on the fast implementation of the image stabilization algorithm presented in [Morimoto *et al.*, 1995]. A basic stabilization system is composed of three modules shown in Figure 1. The *motion estimation* module computes the motion or global transformation between consecutive frames which is used by the *motion compensation* module to determine the global transformation which brings the newest frame into alignment with the reference frame. The *image composition* module generates the stabilized sequence and/or mosaic by warping the current frame using the motion estimates. Section 4 describes how the motion estimates are also used to construct a mosaic in real time by directly aligning the current frame with the mosaic constructed from previous frames.

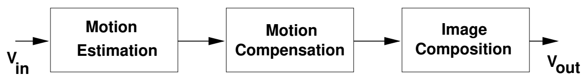

Figure 1: Modules of a general electronic image stabilization system.

A block diagram of the 2D system is shown in Figure 2. The modules inside the dotted line are performed by the Datacube board, while the other modules are processed by the host computer. The Datacube digitizes the video signal from the camera and builds Gaussian and Laplacian pyramids for each new frame. The Laplacian pyramid is used for feature detection and tracking is performed on the Gaussian images. Feature detection and tracking, motion estimation, motion compensation, and mosaic construction are done by the host computer. The Datacube also receives the computed global motion to warp the current frame and generates the stabilized video output.

2.1 Motion Estimation

The structure of the motion estimation module is similar to the feature-based multi-resolution image registration algorithm presented in [Zheng and Chellappa, 1993]. Starting from the coarsest Laplacian pyramid level, a small number of non-overlapping regions are scanned, and the pixel with maximum intensity in each region is selected for tracking. Each feature is tracked between the corresponding Gaussian pyramid level of the current and previous frames, using the sum of absolute differences (SAD) as similarity measure.

The SAD between two windows of size $2W + 1$ centered at feature point $P_t(x, y)$ and its matching candidate $P_{t-1}(u, v)$ is given by

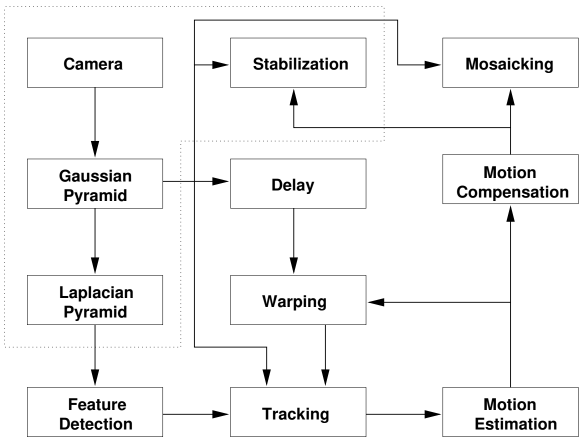

Figure 2: Block diagram of the 2D stabilization and mosaicking system.

$$\text{SAD} = \sum_{ij} |(P_t(x+i, y+j) - P_{t-1}(u+i, v+j)| \tag{1}$$

where i and j vary from $-W$ to $+W$. A match is obtained by searching for the minimum SAD over a neighborhood (search window) around the feature. For a feature at pixel coordinates (x, y), the search is performed by varying the candidate coordinates (u, v) in the interval $[(x-S)\ldots(x+S), (y-S)\ldots(y+S)]$, where $2S+1$ defines the search window size. After the grid-to-grid matches are obtained, displacements with subpixel accuracies are computed using a differential method [Tian and Huhns, 1986]. Subpixel accuracy is necessary to eliminate the quantization error introduced when the images are digitized.

The feature displacements are then used to fit a four-parameter similarity model defined by

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \mathcal{S} \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{2}$$

where (x_i, y_i) are the image frame coordinates at time t_i, $(\Delta x \ \ \Delta y)^t$ is the translation vector measured in the image coordinate system of the frame at t_i (f_i), Θ is the rotation angle between the two frames and $\mathcal{S}$ is the scaling factor. Notice that $\mathcal{S}$ is inversely proportional to the ratio of the distances between two arbitrary image points at times t_i, t_{i-1}. Thus $\mathcal{S}$ can be computed given a set of matched points from both frames, independently of the translation and rotation between them.

The scaling factor $\mathcal{S}$ is estimated first by computing the ratio of the distances in the feature sets relative to their center of mass. Assuming small rotation, the trigonometric terms in (2) can be linearized to compute the remaining translation and rotation parameters. A system of linear equations is then obtained by substituting all N matched feature pairs into the linearized equations. Each pair introduces two equations; hence the linear system has $2N$ equations and three unknowns (Θ, ΔX, and ΔY), and can be solved by a least-squares method.

