Real-time Scene Stabilization and Mosaic Construction

M. Hansen, P. Anandan, K. Dana, G. van der Wal, P. Burt
David Sarnoff Research Center
CN 5300
Princeton, NJ 08543

Abstract

We describe a real-time system designed to construct a stable view of a scene through aligning images of an incoming video stream and dynamically constructing an image mosaic. This system uses a video processing unit developed by the David Sarnoff Research Center called the Vision Front End (VFE-100) for the pyramid-based image processing tasks required to implement this process. This paper includes a description of the multiresolution coarse-to-fine image registration strategy, the techniques used for mosaic construction, the implementation of this process on the VFE-100 system, and experimental results showing image mosaics constructed with the VFE-100.

1 Introduction

Imagery obtained from a sensor mounted on a moving platform frequently requires stabilization in one form or another. In some situations, it is necessary to correct for sensor unsteadiness to enhance the imagery for viewing purposes, as can occur when the sensor platform is vibrating or bouncing rapidly. Automated vision processes, such as moving target detection, typically require camera-induced motion to be removed in order to operate effectively. Other applications, such as surveillance, can utilize stabilized imagery both for image enhancement and for automated mapping and panoramic display construction.

One traditional solution to the image stabilization problem is to use mechanical stabilizers based on accelerometers, gyros, or mechanical dampers. While these techniques are useful for handling large motions caused by unstable sensor platforms, these systems are typically not precise. As a result, even after mechanical stabilization there may be significant residual image motion. Also, as the stabilization performance of the mechanical systems increase, they tend to get physically bulky and very expensive. These shortcomings have led to the use of electronic stabilizers, which can offer potentially more precise stabilization at a reduced cost without the need for mechanical apparatus. The commercial "steady cam" systems available in consumer video recorders are a good example of the lower end of electronic stabilizers which typically compensate for only certain types of translational image motion with low precision (e.g., [1]).

We describe a system for real-time stabilization that can remove full first-order (affine) deformations between images in a sequence, and assemble these aligned images within a single reference coordinate system to produce an image mosaic. This system performs stabilization using pyramid-based motion estimation and image warping. In order to achieve real-time performance at a minimal hardware cost, this system has been implemented using a pyramid-based image processing system called the Vision Front End (VFE), which is specially tailored for such applications.

This paper describes both the algorithm and hardware implementation of the scene stabilization system. Section 2 provides an overview of the affine image registration and mosaic construction process, then describes the goals of the scene stabilization system and the strategies used to construct the stabilized scene from the incoming video stream. The affine image registration process itself is described in detail in Section 3. An overview of the VFE-100 video processing system is provided in Section 4, along with further explanations of the scene stabilization implementation on the VFE-100. Finally, Section 5 shows experimental results using this system from video sequences taken from moving aerial and ground-based imagery, and Section 6 provides conclusions and comments about the system as well as areas for further research.

2 Stabilization Overview

The goal of this system is to provide a stable view of a scene from an unstable incoming video stream provided by an imaging sensor in motion. The two tasks in this process are (1) to estimate and correct for any affine deformations that exist between successive images in the video stream, and (2) to combine these aligned images into a suitable representation for display.

A simple form of scene stabilization can choose an arbitrary video frame as defining a reference coordinate

0-8186-6410-X/94 $4.00

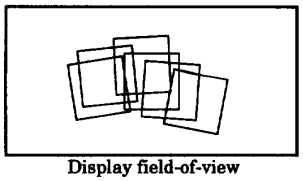

Figure 1: Stabilized scene construction using an image mosaic.

system. Subsequent frames are then aligned to one another, and the affine transformations are cascaded from one frame to the next, yielding the appropriate transformation between that video frame and the reference. Then, each frame is warped to align with the reference and displayed. The net result of this simple stabilization display process will be imagery that appears motionless, except for border effects that will occur as a result of warping the imagery. Scene contents that were not visible in the reference frame will not be displayed in this scenario—the field-of-view of the stabilized video will always be restricted to the field-of-view of the reference frame. Also, the display areas will be undefined in areas where the incoming imagery does not overlap with the reference frame.

If interframe translational motion is large, the simple stabilization display described above will not be adequate to provide a meaningful stable scene. Large motions can occur when a camera is mounted on a ground-based vehicle moving over rough terrain, for example, or when the sensor is deliberately scanning, as would be the case for certain surveillance applications. In these situations, it is advantageous to modify the display in order to (1) keep the original (reference) field-of-view visible and updated with the most recent image information available, and (2) display the stable scene in a manner that all (or at least a significant amount) of the imagery that has been stabilized is visible simultaneously. This leads to an image mosaic representation as shown in Figure 1.

