MVA'98 IAPR Workshop on Machine Vision Applications, Nov. 17-19, 1998, Makuhari, Chiba, Japan

3—25

PARALLEL VISION COMPUTING ON A NETWORK OF WORKSTATION CLUSTERS

J. You[1] *A. Sattar*[1] *L. Vlacic*[2]

[1] School of Computing and Information Technology
[2] School of Microelectronic Engineering
Griffith University, Nathan Campus, Brisbane, QLD, Australia 4111

ABSTRACT

Vision computing involves the execution of a large number of operations on large sets of structured data. In this paper we demonstrate that such vision tasks can be implemented in parallel on a network of workstation clusters for fast processing. We introduce some techniques used in distributed systems and adopt a divide-and-conquer policy to schedule the complex vision tasks for parallelism. The vision-related algorithms for mask convolution, feature extraction, discrete Fourier transform and image matching are implemented in parallel using PVM(parallel virtual machine). In addition, a hierarchical object recognition system is described to conclude that a general distributed system can be applied to parallel vision computing at a low cost.

Keywords and phrases: Vision computing, parallel implementation, parallel virtual machine(PVM), feature extraction, image matching, object recognition.

1. INTRODUCTION

Vision computing is closely allied with three fields: image processing, pattern classification and scene analysis. It involves many types of computing, ranging from two-dimensional correlation and convolution, to image transformation, geometric computing and graph analysis. In general vision computing can be summed as the execution of a large number of operations on large sets of structured data. One of the basic data structure in vision algorithm is the two dimensional array. In many cases a vision task is performed by the application of a set of operations to all the elements of the image array. When such a task is implemented in sequential, it is computationally intensive. Thus the conventional sequential computers are too slow to perform complex visual tasks in real time. Obviously, the speed-up in computing can be achieved by performing different operations concurrently on each element of the array. Such a parallel method lies in the fact that several operators can be applied to the image simultaneously (MIMD style); or the same operator can be proceeded at different parts of the image at the same time (SIMD style)[1].

In contrast to the conventional parallel solutions which relied on specialised parallel machine, we explored the potential of distributed systems for parallelism. It is noted that the advanced technology of computer network has made very high-speed network available (Gigabit/sec.). Thus a distributed computer system can be utilised to replace the specialised machines for the implementation of vision tasks and dynamic load balancing for resource allocation. A divide-and-conquer policy can be adopted for such a scheme, where a complex task is divided into a number of sub-tasks and those sub-tasks are later reorganised into clusters according to granularity before being mapped on computers for simultaneous implementation. Due to the nature of parallel processing used in this scheme, the operation speed can be increased and real-time issues are considered by coordinating the priorities of tasks.

This paper is organised as follows: Section 2 briefly introduce a parallel virtual machine (PVM) as a general-purpose message-passing architecture which is available on most of the existing computer system. Some basic vision-related algorithms and their parallel implementation are highlighted in Section 3. Section 4 summarises a parallel object recognition system on a network of workstation clusters. Finally the conclusion is presented in Section 5.

2. A PARALLEL VIRTUAL MACHINE (PVM)

PVM is viewed as a general and flexible parallel computing environment which enables concurrent executions on loosely coupled networks of processing elements and supports a message-passing model of synchronization[8]. Thus it provides a low-cost homogeneous multi-computing environment by using existing workstation resources for parallelism. PVM can be implemented on different architectures such as single CPU systems, vector machines and multiprocessors, where the connection between these elements may be through different networks. The initiation and termination of processes across the network for a particular application are made via a library of standard interface routines.

It should be also emphasized that the relationships between different processes which are in execution for an application are arbitrary and any process may communicate and/or synchronize with others. Such arbitrary control and dependency structures characterize the application program under PVM with the configuration capacity for efficient computation.

3. PARALLEL VISION COMPUTING USING PARALLEL VIRTUAL MACHINE (PVM)

Many vision algorithms can benefit from parallel processing. A divide and conquer approach to vision computing can be easily implemented in a networking environment. An image is partitioned into subimages and distributed over nodes in the network. The local processing results such as two-dimensional matrix operation, feature detection and transformation for each subimage are merged over the entire image. Figure 1 shows the partitioning and processing for parallel component labeling, where the boundary connectivity of neighboring subimages is represented as a graph. Hence, the original image represented in one large two-dimensional matrix (256×256, or 512×512, or 1024×1024) is broken into a number of small matrices (subimages). The success of the parallel implementation using PVM on a cluster of workstations depends on the effective communication pattern between workstations. In the work reported in this paper, the master/slave parallel programming paradigm is adopted. Each workstation represents a node in the network. One node is selected as the master, and is responsible for spawning slave nodes that process subimages.

