International Journal of Computer Vision, 75(1), 189–208, 2007

The Great Buddha Project: Digitally Archiving, Restoring, and Analyzing Cultural Heritage Objects

KATSUSHI IKEUCHI, TAKESHI OISHI, JUN TAKAMATSU, RYUSUKE SAGAWA, ATSUSHI NAKAZAWA, RYO KURAZUME, KO NISHINO, MAWO KAMAKURA AND YASUHIDE OKAMOTO
University of Tokyo, Japan

Abstruct. This paper presents an overview of our research project on digital preservation of cultural heritage objects and digital restoration of the original appearance of these objects. As an example of these objects, this project focuses on the preservation and restoration of the Great Buddhas. These are relatively large objects existing outdoors and providing various technical challenges. Geometric models of the great Buddhas are digitally achieved through a pipeline, consisting of acquiring data, aligning multiple range images, and merging these images. We have developed two alignment algorithms: a rapid simultaneous algorithm, based on graphics hardware, for quick data checking on site, and a parallel alignment algorithm, based on a PC cluster, for precise adjustment at the university. We have also designed a parallel voxel-based merging algorithm for connecting all aligned range images. On the geometric models created, we aligned texture images acquired from color cameras. We also developed two texture mapping methods. In an attempt to restore the original appearance of historical objects, we have synthesized several buildings and statues using scanned data and a literature survey with advice from experts.

Keywords: cultural heritage, range data, alignment, merging, texturing

1. Introduction

Currently, a large number of cultural heritage objects around the world are deteriorating or being destroyed because of natural weathering, disasters, and civil wars. Among them, Japanese cultural heritage objects, in particular, are vulnerable to fires and other natural disasters because most of them were constructed of wood and paper.

One of the best ways to prevent these objects from loss and deterioration is to digitally preserve them. Digital data of heritage objects can be obtained by using modern computer vision techniques. Once these data have been acquired, they can be preserved permanently, and then safely passed down to future generations. In addition, such digital data can be used for many applications that aim to restore real objects through digital simulation and to plan restoration projects through precise measures given by such digital models. And by creating multi-media content from digital data, a user can view digital contents through the Internet from anywhere in the world, without moving the objects or visiting the sites.

One of the origins of this line of research is Kanade's virtualized reality project [1]. The basic idea of this project was to create 3D virtual reality models through observation of the real objects. Kanade and his students constructed an experimental room equipped with multiple TV cameras, which were focused in the center of the room. By observing some event at the center of the room, such as a player playing basketball, they obtained sequences of 2D images captured through synchronized TV cameras. By applying a multi-baseline stereo algorithm or its variation to these image sequences, they created a series of 3D models of an object, a moving 3D model. By combining this moving 3D model with a background 3D model and rendering the resulting synthesized 3D model, they generated a series of 2D images from many

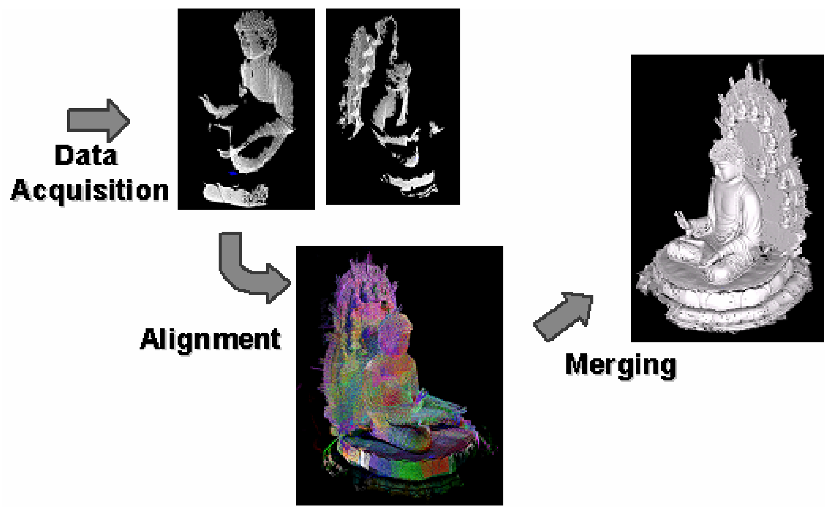


Figure 1. Three steps in geometric modeling.

directions. This line of research has been extensively explored [2-5] and a couple of representative results are included in this special issue.

Another line of research has been directed toward modeling of various cultural heritage objects in a very precise manner, using laser range sensors [6]. Recently, technologies of laser sensing have drastically advanced, and laser range sensors provide very accurate 3D range images of an object. However, those range images, given by a range sensor, are partial mesh models of an entire object, obtained from arbitrary directions. Thus, the research issues include how to align, or to determine relative relations, among such partial meshes, and how to merge these aligned partial mesh models into a unified mesh model of the object. Representative examples include Stanford University's Michelangelo Project [7], IBM's Pieta Project [8], and Columbia University's French cathedral project [9], to name a few.

We have been working to develop digital archival methods by using laser range sensors [6]. Our project has a number of unique features; among them is its focus on digitizing large outdoor objects such as the Kamakura great Buddha and Cambodia's Bayon temple. These large-scale objects present several challenges in processing range data into a unified mesh model. Our project emphasizes not only geometric modeling but also photometric and environmental modeling, particularly for outdoor objects.

The remainder of this paper is organized as follows. Section 2 describes the outline of the geometric pipeline developed, a rapid alignment algorithm based on graphics hardware, a parallel simultaneous alignment algorithm based on a PC cluster, and a parallel voxel-based merging algorithm, solving the issues given under digitizing large objects. Section 3 describes methods to align observed textures from a digital camera with range data for texturing large outdoor objects. Section 4 reports our efforts to restore the original appearance of these objects using acquired digital data and a literature survey. Section 5 summarizes this paper.

2. Geometric Modeling

2.1. Overview

Figure 1 shows an overview of three steps in geometric modeling: data acquisition, alignment, and merging. Several computer vision techniques, such as traditional shape-from-X and binocular stereo, or modern range sensors, provide 3D data points. In this paper, we mainly utilize laser range sensors as the input device of the 3D data points. These 3D data are represented as a set of 3D data points with connecting triangular arcs. This data representation is referred to as a (triangular) mesh model of an object. By repeating the data acquisition process so as to cover the entire object's surface, we have a set of partial mesh models, overlapping each other and covering the entire object surface as the combination of those partial mesh models.

The second step in geometric modeling is to align these partial mesh models. Since each sensor is located at an arbitrary position on data acquisition, we have to determine relative relations of these partial mesh models, referred to as alignment, by considering resemblances in the data set. When we handle a large object, even if we can assume that the sensor itself maintains the same degree of data accuracy as one for a smaller object, each scan covers a smaller portion of the object; many scans are necessary to cover the entire surface of the object. As the result, a long alignment sequence is formed. It is important to avoid accumulation of errors along this sequence of alignment.

The third step is to integrate the aligned multiple mesh models into a complete mesh model, representing an entire surface of an object. This step is referred to as 'merging.' The procedure can be considered as determining one surface position from multiple overlapping surface observations. In the merging procedure, it is important to make the integration framework robust against any noise that may occur when scanning the range images or that may be inherited from the registration procedure.

2.2. Alignment

2.2.1. A Rapid Alignment Based on Ggraphics Hardware.

Our rapid algorithm employs points and planes to evaluate relative distance as the Chen and Medioni and Gagnon et al. methods [13,15]. Scanning a large architectural object requires using different types of range sensors, due to complex scanning conditions at sites, whose resolutions may be different from each other. An alignment algorithm, based on point correspondence such as ICP [10], does not work well on such range data due to inequality in point resolutions. We can mitigate this situation with an algorithm based on point-face correspondence, because it creates virtual points on face patches, and sets up correspondence between the real points and virtual points, even if there is unbalance in point distributions.

Our algorithm uses an M-estimator, in particular, the Lorentzian function, as the error measure, to avoid the effects from the outliers [18]. It is well known that the L-2 norm is susceptive to noise. Surfaces of cultural heritage objects are often covered by foreign materials such as molls or water. Sometimes returned range values are quite noisy from such surfaces. The M-estimator effectively removes outliers. Since we prefer a smooth differentiable function for effective minimization process, we chose the Lorentian function as the evaluation function.

In order to increase computational efficiency, our algorithm is designed to utilize graphics processing hardware. The corresponding pairs are searched along the line of sight for the usage of the graphics hardware. Here, the line of sight is defined as the optical axis of a range sensor. While this does not guarantee that all the corresponding pairs are correct, using the M-estimator removes the effect of such mismatching. It is also true that after several iterations of the minimization process, the process eventually converges into the correct corresponding pairs. But for the sake of rapid computation, we decided to use the line of sight searching method.

Let us denote one mesh as the model mesh and its corresponding mesh as the scene mesh. One vertex in the model mesh is depicted as the point in Fig. 2. Each triangular mesh in the scene model is assigned one particular color, and then the color map is depicted on an index plane using the painting capability of graphics hardware. An extension of the line of sight, from a vertex of the model mesh, crosses a triangular mesh of the scene mesh and creates the intersecting point. This operation is implemented using the Z-buffer capability of the graphics hardware. 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 combination of mesh models.

