Fast Simultaneous Alignment of Multiple Range Images Using Index Images

Takeshi Oishi
Institute of Industrial Science, The University of Tokyo
oishi@cvl.iis.u-tokyo.ac.jp

Atsushi Nakazawa
Osaka University
nakazawa@ime.cmc.osaka-u.ac.jp

Ryo Kurazume
Kyushu University
kurazume@is.kyushu-u.ac.jp

Katsushi Ikeuchi
Institute of Industrial Science, The University of Tokyo
ki@cvl.iis.u-tokyo.ac.jp

Abstract

This paper describes a fast, simultaneous alignment method for a large number of range images. Generally the most time-consuming task in aligning range images is searching corresponding points. The fastest searching method is the "Inverse Calibration" method. However, this method requires pre-computed look-up tables and precise sensor parameters. We propose a fast searching method using "index images," which work as look-up tables and are rapidly created without any sensor parameters by using graphics hardware. To accelerate the computation to estimate rigid transformations, we employed a linear error evaluation method. When the number of range images increases, the computation time for solving the linear equations becomes too long because of the large size of the coefficient matrix. On the other hand, the coefficient matrix has the characteristic to become sparser as the number of range images increases. Thus, we applied the Incomplete Cholesky Conjugate Gradient (ICCG) method to solve the equations and found that the ICCG greatly accelerates the matrix operation by pre-conditioning the coefficient matrix. Some experimental results in which a large number of range images are aligned demonstrate the effectiveness of our method.

1. Introduction

The modeling surface geometry of a real object is an important issue in the fields of computer vision and computer graphics. The theory of modeling from reality was proposed few years ago, and related technologies have been highly developed in recent years [1]. Preserving cultural heritage objects and art works in the world is one of the important applications of these technologies, as discussed in [2, 3]. In these projects, laser range sensors are utilized for digitizing the surface geometry of the objects because of the high accuracy of the sensors.

While the modeling technologies have 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 [4] and are adapted from the method proposed by Chen [5]. 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 [6, 7]. Since the ICP algorithm tends to be affected by false matching and noise, Masuda et al. proposed a robust method that uses random sampling and the Least Median Squares Estimation method (LMedS) [8].

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 [9]. Benjamaa et al. extended the method proposed by Bergevin et al. [10] and implemented a simultaneous alignment method while they accelerated the pair-wise alignment algorithm by using multi z-buffers [11].

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 [12, 13] that use *Kd-trees* and that narrow the search range by using data cache [14, 15, 16]. However, the complexity of *Kd-tree* search is $O(NlogN)$. That is, sufficient acceleration cannot be achieved by these algorithms. The computational complexity of the inverse calibration method proposed by Blais is $O(N)$ [6]. However, this method requires precise sensor parameters (intrinsic parameters of CCD camera, parameters of scanning mechanism) and pre-computed look-up tables. In addition, the creation of the look-up tables is very time consuming because Euclidian distances between each element of a table and every ray of sampled points have to be calculated.

Another problem in aligning a large number of range images is the computation cost of matrix operations in which rigid transformations of range images are computed. To directly solve a non-linear least squares problem is very time consuming [13]. In this case, the linearized algorithm is effective in dealing with a large data set [7]. However, the computation time to solve the linear equations with conventional solvers (SVD, Cholesky decomposition, etc.) rapidly increases as the number of range images increases because the coefficient matrix becomes very large.

We propose a fast method to align a large number of range images simultaneously. Our method has three characteristics. 1) The process of searching corresponding points is accelerated by using index images, which are rapidly created without sensor parameters. 2) The method employs the point-plane error metric and linearized error evaluation. 3) An iterative solver (incomplete Cholesky conjugate gradient method) is applied in order to accelerate the computation of the rigid transformations. In Section 2, the details of our algorithm are described. Some experimental results that demonstrate the effectiveness of our method are shown in Section 3. Our conclusions are described in Section 4.

2. Alignment algorithm

In this section, the details of our alignment algorithm are 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

First, we explain the fast method to search corresponding points. Then, the details of error evaluation and the computation of rigid transformations are described.

