M. Fleury
L. Hayat
A.F. Clark

Indexing terms: Image processing, Image-strip method, Rotation algorithms

Abstract: Methods of parallelising grey-scale coordinate transforms for medium-grained asynchronous parallel processors are considered. The thrice-skew rotation transform, polar and log-polar transforms can all be parallelised by an image-strip method. Where communication bandwidth is limited or some form of shared memory is available, an alternative method of parallelisation is presented. Details of computational enhancements and of interpolation considerations are provided. Specialised rotation algorithms are also discussed.

1 Introduction

Although low-level image processing or pixel-based preprocessing can generally be performed by data-parallelism, the class of coordinate transforms require nonlocal access to data. In this paper we consider parallelisations of image rotation and of polar and log-polar transforms. Image rotation can be used to align images, or it can even be used before feature detection in one-camera range-finding [1]. Polar and log-polar transforms of frequency domain images can be used as part of the Mellin transform, for which the magnitude is scale, rotation and shift invariant [2, 3]. Polar transforms may also be used to transform images distorted by space-variant seidel aberrations, such as coma, so that they can be corrected by space-invariant filters [4, 5].

We assume a message-passing model of parallelism [6] with a distributed localised address space. If the underlying hardware does not give adequate support to global communication, it may be necessary to compromise with an alternative arrangement, which is also applicable to shared-memory machines. Initially, we used a transputer-based tree-like topology [7], in which global communication could be achieved by a mixture of intratree links and centralise switching for intertree messages. When we implemented the algorithms in a parallel-Unix environment [8], the shared bus made such communication problematic. Another rotation algorithm, which is intended for fine-grained implementations, is given in [9], when the coordinate grid is mapped to a hypercube using reflected Gray code. This is a multiphase algorithm capable of use on a lock-step mode (SIMD) computer, when it has computational complexity $O(\log_2 n)$. However, a scalable version on a mesh has complexity $O(n)$. In these terms, the present algorithms are of $O(n)$, but consist of one communication step and are *not* intended for memory $O(1)$ implementation.

Paper 2076K (E4), received 20th February 1995

The authors are with the Department of Electronic Systems Engineering, Essex University, Wivenhoe Park, Colchester, C04 4SQ, United Kingdom

2 Rotation algorithms

Raster scan methods of image rotation were introduced to allow VLSI implementations of image rotation at video rate [10]. Because the number of trigonometric calculations is reduced they are also suitable for general implementations. Since decomposing the rotation matrix into two submatrices [11] introduces distortion from scaling and is not a one-to-one mapping for some angles, an alternative method, developed thereafter, is to use a decomposition into three submatrices [12]

$$\begin{bmatrix} 1 & 0 \\ -\tan(\theta/2) & 1 \end{bmatrix} \begin{bmatrix} 1 & \sin\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\tan(\theta/2) & 1 \end{bmatrix}$$

when the half-angle trigonometric identities can be used to establish the equivalence to

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

The method involves three skews, the first and last in a horizontal direction and the intervening vertical skew as illustrated by Fig. 1. Some form of interpolation will be

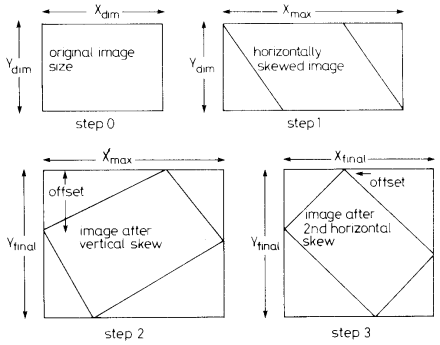

Fig. 1 *Thrice-skew rotation algorithm*

This work was carried out as part of project IED3/1/2171 (Parallel Reconfigurable Image Processing Systems).

necessary where a skewed pixel does not map directly onto the integer grid. The interpolation will depart from the ideal because the combination of horizontal and vertical skews will distort the region of support from the ideal of a circle to an elliptical shape [5]. Linear interpolation was used.* If a binary image is used then it is possible to use interval coding [14], in which case this algorithm is particularly appropriate, as it is only necessary to reassign the start and end run-length indicators for each interval.

