The Great Buddha Project: Modeling Cultural Heritage for VR Systems through Observation

Katsushi Ikeuchi, Atsushi Nakazawa, Kazuhide Hasegawa, Takeshi Ohishi
Institute of Industrial Science, The University of Tokyo
{ki, nakazawa, k-hase, oishi}@cvl.iis.u-tokyo.ac.jp

Abstract

This paper overviews our research on digital preservation of cultural assets and digital restoration of their original appearance. Geometric models are digitally achieved through a pipeline consisting of scanning, registering and merging multiple range images. We have developed a robust simultaneous registration method and an efficient and robust voxel-based integration method. On the geometric models created, we have to align texture images acquired from a color camera. We have developed two texture mapping methods. In an attempt to restore the original appearance of historical heritage objects, we have synthesized several buildings and statues using scanned data and literature survey with advice from experts.

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 are quite vulnerable to fires and other natural disasters because most of them were constructed of wood and paper.

One of the best ways to prevent them from loss and deterioration is to digitally preserve them. Digital data of heritage objects can be obtained by using 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 is suitable for many applications, including simulation, restoration, and creating multi-media contents. Such digital contents can be viewed through the internet from anywhere in the world, without moving the objects nor visiting the sites.

We have been working to develop such digital archival methods by using computer vision and computer graphics technologies [1]. Similar projects include: Stanford's Michelangelo Project [2], IBM's Pieta Project [3], and Columbia's project [4], to name a few. Our project has a number of unique features; among them is its ability to digitize relatively large objects outdoor such as the Kamakura great Buddha, and Cambodia's Bayon. This presents several challenges. Also, our project consists not only of geometric modeling but also of photometric and environmental modeling, as shown in Figure 1.

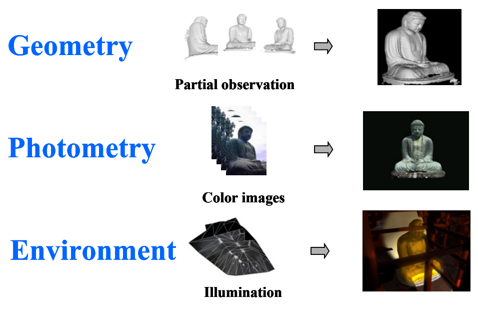


Figure 1. Three components of our project.

The remainder of this paper is organized as follows. Section 2 describes the outline of the geometric pipeline developed, a parallel simultaneous alignment algorithm and a parallel voxel-based merging algorithm. Section 3 describes methods to align observed textures from a digital camera with a range data for texture mapping. Section 4 reports our efforts to restore the original appearance of these objects using acquired digital data and the literature survey. Section 5 summarizes this paper.

2. Geometric Modeling

Several computer vision techniques, such as traditional shape-from-X and binocular stereo, or modern range sensors, provide cloud of point information. The cloud of point information certainly carries three-dimensional information pertaining to observed objects. However, there is no structural information among these points. Namely, there is no information to represent adjacency among the points. The first step of geometric modeling is to convert the cloud of points into a surface representation such as a mesh model.

Since each observation provides only partial information, we have to combine these partial mesh representations into a whole geometric mesh representation. Thus, the second step in geometric modeling is to align these meshes so that their corresponding parts overlap one another (alignment). To accomplish this step, we have developed a simultaneous alignment method to avoid accumulation of errors [3].

The third geometric modeling step is to integrate the pre-registered multiple range images and to compose one complete geometric model, a step usually called 'merging'. The procedure can be considered as extracting one surface from multiple overlapped surfaces. In the merging procedure, it is important to make the integration framework robust against any noise which may be in the scanned range images and can also be inherited from the registration procedure.

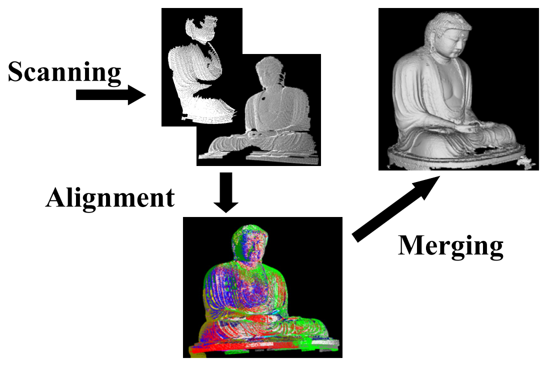


Figure 2. Three steps of Geometric Modeling.

