Reconstruction of Complete 3D Models by Voxel Integration*

Takaki NANYA**, Hiroki YOSHIHARA** and Takashi MAEKAWA**
**Yokohama National University, Department of Mechanical Engineering Digital Engineering Laboratory
Tokiwadai, Hodogaya, Yokohama 240-8501, Japan
E-mail: nanya-takaki-hw@ynu.ac.jp

Abstract
A method for reconstructing a complete three-dimensional (3D) voxel model from multiple images captured during two circular motions of an object is proposed. The two circular motions include the object with a regular orientation and the corresponding reversed orientation. The shape-from-silhouette method, which constructs a voxel-based 3D model from the silhouettes of two-dimensional (2D) images, is used for the reconstruction. In general, a single camera focused on an object’s circular motion cannot capture the entire object using a single fixed orientation. Thus, it is required to invert the orientation of the object at least once to capture the entire object. In this paper, a novel method for integrating two incomplete reconstructed voxel models to produce a single complete voxel model is discussed. Complex examples are provided to demonstrate the effectiveness of the proposed algorithms.

1. Introduction

Recent advances in computer vision and geometric modeling techniques have facilitated the reconstruction of three-dimensional (3D) computer-aided design (CAD) models from images of existing objects such as mechanical parts, insects, plants, and cultural heritage artworks. The process of measuring an object and reconstructing it as a 3D CAD model is known as reverse engineering. Typical measuring devices used for reverse engineering include laser scanners, coordinate measurement machines (CMMs), and computer tomography (CT) scanners. However, these devices are expensive, and they fail to capture any color information.

In many reverse engineering applications, reconstructed models are represented by a triangular mesh. However, voxel representation is preferred to triangular meshes in some types of applications due to the following reasons. Voxel models are not only easy to handle, but also convenient to store properties of solid objects such as color, density, temperature etc. Furthermore, Boolean operations of solids can be treated in a very simple manner. The applications of voxel models include additive manufacturing, building instructions for LEGO models, and the digital assembly of biological fundamental building blocks (e.g., DNA and proteins)\(^1\). 3D printing is an additive manufacturing technique, which creates parts in layers by spreading powder and then ink-jetting binder materials into the powder bed\(^2\). Each layer can be considered to be a 2D array of voxels, where a binder material selectively ink-jets onto voxels to form the object. Recently, Hiller and Lipson\(^1\) introduced digital 3D printing where pre-existing physical voxels were used as a material block for layered manufacturing.

In this paper, we introduce a novel method for reconstructing complete voxel models of complex objects by using a low-cost camera, which can also capture color information. We focus on the generation of additive manufacturing layer information, and the creation of building instruction for LEGO models. As shown in Figure 1, our data acquisition system consists of a camera placed on a tripod and a computer-controlled turntable, which is considerably more cost-effective than a laser scanner. We use the shape-from-silhouette (SFS) method\(^3\),

---


\(^2\) EOSINT M270, EOS, 2009.

which constructs a voxel-based 3D model from silhouette images of an object captured during circular motions on a turntable.

The camera is fixed, so it can only capture information in the scene that is visible to the camera during the circular motions of the object. Thus, complete object information has to be acquired by inverting the orientation of the object from the camera viewpoint, which is then recaptured via another round of circular motions. Using the two sets of image silhouettes $I_P$ and $I_Q$ of the object collected during the circular motions in each orientation, we obtain two incomplete voxel models $V_P$ and $V_Q$ by applying the SFS method to the common region of interest.

The two incomplete voxel models possess partially overlapping regions, which need to be integrated to produce a complete voxel model without redundant data. There are many techniques for combining two triangular meshes into a single surface\(^{(4),(5)}\). However, the integration of two voxel models to generate a single complete voxel model is a difficult problem, and hence a few researches have studied this problem. Park and Subbarao\(^{(6)}\) integrated the two models reconstructed from two different poses of an object based on the signed distance function. A coarse rigid transformation between the two models is estimated by a stable tangent plane of each pose-model. The model representation alternates between voxel and mesh during the reconstruction processes, and hence it is complicated. Furthermore, the two models are integrated by the set intersection, and therefore the thin portions of the object may not be recovered due to errors in the system.

