Parallel Geometric Hashing Algorithm for Protein Tertiary Structure

Khalid Jaber[†], **Rosni Abdullah**[††] **and Fazilah Othman**[†††]

School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia

Summary

The amount of protein structure has become very huge nowadays, which has led to the need for improving algorithms to cope with this exponential increase. At the same time, processors have become more powerful and affordable with low price. By making use of these advantages, we can construct a powerful parallel system that will overcome the problem of exponential increase of data. This paper presents a good Parallel Geometric Hashing Algorithm which will parallelize the sequential Geometric Hashing Algorithm on a cluster. This will lead to powerful, accurate and fast results especially in search and matching. This parallelized algorithm performs coordinate transformations on the feature points of an object to obtain an abstract model of that object. The matching objects are identified by computing correlations between the feature of the query and the model. Our system was run on the Stealth cluster and it used the multiple instruction multiple data (MIMD) paradigm which was applied on protein tertiary structure data. The system allows rapid recognition of unknown protein structure which consists of many residues. We achieved improvement for parallel time speedup compared to the sequential time speedup.

Key words:
Geometric Hashing Algorithm, Parallel Geometric Hashing Algorithm, model-based recognition, Protein Tertiary Structure.

1. Introduction

Geometric Hashing Algorithm (GHA) was first introduced in 1986 by Professor Jacob Schwartz [1]. The application of Geometric Hashing Algorithm in model-based vision object recognition was first introduced 1988 by (Lamdan, Y., Wolfson, H.J) and the technique was also extended to the recognition of arbitrary rigid 3-D objects from single 2D images (Lamdan, Y., Wolfson, H.J, 1988) [1].

GHA has been implemented in more than one area, such as robotic machines, database image queries, structural molecule comparison, security (fingerprint and face recognition) and medical applications (detection of irregularities on images (MRI, X-ray)).

In structural bioinformatics, Geometric Hashing Algorithm is a popular and reliable method for protein function determination, protein searching, structure classification and structure alignment [2]. In this paper, we will apply Parallel Geometric Hashing Algorithm (PGHA) on the protein structure, because the protein molecule may have many residues, and each residue may be large. Therefore, the matching procedure will take a long time to process on huge bits of data. This will pose a major challenge to process. The proposed research is to focus on the parallelization aspect, instead of implementing the algorithm on a single machine. One of the main advantages of the Geometric Hashing technique is that it is inherently parallelizable. Its parallel nature manifests itself both in the pre-processing (work off-line) and the recognition (work on-line) phases of the algorithm [3].

The choice of implementing Parallel Geometric Hashing Algorithm on a cluster was based on the fact that workstation clusters are cheap and readily available. Clusters can also be easily expanded and it has a low maintenance cost. Moreover, development tools on workstations are more mature.

Contribution

This paper makes three principal contributions. First, the Parallel design for Geometric Hashing Algorithm is presented. Second, it is applied on protein tertiary structure and finally, Parallel Geometric Hashing Algorithm is implemented on the Stealth cluster using C programming language and Message Passing Interface (MPI) library.

In this paper, section two, gives a brief explanation about GHA, section three describes the proposed framework. And section four, deals with the implementation results and Performance. Section five, conclude the paper.

2. Geometric Hashing Algorithm in brief

In Geometric Hashing, a collection from coordinate or pair of points for each model is called basis set. The basis set is used to define the reference frame, and the coordinates of the features points of each model are computed in the frame. Therefore, a reference frame system is the set of coordinates defined by a particular basis.

These coordinates are then put as indices into a hash table. In other words, the reference frame enables us to construct a hashing table.

Hash table is a data structure that connects the linked lists. The main aim of using the hash table is to store and retrieve data.

The Geometric Hashing Algorithm consists of two phases; preprocessing and recognition [4] and [5].

The pre-processing phase can be summarized as below [7]:

(i) Extract information features from each model.
(ii) Identify a reference frame.
(iii) Calculate all other points in the model based on the basis pair (reference frame).
(iv) Save the new points in the hash table.
(v) Repeat for each model reference frame.

The pre-processing phase can be done off-line and can be done only once throughout the overall procedure.

The recognition phase can be summarized as below [7]:

(i) Extract information feature from model for a given query.
(ii) Identify a reference frame for query model.
(iii) Calculate all other points in the model based on the basis pair (reference frame).
(iv) Save the new points in the hash table.
(v) Calculate the distance between the models.
(vi) Repeat for each model reference frame.

After applying the recognition phase, the minimum distance will be the maximum of the closest matched model. The recognition phase can be done in real-time.

3. Proposed framework

Our purpose in this section is to design a parallel system of Geometric Hashing Algorithm. We will discuss the design from many angles such as PGHA mechanism, hardware and software, number of processors, input and output.

But before we discuss this, we will explain the Root Mean Square Deviation (RMSD).To compare between two proteins tertiary structures we use the RMSD as the distance measure.

