JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 30, 99–106 (1995)

Multiprocessor Image Rotation

J. E. L. HOLLIS* AND T. E. CRONK†,[1]

*Royal Military College of Science, Cranfield University, Communications & Information Systems Engineering Group, Shrivenham, Oxfordshire SN6 8LA, United Kingdom; and †University of Portsmouth, School of Systems Engineering, Portsmouth, Hampshire PO1 3DJ, United Kingdom

A technique for reducing the time taken to perform image rotation on general purpose computing hardware is presented. A powerful attribute of the technique is that it is easily mapped onto multiprocessor systems, returning a performance proportional to the number of processors employed. In this article, particular attention has been given to the issues of data management and data communication associated wtih a multiple-processor approach to pure rotation of images. The speed enhancement obtained using concurrently operating processors has been quantified and the limitations have been examined. With the performance achieved, low-cost PC-based multiprocessor systems can satisfactorily process the satellite data now readily available from National Remote Sensing Centres.

THE MOTIVATION

The work presented in this article has been driven by the need to perform fast geometric alterations on images produced by synthetic aperture radar (SAR). If algorithms that make SAR images faster to process, ideally to the extent that general purpose, inexpensive computers may be used to manipulate images quickly, can be found, then the benefits of SAR as a remote sensing tool will become available to a much wider audience.

Geometrically uncorrected SAR images contain inherent spatial distortions resulting from the angle of image capture, the curved surface of the Earth, and the relative motion of the Earth and satellite. In addition, satellite-derived imagery is aligned to the satellite track on a locally orthogonal grid, and this is rarely coincident with the standard map projections to which comparisons need to be made [1]. Image correction and rotation are therefore necessary with SAR images. The computationally intensive process of geometrically adjusting a digital image in order to produce picture elements (pixels) which are accurately located on a map drawn to a standard cartographic projection is known as geocoding. Geocoding can be broadly broken down into a mapping function and a pixel value assignment function (usually involving interpolation). In this article the mapping function is a pure rotation, although the methodology developed may easily be extended to include the removal of spatial distortion. The article describes a method of multiple-processor image rotation that allows performance improvements in return for an increase in the number of processors.

[1] Present address: Hitachi UK, Ltd., Maidenhead.

THE PROBLEM

A typical SAR image, generated from radar echo data with a 25-m resolution, covers a 125 by 125 km area. Processor memory restrictions and the geocoding mapping functions necessitate the generation of complete SAR images in sections (image tiles). Image tiles are usually processed in sequence by a single processor, which takes considerable time (minutes) [2, 3]. Further, any rotation operation is performed as an additional process on the final image. It is possible to distribute tiles across processors in a multiprocessor configuration, allowing parallel operation to speed up the computation. Additionally, using the developed methodology, it is possible to include the rotation function into the processing of each image tile, enabling the final image to be produced in the desired orientation and significantly improving the speed of the image corrections.

The input data form a large data set of pixel intensity values on a regular, two-dimensional, nearly orthogonal coordinate grid, covering a rectangular domain (an image tile). The data are derived from a continuous smooth function adequately sampled and are typically of the order of some 5000 by 5000 pixels, each with a 16-bit value. The required output is a set of transformed values for a new coordinate frame differing from the input coordinate frame by a rotation.

The rotation processing methodology described will meet the needs of near real-time applications on low-cost, parallel processing engines of the following general type:

(a) Each processing unit has its own local memory.
(b) There is no global memory.
(c) All input and output is from disk.

IMAGE ROTATION

Generally, any transformation to new grid coordinates requires 2D interpolation of image values. While it is easy to implement 2D interpolators on a computer, the one-dimensional nature of computer memory makes them slow

0743-7315/95 $12.00

in comparison to 1D interpolators. Hence several methods for rotating images using one-dimensional interpolation [4–7] have been derived.

Friedmann [4, 5] established one method by which the 2D interpolation of near critically sampled images may be achieved by resampling followed by two 1D interpolations of an image. This method involves:

(a) Resampling in one axis to avoid undersampling on the diagonal.
(b) A horizontal image shear.
(c) A vertical image shear.

These are illustrated in Fig. 1.

The preliminary resampling pass in the horizontal x direction of the input x, y coordinate frame ensures that the Nyquist sampling condition exists in the directions of the new u, v coordinate axes.

If the image to be rotated is already sufficiently oversampled, then it is not necessary to make a resampling pass, reducing the three 1D interpolation passes to two 1D interpolation passes. In the SAR case, the SAR processor can be configured to produce the necessary degree of oversampling and a two-pass rotation can always be performed.