We introduce a novel method to integrate the two incomplete reconstructed voxel models to generate a single complete voxel model without converting the two models into mesh models. In summary, the major contributions of our work can be summarized as follows:

- We have employed a voxel-based representation throughout the whole process, and hence it is simple to implement and the resulting model is more accurate than the existing methods such as in Park and Subbarao\(^{(6)}\).
- We have increased the robustness of the reconstruction of details by eliminating redundant voxels from the union of the two sets though a visibility check in the voxel integration process.
- Most importantly, it brings together all these techniques into a robust procedure for reconstruction of high-quality colored voxel models of complex objects.

The remainder of this paper is organized as follows. In Section 2, we describe the flow of techniques used by the reconstruction method. We briefly explain the SFS method, and introduce a novel method for projecting voxels onto silhouettes. In Section 3, we explain the computation of the coordinate transformation between $V_P$ and $V_Q$. In Section 4, we integrate the two incomplete voxel models to produce a single complete model. In Section 5, we present examples that demonstrate the effectiveness of the voxel-based complete 3D model reconstruction system. Finally, we provide our conclusions in Section 6.
2. Key steps of the proposed approach

2.1. Key steps

Figure 2 shows a flowchart of our proposed 3D voxel model reconstruction system, which contains six key steps. The data acquisition system shown in Figure 1 consists of a CANON EOS Kiss X4 camera, which captures still images at a resolution of 2100 × 1400 pixels, and a computer-controlled turntable. Still images are transferred to a PC with a Core i7 CPU 870 at 2.93 GHz and 4 GB RAM.

Camera calibration is necessary to determine the relationship between a 3D world coordinate system \((X_W, Y_W, Z_W)\) and its corresponding 2D image coordinates \((u, v)\), as shown in Figure 3. The relationship between a 3D point \(M_{\text{world}} = (X_W, Y_W, Z_W)\) and its image projection \(m_{\text{image}} = (u, v)^T\) is given by

\[
s m_{\text{image}} = [A] [R_{\text{calib}} T_{\text{calib}}] M_{\text{world}},
\]

where \(s\) is a scale factor; \(m_{\text{image}} = (u, v, 1)^T\); \(M_{\text{world}} = (X_W, Y_W, Z_W, 1)^T\); \([A]\) is the camera’s intrinsic matrix; \(R_{\text{calib}}\) is the 3D rotation matrix with three rotation angles \(\alpha, \beta, \) and \(\gamma\); and \(T_{\text{calib}}\) is the 3D translation vector \((t_x, t_y, t_z)\) that produces a 3 × 4 extrinsic matrix \([R_{\text{calib}}, T_{\text{calib}}]\).

The camera’s intrinsic matrix is given by

\[
A = \begin{pmatrix}
    f_x & 0 & u_0 \\
    0 & f_y & v_0 \\
    0 & 0 & 1
\end{pmatrix},
\]

where \(u_0\) and \(v_0\) are the coordinates of the image’s center, which is the intersection point between the optical axis and the image plane, whereas \(f_x = f / dx\) and \(f_y = f / dy\) are the ratios of the focal length \(f\) to the width \(dx\) and the height \(dy\) of a pixel, respectively.

We use a camera calibration method introduced by Atsushi et al.\(^7\), which is a variation of the method proposed by Lavest et al.\(^8\). This method can handle single focus lenses and zoom lenses to determine the intrinsic and extrinsic parameters of the camera. The camera is calibrated using a 40 mm steel cube, as shown in Figure 4. First, the image center \((u_0, v_0)\) is determined by tracing two vertices of the cube while zooming the camera. The intersection point of the two straight lines is the image center \((u_0, v_0)\). The two unknown intrinsic parameters \(f_x\) and \(f_y\), and the six unknown extrinsic parameters \(\alpha, \beta, \gamma, t_x, t_y, t_z\) are related by the distance \(D\) between one of the two end vertices of an edge of the calibration cube and the plane which contains the origin of camera coordinates and the edge. The objective function,
which is the sum of squares of the distance $D$, is minimized to obtain the eight unknowns. Note that we need at least eight distances between the end points of the edges and the planes that contain the edges\(^{(7)}\).