The image mosaic is a display that is comprised of many images aligned to a fixed reference coordinate system. As in the case of simple stabilization, an arbitrary frame is chosen to define the 2D coordinate system for the stabilized scene. All further imagery is warped and aligned to the reference coordinate system, then inserted into the mosaic at the appropriate location.

The mosaic representation is superior to the simple stabilization display in several respects. It provides an expanded field-of-view display which can in many situations encompass all of the imagery acquired in the video stream into a single representation. The reference field-of-view of the scene is always visible at the center of the mosaic, so the original reference point for the stabilized scene is not lost. If large translational motion occurs within the video changing the field-of-view significantly from the reference field-of-view, the mosaic representation expands the field-of-view rather than having to select one over the other.

3 Affine motion estimation

At the heart of the scene stabilization system is the technique to accurately estimate interframe motion. In order to support mosaic construction over many frames in real-time, the alignment must be within subpixel accuracy and the estimation must be fast.

We use a multiresolution, iterative process operating in a coarse-to-fine fashion on Laplacian pyramid images to estimate the affine motion parameters (see Figure 2). Under user control, the system can be made to compensate for any first-order motion transformation, such as translation, rotation, dilation, etc. During each iteration, an optical flow field is estimated between the images through local cross-correlation analysis, then an affine (or other first-order) motion model is fit to the flow field using weighted least-squares regression. The estimated affine transform is then used to warp the previous image to the current image and the process is repeated between them.

3.1 Cross-correlation computation

Figure 3 describes the cross-correlation computation process based on the Laplacian pyramid images. Let L_C denote the spatial resolution with $L_C = 0$ being the highest resolution and $\mathbf{L}_{L_C}$ denote the Laplacian image at resolution L_C. (All images throughout the paper will be referenced with boldface type.) After one of the Laplacian images $\mathbf{L}_{L_C}[t-1]$ has been prewarped by the motion estimate from the previous iteration (or with zero motion if the current iteration is the first) the images are shifted with respect to one another and image-wise multiplied, yielding a product image. For a shift of (i,j), the product image $\mathbf{I}_{i,j}$ is defined as

$$\mathbf{I}_{i,j} = \mathbf{L}_{L_C}[t](x,y)\mathbf{L}_{L_C}[t-1](x+i,y+j) \quad (1)$$

with $i,j \in [-N,N]$. Integrating the image $\mathbf{I}_{i,j}$ fully will yield the cross-correlation value $C_{i,j}$ between the two full Laplacian images at shift (i,j). Local cross-correlation values can be computed by integrating each product image $\mathbf{I}_{i,j}$ over local patches to yield cross-correlation "images" of the form $C_{i,j}(x,y)$. However, in order to avoid border effects and make the results most representative of the information at the centers of the local patches, a weighted integration function $W(x,y)$

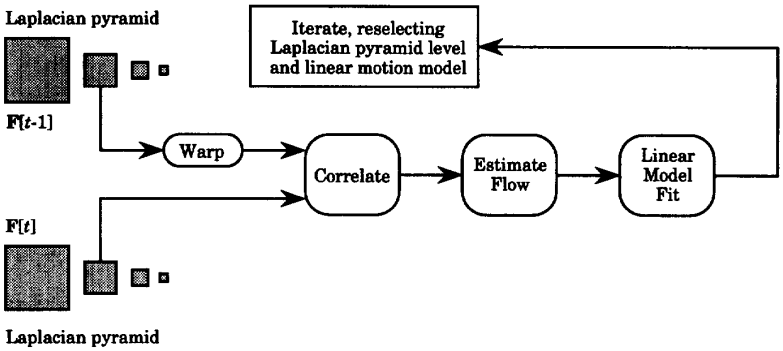

Figure 2: Motion estimation process overview.

is preferred over simple neighborhood averaging for local integration [2]. Therefore, the values for $C_{i,j}(x,y)$ can be computed from the product image $\mathbf{I}(i,j)$ with

$$C_{i,j}(x,y) = \mathbf{I}_{i,j}(x,y) \otimes W(x,y) \qquad (2)$$

where $W(x,y)$ is the integration weighing function and $\otimes$ denotes convolution. (Note that the image $C_{i,j}(x,y)$ is not denoted with bold type, because $C_{i,j}(x,y)$ is referred to both as an image and as a cross-correlation function.)