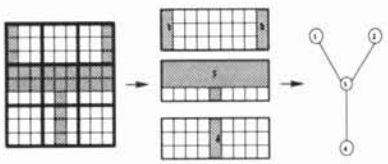


Figure 1. Divide images for parallel labeling

3.1. Mask Convolution

The mask considered is 5×5 size and the operation is performed locally on the image. Therefore, the image can be divided into sub-images with reasonable sizes and the convolution operation can be performed on each sub-images in parallel to speed up the process. In general, the two-dimensional convolution of the image I(i,j) and mask A(i,j) with size 2a+1 by 2a+1 is given by the relation

$$F(i,j) = A(i,j) * I(i,j)$$
$$= \sum_{k=-a}^{a} \sum_{l=-a}^{a} A(k,l) I(i+k, j+l)$$

for i=0,1, ..., M-1 and j=0,1, ..., N-1, where the size of the image is $M \times N$.

On the average the sequential execution time for each convolution with 5×5 mask on 256×256 image implemented on Classic SPARC workstation is about 5.8 sec. while the parallel computation with 4 processes increases the processing speed by 15% to 4.96 sec. for each convolution. The higher speed is expected when more processes are invoked for parallel implementation.

3.2. Discrete Fourier Transform

Fourier transform is widely used in image processing for image enhancement, feature extraction, texture analysis, and image compression. The following is the mathematical representation of the Discrete Fourier Transform for 1-D sampled function $f(x)$:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) exp[-j2\pi ux/N]$$

for $u = 1, 1, 2, ..., N-1$ and

$$f(x) = \sum_{u=0}^{N-1} F(u) exp[j2\pi ux/N]$$

for $x = 0, 1, 2, ..., N-1$

It is noted that the above summations can be represented as nested loop structures and can be easily implemented in parallel using PVM on a network of workstation clusters. In the work reported here, a number of slave processors are dedicated to summation and pass the result to the master. Table 1 lists the speed improvement in comparison to the traditional sequential execution.

Table 1: The speed improvement: parallel vs. sequential $((T_s - T_p)/T_s)$

N	1 Slave $(T_s - T_p)/T_s$	2 Slaves $(T_s - T_p)/T_s$	3 Slaves $(T_s - T_p)/T_s$	4 Slaves $(T_s - T_p)/T_s$
500	-0.660	-0.053	-0.067	0.310
1000	-0.445	0.406	0.504	0.438
1500	-0.262	0.314	0.517	0.617
2000	-0.191	0.402	0.600	0.667
2500	-0.142	0.594	0.591	0.710
3000	-0.115	0.443	0.605	0.703
3500	-0.092	0.439	0.634	0.709
4000	-0.076	0.459	0.625	0.716
4500	-0.059	0.454	0.641	0.719
5000	-0.026	0.470	0.632	0.735
5500	-0.052	0.434	0.638	0.726

3.3. Image Feature Extraction - The Detection of Interesting Points

Most matching algorithms are based on binary images to identify the interested object(s). Therefore, the original image, either greyscale or color images, should be converted into a binary image. Traditional methods that convert an original image into a binary image rely on edge detection. Despite edge detection has been successfully used for many years mostly due to its simplicity, it has some problems which prevent it from being applied on a real-time image matching scheme, such as:

- sensitive to noise in the image,
- feature points may not be well distributed,
- a large number of feature points with redundancy.

This has prompted the research[4] to use interesting point detectors rather than edge detectors to extract feature points from a given image for matching. A number of interesting point detectors have been developed, *e.g.* Plessey operator[5] and Moravec operator[6].