The intermediate image interpolation pass in the vertical y scan direction produces the pixels on a line at an angle θ to the horizontal x scan direction (◆ in Fig. 1).

The final pass along these new pixel lines produces the desired pixels on the new coordinate lattice (+ in Fig. 1).

Paeth [6] and Danielsson and Hammerin [7] describe 1D interpolator image rotation schemes which employ a decomposition of the standard rotation matrix different from that used by Friedman. These require three skewing interpolation passes over the image, which, for the general case of image rotation, can be more efficient than the augmented Friedmann approach. However, in the case where an image is already sufficiently oversampled, as in our case, the Friedman approach may be implemented as a two-pass scheme, making it preferential.

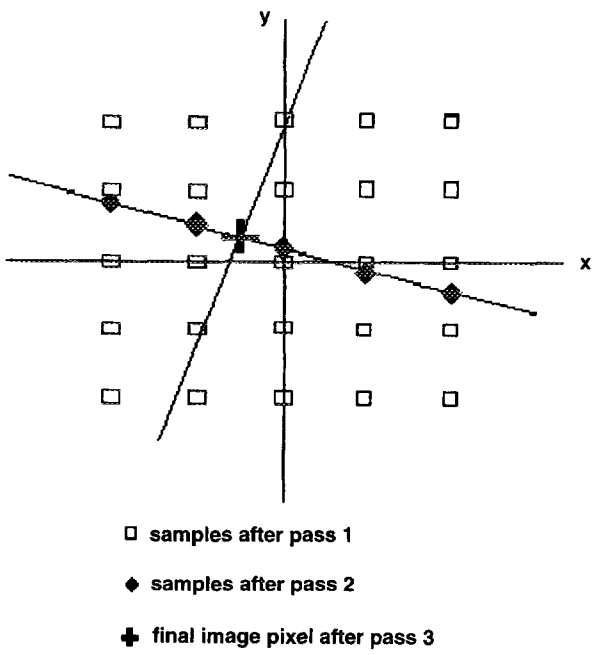

FIG. 1. Three-pass resampling scheme (input: x, y frame; output: u, v frame).

THE ROTATION MAPPING

In general, an image is scanned top left, point (0, 0), to bottom right, point (x_N, y_N), and hence maps easily into computer memory, where each x line is stored sequentially one after the other (i.e., row-major order). However, in order to calculate the pixel transformations required to rotate an image, it is more convenient to relate the image coordinates to the image rotation center.

With the origin at the rotation center, the pixel transformations for a rotation can be calculated from the rotation matrix,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}, \tag{1}$$

where (u, v) is the output coordinate frame, (x, y) is the input coordinate frame, and θ is the rotation angle. A complication arises in real SAR images because aspect ratios other than 1 are usual, where the pixel spacings in the image x and y directions represent different absolute distances in the scene. To adjust for this, one of the axes must be scaled before the pixel transformations are calculated and rescaled afterward.

The rotation transformation generates a direct pixel mapping from the input to the output image frames. It does not specify the intermediate points for interpolation in the multiple-pass, 1D interpolator scheme. To do this and implement the intermediate pass interpolations, it is necessary to consider the grid intercept points of the input and output frames. The choice of intercept options is illustrated by the numbers 1 to 4 in Fig. 2.

The intercepts used for any rotation are dependent upon the degree of rotation being applied, but since the rotation can always be reduced to less than 45° by row/column transposition (90° rotation) and line inversion (180° rotation), the intersections labeled as 1 can always be used. These intercepts provide the least distance between interpolation points in the 0° to 45° range, and therefore ensure the maximum image sampling for this pass.

The importance of correctly defining the size of the output image address space is also illustrated in Fig. 2. The information in the corners of the image will be lost if the address space is defined as that of the input image. To avoid this, the output image address space should be oversized, such that the whole of the input image is contained within the output image address space.

For a pure-rotation operation, the offset to the first interpolation position in any input image line varies with the