The error measure between corresponding points is the cosine distance between the point and the plane. Let the vertex of the model mesh and the corresponding crossing point in the scene 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 model and scene

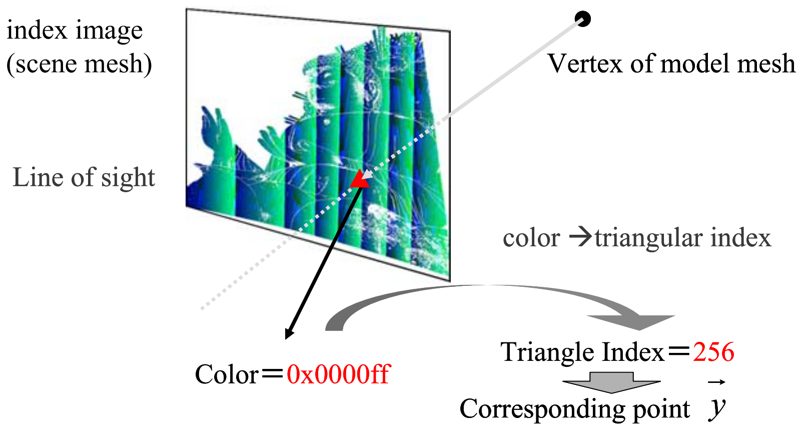


Figure 2. Search procedure.

meshes are computed so that this error measure is minimized. The error evaluation function is rewritten as

$$R_M\vec{n}\cdot\{(R_S\vec{y}+\vec{t}_S)-(R_M\vec{x}+\vec{t}_M)\} \tag{2}$$

Here, the rotation matrix and the translation vector of the model and scene mesh are $R_M, R_s, \vec{t}_M, \vec{t}_s$ respectively. The distance between the model and the scene mesh is expressed as

$$\varepsilon^2=\min_{R,t}\sum_{s\neq m,k}\left(R_M\vec{n}\cdot\{(R_S\vec{y}+\vec{t}_S)-(R_M\vec{x}+\vec{t}_M)\}\right)^2 \tag{3}$$

If it is assumed that the angles of rotation are small, the rotation matrix R can be approximated as

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

The translation vector is expressed as

$$t=\begin{pmatrix}t_x & t_y & t_z\end{pmatrix} \tag{5}$$

After some algebraic manipulations [17], Equation (3) is rewritten as

$$\varepsilon^2=\min_{\vec{\delta}}\sum_{m\neq s,k}\left\|A_{msk}\vec{\Delta}-\gamma_{msk}\right\|^2 \tag{6}$$

Here, *m,s,k* denotes one model mesh, one scene mesh, and one corresponding pair, respectively.

$$\gamma_{msk}=\vec{n}_{mk}\cdot(\vec{x}_{mk}-\overrightarrow{y_{msk}}) \tag{7}$$

$$A_{msk}=\begin{pmatrix}\underbrace{0...0}_{6m\times1}\underbrace{C_{msk}}_{6\times1}\underbrace{0...0}_{6(N-m-1)\times1}\end{pmatrix}+\begin{pmatrix}\underbrace{0...0}_{6s\times1}\underbrace{-C_{msk}}_{6\times1}\underbrace{0...0}_{6(M-s-1)\times1}\end{pmatrix} \tag{8}$$

$$C_{msk}=\begin{pmatrix}\overrightarrow{n_{mk}}\times\overrightarrow{y_{msk}}\\ -\overrightarrow{n_{mk}}\end{pmatrix} \tag{9}$$

$$\vec{\Delta}=\begin{pmatrix}\delta_0\cdots\delta_{N-1}\end{pmatrix} \tag{10}$$

$$\delta_m=\begin{pmatrix}c_{1m} & c_{2m} & c_{3m} & t_{xm} & t_{ym} & t_{zm}\end{pmatrix} \tag{11}$$

where the number of range images is N. By (6) $\vec{\Delta}$ is written as

$$\vec{\Delta}=\left(\sum_{m\neq s,k}A^T{}_{msk}A_{msk}\right)^{-1}\sum_{m\neq s,k}A^T{}_{msk}\gamma_{msk} \tag{12}$$

To avoid accumulation of errors, we have developed a simultaneous alignment method. Traditional sequential methods [10,13,14] such as the Iterative Closest Point (ICP) algorithm align these meshes one by one, and progressively align a new partial mesh with previously aligned meshes. If a few partial meshes can cover the entire surface of an object, the accumulation of alignment errors is relatively small and can be ignored; sequential alignment works well for a small object. However, for large objects, sometimes, we may need more than a hundred partial mesh models. In such cases, the error accumulation would be very large, if we employ sequential alignment. Thus, we align all partial meshes so as to reduce the errors among all the pairs simultaneously in equation (6).

We have implemented a simultaneous alignment algorithm, and verified the effectiveness of the algorithm, with respect to the pair-wise alignment algorithm. Since we need a true value of the aligned result, we generate synthesized 49 data sets by segmenting a whole 3D mesh model of the Great Buddha of Kamakura with small overlapping boundaries. We add a Gaussian noise, whose standard deviation is 3 mm with 1cm cut-off. This level of noise is typical in the data obtained by our sensors. These data were perturbed along a random direction to simulate initial alignment errors in the manual alignment process. Figure 3 shows synthesized range data. The simultaneous and the pair-wise alignment programs are applied to these data. Figure 4 shows the resulting converging process. While the simultaneous algorithm provides a constant error along the number of data sets, the pair-wise algorithm accumulates errors from the true value along the number of data sets processed in the horizontal axis.

2.2.2. Parallel alignment based on a PC cluster. The rapid simultaneous alignment algorithm, described in the previous section, is mainly utilized for rough alignment of the data acquired daily at the site. Although the algorithm employs simultaneous alignment for accuracy, the algorithm, designed for a single notebook type PC, without considering the aspect of the memory capability, may cause memory overflow for large-scale data of the entire target. We have designed a parallel alignment algorithm based on a PC cluster for large-scale alignment of the entire data set.

2.2.2.1. Overview of the Parallel Algorithm. The simultaneous alignment 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

Among these operations, 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.

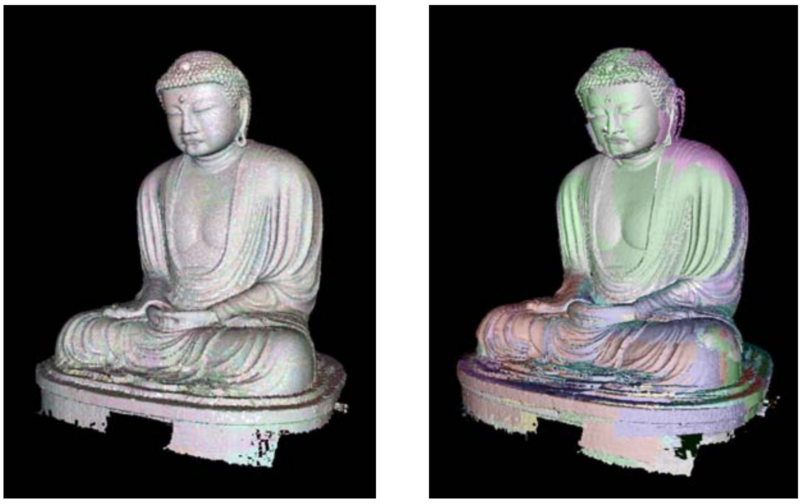


Figure 3. Synthesized range data. (a) Original whole mesh model of Kamakura Buddha, (b) Synthesized range data with initial perturbation and Gaussian noise to simulate alignment error and simulate sensor errors, respectively.

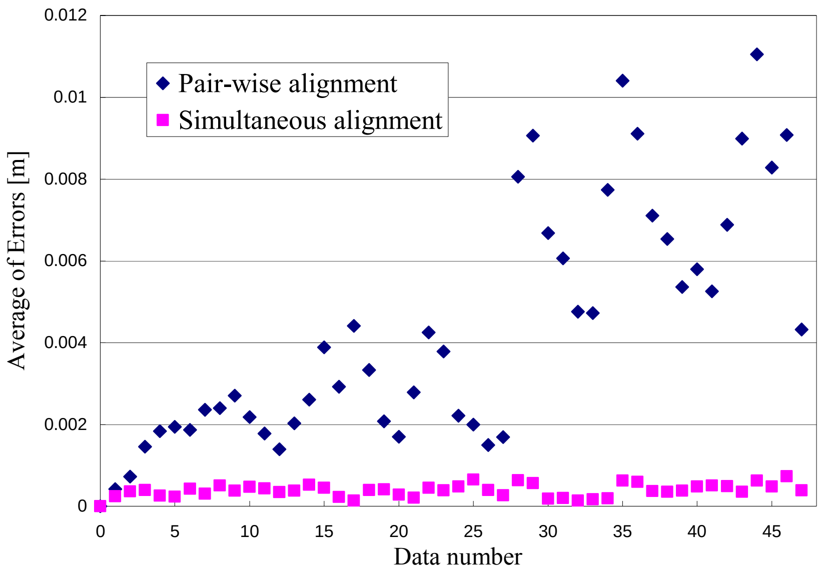


Figure 4. Converged result.

2.2.2.1. Graph simplification. We remove redundant or weak data dependency relations of partial mesh models for the sake of efficiency in parallel computation. Figure 5 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.

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 three conditions:

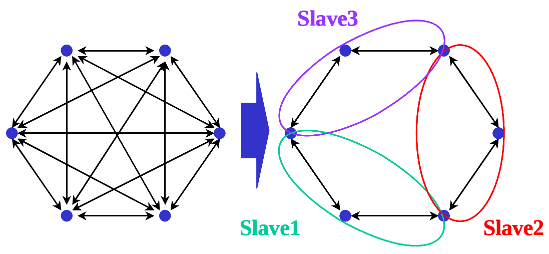


Figure 5. Data dependency relation.

