

Scalable Geometric Hashing on MasPar Machines *

Ashfaq A. Khokhar, Viktor K. Prasanna, and Hyoung J. Kim
Department of EE-Systems, University of Southern California
Email: {ashfaq + prasanna + khj}@halcyon.usc.edu

Abstract

In this paper, we present scalable data parallel algorithms for geometric hashing. We perform implementations of the proposed algorithms on MasPar MP-1/MP-2. In earlier parallel implementations, the number of processors employed is independent of the size of the scene but depends on the size of the model database which is usually very large. We design new parallel algorithms and map them onto MP-1/MP-2. These techniques significantly improve upon the number of processors employed while achieving superior time performance. Our implementations run on a P processor machine, such that $1 \leq P \leq S$, where S is the number of feature points in the scene. Our results show that a probe¹ of the recognition phase for a scene consisting of 1024 feature points takes less than 50 msec on a 1K processor MP-1/MP-2.

1 Background

Object recognition, a high level vision task, is a key step in an integrated vision system. In object recognition using geometric hashing [4], given a set of models and their features points, for each model, all possible pairs of the feature points are designated as a *basis set*. The coordinates of the features points of a model are computed relative to each member of its basis set. These coordinates are then used as indices into a hash table. The records in the hash table comprise of (*model, basis*) pairs. In the recognition phase, an arbitrary pair of feature points in the scene is chosen as a basis and the coordinates of the feature points in the scene are computed. The new coordinates are used to hash into the hash table and the corresponding entries of the hashed bin are accessed. Votes are accumulated for the (*model, basis*) pairs stored in the hashed locations. The pair winning the maximum number of votes is chosen as a candidate for matching.

There have been two prior efforts in parallelizing the geometric hashing algorithm [1, 5]. Both implementations have been performed on SIMD hyper-

cube based machines. One of the major problems in both the implementations is the requirement of large number of processors. These implementations used $O(Mn^3)$ processors, *i.e.*, the number of bins in the hash table, where M is the number of models in the database and n is the number of feature points in each model. Also, the implementation by Bourdon and Medioni [1] suffers due to inefficiency of the routing algorithm. This inefficiency limits the scope of their implementation to a small model database. In [5], Rigoutsos and Hummel suggest to use radix sort to implement histogramming, a technique used in their implementation to count the number of votes for each (*model, basis*) pair. The use of radix sort in histogramming is advantageous only if the number of levels in the histogram is much less than the number of data points. In case of geometric hashing, this is not true.

2 Our Results

The size of the model database (the number of hash bins) is $O(Mn^3)$. We use P processors such that $1 \leq P \leq S$, where S is the number of feature points in a scene. Each Processing Element (PE) in the array is assumed to have $O(\frac{Mn^3}{P})$ memory. In our results, we exploit the fact that the number of votes cast in an iteration of the recognition phase is bounded by S , the number of feature points in the scene. Therefore, no more than S locations of the hash table are accessed during the execution of the recognition algorithm. This allows us to reduce the number of processors employed to at most S . Due to space limitations, we are omitting the details of the asymptotic analysis. The following theorem summarizes the asymptotic results. Additional details can be found in [2, 3].

Theorem 1 *Given a mesh array of S processors, one probe of the recognition phase can be processed in $O(\sqrt{S})$ time on a scene consisting of S feature points.*

We have experimented with three partitioning algorithms. These algorithms differ with respect to partitioning and mapping of the hash table onto the processor array. Various strategies are employed to take into account practical considerations, such as available memory in each PE, processor speed, and I/O speed of the machines. In Algorithm A, we assume that

*This research was supported in part by NSF under grant IRI-9145810 and in part by DARPA and AFOSR contracts F-49260-89-C-0126 and F-49620-90-C-0078.

¹The execution of the recognition phase corresponding to one basis pair is termed as *probe*.

Number of Probes	Machine Size	Encoding Scene Points	Hash Bin Access	Voting	Computing Local Max of Voter	Computing Global Max of Votes	Total Time
1	1K	2.51	18.53	24.10	3.36	0.055	48.76
2	2K	2.51	23.16	30.60	8.16	0.165	64.59
4	4K	2.51	40.36	49.86	8.16	0.314	101.20
8	8K	2.51	54.62	49.72	8.16	0.608	115.62

Table 1: Execution times (in *msec*) of Algorithm C on a scene consisting of 1024 feature points with concurrent processing of multiple probes on various sizes of MP-1. Average bin size is 8.

Methods	# of Models (16 points/model)	Machine ^a Size/Type	# of Scene Points	Total Time
Our Method (A)	1024	1K/MP-1	1024	53.37 <i>msec</i>
Our Method (B)	1024	1K/MP-1	1024	38.89 <i>msec</i>
Our Method (A)	1024	1K/MP-2	1024	41.35 <i>msec</i>
Our Method (B)	1024	1K/MP-2	1024	26.24 <i>msec</i>
Our Method (B)	1024	256/CM-5	1024	5.74 <i>msec</i>
Our Method (B)	1024	1K/MP-1	200	49.50 <i>msec</i>
Our Method (B)	1024	256-CM-5	200	4.50 <i>msec</i>
Hummel et. al. [5]	1024	8K/CM-2	200	800 <i>msec</i>
Medioni et. al. [1]	x	8K/CM-2	x	2.0-3.0 <i>sec</i>

Table 2: Comparison with previous implementations.

^aEach processor in CM-5, CM-2, and MP-1/MP-2 operates at 32MHz, 7MHz and 12.5MHz respectively.

each processor is assigned $\frac{Mn^3}{P}$ distinct hash table locations. In Algorithm B, each sub-array of processors is assigned a complete copy of the hash table. Each processor in a sub-array of size, s^2 , where $1 \leq s \leq \sqrt{P}$, has $\frac{Mn^3}{s^2}$ distinct entries of the hash table. The case of large number of processors is considered next. Algorithm C performs concurrent processing of multiple probes of the the recognition phase. The array is divided into disjoint sets of S processors. Each set of PEs processes a probe using a basis (a different basis for each set). Our serial implementation shows that one probe of the recognition phase takes about 13.4 seconds on a SUN SPARC2 operating at 25MHz and 32 Mbytes of on board RAM.

3 Concluding Remarks

We have developed scalable data parallel algorithms for geometric hashing. Based on these algorithms, we have obtained fast parallel implementations. The implementations achieve much superior time performance than those known in the literature. We have also performed implementations on CM-5 leading to less than 10 *msec* execution times per probe. Additional details of these implementations can be found in [3].

References

- [1] O. Bourdon and G. Medioni, "Object Recognition using Geometric Hashing on the Connection Machine," In *ICPR*, pp. 596-600, 1988.
- [2] A. Khokhar. *Scalable Data Parallel Algorithms and Implementations for Object Recognition*, PhD Thesis, Dept. of EE-Systems, USC, Jan. 1993.
- [3] A. Khokhar, H-J. Kim, V. Prasanna, and C. Wang, "Scalable Data Parallel Implementations of Object Recognition using Geometric Hashing," Tech. Report, Dept. of EE Systems, USC, Feb. 1993.
- [4] Y. Lamdan and H. J. Wolfson, "Geometric Hashing: A General and Efficient Model based Recognition Scheme," In *ICCV*, pp. 218-249, Dec., 1988.
- [5] I. Rogoutsos and R. Hummel, "Massively Parallel Model Matching: Geometric Hashing on the Connection Machine," *IEEE Computer*, pp. 33-42, Feb., 1992.