Parallel Alignment of a Large Number of Range Images

Takeshi Oishi
Institute of Industrial Science, The University of Tokyo

Ryusuke Sagawa
Osaka University

Atsushi Nakazawa
Osaka University

Ryo Kurazume
Kyushu University

Katsushi Ikeuchi
Institute of Industrial Science, The University of Tokyo

Abstract

This paper describes a method for parallel alignment of multiple range images. There are problems of computational time and memory space in aligning a large number of range images simultaneously. We developed a parallel method to address the problems. Searching for corresponding points between two range images is time-consuming and requires considerable memory space when performed independently. However, this process can be preformed in parallel, with each corresponding pair of range images assigned to a node. Because the computation time is approximately proportional to the number of vertices, by assigning the pairs so that the number of vertices computed is equal on each node, the load on each node is effectively distributed. In order to reduce the amount of memory required on each node, a hypergraph that represents the correspondences of range images is created, and heuristic graph partitioning algorithms are applied to determine the optimal assignment of the pairs. Moreover, by rejecting redundant dependencies, it becomes possible to accelerate computation time and reduce the amount of memory required on each node. The method was tested on a 16-processor PC cluster, where it demonstrated high extendibility and improved performance.

1. Introduction

In recent years, the technology of modeling from reality using laser range sensors has been highly developed [1]. Using this technology has enabled the digital preservation of precious cultural heritage objects around the world [2, 3]. Laser range sensors can measure 3D surface geometry with accuracy within a millimeter for a short distance. For a long distance (50m to 100m), a laser range sensor that applies a time of flight can obtain surface geometry with a 1cm degree of accuracy.

While the performance of laser range sensors has been greatly improved, there are still some problems in creating a 3D model from a large number of range images. Because a laser range sensor can measure only the visible surface, it is necessary to take range images from many different directions. Once the scanning has been completed, all range images have to be aligned into a common coordinate system. If an object is small enough to be put on a turntable, it is easy to obtain the relative positions of the range images. But in cases where an object is a large statue, for example, it is difficult to record the accurate position and direction of the laser range sensor. Therefore, computation to obtain the relative positions of the range images is required.

Many methods of aligning range images have been proposed. These algorithms are based on the iterative closest point (ICP) proposed by Besl [8] and are adapted from the method proposed by Chen [9]. With ICP, corresponding points are searched for as the closest points between two range images, and a transformation matrix is computed so that the mean square error of the corresponding points is minimized. The computation is iterated until the mean square error falls below the threshold value. In Chen's method, the relative positions of range images are calculated so that the distance between vertices and the corresponding patches is minimized. In addition, there is a method to search for correspondences by projecting the points along with the ray direction [10].

When the number of range images is very large, a method that simultaneously aligns range images is required. The algorithms described above align two range images; when using these algorithms, error accumulation

increases as the number of range images increases. In such cases, a method that simultaneously aligns range images is useful. Neugebauer et al. proposed a simultaneous registration method that adopted projection search of correspondences and point-plane error metric [12]. (This is the fundamental algorithm used in this study.) Benjamaa et al. extended the method proposed by Bergevin et al. [13] and implemented a simultaneous alignment method while they accelerated the pair-wise alignment algorithm by using multi z-buffers [14].

Although various methods have been proposed, the problem for every method is the computation cost of correspondence search. If the number of vertices of two range images is equally assumed to be n by the original ICP, their complexity is $O(n^2)$ since correspondences are searched for in all vertices. In order to accelerate ICP, there are techniques [15] that use kd-trees and that narrow the search range by using data cache [16], as well as the parallel ICP algorithm proposed by Langis et al., which is implemented on a PC cluster [17].

Despite the many alignment algorithms, it is difficult to align the large number of range images that our activities involve. The computation time for the pairs of range images increases, and it is necessary to read all range images into memory when such algorithms are used (Parallel ICP [17] does not consider the amount of memory used). It is thought, moreover, that the amount of data will increase along with the development of measurement technology. Therefore, we need a method in which the calculation time is short, the amount of memory used is small, and the extendibility is high.

Thus, we propose a parallel simultaneous alignment method that is implemented on a PC cluster because this method is cheap and highly extendible. In Section 2, the fundamental alignment algorithm is described. In Section 3, we present the algorithm of parallel computation. Sections 4 and 5 contain the evaluations of this algorithm and the alignment results of a large number of range images, respectively. Our conclusions are presented in Section 6.

