GEOMETRIC AND PHOTOMETRIC MERGING FOR LARGE-SCALE OBJECTS

大規模観測対象のための幾何および光学情報の統合

by

Ryusuke Sagawa
佐川 立昌

A Doctoral Dissertation
博士論文

Submitted to
the Graduate School of
the University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Engineering
in Information and Communication Engineering

Thesis Supervisor:
Professor Katsushi Ikeuchi
池内 克史 教授

ABSTRACT

In this thesis, we consider the geometric and photometric modeling of large-scale and intricately shaped objects, such as cultural heritage objects. When modeling such objects, new issues occurred during the modeling steps which had not been considered in the previous research conducted on the modeling of small, indoor objects. Since the entire data of these objects cannot be obtained by a single observation, the main theme of this thesis is the "merging" of multiple observations.

The first issue is creating a detailed model from a huge amount of data. When modeling a large-scale and intricately shaped object, a huge amount of data is required to model the object. We would like to propose three approaches to handling this amount of data: the parallel processing of merging range images, the effective nearest neighbor search, and the adaptive algorithm of merging range images. We developed a parallel computation algorithm using a PC cluster which consists of two components, the distributed allocation of range images to multiple PCs, and the parallel traversal of subtrees of octree. We also propose a novel method for searching the nearest neighbor by extending the search algorithm using a k-d tree. We constructed a merged model in adaptive resolution according to the geometric and photometric attributes of range images for efficient use of computational resources.

The second issue involved in handling a large amount of data is the merging of the photometric attributes of range images. We reconstructed a 3D model with an appearance which successfully discarded outliers due to specular reflection, by taking a consensus of the appearance changes of the target object from multiple range images. The photometric attribute of the model can be used for aligning with 2D color images.

The third issue is constructing a precise model using noisy data. We propose an efficient method to refine range images using multiple observations from various viewpoints. The method refines range images during iterative computation by assuming that the distribution of their error is anisotropic according to the viewing direction.

The fourth issue is the need for a new range sensor to cover unobservable surfaces from the ground. We developed a new range scanning system, which obtains a dense range image by stereo matching. It is loaded on a balloon, and scans large-scale objects from the air. The system consists of 9 digital steel cameras, which together construct a high resolution model.

The fifth issue is complementing unobservable surfaces of an object. We propose a novel method to complement such holes or gaps in the surfaces, areas which are not observed by any scans. In this novel method, the surface of an object is scanned using various kinds of sensors. We efficiently complement them by taking a consensus of the signed distances of neighbor voxels.

論文要旨

３次元形状を計測する従来の研究では，観測対象が室内に置かれた物体などの比較的小さな物体であったが，文化遺産の中には建物や大仏などの大規模な構造物を観測対象とする場合，モデリングの各段階において様々な新しい問題が発生する．そのような観測対象を行う場合，１回の観測で全てのデータを取得することは不可能であるので，本論文における主題は複数の観測をいかに「統合」するか，ということになる．

まず，観測対象が大規模かつ複雑な形状を持つ場合，大量のデータを取り扱う必要がある．本論文では以下に提案する３つの手法を用いて大規模なデータの統合を行う．第一に PC クラスタを用いた並列計算により計算時間を削減する．この手法は複数の PC に距離画像を分散配置する方法と，オクトツリーの部分木を並列に探索する方法から構成される．これにより PC クラスタの多数の CPU とメモリが利用可能になった．第二に k-d tree の探索方法を拡張し，効率的に最近傍点を探索する方法を開発し，距離画像の統合過程における計算コストの大部分を占める最近傍点探索の高速化が可能になった．第三に観測対象の形状やテクスチャに応じて適応的に距離画像の統合を行うことにより効率的に計算資源を利用する方法である．

次に距離画像に付加された光学情報の統合する手法を提案する．複数の距離画像に付加された光学情報の合致をとることにより鏡面反射成分による光学情報の外れ値を取り除き，距離画像統合の枠組みにおいて光学情報の統合を行う．光学情報付きモデルは２次元カメラ画像との位置合わせに応用することができる．

またノイズが多い距離画像を用いて精確なモデルを生成する手法を提案する．距離画像に含まれるノイズの分布が視点位置に依存することを仮定し，様々な方向から計測された距離画像を利用して反復計算により距離画像のノイズの除去を行う．

大規模な観測対象の場合には地上から観測できない部分が発生する．そのような部分を観測するためにデジタルスチルカメラ 9 台から構成されるシステムを気球に搭載し，ステレオマッチングによって距離計測を行うセンサを開発した．カメラを用いることにより空中で静止していない状態においても距離計測が可能になった．

様々なセンサを用いて計測を行った場合においても計測できない部分が発生するため，そのような形状データの欠落を補う方法を提案する．本手法では統合過程において符号付距離を計算しているが，その符号付距離の整合性を近傍のボクセル間でとることによりデータの欠落を補間する．

これらの手法を用いていくつかの文化遺産についてモデリングを行い，大規模かつ複雑な形状をもつ観測対象のモデリングが可能になったことを示す．

Acknowledgements

I would first like to thank my advisor, Professor Katsushi Ikeuchi, who gave me the opportunity to study the challenging Great Buddha Project. I would like to thank him for giving me the freedom to develop various new ideas, and for his willingness to give me a great deal of advice, when needed. I had the opportunity to visit a number of historic sites in order to scan cultural heritage objects for this project, including Kamakura, Nara, Kyoto, and Maebashi in Japan, Sukhothai and Ayutthaya in Thailand, and Angkor Wat in Cambodia. I had the chance to see many cultural heritage objects on these trips, including objects which ordinary people don't have the opportunity to see. That was a great experience for me, and our team was able to obtain valuable data on those objects. I would like to thank the staff of Koutoku-in, Todai-ji, and the National Research Institute for Cultural Properties in Japan, and the Department of Fine Art of Thailand, along with all of the people who helped us scan objects.