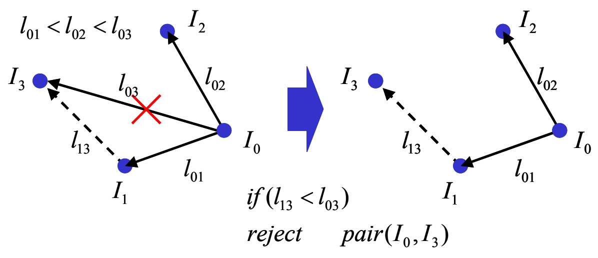


Figure 6. Non-adjacency relation.

1. *The bounding-boxes of two range images 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 $\theta = 90°$. 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 $\theta = 90°$ for the sake of safety and simplicity.

3. *Two range images are adjacent to each other.* This condition removes non-adjacent relations sequentially. For example, as shown in Fig. 6, 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.

2.2.2.3 Parallelization by graph partitioning algorithms. The problem of load balancing with a minimum amount of required memory is an NP-hard problem. 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.

Pair-node hyper-graph. First, we define the pair-node hyper-graph. The left image of Fig. 7 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 Fig. 7. We refer to it as a “pair-node hyper-graph.”

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

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 Fig. 7, 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.

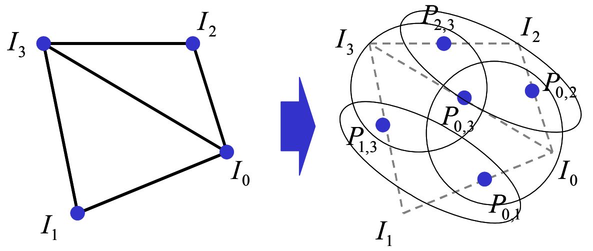


Figure 7. Pair-node hyper-graph.

Initial partitioning: The pair-node hyper-graph is initially partitioned so that the sum of the node-weights in each subset is roughly equal. 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.

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. 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 *k*th 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 Fig. 8 for the flow chart of the refinement process.

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 range images
*/
Create-Pair-Table;
/* Create the lists of the files for each processor */
  Create-File-Lists:
  while(error > threshold){
     /* Slave Process*/
     for(i = 0; i < nImage; ++i)
       for(j = 0; j < nImage; ++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 range images into memory. Then, each slave computes the matrices $A^T{}_{msk} A_{msk}$ and $A^T{}_{msk} \gamma_{msk}$ 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 $A^T{}_{msk} A_{msk}$ and $A^T{}_{msk} \gamma_{msk}$ 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. Equation (12) is evaluated in all the data. The matrix is very sparse; the ICCG provides computation 10 times faster than usual SVD.

2.3. Merging

Once we can determine relative configurations among all the mesh models through the alignment operations, we can connect these aligned mesh models into a unified mesh model representing an entire object's surface. This operation is referred to as merging. One of the simplest methods is to directly connect each mesh, and remove overlapping meshes. This direct method discards multiple observations of overlapping regions, and creates a sharp jump around the connecting boundaries. In order to avoid these shortcomings, we employ a voxel-based merging algorithm.

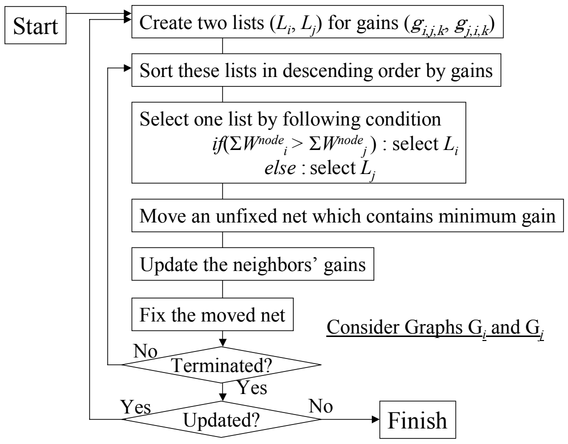

Start
Create two lists (L_i, L_j) for gains ($g_{i,j,k}$, $g_{j,i,k}$)
Sort these lists in descending order by gains
Select one list by following condition
$if(\Sigma W^{node}_i > \Sigma W^{node}_j)$: select L_i
$else$: select L_j
Move an unfixed net which contains minimum gain
Update the neighbors' gains
Fix the moved net
Consider Graphs G_i and G_j
No
Terminated?
Yes
Yes
Updated?
No
Finish

Figure 8. Flow chart of refinement Process.

2.3.1. Voxel-based Merging Algorithm. After all partial mesh models have been aligned, a volumetric view-merging algorithm generates a consensus mesh of the object from them. Our method, first, calculates a volumetric implicit-surface representation, where each voxel in the volumetric space has an estimated signed distance from the position of the estimated consensus mesh. Unlike previous techniques based on implicit-surface representations, our method estimates the signed distance to the object's surface by determining a consensus of locally coherent observations of the surface [20-23]. This consensus method is effective, in particular, for noisy data, often provided at the occluding boundaries.

We utilize octrees to represent volumetric implicit surfaces, thereby effectively reducing the computation and memory requirements of the volumetric representation without sacrificing accuracy of the resulting surface. Then, this signed distance representation is converted to a surface mesh by using a variant of the marching-cubes algorithm [19].

We originally designed software for a single PC. However, because recent input data is unpredictably huge, we decided to build a PC cluster to run this merging software; the cluster parallel-processes the merging algorithm to save computation time and utilize the large memory space of many PCs [24]. We produced one integrated digital of the Great Buddha of Kamakura and another of the Bayon Temple in Cambodia with this software.

2.3.2. Results of merging process. We have digitally archived Japanese Buddhas, including Asuka (7th century), Kamakura (13th century), and Nara (17th century). Here, the great Buddha of Nara, although the name indicates the Nara period of the 8th century, was burned, and the current one was rebuilt in the 17th century. Thus, it is the newest among the three. It is interesting to note that the faces of the Buddhas have become wider as the time passed, as shown in Fig. 9.

2.4. Analysis of geometric modeling

We extended this effort toward foreign cultural heritage objects and obtained all 173 Buddha faces of the Bayon temple in the Angkor ruin in Cambodia. Eighteen of these faces are shown in Fig. 10. According to the JSA (Japanese Government Team for Safeguarding Angkor) research, those faces can be classified into three categories: Deva, Devata, and Ashura. See Fig. 11 for examples. However, some of them are quite difficult to classify due to parallel work by different craftsmen and different techniques employed. Others have deteriorated due to weathering. And still others are unfinished.

In order to classify these faces in an accurate manner, we examined the 173 faces using cluster analysis. First, we converted all the faces into normalized depth images, defined a standard face among 173 faces, and adjusted all the remaining faces onto this standard face using a classification procedure. We extracted three key points: a pair of inner corners of the eyes, and the middle point between the mouth and the nose. We obtained translation, rotation, and scaling parameters to minimize the differences between two sets of key points of the standard and current faces. We applied those parameters to the range data of the current face, and obtained a 64 by 64 range image of the current face. We conducted two types of analyses: supervised and unsupervised. Details are given in [25].

2.4.1 Supervised analysis: Linear discrimination function. The purpose of supervised analysis is to clarify the differences among the given classes. JSA has already classified all faces into three types based on subjective evaluations by an artist. Through such a supervised analysis, we can verify correctness of the process, and then objectively evaluate the differences using statistical analysis methods.

In this paper, we use a linear function $f(\mathbf{x}) = \mathbf{n} \cdot \mathbf{x} + d$ as the classification function. The dimension of the sample space, that is, image size ($= 64 \times 64$, in this case), is much greater than the number of samples; there are only 173 faces in the Bayon Temple. It is preferred that the dimension and parameters of the function are small in order to prevent a so-called "over-fitting" problem.

Table 1. 3D data of great Buddha.

	Height (m)	Number of vertices	Number of meshes
Asuka (7th Century)	2.7	1.5 Million	2.9 Million
Kamakura(13th Century)	13	5.0 Million	9.8 Million
Nara (17th Century)	15	36.3 Million	69.1 Million

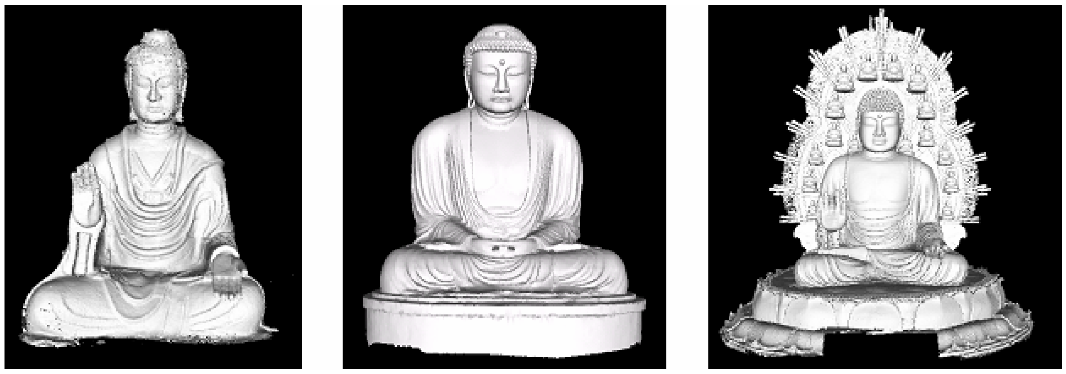


Figure 9. Three great Buddha: Asuka (7th century), Kamakura (13th century), and Nara(16th century) Buddha.

