TEXAS INSTRUMENTS

Video Restoration on a Multiple TMS320C40 System

Application Report

1996 **Digital Signal Processing Solutions**

TEXAS INSTRUMENTS

Printed in U.S.A., November 1996

Video Restoration on a Multiple TMS320C40 System

Man-Nang Chong
Showbhik Kalra
Dilip Krishnan

Email: asmnchong@ntuvax.ntu.ac.sg
School of Applied Science
Nanyang Technological University
Singapore 639798

This application report consists of one of the entries in a design contest, The 1995 TI DSP Solutions Challenge. The report was designed, prepared, and tested by university students who are not employees of, or otherwise associated with, Texas Instruments. The user is solely responsible for verifying this application prior to implementation or use in products or systems.

SPRA154
November 1996

IMPORTANT NOTICE

Contents

List of Figures

ABSTRACT

This application report describes a parallel video restoration system that restores old motion picture archives. A Gaussian-weighted, bi-directional, 3D auto-regressive (B3D-AR) algorithm is used to alleviate the presence of noise in the old archives. Common forms of degradation found in such archives are “dirt and sparkle” and scratches. The distortion is caused either by the accumulation of dirt or by the film material being abraded.

While most of the existing image restoration algorithms blur edges of moving objects in the vicinity of occluded and uncovered image regions, this algorithm is able to suppress mixed-noise processes and recover lost signals in both the covered and uncovered regions in an image sequence. This video restoration system is tested on the artificially corrupted image sequences and naturally degraded video (full PAL image size). Samples of the original and corresponding restored image sequence are contained in this report.

The B3D-AR algorithm is parallel-implemented on an array of 15 Texas Instruments TMS320C40 processors that are connected in a tree configuration. Two different parallel algorithms are implemented in which a close-to-linear speed-up is achieved by means of a load-balanced parallel algorithm.

INTRODUCTION

While many old movies are recorded on flammable nitrate-based negatives that decay rapidly, modern movies are made with safer acetate-based 35-mm films. However, both types of media are susceptible to degradation such as gouges, scratches, and the accumulation of dirt. The result is a variety of artifacts that make the old movies look their age.

The deterioration in old movies can be stopped by adopting digital film archiving technology; however, defects that are already present in the films will be inherited into the digital storage. Restoration of a degraded motion picture is a highly labor-intensive and extremely costly undertaking. A much publicized event [1] is the restoration work of Disney's 1937 masterpiece—*Snow White and the Seven Dwarfs*, which was re-released in digital form in 1993. It would be financially rewarding to reproduce the old movies with as much fidelity to the original negatives as possible so that the movies can be re-released in higher quality formats such as video-on-demand, digital video-disk, and HDTV. Therefore, a video restoration system that can automatically remove artifacts in film archives will be useful to the entertainment and broadcast industries.

This application report describes an auto-regressive (AR) model-based restoration algorithm and its parallel implementation on a network of Texas Instruments TMS320C40 processors. The restoration process begins with the conversion of the degraded film into its digital form with the aid of a real-time video digitizer. The success of automatic video restoration relies on the fact that image frames in a movie do not change significantly from one frame to the next, except for changes due to moving objects in a scene. This means that the frames preceding and succeeding the current image frame will provide enough repeated information to allow detection of the presence of degraded regions in the image. This same redundancy provides a way to mathematically model the image region at the vicinity of these artifacts so that meaningful information can be used to fill in the corrupted image regions, resulting in a restored image frame. A scene changes due to moving objects, and uncovered background signals must be identified to yield an accurate model. To account for the inter-frame changes that are caused by moving objects in the scene, the motion of these objects is first computed by a motion estimation algorithm. Once the moving regions have been compensated for, a 3-dimensional auto-regressive (3D-AR) model is built from the information contained in both the preceding and succeeding frames. Two of these dimensions describe changes within the image frame, and the third describes changes between frames. Restoration is done one frame at a time so that the restored frame can be used to help restore the subsequent corrupted frames in the image sequence. The proposed AR model-based approach has the important advantage over the global filtering strategy,[2] which tends to blur sharp edges or homogenize highly textual regions in both the distorted and uncorrupted image regions. Statistical approaches suchas Markov random field modeling [3] have produced good results, although at a higher computational cost.

The computational demands required to estimate the motion of moving objects in the image and to formulate the 3D image sequence model are still huge. The time required to restore just a single PAL frame using one workstation can run up to a few hundred seconds. Timely restoration can only be achieved through the design of a *fast* (computationally efficient) algorithm and the use of parallel-processing techniques on a network of digital signal processors (DSPs).

This project describes, in detail, a fast video restoration system where distorted old archives can be digitized, restored, and transferred to new storage media with minimal human supervision.

OVERVIEW OF VIDEO RESTORATION ALGORITHM

The schematic diagram of a video restoration algorithm is shown in Figure 1.

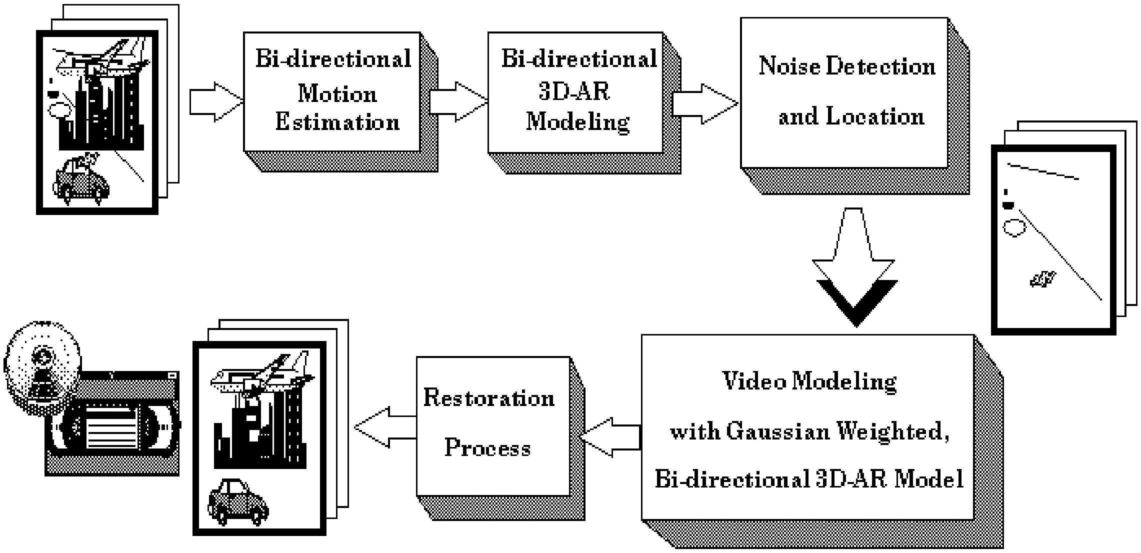

Bi-directional Motion Estimation
Bi-directional 3D-AR Modeling
Noise Detection and Location
Video Modeling with Gaussian Weighted, Bi-directional 3D-AR Model
Restoration Process

Figure 1. Video Restoration Schematic Diagram

BI-DIRECTIONAL MOTION ESTIMATION [4]

The image is first partitioned into blocks of $E \times E$ pixels for the computation of the motion vectors. The motion is first estimated by some motion-estimation algorithms and then processing is directed along the calculated motion trajectories. A robust motion-estimation algorithm is necessary for restoration of the image sequences, and it is noted that motion estimation is a vibrant research field. In this application report, the motion-estimation algorithm used is a robust *overlapped block matching* (OBM) algorithm as shown in Figure 2. In order to get a reliable and accurate displacement estimate, the size of blocks for block-matching must be chosen carefully.