I would also like to thank my colleagues. Ryo Kurazume, Atsushi Nakazawa, and Ko Nishino deserve special thanks for discussing my study with me, and for giving me suggestions and advice. I owe the success of my thesis to their frequent help. Takeshi Oishi worked with me in scanning and in developing software, and his software applications were very useful in our research. I greatly appreciate the hard work of Tomohito Masuda. My thanks also go to Hiroshi Kawasaki and Shutaro Yamazaki, with whom I developed new ideas, and had an enjoyable time while doing so. I would also like to thank Auethavekiat Supatana for her help in developing this project.

I would like to acknowledge the other members of the CVL, the Geometry Group, the Photometry group, the Robot group, the Motion group, the ITS group, and the faculty for these groups. Koichi Ogawara, Masataka Kagesawa, Yoichi Sato, Kiminori Hasegawa, Kentaro Kawamura, Tomikazu Tanuki, Jun Takamatsu and Ryo Ohkubo are especially acknowledged for their help. I greatly appreciate all of the secretaries in the CVL for

keeping everything going. Also, I have greatly enjoyed my life in the CVL, including the stimulating discussions, hard work, lively dinner parties, and so on. These have been precious experiences for me.

Finally, I would like to thank my family and friends for their support during my many years of study at Kyoto University and The University of Tokyo. Most of all, the patience, support, and advice of my parents was of great value to me.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The modeling of the shape and appearance of objects in the real world is an important issue in many fields. If we create a model of a real object by observing it, we can see the model on computer through the techniques of computer graphics. By analyzing the created model instead of the real object itself, we can obtain a lot of information about the object from studying the model. If the object is an artificial object, we can apply the information obtained from studying the model to reverse engineering. If we create a new object by editing the model using a CAD system, we can use the model obtained from observation as an initial model, which is easier than creating a model from scratch. In the research area of intelligent robotics and computer vision, such models are used in the automatic recognition of objects, such as tracking the motion of an object and distinguishing an object from others.

Cultural heritage objects are one of the worthiest candidates to model in shape and appearance. There are three advantages to modeling these objects: presentation, preservation, and restoration. There are many cultural heritage objects in various places throughout the world. Thus, we can only see the objects in photographs and movies, unless we actually visit the sites where they can be found. However if we have models of these objects, we can learn much more about the object by examining it from various viewpoints. In addition, many cultural heritage objects are in danger of collapse due to the effects of weather, or because of accidents or willful destruction. By modeling cultural heritage objects, we can preserve information on them. Also, in case of their collapse, the models can provide virtual information to assist restoration specialists.

Among various kinds of cultural heritage objects, we have blueprints for buildings, and can use these to reconstruct the shape of the objects. However, it is an exception to be able to obtain blueprints of other cultural heritage objects, such as statues created by carving and casting. Moreover, some cultural heritage objects are large-scale objects, and their shapes consist of delicately and intricately curved surfaces.

1.1 Previous Work

When modeling the shape of an object, several approaches have been considered according to the scale of the target object. In Figure 1.1 we classified several approaches. Images observed by satellites [1] and airplanes [56] are used for the modeling terrain. The terrain geometry is measured by the photogrammetric approach using multiple images. A synthetic aperture radar (SAR) [26] can directly measure the terrain geometry by microwave imaging techniques. Another approach to measuring the terrain geometry is airborne laser scanning [48]. This method measures the distance from the airplane to the target by a shooting laser. Since these approaches have been developed to measure the terrain geometry, they can observe a very large area; however, the resolution and precision is lower than that of the measuring equipment used to measure small objects.

Our target objects were mainly outdoor objects, such as statues of the Great Buddha and ancient temple buildings. High resolution and high precision are required for modeling these objects, just as when modeling small objects. Several approaches have been developed to measure small objects, for example a laser range finder and a structured-light range finder. They can measure with high resolution and precision; however, the area which they can scan is small. Thus, these approaches could not be directly applied to our target objects. Because our targets were larger and more intricate than the usual targets for these sensors, however, we had several problems to solve.