2.1 Simultaneous Registration Method

To avoid accumulation of errors, we have developed a simultaneous alignment method [11]. Traditional sequential methods [5,8,9] such as Iterative Closest Point Algorithm (ICP) align these meshes one by one, and progressively align a new partial mesh with previously aligned meshes. If a few partial meshes can cover an object, the accumulation of alignment errors is relatively small and can be ignored. A sequential alignment works well. However, some cultural heritage objects are very large; we need more than one hundred views. In such cases, the error accumulation caused by sequential alignment methods is very large. Thus, by considering alignment of all the pairs simultaneously, we align all partial meshes so as to reduce the errors among all the pairs simultaneously [7,10,11]. Our simultaneous algorithm [11] employs points and planes to evaluate relative distance as does the Chen and Medioni and Gagnon et.al. methods [8,10]. Further, to avoid the effects from the outliers, the algorithm uses M-estimator, in particular, the Lorentian function as the error measure [13].

Currently, we are extending the simultaneous alignment algorithm into a a parallel implementation so as to be able to align a very large data set [12]. The fundamental algorithm, due to the utilization of a graphics hardware, is slightly different from the one described above [11].

The corresponding pairs are searched along the line of sight to use the graphics hardware. Assume that a base range image is the model image and that a corresponding range image is the scene image. In Figure 4, a model image is depicted as points and a scene image is shown as meshes. When a line extended from a vertex of model image along the line of sight crosses a mesh of a scene image, the intersecting point is the corresponding point. In order to eliminate false correspondences, if the distance between corresponding points is larger than a certain threshold value or if the angle of lines of sight is beyond a threshold angle, the correspondence is removed. This correspondence search is computed for every combination of range images, using the Z-buffer capability of the graphics hardware.

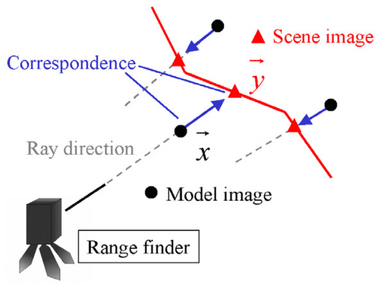


Figure 3. Search corresponding points.

The error measure between corresponding points is the distance between the point and the plane. Let the vertex of model image be $\vec{x}$ and let the point corresponding to the vertex be $\vec{y}$, the error measure between the pairs is written as

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

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

The transformation matrices of the model and scene images 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 image are R_M, R_s, $\vec{t}_M, \vec{t}_s$, respectively. Then, the distance between the model image and the scene image is expressed as

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

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

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

The translation vector is expressed as

$$t = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix} \quad (5)$$

After some algebraic manipulations [16], (3) is rewritten as

$$\varepsilon^2 = \min_{\vec{\delta}} \sum_{i \neq j,k} \left\| A_{ijk}\vec{\delta} - s_{ijk} \right\|^2 \quad (6)$$

$$s_{ijk} = \vec{n}_{ik} \cdot (\vec{x}_{ik} - \overrightarrow{y_{ijk}}) \quad (7)$$

$$A_{ijk} = \begin{pmatrix} \underbrace{0 \ldots 0}_{6i\times1} \underbrace{C_{ijk}}_{6\times1} \underbrace{0 \ldots 0}_{6(l-i-1)\times1} \end{pmatrix} + \begin{pmatrix} \underbrace{0 \ldots 0}_{6j\times1} \underbrace{-C_{ijk}}_{6\times1} \underbrace{0 \ldots 0}_{6(l-j-1)\times1} \end{pmatrix} \quad (8)$$

$$C_{ijk} = \begin{pmatrix} \overrightarrow{n_{ik}} \times \overrightarrow{y_{ijk}} \\ -\overrightarrow{n_{ik}} \end{pmatrix} \quad (9)$$

$$\vec{\delta} = \begin{pmatrix} m_0 \cdots m_{n-1} \end{pmatrix} \quad (10)$$

$$m_i = \begin{pmatrix} c_{1i} & c_{2i} & c_{3i} & t_{xi} & t_{yi} & t_{zi} \end{pmatrix} \quad (11)$$

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

$$\vec{\delta} = \left(\sum_{i \neq j,k} A^T{}_{ijk} A_{ijk} \right)^{-1} \sum_{i \neq j,k} A^T{}_{ijk} s_{ijk} \quad (12)$$

For the parallel implementation of the alignment algorithm, both time and memory performance have to be considered. The simultaneous alignment algorithm, originally designed, requires all range images to be read into memory; even when the computation is distributed over a PC cluster, the amount of memory used on each PC is not reduced.

We will remove redundant or weak data dependency relations. Figure 4 shows overlapping data-dependency relations. Each node in the graph represents one range data and each arc represents an overlapping dependency relations among range data. In the left graph, all the data are overlapping each other. In order to compute alignment of one data, we have to read, into a PC memory, all the remaining range data. 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 can increase the efficiency in memory usage.