Since the image sequences are bound to be degraded, the estimate will be unreliable and affected by noise if small blocks are used. On the other hand, if large blocks are used, the estimate will become inaccurate as the displacement vector field inside the large blocks will not be constant. Therefore, small blocks are required to estimate the displacement vector field sufficiently local and adaptive. The proposed OBM scheme attempts to circumvent the above-mentioned problems when estimating the motion vectors for each frame of an image.

First, the whole frame is divided into blocks of $E \times E$ pixels. For each block of $E \times E$ pixels, a search is conducted for a forward motion vector (FMV) from a past reference frame (temporal index, $t = -1$) and a backward motion vector (BMV) from a future reference frame (temporal index, $t = +1$). For each block of $E \times E$ pixels, one motion vector is selected from the pair of FMV and BMV; the motion vector that yields a smaller sum of absolute error is selected. The OBM scheme is used to estimate the motion vectors. As shown in Figure 2, the block matching is done with the overlapped blocks of $D \times D$ pixels in the current frame where $D > E$ and all $E \times E$-pixel blocks are centered within the $D \times D$-pixel blocks. This $D \times D$-pixel block is compared with a corresponding block within a search area of size $(D+2P) \times (D+2P)$ pixels in the previous frame, and the best match is found based on the minimum absolute error (MAE) cross-correlation.[5] The motion vectors found by comparing the $D \times D$-pixel block in the present frame and the $(D+2P) \times (D+2P)$-pixel block in the previous frame then are assigned to the $E \times E$-pixel block. The search procedure adopted in the proposed OBM scheme is based on a threshold exhaustive search.[5]

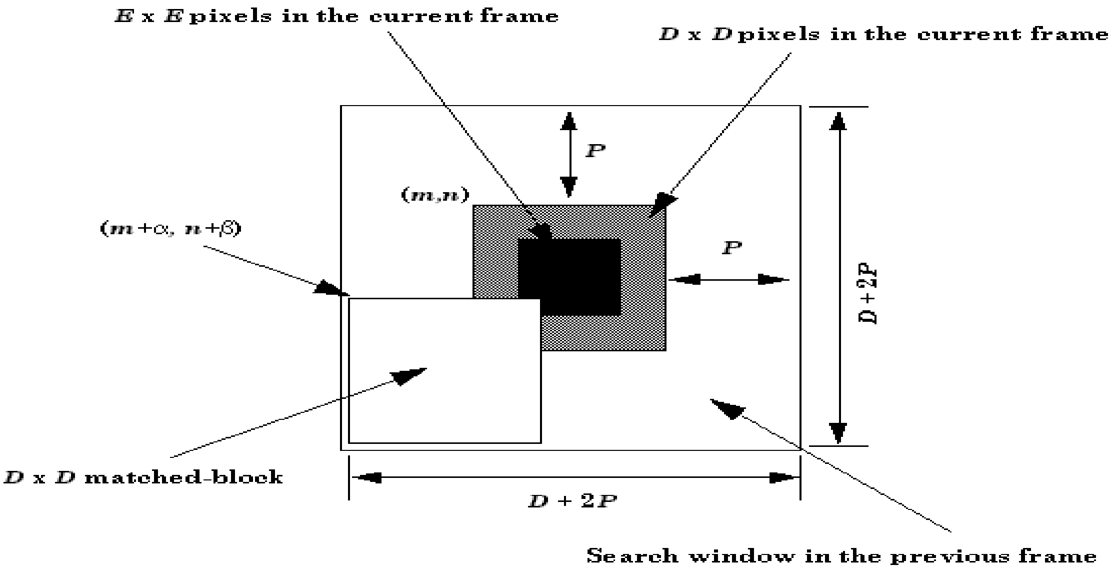

E x E pixels in the current frame
D x D pixels in the current frame
P
(m,n)
(m+α, n+β)
P
D+2P
D x D matched-block
D + 2P
Search window in the previous frame

Figure 2. Overlapped Block-Matching Motion Estimation Algorithm

BI-DIRECTIONAL 3-DIMENSIONAL AUTO-REGRESSIVE (B3D-AR) MODEL [4]

The 3-dimensional auto-regressive (3D-AR) model [6] has been successfully applied to remove impulsive noise and other types of degradation in image sequences, although at a higher computational cost in the interpolation process. Kokaram's 3D-AR model [6] was modified to the *bi-directional 3D auto-regressive* (B3D-AR) model [4] where the computational cost in the interpolation process is reduced significantly. In this application report, B3D-AR as described by Equation (1) is used for the detection of corrupted pixels.

$$\hat{I}(i,j,n) = \sum_{k=1}^{N} a_k I([i + q_{ik} + \alpha_{n,n+q_{tk}}(i,j)],[j + q_{jk} + \beta_{n,n+q_{tk}}(i,j)],[n + q_{tk}]) \quad (1)$$

where

$\hat{I}(i,j,n)$ = Predicted pixel intensity at the location *(i,j)* in the *n*th frame

a_{k} = Auto-Regressive (AR) model coefficients

N = Total number of AR model coefficients

[q_{ik} , q_{jk} , q_{tk}] = Offset vector that points to each pixel neighborhood used for the AR model, as shown in Figure 3. The component of the offset vector that determines the temporal direction of the supporting pixel is q_{tk} and its value is –1 for a support pixel in the preceding frame and +1 for a support pixel in the succeeding frame. Therefore, $I(i+q_{ik}, j+q_{jk}, n+q_{tk})$ is the pixel intensity at the *k*th support position for the pixel at *(i,j,n)*.

$[\alpha_{n,m}(i,j), \beta_{n,m}(i,j)]$ = displacement vector between frame *m* and frame *n* (*i,j*) denotes that the displacement is a function of the position in the image.

For parameter estimation, the task is to choose the parameters in such a manner as to minimize some function of the prediction error $\varepsilon(i, j, n)$, as shown in Equation (2).

$$\varepsilon(i, j, n) = i(i, j, n) - i(i, j, n) \quad (2)$$

There are two sets of parameters to compute (estimate): the model coefficients and the displacement vectors. The motion vectors are to be computed first using a motion-estimation algorithm. Subsequently, the least mean square (LMS) approach is used to compute the model coefficients.