The motion parameters obtained from the coarsest pyramid level are used to warp the next higher pyramid level of the previous Gaussian pyramid, and the process of tracking, estimation, and warping repeats until the highest-resolution image is reached. For an arbitrary pyramid level, the new estimate must be combined with the previous coarser-level estimate before warping the image at the next higher pyramid level (an initial zero motion is assumed for the coarsest level). Assuming that the total motion estimate from the coarser levels is $M_{i-1} = (\Delta x_{i-1}, \Delta y_{i-1}, \Theta_{i-1}, \mathcal{S}_{i-1})$ and the estimate for the current level is $m_i = (d_x, d_y, \theta, s)$, the new total motion estimate M_i used to warp the next higher resolution image can be easily derived to be [Zheng and Chellappa, 1993]

$$M_i = (\Delta x_i \Delta y_i \Theta_i \mathcal{S}_i)^T = \begin{pmatrix} s\cos\theta\Delta x_{i-1} - s\sin\theta\Delta y_{i-1} + d_x \\ s\sin\theta\Delta x_{i-1} + s\cos\theta\Delta y_{i-1} + d_y \\ \theta + \Theta_{i-1} \\ s \times \mathcal{S}_{i-1} \end{pmatrix} \tag{3}$$

2.2 Motion Compensation

The motion compensation module keeps a history of the inter-frame motion to remove what is unwanted and compute the warping parameters that will stabilize the current image frame. One of the advantages of electronic image stabilization systems is that motion can be compensated on demand, offering great flexibility by simply modifying some parameters of the compensation module.

The motion compensation module keeps track of the total combined motion, from the reference frame up to the current frame. When a new estimate is sent from the motion estimation module, the total motion is updated using (3).

Our system does not perform temporal smoothing on any of the motion parameters, but allows the user to dynamically mask (enable/disable) each parameter independently, for display purposes. For example, when the camera moves forward, producing a divergent image flow, the computed transformation includes a reduction in scale, which basically eliminates the perception of forward motion by producing a shrinking image with internal zero flow. The forward motion perception is restored by simply masking the scaling factor on display.

3 3D Image Stabilization Algorithm

The 3D model-based stabilization algorithm uses an extended Kalman filter (EKF) to estimate the rotation between frames, which is represented using unit quaternions. A small set of feature points is tracked as described previously, except that no estimation refinement is computed between pyramid levels, i.e., the features are simply scaled between levels, in a similar way to the greedy multi-resolution search implemented in [Morimoto and Chellappa, 1996].

3.1 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 corresponding image position of $\mathbf{P}$ (see Figure 3). 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

$$\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix} \tag{4}$$

where f is the camera focal length.

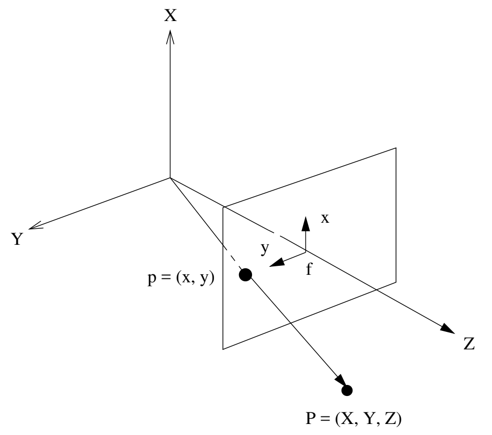

Figure 3: Diagram of the coordinate system fixed on the camera

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 $\mathbf{P}_0$ in the camera coordinate system at time t_0 has a new camera position at time t_1 given by

$$\mathbf{P}_1 = \mathbf{R}\mathbf{P}_0 + \mathbf{T} \Rightarrow$$
$$\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + \begin{pmatrix} T_X \\ T_Y \\ T_Z \end{pmatrix} \tag{5}$$

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

The projection of $\mathbf{P}_1$ can be obtained by combining (4) and (5):

$$\mathbf{p}_1 = \begin{pmatrix} f\frac{X_1}{Z_1} \\ f\frac{Y_1}{Z_1} \end{pmatrix} = \begin{pmatrix} f\frac{(r_{11}x_0 + r_{12}y_0 + r_{13}f + f\frac{T_X}{Z_0})}{(r_{31}x_0 + r_{32}y_0 + r_{33}f + f\frac{T_Z}{Z_0})} \\ f\frac{(r_{21}x_0 + r_{22}y_0 + r_{23}f + f\frac{T_Y}{Z_0})}{(r_{31}x_0 + r_{32}y_0 + r_{33}f + f\frac{T_Z}{Z_0})} \end{pmatrix} \tag{6}$$