The convolution with the kernel $W(x,y)$ (typically a Gaussian) has the effect of smoothing the product images $\mathbf{I}_{i,j}$ into the cross-correlation images $C_{i,j}$. Depending on the size of this kernel, the resulting $C_{i,j}$ will be oversampled to various degrees. Therefore, estimating flow based on analysis of $C_{i,j}(x,y)$ directly will result in correspondingly oversampled flow field. In order to keep computational costs to a minimum, a pyramid reduction process for the integration (see Figure 3) is used instead of performing convolutions of the product images at the correlation resolution level L_C. Different numbers of pyramid reduction steps can be used to achieve integration over correspondingly different spatial areas, with each pyramid level generated at an integration step reducing the size of the flow field (and the computational costs associated with this flow field) by a factor of four.

The critical parameters for the local cross-correlation process are: the spatial resolution level L_C used for the Laplacian images, the half-width N of the correlation search, and the spatial resolution L_i chosen for integration, where $L_i = L_C +$ the number of integration levels used.

The value of L_C (the resolution level used for the correlation) determines the spatial frequency band used for the motion estimation, and hence the motion that will be detected during this iteration. A single pixel shift at level L_C corresponds to a shift of 2^{L_C} at the highest resolution level. This shift dictates the overall range and precision of the estimates yielded from analyzing at this resolution level.

The size of the correlation search area N determines the maximum displacement (range) that can be estimated at spatial resolution L_C. Although larger values of N allow for a larger range of motions to be estimated, the potential for false matches also increases. Also, there is a quadratic increase in the attendant computational costs. Therefore, in practice, the values for N are restricted to 1 or 2.

The level of integration, L_i, determines two things. First, it determines the amount of smoothing that has been performed on the correlation results. More smoothing leads to better signal-to-noise characteristics, but will correspondingly result in poorer estimates of the spatial location of the correlation peak. More significantly, it makes the implicit assumption that the correlation values (and the flow field) within the area of integration are locally smooth, which may not be the case everywhere in the image. Also, L_i determines the size of the resulting flow field, since a flow estimate can be made for each position in the integrated correlation image excluding the borders. Therefore, the integration level L_i is chosen just large enough to provide the necessary support for reliable and accurate flow estimates.

Since the Laplacian is a signed image with approximately zero local mean values, the correlation surface has both positive and negative values. This is similar to using a type of mean normalized correlation on Gaussian blurred images. Note also that the magnitudes of the correlation values are not normalized. While such normalization is completely within the scope of the algorithm, in practice this increases the burden on the computation and, as demonstrated in [3], the resulting increase in the accuracy of the flow field is small.

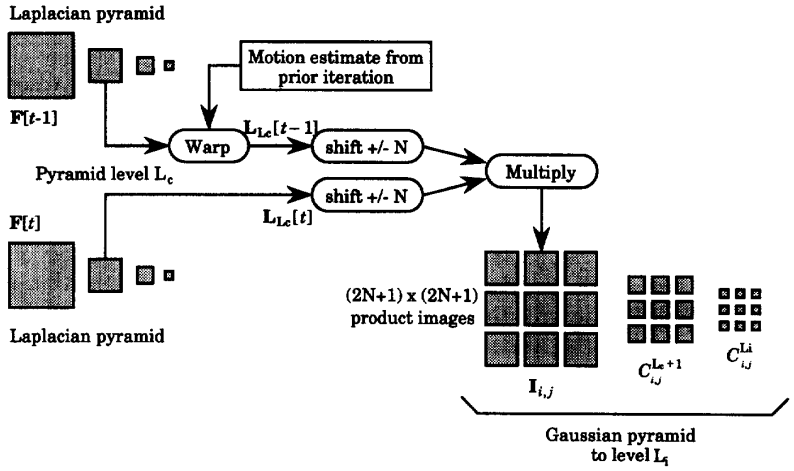

Figure 3: Local cross-correlation computations used in the motion estimation process.

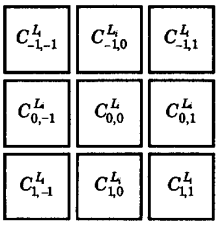

Figure 4: Local cross-correlation surface ($N = 1$) for a given position based on the integrated cross-correlation images after filtering and subsampling.