2.1. Correspondence search

Our algorithm employs points and planes to evaluate relative distance as the Chen and Medioni method [5]. The corresponding pairs are searched along the line of sight (Fig. 1). 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 all pairs of mesh models.

Though the threshold distance is given empirically as l_{given}, the smaller value is selected as l_{th} compared with the average distance of all corresponding points $\hat{r}$.

$$l_{th} = \begin{cases} l_{given} \ (if: l_{given} < \hat{r}) \\ \hat{r} \quad (otherwize) \end{cases} \tag{2}$$

$$\hat{r} = \frac{1}{N}\sum_{i}^{N} \left\| \mathbf{y}_i - \mathbf{x}_i \right\| \tag{3}$$

N is the number of vertices included in the base mesh.

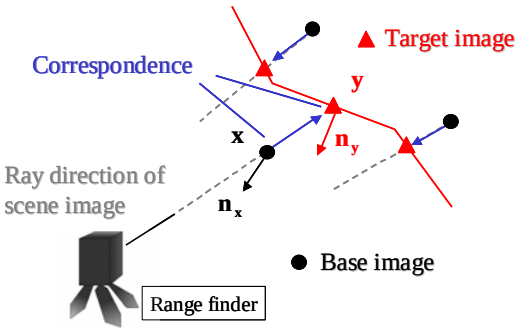

Figure 1. Searching corresponding points

To search correspondences quickly, our method uses *index images*. Though the complexity of this process is *O(N,),* the same as that of the inverse calibration method, sensor parameters are not required. Furthermore, the searching process can be accelerated by graphics hardware. The details of correspondence search using index images are described below.

2.1.1 Creation of index images

An index image works as a look-up table to retrieve the index of corresponding patches. Here, we describe the procedures for creating an index image as follows:

1. A unique index number is assigned to each triangle patch of a target mesh.
2. Index numbers are converted to unique colors.
3. Triangle patches of the target mesh are rendered on an image plane with the index colors.

First, a unique integer value is assigned to each triangle patch. Since the assigned value can be any integer number, 0 to n-1 are assigned sequentially, where n is the number of triangle patches.

Next, the assigned index values are converted to unique colors. If the precision of index values is the same as that of rendering colors, the index values are converted directly. Assume that the precision of each color channel is expressed by q bits. [0 q-1] bits of an index value are assigned to *Red*, and [q 2q-1] bits to *Green*, and the next to *Blue*, and highest q bits are assigned to *Alpha*. If the precision is not the same, the indices have to be converted carefully.

All triangle patches are rendered onto an image plane using the index colors (Fig. 2). The pixels in which the triangle patches are not rendered are filled with an exceptional color like white. The target mesh is assumed to be described in its measured coordinate system.

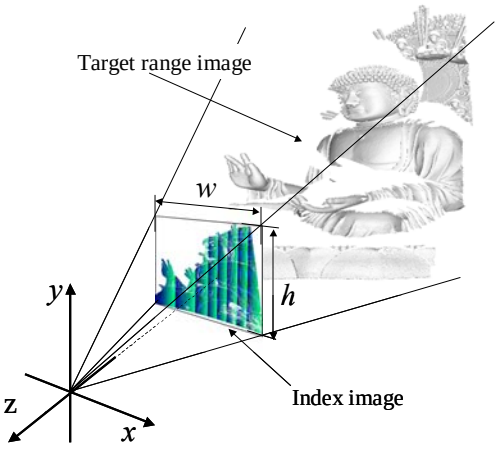

Figure 2. Rendering of index image

Projection method

Generally, perspective projection is used for rendering the index images. Perspective projection works well for the range images that are measured by sensors that adopt a method like light sectioning. On the other hand, in the case of range images taken by sensors with scanning mechanisms using mirrors, the spherical projection is better because the angles between sampled points are equal (or near) to each other.