2. Simultaneous Alignment Algorithm

In this section, the outline of the fundamental alignment algorithm is explained. We assume that all range images have been converted to mesh models. The algorithm is applied in the following steps:

1. To compute, for all pairs of partial meshes,
   (a) to search all correspondence of vertices
   (b) to evaluate error terms of all correspondence pairs
2. To compute transformation matrices of all pairs for immunizing all errors
3. To iterate steps 1 and 2 until the termination condition is satisfied

Our algorithm employs points and planes to evaluate relative distance as the Chen and Medioni method [9]. The corresponding pairs are searched along the line of sight. Here, the line of sight is defined as the optical axis of a range sensor. Let us denote one mesh as the base mesh and its corresponding mesh as the target mesh. An extension of the line of sight, from a vertex of the base mesh, crosses a triangle patch of the target mesh and creates the intersecting point. In order to eliminate false correspondences, if the distance between the vertex and the corresponding point is larger than a certain threshold value, the correspondence is removed. This correspondence search is computed for every pairs of mesh models.

The error measure between corresponding points is the cosine distance between the point and the plane. Let the vertex of the base mesh and the corresponding crossing point in the target mesh be $\vec{x}$ and $\vec{y}$, respectively. The error measure between the pairs is written as

$$\vec{n}\cdot(\vec{y}-\vec{x}) \tag{1}$$

where $\vec{n}$ is the normal of $\vec{x}$ defined around the vertex.

The transformation matrices of the base and target mesh models are computed so that this error measure is minimized. The error evaluation function is rewritten as

$$\varepsilon = \boldsymbol{R}_{\mathrm{B}}\vec{n}\cdot\{(\boldsymbol{R}_{\mathrm{T}}\vec{y}+\vec{t}_{\mathrm{T}})-(\boldsymbol{R}_{\mathrm{B}}\vec{x}+\vec{t}_{\mathrm{B}})\} \tag{2}$$

Here, the rotation matrix and the translation vector of the base and target mesh are $R_M, R_S, \vec{t}_M, \vec{t}_S$ respectively. The distance between the base and the target mesh is expressed as

$$\bar{\varepsilon} = \min_{\mathbf{R},\vec{t}}\sum_{i,j,k}\left(\boldsymbol{R}_i\vec{n}_{ik}\cdot\{(\boldsymbol{R}_j\vec{y}_{ijk}+\vec{t}_j)-(\boldsymbol{R}_i\vec{x}_{ik}+\vec{t}_i)\}\right)^2 \tag{3}$$

If it is assumed that the angles of rotation are minute, the rotation matrix $\boldsymbol{R}$ is written as

$$\boldsymbol{R} = \begin{pmatrix} 1 & -c_3 & c_2 \\ c_3 & 1 & -c_1 \\ -c_2 & c_1 & 1 \end{pmatrix} \tag{4}$$

The translation vector is expressed as

$$\vec{t} = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}^{\mathrm{T}} \tag{5}$$

After some algebraic manipulations [12], (3) is rewritten as

$$\bar{\varepsilon} = \min_{\vec{\delta}}\sum_{i\neq j,k}\left\|\vec{A}_{ijk}\cdot\vec{\delta}-s_{ijk}\right\|^2 \tag{6}$$

$$s_{ijk} = \vec{n}_{ik}\cdot(\vec{x}_{ik}-\vec{y}_{ijk}) \tag{7}$$

$$\vec{A}_{ijk} = \left\{\left(\underbrace{0\ldots0}_{6i\times1}\ \vec{C}_{ijk}^{\mathrm{T}}\ \underbrace{0\ldots0}_{6(l-i-1)\times1}\right)+\left(\underbrace{0\ldots0}_{6j\times1}-\vec{C}_{ijk}^{\mathrm{T}}\ \underbrace{0\ldots0}_{6(l-j-1)\times1}\right)\right\}^{\mathrm{T}} \tag{8}$$

$$\vec{C}_{ijk} = \begin{pmatrix} \vec{n}_{ik}\times\vec{y}_{ijk} \\ -\vec{n}_{ik} \end{pmatrix} \tag{9}$$

$$\vec{\delta} = \left(\vec{\boldsymbol{m}}_0\ldots\vec{\boldsymbol{m}}_{n-1}\right)^{\mathrm{T}} \tag{10}$$

$$\vec{\boldsymbol{m}}_i = \begin{pmatrix} c_{1i} & c_{2i} & c_{3i} & t_{xi} & t_{yi} & t_{zi} \end{pmatrix}^{\mathrm{T}} \tag{11}$$

where the number of mesh models is n. By (6) $\vec{\delta}$ is written as

$$\left(\sum_{i,j,k}\vec{A}^{\mathrm{T}}{}_{ijk}\vec{A}_{ijk}\right)\vec{\delta}=\sum_{i,j,k}\vec{A}^{\mathrm{T}}{}_{ijk}s_{ijk} \qquad (12)$$