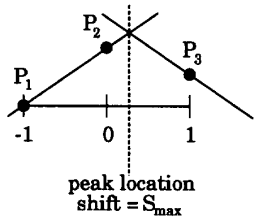

Figure 5: Vector estimation (1-D) from three correlation points using the symmetric triangle model to locate the correlation peak.

Considering this, correlation normalization was not included with the current algorithm.

3.2 Flow estimation

The flow estimation is based on an analysis of the cross correlation surface defined by the cross-correlation values $C_{i,j}^{L_i}(x,y)$ for $i,j \in [-N,N]$ at a given location (x,y) (see Figure 4). The flow value at (x,y) is determined through locating the peak in this correlation surface. The peak-detection process depends on the search range N. Below we describe the method used for analysis of the 3×3 correlation surface. In addition, we use the shape of the cross-correlation surface provides to determine uncertainty in the flow estimate [4].

The cross-correlation surface for the $N = 1$ (3×3 correlation surface) case is shown in Figure 4. Assume that the correlation peak is somewhere within 1/2 a pixel of the center of the correlation surface. In order to obtain subpixel flow estimates, we use a type of interpolation of the available values to determine the peak.

A further reduction in computation is achieved by separately estimating the horizontal and vertical locations of the peak. The one-dimensional horizontal and vertical correlation curves that are generated based the Laplacian of Gaussian can be modeled as a symmetric triangle as shown in Figure 5—see [5] for a related analysis. Using this model, the equation for computing the 1D shift value S_{max} from 3 correlation values P_1, P_2, P_3 is computed with

$$S_{max} = \frac{P_3 - P_1}{2(P_2 - \text{MIN}(P_3, P_1))}$$

What if the assumption of the shift being less than 1/2 a pixel away from the center does not hold? A discrete second derivative can be computed about the center of the correlation surface to determine if the correlation data is suitable for this sort of interpolation. Peaks at a shift of greater than a full pixel will usually result in no maximum being detected at the center of the surface. Shifts of greater than 1/2 a pixel but less

than 1 pixel will pass the second derivative test and can be interpolated, but the shift estimates in this case will not be as accurate as for the measurements for pixels at less than 1/2 pixel shift.

Using the P_1, P_2, P_3 nomenclature, a test using the second derivative about the center of the surface along one of the lines through the surface is given by $T = 2P_2 - P_3 - P_1$. If $T < 0$, then there is no maximum detected around that position in the correlation surface and no vector should be estimated for that point. Likewise, the diagonally-oriented lines on the 3×3 surface passing through the center should be checked using this same test: if one of the diagonal orientations does not show a peak about the center, then no vector should be estimated for that location.

Our current implementation is restricted to binary valued confidence measure, based on the aforementioned test on the sign of T. Although a continuous valued confidence measure would be desirable, our observation has been that practice it is more important to remove the bad flow estimates altogether, than to have a continuous weighting. In general, the relationship between the derivative measure such as T and the actual uncertainty associated with the flow vectors is highly non-linear. We are currently exploring various strategies for this "normalization" process.

3.3 Flow field regression

The final step of the alignment process is to fit the flow field to a linear, first order motion model. A least-squares regression is used to fit the flow field to these models. The vector confidence values are used to weigh each vector's influence to the regression—when vectors have 0 valued confidences, these vectors do not contribute to the regression at all, while positive-valued confidences allow the vectors to contribute to the regression in a weighted manner. As noted earlier, the current algorithm restricts the confidence measures to be binary valued.

One of the more important considerations to this algorithm is the type of motion model to use with the flow field during this regression step. A linear, first-order motion model, for example, allows for motion up to an affine distortion. In many cases, however, full affine distortions do not occur during the video sequence—a simple translation, or translation and rotation, or a similarity transform may be adequate to fully describe the motion occurring within the scene. If prior information can be used to restrict the motion model, it is useful to do so in order to avoid instabilities and ambiguities during estimation.

3.4 The affine registration process: practical considerations

In actual operation, the affine registration process is configured to perform coarse-to-fine, progressive complexity motion estimation using various motion models. A table of the different parameters used to configure the registration process to perform affine registration is shown in Table 1.