For distant points ($Z_0 \gg T_X, T_Y$, and T_Z), the displacement is basically due to rotation, and thus (6) simplifies to

$$\mathbf{p}_1 = \begin{pmatrix} f\frac{(r_{11}x_0 + r_{12}y_0 + r_{13}f)}{(r_{31}x_0 + r_{32}y_0 + r_{33}f)} \\ f\frac{(r_{21}x_0 + r_{22}y_0 + r_{23}f)}{(r_{31}x_0 + r_{32}y_0 + r_{33}f)} \end{pmatrix} \tag{7}$$

The use of distant features for motion estimation and image stabilization has been addressed before in [Davis *et al.*, 1994; Durić and Rosenfeld, 1995; Yao *et al.*, 1995]. Such features constitute very strong visual cues that are present in almost all outdoor scenes, although it might 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).

3.1.1 Quaternions

Common ways to represent rotation include 3×3 orthonormal matrices, Euler angles, axis plus angle, and unit quaternions [Kanatani, 1990]. Young and Chellappa [Young and Chellappa, 1990] used quaternions for the problem of 3D motion estimation from noisy stereo sequences, and Horn [Horn, 1987] used quaternions to solve the absolute orientation problem from three or more point correspondences. Many other applications of quaternions can be found in photogrammetry, robotics and computer vision because of their compactness and good numerical properties, which facilitate the process of rotation estimation.

Quaternions are 4-tuples (q_0, q_x, q_y, q_z) that can be interpreted as complex numbers with one real (q_0) and three imaginary parts (q_x, q_y, q_z), as a scalar plus a 3D vector, or simply as a vector in 4D-space. To see how quaternions can represent rotations, consider a 3D unit sphere defined by $X^2 + Y^2 + Z^2 = 1$. The position of a point on the surface of the sphere can directly represent pan and tilt but not roll. By introducing a fourth parameter, it is now possible to represent an arbitrary 3D rotation by a point on a 4D unit sphere where $q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1$.

3.1.2 Relevant Properties of Unit Quaternions

In this section we present only a few basic properties of quaternions. More detailed treatments can be found in [Horn, 1987; Kanatani, 1990]. Consider a quaternion as composed of a scalar and a vector part, as in

$$\breve{\mathbf{q}} = q_0 + \mathbf{q}, \qquad \mathbf{q} = (q_x, q_y, q_z)^T \tag{8}$$

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 $|\breve{\mathbf{q}}| = \breve{\mathbf{q}}.\breve{\mathbf{q}} = q_0^2 + \mathbf{q}.\mathbf{q}$. Unit quaternions are simply defined as quaternions with unit norm.

A unit quaternion can be interpreted as a rotation θ around a unit vector $\mathbf{w}$ using the 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

$$\breve{\mathbf{q}}^* = q_0 - \mathbf{q} \tag{9}$$

and the multiplication of two quaternions as

$$\breve{\mathbf{r}} = \breve{\mathbf{p}}\breve{\mathbf{q}} = \begin{cases} r_0 = p_0 q_0 - \mathbf{p}.\mathbf{q}; \\ \mathbf{r} = p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}. \end{cases} \tag{10}$$

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 multiplication of quaternions expanded in matrix form as follows:

$$\breve{\mathbf{p}}\breve{\mathbf{q}} = \begin{pmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & -p_z & p_y \\ p_y & p_z & p_0 & -p_x \\ p_z & -p_y & p_x & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} \tag{11}$$

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

$$(0 + \mathbf{P}') = \breve{\mathbf{q}}(0 + \mathbf{P})\breve{\mathbf{q}}^* \tag{12}$$

Composition of rotations can be performed by multiplication of quaternions since

$$\begin{gathered}(0 + \mathbf{P}'') = \breve{\mathbf{q}}(0 + \mathbf{P}')\breve{\mathbf{q}}^* = \\ \breve{\mathbf{q}}(\breve{\mathbf{r}}(0 + \mathbf{P})\breve{\mathbf{r}}^*)\breve{\mathbf{q}}^* = (\breve{\mathbf{q}}\breve{\mathbf{r}})(0 + \mathbf{P})(\breve{\mathbf{q}}\breve{\mathbf{r}})^*\end{gathered} \tag{13}$$