Graphics applications will prefer the complete rotated image, in which case the target image will be enlarged to the following dimensions [15]

$$x_{final} = \lfloor y_{dim} \sin\theta + x_{dim} \cos\theta \rfloor + 1$$

$$y_{final} = \lfloor x_{dim} \sin\theta + y_{dim} \cos\theta \rfloor + 1$$

where x_{final} is the final row length and x_{dim} is the original image row dimension. Image processing applications will typically be content with an in-place mapping of the source image to its rotated counterpart as sensor distortion may affect edge regions. Fig. 2 was produced from

Fig. 2 *Truncated image with central origin, size* 1024 × 768

the complete image by shifting the origin and truncating the image in memory space.

2.1 Parallel implementation

The image was divided into overlapping strips, in a manner suitable for row-major addressing. Each parallel task performs the first horizontal and vertical skews on its strip. The horizontal skew will require a target buffer, which may exceed the horizontal extension of the final image

$$x_{max} = \left\lfloor x_{dim} + y_{dim} \tan\frac{\theta}{2} \right\rfloor + 1$$

Each vertically-skewed column can potentially occupy anywhere within the whole of the target image's equivalent column. The solution is to split the target image into strips and use new tasks to receive those parts of the vertically skewed columns which occupy the appropriate target image strip (Fig. 3). Each strip portion is referenced by destination task, polar coordinates and length.

The advantage of this method is that, where communication can take place independently of the CPU, as is the case for the link engines on the transputer [16], once the destination is selected, calculation of the next column can proceed in parallel. The destination for each column

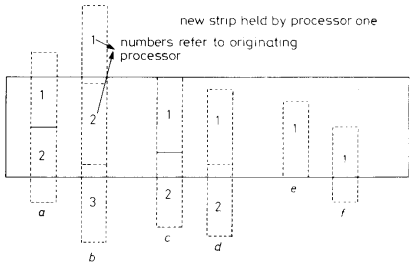

Fig. 3 *Receiving strip portions at a destination task*

The new strip is held by processor 1
a Part of strip must be truncated; two strip portions map onto new strip
b Three strip portions map onto new strip
c No truncation occurs; two portions map onto new strip
d Vertical skew results in two portions mapping onto new strip
e Exactly one portion maps onto new strip
f Portion maps partially onto new strip

portion is selected from a look-up-table (LUT) of target image boundaries. When provision for threads is available [17], the receiver thread can be accommodated on the same processor and within the same task as the generating thread, with limited context switching overheads, particularly if linked-list context switching is supported in firmware [18].

The final phase of the algorithm is the horizontal skew of the target strips, which is performed on the receiver tasks. Before this can commence, the receiver must ensure that it has received all its column portions. A simple solution to this problem is to use a global broadcast from the generating tasks, but this will not allow full parallelism. The alternative is to precompile a LUT of expected message numbers for each receiver task and for each of a range of expected rotation angles. The difficulties with the LUT approach are that

(i) the table may use up excessive memory
(ii) the range of image sizes is not flexible, and
(iii) the construction of the LUT generating program is not easy as the inverse mapping of the vertical skew operation is further complicated by the need to allow for boundary effects and fractional image strip sizes. The number of messages can alternatively be calculated at run-time, if the calculation time can be offset against start-up operations on the other tasks.

Rotations of $n\pi/2$ for integer n are related to image transpose algorithms. One goes between transpose and rotation of $\pi/2$, and vice versa, by column reversal. A transpose is one of a class of ordered rearrangements of data specified by a bit-permutation vector addressing method [19]. Remembering the maximum journey time on a hypercube, the algorithm has complexity $O(\log_2 n)$ for memory $O(1)$ machines. When this approach is extended to arbitrary angles, one has the problem of interpolation, except for binary images. The feasibility of SIMD implementations for rotation and other affine transformations will be dependent on what processing is necessary later on. Rotation of binary images might be better served by run-length coding of the image, combined with the thrice-skew algorithm on a medium-grained architecture. This usually means a MIMD

* This is not the only possibility. Choosing the best of three out of four values [13] alternative interpolation schemes are under investigation.