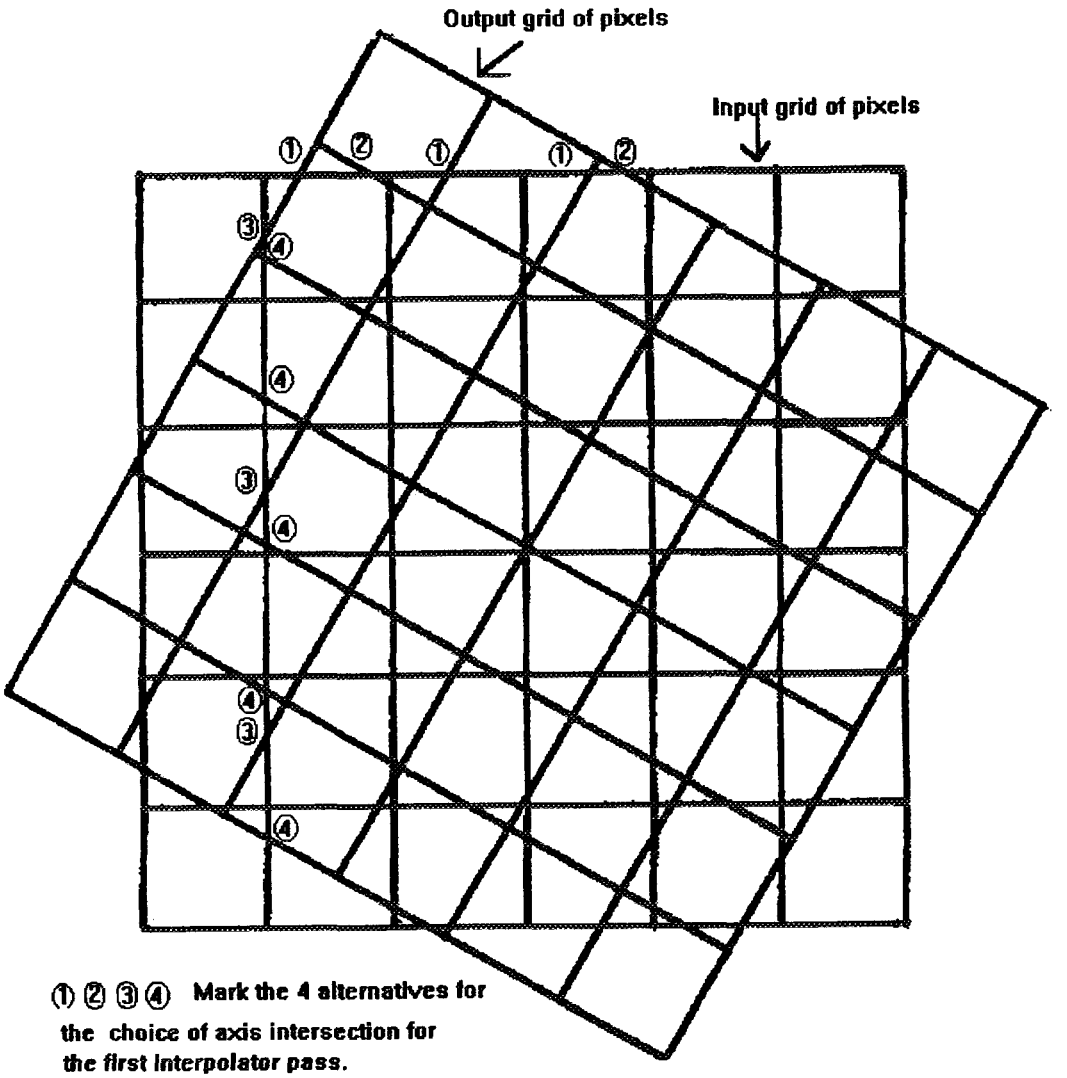

FIG. 2. Input and output frame intercepts.

angle of rotation. However, the offsets between all subsequent adjacent pixels in an input line are always the same. Thus an initial displacement for each scan line and a standard pixel offset are all that need be passed to the interpolating function to perform the first pass of a rotation.

For the pass along the intermediate u lines the interpolation positions are most easily represented as pixel offsets. However, there is a choice of datums; the offsets may be measured either from the first pixel in each intermediate frame line or from an edge of the intermediate image frame. If the intermediate image is stored so that addressing relates directly to image positions, then the second method is more easily implemented, since pixel offsets relate directly to the image structure.

A program was written to generate a library of look-up tables for pixel mapping for any given rotation angle. The resolution and input image size were input as parameters for table generation. The first-pass table contained the standard pixel displacement, the offset for the first pixel in any given scan line, and the number of interpolations in that line. The main advantage in taking a look-up table approach is that it allows the actual interpolation algorithm to be independent of the mapping function.

Methods for optimizing interpolator implementation and assessing interpolator speed have been treated in detail [8–10]. These works show that in the case of SAR data a minimum of four-point interpolation is necessary.