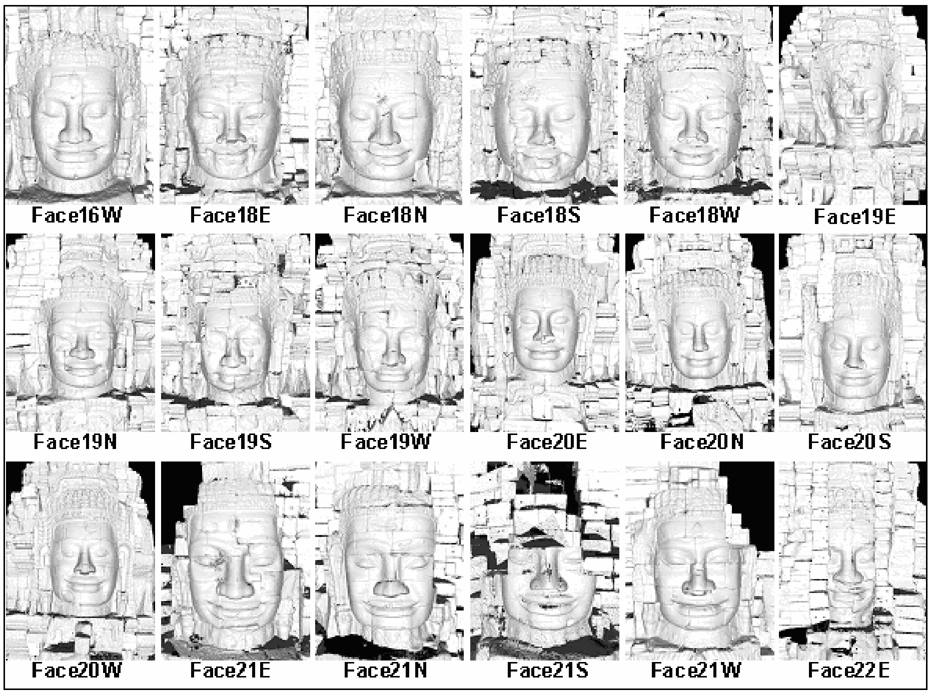


Figure 10. Bayon face library: 18 faces out of 153 faces in Bayon temple in Angkor ruin, Cambodia.

Roughly speaking, the parameters of the function $\mathbf{n}, d$ can be determined by maximizing S_B/S_T, where S_B and S_T are intraclass and interclass variances, respectively. Because the matrix S is not full-rank matrix, we solve the equation using the singular value decomposition (SVD) method while minimizing $|\mathbf{n}|$. Figure 10 illustrates the result of the analysis. The graph is obtained by projecting all vector points of faces on the 2D flat surface that is determined by two linear discriminant functions in order to clearly express the classification result.

The former discriminant function classifies them into Devata (female god) and Deva (male god) with separate planes corresponding to the y-axis in Fig. 12. The right side area is a female area, and the left side area is a male area. Although Asura data (male devil) are not used for training the function, almost all of Asura data are in the male area.

The second function classifies them into Deva (god) and Asura (devil) with separate planes corresponding to the diagonal line. The upper side area of the diagonal line is a god area and the lower side is a devil area. Almost all of Devata (female god) are also in the god area, even though we did not use Devata data for training this function.

2.4.2. Cluster analysis. The purpose of unsupervised analysis is to discover new knowledge through classification of the faces without any a priori standards.

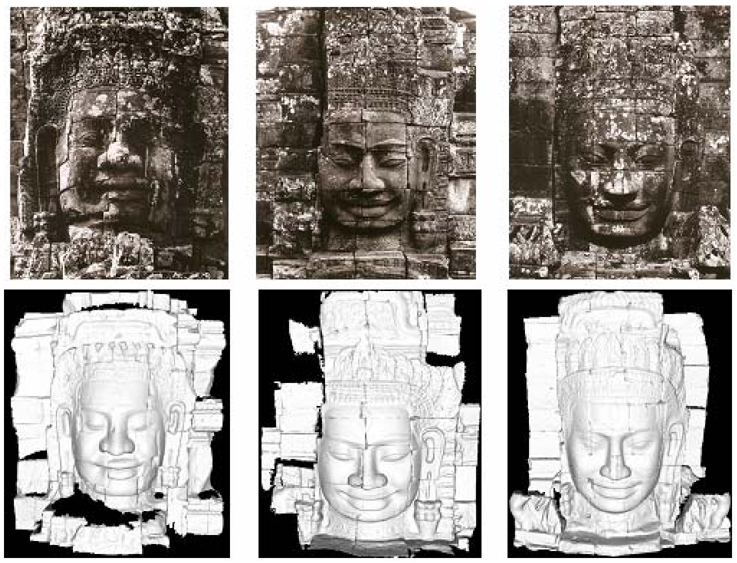


Figure 11. Ashura, Dava, and Devata. The black parts in the figures indicate areas where data is missing due to occlusion.

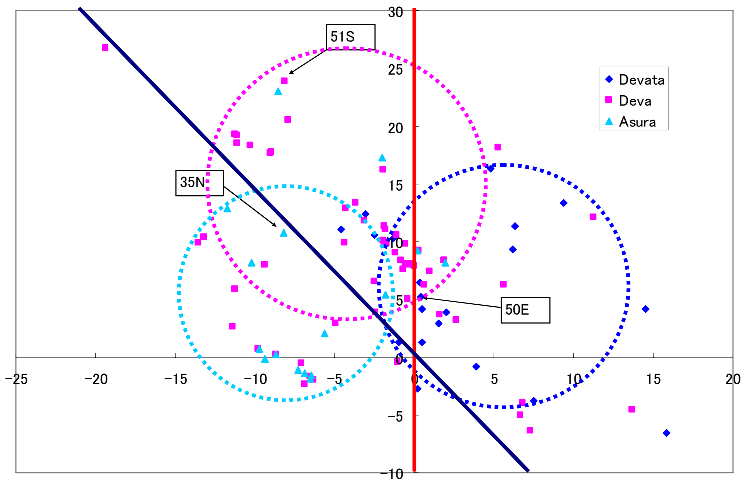


Figure 12. Linear discrimination function

Cluster analysis provides us with some classification of samples according to distances among them. We calculate the distance based on the Ward method, and define the distance as the Euclidean distance in the distance image space. We employ agglomerative hierarchical cluster analysis. This analysis begins with each sample being considered as one cluster and then proceeds to combine the nearest two clusters until all samples belong to one cluster. By conducting this cluster analysis, we observed spatial resemblance groups among faces, as shown in Fig. 13. We determined that there were four or five independent worker groups working on these faces in a parallel manner.

3. Photometric modeling: Texturing

Surface color distribution is important in representing the appearance of cultural heritage objects. Some clay statues still retain the surface colors that were painted at the time the statue was originally made. For such types of cultural properties, we have to archive color information as well as geometric information. For this purpose, we have developed two kinds of texturing methods: calibration-based and reflectance edge-based methods.

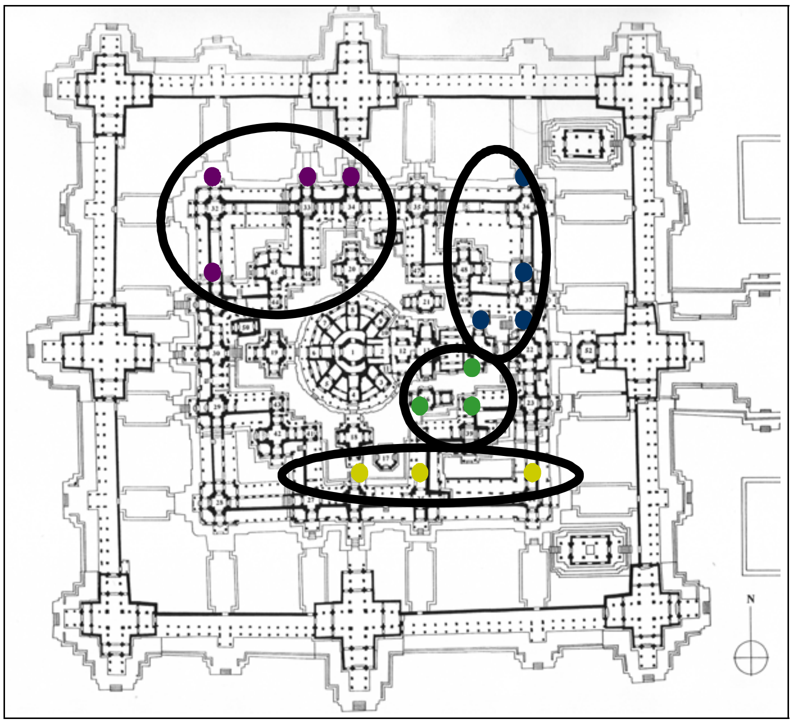


Figure 13. Similarity group.

3.1. Calibration-based Method

The main issue in texturing is how to determine the relationship between image sensors and geometrical sensors. When short-distance range sensors can be used, as shown in Fig. 14, the most promising method is to calibrate the geometrical relationship between the image sensor and the range sensor before scanning.