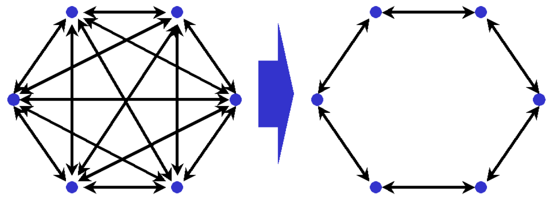


Figure 4. Removing redundant dependencies.

We can define and remove redundant dependencies in the range images. A pair of range images that does not satisfies all these three conditions will be removed as redundant relations.

1. The bounding-boxes of two range images overlap each other.
2. The angle θ between ray directions of two range images is less than a threshold value.
3. Two range images are adjecent to each other.

Under the assumption that initial positions of two range images are accurately estimated, Requirement 1 is satisfied when a sufficient overlapped region exists between two range images. Requirement 2 is satisfied when two views are relatively near. As the side effect, this requirement also reduces the possibility of false correspondences. Since our method utilizes the ray direction to establish correspondences, by setting the threshold, as $\theta = 90°$, we can avoid correspondence pairs between one and the other side. Requirement 3 is set up for removing non-adjacent relation sequentially. For example, as shown in Figure 5, if the length from I_0 to I_n is larger than the length from I_m to I_n $(l_{0m} < l_{0n})$, the arc between I_0 and I_n is removed. Here, the distance is evaluated from the center of a range image's surface.

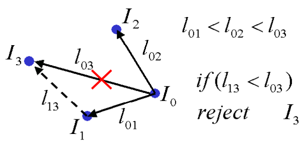


Figure 5. Definition of the distance measure.

For our parallel algorithm to read data only related, we can set up a relational table as shown in Figure 6. The row of the table represents scene images, and the column represents model images. Each element $d_{j,k}$ represents dependency between a scene, j and model image, k.

$$d_{j,k} = \begin{cases} 1 & j \text{ and } k \text{ have a dependency relation} \\ 0 & \text{otherwise} \end{cases}$$

Since each pair of a scene and model image can be evaluated independently in equation (6), roughly speaking, we will assign those pairs to nodes of a PC cluster so as for each node to have an average number of pairs. Here, we assign pairs to nodes based on scene images.

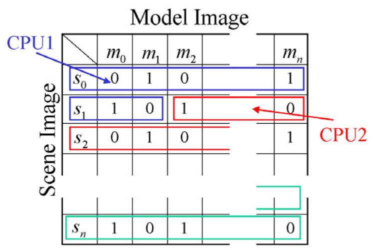


Figure 6. Pair Assignment table.

The amount of memory used depends on the number of vertices included in range images required for each node. By assuming the number of vertices of each range image is roughly equal, the amount of memory required, M_i for each node i is proportional to the number of range images assigned to the node r_i, the number of 1's in the assignment.

More precisely,

$$M_i = M_r \times r_i + M_c \qquad (13)$$

M_r is the average of the amount of data contained in each range images, and M_c is the amount of memory used for computation and independent to the number of range images (constant). The number of range images read into each node is expressed as union of sets of model image $r_{model,i}$ and scene image $r_{scene,i}$.

$$r_i = r_{model,i} \cup r_{scene,i} \qquad (14)$$

For the sake of computational efficiency described below, we will re-arrange the table. The range images, read into each node, are unions of model and scene images to be read in Eq (14). The assignments to the nodes are based on the scene images, as shown in Figure 8; it is preferable to read the same model images with respect to the scene images We define the distance $l_{i,j}$ between two scene images (i, j) as follows:

$$l_{i,j} = n - \sum_{0\le k\le n}\left(d_{i,k} \cup d_{j,k} - d_{i,k}(xor)d_{j,k}\right) \qquad (15)$$

The column of the table, the scene images, are sorted in the descending order of this distance under the assumption that the adjacent scene images are more likely to share the same set of model images.