The SFS method is a popular method for reconstructing 3D objects from multiple silhouette images\(^{(3),(7),(9)}\). For each turntable rotation of $10^\circ$, an image of the object is captured by the camera placed on a tripod at a fixed angle. In Section 2.2, we briefly explain the concept of SFS method. In order to apply SFS method, we need to extract silhouettes from all the images taken during the circular motion of the turntable with two complementary orientations. Let us denote the set of images in the regular orientation as $I_P = \{I_1^P, I_2^P, \ldots, I_N^P\}$ and those of in the reversed orientation as $I_Q = \{I_1^Q, I_2^Q, \ldots, I_N^Q\}$. In this study, the number of images with each orientation is typically $N = 36$. It is vital to extract the silhouettes of the object from images as accurately as possible. We use the background subtraction algorithm developed by\(^{(7)}\), which utilizes the $CIE L^*a^*b^*$ color space instead of the RGB color space (see Figure 5).

Full details of steps 4, 5, and 6 are provided in Sections 2.2, 3, and 4, respectively.

2.2. SFS method

A visual cone that represents the bounding volume of the object is constructed by projecting the 2D silhouette image into 3D space through the center of the camera. If we intersect all of the visual cones from each silhouette, we obtain an approximated object known as a visual hull. Laurentini\(^{(3)}\) described the visual hull as the maximal object that gives the same silhouette of the 3D object from any possible viewpoint.

In order to represent the visual hull using voxels, the region of interest is subdivided into $N_1 \times N_2 \times N_3$ voxels and each voxel is projected onto all the silhouette images. A voxel is classified as inside, if its projection onto each of the silhouette images is inside the corresponding silhouette images or if it partially overlaps them. If the projection of the voxel is completely outside the silhouette for even a single image, it is classified as outside. After classifying all the voxels as either inside or outside, we can construct an approximation of the visual hull.
from the union of all the inside voxels. Figure 6(a) shows the inside voxels, which are the visual cone generated by removing the outside voxels from the region of interest (black rectangular box). After the six visual cone intersections, the remaining inside voxels generate the stag beetle as depicted in Figure 6(b).

We use voxel octree to further classify the inside voxels as completely inside or partially overlapping ambiguous voxels\(^{(10)}\). When octree are used for refinement, we start from a cubical bounding box rather than a rectangular parallelepiped bounding box. Ambiguous voxels are subdivided recursively into eight subcubes, until no ambiguous voxels remain or the size of the subdivided voxel reaches the minimum size criterion (see Figure 7). We project each voxel onto all the silhouette images and perform the inside/outside check. If the voxel projection is completely outside the silhouette for even a single image, it is considered to be outside, even if the remaining check results are inside or ambiguous. The size of the smallest voxel determines the resolution of the visual hull. After the SFS method has been applied, we subdivide the voxels into plain voxels using the minimum size of the voxel octree.

The relation between the voxel size and the pixel size can be determined as follow. Let us denote the characteristic linear dimension of an object as \(l\). Suppose the object is captured by a camera with \(m \times n\) (\(m > n\)) pixel resolution, and the characteristic linear dimension of the object occupies half of the \(m\) pixels, then one pixel represents \(2l/m\). If the length of the root-cube is \(L\), and the octree depth is \(k\), then we must have the following relation:

\[
\frac{L}{2^k} > \frac{2l}{m}.
\]  

(3)

We apply SFS method to the image silhouettes \(I_P\) and \(I_Q\), which yields the voxel models \(V_P\) and \(V_Q\), respectively. Throughout this paper, \(V_P\) is the base voxel model, and \(V_Q\) is the voxel model which will be integrated to \(V_P\).