If the input image pixel values are stored in computer memory in sequential order as they are scanned, the interpolator pass along the scan lines can be efficiently implemented. The data access is sequential; lines of data can be transferred via pointers to cache memory. Pointers can be used for data block distribution to tiles. The best method of storing the intermediate pixels generated by this pass is not so obvious. Two considerations arise. The first consideration is that, ideally, intermediate pixels should be written back over the input image to save memory. This problem has to be addressed twice, as the final image also needs to be stored. The second consideration is that pixels in the intermediate image (u, y frame) are generated along

the lines of y. The final interpolation pass however, is across these y lines. Therefore, if the intermediate image pixels are stored in memory sequentially as they are generated, then they are not sequentially positioned for the final image interpolator pass, and a data-addressing overhead is incurred.

With the provision of a small buffer to cover the interpolator length, it is possible to write the intermediate image back over the input image. It is not possible, however, to write the final image over the intermediate image, as intermediate image pixels are not used from sequential positions in memory. The use of a full intermediate image buffer is therefore forced. This has the advantage that the intermediate image pixels may then be stored in a convenient order in this buffer and the final output image may be written back over the input image.

MULTIPLE-TILE/MULTIPLE-PROCESSOR ROTATION

At first glance it appears as if the interpolator passes needed in the multiple-pass image transformation scheme may be effectively pipelined in a coarse-grain parallel solution with the interpolator passes implemented on different processors. However, because of the different directions of interpolator travel across the image, the intermediate image pass must be almost completed before the final pass can proceed. Therefore, the rotation of the whole image cannot be effectively pipelined in this manner.

A processor "farming" structure lends itself more naturally to the processing of image data in tiles (sections). The master processor controls distribution of tile data to slave processors. On completion of the tasks in the slaves, the master processor reads back the processed data from the tiles and rebuilds the modified image.

THEORY OF TILED IMAGE ROTATION

Although it is simple to partition an image into tiles, those tiles do not necessarily map easily to the output image.

To perform an image rotation, all pixels and all tiles must be rotated about the same point, usually the image center. This implies that tiles, in either the input or the output image, will have nonuniform geometrical structures. Nonuniform tile structures lead to several complications in image rotation. If tiles are nonuniform in the input image, then image partitioning will be complicated or large areas of image pixel values will need to be duplicated. Further, joining tiles to form the final image will be difficult. It is also awkward to facilitate efficient interprocessor communication with irregularly shaped tiles, since block movements are less feasible. An additional complication of nonuniform geometrical structure is that each tile requires its own specific interpolator driving algorithm.

Figure 3 shows the partitioning of an input image into tiles prior to rotation, a rotation about the image center, and the desired position of the tiles in the output image after rotation. Output tile "1 out" requires data from input tiles "1 in" and "2 in." If each input tile were stored on a separate processor, this would imply a large amount of interslave communication and interslave data transfer. The problem is aggravated when images are divided into a greater number of tiles. If image rotation in tiles is to be pursued, then interslave processor communication of this nature should be avoided, as it is both complicated and slow.

A better solution to the problem of image rotation is as illustrated in Fig. 4, where the rotation is about each tile center. In Fig. 4, the input image is divided into uniform tiles and each tile is then rotated about its own center. This gives identical operations for all image tiles, simplifying parallelism. On completion of tile rotation, each tile is correctly positioned in the output image by a tile translation operation.

The advantages gained in using this approach are:

(1) Image partitioning requires few operations.

(2) Tiles may be transferred by efficient block data transfers.

(3) The same rotation operation is performed on all tiles.

The following points are important:

(i) Allowance for interpolator length must be made at tile boundaries. The most efficient approach was found to be duplicating the edge of image tiles, i.e., overlapping adjacent tiles by half the interpolator length. An alternative approach would be to establish an interprocessor communication protocol for transferring tile edge pixels as required. Care would need to be taken to ensure that pixels from the appropriate frames were transferred.

(ii) The most effective way of transferring tiles is to transfer the entire image tile address-space en bloc. This is highly efficient and straightforward for input tiles, as every address location contains an image pixel value. The output tile address space, however, contains blank address locations at the tile corners, with the amount of blank address locations varying with block shape and the angle of rotation applied. Square tiles and small angles of rotation lead to minimum unused address space, while a 45° rotation angle gives the worst case. The oversized address space of output tiles causes problems in the building of the final image, as the required output image tile pixels must be extracted from output tile address space. A good solution to the pixel extraction problem is to initialize the output tile address-space contents prior to a tile rotation, to a value outside the image intensity range. Then, on completion of a tile rotation, unused output tile memory addresses contain invalid pixel values, making it a simple task to extract valid image tile pixel values from the output tile address space.

(iii) To avoid losing the corners of each rotated tile, the memory address space of rotated tiles (output tiles) must be significantly larger than the address space of the input tiles.