It costs too much to model the shape of cultural heritage objects manually, by photogrammetric methods. Therefore, these methods are not practical to use for objects with a highly intricate shape. However, 3D digitizing techniques using laser light have been recently developed. In these techniques, the measuring device, which is called a laser range finder, measures distance by shooting a laser and receiving its reflection from the target object. The distance is computed by measuring the time duration between the laser shot and when

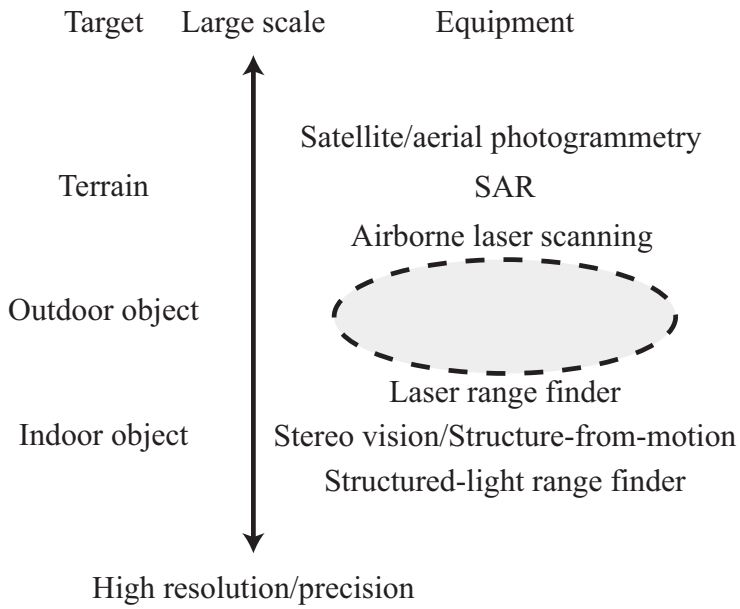


Figure 1.1: Modeling approach for objects of different scales

it is received back, or by triangulation between the shooting point and the receiving point. Laser range finders scan 2D dimensional areas by changing the direction of the laser by moving mirrors. Thus, the acquired 2D data is called a range image. Since some of their measuring ranges are over 50 meters, we can utilize these range finders for modeling cultural heritage objects. So far, several researchers [51, 39, 44, 59] have studied the modeling of cultural heritage objects using those sensors.

Other methods for measuring the shape of objects are automatic photogrammetric techniques such as dense stereo matching [69, 15], and methods of projecting structured light [81, 36]. Both methods compute distance by triangulation. Stereo matching is passive measuring by observing the texture of the surface of an object using multiple cameras. Because of the passiveness, the precision of range measurements are worse than those of laser range finders and structured-light sensors. However, the advantage of this method is that the images are acquired in less than a second. A structured-light method projects various structured lights and encodes 3D space uniquely by these lights. Since the lights are distinguished by a camera, this system is not suitable for a bright environment such as outdoor scanning.

As most range sensing systems, e.g., stereo, structured light, and laser range finders return range images obtained from particular viewing points, each output range image covers only a portion of the target object surface. To ensure that all of the 3D shape data is acquired during the scanning process, several range images must be taken of each portion of the object surface from various viewpoints. Thus, the issue of geometric modeling is creating the entire model of an object from multiple range images. The "merging" of multiple observations into a unified model is the main issue considered in this thesis.
In previous research on geometric modeling, the researchers dealt only with small objects. Figure 1.2 shows the modeling steps for a small object. The 3D modeling of the shape of the object is accomplished by performing the following three steps:

1. Acquiring the range images (Scanning).

2. Aligning of the many range images acquired from different viewpoints (Aligning).

3. Re-generating the unified meshes (Merging).

In the first step, an object is easily observed from various viewpoints by rotating it on a turntable or mounting it on a robot arm. This is possible because small objects are small enough to handle. Also, it is easy to acquire the entire surface of a small object, since the scanning cost is negligible. If an occluded region of the surface exists, we can observe that region by rotating the object and scanning it one more time.
Second, the aligning of multiple range images is accomplished by recording each local coordinate system a priori, e.g., by using motion controlled stages, mounting range finders on the robot arms or by establishing point correspondences and minimizing the total distance between those points, e.g., feature-based methods [90, 45], ICP-based methods [5, 6, 24, 25, 72, 63], etc. Besl and McKay [6] proposed a feature point-based matching method, while Chen's method [11] is based on the evaluation between the point and the polygons. Neugebauer proposed the idea of 'simultaneous registration' that aligns range images simultaneously to avoid the error accumulation of the pairwise alignment methods [63]. A similar idea was proposed by Nishino et al. [66].
In the third step, for merging multiple pre-aligned range images, several approaches have been proposed. Turk and Levoy [95] proposed a method to "zipper" two range images at a time, by first removing overlapping portions of the meshes, clipping one mesh against