The detection of interesting points is based on the measure of how interesting a point is. *Interesting* here has its own special meanings depending on different applications. In order to reduce the number of points used for matching while still preserving the features of the original image, such points must be distinguishable from immediate neighbours, which excludes points sitting on the same edge. Moravec[4]

suggested that a point is considered interesting if it has local maximum of minimal sums of directional variances. For a local window ranging from 4×4 to 8×8, the directional variances can be expressed as

$$I_1 = \sum_{i,j} (I(i,j) - I(i,j+1))^2$$
$$I_2 = \sum_{i,j} (I(i,j) - I(i+1,j))^2$$
$$I_3 = \sum_{i,j} (I(i,j) - I(i+1,j+1))^2$$
$$I_4 = \sum_{i,j} (I(i,j) - I(i+1,j-1))^2$$

where (i,j) represents the elements in the window. The interestingness of a point is then given by

$$I(i,j) = \min(I_1, I_2, I_3, I_4).$$

Thus a point whose local maximum is over a pre-set threshold will be considered good as an interesting point, where the pre-set threshold can be chosen based on the image histogram. In the work reported here, the threshold is determined dynamically for optimal performance based on the interestingness histogram of the filtered image after Moravec operation. In addition, both data parallelism and functional parallelism was applied to the detection of interesting points, where the calculation of I_1, I_2, I_3 and I_4 within a local window was performed simultaneously while the whole image was divided into sub-regions for the same operation. The performance of both sequential and parallel detection of interesting points based on Moravec operator is compared as below. Table 2 lists the performance evaluation in terms of the average execution time on different images with various sizes ranging from 128×128 to 512×512.

Table 2: The comparison of execution time

image size	execution time (in sequential)	execution time (in parallel)
128×128	1.53 sec.	1.15 sec.
256×256	7.24 sec.	5.23 sec.
512×512	26.34 sec.	19.11 sec.

Our test data shows that the processing speed is increased by parallel detecting interesting points to remove redundant edge pixels without any specific architecture requirement for parallelism. It is clear that the speedup will be more effective when the image size is larger, the algorithm is more complicated and more processors are used for the parallel implementation.

3.4. Image Matching

Comparing a template image to a larger target image usually requires that the the matching algorithm assigns a value as to how good the match is for every possible overlay position of the template image over the target image. The position with the best matching measurement such as the lowest distance is then regarded as being the position of the best match. Although this method can be highly accurate, depending on the adopted matching algorithm, it is computationally expensive to search for every possible combination position of the template window within the target image. The searching for the best matching can be operated on the subimages simultaneously. Table 3 shows the effectiveness of such a parallel matching scheme, where the average value within the template matrix is used to search for the closest indices over the original matrix.

Table 3: Slaves and Block Size vs. Execution Time

Matrix Size	Block Size	3 Slaves Time(sec.)	5 Slaves Time(Sec.)	8 Slaves Time(Sec.)	No. of Iterations
128	4	38.67	39.22	40.97	1024
128	8	8.80	10.23	11.64	256
128	16	2.52	3.35	4.07	64
128	32	1.42	1.50	1.96	16
256	4	139.36	142.71	151.84	4096
256	8	36.56	39.30	42.17	1024
256	16	12.19	12.62	16.10	256
256	32	6.27	7.05	7.85	64

4. PARALLEL OBJECT RECOGNITION BY HIERARCHICAL IMAGE MATCHING

The hierarchical guided matching scheme was first introduced by Borgefors[2] in order to reduce the computation cost required to match two images. We extended it by using interesting points rather than edge points in a similar fashion, i.e., an interesting point pyramid is created and the matching starts from the lowest resolution and the results of this match guides the search on the possible area of the higher resolutions. We further extend it by using the Hausdorff distance as a measure of similarity instead of Chamfer matching. The advantage of using Hausdorff distance in a matching process relies on the capability of searching for portions of images, which allows us to partition the target image into a number of subimages and simultaneously match the template image on these subimages.

Given two finite point sets $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, the Hausdorff distance D_H between these two sets is defined as

$$D_H = \max(d_{AB}, d_{BA})$$

where d_{AB} is the distance from set A to set B expressed as

$$d_{AB} = \max_{a_i \in A}(d_{a_i B})$$

while $d_{a_i B}$ is the distance from point a_i to set B given by

$$d_{a_i B} = \min_{b_j \in B}(d_{a_i b_j})$$

Obviously the Hausdorff distance D_H is the maximum of d_{AB} and d_{BA} which measures the degree of mismatch between two sets A and B.

