Evaluating the performance of parallel programs in a distributed environment

M.Fleury
L.Hayat
A.F.Clark

Indexing terms: Parallel programming, Distributed processing, Data-farming programming, Message-passing software, Diffision model

Abstract: The paper considers methods of evaluating the performance of programs using recent communication harnesses. A demand-based data-farming parallel programming paradigm is used. Possible techniques for performance prediction are examined and a description of a particular method is given, for a shared distributed environment, involving a diffusion approximation. Selected results from a benchmarking study are given to establish the validity of the performance model. A theme of the study is a comparison with a previous transputer-based implementation.

1 Introduction

This paper charts the development of a performance model applicable to running parallel programs in a distributed environment. Specifically, a diffusion approximation has been adapted to fulfil this purpose. The validity of this approximation is detailed in the paper. This performance model is one of a number discussed herein that are relevant to the use of a data-farming programming paradigm. It is assumed that other users or processes share the distributed environment. The results of the study have a bearing on message-passing communication harnesses such as PVM [11], MPI [28], TCGMSG [14], P4 [2] and Express [19].

We describe the background to the development of the performance model and indicate the distinctive features of the distributed environment which any model should account for. We also examine other analytic solutions to the performance evaluation of message-passing software for static and dynamic processor networks. These assume store-and-forward communication. We give the diffusion approximation and establish its adequacy for the present purposes. Non-analytic approaches to performance evaluation for the present environment are adumbrated.

IEE Proceedings online no. 19960068

Paper first received 20th February 1995 and in final revised form 18th September 1995

Dept. E.S.E., Essex University, Wivenhoe Park, Colchester CO4 4SQ, UK

2 Parallelisation methods

Our problem was to develop a suite of low-level image-processing software which could operate in parallel as well as serially. A typical image-processing application passes a 3 × 3 window over a 1024 × 1024-sized image which, without taking short cuts, needs about 9 million multiplications, 9 million additions and 1 million divisions. At least one Mbyte is needed to store the image, which may require manipulation of the virtual memory system of the processor. In the main, this is an engineering problem as opposed to one of algorithm design, since most, though not all, algorithms can employ data-parallelism as the method of problem decomposition. It is then possible to devise optimised sequential code for processing overlapping strips of the image.

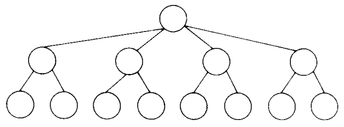


Fig. 1 *Tree-type topology for the reconfigurable transputer array*

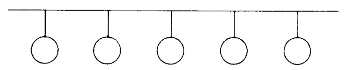


Fig. 2 *Shared bus topology for parallel-Unix applications*

Having initially used a dedicated transputer-based parallel processor [20], we wondered whether an accessible solution was possible using the PVM (Parallel Virtual Machine) communication harness, which is freely available. The attractions for us were portability and ease of use. Although it was an original intention of the PVM designers to accommodate heterogeneous hardware, we performed tests on a homogeneous set of SUN 4 architecture workstations. In common with the likely end-users of our package we worked in a shared environment. For a study of other communication harnesses used in an isolated environment see [7]. We effectively moved from a modified tree topology to a shared bus topology (Figs. 1 and 2). Any global communications can be adequately achieved in the transputer set-up by intratree routing, with centralised routing for intertree communication, but in the shared bus environment all communication competes for the same resource. The transputer link, used here at 10

Mbps, has the same raw rating as the Ethernet (or IEEE 802.3 MAC standard), but communication can effectively take place in parallel and independent of the CPU.

3 Parallel Unix environment

When using a shared environment there are two inherent uncertainties: the effect of short-term process scheduling and the effect of the network. If a demand-farming method of processing is used, then the data-farmer and its set of worker tasks are both called upon to act sporadically, which will cause difficult-to-predict interactions with the short-term process scheduler. Unix uses a round-robin scheduling algorithm with multilevel feedback [1] which, as implemented on SUNs [4], strongly favours short jobs. There will be a reduction in the mean response time; but it will also mean that pinging a processor by testing a sample computation run will overestimate the speed of a processor under a particular workload if the pinging duration is less than that of the job. (Here, 'pinging' means timing a round-journey between data-farmer to worker, including the time for an intermediate sample computation). This means we have to adjust a short pinging forecast by the use of a nonlinear function, which will be based on heuristics. Though we did have some success with static scheduling of PVM tasks, in the main we found that the demand-farming method of load-balancing was likely to give better results [9]. Comparison between serial and distributed performance is also problematic under this regime. Exponential process service times with FCFS (first-come-first-served) service discipline can provide an upper bound for scheduling efficiency, but this tells one nothing about the experience of an individual set of processes.