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 (5) can be represented by the parameters of a unit quaternion simply by expanding (12) using (11), so that

$$\mathbf{R} = \begin{pmatrix} 1-2q_y^2-2q_z^2 & 2(-q_0q_z+q_xq_y) & 2(q_0q_y+q_xq_z) \\ 2(q_0q_z+q_xq_y) & 1-2q_x^2-2q_z^2 & 2(-q_0q_x+q_yq_z) \\ 2(-q_0q_y+q_xq_z) & 2(q_0q_x+q_yq_z) & 1-2q_x^2-2q_y^2 \end{pmatrix} \tag{14}$$

3.2 3D Motion Estimation

The dynamics of the camera is described as the evolution of a unit quaternion, and an IEKF is used to estimate the inter-frame rotation $\breve{\mathbf{q}}$. EKFs have been extensively used for motion estimation from a sequence of images [Broida and Chellappa, 1986; Yao *et al.*, 1995; Young and Chellappa, 1990]. In order to achieve real-time performance, this framework was simplified 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 will be represented using only the vector parameters. The remaining scalar parameter is computed from

$$q_0 = (1 - q_x^2 - q_y^2 - q_z^2)^{\frac{1}{2}} \tag{15}$$

Only nonnegative values of q_0 in (15) are considered, so that (14) can be rewritten using (q_x, q_y, q_z) only.

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

$$\left.\begin{aligned} \mathbf{x} &\overset{def}{=} \breve{\mathbf{q}} + \mathbf{n} \\ \dot{\mathbf{x}} &= \mathbf{0} \end{aligned}\right\} \Rightarrow \mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) \tag{16}$$

The following measurement equations are derived from (7):

$$\mathbf{z}(t_i) = \mathbf{h}_{i|i-1}[\mathbf{x}(t_i)] + \eta(t_i) \tag{17}$$

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

$$\begin{aligned} \hat{\mathbf{x}}(t_i^-) &= \hat{\mathbf{x}}(t_{i-1}^+) \\ \mathbf{\Sigma}(t_i^-) &= \mathbf{\Sigma}(t_{i-1}^+) + \mathbf{\Sigma}_{\mathbf{n}}(t_{i-1}) \end{aligned} \tag{18}$$

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

$$\begin{aligned} \mathbf{K}(t_i) &= \frac{\mathbf{\Sigma}(t_i^-)\mathbf{H}_{i|i-1}^T}{\mathbf{H}_{i|i-1}\mathbf{\Sigma}(t_i^-)\mathbf{H}_{i|i-1}^T + \mathbf{\Sigma}_\eta(t_i)} \\ \hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\{\mathbf{z}(t_i) - \mathbf{h}_{i|i-1}[\hat{\mathbf{x}}(t_i^-)]\} \\ \mathbf{\Sigma}(t_i^+) &= [\mathbf{I} - \mathbf{K}(t_i)\mathbf{H}_{i|i-1}]\mathbf{\Sigma}(t_i^-) \end{aligned} \tag{19}$$

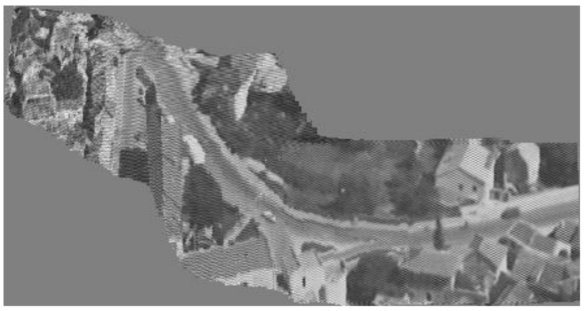

Figure 4: Mosaic from the 2D fast stabilization algorithm

where $\mathbf{K}(t_i)$ is a $3\times N$ matrix which corresponds to the Kalman gain, $\mathbf{\Sigma}_\eta(t_i)$ 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

$$\mathbf{H}_{i|i-1} = \left.\frac{\delta \mathbf{h}_{i|i-1}}{\delta \mathbf{x}(i)}\right|_{\hat{\mathbf{x}}(t_i^-)} \qquad (20)$$

A batch process using a least-square estimate of the rotational parameters can be used to initialize the algorithm, as in [Yao *et al.*, 1995], 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 [Jazwinski, 1970; Maybeck, 1982].

4 Fast 2D Mosaicking Implementation

The goal of creating a mosaic image is to represent a scene which has a larger field of view than the input sensor. This may be accomplished by combining individual frames which depict portions of the overall scene. By appropriately pasting these frames together, we can generate a panoramic view of the original scene. We refer to this procedure as mosaicking.

Through the use of the motion stabilization technique described in Section 2, it is possible to not only compensate for inter-frame motion, but also to keep track of the motion parameters which represent image frame changes with respect to a global coordinate system. We are then able to paste each incoming frame into our global picture. This process generates a mosaic of the imaged scene.