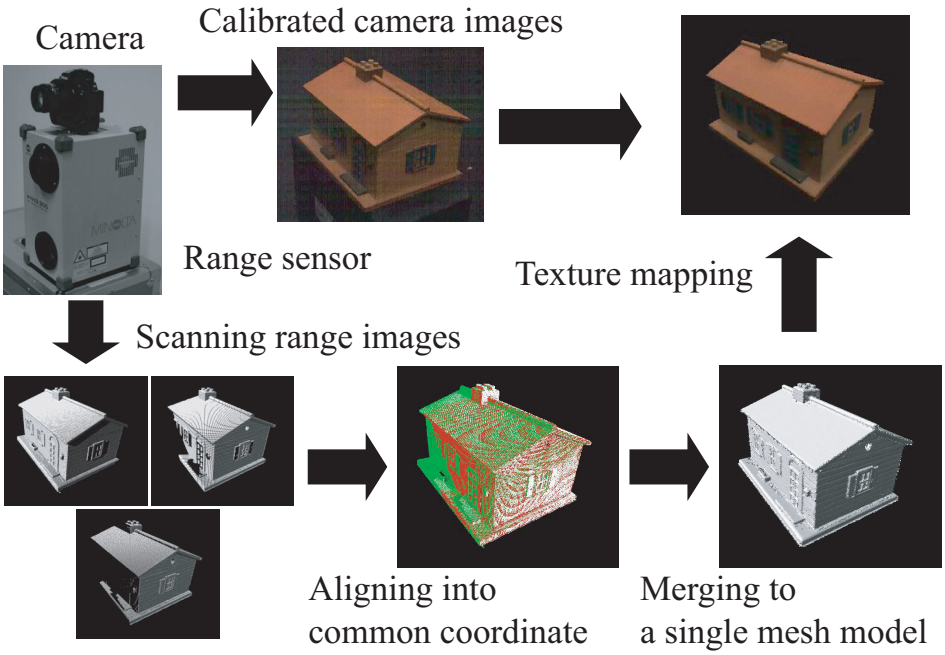


Figure 1.2: Steps of geometric and photometric modeling of a small object.

another, and then re-triangulating the mesh on the boundary. Although this is an intuitive process to merging two range images, pairwise merging does not work when merging multiple range images. Given a number of range images overlapping each other, a merging procedure which extracts the isosurface is necessary. Merging methods that make use of volumetric, implicit-surface representation and the marching-cubes algorithm [53] are suitable for this purpose. Hoppe et al. [43] constructed 3D surface models by applying the marching-cubes algorithm to a discrete, implicit-surface function generated from a set of range images. After inferring local surface approximations from clouds of points based on tangent plane estimations, a local search was accomplished to compute the signed distance from each voxel to the surface of the point set. Curless and Levoy [17] enhanced Hoppe's algorithm in a few significant points. However, none of these methods, including several others [40], compensate for noise or extraneous point data; the data is assumed to be part of the object, and noise is assumed to be negligible. Each of these methods suffers from inaccuracy due to their integration strategy, e.g., integrating unrelated observations, and these accuracy problems will affect the result even when the data is noise-free. Whitaker [98]

proposed a level-set approach for integrating range images, which introduced a smoothness constraint using a Bayesian formulation. Wheeler et al. [96, 97] addressed these important problems by designing a consensus surface algorithm. The consensus surface algorithm attempts to justify the selection of observations used to produce the average by finding a quorum or consensus of locally coherent observations. This process successfully eliminates many troublesome effects of noise and extraneous surface observations, and also provides desirable results with noise-free data.

As for photometric modeling of an object, color images are captured by a pre-calibrated camera. Since the mutual posture of the camera and the range finder is calibrated, the images can be directly mapped as texture on the 3D model. To calibrate the camera, a marker object is usually used, which has some features that can be recognized by a camera and range finder. The system acquires a color image and a range image of the marker at the same time. The 3D positions of features are obtained by the range finder. After taking matching 3D features and 2D features, which are obtained by the camera, the camera parameters are computed using the techniques of camera calibration [94, 102]

1.2 Thesis Overview

Previous research which studied the modeling of the 3D shape of objects considered the modeling of small, indoor objects. And yet some cultural heritage objects, like statues of the Great Buddha, ancient temple buildings, and bas reliefs on walls are quite large, and their shapes are intricate. In order to model such objects, various issues need to be solved at each stage of the modeling process. This thesis proposes new techniques for solving the following issues:

1. Creating a detailed model from a huge amount of data
2. The merging of photometric attributes of range images
3. Constructing a precise model using noisy data
4. Covering the entire surface of an object
5. Complementing unobservable surfaces of an object

Figure 1.3 shows the new modeling process, which has been developed after considering these issues.

Previously, it was not necessary to consider the amount of data and the computational costs of compiling this data, because the shape of the objects modeled were simple, and only a small number of range images were required. However, to create a detailed model of a large-scale object, such as a cultural heritage object, a huge amount of data is needed. Therefore, this paper proposes new methods for creating a detailed model from a huge amount of data, during the step in which range images are merged. The methods consist of the following three approaches:

- The parallel processing of merging range images
- A new technique of nearest neighbor search
- An adaptive algorithm of merging range images

When an object is large and far away from the sensors, it is difficult to calibrate between the camera and range finder. That is because we have to use a marker for calibration that is a similar size as the object, in order to sufficiently reduce the error of calibration. If we used this approach when modeling an object the size of a Great Buddha, we would have to use a marker of the same size, which is obviously not practical. Therefore, we propose a new method which merges the photometric attributes of range images in the framework of merging range images. The photometric attributes attached with the 3D model are utilized in calibrating with color images.

An issue we considered is that the size of our target objects varied from 1 meter or less to 100 meters, and we did not have a suitable sensor for every object of these various sizes. Thus, in some cases, we had to choose sensors that did not have sufficient precision to acquire the intricate shape of objects. Therefore, we propose a novel method which reduces the measurement errors of range images using multiple observations. Each range image has an anisotropic error distribution which depends on the viewpoint from which the object is scanned. If there is an overlapping area of range images after aligning into the common coordinate system, our new method refines the range images by taking a consensus of them.