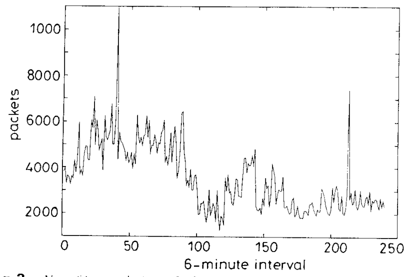


Fig.3 *No. of inputs during a 24-hour weekday, from midnight*

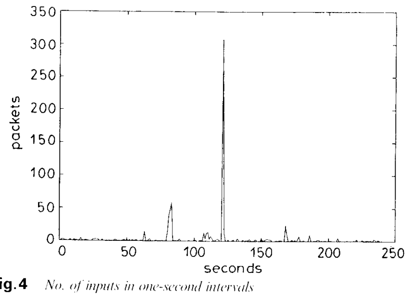


Fig.4 *No. of inputs in one-second intervals*

We made measurements of network activity using a passive software monitor. A run from midnight to midnight on 15 SUN workstations gave figures for packet or frame input numbers; these are presented in Fig. 3. The traffic is 'bursty', with few intermissions when one might hope to run a lengthy PVM application. If we inspect this pattern at a lower resolution then we find a similar 'bursty' pattern, for instance Fig. 4. When a PVM application was repeatedly run, we observed a noticeable change both in the volume of messages and the congestion, indicated by the packet collisions occurring after the application starts, at about 200 on the horizontal scale in Figs. 5 and 6. The recorded collisions may represent a small part of the delay due to congestion, since the CSMA/CD (Carrier Sense Medium Access/Carrier Detect) protocol [13] causes a transmitter to delay if the transmission medium is sensed to be busy (which also means return order from different network nodes is not guaranteed). We tested the hypothesis that the number of packets in the system, without PVM running, formed a Poisson distribution by means of a χ^2 goodness-of-fit estimation using an index of dispersion method [17], but failed to establish a 95% confidence interval. We cannot therefore rely on analytic expressions for network behaviour, which normally assume Poisson statistics, to suggest the congestion that a PVM application will meet, though they might give lower bounds.

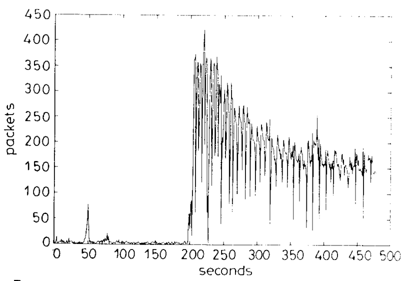


Fig.5 *No. of inputs for a PVM data-farming application (12 processors)*

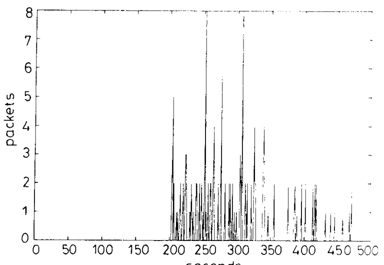


Fig.6 *No. of collisions for a PVM data-farming application*

4 Possible analytic approaches to performance prediction

Demand-based data farming is a method of load-balancing which has wide applicability, but it has the defect that queueing methods of performance prediction are doubtful, because the arrival and departure service rate processes are not independent. A method which has had wide currency [21, 27, 24] in predicting an upper bound to the performance of store-and-forward networks is by the use of a linear programming method. The constraint is the finite capacity of the links leading to the data-farmer at the root of the processor tree. The method can be adapted to any tree arrangement and does not require a homogeneous data-load. A sample result for a linear chain is

$$\frac{1}{(t_{\text{comm}} + t_{\text{setup}})} = \frac{1}{2t_{\text{setup}}}\left(1 - \left(\frac{t_{\text{calc}} - t_{\text{setup}}}{t_{\text{calc}} + t_{\text{setup}}}\right)^n\right)$$

where t_{calc} is the per work packet computation time (which might be a mean), t_{comm} is the time to transmit a work packet over a link and t_{setup} is the time the CPU uses to set up a transmission. The left-hand side of this expression represents the condition of link saturation to the head of the chain. The right-hand side must equal this to achieve maximum throughput.