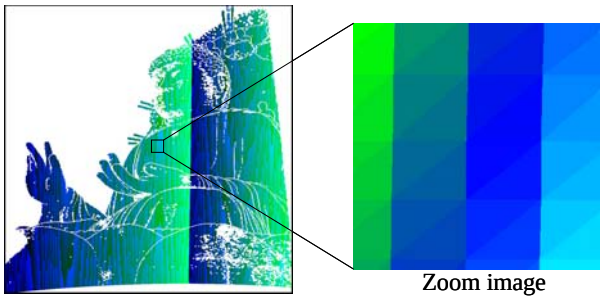

Figure 3. Example of index images

View frustum

To obtain a sufficient number of corresponding points, rendering areas have to be determined properly. All vertices of a target mesh are projected onto the index image plane. Then, the rectangular area ($u_{\min}$, $v_{\min}$, $u_{\max}$, $v_{\max}$) that involves all projected vertices is obtained. The view frustum is computed so that all vertices are rendered in this area. Minimum and maximum depths are also acquired at the projection process.

Image resolution

The resolution of an index image (L_u, L_v) has to be determined as the following conditions are fulfilled.

$$L_u \geq 2 \times w / \Delta w_{\min} \quad (4)$$

$$L_v \geq 2 \times h / \Delta h_{\min} \quad (5)$$

$$w = u_{\max} - u_{\min} \quad (6)$$

$$h = v_{\max} - v_{\min} \quad (7)$$

Variables w and h represent the height and width of the rendering area respectively (Fig. 2). $\Delta w_{\min}$ and $\Delta h_{\min}$ represent the minimum height and minimum width of all triangles projected onto the index image plane.

The image resolution (L_u, L_v) can be roughly determined so that the conditions described in inequality 4 and inequality 5 are satisfied. Although the parameters (w, h, Δw_{min}, Δh_{min}) are different in each partial mesh, a unique resolution that satisfies the conditions of all mesh models works well for all index images.

The rendering process is accelerated by using graphics hardware. The rendering time becomes small enough to ignore even if the images are rendered at each iterative step. A large memory space for storing look-up tables is not required. The memory space for only one index image can be shared by all mesh models. Figure 3 shows an example of index images.

2.1.2 Acquisition of corresponding points

By using the index image, corresponding points are rapidly searched. The procedure for this process is the reverse of that used to make an index image. Here, we assume that all vertices of the base mesh are previously converted to the local coordinate system of the target mesh. The following steps are applied to all vertices of the base mesh:

1. A vertex is projected onto the index image plane by the same projection method as is used for the index image.
2. A color is obtained from the projected pixel.
3. The obtained color is converted to the index value of a patch of the target mesh.
4. The vertex is projected onto the corresponding patch; a corresponding point is acquired.

The procedures are depicted in Fig. 4. A vertex of the base image is projected onto the index image plane. Then, a color is acquired from the projected pixel and is converted to the index of a corresponding patch. Since a correct index value may not be obtained because of round-off errors, the vertex is re-projected onto a corresponding patch, and the crossing point is checked to see whether it is inside the patch or not. If the crossing point is inside the patch, the accurate corresponding point is computed. Until the correct corresponding point is obtained, steps 2-4 are applied to 3×3 pixels around the projected pixel. The computational complexity of this process is also *O(N)*.

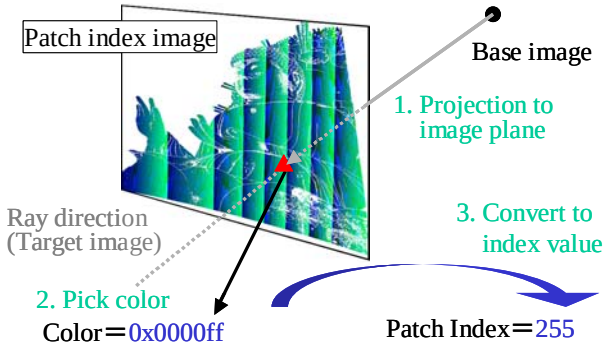

Figure 4. Searching a corresponding patch

2.2 Error metric

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 of a pair k is written as

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

$$\vec{n} = \frac{\vec{n}_x + \vec{n}_y}{\left\| \vec{n}_x + \vec{n}_y \right\|}, \tag{2}$$

where $\vec{n}_x$ and $\vec{n}_y$ are the normal vectors of $\vec{x}$ and $\vec{y}$ defined around the vertices respectively. Since normal vectors tend to be greatly influenced by measurement errors, we used the average normal vectors for error evaluation.

