Parallelization of 3-D Range Image Segmentation on a SIMD Multiprocessor

Vipin Chaudhary and Sumit Roy
Parallel and Distributed Computing Laboratory
Wayne State University
Detroit, MI 48202

Bikash Sabata
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025-3493

Abstract

The paper presents a parallelized algorithm for segmentation of 3-D range images. The segmented image is useful as an input to higher level image processing tasks. The algorithm is implemented on a mesh connected multiprocessor machine, the MasPar series of massively parallel computers. A 2-dimensional hierarchical data distribution scheme is used to allocate the image pixels on the machine. The original sequential segmentation procedure is modified to take into account this data distribution. The performance of the new programs is compared to that of the original code, executed on an IBM RS6000 520H workstation. The speed-up is found to be about three for a MasPar consisting of an 128 x 128 processor array. This is due to the architectural constraints of the MasPar, in particular the implementation of its interprocessor communications. The utilization of processors on the MasPar is found to be as low as 56 % for this algorithm. Results of executions on machines of various sizes are also shown. Suggestions for better performance are discussed.

1 Introduction

In most computer vision systems, the acquired data needs to be represented using symbolic descriptions. These descriptors can then be used for higher level vision tasks. The symbolic description is obtained by partitioning or segmenting the data into a set of primitives, depending on the nature of the input data used at the higher level. Sabata [8] developed a modular framework to the problem of segmenting a 3D range image of a scene containing multiple arbitrarily shaped objects into a set of homogeneous surface patches. The first module of the segmentation procedure groups the image pixels in terms of local properties derived from the input data. This leads to a preliminary oversegmentation that is later refined by the final vision task. In the second module, the oversegmented results of the low level segmentation are merged into homogeneous regions based on surface descriptions. Since the description of the surface is not an invariant with respect to the representation, the output of the merging module is a function of the surface representation. The second module uses the result of the low level segmentation, the raw input data as well as the surface representation of the higher level task to arrive at a final partition of the scheme.

The initial data driven phase uses a pyramidal clustering algorithm based on seven properties: the surface normal at very point and the six projections along each plane from the two possible directions. This corresponds to *illuminating* the scene from three orthogonal directions. Zeroth- and first-order properties are exploited in this phase to reduce the sensitivity to noise that higher order surface properties tend to exhibit[5]. This also makes the process less dependent on smoothing algorithms and avoids the distortions that they tend to introduce[3].

Another consequence of noise in data is that most segmentation schemes require the specification of various tolerance thresholds, which depend on the quality and type of input data. The *Pyramidal Clustering Algorithm* used here has the useful property that it does not depend on such arbitrary or empirically determined values.

This paper reports out parallel implementation of the segmentation and clustering phases on the MasPar series of massively parallel machines. The result of the parallel low level segmentation can then be sent to a merge module for higher level vision tasks. The rest of the paper is organized as follows: Section 2 describes the architecture of the MasPar. Section 3 explains the algorithm used and describes the changes made for the MasPar. Section 4 shows the output generated by the algorithm on the MasPar and also tabulates execution timings for machines of various sizes. Section 5 briefly introduces some of the possible applications of the algorithm. Finally, we draw some conclusions in Section 6.

2 MasPar Architecture

The MasPar computer system works in a Single Instruction Multiple Data fashion, with scalability in terms of number of processing elements, system memory as well as communication bandwidth[2]. It can be divided into a number of components:

Array Control Unit: This unit can control the actual Processor Element core or independently execute programs. Processor Elements are controlled by broadcasting instructions from the ACU. Limited memory is available on the ACU.

Processor Element Array: The PE array is the computational core of the machine. Each element has on-chip registers, off-chip data memory, an integer and floating point ALU and a communication interface. Logic is provided for conditional execution based on internal data. This allows us to selectively disable or enable processors.

Communication Interfaces: The MasPar provides three major interfaces for communication:

- An *X Network* for nearest neighbor communication in 8 directions and wrap-around. It assumes a 2-D mesh topology.
- A circuit switched global router network permitting random simultaneous communication using a three level hierarchical crossbar switch.
- Two global buses to communicate with the ACU. The processor instructions are broadcast on one bus and status information is collected by the ACU on the other.

UNIX Subsystem: This provides a user interface as well as all job management and low speed network access. This is implemented in a separate Front End.

I/O Subsystem: This consists of a separate high speed disk array and I/O RAM and allows overlapped communication and I/O operations.

3 Segmentation Algorithms

The initial 3D image consists of a data file with views of the scene from the X, Y and Z direction. The data points represent the coordinates of the objects surface in 3D space. This data is first sent to a *preprocessing stage* and then to the main *pyramidal clustering stage.*

3.1 Preprocessing

The preprocessing stage generates six intensity images from the input 3D data and a seventh image that is the surface normal image of the scene. Prior to this several steps are performed.