3. Parallel alignment based on a PC cluster

Among the simultaneous alignment operations described in Chapter 2, 1(a) correspondence search and 1(b) error evaluation require a large amount of computational time. They also require data space to read in data of all vertices. On the other hand, these two operations can be conducted independently in each pair of partial mesh models. Computation of transformation in step 2 does not require much computational time or memory space. Thus, we designed correspondence search and error evaluation in step 1 to be conducted in slave PCs in a PC cluster, and computation of transformation in step 2 to be conducted in a master PC.

3.1. Graph simplification

We remove redundant or weak data dependency relations of partial mesh models for the sake of efficiency in parallel computation. Figure 1 shows overlapping data-dependency relations. Each node in the graph represents one mesh model, and each arc represents an overlapping dependency relation among mesh models. The left graph shows the original state in which all the mesh models overlap each other. If we conduct alignment of one mesh as is, we would have to read into a PC's memory all the remaining mesh models. By removing some of redundant overlapping dependencies, we can transform the original graph into a simpler one as shown in the right figure. By using this simpler relational graph, we only need adjacent data with respect to a vertex for alignment of a vertex, and we can reduce the necessary memory space.

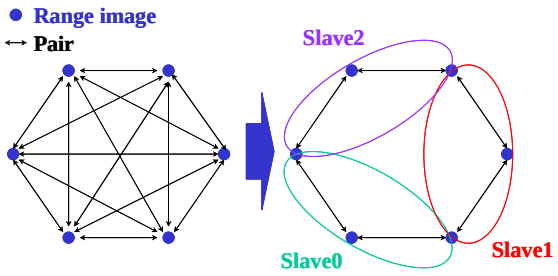


Figure 1. Data dependency relations

We will remove the dependency relation between the two mesh models if any of the mesh pairs does not satisfy any one of the following four conditions:

1. **The bounding-boxes of two mesh models overlap each other.**
   A sufficient overlapped region exists between two mesh models, provided that initial positions of two meshes are accurately estimated.

2. **The angle θ between ray directions of two mesh models is less than a threshold value.**
   Two observation directions of the meshes are relatively near. This condition also reduces the possibility of false correspondences between front- and backside meshes, by setting the threshold, as θ=90º. We could use a more accurately estimated value for this threshold, but since this value is used as a constraint to reduce the possibility described above, we use this θ=90º for the sake of safety and simplicity.

3. **The overlapping area of two meshes is larger than a threshold value.**
   Overlapping area is expressed as the ratio of the number of vertices included in one mesh model and the number of corresponding points between two meshes. Corresponding points are searched for a few vertices selected randomly. We used 10% of the vertices for this search. A pair whose overlapping area is less than threshold value will be removed as weak data dependency. We set the threshold value as 0.03 to 0.05. Since the computation of overlapping areas can be performed independently and sequentially for each pair, the computations are performed easily in parallel without the problem of memory usage.

4. **Two range images are adjacent to each other.**
   This condition removes non-adjacent relations sequentially. For example, as shown in Figure 2, if the length from I_0 to I_3 is larger than the length from I_1 to I_3 ($l_{01} < l_{03}$), the arc between I_0 and I_3 is removed. Here, the distance is evaluated from the center of a mesh model.

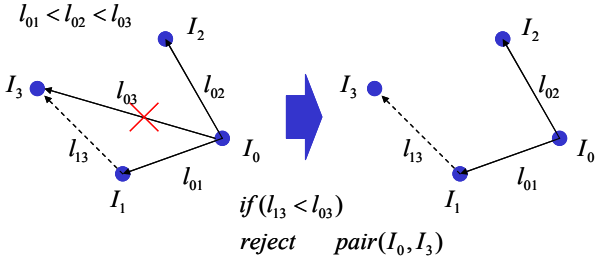


Figure 2. Non-adjacency relation

3.2. Parallelization by graph partitioning algorithms

The problem of load balancing with a minimum amount of required memory is an NP-hard problem [18]. It is difficult to obtain an optimal solution in a reasonable time. Alternatively, we employ an approximation method to solve this problem by applying heuristic graph-partitioning algorithms.