Table 1: Parameters for successive skews and memory offsets

	Positive angle		Negative angle	
	offset	skew-factor	offset	skew-factor
Horizontal skew 1	0	$\tan\frac{\theta}{2}$	$(y_{dim}-1)\tan\frac{\theta}{2}$	$-\tan\frac{\theta}{2}$
Vertical skew	$(x_{dim}-1)\sin\theta$	$\sin\theta$	$-(y_{dim}-1)\tan\frac{\theta}{2}\sin\theta$	$\sin\theta$
Horizontal skew 2	$\tan\frac{\theta}{2}$	$\tan\frac{\theta}{2}$	$x_{dim}\sin\theta\tan\frac{\theta}{2}+x_{final}-x_{max}$	$\tan\frac{\theta}{2}$

implementation. Another topology suitable for transposition is the shuffle-exchange network. On fixed-valence machines it is possible to map this topology to a de Bruijn network in MIMD-style [20]. This and others are specialist topologies, which cannot necessarily also be used for the ubiquitous data-farm [21] because they do not maximise bandwidth to the data farmer.†

Apart from positive rotations ($0 < \theta \leqslant \pi/2$), the full range of rotations was achieved here by including image-flips and negative (clockwise) rotations. Image flips to achieve a rotation of π can be made in parallel on a per-strip basis. The strip identity must also be complemented

$$id_{new} = s - id_{old} - 1$$

where s is the number of strips. For each horizontal skew the new x coordinate is chosen by x_{new} = (skew-factor) $\times\ x_{old}$ + (offset). The offset is necessary to ensure the skewed row remains within the computer's memory space, where the image origin is conventionally located at the top left-hand corner as the image is looked at. The values for the skews and offsets are listed in Table 1.

Although the number of messages passing between processors was not small, Fig. 4, speed-ups were achieved

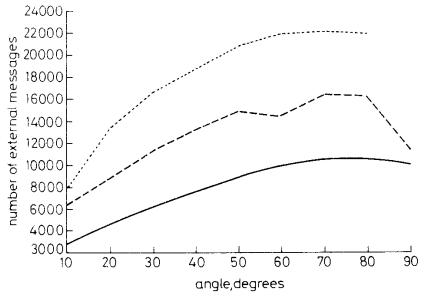

Fig. 4 *Message numbers for parallel rotation*
15 processors
12 processors
9 processors

by the strip method on a tree-like arrangement of transputers. The timings depended on the angle of rotation with larger rotations generating more messages, Table 2.

2.2 Distributed implementation

When the algorithm was transferred to a parallel-Unix environment, utilising one of the recent message-passing harnesses [25], it was found that the absence of threads and the limited bandwidth on the shared bus made the strip method unsatisfactory. If localised access to the complete image is possible, either by shared memory or local copies of the image, then an alternative, conceptually simpler, algorithm is possible. The target image is divided into strips. Each row of the target image is scanned, performing the inverse transform to locate the matching position in the original image. As this will not normally be an exact match on the integer grid, the surrounding four pixels are selected, so as to use bilinear interpolation. The timings can be enhanced by using an incremental method, once the rotation calculation for the first position is made, by observing that the difference between the following two calculations is only $\cos\theta$ in the x direction and $\sin\theta$ in the y direction.

$$\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}x\cos\theta - y\sin\theta\\ x\sin\theta + y\cos\theta\end{bmatrix}$$

$$\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x+1\\ y\end{bmatrix}=\begin{bmatrix}x\cos\theta - y\sin\theta + \cos\theta\\ x\sin\theta + y\cos\theta + \sin\theta\end{bmatrix}$$

This calculation method, which can also be applied in the y direction, may be prone to rounding errors [10]. The inverse-transform method takes constant time whatever the angle.

Even with small images, using the inverse-transform method gave no advantage as there is no local image access. It can be concluded that for standard parallel-Unix implementations there exists as yet *no satisfactory* method of parallelising image rotation.