2.1.5 Averaging computational time

Next, we will make even distribution of the computational load over a PC cluster. The computational cost of the I node, t_i is written as:

$$t_i = t_s \times \sum_{n_{0,i}\le j\le (n_{0,i}+n_i)} v_j + t_m \times \sum_{n_{0,i}\le j\le (n_{0,i}+n_i)} \sum_k \left(v_{j,k} \times d_{j,k}\right) + t_c \qquad (16)$$

where n_i scene images are assigned to node i from the number $n_{0,i}$ through $n_{0,j} + n_i$. Here, v is the number of vertices; t_s is the computation time for the number of vertices included in scene images, and t_m is computation time for the number of vertices included in model images. t_c is independent of the number of vertices, and because t_c is very small, t_c can be considered to be 0. Assuming that the computation time for the number of scene images and model images t_s and t_m is equal to t_v, (16) is rewritten as

$$t_i = t_v \times \left\{ \sum_{n_{0,i}\le j\le (n_{0,i}+n_i)} v_j + \sum_{n_{0,i}\le j\le (n_{0,i}+n_i)} \sum_k \left(v_{j,k} \times d_{j,k}\right) \right\} \qquad (17)$$

$$t_s = t_m = t_v \qquad (18)$$

That is, the computation time is proportional to the number of vertices computed. Then, in consideration of a scene image being assigned to multiple nodes, assuming that the number of vertices of the divided scene image is the average of the number of vertices of all range images $v_{average}$, the number of vertices v_i that are computed on node i is expressed as follows.

$$v_i = \left(\sum_{0\le j\le n} v_j + \sum_{0\le j\le n} \sum_{0\le k\le n} \left(v_{j,k} \times d_{j,k}\right) + v_{average} \times (n_{node} - 1) \right) / n_{node} \qquad (19)$$

n_{node} expresses the number of nodes. By using this, v_i expressed in (19) as an index, we will assign nodes to the scene images.

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

```
Algorithm Procedure of Parallel Alignment
  /* Check correspondence of all pair of */
  /* the range images */
  CreatePairTable;
  /* Create the lists of the files for each
processor */
      CreateFileLists:
      while(error > threshold){
         /* Client Process*/
         for(i = 0; i < nImage; ++i)
          for(j = 0; j < nImage; ++j)
            if(List[i][j]){
              CorrespondenceSearch(i, j);
              CalculationEachMatrix(i, j);
            }
        /* Server Process */
        CalculationMatrix(all);
        /* Server & client process */
      UpdatePosition;
      }
```

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

Computation time is defined as the time taken for one iteration, and an average of time of all iterations is used for the evaluation. We used the 20 range images created from hand model (model A) and the 15 range images measured the great Buddha of Kamakura (model-B) for time evaluation. Figure 4 shows the result of initial and simultaneous alignment of Kamakura Buddha (after 20 iterations).

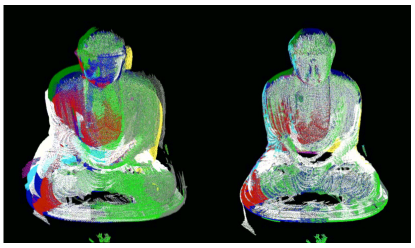


Figure 7. Simultaneous registration result.

We have observed the computation efficiency linearly proportional to the number of processors. In Table 1, the computation times with 1 processor and 16 processors are described. The computation time with 16 processors is approximately 8.4 times faster than that with 1 processor for the hand model, and 8.9 times for the Great Buddha of Kamakura. As for the memory usage, we can reduce the amount of memory used decreases as the number of processor increases.

Table 1. Computational time.

	1Processor(ms)	16Processor(ms)
Hand Model	**11723**	**1391**
Kamakura Buddha	**51715**	**5799**

As shown in Table 2, our method can reduce the amount of memory used in approximately 60% for the hand model and in approximately 43% for the model of the Great Buddha of Kamakura. See [12] in more details.

Table 2. Amount of memory ratio.

	1Proc(MB)	16Proc(MB)	Ratio
Hand Model	**49.2**	**29.3**	**0.5961**
Kamakura Buddha	**258.2**	**110.8**	**0.4293**

2.2 Voxel-based Merging Algorithm

After all range images have been aligned, a volumetric view-merging algorithm generates a consensus surface of the objects from them. Our method merges a set of range images into a volumetric implicit-surface representation, which is converted to a surface mesh by using a variant of the marching-cubes algorithm [14]. Unlike previous techniques based on implicit-surface representations, our method estimates the signed distance to the object surface by determining a consensus of locally coherent observations of the surface [15,16,17,18].

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. We originally design software that merges the aligned 20 range data. However, recent input data is unpredictably huge, we decide to built up a PC cluster to run this merging software; the cluster parallel-processes the merging algorithm for saving the computation time and utilizing a large memory space of many PCs. We produced one integrated digital Great Buddha with this software. The whole data set consists of 3.3 M points, and 5 M polygons that can be merged in approximately 20 minutes on the PC cluster.

We have digitally archived Japanese Buddhas, including Asuka, Kamakura, Nara, and foreign ones, including Thailand’s Wat Si Chum and Cambodia’s Biyon. We are continuing this effort toward completing the world Buddha library, as shown in Figure 8, which digitally display transitions of Buddha shapes in time and region.