3.2.1 Pair-node hyper-graph

First, we define the pair-node hyper-graph. The left image of Figure 3 shows a graph that expresses the relations of partial meshes I_n. The graph is converted to the hyper-graph in which each node expresses pairs $P_{i,j}$ of two partial meshes i and j, and networks represent meshes, as shown in the right figure of Figure 3. We refer to it as a "pair-node hyper-graph."

The weight of the network $W^{\text{net}}{}_i$ is defined as the number of vertices v_i in the partial mesh, i; the weight $W^{\text{node}}{}_{i,j}$ of the node is defined as the sum of the number of vertices v_i and v_j.

$$W^{net}{}_i = v_i \tag{13}$$

$$W^{node}{}_{i,j} = v_i + v_j \tag{14}$$

A pair-node hyper-graph is partitioned so that the sum of the node weights in each subset is roughly equal for computational load balance, and summation of all the net-weight in each subset is minimized for efficiency of memory usage.

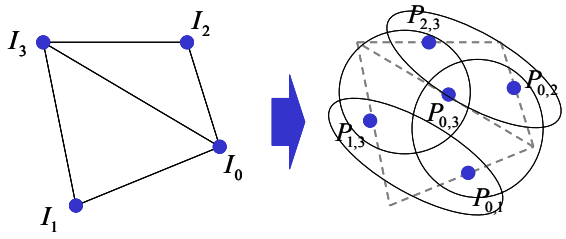


Figure 3. Pair node hyper-graph

It is necessary to consider both node weights and net weights in optimization, even though they are related to each other, and using them seems to be redundant. Reducing the computational load requires each sub-group to have equal values in the node-weights. On the other hand, even when a hyper-graph is portioned equally in terms of node-weight, depending on the method, each sub-group has different memory usage. Let us consider the example, shown in Figure 3, to divide the hyper-graph into two sub-graphs. For the sake of simplicity, we assume that all node-weights and net-weights are the same in all the nodes and all the networks. When the hyper-graph is divided into two groups, $\{P_{0,2}, P_{1,3}, P_{2,3}\}$ and $\{P_{0,1}, P_{0,3}\}$, the node balance is achieved in two sub-graphs. The first sub-graph needs to load in all the data $\{I_0, I_1, I_2, I_3\}$. The maximum value in sums of net-weights is four units. When the hyper-graph is divided into two groups, $\{P_{0,2}, P_{0,3}, P_{2,3}\}$ and $\{P_{1,3}, P_{0,1}\}$, each sub-group needs only to load in three data sets. The maximum value in the sum of net-weights is three units. In these two cases, both portioning methods have roughly equal load balance in terms of node-weights, but have different memory usage. When we divide the graph by considering only memory usage, it is not guaranteed that each sub-graph has equal load balance. Thus, we will consider both node-weights and net-weights in the optimization procedure.

3.2.2 Initial partitioning

The pair-node hyper-graph is initially partitioned so that the sum of the node-weights in each subset is roughly equal. Spectral bisection methods [19, 20] that minimize the edge-cut by using second eigenvector are widely available, but it is difficult to apply the method to our problem. Intelligent graph growth algorithm [21] can obtain a fairly optimal solution in a small computation time. However, this method tends to be trapped in a poor partitioning [22]. We used the random seeded breadth first search method for initial partitioning. Since the sum of net-weight included in each subset is greatly influenced by the selection of the seed, we created initial partitions for multiple seeds and adopted the partition in which the sum of net-weight included is minimized. In order to obtain k-way partitions, the recursive bisection method is used. After log$k$ phases, the hyper-graph is partitioned into k sub-graphs [23].

3.2.3 Refinement of the partition

The partitioned graphs are refined so that the sum of net-weights included in each subset graph is minimized. We improved the KLFM algorithm, which is an iterative refinement algorithm [24, 25]. The algorithm moves a node from one partition to another so that the operation causes the greatest improvement in the cut-size. While the original KLFM algorithm moves a node at one iteration, our method moves a net at one iteration. That is, all nodes connected to the net are moved at the same time. For k-way refinement, the subset graph of which the sum of net-weight is maximum weight is computed with all other subsets. The refinement process is reiterated until there is no more improvement.