The coefficients chosen to minimize the square of the error in Equation (2) lead to the normal equations:

$$\mathbf{Ra} = -\mathbf{r} \tag{3}$$

Where **R** is a $N \times N$ matrix of correlation coefficients, **a** is the vector of model coefficients, and **r** is a $N \times 1$ vector of correlation coefficients. The solution to Equation (3) yields the model coefficients.[5]

In our implementation, as shown in Figure 3, each block of 16 × 16 pixels in the current frame *n* is modeled with a set of nine AR coefficients. The predicted intensity of a pixel within the 16 × 16 block in frame *n* is calculated from its corresponding motion-compensated 3 × 3 support region in either the previous or next frame.

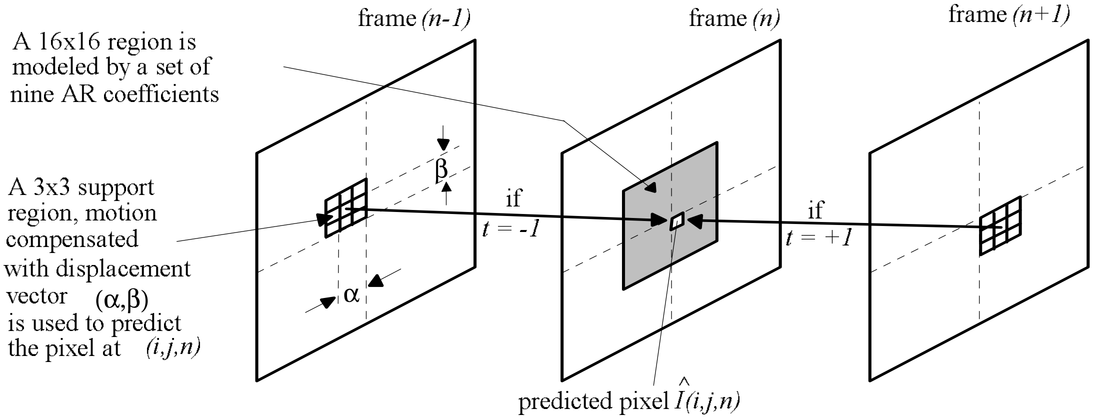

frame (n-1)
frame (n)
frame (n+1)
A 16x16 region is modeled by a set of nine AR coefficients
A 3x3 support region, motion compensated with displacement vector (α,β) is used to predict the pixel at (i,j,n)
β
α
if
t = -1
if
t = +1
predicted pixel $\hat{I}(i,j,n)$

Figure 3. Selected Support Region
(Based on value of *t* obtained from *bi-directional* motion estimator)

DETECTING THE DISTORTIONS IN IMAGE SEQUENCE

The position of a local distortion can be detected by applying some threshold to $\varepsilon_d(i,j,n)$, the square of the error between the actual and predicted intensity of the pixel at location (*i,j,n*), which is given by Equation (4).

$$\varepsilon_d(i,j,n) = (I(i,j,n) - \hat{I}(i,j,n))^2 \tag{4}$$

where the predicted intensity $\hat{I}(i,j,n)$, given in Equation (1), is calculated from the AR coefficients $\{a_1, a_2, \ldots\ldots.a_N\}$, estimated in Equation (3).

REMOVING THE DISTORTIONS IN IMAGE SEQUENCE

The restoration process can be seen as a threefold process. First, the pixels that are detected as "distorted" pixels are weighted according to a Gaussian weighting scheme. Second, a set of newly estimated unbiased AR coefficients are re-computed using Equation (3). Finally, the "distorted" pixels, identified by using Equation (4), are removed by substituting them with the value of $I(i,j,n)$, which is calculated with the new set of AR coefficients.

To restore the "distorted" pixels, the re-computed model coefficients are required. As shown in Figure 3, the support region for each predicted pixel has a size of 3 × 3 pixels only. Therefore, the *normal* equation (Equation (3)) must be altered to solve for the AR coefficients using the Gaussian-weighted coefficients estimation. Normally, the model coefficients chosen are such as to minimize the expected value of the squared error at the concerned point. Once dirt has been detected, some of this data is known to be degraded. Therefore, the prediction error at these points can be weighted by a function $\phi(\varepsilon(i,j,n))$, so that these degraded portions do not affect the estimation process.

The new weighted error equation can be written as

$$\varepsilon_w(i,j,n) = \phi(\varepsilon(i,j,n)) \sum_{k=0}^{N} a_k I([i + q_{xk} + \alpha_{n,n+q_{nk}}(i,j)],[j + q_{yk} + \beta_{n,n+q_{nk}}(i,j)],[n + q_{tk}]) \tag{5}$$

The Gaussian weighting function, $\phi(\varepsilon(i,j,n))$, is assigned to each degraded point depending on the magnitude of the error, ε, at location (*i,j*) in the *nth* frame, during the re-computation of the video model. The rest of the symbols have their usual meaning as presented in Equation (1) and a_0 = 1. The Gaussian weighting function can be described by Equation (6).

$$\phi(\varepsilon(i,j,n)) = \frac{1}{\pi}\exp[-(\frac{\varepsilon(i,j,n)-t_1}{t_2-t_1})] \qquad \text{for} \qquad t_1 < \left|I-\hat{I}\right| < t_2$$

$$\phi(\varepsilon(i,j,n)) = 0 \qquad \text{for} \qquad \left|I-\hat{I}\right| \geq t_2$$

$$\phi(\varepsilon(i,j,n)) = 1 \qquad \text{for} \qquad \left|I-\hat{I}\right| \leq t_1 \qquad (6)$$

The square of the new Gaussian-weighted error Equation (3) is minimized with respect to the coefficients and yields a normal equation similar to Equation (3).

DETAILS OF THE RESTORATION PROCESS

To restore a block of $B \times B$ pixels centered within a block of size $M \times M$ pixels in the current frame, the $M \times M$ block's motion estimate in the previous *or* next frames must be determined. The choice of the previous or next frames is decided by the B3D-AR model, as discussed earlier. Then, using these two blocks of pixels, a set of coefficients, a_k, is derived by the normal equation. It is assumed that the information within a block of size $M \times M$ is stationary enough to enable the use of one model—that is, one set of coefficients for all the M^2 pixels within the block. The model is applied to the $B \times B$ block, and pixels that are identified as “noise” are restored.

The support region used for prediction can be represented as *x:y*. A 9:8 support region means that the support region consists of 9 pixels from the previous frame and 8 from the current frame. We have implemented this model considering information only from the previous *or* next frame. In other words, we employ the 9:0 or 0:9 model. Each pixel in the current frame is thus modeled by 9 pixels in the previous or next frame. A support region wholly in the previous/next frame is unlikely to be affected by noise around the same relative areas as in the current frame (noise is essentially temporally isolated). The use of the 9:0 support region ensures that the current frame information is not used for detection and cleaning.