Labeling. Each point in the input data is initially labeled as BACKGROUND, NOISE or OBJECT, based on the range values for the particular pixel and its North, West, East and South neighbors. If the pixel has 0 valued X,Y and Z components, it is set to BACKGROUND. A pixel surrounded by neighbors with 0 X and Y components is set to NOISE, everything else is an OBJECT. Also isolated BACKGROUND pixels are set to the average of their neighbors.

Normals computation. The procedure is an extension to the process of using separable operators to compute normals to graph surfaces [1]. Assuming that the position vectors of the points in a scene are given as a function of the (u, v) coordinates of a parametric space, define

$$\mathbf{X} = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$$

where u and v are some parameters. Then, if $\mathbf{X}_u$ and $\mathbf{X}_v$ are the partial derivatives of $\mathbf{X}$ with respect to u and v respectively, then:

$$\mathbf{X}_u = \begin{pmatrix} x_u(u,v) \\ y_u(u,v) \\ z_u(u,v) \end{pmatrix} \text{ and } \mathbf{X}_v = \begin{pmatrix} x_v(u,v) \\ y_v(u,v) \\ z_v(u,v) \end{pmatrix}$$

These derivatives are computed using a separable convolution operator. The partial derivatives of a function $f(u, v)$ are given by:

$$f_u = D_u * f \text{ and } f_v = D_v * f$$

where D_u and D_v are given by

$$[D_u] = \mathbf{d}_0\mathbf{d}_1^T \text{ and } [D_v] = \mathbf{d}_1\mathbf{d}_0^T$$

where

$$\mathbf{d}_0 = \frac{1}{7}[\,1\,1\,1\,1\,1\,1\,1\,]$$

$$\mathbf{d}_1 = \frac{1}{28}[\,-3\ -2\ -1\ 0\ 1\ 2\ 3\,]^T$$

Thus each component of $\mathbf{X}_u$ and $\mathbf{X}_v$ can be computed and the normals result from the partial derivatives as

$$\hat{\mathbf{n}} = \frac{\mathbf{X}_u \times \mathbf{X}_v}{|\mathbf{X}_u \times \mathbf{X}_v|}$$

Intensity image generation. In addition to computing the surface normals, intensity images are generated by assuming a light source from six different coordinate axes. This produces six different images which are individually segmented. This is combined with the segmentation of the surface normals to provide the desired over segmentation for the higher level merging routines.

For the MasPar implementation we used a blockwise data distribution to minimize communication. Parallel I/O is used to read in the data file directly into the PE's memory. It should be noted that the ACU does not have enough space to store the program and a 256×256 3D image. The data points on the border of each data block require information from the neighboring processor and incur some communication overhead. A partial data replication scheme could have avoided this cost, but would have made the parallel I/O more complicated and expensive.

3.2 Pyramidal Clustering

The segmentation is accomplished using a *Pyramidal Clustering Algorithm.* The desirable property of this algorithm is a relative independence from arbitrarily determined thresholds. The approach is a type of a "fuzzy" algorithm, where the groupings depend on the statistics of the input data.

The pyramid consists of a regular array of nodes, decreasing in size from the bottom up [4, 6]. Each node at a higher layer is initialized to the average of a neighborhood in the next lower layer. Thus the image resolution decreases from the bottom up due to this averaging process. The base level stores the original input image, each pixel corresponding to a node. An h level pyramid would have $2^h \times 2^h$ nodes at the bottom level. The nodes at level l are linked to some nodes at level $l + 1$ as well as to some at level $l - 1$. Nodes at level $l + 1$ and linked to a node $\mathbf{n}$ at level l are referred to as the $fathers$ of $\mathbf{n}$ and similarly the nodes at level $l - 1$ linked to $\mathbf{n}$ are called its *sons.*

The clustering algorithm itself is divided into three stages, an *initialization* stage, a *node linking* stage and a *tree generation* stage. The pyramidal data structure is initialized in the first stage, ie. the source data is read into the base level and the higher level nodes are computed as averages of some regions of the preceding layer. The next stage is the iterative *node linking* stage, which was considered for the timing measurements. The clusters or segments are generated in this stage and assigned unique labels using the *tree generation* stage.

The algorithms were adopted for the MasPar by executing the code on all processors simultaneously. The data distribution was maintained from the preprocessing stage. The iterations at higher levels of the pyramid tend to use only a part of the processor array since the image size decreases by 75 % at each level.