The net gain is computed for all nets along the boundary of two subset graphs. Now, we consider the kth net at the boundary between the subset graphs, G_i and G_j. In the case the net $N_{(i,j),k}$ is moved to G_i, the gain $g_{i,j,k}$ is expressed using two values, $D^{int}{}_{i,j,k}$, the variation of the sum of net weight of G_i and $D^{ext}{}_{i,j,k}$, the variation of the sum of net-weight of G_j as

$$g_{i,j,k} = D^{ext}{}_{i,j,k} - D^{\text{int}}{}_{i,j,k}. \tag{15}$$

On the other hand, in the case where $N_{(i,j),k}$ is moved to G_j, the gain $g_{j,i,k}$ is expressed in the similar way as

$$g_{j,i,k} = D^{ext}{}_{j,i,k} - D^{\text{int}}{}_{j,i,k}. \tag{16}$$

The two lists, L_i, L_j, consisting of all gains of the all nets at the boundary, are created. The list with the larger sum of the total node-weight (computational time) is selected for consideration of the movement, and the components, candidate nets in the list, are processed one by one in descending order of the gain. At each movement of one net, all nets and nodes concerned with the net are updated, and the moved net is locked in order to avoid thrashing. The sum of the net-weight (memory usage) and the moved net's ID are also recorded at each movement. After all nets are moved, the minimum value of the sum of the net-weight (memory usage) is compared with the

value at the starting stage. If the minimum value is smaller than that of the starting state, the corresponding movement-sequence is performed, and the next iteration begins. If not, the refinement process is terminated. See Figure 4 for the flow chart of the refinement process.

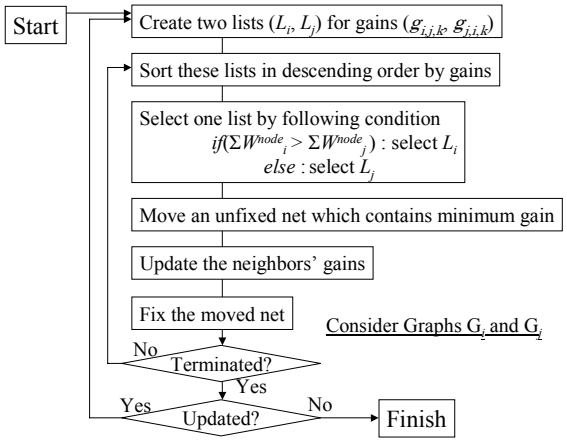


Figure 4. Flowchart of refinement process

3.3. Implementation

We implemented our method as a master/slave system. The procedures of the computation is are as follows

Algorithm Procedure of Parallel Alignment

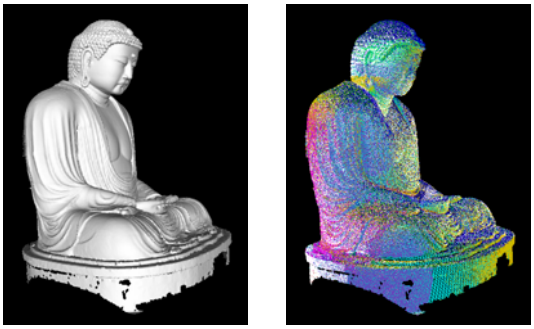
```
/* Check correspondence of all pairs of the partial meshes
*/
  Create-Pair-Table();
  /* Create the lists of the files for each processor */
  Create-File-Lists():
  while(error > threshold){
      /* Slave Process*/
      for(i = 0; i < nmeshes; ++i)
        for(j = 0; j < nmeshes; ++j)
           Whether-i-and-j-overlap-each-other?{
              Correspondence-Search(i, j);
              Calculation-Each-Matrix(i, j);
           }
    /* Master Process */
    CalculationMatrix(all);
    /* Master & Slave process */
    UpdatePosition();
  }
```

The master program holds bounding-boxes and transformation matrices from initial position to current position of all partial meshes, checks all pairs, and creates the list of computations for each node. The pairs list for each slave is computed at the beginning of the entire iteration process based on the relational table using the algorithm described above. The slave programs receive the lists and read the required partial meshes into memory. Then, each slave computes the matrices $A^{T}_{ijk}A_{ijk}$ and $A^{T}_{ijk}s_{ijk}$ in (12) independently, and sends the matrices to the master program. The master program computes the transformation matrices of all range images from the matrices $A^{T}_{ijk}A_{ijk}$ and $A^{T}_{ijk}s_{ijk}$ received from the slave programs. The results are applied to all master/slave data. Each iteration process is continued until the error falls below a certain threshold value.