The “noisy” pixels are now interpolated with their predicted values after re-calculating the model coefficients. The interpolation equation now used is:

$$\boldsymbol{i}_u = \boldsymbol{i}_k \boldsymbol{A}_k \qquad (7)$$

The two vectors $\boldsymbol{i}_u$ and $\boldsymbol{i}_k$, represent the known and unknown (noisy) pixel intensities, respectively. $\boldsymbol{A}_k$ is the matrix of coefficients a_k. The structure of the two matrices $\boldsymbol{i}_k$ and $\boldsymbol{A}_k$ has been modified to set up Equation (7) for a *simple* and *computationally efficient* solution. Matrix $\boldsymbol{i}_k$ is of size $u \times N$, and $\boldsymbol{A}_k$ of size $N \times 1$, where u is the total number of unknown (noisy) pixels in the $B \times B$ block and N is the number of model coefficients a_k. The above solution consists of $N \times u$ operations, which is $\boldsymbol{O(u)}$, since N is fixed. The cleaning process now is dependent on the level of noise present in the $B \times B$ block. Therefore, a less noisy block takes less time to restore than one with more noise, since less computations are involved.

PARALLEL IMPLEMENTATION OF THE VIDEO RESTORATION ALGORITHM ON A NETWORK OF TMS320C40 PROCESSORS

The parallelism inherent in image restoration is geometric parallelism. Each frame in the image can be partitioned into independent sub-blocks. These sub-blocks are then distributed among the *worker* (also known as *slave*) processors by a *master* (root) processor. Each of these blocks will undergo the same restoration operations. Since a *master* processor distributes different data packets to the *slave* processors, each of which performs the same sets of operations on it, the parallel machine can be said to be employing the SPMD (Single-Program Multiple-Data) paradigm.

The parallel implementation of the B3D-AR model is carried out on a network of 15 TMS320C40 DSPs. Each TMS320C40 has 8 MB of DRAM while the root processor has 32 MB of DRAM. The processors are connected in a *tree* configuration. This particular configuration was chosen as it strikes the right balance between efficiency and algorithm simplicity. The architecture of the TMS320C40 also limits the maximum number of possible connections to each processor to 6. The *logical* configuration is shown in Figure 4.

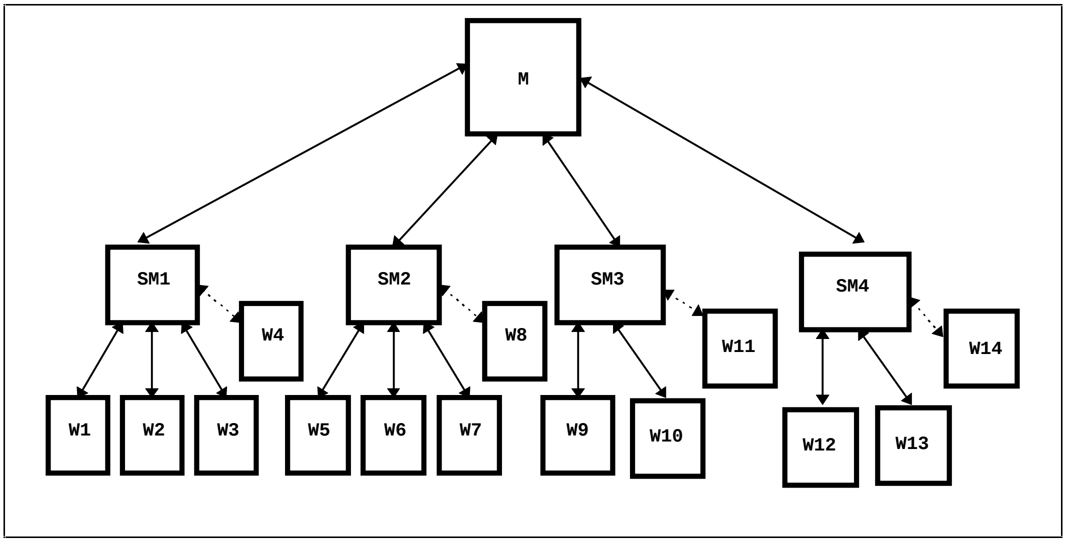


Figure 4. Logical Arrangement of Tasks

There are two entities involved in our system: *tasks* and *processors*. The tasks represent the logical configuration of the system, while the processors represent the physical configuration. The logical configuration is decided by the parallel algorithm used, while the physical configuration of the system is decided by the underlying hardware layout. Figure 4 shows the logical layout. There are three different tasks: master, sub-master, and worker. M represents the master task; SM1–SM4 are the four sub-master tasks; W1–W14 are the worker tasks. The master task resides on the root processor (first-level processor), which also communicates with the host Sun SPARC10™ workstation.[1]

A single processor can have more than one task running concurrently on it. On the second level of the tree configuration, there are two tasks—namely, sub-master and worker tasks—running on the four processors. The dashed arrows depicted in Figure 4 show a logical (non-physical) channel that communicates between the sub-master and worker tasks within a processor. Therefore, 10 of the 14 worker tasks are dedicated—that is, the processors to which they are designated perform only the processing job. The four remaining worker tasks are non-dedicated—that is, they are placed on

[1] **SPARC** is a trademark of **SPARC** International, Inc.

processors that perform both the distribution and processing jobs. It is obvious that the performance of the non-dedicated worker tasks will be lower due to the additional distribution workload on them. The master task M distributes packets of work to the sub-master tasks which, in turn, distribute to the workers.

The “C” programming language was used to implement the restoration algorithm and compiled using the 3L parallel C compiler.[7]

LOAD BALANCING

Load balancing of the entire workload is the most important consideration in the case of parallel algorithms. The authors of this report have employed the RILB (Receiver-Initiated Load Balancing) technique.[8] This scheme is characterized by the fact that the distribution of work is performed only when an idle task requests work. The request for work may be *explicit* (that is, the passing of a message requesting work) or *implicit* (that is, the task finishes processing the work packet assigned to it and passes back the results). Implicit requesting is used because the processed (cleaned) block of data must eventually be sent back up to the master for re-combination with the rest of the processed image.

Each workload (packet) consists of a 16 × 16-pixel block in the current frame as well as its search space in the previous and next frames. The sub-masters serve as work distributors. Initially, they send out work packets to all workers under them. The workers receive this work packet, perform motion estimation and noise detection, and restore the corrupted pixels. The workers then pass back the pixel positions of the noise and their new “clean” intensity values for the final image tiling to the master. This is also a signal indicating that the workers have finished with their assigned task. Whenever a sub-master receives such a signal from any worker, it will relay the signal upwards to the master. The sub-master will then receive the next work packet from the master.

The size of each work packet must be small enough to ensure that while the master is distributing work packets, no worker has to wait too long. Performance degradation would set in if a processor had to wait for long.