3. Texture Mapping and Rendering

The geometric model is vital information regarding the cultural heritage objects because it enables us to analyze object in details. In addition to the geometric model, surface color distribution (texture) is also very important for some kinds of cultural properties. Figure 9 shows one of the Japanese national treasures, the 'Koumoku-Ten' statue. This clay statue still retains the surface colors that were painted at the time the figure was originally made. For such types of cultural properties, we have to archive geometry information, texture information and their

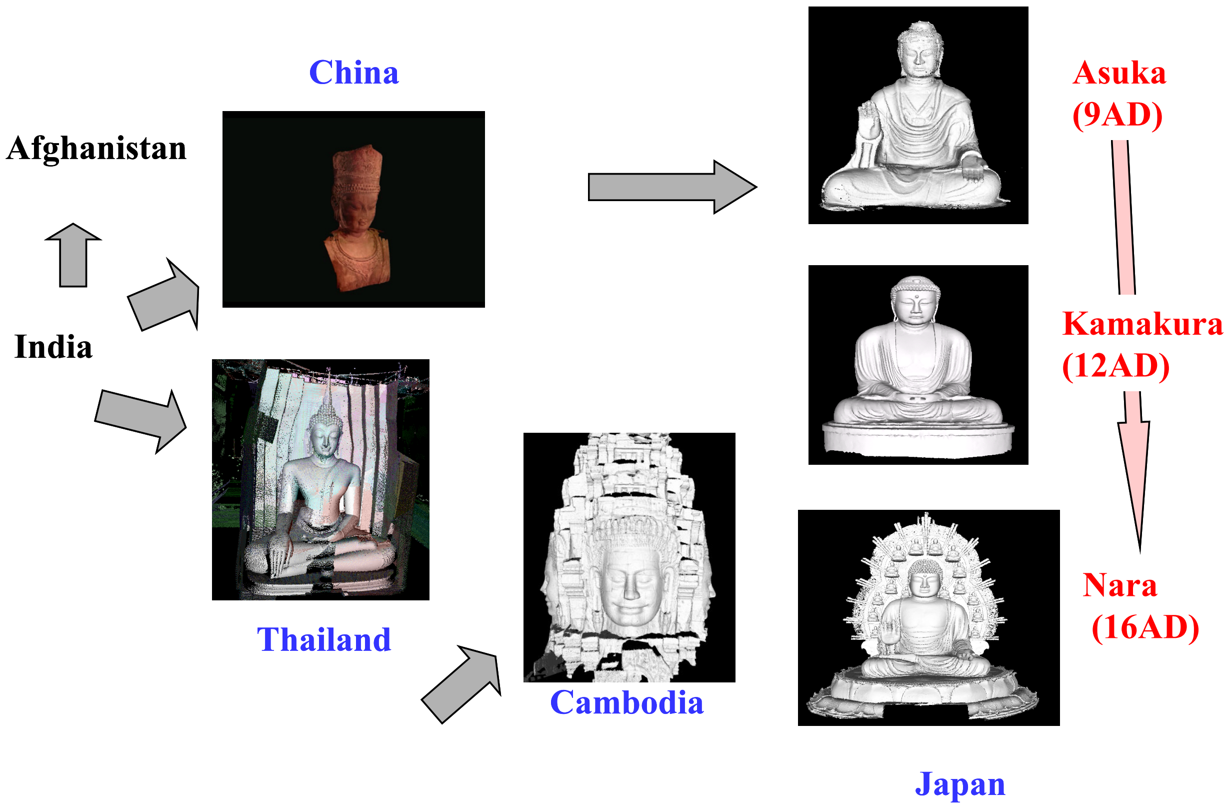


Figure 8. The World Buddha Library (Current Status).

relationship at the same time. For this purpose, we have developed two kinds of texture mapping methods: calibration-based and reflectance-edge based methods.

3.1 Calibration-based Method

The problem of the texture mapping method is how to determine the relationship between image sensors and geometrical sensors. When a short-distance range sensors can be used, as shown in Figure 9, 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 range image is (X,Y,Z); the relationship between them can be described as:

$$\begin{bmatrix} hx_c \\ hy_c \\ h \end{bmatrix} = C_{34} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P_{34} \begin{bmatrix} R_{33} & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The matrix C_{34} represents the relationship between image coordinate and world coordinate, and it can be calculated by scanning the calibration box. Inversely, when we map the texture images onto the geometrical triangular mesh Xn={(Xn,Yn,Zn)| 1<= n <=3} , the corresponding points in image coordinate $x_c=\{x_c,y_c\}$ can be easily calculated as:

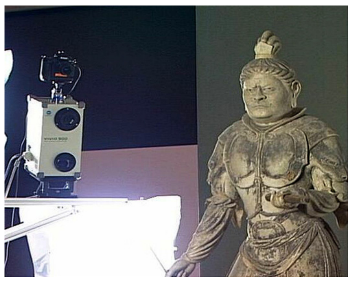


Figure 9. The scanning scene of the Koumoku-Ten clay figure using the VIVID 900.

$$x_c = \frac{c_{11}X_n + c_{12}Y_n + c_{13}Z_n + c_{14}}{c_{31}X_n + c_{32}Y_n + c_{33}Z_n + c_{34}}, \quad y_c = \frac{c_{21}X_n + c_{22}Y_n + c_{23}Z_n + c_{24}}{c_{31}X_n + c_{32}Y_n + c_{33}Z_n + c_{34}}$$

For the modeling of the Koumoku-Ten clay figure, we used 60 range images and color images that taken at the same time. Figure 10 shows the geometrical model (upper), the texture mapped result (lower) and synthesis results under the different lighting conditions generated using the texture mapped result (See Figure 11).

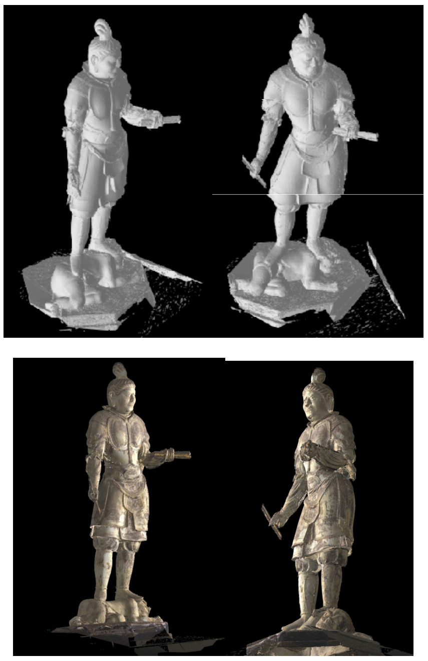


Figure 10. Geometrical model of Koumoku-Ten and texture-mapped model.

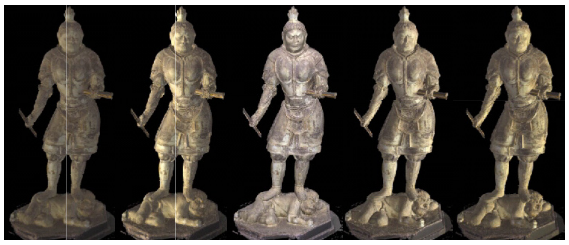


Figure 11. Synthesized results under various lighting conditions (Sunrise->Daytime->Sunset).

3.2 Reflectance-Edge based Method

One solution for determining the relationship between range and color image is, as shown in the previous section, 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 also need a method that does not rely on calibration.

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 [3,16]. Reflectance images share characteristics similar to color images due to the fact that both images are somehow related with surface roughness as shown in Figure 9. Since our CYRAX range scanner uses a green laser diode, reflectance edges can be observed along the boundary between two colors or material boundaries along difference 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.

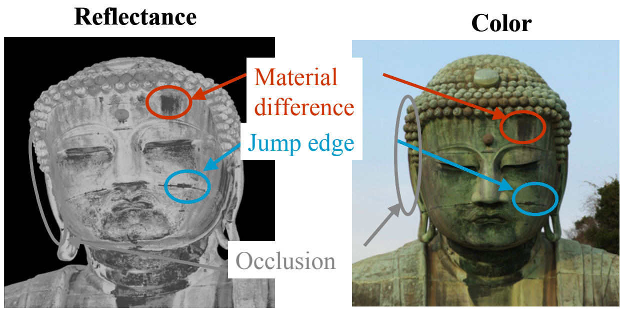


Figure 12. Reflectance and color edges. They share the similar characteristics.

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. On the other hand, 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 the 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 Figure 13. Extracted edges are represented as a collection of points along them. The alignment is done between 3D reflectance points on 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

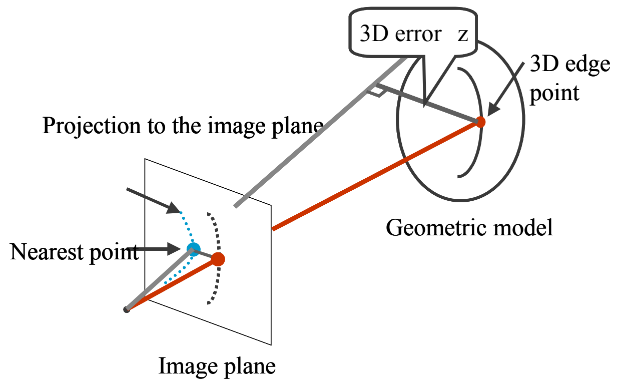


Figure 13. Correspondence between 3D reflectance and 2D color edges.