2.3. Projection of voxels onto silhouettes

If the octree depth is low and the eight corners of the voxels are projected onto silhouettes, as in Szeliski\(^{(10)}\), an ambiguous voxel may be judged to be an outside voxel, as shown in Figure 8. In order to avoid such problems, we use the voxel edges for an inside/outside check. As shown in Figure 9, the 12 edges of a voxel are projected onto a silhouette and a decision is made based on the intersection of the projected edges and the silhouette (denoted by the red line in Figure 9). If there is an intersection at any point, it is considered to be ambiguous.
3. Coordinate transformation

$V_P$ and $V_Q$ are generated in different coordinate systems, so rigid-body translation $T$ and rotation $R$ must be applied iteratively to $V_Q$ to ensure its closest possible correspondence with $V_P$. We employ the iterative closest point (ICP) algorithm\textsuperscript{(11)} to find $T_{ICP}$ and $R_{ICP}$, which minimizes the difference between two voxel models. However, the ICP requires an initial estimation of the transformation, for example the spin-image techniques\textsuperscript{(12)}. For the sake of clarity, we briefly review the basic concept of the spin-image technique in Section 3.1 and describe its application to voxel models in Section 3.2. Details of the application of the spin-image technique to triangular meshes are given in Johnson\textsuperscript{(12)}.

3.1. Spin-image technique

The spin-image technique was developed by Johnson\textsuperscript{(12)} for matching two polygonal models. This method describes a vertex point $v$ on the triangular mesh surface of an object with respect to an oriented point $p$ (a 3D point $p$ with the surface normal $N$) using two parameters $(\alpha, \beta)$. The vertical coordinate $\beta$ denotes the signed distance from $v$ to the tangent plane at $p$, while the horizontal coordinate $\alpha$ indicates the distance between $p$ and the vertical projection point of $v$ onto the tangent plane, as shown in Figure 10. The accumulation of the $(\alpha, \beta)$ coordinates for all points on the 3D mesh is known as a “spin-map $S_0(v)$”.

The spin-map is formally defined as follows:

$$S_0(v) \rightarrow (\alpha, \beta) = \left( \sqrt{||v - p||^2 - (N \cdot (v - p))^2}, (N \cdot (v - p)) \right).$$ \hfill (4)

Each point on the spin-map is bilinearly interpolated to the four surrounding bins in the 2D array and the sum of the bilinearly interpolated points of the spin-map in each bin is mapped to grayscale, which is known as the “spin-image”. Spin-images are independent of the rigid body transformations applied to the mesh.

We apply the spin-image technique to voxel models. Common regions in two partially overlapping voxel models can be detected by directly comparing spin-images from voxel model $V_P$ with spin-images from voxel model $V_Q$. After determining the correspondences between points on $V_P$ and points on $V_Q$ with a similar spin-image, we can localize $V_Q$ with respect to $V_P$ to find $R_0$ and $T_0$ using quaternion\textsuperscript{(11)}. In the implementation, three points on $V_P$ are randomly selected, and the distances between two points are computed. If all the three distances exceed 10% of the bounding box diagonal of the object, the selected three points represent the points on $V_P$, otherwise the three new points are selected randomly to repeat this process. The transformations $R_0$ and $T_0$ are used for the initial approximation of the ICP.

3.2. Evaluation of surface normals

In this study, we apply spin-image techniques to match two voxel models. We use the surface voxel’s centroid $v_{i,j,k}$ as the vertex coordinates. The normals of the surface voxels are evaluated by computing the distance field from the surface voxels as shown in Figure 11, 12.
Let us denote $S = s_1, \ldots, s_m$ as a dataset of surface voxels. The distance function $d(x) = \text{dist}(x, S)$ from $S$ needs to be computed to evaluate the surface normals. We formulate the boundary value problem, which describes the motion of a surface in a direction normal to the surface, as follows:

$$|\nabla d(x)| = 1, \quad d(x) = 0, \quad x \in S.$$  \hspace{2cm} (5)

Equation (5) is known as the Eikonal equation. Sethian \cite{13,14} developed a fast marching method (FMM) to solve the Eikonal equation. In the first step of the FMM, 3D grids (voxels) with uniform spacing and the minimum voxel size $h$ are generated outside the voxels $V_P$ and $V_Q$. The surface voxels are then extracted and assigned a zero distance, i.e., $d(x) = 0$ (see Figure 12). Equation (5) is then discretized based on the following upwind scheme:

$$\left[ \max(D^+ x d_{ijk}, -D^+ x d_{ijk}, 0)^2 + \max(D^+ y d_{ijk}, -D^+ y d_{ijk}, 0)^2 + \max(D^+ z d_{ijk}, -D^+ z d_{ijk}, 0)^2 \right]^{\frac{1}{2}} = 1,$$  \hspace{2cm} (6)

where $D^+ x$, $D^+ y$, $D^+ y$, $D^+ z$, and $D^+ z$ are the forward and the backward difference operators in the x, y, and z directions, respectively.