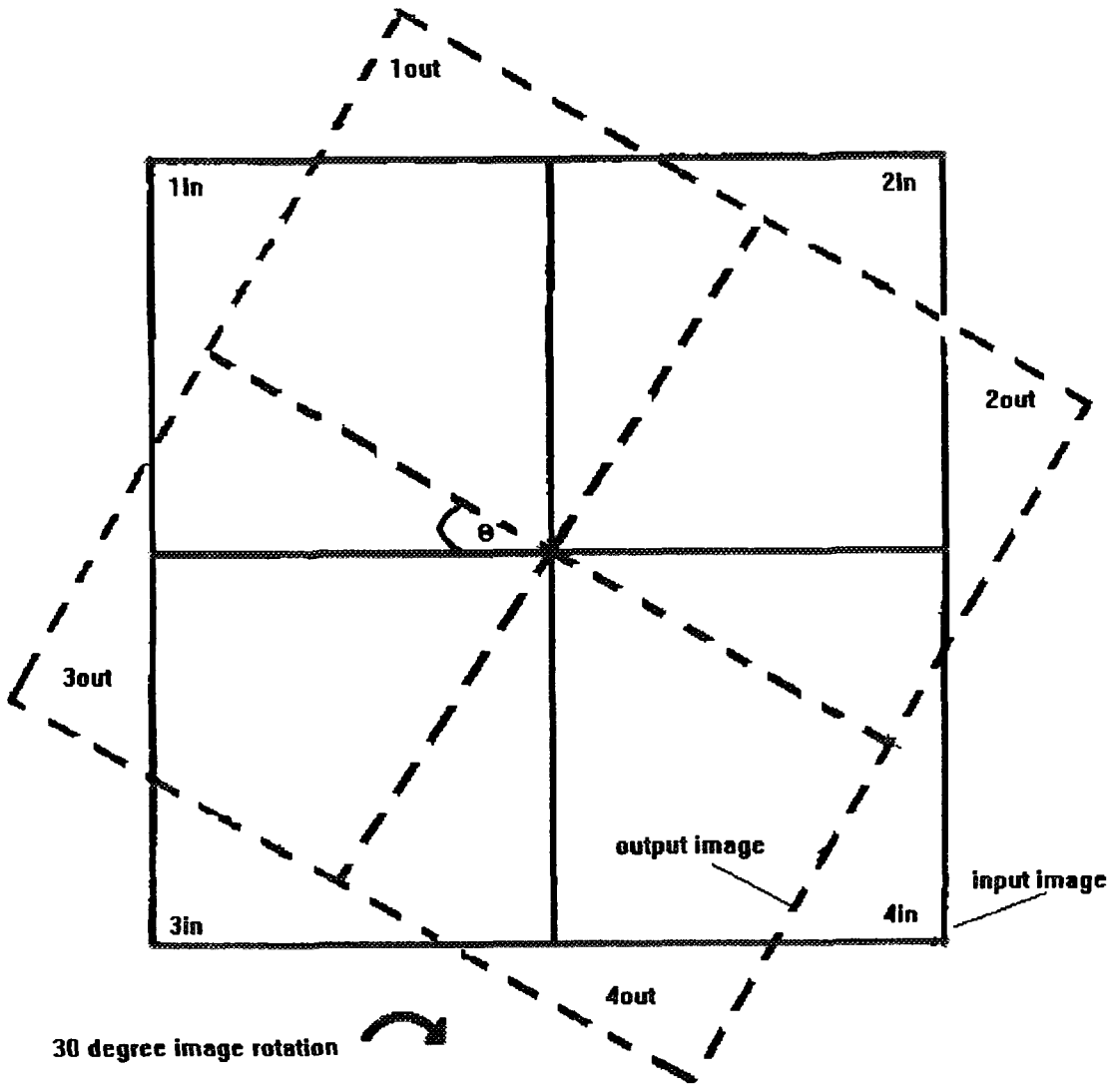

FIG. 3. Image partitioning.

(iv) If every tile is assigned its own coordinate frame relative to a global coordinate frame, then the output tiles can be relocated in the output image using a homogeneous coordinate transformation.

(v) For memory economy, it is desirable to build the final output image over the input image. For these savings to be implemented when there are more tiles to be processed than there are processors, the order of tile processing must be carefully managed to ensure that the input image data are not overwritten until they have been fully used. Since disk space is cheap, the simplest solution is to rebuild the output image from disk via a suitable buffering process.

IMPLEMENTATION

A PC-based, software-configurable, transputer array of 18 processors running under the Helios operating system with the Helios C compiler and Helios PC graphics library was the platform used in this work to develop the techniques. To perform a tiled image rotation, the communication tasks and the master and slave tasks were coded into a range of multiprocessor structures.