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

$$\varepsilon = \boldsymbol{R}_{\mathrm{B}}\vec{n} \cdot \{(\boldsymbol{R}_{\mathrm{T}}\vec{y} + \vec{t}_{\mathrm{T}}) - (\boldsymbol{R}_{\mathrm{B}}\vec{x} + \vec{t}_{\mathrm{B}})\} \tag{3}$$

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. To make the function simple, the average normal $\vec{n}$ is assumed to be rotated by the matrix of base range image $\boldsymbol{R}_B$. The distance between the base and the target mesh is expressed as

$$\bar{\varepsilon} = \min_{\boldsymbol{R},\vec{t}} \sum_{i,j,k} \left(\boldsymbol{R}_i \vec{n}_{ik} \cdot \{(\boldsymbol{R}_j \vec{y}_{ijk} + \vec{t}_j) - (\boldsymbol{R}_i \vec{x}_{ik} + \vec{t}_i)\} \right)^2 \tag{4}$$

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

$$\boldsymbol{R} = \begin{pmatrix} 1 & -c_3 & c_2 \\ c_3 & 1 & -c_1 \\ -c_2 & c_1 & 1 \end{pmatrix} \tag{5}$$

The translation vector is expressed as

$$\vec{t} = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}^{\mathrm{T}} \tag{6}$$

After some algebraic manipulations [8], equation 4 is rewritten as

$$\bar{\varepsilon} = \min_{\vec{\delta}} \sum_{i \neq j} \left\| \vec{A}_{ijk} \cdot \vec{\delta} - s_{ijk} \right\|^2 \tag{7}$$

$$s_{ijk} = \vec{n}_{ik} \cdot (\vec{x}_{ik} - \vec{y}_{ijk}) \tag{8}$$

$$\vec{A}_{ijk} = \left\{ \left(\underbrace{0 \ldots 0}_{6i \times 1} \vec{C}_{ijk}^{\mathrm{T}} \underbrace{0 \ldots 0}_{6(l-i-1) \times 1} \right) + \left(\underbrace{0 \ldots 0}_{6j \times 1} - \vec{C}_{ijk}^{\mathrm{T}} \underbrace{0 \ldots 0}_{6(l-j-1) \times 1} \right) \right\}^{\mathrm{T}} \tag{9}$$

$$\vec{C}_{ijk} = \begin{pmatrix} \vec{n}_{ik} \times \vec{y}_{ijk} \\ -\vec{n}_{ik} \end{pmatrix} \tag{10}$$

$$\vec{\delta} = (\vec{m}_0 \ldots \vec{m}_{n-1})^{\mathrm{T}} \tag{11}$$

$$\vec{m}_i = \begin{pmatrix} c_{1i} & c_{2i} & c_{3i} & t_{xi} & t_{yi} & t_{zi} \end{pmatrix}^{\mathrm{T}}, \tag{12}$$

where the number of mesh models is n. By (7) $\vec{\delta}$ is written as

$$\left(\sum_{i \neq j} \vec{A}^{\mathrm{T}}{}_{ijk} \vec{A}_{ijk} \right) \vec{\delta} = \sum_{i \neq j} \vec{A}^{\mathrm{T}}{}_{ijk} s_{ijk} \tag{13}$$

2.3 Solving linear equations

From Eq. 13, $\vec{\delta}$ is computed as the solution of linear equations that include n×6 arguments. However, ambiguity remains in the equation. Then, the first mesh model is assumed not to be moved. That is, as shown in Fig. 5, the linear equations with ((n-1)×6) × ((n-1)×6) coefficient matrix are solved. If all mesh models are connected to the first mesh, the coefficient matrix is symmetric positive definite. Also, the matrix becomes

larger and sparser as the number of mesh models increases and has 6×6 non-zero patterns as shown in Fig. 5.