This particular configuration of parameters shows the basic strategies that can be used for the image registration, and may be varied depending on the assumptions that can be made about the scene and the motion. Note that the entire process is performed in 5 iterations. The first two iterations are performed correlating images at $L_C = 5$ and $L_C = 4$, integrating to level $L_i = 6$ and $L_i = 5$. The first iteration uses a cross-correlation half-size of $N = 2$ (5×5 correlation surface) while the second iteration performs the correlations using $N = 1$ (3×3 correlation surface). The goal of the first iteration is to identify very large displacements at level 5, then iterate again at level 4 to get a better estimate of the displacement. Also note that the coarse level estimation uses only a translational motion model. This is because often the effects of the linear terms are small at coarse resolution; moreover, the reliable estimation of the linear terms requires higher-frequency spatial features.

Once a rough estimate of the translational shift in the imagery has been determined at level $L_C = 5$ and 4, the processing is performed at levels $L_C = 3$ and $L_C = 2$, and an affine model is used for the motion estimation. At these resolution levels, typically there is enough structure present to perform more complex motion estimation from the flow fields determined. Note that, for all later iterations, the same integration level is used. The 16×15 flow field computed from $L_i = 5$ images is normally sufficient for all types of first-order motion and the computational requirements for a flow field of this size is modest.

It is worth noting that the registration algorithm never proceeds beyond $L_C = 2$. This restriction is imposed because our desired accuracy is achieved at $L_C = 2$ and additional processing time is unnecessary.

4 Real-time implementation

As mentioned earlier, the real-time scene stabilization system was designed based on a video processing system called the VFE-100, a commercially available image processing platform currently being sold by Sensar, Inc. The VFE-100 was specifically designed to perform multiresolution (pyramid) processing tasks such as im-

Iteration Number	Correlation Resolution L_C	Integration Resolution L_i	Correlation Search Area N	Motion Model Used
1	5 (16 × 15)	6 (8 × 7)	2 (5 × 5 area)	Translation only
2	4 (32 × 30)	5 (16 × 15)	1 (3 × 3 area)	Translation only
3	3 (64 × 60)	5 (16 × 15)	1 (3 × 3 area)	Affine
4	3 (64 × 60)	5 (16 × 15)	1 (3 × 3 area)	Affine
5	2 (128 × 120)	5 (16 × 15)	1 (3 × 3 area)	Affine

Table 1: Description of parameters to perform affine registration using the described algorithm. Resolution levels are shown as pyramid levels followed by the actual image size in parenthesis. The original image was 512 × 480 pixels in size.

age registration at real-time video rates. The VFE-100 is based on a prototype system that was developed by the David Sarnoff Research Center under contract with the US Army Missile Command (MICOM) to perform image alignment and motion detection for missile tracking applications. This prototype was augmented and commercialized by Sensar, Inc. into the more general-purpose image processing platform that was used to implement the scene stabilization system.

4.1 VFE-100 architecture

Most of the processing functions present within the VFE-100 system are based on pyramid techniques. To obtain the processing power required to perform many pyramid operations quickly, the VFE-100 contains six proprietary VLSI chips called PYR-1 pyramid chips designed to perform the image convolutions and other arithmetic operations required for generating a variety of different image pyramids and reconstructing imagery based on those pyramids.

The overall block diagram of the VFE-100 system architecture is shown in Figure 6. The VFE-100 consists of a stereo digitizer, a number of frame stores, an affine warper, six PYR-1 pyramid chips, three image correlators, an ALU, and a display frame store with color graphics overlay. All of these devices are connected with a 16 × 16 crosspoint switch, which enables the processing units to communicate flexibly. Timing streams are provided with each data stream within the VFE-100, making the interconnections and processing between VFE-100 devices completely independent of the operations of other devices.

Hardware sequencing and other image processing tasks that cannot be performed in hardware are handled by an on-board TI TMS320C30 DSP. Software is written for the VFE-100's C30 using standard ANSI compatible C programs. A special hardware control library provides a high-level interface for accessing the VFE-100 hardware from C programs without having to program hardware registers or other operations directly. The software run on the VFE-100 is written, compiled, and loaded into the VFE-100 through a host computer (currently either an IBM-PC or Sun SPARC) which can communicate with the C30 processor through a standard VME interface.

4.2 Scene stabilization with the VFE-100

The VFE-100 architecture was designed with this image alignment application in mind; therefore, this algorithm can be implemented on the VFE-100 efficiently. In fact, with the exception of the vector estimation and flow field regression steps of the alignment algorithm, all of the required image processing tasks can be performed using the VFE-100 hardware.