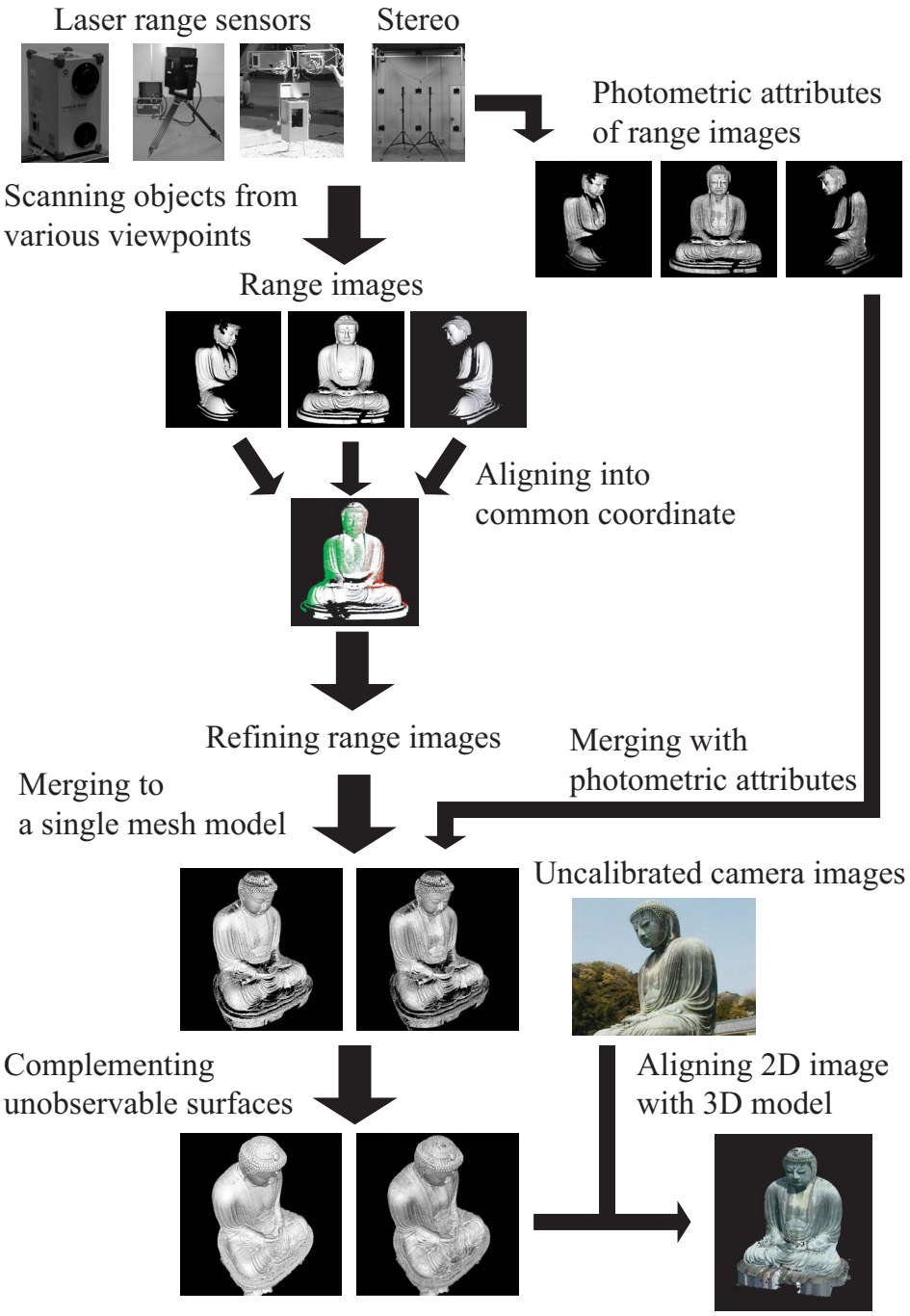


Figure 1.3: Extended modeling steps for modeling a large-scale, intricately-shaped object

Another issue we considered is that it is difficult to observe the entire surface of a large-scale object. Our main sensor was a laser range finder which was set on the ground, and it was heavy, weighing more than 20kg. Consequently, one of the largest portions of unobservable surface for that sensor is the top of the target object. Thus, we developed a new sensor that scans the area which cannot be observed from the ground. This sensor is a stereo range finder which consists of multiple cameras, and which is loaded on a balloon. There are often unobserved surfaces, even when full use of various kinds of sensors is made. If the object has an intricate shape, many occluded areas can occur. However, it is too costly to cover every hole which is not observed by any range images. In the worst case scenario, we are not able to obtain the data needed to fill those holes or gaps. However, we want to fill such holes to make use of the constructed models in many different applications. Therefore, we propose a new method to complement the geometric and photometric models of the holes by estimating the neighborhood area of the holes.