TESTS

Initial tests were carried out on simulated images. The input image was divided into sets of tiles by repeated bisection of the image. Each of the tiles was then sent to a separate slave processor for independent rotation. Bisection was chosen for its ease of implementation, since, with this, tiles could be stored contiguously in the input image memory and related by a single pointer offset. Data communication refinements and removal of processing/communication bottlenecks led to overall speed enhancements. A discussion of these is given after the results.

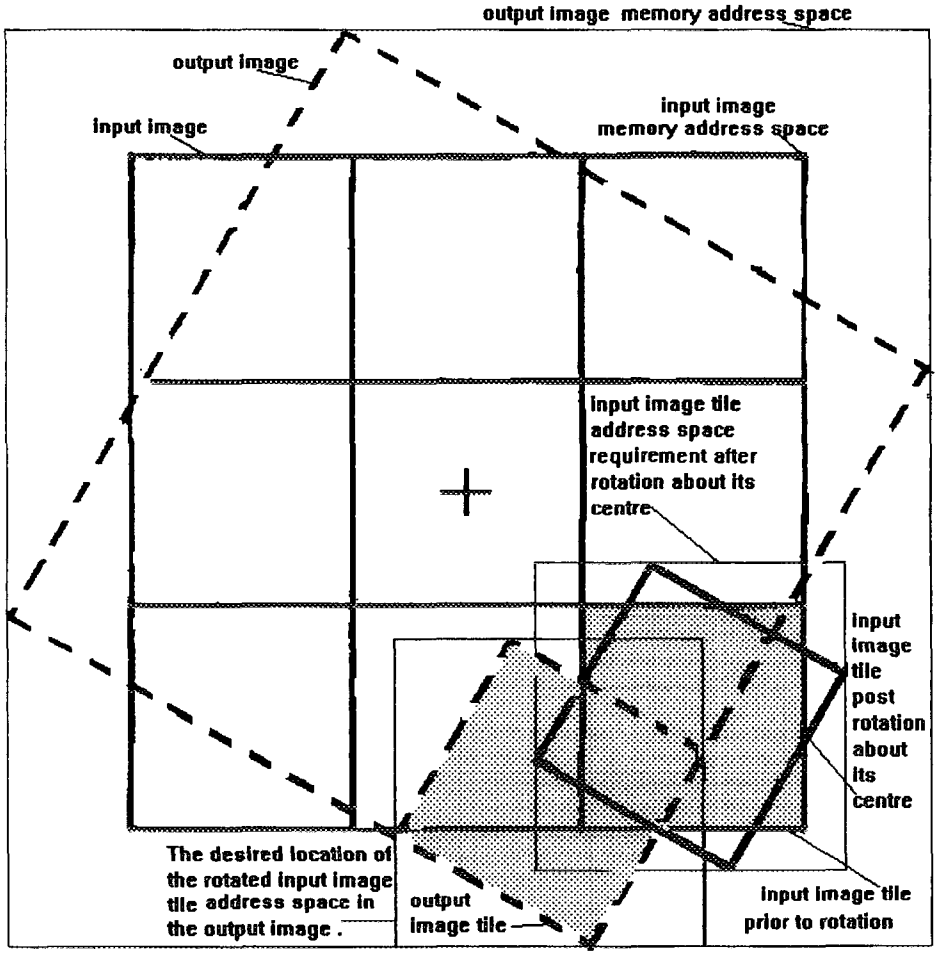

FIG. 4. Image rotation in tiles.

RESULTS—REFINED IMPLEMENTATION

The methodology developed was run with real images provided by Matra–Marconi Space (ERS-1 data).

Results from interpolating for a rotation of the image grid through 30° with our multiprocessor rotation scheme, using the optimized interpolation techniques, show the speed enhancement; see Table I.

Result A gives the time taken to rotate one tile using one slave processor (nontiled image rotation). This time includes the time taken to communicate the rotation pixel mapping data onto the slave.

Result B shows the time taken to rotate two tiles using two slaves. This time includes the image reconstruction times. It shows that the image reconstruction imposes only a small overhead (0.16 s).

Result C gives the time taken to rotate four tiles using two slaves. Note that result C is less than two times that of B. This result illustrates that the processing required to reconstruct the image can be largely compensated by subsequent tile processing on the same slave processor, since the table is only communicated to each slave once (the benefit of using an identical pixel mapping table for all tiles). Results D and E reinforce this.

Results F, G, and H illustrate the further benefit to speed of adding additional slave processors.