Assume that the coordinate system of the image sensor is (x_c, y_c) and the corresponding point in the range image is (X,Y,Z); the relationship between them can be described as:

$$\begin{bmatrix} hx_c \\ hy_c \\ h \end{bmatrix} = c_{34} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{17}$$

$$= \begin{bmatrix} k_u & -k_u \cot\theta & u_0 \\ 0 & k_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{33} & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The matrix C_{34} represents the relationship between the image and the world coordinate, and it can be calculated by scanning the calibration box. Inversely, when we map the texture images onto the geometrical triangular mesh $\overrightarrow{X_n} = \{(x_n, y_n, z_n \mid 1 \le n \le 3\}$, the corresponding points in image coordinate $x_c = \{x_c, y_c\}$ can be easily calculated as:

$$x_c = \frac{c_{11}X_n + c_{12}Y_n + c_{13}Z_n + c_{14}}{c_{31}X_n + c_{32}Y_n + c_{33}Z_n + c_{34}},$$

$$y_c = \frac{c_{21}X_n + c_{22}Y_n + c_{23}Z_n + c_{24}}{c_{31}X_n + c_{32}Y_n + c_{33}Z_n + c_{34}} \tag{18}$$

For the modeling of the Koumoku-Ten clay figure, we used 60 range images and color images that were taken at the same time. Figure 15 shows a picture of Komoku-Ten (a), the geometric model (b), and the synthesis result under a new lighting condition generated using the texturing result (c).

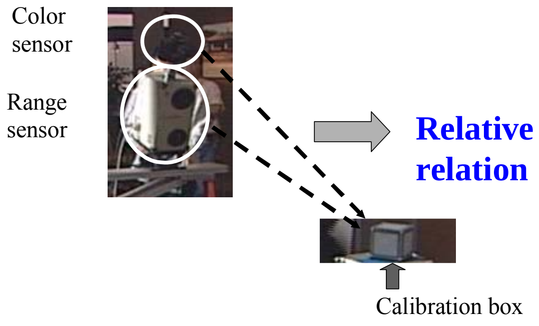


Figure 14. Calibration method.

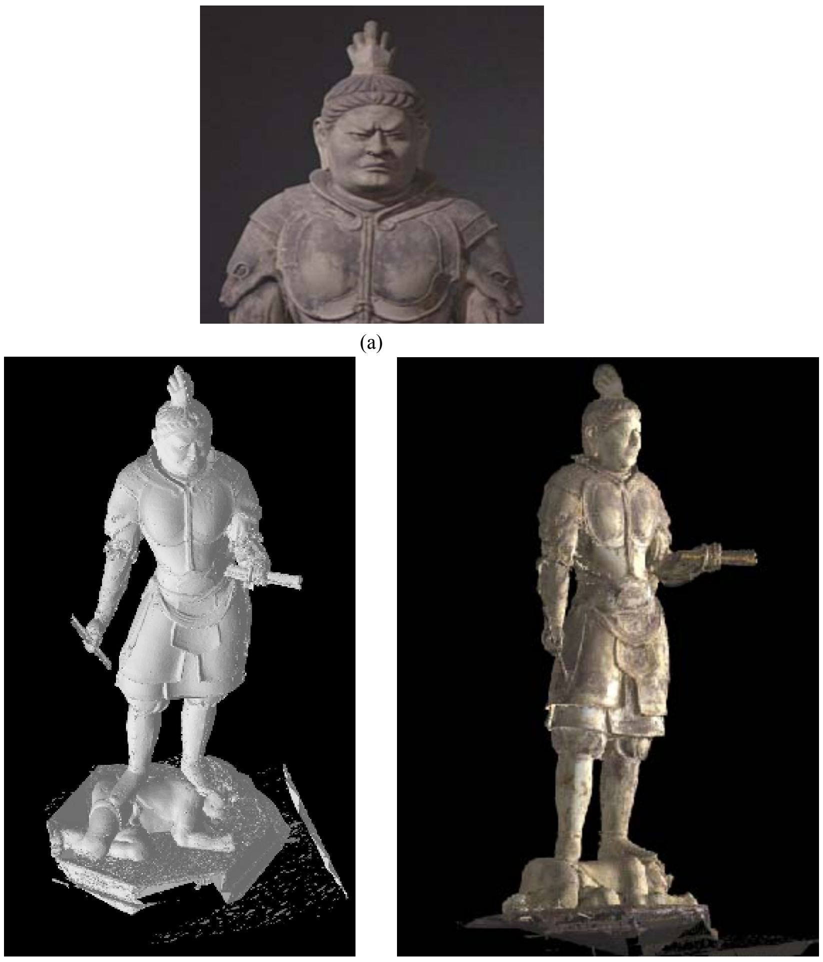


Figure 15. Synthesized Komoku ten. (a) Picture of Komoku Ten, (b) 3D model, (c) Synthesized result under a new lighting condition generated using the texturing result.

3.2. Reflectance Edge-based Method

As shown in the previous section, one solution for determining the relationship between range and color image is through calibration using a calibration fixture. However, this method requires that the range and color sensors be fixed on the fixture once the relationship is calibrated. Further, the calibration-based method is accurate only around the position occupied by the calibration fixture. When a target object is very large, this method becomes unreliable due to the lens distortion. Thus, we need a method that does not rely on calibration for handling a large object.

Generally speaking, range sensors often provide reflectance images as side products of range images. The returned timing provides a depth measurement, while the returned strength provides a reflectance measurement. A reflectance image is a collection of the strength of returned laser energy at each pixel. This reflectance image is aligned with the range image because both images are obtained through the same optical receiving device. Commonly available range sensors, including ERIM, Preceptron, and our main sensor, CYRAX, provide this reflectance image.

We employ this reflectance image as a vehicle for the alignment of range images with color images [9,26]. Reflectance images share characteristics similar to color images due to the fact that both images are somehow related with surface roughness, as shown in Fig. 16. Since our CYRAX range scanner uses a green laser diode, reflectance edges can be observed along material boundaries between two different reflectance ratios for this wavelength. Since different materials are of different colors, a discontinuity also appears in the color images. Jump edges along small ranges in a range image also appear as jump edges in a reflectance image as well as in a color image. Occluding boundaries are observed both in reflectance images and in color images.

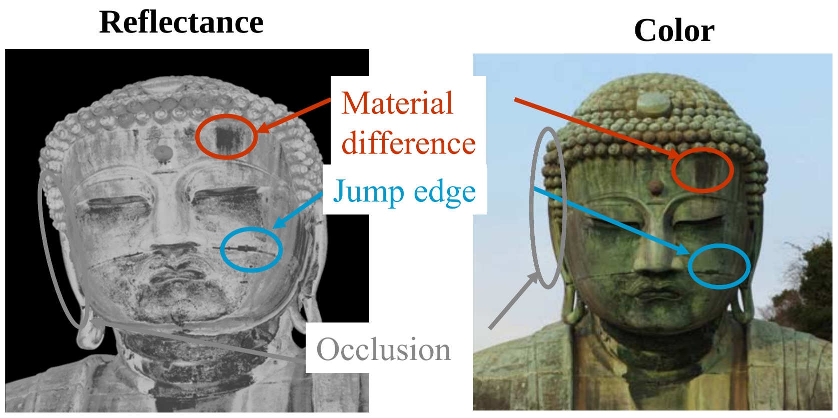


Figure 16. Reflectance and color images.

Prior to the alignments, we paste the necessary reflectance edges onto the 3D geometric model. As mentioned above, since occluding boundaries vary depending on the viewing direction, edges along the occluding boundaries are first removed from the reflectance images. Edges along the current occluding boundaries will be estimated from the 3D geometric model and the current viewing direction. Our algorithm extracts them automatically, and uses them for alignment.

We align edges extracted from reflectance images with those in color images so that the 3D position error of those edges is minimized by iterative calculation as shown in Fig. 17. Extracted edges are represented as a collection of points along them. The alignment is done between 3D reflectance points on a 3D geometric model projected on the image plane and 2D color edge points in the 2D image.

To establish correspondence, the system finds the color image points that are nearest to the projected reflectance points. This operation is similar to the ICP operation.

To determine the relative pose that coincides with the position of 2D color edges and projected 3D reflectance edges, we use the M-estimator.

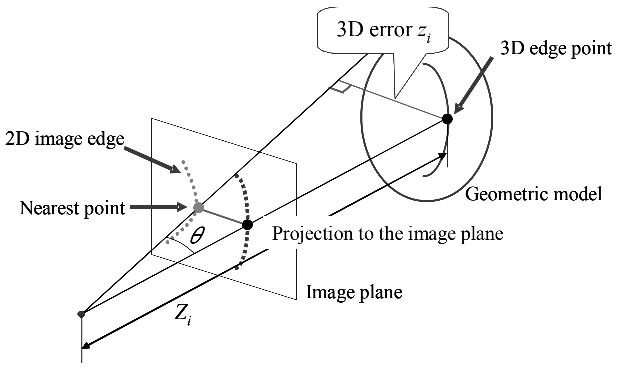


Figure 17. Texturing algorithm.

First, the distance between corresponding 2D color edge points and 3D reflectance edge points is evaluated as shown in Fig. 17, where z_i is a 3D error vector that is on a perpendicular line from a 3D reflectance edge point to the stretched line between the optical center and a 2D color edge point on the image plane.

$$\varepsilon_i = Z_i \sin\theta \tag{19}$$

where Z_i is the distance between the optical center and a 3D reflectance edge point, and θ is the angle between the color edge point and the reflectance edge point.