A fast mosaicking system for PREDATOR video data was implemented as an extension to the stabilization system, and runs on the same platform. This extended system can be broken down into two distinct processing threads: a real-time thread for performing the image processing in conjunction with the Datacube card, and a mosaic thread which controls the display and user interface. The two threads communicate with each other through UNIX pipes. By using an X-Windows interface, the actual display head for the mosaic may be thousands of miles from the processor. This may be accomplished by setting the display head to be any X-Windows-compatible terminal which is networked to the mosaic processor.

The last stage of the real-time thread warps the highest pyramid level of the current frame to be an integer offset from the global coordinate frame. This warp is accomplished on the Datacube through the use of an affine transform, and removes all scale, rotation, and fractional pixel shifts. This warped image is then sent to the display process along with its location in the global system. The display process starts with a blank global image, and adds each incoming image segment to this image.

In our current implementation of the mosaicking process, the global image is treated as a write-once medium which is unlimited in size. There are two reasons behind this. The first is that the error in the stabilization algorithm is cumulative. Therefore, if the camera dwells on a particular area for an extended period of time, image drift which distorts the mosaic is noticeable. By providing a write-once memory, mosaic discontinuities are confined to a single line in the image. This makes for a much more viewable mosaic. The second reason for the write-once memory is to reduce the display channel bandwidth. It is possible to foresee applications where the image processor may be located at a remote ground (or air) station and may only be connected to the user's console by a low-bandwidth link. We hope to show that our mosaic creation technique allows for realistic scene generation while consuming little communication bandwidth.

In order to avoid running out of room for the generation of a mosaic, the global image is treated as if it were unlimited in size. In reality, the image storage size is constrained by the user, and the image is scrolled to maintain the current view on the screen. Areas which scroll off the edge of the display are lost.

5 Experimental Results

Since still images are not the most appropriate way of displaying the results of a dynamic process such as stabilization, we have made the original, stabilized and mosaicked sequences available in MPEG format at **http://www.cfar.umd.edu/~carlos/IUW97.html**. In the following sections, an MPEG file at this address named *Mosaic* will be referenced by http:Mosaic.

5.1 2D Stabilization and Mosaicking Results

This section presents experimental results from the fast 2D stabilization and mosaicking system applying to PREDATOR video data. Our data tape has been through several recording generations, and is of moderate to poor quality.

Figures 4 and 5 show mosaic images from PREDATOR video data using the mosaicking process described above. Reliable image stabilization requires large overlap between image frames and reasonably high frame rates. Fortunately this is significantly less important for the display thread. Since the mosaic is treated as write-once memory, overlapping areas of the image are ignored. Therefore, it is necessary that there be only a small overlap between frames in order to provide a continuous mosaic. This allows us to discard frames from the real-time thread without displaying them. Running on the hardware described above, our real-time thread was able to run at approximately 10 frames per second. The display thread was set up to process every fourth image which was generated by the real-time thread. The rest of the images were discarded.

The images used to generate the mosaic were from the top level of the image pyramid. This corresponds to an image resolution of 128x120 pixels. The bandwidth which would be necessary to transmit the mosaic in real time is image sequence dependent. The use of the write-once memory dictates that the required bandwidth is directly related to the amount of new information contained in each mosaic frame. For the mosaics shown in Figures 4 and 5, we recorded an average bandwidth of 15993 bytes per second, for a two frame per second mosaic update rate. This is an 89% improvement over sending the entire raw sequence.

5.2 3D Stabilization and Mosaicking Results

Figure 6 is an example of the results from our real-time 3D de-rotation 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 fifth frame from the input sequence (left) and its corresponding stabilized frame (right). The original and stabilized sequences are available at http:OffRoad3DStabilization. The difference between the fifth and tenth input frames 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. In this example, most of the features lie on the horizon, so that the horizon is particularly well stabilized. Errors are bigger around objects that are closer to the camera (darker regions), since they have large translational components which are not compensated by this method.