Figure 5 gives a graph of the interpolation rates derived from these results. The graph shows the image interpolation rate versus the number of slave processors employed. The processing rate increases in proportion to the number

TABLE I
Rotation Times for Increasing Number of Tiles

Test	Tiles/processor	Image size (pixels)	Rotation time (s)
A	1 tile, 1 slave	160 by 160	4.5
B	2 tiles, 2 slaves	160 by 320	4.66
C	4 tiles, 2 slaves	320 by 320	8.33
D	6 tiles, 2 slaves	480 by 320	12.01
E	8 tiles, 2 slaves	640 by 320	15.68
F	12 tiles, 3 slaves	640 by 480	16.11
G	16 tiles, 4 slaves	640 by 640	16.63
H	20 tiles, 5 slaves	640 by 800	17.11

Note. Test input image tile dimensions: 160 by 160, 8-bit pixels; test output image tile dimensions: 176 by 170, 8-bit pixels; slave rotation times: intermediate image pass, 1.83 s; final image pass, 1.81 s; total tile rotation, 3.64 s.

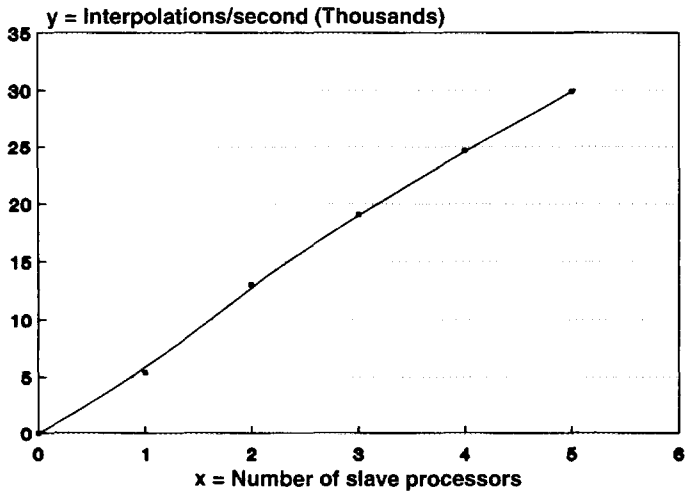

FIG. 5. Multiprocessor advantage.

of processors used, except for a small loss due to the cross-tile activity.

Memory Space Reduction

With the technique described, the interpolator driving look-up table for any image line needed only the pixel offset for that line and the address for the first pixel value in the new coordinate frame. This was sufficient information to perform the image rotation, as all subsequent pixels in a line were separated by the same offset.

Data File Transfer

The retrieval of look-up table data and image data from disk can present a considerable overhead. A 20 Mbit/s transputer link, through a link interface to a host, should be capable of sustaining a 2 Mbit/s transfer rate. To approach this rate, the storage and retrieval of all information to-and-from disk were done in blocks, giving a data access rate of 1.789 Mbit/s.

Master Processor Operations

The time taken for the master processor to complete slave communication encompassed the time taken for the master to initialize a separate thread communication for each slave, communicate all input tiles and look-up tables to the slaves, and receive all rotated tile data returns. The communications and the complexity of the message routing increased with increasing number of slaves.

Rotated Image Tile Reconstruction

The time taken to rebuild the final image as the number of slaves was increased was largely offset in our method by the savings gained from transfer of the mapping table only once per slave (only a small tapering-off of speed improvement was incurred as the number of slave processors increased).

Slave Processor Performance Improvement

In the intermediate image pass of a single tile rotation in the two-slave example, 29,920 interpolations were implemented in 1.8 s. Since each interpolation should theoretically require 13 floating point operations, this gives the processing rate actually achieved as $(29{,}920 \times 13)/1.8 = 0.216$ MFLOPS. The performance of the image rotation scheme as a whole could be improved by an optimized interpolator [8, 10]. With this, a processing rate of 0.89 MFLOPS would be attained.

Tile Communication Improvement

The way an image was tiled had a significant effect. If the tile geometry was far from square, the output tiling had to be significantly larger than the input tiling to achieve rotation. The output tiles, however, only contain the same amount of useful image information as the input tiles. In a 45° rotation, two-slave case, for example, an output tile contains $208 \times 160 = 33{,}280$ pixels, of which only $200 \times 72 = 14{,}400$ pixels, or 43.3%, contain useful image information. The reason for this unsatisfactory situation is the rectangular shape of the input tile. In general, the most efficient output tile data transfer occurs with square input tiles.