4. Performance evaluation

This method was implemented on a PC cluster that consisted of 8 PCs. Each PC had dual AthlonMP2400+ processors and 4Gbytes of memory, and was connected by 100Base-TX ethernet. The range images used for evaluation were 50 images created artificially from the complete 3D model of the Great Buddha of Kamakura. Figure 5 shows the original 3D model of the statue and the partial mesh models created artificially. These mesh models contain an average of 83,288 vertices and 158,376 patches.

In this section, our method is evaluated from the viewpoints of convergence and accuracy, computation time, and memory usage.



(a) Original model (b) Created mesh models

Figure 5. Partial mesh models for evaluations

4.1. Convergence and accuracy

Because our method rejects redundant dependencies, the influence of the rejection on convergence and accuracy has to be evaluated. In this case, the number of all pairs is 2,450, but it is reduced to 160 by the rejection process. We needed to verify whether accurate convergence is performed even when the number of pairs becomes very small. Virtually created mesh models have accurate positions of measured points, so convergence and accuracy can be evaluated by the distance between an accurately aligned mesh model and the target mesh model. The distance between two meshes is defined as an average of the Euclidean length of all vertices. Each mesh model added Gaussian noise along the line of sight at maximum length 10mm. All mesh models were moved at random in the maximum length of 100mm in the directions of x, y, and z, respectively, and rotated at random in the maximum angle of 0.05 radians to the x-axis, y-axis, and z-axis, respectively.

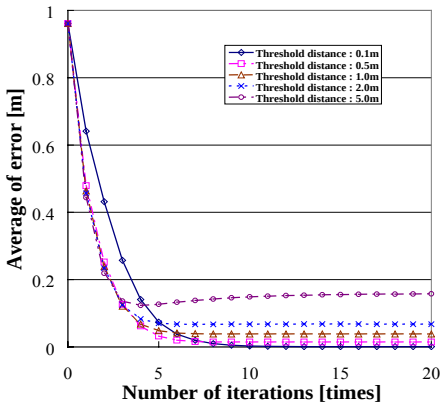


Figure 6. Convergence with original method

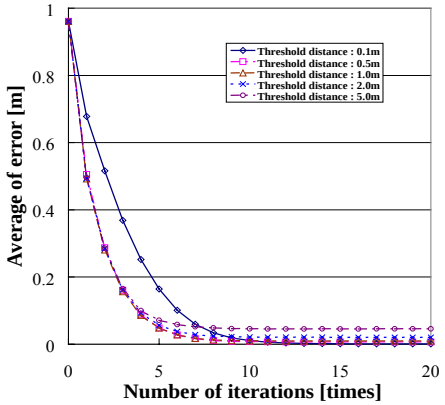


Figure 7. Convergence with our method

The results of the original method and our method are shown in Figures 6 and 7, respectively. The threshold distance for rejecting outliers while searching for correspondences is changed gradually. Although both the original method and our method do not converge at the correct positions when the threshold distance is 5m, our method converges at a better position than the position of the original method. Although the convergence speed of our method is slower than that of the original method, our method tends to converge at a better position than the position of the original method. It is thought that this is because the rejection of redundant pairs reduces false correspondence of mesh models. A feature of the alignment algorithm that we used is that it tends to be influenced by false correspondence and noise. Therefore, by rejecting redundant pairs, transformations are accurately estimated. When the threshold values are 0.1m and 0.5m, the error converges at approximately 0. So we see that accurate estimation is acquired by our method.

4.2. Computational efficiency

Here, the computation time is evaluated. Computation time is defined as the time taken for one iteration, and an average of time of all iterations is used for the evaluation. Figure 8 shows the time ratio with the number of processors. Computation time T_n is expressed as the ratio to the computation time with one processor T_0. This figure shows that the computation time is linearly improved as the number of processors increases. Moreover, our method improves computation time in a predictable way unlike the sequential method in which the mesh models are assigned in arbitrary selected order. The actual computation time with one processor averages 20560ms, and the computation time with 16 processors averages 1784ms. Thus, the computation time with 16 processors is approximately 11.5 times faster than that with 1 processor.