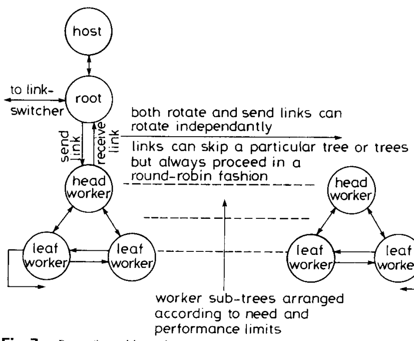

host
to link-switcher
root
both rotate and send links can rotate independantly
send link
receive link
links can skip a particular tree or trees but always proceed in a round-robin fashion
head worker
head worker
leaf worker
leaf worker
leaf worker
leaf worker
worker sub-trees arranged according to need and performance limits

Fig. 7 *Reconfigurable multitree*

If we use a dynamic architecture, as we did to achieve the topology of Fig. 1 by means of link switching (illustrated in Fig. 7), then this method can only be applied to the individual processor trees; but it is possible to arrive at a steady-state formula for the switching time delay W_{switch} [8] as

$$W_{\text{switch}} = \frac{Mt_{\text{switch}}}{1 - (M-1)(t_{\text{comm}} + t_{\text{setup}})T}$$

where M is the number of processor trees, t_{switch} is the time to switch a link and T is the throughput on an individual tree, assuming symmetrical output.

When the algorithm used results in Poisson flows, it is possible to move away from this type of formula to a polling model queueing analysis [26, 29]. We now find the mean delay per message as:

$$W = \frac{M\lambda E[S^2]}{2(1 - M\rho)} + \frac{(1+\rho)Mt_{\text{switch}}}{2(1 - M\rho)}$$

Here, M is the number of networks, S is the service rate, λ is the per network message arrival rate and ρ is the traffic intensity for each processor tree. We no longer need to use a hybrid combination of throughput and switching delay as they are subsumed in the same formula.

We now turn from the isolated environment of the transputer, where performance is understood, to that of the shared environment of the Ethernet network. A basic question is what is the communication bandwidth? In fact, there are analytic studies of a CSMA/CD protocol network using regeneration cycles, summarised in [15], which give the formula:

$$\bar{u}(\nu) = \frac{\gamma(\nu)e^{-\gamma(\nu)}}{a + a'(1 - e^{-\gamma(\nu)} - \gamma e^{-\gamma(\nu)}) + \gamma(\nu)e^{-\gamma(\nu)}}$$

where u is the traffic utilisation in a state where there are n stations out of a population of N delayed in access to the communication channel. The throughput is γ. $\nu = n/N$. a is the ratio of packet propagation time within the network segment to packet size and a' is the fraction of this ratio used up when there is a collision. This formula is arrived at by assuming a Poisson distribution for the arrival rate to the channel. If we took $a = 0.1$ and $a' = 0.1$, an asymptotic utilisation curve near to the overall bandwidth we experienced using PVM on the network is arrived at (Fig. 8). Since the measured distribution is not Poisson, we were led in other directions in searching for a solution to the PVM system, though the bandwidth analysis is useful in gauging benchmarking results.

A caveat to this is that the bandwidth for this measure is worst-case since it takes the maximum propagation time, while messages from centrally placed nodes will propagate to the whole of the network more quickly [3]. For $a = 0.1$, avoiding propagation differences does mean an underestimate, which implies that if the arrival traffic were to be Poisson then the ratio of packet propagation time to packet size is more unfavourable. Another *caveat emptor* is that where the access backlog is substantial then the system may suffer a cusp catastrophe [30].

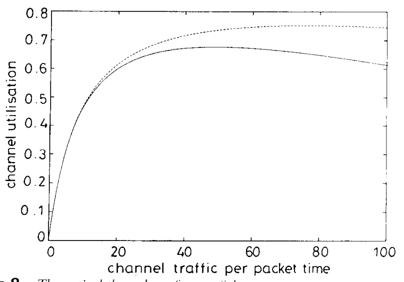


Fig. 8 *Theoretical throughput for $a = 0.1$*
$a = 0.1$, $a' = 0.5$ (solid line); $a = 0.1$, $a' = 0.1$ (broken line)

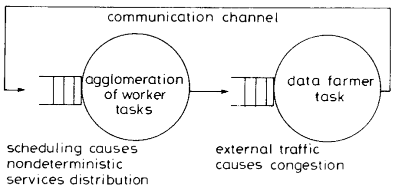

communication channel
agglomeration of worker tasks
data farmer task
scheduling causes nondeterministic services distribution
external traffic causes congestion