The system finds the configuration, P, which minimizes the total error, E, where ρ is an error function. The minimum of $E(p)$ can be obtained by

$$\frac{\partial E}{\partial P} = \sum_i \frac{\partial \rho(\varepsilon_i)}{\partial \varepsilon_i}\frac{\partial \varepsilon_i}{\partial P} = 0 \tag{20}$$

We can consider $\omega(\varepsilon)$ as a weight function to evaluate error terms.

$$\omega(\varepsilon) = \frac{1}{\varepsilon}\frac{\partial \rho}{\partial \varepsilon} \tag{21}$$

By substituting Eq(21) into (20) we obtain

$$\frac{\partial E}{\partial \rho} = \sum_i \omega(\varepsilon_i)\varepsilon_i \frac{\partial \varepsilon_i}{\partial P} = 0 \tag{22}$$

We choose the Lorentzian function for this function.

$$\omega(\varepsilon) = \left(1 + \frac{1}{2}\left(\frac{\varepsilon}{\sigma}\right)^2\right)^{-1} \tag{23}$$

By solving this equation using the conjugate gradient method, we can obtain the configuration P that minimizes the error term and gives the relative relationship between the camera and the range sensor. Figure 18 shows the texture-mapped Kamakura Buddha. Since this method minimizes a non-linear equation, we need an initial alignment. The initial alignment is given manually using our GUI. For the current implementation, relatively accurate alignment is necessary for rotation, but it is not the case for translation.

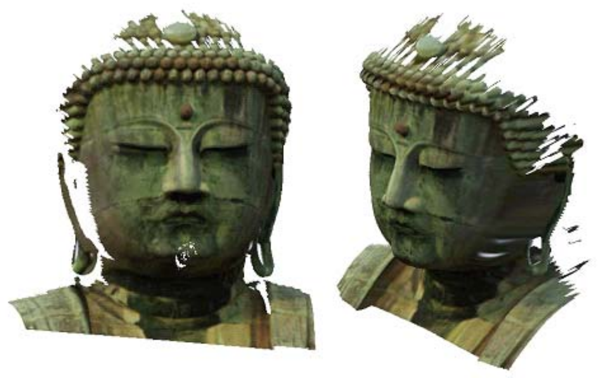


Figure 18. Texturing result.

4. Restoring the Original Nara Great Buddha

One of the advantages of obtaining digital data of cultural heritage objects is to modify those data and display the original appearance of the object. In order to demonstrate this ability, after we obtain the precise geometric and photometric information about cultural heritage objects in their current state, we can modify the current data into a hypothesized original state. In this section, we describe one of the examples: the restoration of the Nara Great Buddha and its main hall.

The Nara Great Buddha is one of the most important heritage objects in Japan. The Buddha statue is sitting in the Buddha palace at the Toudaiji temple in Nara, Japan. Originally, this temple and statue were constructed by order of the Shomu emperor, in the 8th century. Here, the original one is referred to as "Tempyou Big Buddha." The original Tempyou Buddha was made of bronze and covered with gold plate. Unfortunately, however, the palace was burned and the statue was melted twice due to civil war in Japan. The current Buddha and palace were rebuilt in the 17th and 18th centuries. Accordingly, the shape of the current Great Buddha is different from that of the original one in the 8th century.

4.1. Restoring the Nara Great Buddha

As the first step, we acquired the complete 3D mesh model of the Nara Great Buddha in its current state by using the geometrical modeling techniques described in Section 2. We collected more than 100 partial mesh models using CYRAX sensors. Those partial mesh models were aligned using the parallel alignment algorithm on a PC cluster and merged into a unified mesh model with 70M polygon. Figure 19 shows the picture of the current Buddha, and its 3D geometric model.

We synthesized the original state by morphing the 3D mesh of the model from this mesh model. From some literature inherited at various temples, we knew the sizes of various face parts such as the nose and mouth. "Enryaku-so-rokubun," "Daibutsuden-hibun," "Hichidaiji-nikki," and "Gokokuji-honnsyoji-enngishu" are representative documents that contain those sizes. Unfortunately, however, those numbers often contradict each other. Some researchers investigated which number is the most reliable one. We followed their method to compare them and determined a common figure for each part.

Table 2 shows the obtained estimated and the current dimensions of various face parts. Here, all the documents employ the unit called "shaku." We interpreted shaku as the tempyo shaku, and one shaku is assumed to be 0.2964 meters among the various interpretations of shaku. Notice that relatively large differences exist in height measurements.

Using these data, we designed a two-step morphing algorithm. First, we globally changed the scale of the whole portions (for example, Height when sitting, Face Length, Nose Length); these are gradually modified. In the second stage, vertices were moved one by one iteratively, similarly to the constraint propagation algorithm, using smoothness and uniform constraints. The two-stage morphing enabled us to obtain the complete model of the original Great Buddha. Figure 20 shows the 3D models of the current (a) and the original Great Buddha (Tempyou Buddha) (b). We can easily recognize that the original Buddha is larger and rather thin.

4.2. Restoring the Toudaiji Main Hall

The main hall of the Toudaiji Temple was built during the same decades as those of the Great Buddha (8th century). It was also rebuilt twice: in the 12th and 17th centuries. In the 12th century, Tenjiku architecture was imported from China, and the main hall was rebuilt in a totally different architecture style. The rebuilding in the 18th century followed the same new style. As a result, the style of the current main hall is entirely different from that of the original building.

Table 2. Current and estimated dimensions of various face parts.

Parts Name	Current (m)	Original (m)
Height when sitting	14.98	15.85
Eye length	1.02	1.16
Face length	3.20	2.82
Ear length	2.54	2.52
Palm length	1.48	1.66
Foot length	3.74	3.56
Nose height	0.50	0.47
Mouth length	1.33	1.10

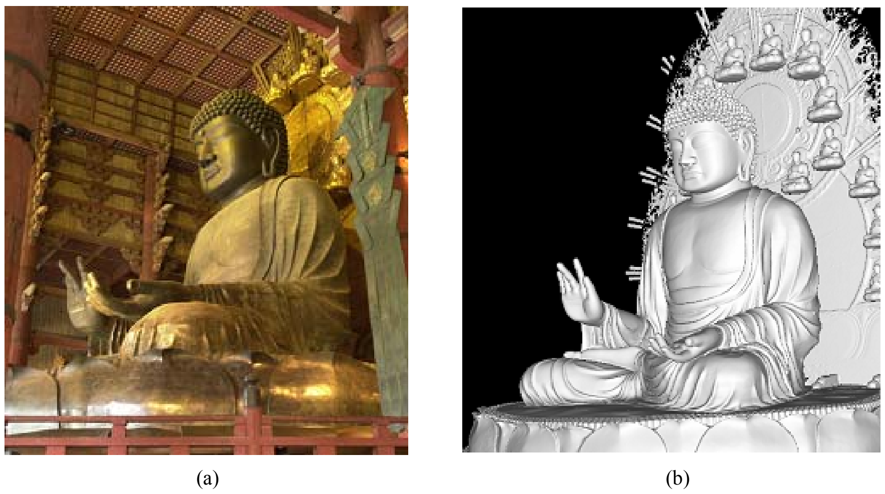


Figure 19. Nara Buddha (a) Picture of current Buddha, (b) 3D geometric model of current Buddha.

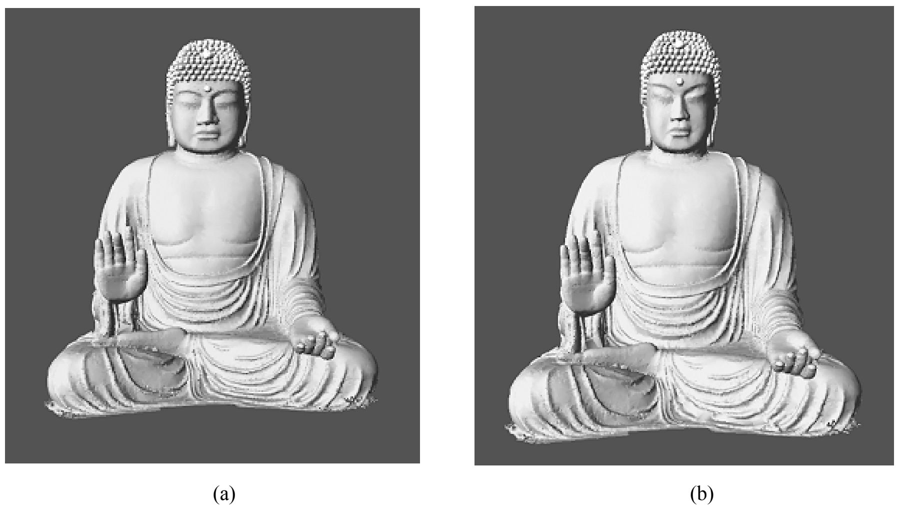


Figure 20. Comparison in 3D models. (a) current buddha, (b) original Buddha.

Fortunately, the Toudaiji temple has been displaying a miniature model of the original hall, constructed for the Paris Expo in 1900, as shown in Fig. 21. We digitized it using the Pulsteck TDS-1500 and scaled it up to the original size as shown in Fig. 22(a). The TDS-1500 can scan a range from 3.5 meters through 10 meters with the accuracy of 0.5mm to 5mm and the spatial resolution of 420 X 280. We obtained 12 range images from various observation directions. As shown in Fig. 22(b), due to the limits of the sensor's accuracy and constraints of observation directions, though the model provides rough dimensions of locations of columns and walls, it does not provide a precise and accurate picture of the detailed parts.

According to Prof. Keisuke Fujii, who is an architecture professor at the University of Tokyo and one of the experts on building style in the era, the Toudaiji and Toushoudaiji temples share a similar format. The main hall of the Toushoudaiji Temple was also built during the same period (8th century). We have decided to combine the detailed part model of the Toushoudaiji Temple with the rough whole model of the Todaiji temple.

We digitized various key parts of the main hall at Toushoudaiji. Using the suggestions of Prof. Fujii, we chose 20 important parts of the main hall. Figure 23 shows 6 parts among 20 important parts. We employed Cyrax 2004 and Pulsteck TDS-1500, which have a range from 0.5 meter through 1meter, with resolution of 0.23 mm through 0.83 mm, to obtain 780 range images. Figure 24 shows the obtained range images of the detailed parts.