The utilization of the VFE-100 hardware is as follows. The warper is used when images are warped both for the alignment (once for each iteration of processing) and for the final mosaic construction. The pyramid modules are used to generate the Laplacian pyramids used for the alignment and the Gaussian pyramids used for constructing the image mosaic. The correlation modules compute the product images (for the various shifts) between the two Laplacian images, with three of these modules present to make the correlation process even faster. The pyramid chips present in the correlation modules are there for performing the pyramid-based integration after the cross-correlation multiplication step.

The C30 DSP, as mentioned before, is responsible for sequencing the image processing hardware. In addition to that, however, the C30 also takes care of all computations that cannot be performed completely with the video hardware. For the scene stabilization algorithm, the most computationally intensive tasks for the C30 are the estimation of the flow field by analyzing the cross-correlation images generated by the correlation modules, and fitting a linear model to the estimated flow field. In addition, the C30 is responsible for keeping track of the various affine coefficients used in the alignment and estimation process, the mosaic construction information, and for other bookkeeping tasks required for this process.

5 Experimental results

Demonstrating the efficacy of the scene stabilization algorithm in hard copy form is extremely difficult because it is based on the processing of live video. Here, in order to show the idea of scene stabilization through automated mosaic construction, a small subset of frames from a video sequence has been provided, along with the mosaic that was generated from the (entire) sequence. The sample image frames provided in the figures are taken approximately 5 seconds apart. The scene stabilization system used to generate these results performed at a processing speed of 10 Hz on the VFE-100 system.

The video sequence is aerial photography of varying terrain, shown in Figure 7. The camera itself moves with the consistent translational motion of the aircraft and also suffers occasional rotations. Using the affine alignment configuration described in Table 1, all translational and rotational motion has been compensated for, and a panoramic view of the scenes from the first to the last frame has been constructed using the mosaic representation.

6 Conclusions

The real-time scene stabilization system described in this paper performs two basic functions. First, the system accurately estimates the affine deformations between images comprising a video sequence with low computational cost. Second, it compensates for these deformations and constructs an image mosaic of the sequence providing a stable scene in a single 2D coordinate system. The implementation of this system using the Sensar VFE-100 operates with a throughput of 10-15 Hz.

From this initial real-time scene stabilization system, there are many avenues for expanding research. Ongoing research by the authors and others is focused on solving the case of scene stabilization where there is little if any interframe overlap in the video sequence. To solve this extreme stabilization case, a reference mosaic will be constructed not only for display, but also used for alignment. The mosaic itself can be expanded using a technique called tiling, which divides the initial mosaic into a group of smaller mosaics that are linked through affine transformations [6].

Other research is focusing on better mosaic storage and construction. In the current implementation, the image mosaic must be shown at a reduced resolution because of the display size and frame store limitations. Using a much larger display, a "viewport" can be used to scroll and display portions of a much larger, full resolution mosaic. During mosaic construction, rather than just inserting aligned frames into the mosaic, these frames can be merged into the mosaic using multiresolution interpolation techniques [7]. This creates a seamless mosaic representation capable of compensating for changes in image intensity (due to camera automatic gain control, aperture, and so on) and other border effects that are visible in the current system.

Finally, another extension involves handling the parallax induced by the three-dimensional structure of the scene. A full 3D representation in terms of pointwise range or shape information would be desirable, but would require significant extensions of the hardware. Instead, we intend to represent the scene in terms of layers of surface patches corresponding to planes at different depths. Such a patchwise representation has the benefit of ease of implementation as well as more reliable estimation in terms of parametric motion models.

Acknowledgements

The work described in this paper was supported in part by Sensar Inc., and by the Advanced Research Project Agency under contract DAAA15-93-C-0061. Besides the authors, many of the members of the vision groups at Sarnoff have contributed to this paper. Specifically, we would like to acknowledge the contributions of Rob Bassman, Joe Sinniger, and Gary Greene for hardware design and development.

References

[1] K. Uomori, A. Morimura, A. Ishii, H. Sakaguchi, and Y. Kitamura, "Automatic image stabilizing system by full digital signal processing," *IEEE Trans. Cons. Elec.*, vol. 36, no. 3, pp. 510–519, 1990.

[2] P. J. Burt, "Fast filter transforms for image processing," *Comp. Graph. and Image Proc.*, vol. 16, pp. 20–51, 1981.