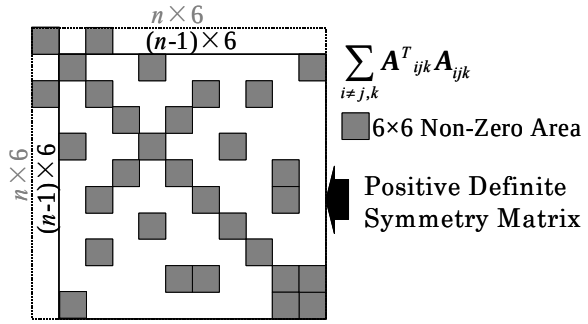

Figure 5 Characteristics of coefficient matrix

Since the computational complexity of direct solvers is too high, we applied an iterative solver to this problem. The computational complexity of Cholesky decomposition that is the most popular direct solver for a symmetric positive definite matrix is $O(n^3)$. Then, we employed the pre-conditioned conjugate gradient method (PCG). Though the complexity of PCG is $O(n^3)$, the same as Cholesky decomposition, the number of iterations can be drastically reduced by pre-conditioning. We employed incomplete Cholesky decomposition as the pre-conditioner. Since it is known that the coefficient matrix has 6×6 non-zero patterns, we implemented the incomplete Cholesky conjugate gradient method (ICCG) specialized for the matrix pattern.

Assume that the matrix $\boldsymbol{M}$ is the $((n\text{-}1)\times6) \times ((n\text{-}1)\times6)$ coefficient matrix shown in Fig. 5. $\vec{\beta}$ is the $(n\text{-}1)\times6$ vector that is a part of the right side of Eq. 13. $\vec{\beta}$ does not include the transformations of the first mesh model. Equation 13 is re-written as follows.

$$\boldsymbol{M}\vec{\alpha} = \vec{\beta} \tag{14}$$

$$\vec{\alpha} = (\vec{m}_1 ... \vec{m}_{n-1})^{\mathrm{T}} \tag{15}$$

A matrix $\boldsymbol{C}$ is assumed to be a regular ((n-1)×6)×((n-1)×6) matrix. Equation 14 is written as follows:

$$\boldsymbol{C}^{-1}\boldsymbol{M}(\boldsymbol{C}^{\mathrm{T}})^{-1}\boldsymbol{C}^{\mathrm{T}}\vec{\alpha} = \boldsymbol{C}^{-1}\vec{\beta} \tag{16}$$

To simplify the equation, we define the matrix $\tilde{\boldsymbol{M}}$ and the vector $\vec{\beta}'$ as follows.

$$\tilde{\boldsymbol{M}} = \boldsymbol{C}^{-1}\boldsymbol{M}(\boldsymbol{C}^{\mathrm{T}})^{-1} \tag{17}$$

$$\vec{\beta}' = \boldsymbol{C}^{-1}\vec{\beta} \tag{18}$$

Equation 16 is redefined by using the variables above.

$$\tilde{\boldsymbol{M}}\vec{\alpha}' = \vec{\beta}' \tag{19}$$

$$\boldsymbol{C}^{\mathrm{T}}\vec{\alpha} = \vec{\alpha}' \tag{20}$$

If the coefficient matrix $\tilde{\boldsymbol{M}}$ is near the identity matrix, solving Eq. 19 is drastically accelerated by the conjugate gradient method. That is, the $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}$ has to be nearly equal to the original coefficient matrix $\boldsymbol{M}$.

$$\boldsymbol{M} \cong \boldsymbol{C}^{\mathrm{T}}\boldsymbol{C} \tag{21}$$

But the computation cost to decompose the matrix is very high. The matrix $\boldsymbol{M}$ is incompletely decomposed by the Cholesky decomposition. In this process, only non-zero areas are computed: Another element is filled with zero.

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D} \tag{22}$$

Since the matrix $\boldsymbol{M}$ is sparse, the decomposition process is performed very quickly. Then, matrix $\boldsymbol{C}$ is given as follows:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}^{1/2} \tag{23}$$

Once matrix $\boldsymbol{C}$ has been computed, rigid transformations are calculated from Eq. 19 and Eq. 20 by the conventional conjugate gradient method.