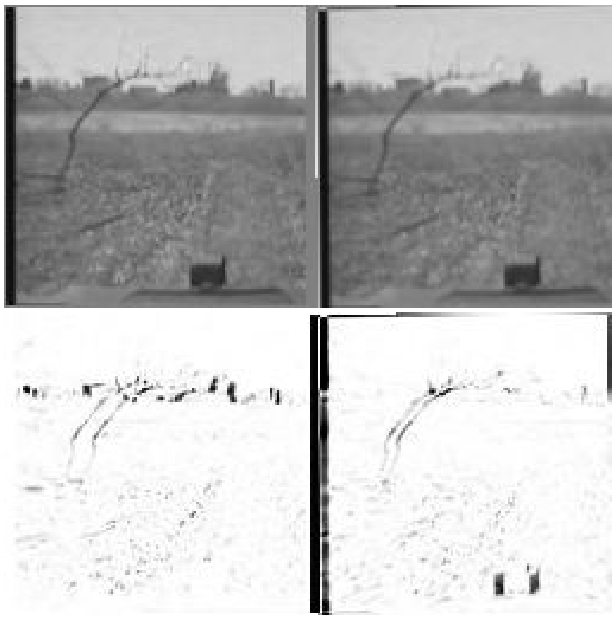
Figure 6: Image stabilization results. The top row contains the fifth frame (left) and its corresponding stabilized frame (right). The bottom row shows the difference between two frames of the input sequence (left) and the difference of the corresponding stabilized frames (right). Stabilization minimizes the difference in regions close to the horizon.

The real-time implementation typically detects and tracks nine features with a maximum feature displacement of 15 pixels. Under these settings, the system is able to process approximately 9.8 frames per second.

Figure 7 compares the mosaic images from the 2D and 3D models. They both use the same set of 11 feature points for motion estimation, and the 3D system selects the four points which best approximates the current rotation estimate to update its state.

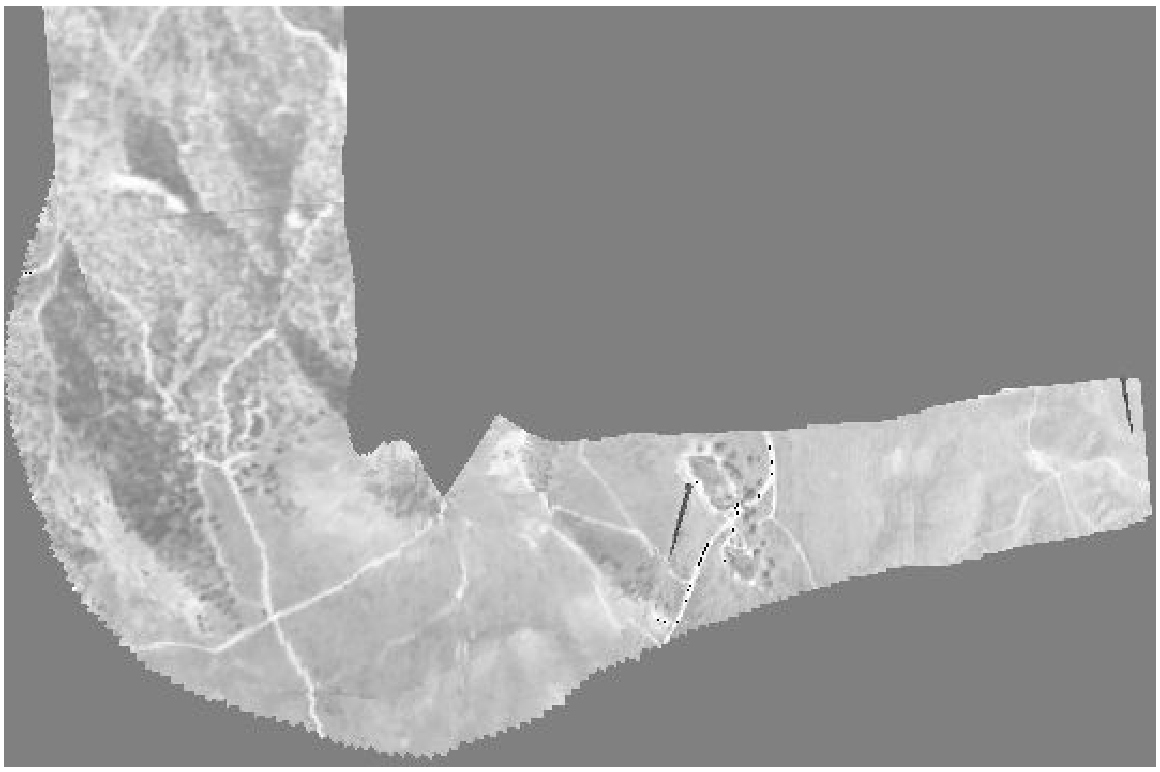

Figure 5: Mosaic from the 2D fast stabilization algorithm

The original sequence is composed of 200 frames and the dominant motion is left-to-right panning. To help 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 degrees. The mosaic from the 2D estimates (http:2DMosaic1) does a good job locally, but the 3D mosaic (http:3DMosaic1) looks much more natural, as if it were a panoramic picture taken using a fish-eye lens. The original sequence and the 3D stabilized output can be seen at http:3DStabilization1.