1.3 Merging a Huge Amount of Range Images

This paper proposes a new merging algorithm based on Wheeler's method [97]. We extended the method for creating a detailed model from a huge amount of range images. The proposed method consists of the following three approaches:

- The parallel processing of merging range images
- A new technique for conducting a nearest neighbor search
- An adaptive algorithm of merging range images

1.3.1 Parallel Merging of Range Images

The first approach is parallel processing of the merging algorithm. Our merging algorithm first converts range images to volumetric representation, which is called a signed distance field (SDF) [31]. Then, we convert the volumetric representation to a unified mesh model by extracting the isosurface based on marching-cubes algorithm [53] (we will abbreviate it as MC throughout the rest of this paper). Though several parallel processing algorithms of MC [2, 54] have been proposed, the most time-consuming step in the merging algorithm

is the computation of a signed distance. As a choice for reducing the computational time, a PC cluster is widely used. It is composed of multiple PCs connected by a network such as an ethernet or Myrinet [60]. Since each PC is inexpensive, a PC cluster is a reasonable system to use compared with shared-memory workstations which have the same number of CPUs. Thus, we developed a new technique of parallel computation of signed distances by making use of a PC cluster. This technique consists of the following two components:

1. Distributed allocation of range images to multiple PCs
2. Parallel traversal of subtrees of octree

In Chapter 2, we first describe Wheeler's merging algorithm, which our method is based on. Next, we will explain how we extended the method for parallel processing with a PC cluster in the proposed method.

1.3.2 Effective Nearest Neighbor Search

To compute the signed distance from a voxel, the merging process finds the nearest neighbor points of range images for each voxel. The original merging algorithm uses the k-d tree [4, 30], which is one of the most widely used structures for searching for nearest neighbors. It is a kind of binary tree that partitions space using hyperplanes that are perpendicular to the coordinate axes. If a k-d tree consists of n records, the k-d tree requires $O(n \log_2 n)$ operations to construct and a $O(\log_2 n)$ to search.

A case in which a search finishes in $O(\log_2 n)$ is an ideal case, in that only a leaf node is examined, and the nearest neighbor belongs to it. However, a search using a k-d tree does not actually finish in logarithmic time. In many cases, a search needs to examine several leaf nodes before finding the nearest neighbor point. In the worst case scenario, all leaf nodes must be examined. When the nearest neighbor is far from a query, the number of leaf nodes is apt to increase.

However, if the nearest neighbor point of a range image is far from a voxel, the voxel is used for extracting the isosurface by MC. Namely, the nearest neighbor point is not important for merging. It is sufficient that we find out that there are no range images near the voxel. Therefore, we introduced a novel test that takes place during the traversing of

the k-d tree. This test compares the distance from a query to the nearest neighbor, with a threshold defined by the user. We explain this technique in Chapter 3.

1.3.3 Adaptive Merging Algorithm

The third approach is adaptive integration of the range images according to the shape of the object. The original algorithm produces a mesh model of the finest resolution everywhere; however, the dense sampling for generating a mesh model is not necessary where the shape of the object is near planar. Thus, we propose an algorithm to construct the 3D model in an efficient representation. By taking the surface curvature into account when splitting the voxels recursively in an octree manner, the resulting 3D surface will be subdivided more in high curvature areas and less in surface areas that are near planar. Therefore, the resulting geometric model will require less triangular patches to represent the object surface. This is similar to research on mesh model simplification algorithms based on surfaces [33, 41, 42], while we reconstructed a simplified 3D model through a range image merging process based on implicit surface representation.
The simplification is done when splitting voxels recursively, enabling better preservation of the topology and mass of the object compared with the results of other volume based simplification methods [86, 87]. Frisken et al.[31] proposed adaptive sampling of the signed distance field; however, their method does not produce a triangular mesh model. For converting the volumetric representation of the 3D model to a triangular-based mesh model, we propose an extended version of the marching-cube algorithm; this version handle voxels in different resolutions. The proposed method is described in Chapter 4.

1.4 Merging Photometric Attributes of Range Images

Based on Wheeler's algorithm to robustly integrate multiple range images, we propose a range image integration method which can construct 3D models with photometric attributes. Considering applications that utilize geometric models, for instance, 3D object recognition and localization tasks, it is desirable to construct 3D models with additional attributes such as color and intensity. With the additional information provided by photometric attributes, higher accuracy and robustness can be expected from those applications. By taking a consensus of the appearance changes of the target object from multiple range

images, we reconstructed the 3D model with appearance values attached per vertex, successfully discarding outliers due to noise produced in the image-capturing process. This algorithm, taking a consensus of photometric attributes, can also be used to derive the rigid part of the appearance change on the object surface, providing a 3D model with Lambertian reflected light values under a static illumination environment. In some sense, our method is analogous to "Voxel Coloring" [84, 16, 50], where photometric consistency is used to carve a volume to reconstruct a geometric model with texture. Our framework goes the opposite way; we already have 3D information; from it, we construct a photometric consistent geometric model. The method is described in Chapter 5.

1.5 Refining Range Images

The errors of the final 3D model are considered to come from the following factors:

(a) A measurement error on the range images

(b) A matching error on the aligning process

(c) A quantizing error on the merging process

The type (b) and (c) errors depend on the object shape and the algorithm, and so are solved by selecting a suitable algorithm according to the object. For the type (a) error, Taubin [91] used the spatial smoothing filter, but fine features can be lost during this procedure. Basically, this kind of error cannot be avoided from one range image by using any software algorithms.