We pasted these partial range data of Touhoudaiji parts (Fig. 24) to the scaled-up range data of the Toudaiji (Fig. 22(b)), using as a scale the average size difference between those temples, roughly 1 to 2.3. But each part

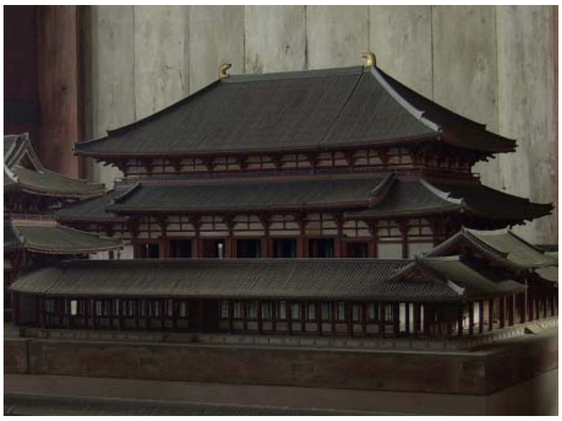
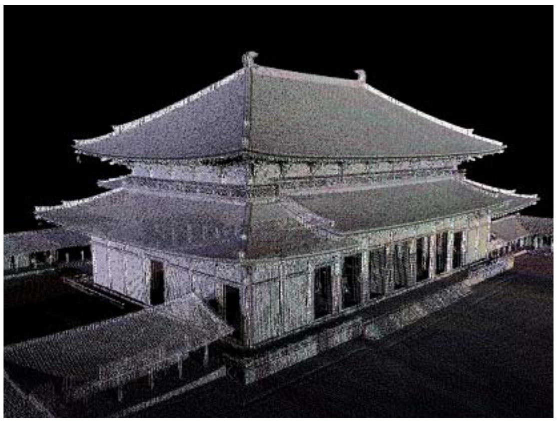
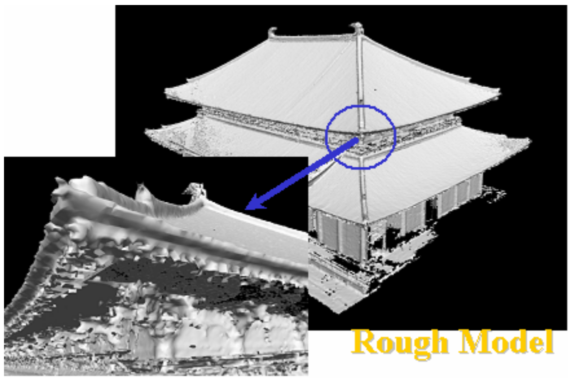


Figure 21. Miniature model of Buddha palace.



(a) a cloud of points representation



(b) mesh model (Closed up)

Figure 22. 3D model acquired from the Miniature model.

needed more precise scaling parameters. The traditional alignment algorithm determined translation and rotation parameters as six unknown parameters. However, we designed an extended alignment algorithm that determined not only translation and rotation parameters but also scale parameters.

$$\varepsilon^2 = \min_{R,t} \sum_{s \neq m,k} \left(R_m \vec{n} \cdot \{ (R_s \overrightarrow{K}(\vec{y}, k_s) + \vec{t}_s) - (R_m \vec{x} + \vec{t}_m) \} \right)^2 \qquad (24)$$

where $K(,)$ is a scaling function to expand each arc length in a mesh, and k_s is an unknown scaling parameter. By using this extended alignment algorithm, we completed the 3D model of the original Buddha's palace.

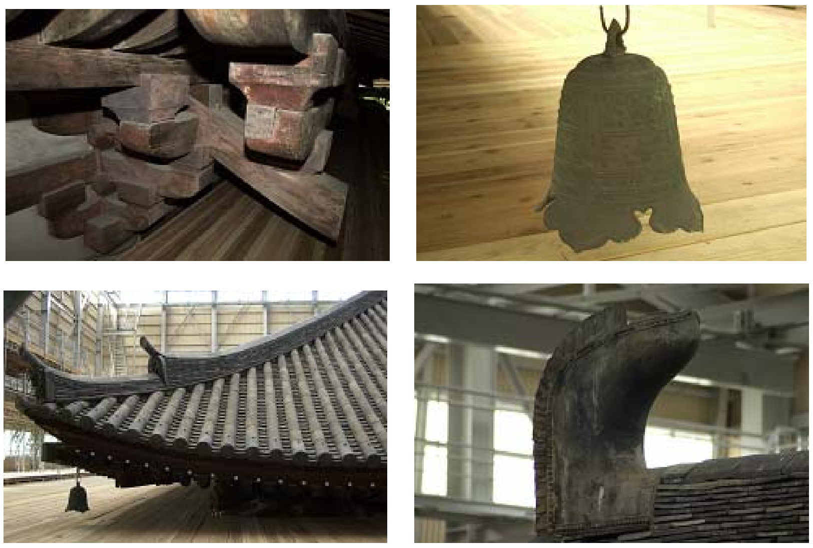


Figure 23. Key parts of the main hall at Tousho-daiji digitized.

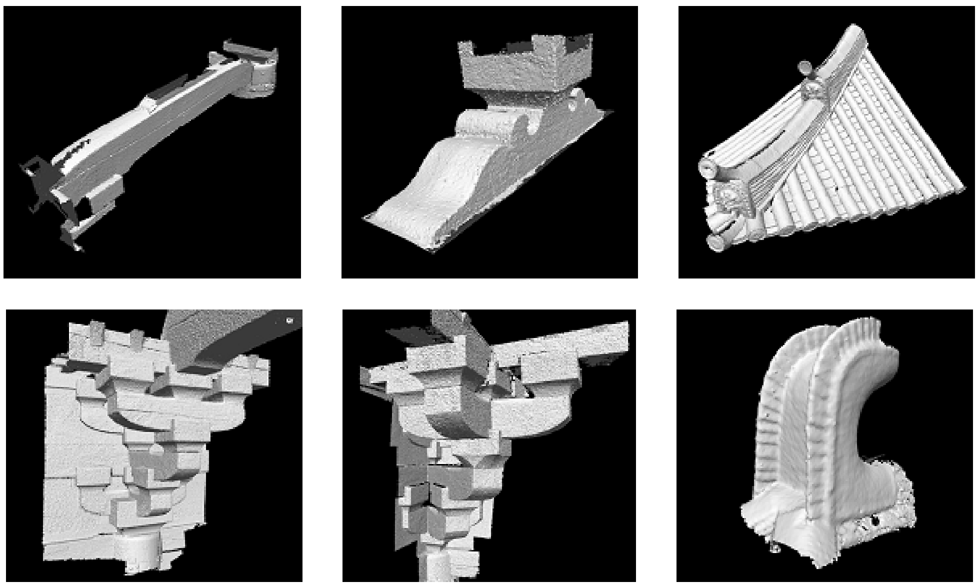
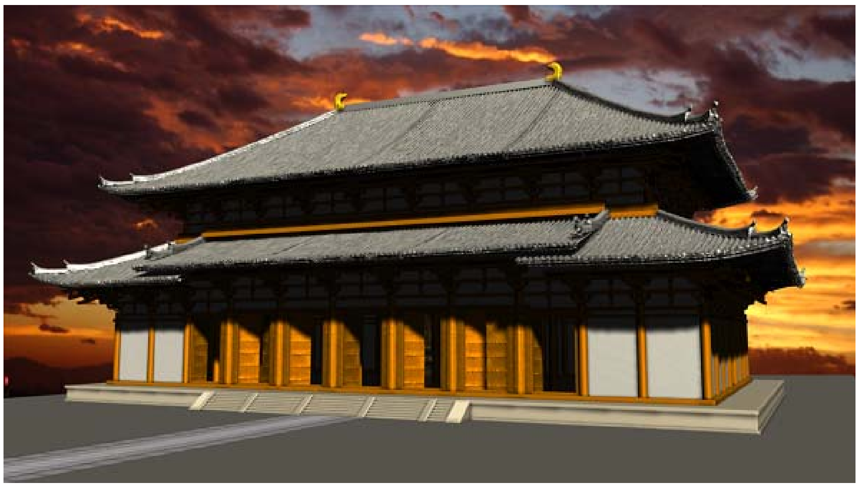


Figure 24. 3D models of those key parts.

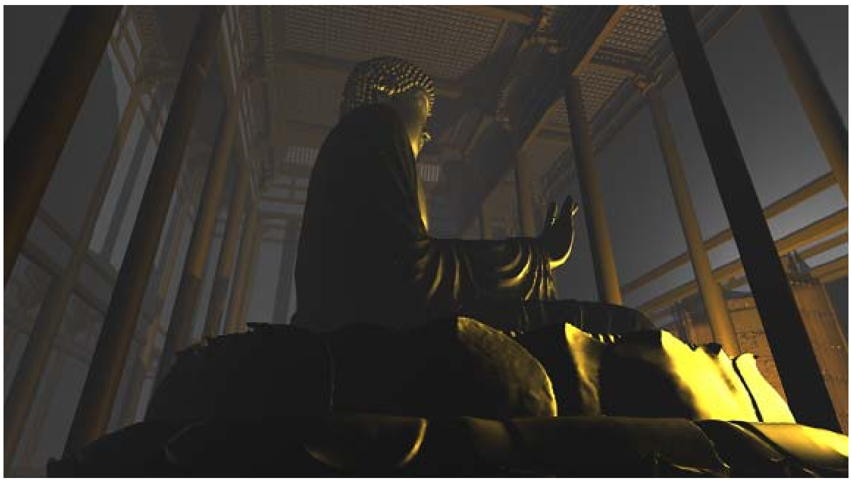
Figure 25(a) shows the original Buddha's palace digitally restored by our method. By combining the original Buddha's palace and the original Buddha, we created the virtual appearance of the Nara Buddha in the 8th century, as shown in Fig. 25(b-c). The virtual appearance of this and other historic objects can be used for education about and promotion of our cultural heritage.