Figure 8 shows a second comparison between 2D and 3D mosaics. The original sequence is composed of 150 frames and the dominant motion is right-to-left panning. The reference frame was assumed to be the 75th frame, and appears at the center of the mosaics. The top row shows the 70th and the last frame of the sequence, from left to right. The second row shows the 2D mosaics constructed up to the corresponding frames in the first row, and the bottom row shows the corresponding mosaics using the 3D models and using the same camera parameters that were used to generate Figure 7. The original sequence and the mosaics can be viewed at http:UGVsequence, http:2DUGVMosaic, and http:3DUGVMosaic.

6 Conclusion

We have presented in this paper a fast electronic image stabilization and mosaicking system based on a two-dimensional feature-based multi-resolution motion estimation algorithm, that tracks a small set of features to estimate the motion of the camera. Stabilization is achieved by combining all motion from a reference frame and subtracting this motion from the current frame. Mosaics are constructed in real time by directly aligning new frames with the current mosaic. The system was implemented on a Datacube MaxVideo 200 board connected to a Themis 10MP. Preliminary tests using PREDATOR video data demonstrate the robustness of the system, which is able to process 10 frames per second and handle displacements of up to ± 12 pixels between consecutive frames.

We have also 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

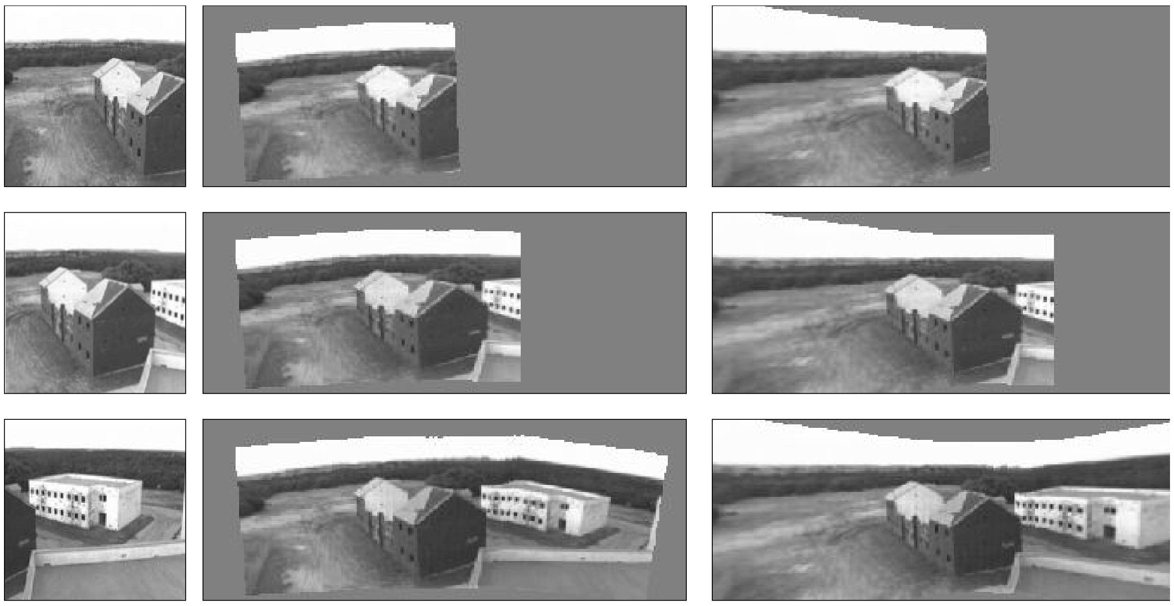

Figure 7: 2D and 3D mosaics from 200 frames of a panning sequence. The leftmost column shows the 50th, 100th, and 200th frames from the input sequence. The second and third columns show the corresponding 2D and 3D mosaics,

quaternions, whose good numerical properties contribute to the overall performance of the system. The system was implemented on the same platform and it is able to process 128×120 images at approximately 10 Hz.

References

[Balakirsky, 1995] S. Balakirsky. Comparison of electronic image stabilization systems. Master's thesis, Department of Electrical Engineering, University of Maryland, College Park, 1995.

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

[Burt and Anandan, 1994] P. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. In *Proc. DARPA Image Understanding Workshop*, pages 425–434, Monterey, CA, November 1994.

[Davis *et al.*, 1994] L.S. 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.

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

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

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

[Irani *et al.*, 1994] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, Seattle, WA, June 1994.

[Irani *et al.*, 1995] M. Irani, S. Hsu, and P. Anandan. Mosaic-based video compression. In *Proc. SPIE Conference on Electronic Imaging, Digital Image Compression: Algorithms and Techniques*, volume 2419, pages 242–253, San Jose, CA, 1995.