3.1 Root Mean Square Deviation (RMSD)

The root mean square deviation (RMSD) is the measure of the average distance between the backbones of proteins [6]. We calculate RMSD to show that the least value matches the query. The formula for root mean square deviation is defined as follows [6]:

$$RMSD = \sqrt{\frac{\sum_i d_i^2}{n}} \quad (1)$$

$$d_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (2)$$

Where:

d_i = distance between N pairs of equivalent atoms.
n = number of atoms.

3.2 PGHA Mechanism

Parallel Geometric Hashing Algorithm (PGHA) is implemented using Task Farming (or Master/Slave) model. The master works the preprocessing phase in which the number of files (PDB files) to be put in the hash table is read, which is our project database. Then it extracts information from the PDB file and saves it into the hash table. After that, it chooses a reference frame and calculates new coordinates for each atom based on the reference frame which we choose. It carries out the same steps for query protein.

The master node's main responsibility is to distribute the hash table and query among slaves. The program runs on two, three and five nodes. Here is the summary for master's responsibilities:

(i) Carry out the pre-processing phase.
(ii) Decompose problem into small tasks.
(ii) Distribute the Hash table and reference frame query among the slaves.
(iii) Gather results from slaves and announce who is the protein tertiary structure winner.

Reference frames in the query are assigned to slave nodes based on a number of reference frames (i.e. if we have 20 reference frames, in three nodes (1 master, 2 slaves), master sends hash table and 10 reference frames to each slave node). Each slave node compares its reference frames with the hash table reference frames using RMSD, then saves score and sends it to the master node. Here is the summary for the slave's responsibilities:

(i) Get task from master (hash table, query reference frame).
(ii) Process the task (Recognition phase).
(iii) Send results to master.

When the master node receives the results from the slaves, it calculates the final results and announces which protein structure is the winner. Here is the summary for of design methodology:

Hardware	Stealth Cluster
Software	C compiler with MPI

	under Sun Solaris 9
Parallel paradigm	(MIMD)
Number of processors	1,2,3,5
Taxonomy of parallel architecture	Data Partitioning
Application of parallel methods	Recognition Phase
Input	Unknown Protein Tertiary Structure
Database	PDB file
Output	known Protein Tertiary Structure

Table 1: Summary of Design Methodology

4. Implementation and performance

We have implemented the algorithm described in section 3 in C programming language using MPI library. The experiments were done on a Stealth cluster with 2 Sun Sparc-III 900 MHz and 2 GB of RAM. The three-dimensional coordinate data was taken from the Protein Data Bank (PDB) which was in turn, based on backbone (N-Cα-C) [2].

Experiments were run on a group of proteins that are known to be related. We tested the system on five families which are Lyase which has 23 proteins, Ligase which has 837 proteins, Isomerase which has 1157 proteins, Hydrolase which has 10183 proteins and Seryl which has 43 proteins. We have chosen 10 proteins from each family with 60 atoms, and from those 60 atoms, we produced 20 reference frames. All these members are contained in our database and our query protein is 1SERB [8]. Table 2 shows the results.

Family Name	**PDB ID**	**RMSD**
Lyase	**1F1O**	**536.49**
	1F9G	**185.55**
	1IDJ	**175.99**
	1IDK	**208.83**
	1SC9	**183.63**
	1SCI	**183.70**
	1SCK	**183.11**
	1SCQ	**183.37**
	1YB6	**183.58**
	1YB7	**212.11**
Ligase	**1A0I**	**168.12**
	1A35	**185.14**
	1A4I	**176.49**
	1A6R	**169.47**
	1A8H	**275.99**
	1A9X	**169.36**
	1A48	**199.67**
	1A82	**191.47**
	1ADE	**162.61**
	1ADI	**169.32**
Isomerase	**1A0C**	**178.51**
	1A0D	**173.71**
	1A0E	**167.33**
	1A2J	**181.90**
	1A2L	**187.44**
	1A2M	**295.71**
	1A31	**187.62**
	1A33	**178.05**
	1A35	**185.14**
	1A36	**185.29**
Hydrolase	**102L**	**178.44**
	103L	**178.29**
	104L	**177.28**
	107L	**180.70**
	108L	**180.97**
	109L	**181.03**
	110L	**597.53**
	111L	**177.96**
	112L	**180.62**
	113L	**181.22**
Seryl	**1SERA**	**30.25**
	1SERB (Query)	**0.00**
	1SESA	**24.75**
	1SESB	**24.75**
	1SRYA	**29.84**
	1SRYB	**26.82**
	1SETA	**27.18**
	1SETB	**22.31**
	1WLE	**174.72**
	2CF9	**172.38**

Table 2: Results Using 1SERB as Query

Table 2 shows that the least score is equal to zero which means the distance between each residue from query and hash table equals zero, this shows that the query exactly matches one of them in the hash table. The query is selected from one of the protein families on purposely to confirm accuracy.

When we look at the table we note that seryl proteins scores are close to each other but when compared to other proteins families' scores, we find a big difference. Thus has happened because query is part of the seryl family.

To accept matching we divided the results into three groups from 0-100 (Angstrom), 100-170 (A), 170-190 (A), 190-300 (A), 300-600 (A). If the result is between 0-100 we assume as that it is very similar to the target and thus, we accept as matching.