Taking many range images of the same surface is one of the solutions for this problem. Generally, some range finders improve measurement accuracy by multiple observations. Wheeler's method [97] reduces the error using multiple range images based on this consideration, but is weak if range images have high frequent errors. The signed distance is calculated along the normal direction of the surface. If the normal directions are not reliable because of a measurement error, then the final merging result is also not reliable.

We propose a new method to avoid this weakness and improve the accuracy of the final 3D model. This reduces the measurement errors on the distance value in the overlapping areas of the aligned range images. By applying this method before the merging process,

we extract a more precise model by taking a consensus of the range images. Unlike the existing spatial filtering method, our method is able to not only smooth the surface of the final 3D mesh model, but also to extract the fine features. The method is described in Chapter 6 in detail.

1.6 Flying Stereo Range Finder

To solve the issue of covering the unobservable surface from the ground, we developed a new range scanning system, which obtains a dense range image by stereo matching. It is loaded on a balloon, and scans large-scale objects from the air. If a range finder is loaded on a balloon, it will move because of wind or other reasons. Therefore, two approaches can be considered. First, we can use a sensor whose acquisition is finished at a moment at which we can ignore the motion of the sensor. Second, we can compensate for the motion by measuring the motion using other sensors. In this paper, we chose the first approach, which is the simpler one. The precision of a stereo range finder is lower than that of a laser range finder; however, it has the great advantage of speed, in that the acquisition of the stereo range finder is completed in less than a second, while a typical laser range finder takes about 10 seconds to several minutes for each scan.

Though stereo matching can be accomplished by 2 cameras, the multi-baseline stereo matching method [69] improves the precision of range sensing, which takes matching by multiple pairs of cameras using more than 3 cameras. Thus, we constructed a system consisting of 9 cameras. Chapter 7 describes the system we developed.

1.7 Complementing Unobservable Surfaces

Even after scanning an object using various kinds of sensors, there may still be unobserved surfaces. As an approach to fill these scanning holes, we developed a new stereo range finder, which will be described in Chapter 7. We propose one more alternative approach to filling these holes. It complements the geometric and photometric model of the object by estimating the neighborhood area of the holes.

Since filling the holes of a modeling is a major issue in this field, several approaches have been proposed. The simplest approach is interpolation by triangulating the bound-

ary vertices of a hole. If a hole is small and the topology is simple, the triangulation works well; however, triangulation becomes difficult if the surface is intricate and the hole is large. The second approach is fitting a mesh model around the hole, that is, a three-dimensional version of snakes [46]. In these methods [22, 12], a deformable surface moves iteratively to fit the model with a satisfying smoothness constraint. The third approach is called "space carving". Curless and Levoy [17] tagged one of the states, unseen, empty and near the surface, to each voxel during the merging process. The hole filling is accomplished by generating a surface between voxels of unseen state and ones of empty state. The fourth approach is interpolation by volumetric representation, such as level-set approaches [98, 103] and re-computation of implicit surface [10, 20].

Our method is similar to the third and fourth approaches. In our merging framework, we have already computed the SDF; however, the sign of the SDF becomes sensitive around the holes. Thus, we take a consensus of the sign of the SDF with those of the neighbor voxels. We describe this technique in Chapter 8.

Chapter 2

Parallel Processing of Merging

2.1 Consensus Surface Algorithm

In this chapter, we first describe the basic merging algorithm [97], which is called the "consensus surface algorithm."

2.1.1 Signed Distance Field

This method first constructs a volumetric representation, which is called a signed distance field (SDF), using all range images. Those range images are assumed to be already aligned into a common coordinate system. In this volumetric representation, 3D space is partitioned into a three dimensional grid of cubes. Each cube is called a"voxel" (volume element). A voxel has a signed distance $f(\boldsymbol{x})$ from the center of the voxel $\boldsymbol{x}$ to the nearest surface. The sign of $f(\boldsymbol{x})$ is positive if the center $\boldsymbol{x}$ is outside the object; it is negative if the center $\boldsymbol{x}$ is inside the object. Because the surface of the object is represented implicitly by $f(\boldsymbol{x}) = 0$, $f(\boldsymbol{x})$ is called the implicit surface function. Figure 2.1 shows a 2D slice view of an example of SDF, which is composed of 9 voxels. If the surface is converted to SDF, the $f(\boldsymbol{x})$ of each voxel is computed as shown. The voxels inside the object are dark gray, and the ones outside the object are white. Consequently, the issue of merging range images is how we compute the implicit surface function $f(\boldsymbol{x})$ of each voxel using multiple range images.