[3] P. J. Burt, C. Yen, and X. Xu, "Local correlation measures for motion analysis: a comparative study," in *Proceedings of Conf. on Patt. Recog. and Image Proc.*, pp. 269–274, June 1982.

[4] P. J. Burt, C. Yen, and X. Xu, "Multi-resolution flow-through motion analysis," in *Proceedings of IEEE Comp. Vision and Patt. Recog. Conf.*, pp. 246–252, 1983.

[5] K. Nishihara, "PRISM: a practical real-time stereo matcher," *MIT AI Memo 780*, 1984.

[6] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt, "Real-time scene stabilization and mosaic construction," in *The 1994 Image Understanding Workshop*, Nov. 1994.

[7] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Trans. on Graphics*, vol. 2, no. 4, pp. 217–236, 1983.

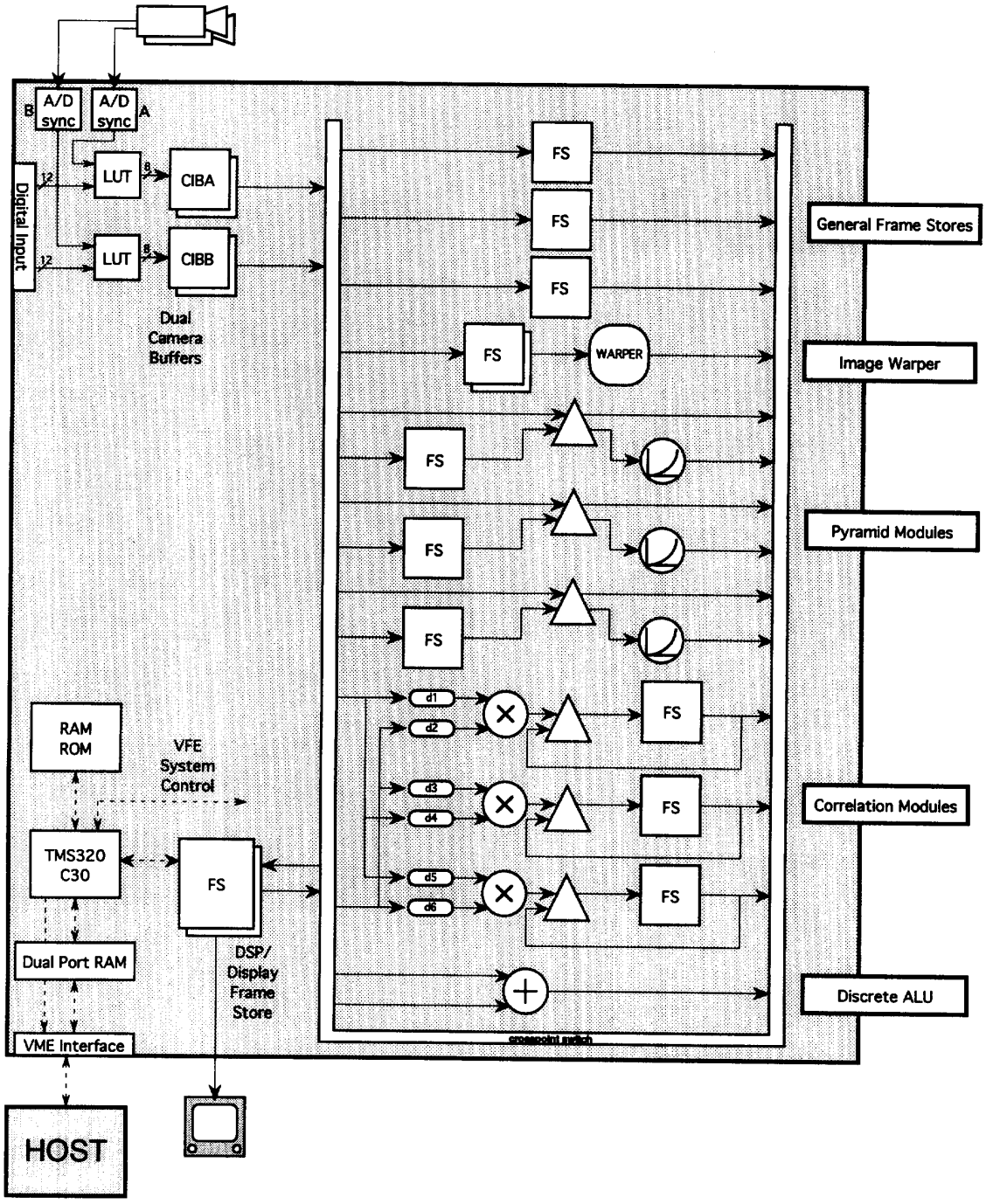

Figure 6: VFE-100 video processing system architecture.