Table 2: Timings for various angles of rotation for a 1024 x 768 pixel image

Angle, degrees	Timing, s			
	Number of processors			
	9	12	15	18
10	5.28	4.13	3.54	3.26
20	6.12	4.77	4.45	4.53
30	6.87	5.63	5.61	5.73
40	7.21	5.80	5.69	6.49
50	7.56	6.80	6.56	6.74
60	7.77	5.94	6.34	6.92
70	7.17	6.56	6.72	6.93
80	6.37	5.98	6.35	6.94

3 Polar and log-polar algorithms

This Section describes how the rotation algorithm method was extended to polar-type transforms. The log-polar transform differs from the polar transform only by a different quantisation in the radial direction (Figs. 5–7).

† A number of serial methods exist for matrix transpose on a memory-restricted computer (or for large matrices) [22–24], but a parallelisation of these will only be suitable for time-critical applications, as the speed-up is $S = p/\log_2 n$, where p is the number of processors.

In Reference 3 a sequential log-polar transform phase is identified as the bottleneck in an otherwise parallel implementation of a Mellin transform. In the present implementation, a strip method algorithm made it possible to use a similar hardware setup as for image rotation (Fig. 8). Using transputers on the Parsys Supernode

Fig. 5 *Original 512 × 512 image*

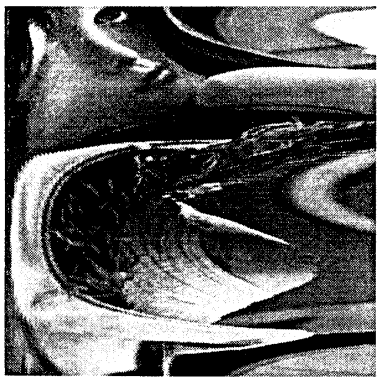

Fig. 6 *Polar transform*

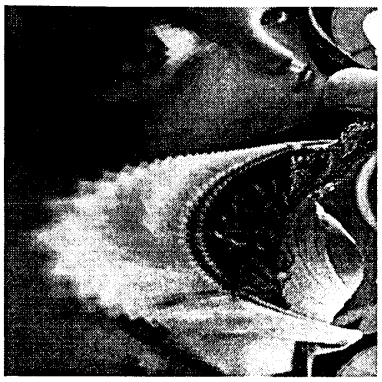

Fig. 7 *Log-polar transform*

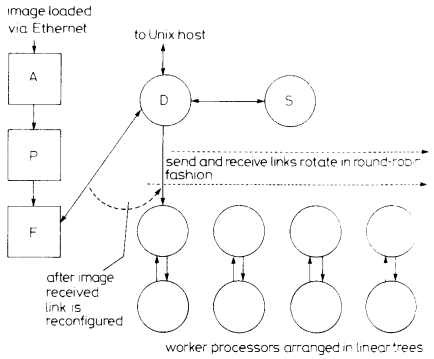

Fig. 8 *Hardware arrangement using reconfigurable links for image acquisition and data distribution*

A Abekas framestore
P PAL encoder
F frame grabber
D data manager
S switch control unit

[26], a data-manager processor could service a number of small groups of processors by means of link switching, without the overhead of store-and-forward communication over a single large tree. A one-level ternary x-tree for each group is optimal, but the reduced scale of message-passing for the polar-type algorithms made it easier to implement linear trees. Also shown is a way of using the reconfigurable link to acquire images directly, which is advisable since the transputer does not have hardware-assistance for memory management.

These algorithms were performed in an in-place fashion using the same memory on the data manager for the input and output image. A central origin is chosen, thus restricting the message destinations for those tasks holding strips in one half of the image to destinations in the same half. The use of a central origin requires the worker tasks to remap the x and y co-ordinate ranges. If threads are used, the same task will also collect a range of angles. A suitable mapping from the order of loading image strips to the order of collecting results in a designated angle range is shown in Fig. 9. To achieve an

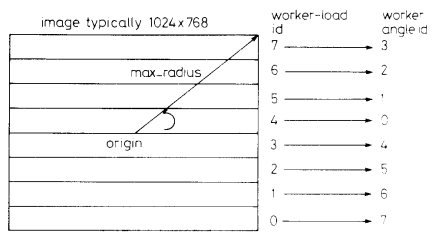

Fig. 9 *Mapping of load order and angle range identities*

optimal angular range it may be necessary to interchange the dimensions for rectangular images (Fig. 10). If the polar field is taken to be the inscribed circle, then it is possible to fill the target image. However, in the parallel

algorithm, there are points in favour of using an exscribed circle (Fig. 11).