3. Experimental results

In this section, the effectiveness of our method is demonstrated by some experimental results. Two data sets are used for the experiments. Target objects are the face of Deva in Cambodia (Fig. 6(a)) and the Nara Great Buddha statue in Japan (Fig. 6(b)). The face of Deva was measured by VIVID900 [19]. The resolution of VIVID900 was fixed to 640×480, and the view angle depended on mounted lenses. We used a wide lens for scanning the face of Deva. The Great Buddha statue was measured by Cyrax2400 [20]. The resolution and view angle of the sensor were flexible: users could change them arbitrarily. In the scanning of Nara Great Buddha, we adjusted the parameters according to measurement environments. Generally, 800×800 was used as the measurement resolution. The details of these data sets are shown in Table 1 and Table 2 respectively.

Table 1. Data set 1: The face of Deva

Sensor	VIVID900
Images	45
Vertices	Max: 76612, Min: 38190, Ave: 67674
Triangles	Max: 150786, Min: 71526, Ave: 130437

Table 2. Data set 2: The Nara Great Buddha

Sensor	Cyrax2500
Images	114
Vertices	Max: 155434, Min: 11231, Ave: 81107
Triangles	Max: 300920, Min: 18927, Ave: 148281

Vertices that measured outside the objects had been removed previously. Obtained point crowds had been converted into triangle mesh models. Since the original data sets were too large to deal with using one PC, the sizes of the data were reduced to 1 / 4.

Our method is evaluated according to the following three criteria:

1. Number of corresponding points with respect to the resolution of index images
2. Computation time of matrix operations with respect to the number of mesh models
3. Computation time of alignment with respect to the

number of vertices

The PC used for the experiments had Athlon MP 2400+ processor, 2Gbyte memory, and a GeForce4Ti4600 graphics card.

(a) (b)

Figure 6. Target objects
(a : the face of Deva, b : the Nara Great Buddha)

3.1 Number of corresponding points with the resolution of the index image

The number of corresponding points is evaluated with respect to the resolution of the index image. As described above, if enough resolution cannot be assigned to the index image, all triangle patches are not rendered: all the corresponding points cannot be acquired. Here, the relation between the resolution of index images and the number of corresponding points is verified.

Corresponding points were searched for several pairs of mesh models by gradually changing the resolution of index images. We selected a set of mesh models that have minimum and maximum number of triangle patches from each data set as target meshes. Base meshes were arbitrarily selected. Experimental results are shown in Fig. 7. The vertical axis represents the ratio of the number of corresponding points acquired by our method v'_c to the ground truth v_c. The resolutions of index images are represented in the horizontal axis as the square root of total pixels.

As shown in Fig. 7, when the resolution of the index image becomes larger than a certain size (800×800), almost all of the corresponding points are obtained. Furthermore, the resolution required for obtaining enough corresponding points becomes larger as the number of triangle patches of target mesh increases. In fact, instead of the number of triangle patches, the measurement resolution is concerned with the required index image resolution. However, in this case, it can be said that the number of triangle patches has the similar characteristics with the measurement resolution because the sampled points are distributed densely and uniformly in both vertical and horizontal directions.

It is not required to estimate the image resolution for each mesh model. Even if, due to any problems, for example the limitation of graphics memory space, a large enough resolution cannot be assigned to the index image, several corresponding points are acquired in compliance with the image resolution.