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.

First the distance between corresponding 2D color edge points and 3D reflectance edge points is evaluated as shown in Figure 13 : where z_i is a 3D error vector which 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.

$\varepsilon_i = Z_i \sin\theta$

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:

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

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

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

Thus, the minimization can be derived as the following least squared equation:

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

We choose the Lorentzian function for this function.

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

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 14 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.

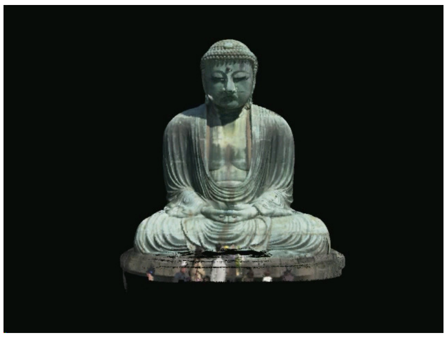


Figure 14. Texture Mapped Kamakura Buddha..

4. Restoring Hypothesized Original State

After we obtain the precise geometry and photometry information of the cultural assets in the current state, we can restore them to their hypothesized original state. In this section, we describe one of the examples: the restoration of the Nara Great Buddha and its main hall

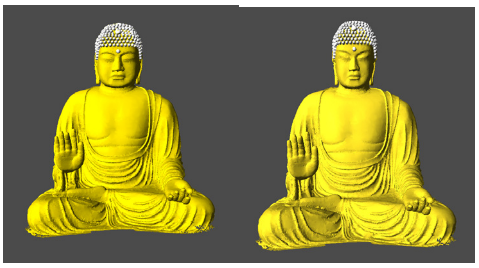
4.1 Restoring Nara Great Buddha

Nara Great Buddha is the main statue of Toudaiji Temple. Unfortunately, the current statue is a rebuilt and repaired one because the original statue was burned and melted down due to a couple of civil wars. Accordingly, the shape of the current Great Buddha is different from that of the original one.

By using the geometrical modeling shown in Section 2, we have acquired the complete 3D geometrical model of Buddha in its current state. From this model, we have attempted to synthesize the original state by morphing the 3D geometry of the model.

From some literature inherited at the temple, we know the sizes of various face parts such as the nose and mouth. Using these data, we design a two-step morphing algorithm.

First, we globally change the scale of the whole portions (e.g., Sitting Height, Face Length, Nose Length); these are gradually modified. In the 2nd stage, vertices are moved one by one iteratively, similar to the constraint propagation algorithm, using smoothness and uniform constraints. The 2-stage morphing enables us to obtain the complete model of the original Great Buddha. Figure 15 shows the 3D models of the current (a) and the original Great Buddha (b). We can easily recognize that the original Buddha is larger and rather skinny.


(a) Current (b) Hypothesized
Figure 15.. Restored result of Nara

4.2 Restorating Toudaiji Main Halle

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 18th 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.

Fortunately, the Toudaiji temple has been displaying a miniature model of the original hall, constructed for the Paris Expo in 1900, as shown in Figure 16(a). We digitized it using the Pulsteck TDS-1500 and scaled it up to the original size as shown in Figure 16(b).

Due to the limitation of resolution, the detail parts cannot be obtained precisely. According to Prof. Keisuke Fujii, an architecture professor at the University of Tokyo, one of the experts on building style in the era, the Toudaiji and Toushou-daiji Temples share a similar format. Here, the main hall of Toshoudaiji Temple were also built during the same period (8th century). After scanning various key parts of the main hall at Toushoudaiji, as shown in Figure 14, we morphed these partial range data by expanding and shrinking the Touhoudaiji parts (Figure 17) to the scaled-up range data of the Toudaiji (Figure 16). The process was conducted by an extended alignment algorithm that allows scale change as well as configuration differences. Figure 18 shows the current Nara Great Buddha main hall and the

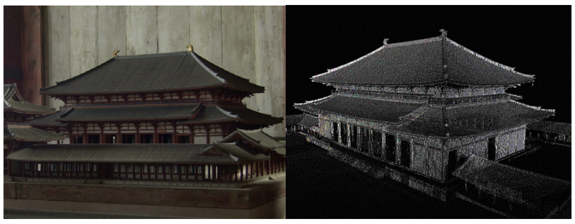


Figure 16. Miniature and its 3D scanned models.