RESULTS AND DISCUSSION

The proposed algorithm is evaluated by applying different image sequences containing different noise processes:

1. Salesman Sequence: uncovered-background region in an image sequence that is artificially corrupted by single-to-multiple pixel-sized impulses
2. Salesman Sequence: occluded region in an image sequence that is artificially corrupted by single-to-multiple pixel-sized impulses
3. Salesman Sequence: a sequence that undergoes translational motion and is artificially corrupted by single-to-multiple pixel-sized impulses
4. Corridor Sequence: area that undergoes a zooming process and is artificially corrupted by single-to-multiple pixel-sized impulses
5. Frankenstein Sequence: real degraded image sequence

All the artificially added noise is temporally isolated, which is usually the case in a real degraded motion picture.[9]

Figure 5 shows an artificially corrupted frame in the Salesman sequence. The blotches and scratch line are temporally isolated. The proposed algorithm (BAR3D) is applied to the Salesman sequence that contains regions undergoing self-occlusion and to the Corridor sequence that consists of a scene undergoing zooming. Multiple pixel-sized blotches and artificial scratches are synthetically added to several frames of each sequence. The picture quality can be seen in Figure 6, which shows the corresponding restored frame using the B3D-AR model.

Figure 7(a) shows a magnified portion of the degraded Salesman frame. Figure 7(b) shows the corresponding restored frames using the BAR3D models. Figure 8(a) shows a degraded Corridor frame. The Corridor sequence exhibits a motion called zooming. Figure 8(b) shows the corresponding restored frames using the BAR3D model.

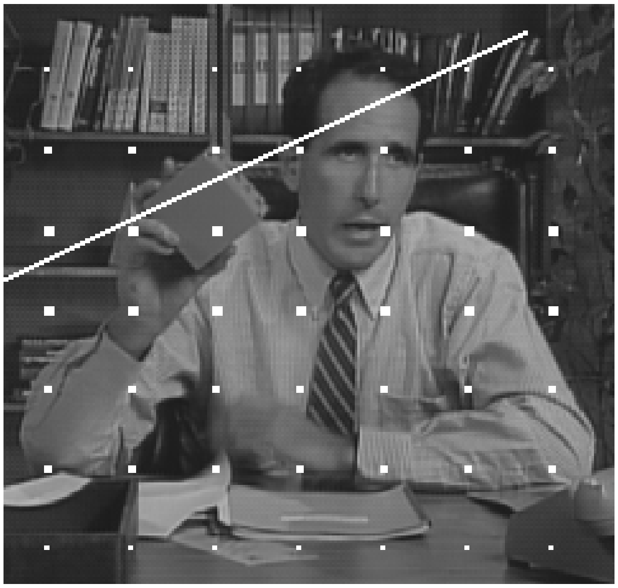


Figure 5. Corrupted Frame of Salesman Sequence
(Blotches vary from sizes of 2×2 to 4×4 and a line of width 2)

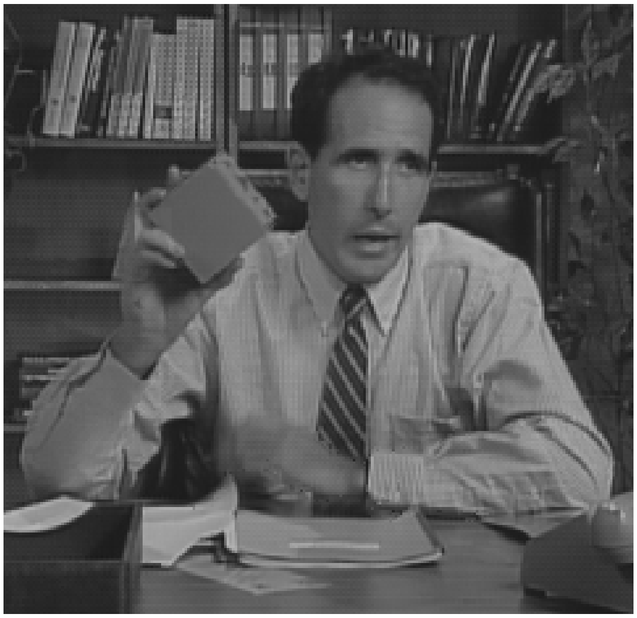
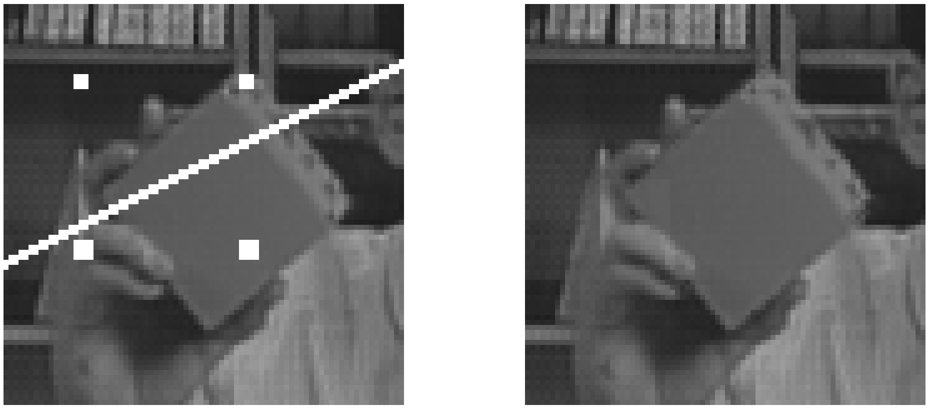


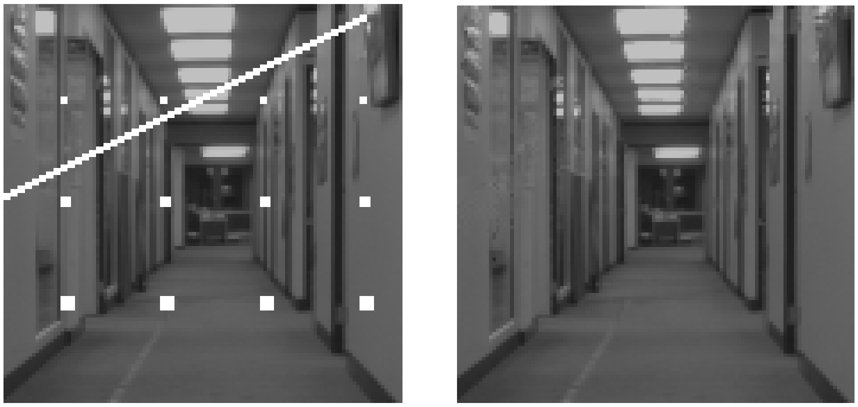
Figure 6. Restored Frame—Using Bi-Directional 3D-AR Model



7(a)

7(b)

Figure 7(a). Magnified Portion of Corrupted Salesman Frame—at Region of Self-Occlusion
Figure 7(b). Corresponding Restored Frame—Using Bi-Directional 3D-AR Model



8(a) 8(b)

Figure 8(a). Corrupted Frame in Corridor Sequence
Figure 8(b). Corresponding Restored Frame—Using Bi-Directional 3D-AR Model

The restoration quality of the algorithm on naturally degraded image sequences (obtained by digitizing the Frankenstein video) is shown in Figures 11(a), 11(b), 12(a), and 12(b). The video was first digitized from a PAL-format video tape before applying the algorithm onto the image sequences. The size of each frame in the PAL image sequence is 576 × 720. The original Frankenstein sequence is heavily blotched and has been restored effectively by the Gaussian-weighted, B3D-AR algorithm.