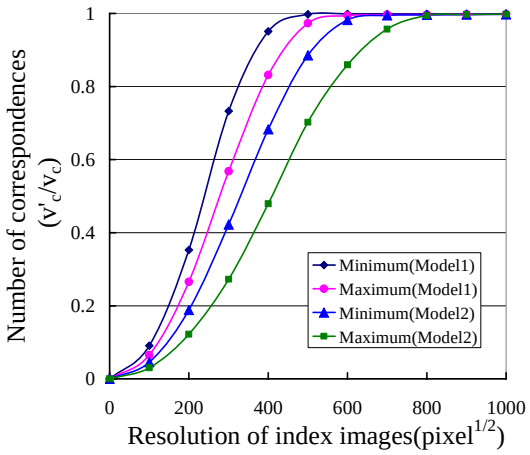

Figure 7. Number of corresponding points with resolution of index images

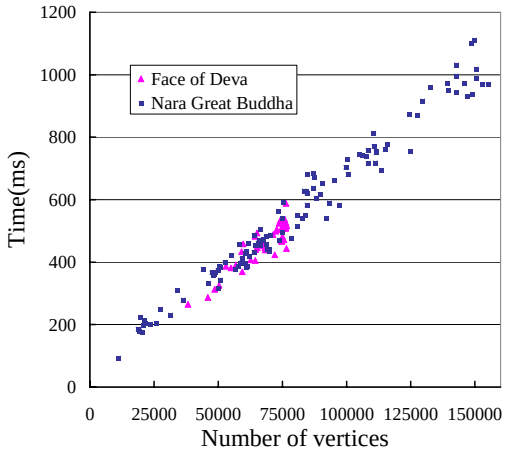

Figure 8. Time to solve linear equations

3.2 Time to solve the linear equations

The computation time to solve the linear equations is evaluated. As described above, the computation time to solve the linear system is greatly influenced by the number of mesh models. Thus, the relation between the computation time and the number of mesh models is evaluated here. Data set 2 is used for this experiment. The computation time of the matrix operations only is sequentially measured by changing the number of mesh models.

The experimental results are shown in Fig. 8. The horizontal axis represents the number of mesh models, and the vertical axis represents the computation time. The results with usual Cholesky decomposition are also shown in this figure in comparison with our method. The threshold value of the ICCG was set to 1.0×10^{-6}.

In the case that the number of mesh models is lower than 60, Cholesky decomposition is faster than our method. On the other hand, the computation time of our

method increases at a slow rate and becomes smaller than that of Cholesky decomposition when the number of mesh models is higher than 70. Moreover, the differences between Cholesky decomposition and our method become larger as the number of mesh models increases. That is, it can be said that ICCG is effective for aligning a large number of mesh models simultaneously.

3.3. Computation time with number of vertices

The computation time is evaluated with respect to the number of vertices included in the mesh models. In this experiment, it is proven that the computational complexity of aligning a pair of mesh models is *O(N)*, where *N* is the number of vertices. It can be also said that the complexity of searching corresponding points is *O(N)* because it can be assumed that the computation time of another task is small enough to ignore.

Each mesh model was aligned to itself so that the number of vertices of base model and target model were equal to each other. Since the number of corresponding points also affects the computation time, all mesh models were not moved. That is, the amount of movement is infinitely zero; the number of corresponding points is nearly equal to the number of vertices. Index images were rendered at each iterative step. The image resolution was fixed to 800×800. The computation time was evaluated according to the average time taken for 20 iterations.

Experimental results are shown in Fig. 9. The horizontal axis represents the number of vertices, and the vertical axis represents the computation time. It is clear that the computation time is increasing linearly with the number of vertices. Moreover, there are no differences between two data sets though these data were taken by different sensors. That is, the efficiency of our method does not depend on the sensors used for measurements.

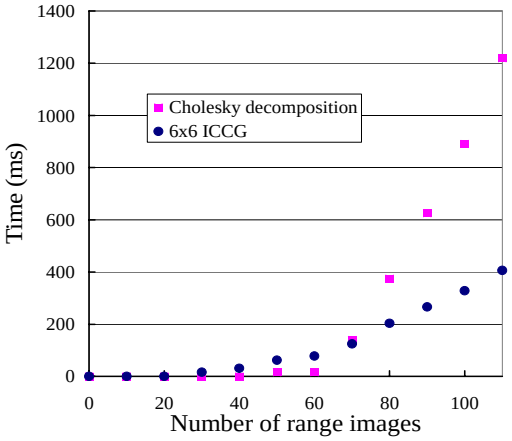

Figure 9. Computation time with number of vertices

3.4. Alignment results

The alignment results of these data sets are shown in Fig. 10 and Fig. 11. Previously, we had aligned all mesh models one by one. Then we applied the simultaneous alignment method to these models. Figure 10 shows the alignment results of data set 1. The total computation time was 1738 seconds after 20 iterations. Figure 11 shows the results of data set 2. The number of iterations was 20, the same as for data set 1. Total computation time was 7832 seconds. The figures show that all mesh models were correctly aligned.