In the FMM, the moving front is restricted to movement in the same direction using only the upwind values. Assuming that we know the neighboring grid values for $d_{ijk}$, we can update $d_{ijk}$ by solving the quadratic equation (6) for $d_{ijk}$. The FMM systematically constructs the solution $d$ using only upwind values, i.e., from lower values of $d$ to higher values. Only voxels in a thin zone around the current front are considered to move the thin zone forward. The values of existing points are frozen and new ones are brought into the narrow band zone. We can repeat this procedure to determine the values of the distance function for all the voxels within a constant distance from the surface voxels. Figure 11 shows the propagation of the fronts projected onto a plane. Initially, the front starts from the surface voxels and the voxels merge with each other to form iso-distance contours. Warm colors denote a small distance from $S$. The propagation of the front is terminated after the front reaches the prescribed distance, as
shown in Figure 11. Finally, the value of the distance function of voxels designated as inside according to the SFS method is multiplied by “-1”, which generates a signed distance field (see Figure 12)(15).

![Fig. 12 Signed distance from surface voxels in 2D. Surface voxels are colored in gray (zero distance), inner voxels are in blue (negative distance), and outer voxels are in yellow (positive distance).](image)

Using the value of the distance field, we calculate the normal vector of voxel \( \mathbf{n}_{(i,j,k)} = (n_{x(i,j,k)}, n_{y(i,j,k)}, n_{z(i,j,k)})^T \) according to the central difference scheme. The calculation will be explained in terms of component \( x \) of the normal vector, i.e., \( n_{x(i,j,k)} \), but the calculation of components \( y \) and \( z \) can also be achieved in the same manner. We use the value of the signed distance function of the 26 voxels adjacent to the surface voxel. As shown in Figure 13, the \( x \)-component of the surface normal vector \( n_{x(i,j,k)} \) is obtained by applying the central difference formula nine times, which is weighted by the value of the Gaussian filter mask as follows:

\[
\begin{align*}
    n_{x(i,j,k)} &= \frac{1}{16} \left( (d_{(i+1,j+1,k+1)} - d_{(i-1,j+1,k-1)}) + 2(d_{(i+1,j-1,k+1)} - d_{(i-1,j-1,k-1)}) \\
    &+ (d_{(i+1,j,k+1)} - d_{(i-1,j,k-1)}) + 2(d_{(i+1,j,k-1)} - d_{(i-1,j,k+1)}) \\
    &+ 4(d_{(i+1,j,k)} - d_{(i-1,j,k)}) + 2(d_{(i+1,j+1,k)} - d_{(i-1,j+1,k)}) \\
    &+ (d_{(i+1,j-1,k+1)} - d_{(i-1,j-1,k+1)}) + 2(d_{(i+1,j+1,k+1)} - d_{(i-1,j,k+1)}) \right). \\
\end{align*}
\]

(7)

After applying the same method to \( n_{y(i,j,k)} \) and \( n_{z(i,j,k)} \), we can determine the normal vector \( \mathbf{n}_{(i,j,k)} \) for the voxel \((i, j, k)\). The unit normal vector \( \mathbf{N}_{(i,j,k)} \) can also be found, as follows:

\[
\mathbf{N}_{(i,j,k)} = \frac{\mathbf{n}_{(i,j,k)}}{||\mathbf{n}_{(i,j,k)}||}.
\]

(8)
Figure 14 shows the normal vector distributions of a voxel model of a stag beetle. This illustration clearly shows the effectiveness of Equations (7) and (8).

Fig. 14 (a) Normal vector distributions of the voxel model. (b) Close-up view of (a).