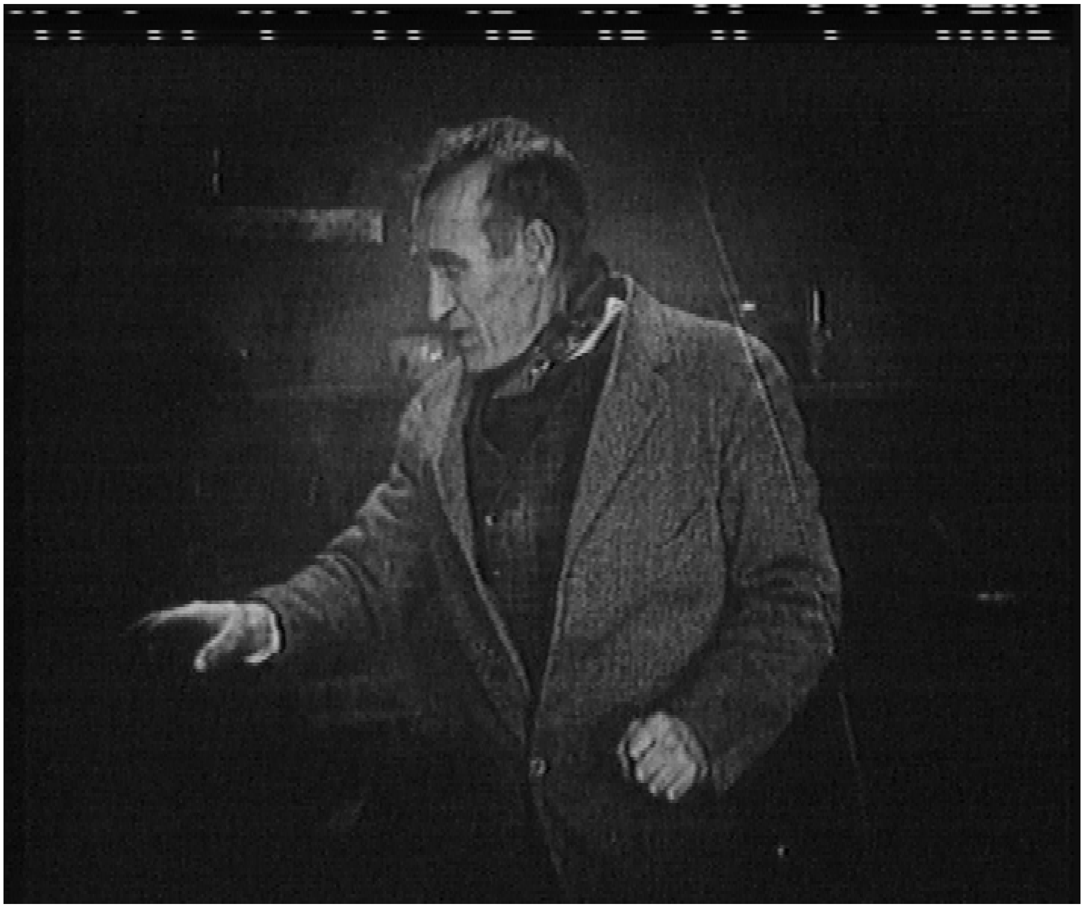


Figure 9(a). Sample A—Selected Frame of Noise-Corrupted Image Sequence

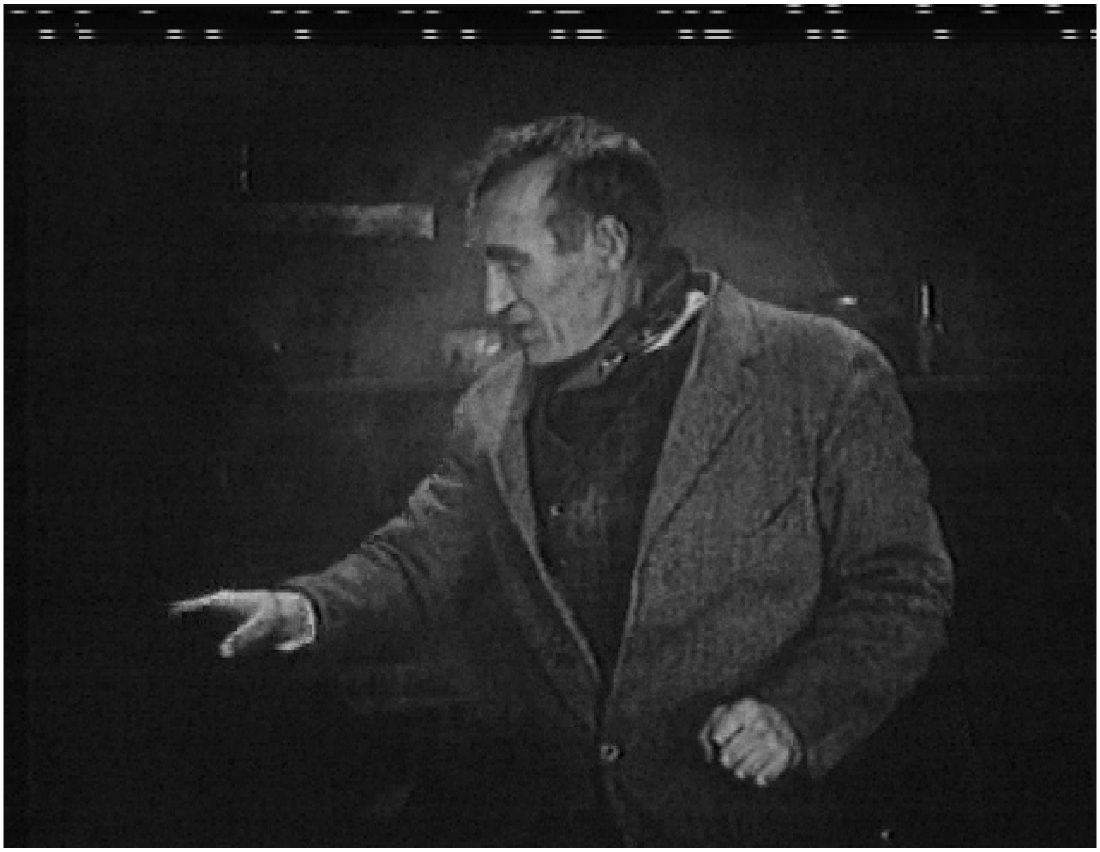


Figure 9(b). Corresponding Restored Sample A Frame—Using Bi-Directional 3D-AR Model