Fig. 9 *A simple model for PVM demand-farming*

5 Using diffusion theory

In this Section we suggest an alternative and time-effective way of modelling a PVM application, which is not necessarily characterised by Poisson distributions, through use of a diffusion approximation [23]. We model a PVM demand-based farming application by a simple cyclic model as in Fig. 9, which is a scheme borrowed from multiprogramming at a time when such operating systems were first being developed [10]. The queue considered is that for the agglomerated workers. The service rate would be deterministic (constant) at each worker if we did not have a generally distributed scheduling regime. The arrival rate covers the combined effect of service by the data farmer and transport over the communication channel. The model requires mutually independent service times and arrival times, to remove cross-terms, which is more justifiable than in the transputer-based models because of the shared environment. One difficulty with the diffusion approximation is establishing boundary conditions which ensure that the number of packets waiting, due to congestion in the communication channel, never becomes negative. We therefore expect a heavy traffic regime where one job leaving is replaced by another. In fact, apart from the initiation and wind-down, demand-farming is like this.

The approximation arises by setting the queue length at the agglomerated workers' queue, $N(t)$, as

$$N(t) = A(t) - D(t)$$

where $A(t)$ and $D(t)$ are time-dependent arrival and departure renewal processes respectively. $N(t)$ is then represented by a diffusion process probability density function, $f(n, t)$, which, in steady state, satisfies:

$$(\sigma^2/2)\frac{\mathrm{d}f}{\mathrm{d}n} - \nu f = 0$$

We have ν and σ^2 since as $A(t)$ and $D(t)$ are, at any time instance, approximately normally distributed [5], then $N(t)$ has instantaneous mean $\nu = \lambda - \mu$ and variance $\sigma^2 = C_a\mu + C_s\lambda$, with λ the arrival-rate, μ the departure-rate, C_s the coefficient of variation of the service times and C_a the coefficient of variation of the arrival times.

A solution [18] is found to be:

$$f(n) = \frac{(2/L)\exp(-2n/L)}{1 - \exp(-2c/L)}$$

where c is a finite limit to the number of jobs and

$$L = \frac{-(1-\rho)}{C_a^2\rho + C_s^2}$$

with the queue availability $\rho = \lambda/\mu$. For large c and infinite queue buffer we find the discrete approximation for the probability of queue length n, that is $\hat{\pi}(n)$:

$$\hat{\pi}(n) = \int_n^{n+1} (2/L)\exp(-2n/L)\mathrm{d}n$$
$$= (1 - \exp(-2/L))\exp(-2n/L)$$

If we set the approximate availability as:

$$\hat{\rho} = \exp\left(\frac{-2(1-\rho)}{C_a^2\rho + C_s^2}\right)$$

then, by a corollary to Little's theorem [16] for the value of $\pi(0)$, we have a formula for the probabilities of queue length n:

$$\hat{\pi}(n) = \begin{cases} 1-\rho & n = 0 \\ \rho(1-\hat{\rho})\hat{\rho}^{n-1} & n > 0 \end{cases}$$

It is then easy to find the mean number in the queue, n, (by a generating function method) from which, by Little's theorem, the waiting time and total cost can be found:

$$n = \frac{\rho}{1-\hat{\rho}}$$

The denominator in this formula represents the congestion effect. $\lambda = \mu$ is a pathological case, though the accuracy of the approximation increases as one approaches $\rho = 1$ because the number in the queue will be large.

The accuracy of this approximation (which can be improved by replacing a reflecting boundary by an instantaneous return process [12]) is illustrated by Table 1. (The exponential distribution has squared coefficient of variation equal to 1.) Thus columns 2 and 3 should be approximately the same. Column 5 uses the Pollaczek Khintchine (P K) formula [16] for an M G/1 queue where the arrival rate is still exponentially distributed. (The P–K formula ($n = \rho + \rho^2(1 + C_s^2)/2(1 - \rho)$) is one of the few results for which exact results are known, but note that as a point of comparison it is only possible to compare the diffusion approximation to a few exact analytic results for either Erlang-k or Hyperexponential distributions. These latter distributions have been used to approximate non-exponential distributions by cascading exponential-like stages but the solutions for the 'wrong' number of stages give unhelpful polynomial equations to solve).