4.3. Analysis

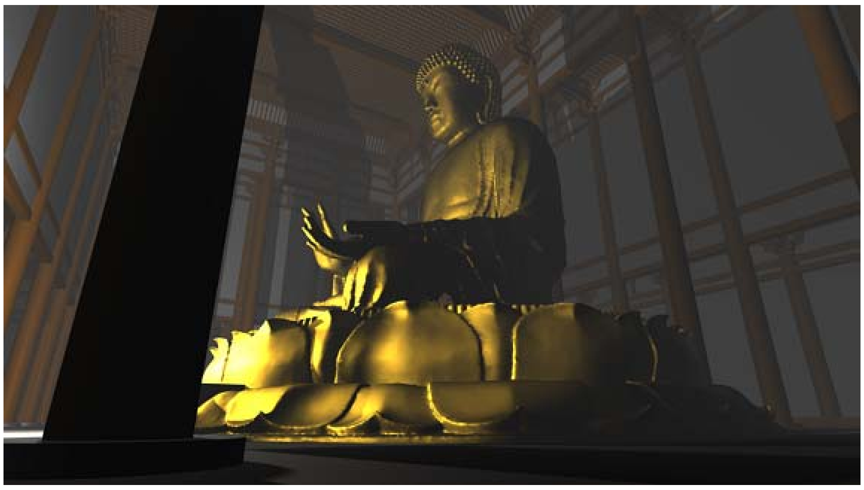
As one of the demonstrations of utilizing digital restoration, we conducted an experiment to determine the amount of gold used to plate the surface of the Buddha. It is well known that the original Buddha was golden due to gold plating of its surface. However, several contradictory numbers exist in documents. For example, "Daibutu-den-hibun" and "Enryaku-sorokubun" say it required 5412 ryou and 4187 ryou of gold, respectively, to cover the body of the great Buddha. Moreover, there were



(a) Outside appearance



(b) Insider appearance



(c) Inside appearance

Figure 25. Restored nara Buddha.

Table 3. Four interpretations determining the amount of gold used.

Document name	Written amount in the document	Interpretation	
		Large ryou (42g)	Small ryou (14g)
Daibutu-den-hibun	5412 ryou	227 kg	76 kg
Enryaku-sorokubun	4187 ryou	176 kg	59 kg

two interpretations of “ryou”; A large ryou was 42 g, while a small ryou was 14 g. Thus, there are four interpretations determining the amount of gold required.

In order to disambiguate this discussion, we used our restored digital model of the Tempyou great Buddha. The surface area, $597\,m^2$, is obtained from the restored digital model by taking a summation of all surface areas of triangular meshes. For comparison, the surface area of the current Buddha is $556\,m^2$. From the documents, it is known that the amalgam method was used to put gold over the Buddha’s surface. Usually, this method puts $6 \sim 10\ mg/cm^2$. This number was also confirmed by examining the thickness of gold plate on various treasures stored in Sho-so-in. By multiplying the surface area of Tempyo and the current Buddha with this number, we obtained the gold amount as 36kg ~60kg and 33kg ~ 56kg. Those numbers indicate that the interpretation of “enryaku-sorokubun” with a small ryou is most likely.

5. Conclusion

This paper introduced our project to digitally archive and restore cultural heritage objects. Our project’s main goal was to develop software to create VR models of heritage objects through observation of real heritage objects. As the input sensor of geometrical information, we used laser range sensors because of their accuracy. Since only partial mesh models of an object are obtained from such sensors, we have developed post-process algorithms. These included a rapid alignment algorithm based on graphics hardware, a parallel alignment algorithm based on a PC cluster, and a parallel merging algorithm based on a PC cluster. For texturing color information onto geometric modes, we developed a non-calibrated texturing method based on laser reflectance features.

Digital restoration of lost cultural heritage objects has a great advantage compared with other restoration methods such as physical construction of actual temples, because we can examine various hypotheses without any physical changes or long building periods. We demonstrated the effectiveness of this method through the restoration of the Nara Great Buddha and its main hall.

Acknowledgments

This research is sponsored, in part, by JST under Ikeuchi Crest program, and, in part, by Ministry of Education under Leading Project. The Bayon temple in Cambodia was digitized with the cooperation of the Japanese Government Team for Safeguarding Angkor (JSA).

References

[1] T. Kanade, P. Rander, and P.J. Narayanann, “Virtualized Reality: Constructing Virtual Worlds from Real Scenes,” *IEEE Multimedia*, 4(2):34-47, Januray 1997.

[2] K.N. Kutulakos and S.M. Seitz, “A Theory of Shape by Space Carving,” *Intern. Journal Computer Vision*, 38(3): 199-218, July 2000.

[3] E. Borovik and L. Davis, “A Distributed System for Real-time Volume Reconstruction,” *Proc. Of CAMP2000*, pp.183-189, 2000.

[4] I. Kitahara and Y. Ohta, “Scalable 3D Representation for 3D Video in a Large-Scale Space,” *Presence*, 13(2): 164-177, 2004

[5] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara, Real-Time 3D Shape Reconstruction, Dynamic 3D Mesh Deformation, and High Fidelity Texture Mapping for 3D Video,” *IEEE Trans. Circuits and Systems for Video Technology*, Vol. CSVT-14(3): 357-369, March 2004.

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

[7] M. Levoy et. al., “The digital Michelangelo project,” *SIGGRAPH 2000*, pp.131-144, New Orleans.

[8] J. Wasserman, *Michelangelo’s Florence Pieta*, Princeton University Press 2003.

[9] I. Stamos and P. Allen, “Automatic registration of 2-D with 3-D imagery in urban environments,” *ICCV2001*, pp.731-737, Vancouver.

[10] P.J. Besl and N.D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Patt. Anal. Machine Intell.,* 14(2):239-256, 1992.

[11] R. Benjemma and F. Schmitt, “Fast global registration of 3D shample surfaces using a multiple-z-buffer technique,” *Int. Conf on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 113-120, May 1997.

[12] P. Neugebauer, “Geometrical cloning of 3D objects via simultaneous registration of multiple range images,” *Int. Conf on Shape Modeling and Application*, pp.130-139, March 1997.

[13] Y. Chen and G. Medioni, “Object modeling by registeration of multiple range images, *Image and Vision Computing*, 10(3):145-155, April 1992.

[14] S. Rusinkiewicz and M. Levoy, “Efficient variants of the IPC algorithm,” *Int. Conf 3-D Digital Imaging and Modeling,* pp.145-152, May 2001.

[15] H. Gagnon, M. Soucy, R. Bergevin, and D. Laurendeau, “Registeration of multiple range views for automatic 3-D model building,” *CVPR94*, pp.581-586.

[16] K. Nishino and K. Ikeuchi, "Robust Simultaneous Registration of Multiple Range Images", *Fifth Asian Conference on Computer Vision ACCV '02*, pp454-461, 2002.

[17] T. Oishi, R. Sagawa, A. Nakazawa, R. Kurazume, and K. Ikeuchi, “Parallel Alignment of a Large Number of Range Images on PC Cluster,” *Int. Conf 3-D Digital Imaging and Modeling,* pp. 195-202, Oct 2003.

[18] M. D. Wheeler and K. Ikeuchi, "Sensor Modeling, Probablistic Hypothesis Generation, and Robust Localization for ObjectRecognition", *IEEE PAMI*, 17(3): 252-265, 1995.

[19] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” *SIGGRAPH 96*, pp.303-312, New Orleans, LA.

[20] M. Wheeler, Y. Sato, and K. Ikeuchi, “Consensus surfaces for modeling 3D object from multiple range images,” *ICCV98,* pp.917-924.

[21] R. Sagawa, K. Nishino, M.D. Wheeler and K. Ikeuchi, "Parallel Processing of Range Data Merging", *IEEE/RSJ International Conference on Intelligent Robots and Systems,* Vol. 1, pp577-583, 2001

[22] R. Sagawa, T. Masuda, and K. Ikeuchi, “Effective Nearest Neighbor Search for Aligning and Merging Range Images,” *Int. Conf 3-D Digital Imaging and Modeling,* pp. 79-86, Oct 2003

[23] R. Sagawa and K. Ikeuchi, “Taking Consensus of Signed Distance Field for Complementing Unobservable Surface,” *Int. Conf 3-D Digital Imaging and Modeling,* pp. 410-417, Oct 2003

[24] R. Sagawa and K. Ikeuchi, “Adaptively Merging Large-Scale Range Data with Reflectance Properties,” *IEEE Trans. Patt. Anal. Machine Intell.*, 27(3): 392-405, March 2005.

[25] M. Kamakura, T. Oishi, J. Takamatsu, and K. Ikeuchi, “Classification of Bayon faces using 3D models,” *Virtual Systems and Multimedia,* pp.751-760, October 2005.

[26] R. Kurazume, K. Nishino, Z. Zhang, and K. Ikeuchi, "Simultaneous 2D images and 3D geometric model registration for texture mapping utilizing reflectance attribute," *Fifth Asian Conference on Computer Vision*, Vol. 1, pp. 99-106, 2002.

[27] http://www.cvl.iis.u-tokyo.ac.jp