Figure 10. Alignment results (the face of Deva)

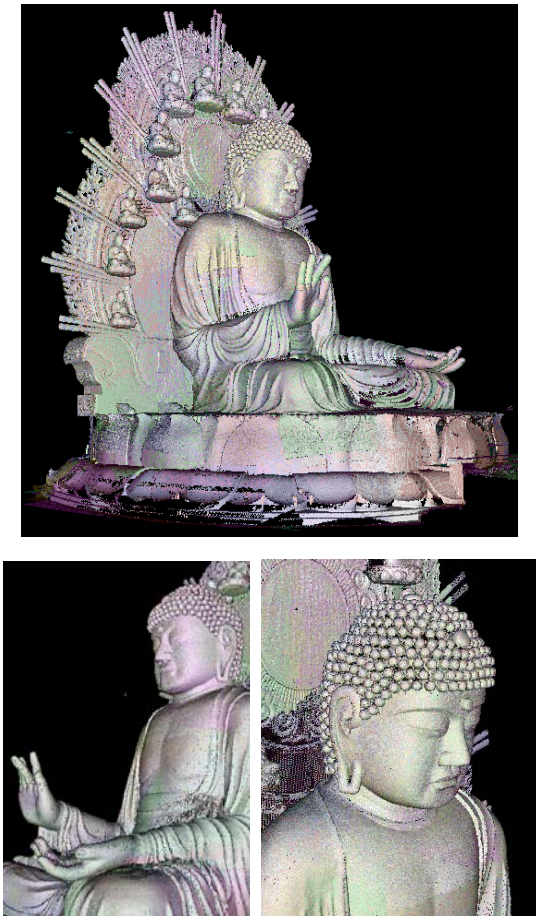

Figure 11. Alignment results (Nara Great Buddha)

4. Conclusion

In this paper, we proposed a fast, simultaneous alignment method for a large number of range images. In order to accelerate the task of searching corresponding points, we utilized index images that are rapidly rendered using graphics hardware and are used as look-up tables. Instead of sensor parameters, only an approximate resolution of index images is required for this search. In order to accelerate the computation of rigid transformations, we employed a linearized error function. Since the computation time to solve the linear system becomes large as the number of range images increases, we applied the incomplete Cholesky conjugate gradient (ICCG) method. Experimental results showed the effectiveness of our method. One of our future works will be to improve the accuracy of alignment results.

Acknowledgment

This work is supported by Leading Project (Digital archive of cultural properties) of the Ministry of Education, Culture, Sports, Science and Technology of the Japanese Government. The author would like to thank the staffs of the Todaiji Temple in Nara, Japan. The Bayon temple in Cambodia was digitized with the cooperation of JSA (Japanese Government Team for Safeguarding Angkor).

References

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

[2] K. Ikeuchi, A. Nakazawa, K. Hasegawa and T. Oishi, "The Great Buddha Project: Modeling Cultural Heritage for VR Systems through Observation," *IEEE ISMAR'03*, Tokyo, Japan. Nov., 2003.

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

[4] 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), 239-256, 1992.

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

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

[7] S. Rusinkiewicz, O. Hall-Holt and M. Levoy, "Real-Time 3D Model Acquisition," *ACM Transactions on Graphics.* 21(3): 438-446, July 2002.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] K. Nishino and K. Ikeuchi, "Robust Simultaneous Registration of Multiple Range Images," *Proc. Fifth Asian Conference on Computer Vision*, pp454-461, Jan., 2002.

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

[15] M. Greenspan and G. Godin, "A Nearest Neighbor Method for Efficient ICP," *Proc. International Conference on 3D Digital Imaging and Modeling (3DIM)*, pp.161-168, 2001.

[16] R. Sagawa, T. Masuda and K. Ikeuchi, "Effective Nearest Neighbor Search for Aligning and Merging Range Images," *Proc. the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM-03)*, pp.79-86, 2003.

[17] http://www.minoltausa.com/vivid/

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