For each radial spoke that is formed according to the angular quantisation, each task steps out on the segment of the spoke falling across its strip (Fig. 12). In the figure,

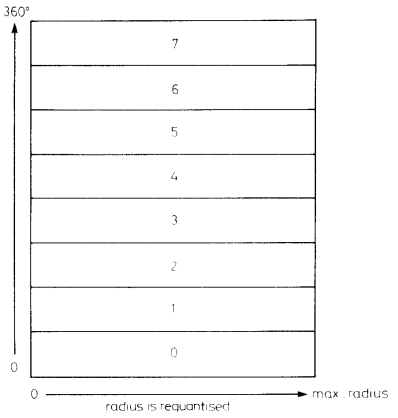

Fig. 10 *Quantisation of the target image for a parallel in-place computation*

Target image has side lengths interchanged. Suitable quantisation is used to fit angle range into original side length

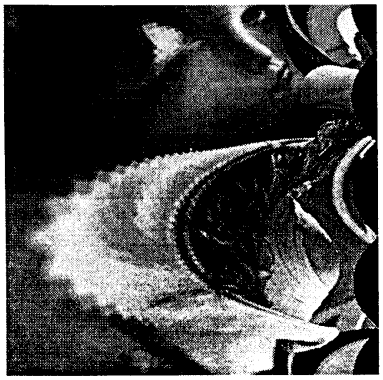

Fig. 11 *Polar transform, using an exscribed circle*

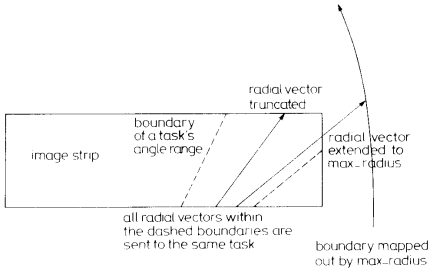

Fig. 12 *Image strip geometry for an exscribed circle*

the radial vector is allowed to pass across the complete strip. Where sampled segments fall outside the dashed lines, these segments are relayed to the appropriate collecting task. Within the dashed lines, the originating task retains any sampled segments. At vertical strip boundaries, overlap can be used, but it may also be possible to repeat the preceding row where the image field is well-correlated. Beyond the vertical edge the segment is set to a background shade until the maximum radius is reached. At horizontal edges, wrap-around is one possibility. When an inscribed circle is used, separate calculation is needed to find whether the radial spoke has breached the limit of the circle. This calculation is not necessary for an exscribed circle since the radial spoke is always breached by the fixed boundaries of the strip. Whatever the method, a number of trigonometric calculations are necessary to establish the boundaries of the strips. For instance, the start of a worker task's angular range is given by arctan (y_{strip}/x_{strip}), where x_{strip}, y_{strip} are given in the remapped coordinates. Message bookkeeping calculations are performed in a similar manner to the rotation algorithm. The pattern of message passing is unavoidably irregular. For instance, in the log-polar case many more messages originate from the central strips due to the nonlinear quantisation. A sample timing using eight transputers is 9.44 s for a 1024 × 768 image using a polar transform.