4. Voxel integration

In the first step, a cubical bounding box $V_{P0}$ of the voxel model $V_P$ is generated which is obviously tighter than cubical bounding box used to obtain $V_P$. Voxel models $V_P$ and $V_Q$, which are used to compute $R_{iCP}$ and $T_{iCP}$ in Section 3, are generated from larger size cubical bounding box of the object, since we want to make sure that the object is completely inside the cubical bounding box. Moreover, the maximum octree depth of $V_P$ and $V_Q$ used for the computation of ICP does not need to be fine compared with that of the integrated voxel model $V_I$. This owes to the fact that the transformations do not change much for voxel models as long as the shape of the object is roughly represented. Accordingly, voxels used in the integration are different from ones used in the computation of ICP.

From Equation (1) and referring to Figure 15, we have the following relations:

$$s \tilde{p}_{image} = [A] \left[ R_{Pcalib} T_{Pcalib} \right] \tilde{p}_{world}, \quad (9)$$

$$s \tilde{q}_{image} = [A] \left[ R_{Qcalib} T_{Qcalib} \right] \tilde{q}_{world}, \quad (10)$$

$$\tilde{p}_{world} = \begin{bmatrix} R_{iCP} & T_{iCP} \\ 0 & 1 \end{bmatrix} \tilde{q}_{world}, \quad (11)$$

where $l = 1, \ldots, N$. From Equations (10) and (11) we obtain

$$s \tilde{q}_{image} = [A] \left[ R_{Qcalib} T_{Qcalib} \right] \left[ R_{iCP} T_{iCP} \right]^{-1} \tilde{p}_{world}. \quad (12)$$

SFS method is applied to $V_{P0}$ where each voxel octree element of $V_{P0}$ is projected to each silhouette of $I_P$ and $I_Q$ using Equation (9) and (12), respectively. We denote the resulting voxel models by $V_P$ and $V_Q$. Since both $V_P$ and $V_Q$ are carved from $V_{P0}$, they are aligned in the same voxel coordinate system. The voxel model $V_I$ is created from the integration of the two generated voxel models $V_P$ and $V_Q$.

Before integration, it is necessary to test whether $V_I$ should be considered as the set union or the set intersection of the voxel sets $V_P$ and $V_Q$. If $V_I$ is the set intersection $V_{inter}$ of the sets $V_P$ and $V_Q$, a thin portion of the object, such as the leg of the stag beetle, may be lost during reconstruction because of various errors in the system (see Figure 16(c)), e.g., camera calibration errors, silhouette extraction errors, errors caused by the ICP, which has a tendency to fall into local minima, and the backlash of the gears used in the turntable. However, if $V_I$ is the set union $V_{union}$, such as the case of the stag beetle’s abdomen, the reconstruction becomes redundant and the restoration accuracy may decrease (see Figure 16(b)). Therefore, $V_I$ can
be obtained by eliminating redundant voxels by focusing on the set union in this study. Let us denote the camera centers of images \( I_P \) and \( I_Q \) as \( C_P = \{C_{P1}, \ldots, C_{PN}\} \) and \( C_Q = \{C_{Q1}, \ldots, C_{QN}\} \), respectively. The redundant voxels are removed using the following rule. If a voxel is visible only from one of the camera centers \( C_P \) or \( C_Q \), the decision is made based on the visible side. First, the \( V_{\text{union}} \) is created by taking the union of \( V_P' \) and \( V_Q' \), as shown in Figure 18. The surface voxels of \( V_{\text{union}} \) are examined to determine whether they are visible from \( C_P \) or \( C_Q \), or both \( C_P \) and \( C_Q \), by checking the angle between the surface normal \( N_{(i,j,k)} \), evaluated from Equation (8), at the centroid of a voxel \( v_{(i,j,k)} \) and a vector from \( v_{(i,j,k)} \) to \( C_l \) (or \( C_l' \)), as shown in Figure 17. The angle \( \theta \) is obtained by

\[
\theta = \cos^{-1}\left( \frac{N_{(i,j,k)} \cdot (C_l' - v_{(i,j,k)})}{|C_l' - v_{(i,j,k)}|} \right). \tag{13}
\]