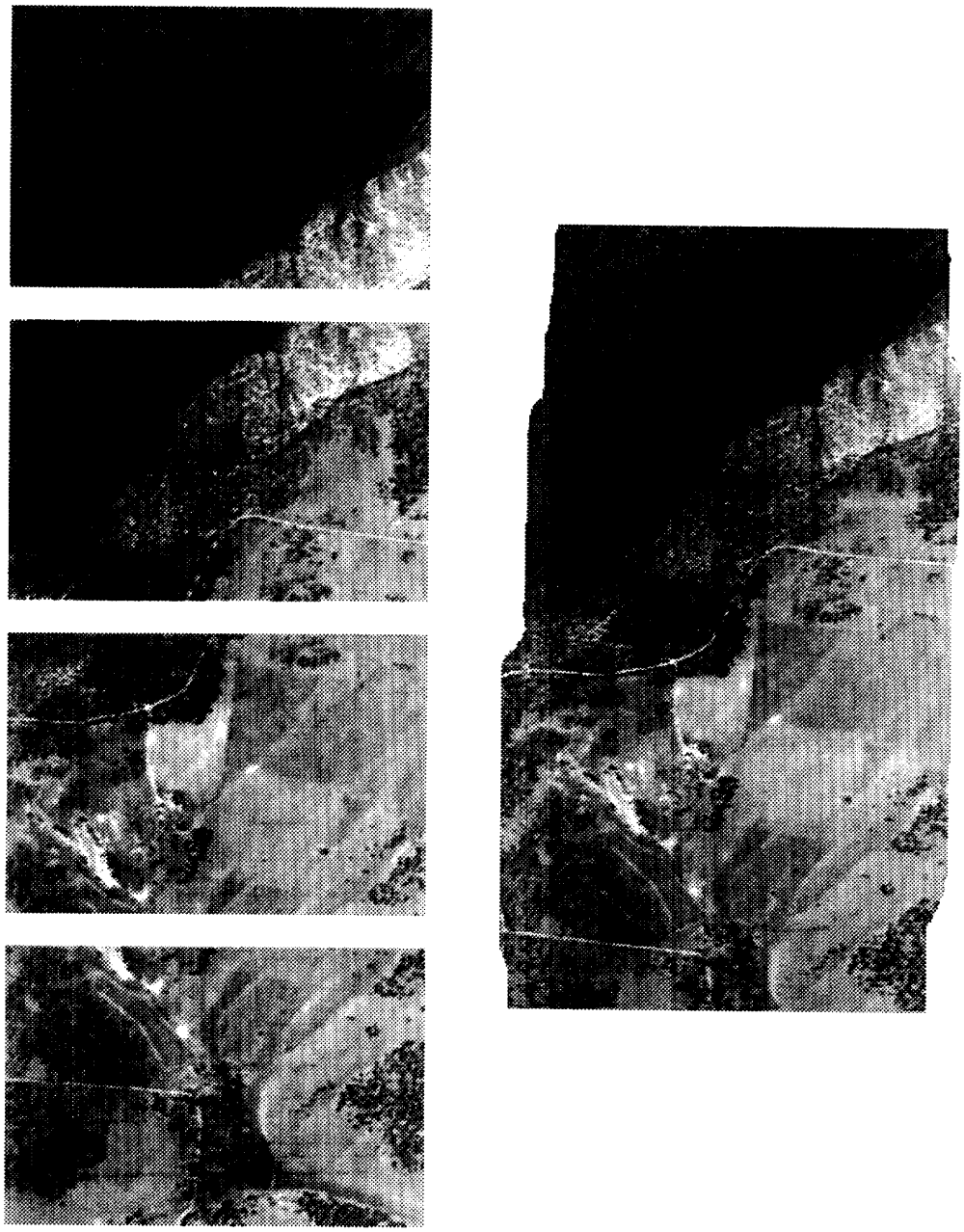

Figure 7: Image mosaic generated by the VFE-100 system. Four frames of the 20 second video sequence, spaced approximately 5 seconds apart, are shown at the left. The completed mosaic, shown at right, represents the assembly of the full 20 second sequence. The processing speed of the VFE-100 was 10 Hz for this example.