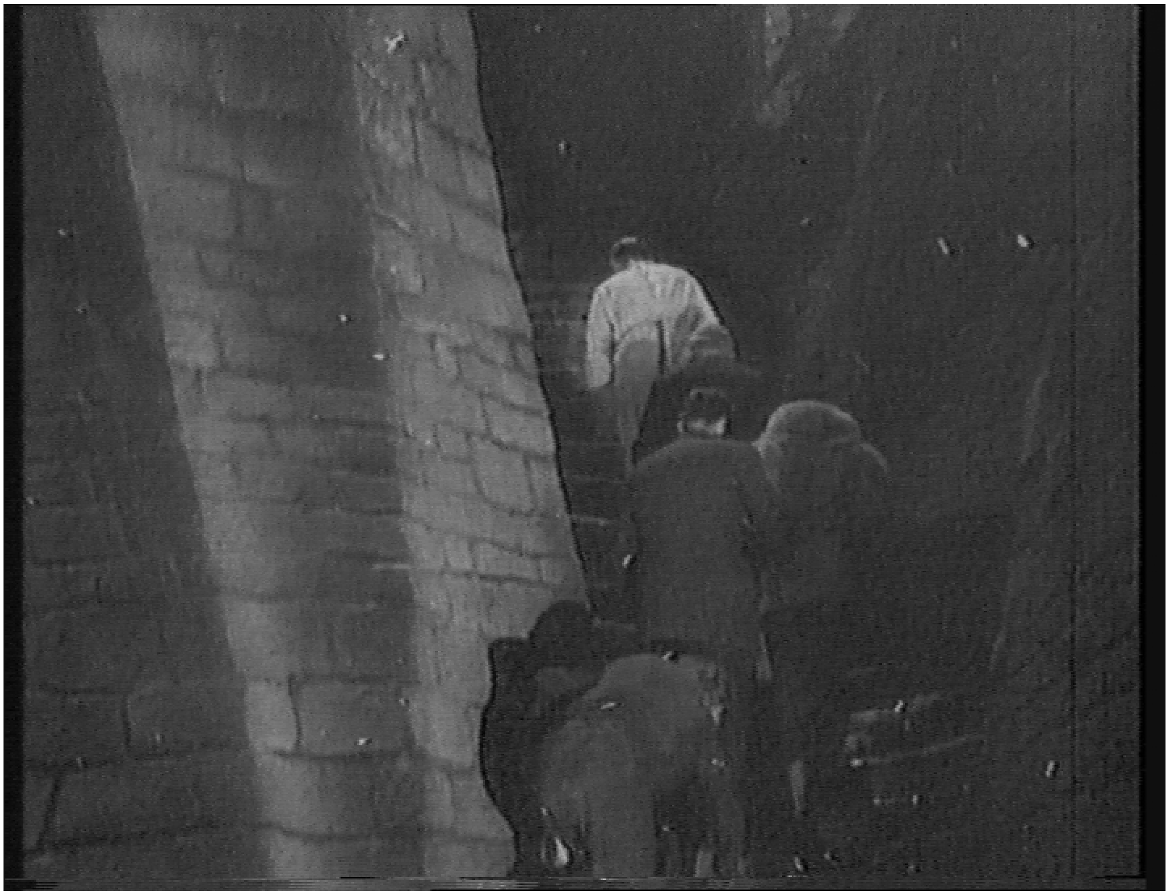


Figure 10(a). Sample B—Selected Frame of Noise-Corrupted Image Sequence

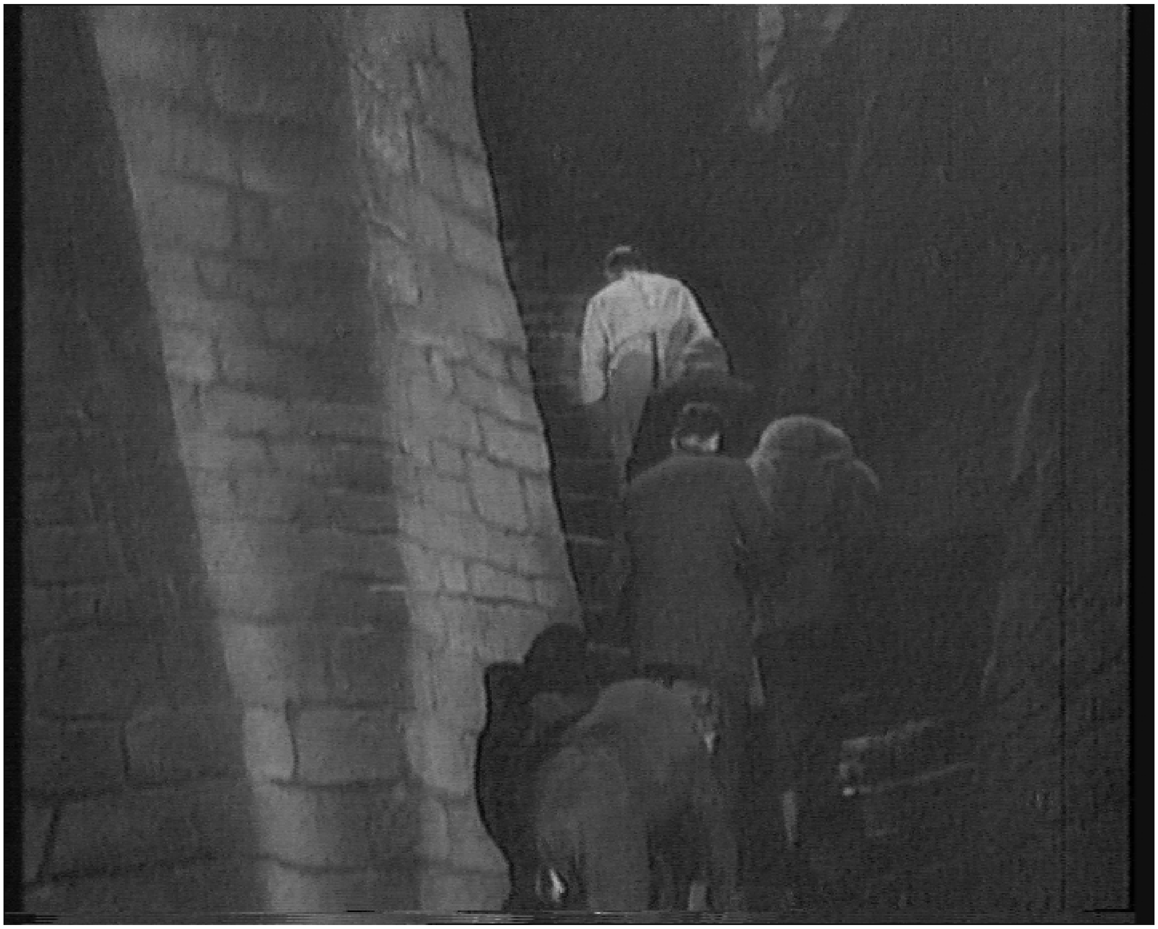


Figure 10(b). Corresponding Restored Sample B Frame—Using Bi-Directional 3D-AR Model

PROCESSING PERFORMANCE OF THE PARALLEL VIDEO RESTORATION SYSTEM

The second-level processors (shown in Figure 4) actually execute two different tasks: sub-master and worker tasks. These tasks run concurrently. This means that if the distribution of the workload exceeds the processing of the workload in the sub-master processor, drastic performance degradation can take place.