Performance Measures

Parallel system results are compared to a sequential system that was implemented using C on Stealth cluster. It took about 2.81 *seconds* to perform a query. We implemented a parallel system on two, three and five processors. The execution time was about 1.11 *seconds* using five processing nodes. Therefore, our solution can achieve a speedup of time about 2.53 *seconds*, overhead about 2.73 and efficiency about 0.5.

A comparison with sequential system for two, three, five processing nodes is shown in table 3.

Number of processor	Time(sec)	Overhead	Speedup	Efficiency
One processor (sequential)	2.819461			
Two	2.723072	2.626683	1.035397154	0.517698577
Three	1.493565	1.661234	1.887739067	0.629246356
Five	1.111529	2.738184	2.536560899	0.50731218

Table 3: Performance Measures

5. Conclusion

This research paper presents the design of Parallel Geometric Hashing Algorithm applied on protein tertiary structure to recognize the unknown protein tertiary structure.

This system is designed for Multiple Instruction Multiple Data (MIMD) and implemented on Stealth cluster using C programming language and MPI library. In our design we applied data partitioning, while the master sends a query and the hash table which contains known protein to slaves. The recognition phase is also parallelized. Our design avoids memory congestion since each processing node has the entire set of score boxes and has the same hash table.

This system allows rapid recognition of unknown protein tertiary structure which consists of many residues. With 50 proteins, we achieved execution time of about 1.11 *seconds* on five nodes processing, compared to 2.81 *seconds* for the sequential execution time.

Acknowledgments

This research has been funded by Universiti Sains Malaysia short-term Research Grant for "Tertiary Protein Structure Matching Based on Geometrical Features" [304/PKOMP/636038].

References

[1] Isidore R., MASSIVELY PARALLEL BAYESIAN OBJECT RECOGNITION, PhD thesis, New York University, August 1992.

[2] R. A. Fazilah Othman, Rosalina A.Salam, "Geometric Hashing Algorithm for Protein Tertiary Structure Matching: A Preliminary Study," 1st International Symposium on Bio-Inspired Computing (BIC'05), Puteri Pan Pacific Hotel, Johor Bahru, Malaysia, 5th-7th September 2005.

[3] I. Rigoustos and R. Hummel, "Massively parallel model matching: geometric hashing on the Connection Machine", Computer, vol. 25, pp. 33-42, 1992.

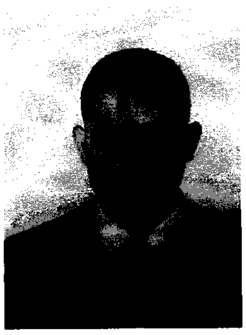
[4] C. Yongwha, S. Choi, and V. K. Prasanna, "Parallel object recognition on an FPGA-based configurable computing platform", 1997, pp. 143-152.

[5] A. Khokhar, V. Prasanna, and W. Cho-Li, "Scalable data parallel implementations of object-recognition on Connection Machine CM-5", 1994, pp. 130-139.

[6] R. Laakso, "Protein structure analysis", 2005, www.lce.hut.fi/teaching/S-114.500/Protein_Structure1.pdf

[7] O. Bourdon and G. Medioni, "Object recognition using geometric hashing on the Connection Machine", 1990, pp. 596-600 vol.2.

[8] S. Canzar and J. Remy, "Shape Distributions and Protein Similarity",http://www.ti.ethz.ch/as/people/remy/preprints/shape_2006.pdf, 2006.



Khalid Jaber received his Bachelors degree in Computer Science from Al-Isra University, Amman, Jordan in 2005 and Masters Degree in Computer Science from Universiti Sains Malaysia, Penang, Malaysia in 2007. He is currently a research officer at the Parallel and Distributed Processing Research Group and a PhD candidate as well under the supervision of Associate Professor Dr. Rosni Abdullah at Universiti Sains Malaysia.

Rosni Abdullah received her Bachelors Degree in Computer Science and Applied Mathematics and Masters Degree in Computer Science from Western Michigan University, Kalamazoo, Michigan, U.S.A. in 1984 and 1986 respectively. She joined the School of Computer Sciences at Universiti Sains Malaysia in 1987 as a lecturer. She received an award from USM in 1993 to pursue her PhD at Loughborough University in United Kingdom in the area Parallel Algorithms. She was promoted to Associate Professor in 2000. She has held several administrative positions such as First Year Coordinator, Programme Chairman and Deputy Dean for Postgraduate Studies and Research. She is currently the Dean of the School of Computer Sciences and also Head of the Parallel and Distributed

Processing Research Group which focus on grid computing and bioinformatics research. Her current research work is in the area of parallel algorithms for bioinformatics applications.

Fazilah Othman is currently a research officer at the Parallel and Distributed Processing Research Group and a PhD candidate as well under the supervision of Associate Professor Dr Rosni Abdullah at Universiti Sains Malaysia, Penang, Malaysia. She received her Bachelor and Masters Degree in Computer Science from the same university in 2002 and 2004, respectively. Her PhD research is mainly focused on parallel processing for matching algorithms in Bioinformatics. She is now going through a research attachment at Software Department, Universitat Politècnica de Catalunya, Barcelona, Spain.