3.1 Inverse-transform method

An inverse-transform method, mapping to the target image from the original, is simpler to implement. It also has the virtue that consistent sampling in the output image is ensured. This method is used in serial routines [27]. Computational efficiency can be enhanced by using a LUT to map from the logarithmic quantisation to the linear image scale, as indicated in pseudo-code

```
start = 1.0
finish = log(rowSize)
step = (finish − start)/rowDimension
for each line in the target image
    lineRadius = exp(start + lineNumber * step)
```

Quick calculation of trigonometric values is accomplished by using an iterative method, based on De Moivre's theorem, which also avoids rounding error [28]

$$\zeta_0 = 1$$

$$\zeta_{k+1} = \zeta_k + \eta\zeta_k$$

$$\eta = \exp(i\theta) - 1$$

$$= 2i \sin(\theta/2) \exp(i\theta/2)$$

$$= -2 \sin^2(\theta/2) + i \sin(\theta)$$

If four tasks are used, the image is split into quarters rather than into strips so that all calculation is self-contained. A problem arises with identifying the origin, which is conveniently avoided by a one-pixel overlap upon the central pixel (Fig. 13). For even dimensions, the central pixel is taken as offset by one from the actual origin. This method avoids the quantisation errors which otherwise arise, especially with logarithmic sampling.

As there is almost no contention, this method is appropriate for shared memory or overlapped memory machines [29]. For message passing machines, the

method only slightly improves upon the rotation algorithm equivalent routine if it is necessary to transfer each quarter of an image. The method also scales weakly because the data transfer load does not decrease when more tasks are used.

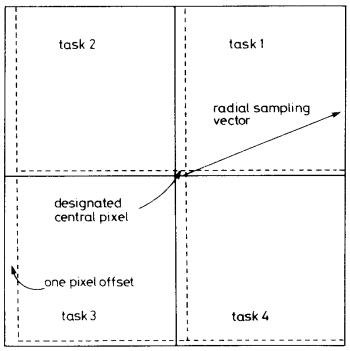

Fig. 13 *Coping with the origin*

4 Conclusions

There are no obvious ways to parallelise the class of algorithms given here, though an efficient parallelisation would be advantageous for the range of applications reviewed. For image rotation, polar and log-polar transforms (as well as no doubt for other mappings not considered here) it is possible to use a generic strip method, giving competitive, though not real-time, performance. This method is most suitable when there is a good compute-to-communicate ratio, as provided on modular parallel machines, such as those that are based on the transputer or the C40 [30]. Each parallel task performs the initial stage(s) of the transform, passing output to another set of processes responsible for collation of the resulting image strips. An efficient implementation will make use of threads or light-weight processes. Different algorithms will require detailed consideration of the correct mapping from the original to target image. An alternative method, which avoids the mapping problem, is to scan the target image, selecting from the original image. This is most suitable for shared-memory machines, where a way of limiting contention in the case of the polar/log-polar transform is described. For parallel environments which fall outside the two paradigms, it is a moot point whether parallelisation is possible.

5 References

1 ISHIGURO, H., YAMAMOTO, M., and TSUJI, S.: 'Omni-directional stereo for making global map'. 3rd international conference on *Computer vision*, Osaka, Japan, 1990, pp. 540–547

2 ALTMANN, J., and REAITBOCK, H.J.P.: 'A fast correlation method for scale and translation-invariant pattern recognition', *IEEE Trans. Pattern Anal. & Machine Intel.*, 1984, **6**, (1), pp. 46–57

3 HALL, G., and MATIAS, A.: 'Rotation, scale and translation invariant template matching on a transputer network', *Microprocess. & Microsyst.*, 1993, **17**, (6), pp. 333–340

4 SAWCHUK, A.A.: 'Space-variant restoration by coordinate transformations', *J. Optical Soc. Am.*, 1974, **64**, (2), pp. 138–144

5 SCHALKOFF, R.J., and NAG, H.: 'Decomposition and parallel architecture for the geometric transformation of digital images', *Image & Vision Comput.*, 1991, **9**, (5), pp. 275–284

6 HOARE, C.A.R.: 'Communicating sequential processes' (Prentice-Hall, Englewood Cliffs, NJ, 1985)

7 FLEURY, M., CLARK, A.F., and HAYAT, L.: 'Performance estimation for a dynamically reconfigurable multiprocessor system'. In: 'Performance evaluation of parallel systems'. PEPS '93, University of Warwick, 1993