We now describe the method used for visibility checking for \( v_{(i,j,k)} \) from \( C_l' \) (\( l = 1, \ldots, N \)). The same visibility check can be applied from \( C_l'' \). If \( \theta \) is greater than 90°, the voxel is back-facing and not visible, whereas the voxel is front-facing if \( \theta \) is less than 90°, and it is visible unless there are self-occlusions. Self-occlusions can be detected by checking whether there are intersections between the line connecting \( v_{(i,j,k)} \) and \( C_l' \) with other surface voxels. We conduct a visibility check for each centroid of the surface voxel \( v_{(i,j,k)} \), and it is determined to be visible from \( C_l' \) if it is visible from at least one of the camera centers \( C_l' \) (\( l = 1, \ldots, N \)).

If a voxel is visible from both \( C_P \) and \( C_Q \), the voxel remains in \( V_I \). However, if a voxel is visible from \( C_P \) but not from \( C_Q \), and the voxel does not belong to \( V_P' \), the voxel is removed from \( V_I \), and vice versa. These operations are performed repeatedly until no more voxels can be removed (see Figure 18). In the final stage, the isolated voxels consist of a small number.
5. Examples

In this section, we provide an example of a real 85 mm long stag beetle. A maximum octree depth of seven is used for the spin-image and ICP techniques to speed up the computation. The maximum octree depth for this model is eight, which yields a voxel size of 0.323 mm. Various errors in the system do not allow us to recover the thin parts of the stag beetle, such as the antennae and legs, if the octree depth exceeds eight. As shown in Figure 19(a) and (b), the stag beetle is fixed to the turntable with a pin to prevent any effects from its possible movement or vibration. The pin is erased manually from one image, so it is not reconstructed during the SFS process. Using the SFS method, we create two voxel models, \( V_P \) and \( V_Q \), with different orientations (see Figure 19(c)). Rough matching results are produced using the spin-image technique (see Figure 19(d)). The results are then used as the initial values for the ICP for further alignment, as shown in Figure 19(e).

Finally, the two voxel models \( V_P \) and \( V_Q \) are integrated by taking the union and removing the redundant voxels during a visibility check. The number of redundant voxels deleted during the visibility check is 11838, which are 6.95% of \( V_P \cup V_Q \). Color images of the final integrated voxel model \( V_I \) are shown in Figure 19(f), (g), and (h). Colors are assigned to the voxels when they are projected onto the images. Table 1 summarizes the number of images and the voxel data information used during the reconstruction of the stag beetle, while Table 2 lists the decomposed computational time. We note here that the time spent on data acquisition and the calibration are not included, since they depend largely on the users.

As shown in Figure 19(h) and (i), the voxels of the root of the right antenna are not recovered. This problem is due to various errors in the system. Furthermore, the reconstructed antennae appear to be thicker than the originals, as shown in Figure 19(a) and (f). The maximum octree depth of the ambiguous voxels remain as part of the object, so the thin portions of objects where the thickness is close to the minimum voxel size, such as antennae, are reconstructed with twice the thickness of the original object.

In addition, Figure 20 shows the difference in the accuracy of the reconstruction at maximum octree depth of 5, 6, 7, and 8. The reconstructed model becomes closer to the target object as the maximum octree depth increases.

Finally, the voxel integration approach used in this technique is compared with the integration obtained using voxel union and intersection approaches. Figure 16(b) shows the union of \( V_P \) and \( V_Q \), i.e., \( V_{union} \). Compared with Figure 16(a), it is clear that there are redundant voxels in the reconstruction of the abdomen. Figure 16(c) shows the intersection of \( V_P \) and
VQ, i.e., Vinter, where there is a lack of detailed reconstruction in many areas. If there are no errors in the system, \(V_{\text{inter}}\) should be used as the integration model. If there are errors in the system, however, \(V_{\text{inter}}\) fails to recover the original shape. Figure 16(d) shows the result of the proposed method where abdomen reconstruction is achieved without redundancy, whereas voxel model integration is achieved with minimal loss in the details of the legs (see Figure 19 (f), (g) and (h) etc.).