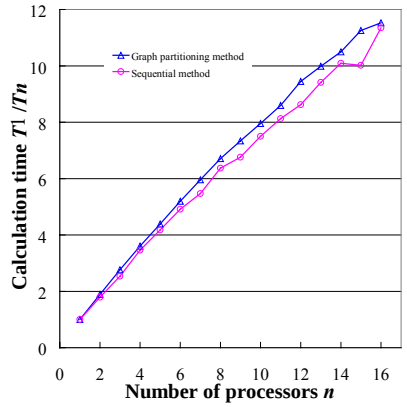


Figure 8. Computational Efficiency

4.3 Amount of required memory

Next, the evaluation of memory performance is shown. The amount of memory usage is shown in Figure 9 with the number of processors. Each value shows the ratio with the amount of memory used with a single processor. It appears that the amount of required memory decreases as the number of processors increases. Compared with the sequential method, the performance is highly improved by our method. An actual maximum size of required memory with a single processor is 269Mbytes and that with 16 processors is 48Mbytes. Therefore, our method could reduce the amount of memory used by approximately 17% for these mesh models.

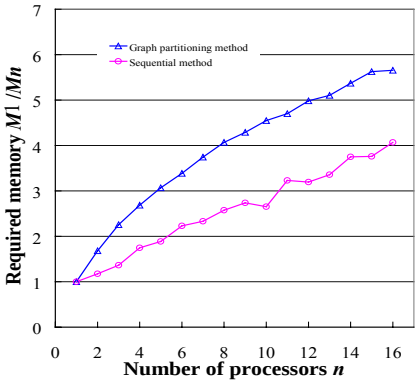


Figure 9. Required memory

Figure 10. Alignment result (Nara Buddha)

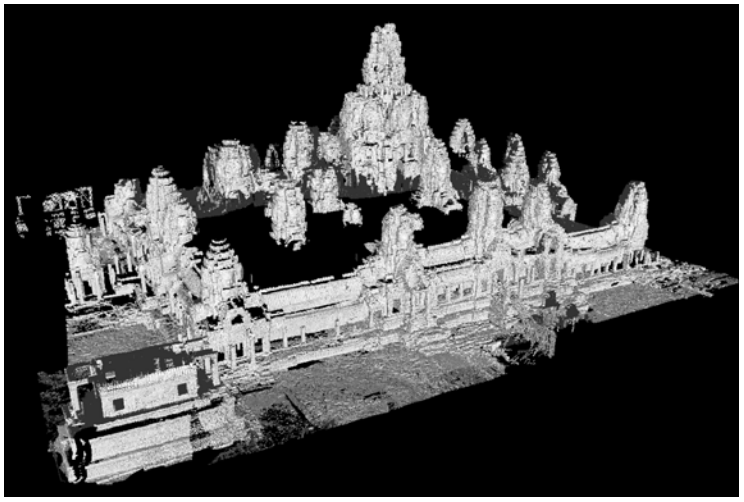


Figure 11. Alignment result (Bayon)

5. Experimental results

In this section, we will show the result of parallel alignment of a large number of range images that could not be aligned by one PC because of limitation of memory space. We used the following two sets of partial mesh models.

Model-1. 114 mesh models that measured the Great Buddha of Nara by Cyrax2400 [18]. These models contain an average of 327,470 vertices and 606,072 meshes.

Model-2. 210 mesh models that measured the Bayon Temple in Cambodia by Cyrax2500. These models contain an average of 433,785 vertices and 798,890 meshes.

Due to the limitation of memory space, the minimum numbers of processors required for aligning these data sets were 2 and 4 for Model-1 and Model-2, respectively.The alignment results computed by the minimum and maximum numbers of processors are shown in Table 1 and Table 2. These tables show the average computation time, the maximum amount of memory usage, and the minimum amount of memory usage.

Table 1. Total performance (Nara)

Processors	Ave. Time(s)	Max. Mem(MB)	Min. Mem(MB)
2	76	1287	1275
16	13.2	292	254

Table 2. Total performance (Bayon)

Processors	Ave. Time(s)	Max. Mem(MB)	Min. Mem(MB)
4	103.9	1608	1456
16	40.2	559	472

In the case of Model-1, the computation time with 16 processors is 5.75 times faster than that with 2 processors, and the amount of required memory is reduced 22.6%. For Model-2, the computation time with 16 processors is 2.58 times faster than that with 4 processors, and required memory is reduced 34.8%. As for the size of required memory, these results show an improvement better than that described in the previous section (30% for 2-16 and 47 % for 4-16). On the other hand, in the case of Model-2, although the number of processors is increased 4 times, the reason the computation time is not greatly improved (2.58 times) is that the time taken for calculation of the transformation matrix, which is not parallelized and is performed on the server program, is lengthened. An actual computation time taken by the server program is an average of 14 seconds, and is 35% of the total time taken for one iteration.