Two algorithms were implemented on these four second-level processors to measure the effect of the distribution workload on their performance. The first algorithm (Algorithm A) consisted of a simple mechanism where no distinction was made between the concurrently executing tasks. The sub-master processor divides time slices equally between the two tasks.

The second algorithm (Algorithm B) followed from a careful analysis of the sub-master and worker processing burdens. It was found that the computational load of the worker exceeded that of the sub-master by a substantial margin. It is, therefore, not justified for the two tasks to receive the same share of processing time. The sub-master task needs to be active only when a worker under it has completed a work task and requires a fresh work packet. Therefore, the worker should receive a larger share of processor time. This was achieved by using *priority*.[7] The worker tasks were given the highest priority. The sub-master task was accorded a low priority, becoming active only when required; otherwise, it remained *descheduled*. This is in contrast to Algorithm A, where the distributor task is constantly running without doing any useful processing.

Figure 11 shows the speedup characteristics of Algorithms A and B.

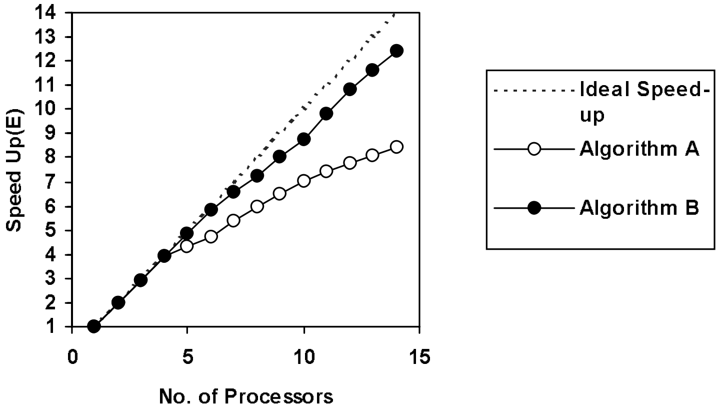


Figure 11. Speed-Up Characteristics—Both Algorithms

The improvements in the results for Algorithm B are evident. In a network consisting of up to four processors, the two algorithms display the same speed-up. This is due to the fact that only the four first-level processors are used as dedicated workers only. When the tree-configured network grows into the third level, the degradation of the performance of Algorithm A starts to take place. Algorithm B, on the other hand, does not degrade as much. This clearly demonstrates the precedence that the worker tasks should take over the distributor tasks. It is also observed that the performance of Algorithm B is well above the $P/\log P$ speed-up that is usually accepted as good performance for a parallel algorithm.

CONCLUSION

The video restoration algorithm and its implementation on a network of 15 TMS320C40s are presented in this application report. The algorithm is shown to have better restoration quality when tested on a set of image sequences. The results and analysis show that the B3D-AR model is capable of restoring noise-corrupted video. While the Gaussian weighting scheme provides good spatial support, the *bi-directional* scheme prevents the progressive degradation of image sequences due to the corruption in regions exhibiting different motion processes (such as occlusion, zooming, rotation, and panning). The video restoration has been tested on different image sequences containing different noise processes such as variable-size blotches and line scratches. Its effectiveness in the restoration of these different noise artifacts has been demonstrated. More importantly, when the system is applied to a naturally degraded (PAL-size) video, the noise level of the image sequence is significantly reduced while the crispness and sharpness of the original image sequence are retained.

Parallel implementation of the proposed algorithm is realized where close to linear speed-up is achieved on a 15-node TMS320C40 system hosted by a Sun SPARC10 workstation.

SUMMARY

This application report describes a video restoration algorithm and its implementation on a network of 15 Texas Instruments TMS320C40 DSPs. The video restoration algorithm used is a Gaussian-weighted, *bi-directional* 3D auto-regressive (B3D-AR) model. This restoration algorithm alleviates the presence of noise in old video archives. Common forms of degradation found in such archives are “dirt and sparkle” and scratches. The distortion is caused by the accumulation of dirt, the degradation of the films due to chemical process, or by the film material being abraded.

This parallel video restoration system is shown to have better restoration quality when tested on a set of image sequences. The results and analysis show that the B3D-AR model is capable of restoring noise-corrupted video. While the Gaussian weighting scheme provides good spatial support for the model, the *bi-directional* scheme prevents the progressive degradation of image sequences due to the corruption in regions exhibiting different motion processes (such as occlusion, zooming, rotation, and panning). The video restoration has been tested on different image sequences containing different noise processes (such as variable-size blotches and line scratches). Its effectiveness in the restoration of these different noise artifacts has been demonstrated. More importantly, when the system is applied to a naturally degraded (PAL-size) video, the noise level of the image sequence is significantly reduced while the crispness and sharpness of the original image sequence are retained. The results are in contrast with most of the existing image-restoration algorithms which blur the edges of moving objects in the vicinity of occluded and uncovered image regions; the video-restoration algorithm described here can successfully suppress mixed-noise processes and recover lost signals in both the covered and uncovered regions in image sequences.

Parallel implementation of the proposed algorithm is realized where close to linear speed-up is achieved on a 15-node TMS320C40 system hosted by a Sun SPARC10 workstation. This application describes a fast video restoration system that distorted old archives can be digitized, restored, and transferred into new storage media with minimal human supervision.

REFERENCES

1. B. Fisher, “Digital Restoration of Snow White: 120,000 Famous Frames are Back”, *Advanced Imaging*, September 1993, pp. 32–36.
2. G. R. Arce, “Multistage Order Statistic Filters for Image Sequence Processing”, *IEEE Transactions on Signal Processing*, 39(5), May 1991, pp. 1146–1163.
3. R. D. Morris, “Image Sequence Restoration using Gibbs Distributions”, PhD. Thesis, Department of Engineering, University of Cambridge, U.K., May 1995.
4. W. B. Goh, M. N. Chong, S. Kalra, D. Krishnan, “A Bi-Directional 3D AR Model Approach to Motion Picture Restoration”, *IEEE Int. Conf. on Acoustics, Speech & Signal Processing* , May 1996, pp. 2277–2280.
5. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.
6. A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, P. J. W. Rayner, “Interpolation of Missing Data in Image Sequences”, *IEEE Trans. On Image Processing*, Vol. 4, No. 11, Nov. 1995, pp. 1509–1519.
7. 3L Ltd., *Parallel C User Guide for Texas Instruments TMS320C40* , 1995.
8. Vipin Kumar, Ananth Y Grama, and Nageshwara Rao Vempaty, “Scalable Load Balancing Techniques for Parallel Computers”, *Journal of Parallel and Distributed Computing*, 1994, pp. 60–79.
9. S. Geman and D. McClure, “A Nonlinear Filter for Film Restoration and Other Problems in Image Processing”, *CGVIP Graphical Models and Image Processing*, July 1992, pp. 281–289.