The proposed technique is also applied to the generation of LEGO building instructions, as shown in Figure 21(a). Figure 21(b) shows the LEGO model constructed based on the instructions. Figure 22 shows the plastic stag beetle model manufactured by the laser sintering technology of Aspect Inc. using our voxel model. Unfortunately, the de facto industry standard for representing geometric information for additive manufacturing is the STL file format. Therefore the voxel data is converted to STL file format and then it is manufactured. We hope that the voxel models will become the standard in some types of additive manufacturing technology in the near future.

6. Conclusions

We have developed a low-cost system, which consists of a digital camera and a computer-controlled turntable, for reconstructing complete voxel models from multiple images captured during two circular motions, i.e., a regular orientation and the corresponding reversed orientation.

The advantages of our proposed system are as follows:

- We can generate a complete integrated voxel representation of a complex model without converting the voxel into a triangular mesh. The resulting voxel structure can be used to provide printing information for additive manufacturing, such as 3D printing, and the automatic generation of building instructions for LEGO models.
Fig. 19 Stag beetle: (a) Regular orientation. (b) Reversed orientation. (c) Reconstructed voxel models. $V_P$ (regular) is shown in blue, and $V_Q$ (reversed) is shown in red. (d) $V_P$ and $V_Q$ are matched roughly using the spin-image technique. (e) $V_P$ and $V_Q$ are matched completely after the application of the ICP using (d) as the initial transformation. (f) Redundant voxels are removed and the textures are mapped to the voxels. (g) Bottom view of the reconstructed model. (h) The root of the right antenna has not been recovered. (i) Close-up view of (h).

Table 1 Reconstructed results of the stag beetle.

<table>
<thead>
<tr>
<th>Model</th>
<th>Voxel Model</th>
<th># of images</th>
<th>Length of cubical bounding box (mm)</th>
<th>Max. octree depth</th>
<th># of voxels</th>
<th>Voxel length (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stag beetle</td>
<td>$V_P$</td>
<td>36</td>
<td>90</td>
<td>7</td>
<td>17923</td>
<td>0.703</td>
</tr>
<tr>
<td></td>
<td>$V_Q$</td>
<td>.36</td>
<td>90</td>
<td>/</td>
<td>18579</td>
<td>0.703</td>
</tr>
<tr>
<td></td>
<td>$V_I$</td>
<td>.72</td>
<td>82.75</td>
<td>8</td>
<td>170367</td>
<td>0.323</td>
</tr>
</tbody>
</table>

Table 2 Decomposed computational time (sec).

<table>
<thead>
<tr>
<th>Model</th>
<th>Spin-image</th>
<th>ICP</th>
<th>Voxel integration</th>
<th>Coloring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stag beetle</td>
<td>2.472</td>
<td>33.056</td>
<td>138.16</td>
<td>31.318</td>
</tr>
</tbody>
</table>

Fig. 20 Reconstructed stag beetle models represented as voxels at a maximum octree depth of (a) 5, (b) 6, (c) 7, and (d) 8.
We have developed a novel method to apply spin-image technique to voxel models. We have increased the robustness of the reconstruction of details (such as in the case of the stag beetle’s leg) by introducing inside/outside checking based on the edges. We have improved the accuracy of the voxel integration process by eliminating redundant voxels from the union of the two sets during the visibility check. However, our current system has the following drawbacks:

- Concave regions that do not appear in the silhouettes cannot be reconstructed.
- Extremely thin parts of an object, such as stag beetle antennae, cannot be reconstructed as precisely as other sections.

The proposed system can be improved in several ways, and we have listed some of these as future work:

- We aim to employ stereo vision techniques in order to improve the reconstruction of concave regions that do not appear in the silhouettes.
- We plan to improve the entire system by reducing errors in calibration, silhouette extraction, and the hardware system.

Acknowledgments

We would like to express our sincere gratitude to a former member of the Digital Engineering Lab at YNU, Koutaro Atsushi, for his constructive discussions, and the current members, Yusuke Yamaura, Hui Lin, and Tomonori Kitagawa, for their help. Finally we would like to thank ASPECT INC. for manufacturing the stag beetle model.

References

(3) Laurentini, A., The visual hull concept for silhouette-based image understanding, IEEE