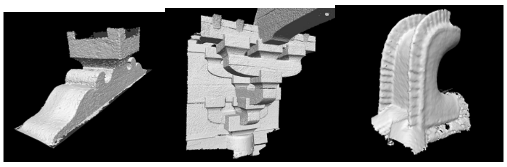
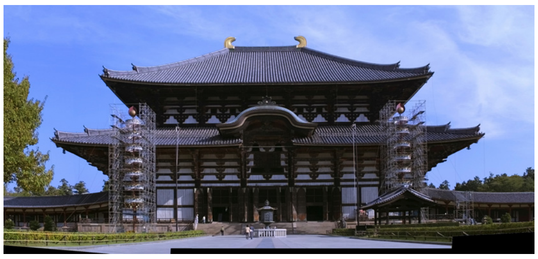
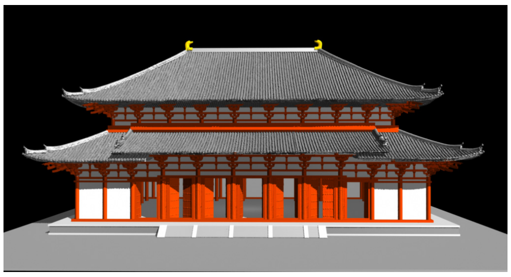


Figure 17. Partial models acquired at Tousyoudaiji.


(a) Current Toudaiji main hall built in 18AD.


(b) Hypothesize 8AD Toudaiji main hall.

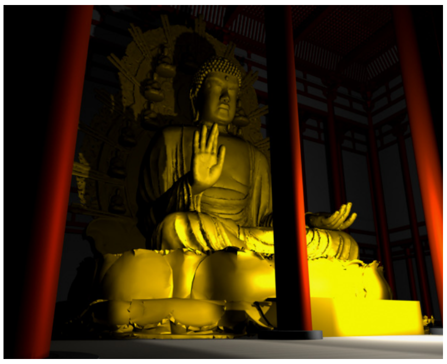

(c) Hypothesized 8AD Nara Buddha in the hypothesized 8AD main hall.

Figure 18. Digitally restored Nara Great Buddha..

original one digitally restored by our method.

5. Conclusion

In this paper, we introduced our project to digitally archive and restore cultural heritage objects. Our project's main goal is to develop a method of 'modeling from reality', in which the digital model of cultural properties is created by using various computer vision methods. For the observation of geometrical information, we used laser range finders and post process algorithms, including registrating and merging the range images. For the texture information, we have developed several texture mapping methods. For the short distance range sensors, we calibrated the relationship between the range sensor and the image sensor for complete texture alignment. For the long distance range sensors, we developed a non-calibrated texture alignment method by using laser reflectance features. Digital restoration of lost cultural heritage objects has a big advantage compared with other restoration methods such as physical construction of actual temples, because we can examine various hypotheses without any physical changes nor long building periods. We demonstrated the effectiveness of this method through the restoration of the Nara Great Buddha and its main hall. We are also conducting a project to create a digital library of the world great Buddhas, including three Japanese Buddhas, Sri Chum Buddha in Thailand, and Biyon's in Cambodia. The models and restoration results constructed so far can be viewed at [21]

Acknowledgment

This research is sponsored, in part, by JST under Ikeuchi Crest program. The Bayon in Cambodia was digitized with the cooperation of Japanese Government Team for Safeguarding Ankor (JSA).

References

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

[2] M. Levoy et. al., "The digital Michelangelo project," *SIGGRAPH 2000*, New Orleans.

[3] J. Wasserman, *Michelangelo's Florence Pieta*, Princeton University Press 2003.

[4] I. Stamos and P. Allen, "Automatic registration of 2-D with 3-D imagery in urban environments," *ICCV2001*, Vancouver.

[5] 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.

[6] 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.

[7] 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.

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

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

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

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

[12] 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,* Oct 2003.

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

[14] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *SIGGRAPH 96*, New Orleans, LA.

[15] M. Wheeler, Y. Sato, and K. Ikeuchi, "Consensus surfaces for modeling 3D object from multiple range images," *ICCV98.*

[16] 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

[17] 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,* Oct 2003

[18] R. Sagawa and K. Ikeuchi, "Taking Consensus of Signed Distance Field for Complementing Unobservable Surface," *Int. Conf 3-D Digital Imaging and Modeling,* Oct 2003

[19] R. Kurazume, M. D. Wheeler, and K. Ikeuchi, "Mapping textures on 3D geometric model using reflectance image," *Data Fusion Workshop in IEEE Int. Conf. on Robotics and Automation,* 2001.

[20] 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*, 2002.

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