3.3 Postprocessing

To compare the results from various machines some postprocessing is also carried out. The seven segmented images are integrated together to arrive at one resultant over-segmented image. Maximal segmentation is achieved by converting the region R_1 in the first partition and region R_2 in the second partition into three resultant regions, namely $R_1 \cap R_2, R_1 - (R_1 \cap R_2)$ and $R_2 - (R_1 \cap R_2)$. Additionally small regions, consisting only of a few pixels are merged into larger neighbors using a median filtering operation.

The median filter was converted to MasPar parallel code, however the region integration routines use an inherently sequential algorithm and the original programs are used.

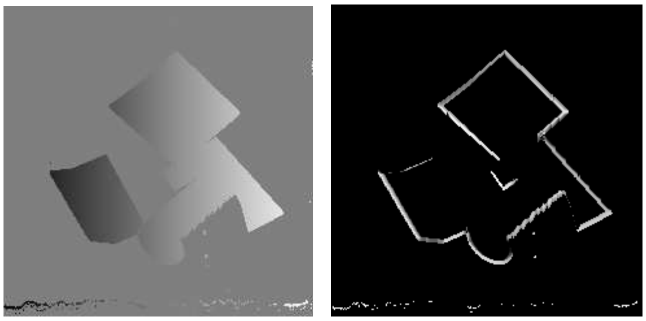
4 Results and Analysis

This section will show some of the results obtained by the algorithm on a particular Range Image.

4.1 Processing Results

The image input file consists of a 512-byte header, which contains information about the size of the image, followed by the views from three orthogonal directions, ie. the X-view, the Y-view and the Z-view, Fig.1 (a). Each data element is a 32-bit floating point number. The preprocessing routine produces a vector file of the surface normals of the image. The the i-component of the normal is as shown in Fig.1 (b). The edges of the objects can be clearly seen in the figure.

The segmented and merged images, are shown for a IBM RS 6000 520H in Fig.2 (a), a 32×32 MasPar in



(a) X-view of original image

(b) i-component of normals

Figure 1: Original and Preprocessed Images

Fig.2 (b), a 64×64 MasPar in Fig.2 (c), and a 128×128 MasPar in Fig.2 (d).

It can be seen that there are only minor differences in the results, which are mostly due to quantization errors and the fuzzy logic nature of the iterations. At present the algorithm is portable across all machines with a square processor array. This is due to limitations in the *tree generation* part and the integration programs.

4.2 Performance Analysis

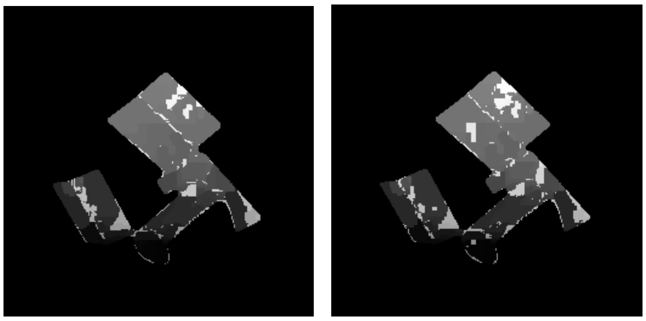
While profiling the original code, it was found that the node linking stage is the slowest part in the segmentation scheme, hence detailed timings and speed up estimatess were obtained for this part.

Calculating the speedup was not straight forward, since the number of processors required could reduce by 75 % at higher levels of the pyramid. Also processors having only background pixels do not take part in any computation. Since a typical scene, even with multiple objects, consists of 60 % background pixels, one uses a correspondingly lower number of processors. Thus the machine utilization does not remain constant over the execution of the linking stage.

To account for this we first measured the time taken for each level and added the values that use the same number of processors, ignoring the problem of background pixels. The average processor utilization can then be obtained as:

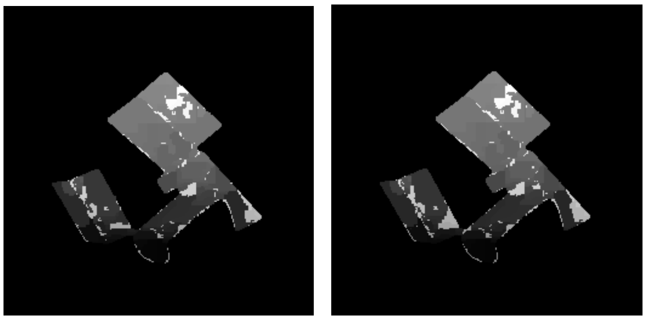
$$\text{Average used} = \frac{\sum (\text{Processors at this level} \times \text{Time for this level})}{\text{Total time}}$$

The timing data is presented in Table 1. The 32×32 and the 64×64 machines were simulated on the 128×128 machine by using the *mpswopt* command. This command allows us to reduce the number of processors in the active set for the duration of the program. The timings have been added across the seven runs ie. the six projection runs and the normal run.