Table 1: Comparing the diffusion approximation to exact results for mean queue number

C_a	1	1	1	1	2.5	0.5	1	1
C_s	1	1	0.5	0.5	1	1	2.5	2.5
ρ	Diffuse	M/M/1	Diffuse	M/G/1	Diffuse	Diffuse	Diffuse	M/G/1
0	0	0	0	0	0	0	0	0
0.1	0.12	0.11	0.11	0.11	0.13	0.12	0.20	0.12
0.2	0.27	0.25	0.22	0.24	0.30	0.26	0.45	0.29
0.3	0.45	0.43	0.36	0.40	0.54	0.43	0.76	0.53
0.4	0.69	0.67	0.54	0.60	0.89	0.63	1.18	0.87
0.5	1.03	1.00	0.79	0.88	1.39	0.91	1.76	1.38
0.6	1.52	1.50	1.16	1.28	2.19	1.31	2.64	2.18
0.7	2.35	2.33	1.78	1.93	3.57	1.95	4.09	3.56
0.8	4.01	4.00	3.02	3.20	6.41	3.22	7.01	6.4
0.9	9.01	9.00	6.76	6.98	15.08	6.99	15.75	15.08
0.95	19.00	19.00	14.26	14.49	32.54	14.49	33.25	32.54

Note that the approximation is in this case ($C_s = 0.5$) always lower and in the M/M/1 comparison it is higher. It is generally true, that for C_a fixed at 1, $C_s < 1$ underestimates and $C_s > 1$ overestimates. The next two columns explore areas where classical queueing theory has little to say. Thus columns 6 and 7 in Table 1 show the effect of raising or lowering C_a above or below 1. For completeness, the effect of lowering and raising C_s is also illustrated.

One might also compare the error functions for the difference in variation for each method (diffusion approximation or P K formula), when it would be found that the diffusion approximation is not such a good approximation for consideration of variance. In fact, a systematic error analysis has been made [22], where it is shown that the absolute error of the mean queue length is bounded as $\varepsilon_{abs} \leq (C_s^2 - 1)/2$ except in the region $C_s^2\ \varepsilon[0.08, 1.06]$, where $\varepsilon_{abs} \leq 0.08$. The absolute error function is plotted in Fig. 10 for a few sample values of C_s^2 so as to show this behaviour. An important point to emerge from this analysis is that the diffusion approximation is a better approximation for nonexponential distributions than using an exponential approximation. Our results would lie in the linear 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, 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.

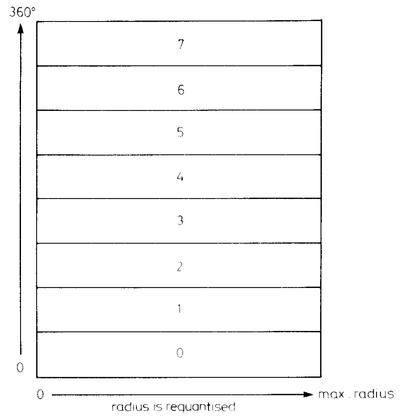


Fig. 10 *Quantisation of the target image for a parallel in-place computation*

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

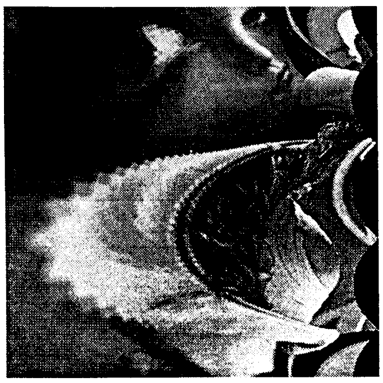
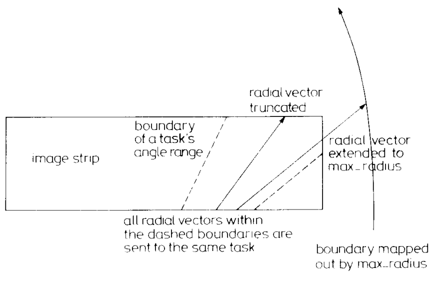


Fig. 11 *Polar transform, using an exscribed circle*



radial vector truncated
boundary of a task's angle range
image strip
radial vector extended to max_radius
all radial vectors within the dashed boundaries are sent to the same task
boundary mapped out by max_radius

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

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]

$$\zeta_0 = 1$$

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

$$\begin{aligned} \eta &= \exp(i\theta) - 1 \\ &= 2i \sin(\theta/2) \exp(i\theta/2) \\ &= -2 \sin^2(\theta/2) + i \sin(\theta) \end{aligned}$$

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.

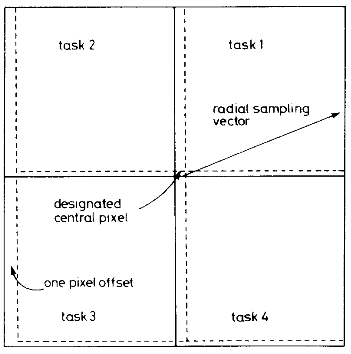


Fig. 13 *Coping with the origin*

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