8 FLEURY, M., and HAYAT, L.: 'PVM performance: theory and practice'. Technical report, University of Essex, 1994

9 RANKA, S., and SAHNI, S.: 'Hypercube algorithms for image transformation'. International conference on *Parallel processing*, Pennsylvania State University. Vol. 3, 1989, pp. 24–31

10 TSUCHIDA, N., YAMADA, Y., and UEDA, M.: 'Hardware for image rotation by twice skew transformations', *IEEE Trans. Acoustics, Speech & Signal Process.*, 1987, **35**, (4), pp. 527–531

11 CATMULL, E., and SMITH, A.R.: '3D-transformations of images in scanline order', *Comput. Graphics*, 1980, **4**, (3), pp. 279–285

12 TANAKA, A., KAMEYAMA, M., KAZAMA, S., and WATANABE, O.: 'A rotation method for raster image using skew transformation'. Int. conference on *Computer vision and pattern recognition*, 1986, pp. 272–277

13 SMITH, P.R.: 'Bilinear interpolation of digital images', *Ultramicroscopy*, 1981, **6**, pp. 201–204

14 PIPER, J., and RUTOVITZ, D.: 'Data structures for image processing in a C language and Unix environment', *Pattern Recogn. Lett.*, 1986, **3**, pp. 119–129

15 WOLBERG, G.: 'Digital image warping' (IEEE Computer Society, 1990)

16 GRAHAM, I., and KING, T.: 'The transputer handbook' (Prentice-Hall, Englewood Cliffs, NJ, 1990)

17 3L LTD: 'Parallel C Version 2.2.2' (The Company, Peel House, Ladywell, Livingston, Scotland, 1991)

18 MITCHELL, D.A.P., THOMPSON, J.A., MANSON, G.A., and BROOKS, G.R.: 'Inside the transputer' (Blackwell Scientific Publications, London, 1990)

19 NASSIMI, D., and SAHNI, S.: 'Optimal BPC permutations on a cube-connected computer' *IEEE Trans. Comput.*, 1982, **31**, pp. 338–341

20 CREUTZBURG, R., MINKWITZ, J., ROGGENBACH, M., and UMLAND, T.: 'Parallel FFT-like algorithms on transputers', *in* PITAS, I. (Ed.): 'Parallel algorithms for digital image processing, computer vision and neural networks' (Wiley & Sons, New York, 1993), pp. 53–81

21 DOWNTON, A.C.: 'Top-down structured parallelisation of embedded image processing applications', *IEE Proc. Vision Image & Signal Process.*, 1994, **141**, (6), pp. 431–437

22 EKLUNDH, J.O.: 'A fast computer method for matrix transpose', *IEEE Trans. Comput.*, 1972, **21**, (7), pp. 801–803

23 PORTNOFF, M.R.: 'An efficient method for transposing large matrices and its application to separable processing of two-dimensional signals', *IEEE Trans. Image Process.*, 1993, **2**, (1), pp. 122–124

24 TWOGOOD, R.E., and EKSTROM, M.P.: 'An extension of Eklundh's matrix transposition algorithm and its application in digital image processing', *IEEE Trans. Comput.*, 1976, pp. 950–952

25 GEIST, A., BEGUELIN, A., DONGARRA, J., JIANG, W., MANCHEK, R., and SUNDERAM, V.: 'PVM3 user's guide and reference manual'. Oak Ridge National Laboratory, 1993

26 PARSYS LTD: 'Hardware reference manual for the Parsys SN1000 Series' (The Company, Boundary House, Boston Road, London)

27 CLARK, A.F.: 'ALG collection of image processing routines'. Available from Essex University Image Processing Archive.

28 SINGLETON, R.C.: 'An algorithm for computing the mixed radix fast Fourier transform', *IEEE Trans., Audio & Electracoust.*, 1969, **17**, (2), pp. 93–103

29 HAYAT, L., and SANDLER, M.B.: 'An efficient multidimensional multidirectional parallel pipelined architecture for image processing'. IEE international conference on *Digital signal processing*, UK, 1991, pp. 105–110

30 TEXAS INSTRUMENTS: 'TMS 320C40 user guide', 1991