In general, image data are derived from a raster device and represented by grid points as pixels. For a feature detected image, the characteristic function of the set A and B can be represented by a binary array $A[i,j]$ and $B[i,j]$ respectively, where the $(i,j)th$ entry in the array is non-zero for in the array is non-zero for the corresponding feature pixel in the given image. Therefore, distance array $D[i,j]$ and $D'[i,j]$ are used to specify for each pixel location (i,j) the distance to the nearest non-zero pixel of A or B respectively, where $D[i,j]$ denotes the distance transform of A and $D'[i,j]$ denotes the distance transform of B. Consequently, the Hausdorff distance as a function of translation can be determined by computing the pointwise maximum of all the translated D and D' array in the form of:

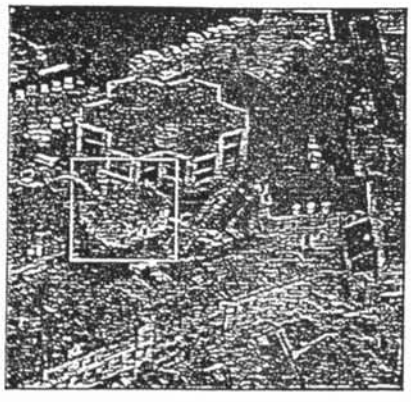
$$F[i,j] = \max(\max_a, \max_b)$$

where

$$\max_a = \max_a D[a_i - i, a_j - j]$$
$$\max_b = \max_b D'[b_i + i, b_j + j]$$

In order to avoid the blind searching for the best fit between the given patterns, a guided search strategy is essential to reduce computation burden. Our extension of the hierarchical image matching scheme (H.I.M.S) was based on a guided searching algorithm that searches first at the low level, coarse grained images, to the high level, fine grained images. To do this we needed to obtain a Hausdorff distance approximation for each possible window combination of the template and target image at the lowest resolution. Those that returned a Hausdorff distance approximation equal to the lowest Hausdorff distance for those images were investigated at the higher resolution.

To evaluate the performance gain, a system was implemented on a group of networked workstations (8 DECstations), where PVM (Parallel Virtual Machine) was used to provide a parallel execution environment. Based on the developed system, a number of experiments were carried out to measure the effectiveness of using the hierarchical approach in matching, and to measure the speedup of using parallel guided matching against using sequential matching. The results are shown in Table 4 and 5, respectively. Figure 2 shows an example of object identification and localisation using our hierarchical matching scheme. Figure 2(a) is a 300×300 target image with certain object to be identified and Figure 2(b) shows the matching result of our hierarchical scheme, which returns a match at position $(56, 142)$.



(a) Target image

(b) Template image

Figure 2: Object identification and localisation by matching

Table 4: Hierarchical image matching scheme

Pyramid levels	image 1	image 2
	(256 × 256)	(362 × 362)
1	239.66 sec.	544.27 sec.
2	21.36 sec.	38.12 sec.
3	1.9 sec.	4.32 sec.
4	1.32 sec.	2.18 sec.
5	1.28 sec.	2.14 sec.

Table 5: Matching a rotated template image

Matching method	image 1	image 2
	(256 × 256)	(362 × 362)
sequential	209.45 sec.	410.13 sec.
parallel	32.7 sec.	67 sec.

5. CONCLUSION

Parallelism provides more powerful computing ability for vision systems where simple operations on large set of data is required. The advanced technology of computer network has made very high-speed network available. Obviously such a distributed computer system has the potential to parallelize some vision tasks which used to rely on specialised machines. A parallel object matching system is implemented on a distributed system, which aims to support real-time image matching. The experiment results confirm the effectiveness of such an approach in speedup.

6. REFERENCES

[1] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles and H.C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching", *Proc. 5th Int. Joint Conf. Artificial Intelligence*, Cambridge, MA, pp. 659-663, 1977.

[2] G. Borgefors, "Hierarchical chamfer matching: a parametric edge matching algorithm", *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. PAMI-10, pp. 849-865, 1988.

[3] D.P. Huttenlocher, G.A. Klanderman and W.J. Rucklidge, "Comparing images using the Hausdorff distance", *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. PAMI-15, pp. 850-863, 1993.

[4] J. You, E. Pissaloux, J.L. Hellec and P. Bonnin, "A guided image matching approach using Hausdorff distance with interesting points detection", *Proc. of IEEE ICIP'94*, Austin, USA, Nov. 13-16, 1994, pp. 968-972.

[5] J.A. Noble, "Finding corners", *Image and Computing*, Vol. 6, No. 2, 1988.

[6] H.P. Moravec, "Towards automatic visual obstacle avoidance", *Proc. 5th Int. Joint Conf. Artificial Intelligence*, Cambridge, MA., pp. 584, 1977.

[7] R. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Vol. 2, Addision-Wesley, 1993.

[8] Al Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM: Parallel Virtual Machine, A User's Guide and Tutorial forNetwork Computing*, MIT Press, Cambridge, MA, 1994.