This technique provides an alternative solution to the process and data decomposition approach [11]. With our technique, computational time is not dependent on the angle of rotation. Further, our technique is suited to real-time applications on low-cost systems. It has been used effectively on a low-cost workstation for correction of image warping at SAR image edges.

CONCLUSIONS

A technique for the rotation of a large image, involving subtiling, concurrently rotating subtiles, and reassembling these to form the transformed image, has been demonstrated. Our technique of image rotation provides a good method for improving the image rotation speed for large digital images. It brings such image processing for near real-time applications into the realm of small, low-cost systems.

ACKNOWLEDGMENTS

The authors thank Matra-Marconi Space for their considerable scientific input in the work described in this paper and the supply of SAR image data used for this work. Thanks must also be given to IBM for their sponsorship of the parallel processing equipment. The work was carried out under a Science and Engineering Research Council (SERC) grant together with Matra-Marconi Space CASE support. It was undertaken at the request of Matra-Marconi Ltd.

REFERENCES

1. Meier, E. H., and Nuesch, D. R. Registration of spaceborne SAR data to large scale topographical maps. *19th International Symposium on Remote Sensing*, 1985.

2. Schreier, G., and Winter, R. The German ERS-1 SAR post-processing system plans and concepts. *Proceedings IGARSS '86 Symposium*, Zurich, Sept. 8–11, 1986, Ref. ESA SP-254.

3. Flannery, W. H., Teukolsky, S. A., and Vetterling, W. T. Numerical Recipes. Cambridge Univ. Press, Cambridge, UK, 1986.

4. Friedmann, D. E. Two-dimensional re-sampling of line scan imagery by one-dimensional processing. *Photogram. Engrg. Remote Sensing* **47,** 10 (1981), 1459–1467.

5. Friedmann, D. E. Operational re-sampling for correcting images to a ge-coded format. *Fifteenth International Symposium on Remote Sensing of Environments*, Ann Arbor, MI, 1981, pp. 195–212.

6. Paeth, A. W. A fast algorithm for general raster rotation. *Proceedings of Graphics Interface 86-Vision Interface 86*, pp. 77–81.

7. Danielsson, P. E., and Hammerin, M. High accuracy rotation of images. *CVGIP Graphical Models Image Process.* **54,** 4 (1992), 340–344.

8. Cronk, T. E. Application of transputers to high speed interpolation of complex SAR images. Doctorate Thesis, Univ. of Portsmouth, July 1993.

9. Hollis, J. E. L., and Cronk, T. E. Transputers used in interpolation of SAR image re-sampling. *Meiko International Conference*, Southampton, Apr. 1993.

10. Hollis, J. E. L., and Cronk, T. E. Measurement techniques for the performance of multipoint interpolations. *Trans. Inst. MC* **17** (1995).

11. Arabnia, H. R. A parallel algorithm for the arbitrary rotation of digitised images using the process and data decomposition approach. *J. Parallel Distrib. Comput.* **10** (1990), 188–192.

Received July 5, 1994; revised December 20, 1994; accepted February 6, 1995

J. E. L. HOLLIS graduated in electronic engineering from Imperial College, London University in 1962. He then undertook, on invitation, a doctorate at Imperial College (1966). A 10-year period of service with the Ministry of Defence provided experience in the design and installation of electronic systems. He entered the academic environment first at Imperial College and later at Portsmouth Polytechnic (University of Portsmouth). He was appointed in 1984 to Director of Resources in the Department of Electronic Engineering, at Portsmouth. He has also worked in industry at Plessey Ltd., UK, and Motorola Ltd., FRG. In 1991, he joined the Cranfield University Communications and Information Systems Engineering Group, where he is currently engaged in research leadership on image processing. He has held many research contracts and consultancies which have resulted in invitations to give invited papers and chair sessions at international conferences. He holds several patents for his ideas in the field of instrumentation systems. Currently, he is engaged in research on ground classification with the Silsoe Agricultural Centre.

T. E. CRONK was initially schooled at the Nautical College, Sunningdale, and pursued his higher education with the University of Portsmouth, graduating in electronic engineering in 1989. He joined Matra–Marconi Space Ltd., Portsmouth on a sponsored research studentship in the space systems group. Research into application of transputers to high-speed interpolators for complex radar images with matra–Marconi led to his registering for a higher degree at Portsmouth, which he completed in 1992. His previous experience includes digital design work with British Petroleum Research Centre, in the computers and communications engineering department. Since obtaining his doctorate, he has held a teaching associateship with the University of Portsmouth in computer-aided design. Subsequently, he was with Bytech Engineering Ltd., Basingstoke, where he was responsible for transputer applications. Currently, he is with Hitachi Europe Ltd.