(a) IBM RS 6000

(b) 32×32 MasPar



(c) 64×64 MasPar

(d) 128×128 MasPar

Figure 2: Segmented Result on Different Machines

The machine utilization is given by:

$$\text{Machine Utilization} = \frac{\text{Average used}}{\text{Machine size}}$$

Table 2 shows the speedup achieved with respect to the IBM RS 6000. It is not possible to compare the results with a single node MasPar since the data set would not fit into the local memory of a single Processing Element. The execution time for the *node-linking* sub-routine in the *pyramidal clustering* programs was evaluated on the IBM workstation with and without optimization. Optimization on the MasPar produced very minor differences in execution times.

It may be noted that the relative speed up between machines of increasing sizes is ≈ 2, though the machine size increases by a factor of 4. The cost performance

Machine	Total Time		Speed Up	
	Opt. On	Opt. Off	Opt. On	Opt. Off
IBM RS 6000 520H	236.21	595.44	-	-
32×32 MasPar	388.95	-	1.53	0.61
64×64 MasPar	137.17	-	4.34	1.72
128×128 MasPar	81.90	-	7.27	2.88

Table 2: Speedup with respect to IBM RS 6000 520H

analysis becomes very complex, since the number of processors used at each level is dependent on the input data set. Besides, the cost of the MasPar system does not scale linearly with the number of processors.

5 Applications

An important application of the above algorithm is the estimation of motion from sequences of range images. Most early approaches used only local features, like corners and edges, which tend to be sensitive to noise and quantization errors. The use of global features like surfaces would make such schemes more robust. These surfaces would be generated by the method described in [9].

Numerous other vision applications also depend on the coarse description provided by the segmented image, including tasks like object recognition and navigation. However for practical applications one would expect real time processing of this data, and the speed up obtained on the MasPar seem to show that this is a distinct possibility.

6 Conclusion

This paper presented an implementation of a sequential image segmentation algorithm on a massively parallel SIMD machine. The proposed implementation is not suitable for the MasPar since the data set size decreases with time and there is considerable communication overhead. However it was still possible to obtain some speed up on the machine, and a major bottleneck appears to be communication overhead. This could be reduced by using a different data layout for the various phases of the program. A scheme for maximum utilization of the available processing power was suggested in [7]. Thus further speed–up may be obtained at the cost of implementation complexity.

Acknowledgements

We would like to thank the MasPar Computer Corporation and the Northeast Parallel Architectures Center for providing support and access to their machines during the 1993 MasPar Challenge Contest.

References

[1] P. J. Besl,"Surface in Range Image Understanding",*Springer Verlag, New York*,1989.

[2] T. Blank, "The MasPar MP-1 Architecture",*Proceedings, COMPCON Spring 90*,1990, pp. 20 – 24.

[3] M. Brady, J. Ponce, A. Yullie, and H. Asada, "Describing Surfaces",*Computer Vision Graphics Image Processing* 32,1985, pp. 1–28.

[4] P. J. Burt, T. H. Hong, and A. Rosenfeld,"Segmentation and Estimation of Image Region Properties through Cooperative Hierarchical Computation",*IEEE Trans. Systems Man Cybernetics.* SMC -11, No. 12, 1981, pp. 802 –809.

[5] P. J. Flynn and A. K. Jain, "On Reliable Curvature Estimation",*Proceedings, Computer Vision and Pattern Recognition*,1989, pp. 110–116.

[6] W. J. Grosky and R. Jain, "A Pyramid-based Approach to Segmentation applied to Region Matching",*IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8, No. 5, 1986, pp. 380 — 386.

[7] N. Ramesh and V. Chaudhary "Complexity Analysis of Range Image Segmentation on MasPar MP–1", *Proceedings, 35th Midwest Symposium on Circuits and Systems*,1992.

[8] B. Sabata, F. Arman, and J. K. Aggarwal, "Segmentation of 3D Range Images Using Pyramidal Data Structures", *CVGIP: IMAGE UNDERSTANDING*,Vol. 57, pp. 373–387, 1993.

[9] B. Sabata and J. K. Aggarwal, "Estimation of Motion from a Pair of Range Images: A Review", *CVGIP: IMAGE UNDERSTANDING*, Vol. 54, pp. 309–324, 1991.

Machine Size	Time on $n \times n$ procs. (s)					Total time (s)	Average used	Machine Utilization
	128×128	64×64	32×32	16×16	8×8			
32×32	-	-	376.67	6.34	5.99	388.95	996.83	97 %
64×64	-	113.09	11.84	6.27	6.02	137.17	3479.85	85 %
128×128	41.50	15.25	12.64	6.52	5.91	81.90	9247.75	56%

Table 1: Execution time on the MasPar