Figures 10 and 11 show the alignment results of the Great Buddha of Nara and the Bayon Temple in Cambodia, respectively. Alignment takes approximately 5 minutes for 20 iterations for Model-1 and approximately 15 minutes for Model-2.

6. Conclusion

In this paper, we have proposed the parallel method for simultaneous alignment of multiple range images. In considering time performance and memory performance, we parallelized the alignment algorithm. Then, we implemented this method on a PC cluster, and showed its validity by aligning a large number of range images simultaneously. Future work will deal with accelerating the computation of transformation matrices.

Acknowledgments

This work is supported by Ikeuchi CREST (Core Research for Evolutional Science and Technology) of JST (Japan

Science and Technology Corporation). The authors would like to thank the staffs of the Todaiji Temple in Nara, Japan, and the Koutokuin Temple in Kamakura, Japan. The Bayon Temple in Cambodia was digitized with the cooperation of the Japanese Government Team for Safeguarding Angkor (JSA).

References

[1] K. Ikeuchi and Y. Sato, *Modeling from Reality*, Kluwer Academic Press, 2001.

[2] K. Ikeuchi, "Modeling from Reality," *Proc. Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Quebec City, Canada. May 28- June 1, 2001

[3] M. Levoy et al. "The Digital Michelangelo Project, "*Proc. SIGGRAPH 2000*, pp.131-144.

[4] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Proc. ACM SIGGRAPH '94*, pp. 311-318, 1994.

[5] Curless, B., Levoy,M., A Volumetric Method for Building Complex Models from Range Images, *Proc. ACM SIGGRAPH '96*, pp. 303-312, 1996.

[6] M. Wheeler, Y. Sato, and K. Ikeuchi, Consensus surfaces for modeling 3D objects from multiple range images, *Proc. IEEE International Conference on Computer Vision*, pp.917-923, 1997.

[7] R. Sagawa, K. Nishino, K. Ikeuchi, "Robust and Adaptive Integration of Multiple Range Images with Photometric Attributes", *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol.2, pp.172-179, 2001.

[8] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2) 1992, 239-256.

[9] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Image and Vision Computing* 10(3), 1992, 145-155.

[10] Blais, G. and Levine, M. "Registering Multiview Range Data to Create 3D Computer Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, 1995.

[11] T. Masuda, K. Sakaue, and N. Yokoya, Registration and Integration of Multiple Range Images for 3-D Model Construction. Proc. Computer Society Conference on Computer Vision and Pattern Recognition, 1996.

[12] P. J. Neugebauer. "Reconstruction of Real-World Objects via Simultaneous Registration and Robust Combination of Multiple Range Images." *International Journal of Shape Modeling*, 3(1&2):71-90, 1997.

[13] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. "Towards a general multi-view registration technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):540–547, May 1996.

[14] R. Benjemaa and F. Schmitt. "Fast global registration of 3d sampled surfaces using a multi-z-buffer technique," *Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modelin*g, pages 113–120, May 1997.

[15] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[16] David A. Simon, Martial Hebert and Takeo Kanade. Realtime 3-D pose estimation using a high-speed range sensor. In *IEEE Intl. Conf. Robotics and Automation*, pages 2235-2241,San Diego, California, May 8-13 1994.

[17] C. Langis , M. Greenspan and G. Godin "The parallel iterative closest point algorithm," In *Proc. International Conference on 3D Digital Imaging and Modeling (3DIM)*, Quebec City, Quebec. May 28- June 1, 2001.

[18] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: Freeman, 1979.

[19] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, 1990.

[20] L. Hagen, A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering." *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 9, pp. 1074-1085, September, 1992.

[21] G. Karypis and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, 20(1):359-392, 1998.

[22] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techniques", *IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems*, Vol. 16, No. 8, pp. 849-866, August 1997.

[23] A. George and J. W.-H. Liu. Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[24] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning of Electrical Circuits", Bell Systems Technical Journal, Vol. 49, No. 2, pp. 291- 307, 1970.

[25] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improved Network Partitions", Proc. Design Automation Conference, pp. 241-247, 1982.

[26] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", IEEE Transactions on Computers, Vol. C-33, No. 5, pp. 438-446, 1984.

[27] http://www.cyra.com.