[Jazwinski, 1970] A. Jazwinski. *Processes and Filtering Theory.* Academic Press, New York, 1970.

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

[Kwon *et al.*, 1995] O.J. Kwon, R. Chellappa, and C.H. Morimoto. Motion-compensated subband coding of video acquired from a moving platform. In *Proc. IEEE International Conference on Acoustics, Speech, and Sig-*

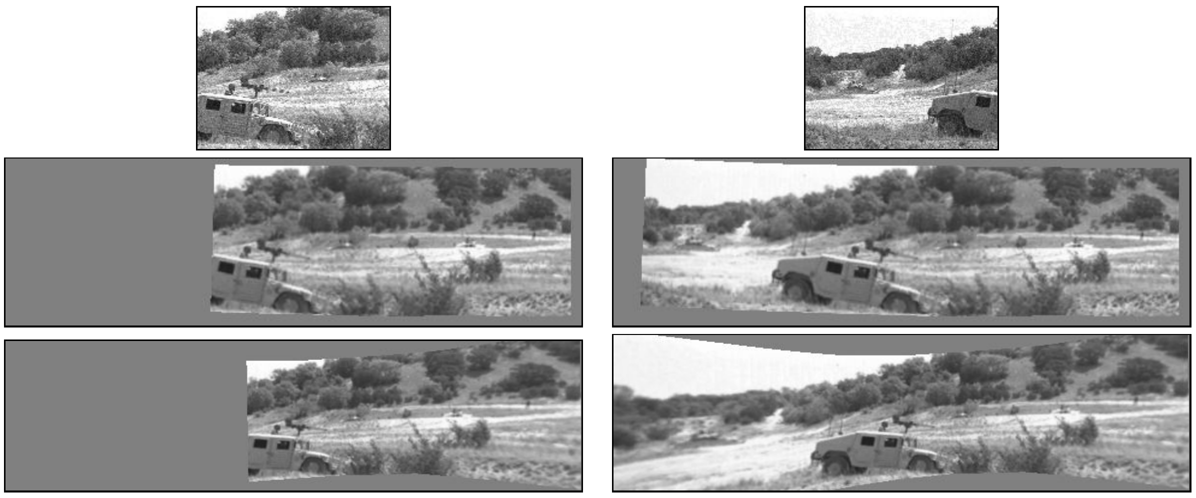

Figure 8: 2D and 3D mosaics from 150 frames of a panning sequence. The top row shows the 70th and the last frame of the sequence. The second row shows the 2D mosaics constructed up to the corresponding frames, and the bottom row shows the corresponding mosaics using the 3D models and guessed camera parameters.

nal Processing, pages 2185–2188, Detroit, MI, January 1995.

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

[Morimoto and Chellappa, 1996] C.H. Morimoto and R. Chellappa. Fast electronic digital image stabilization. In *Proc. International Conference on Pattern Recognition*, volume III, pages 284–288, Vienna, Austria, August 1996.

[Morimoto *et al.*, 1995] C.H. Morimoto, D. DeMenthon, L.S. Davis, R. Chellappa, and R. Nelson. Detection of independently moving objects in passive video. In I. Masaki, editor, *Proc. Intelligent Vehicles Workshop*, pages 270–275, Detroit, MI, September 1995.

[Morimoto *et al.*, 1996] C.H. Morimoto, P. Burlina, R. Chellappa, and Y.S. Yao. Performance analysis of model-based video coding. In *Proc. International Conference on Image Processing*, volume III, pages 279–282, Lausanne, Switzerland, September 1996.

[Sawhney *et al.*, 1995] H. Sawhney, S. Ayer, and M. Gorkani. Model-based 2D and 3D dominant motion estimation for mosaicking and video representation. In *Proc. International Conference on Computer Vision*, pages 583–590, Cambridge, MA, June 1995.

[Tian and Huhns, 1986] Q. Tian and M.N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics and Image Processing*, 35:220–233, 1986.

[Viéville *et al.*, 1993] T. Viéville, E. Clergue, and P.E.D.S. Facao. Computation of egomotion and structure from visual and internal sensors usig the vertical cue. In *Proc. International Conference on Computer Vision*, pages 591–598, Berlin, Germany, 1993.

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

[Young and Chellappa, 1990] 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*, 12:735–759, 1990.

[Zheng and Chellappa, 1993] Q. Zheng and R. Chellappa. A computational vision approach to image registration. *IEEE Trans. Image Processing*, 2:311–326, 1993.