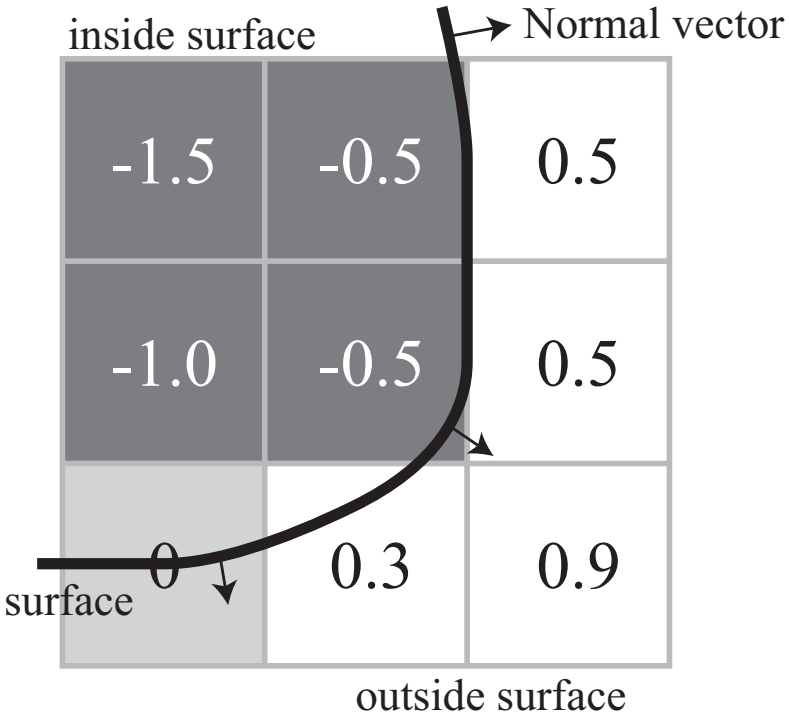


Figure 2.1: An example of implicit surface function computed from an explicit surface.

2.1.2 Taking Consensus of the Range Images

For computing the signed distance $f(\boldsymbol{x})$ of an arbitrary point $\boldsymbol{x}$ using multiple range images, the simplest solution is to find the nearest neighbor point of range images from $\boldsymbol{x}$. Figure 2.2 shows an example of a situation in which there are three range images which are intersecting between two neighboring voxels. The center points of the two voxels are $\boldsymbol{x}$ and $\boldsymbol{x}'$. Each range image consists of 3D vertices and triangles that connect the neighboring vertices. The normal vectors of the range images in Figure 2.2 look outside the object. In the simplest solution, it finds the nearest neighbor point of all range images from the center of each voxel. The magnitude of the signed distance $|f(\boldsymbol{x})|$ is the distance from the center $\boldsymbol{x}$ to the nearest point $\boldsymbol{p}$. If $(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n} > 0$, where $\boldsymbol{n}$ is the normal vector of $\boldsymbol{p}$, the sign of $f(\boldsymbol{x})$ is positive; otherwise, the sign of $f(\boldsymbol{x})$ is negative. In the situation of Figure 2.2, since $\boldsymbol{x}$ is outside the object and $\boldsymbol{x}'$ is inside, it should be $f(\boldsymbol{x}) > 0$ and $f(\boldsymbol{x}') < 0$. However, $f(\boldsymbol{x})$ becomes negative because $(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n} < 0$. Such a misjudgment occurs

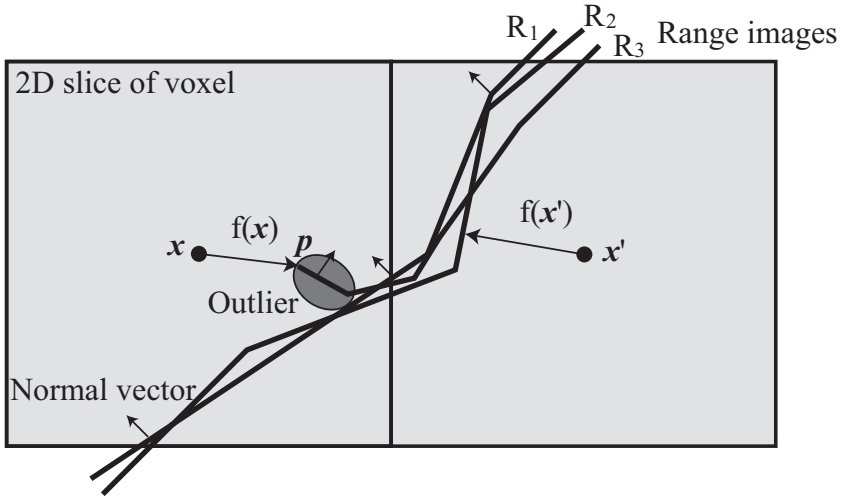


Figure 2.2: The incorrect sign of $f(\boldsymbol{x})$ occurs due to a single noisy triangle in the simple solution.

since the nearest point $\boldsymbol{p}$ is an outlier caused by the error of scanning. Consequently, the implicit surface of the object is not generated between these two voxels.

To solve the problem, the consensus surface algorithm first finds the nearest point for each range image respectively. In Figure 2.3, the point $\boldsymbol{p}_0$ is one of the nearest points from $\boldsymbol{x}$. Next, it searches the nearest points of the other range images from the point $\boldsymbol{p}_0$. In this case, the points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are found as the corresponding points. The algorithm compares the position and normal vector of the nearest point from $\boldsymbol{x}$ and those of its corresponding points. Now, the normal vector of $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ is $\boldsymbol{n}_0$ and $\boldsymbol{n}_1$. If the following predicate is satisfied, the surface $\langle \boldsymbol{p}_0, \boldsymbol{n}_0 \rangle$ and $\langle \boldsymbol{p}_1, \boldsymbol{n}_1 \rangle$ are regarded as the same surface,

$$\textbf{SameSurface}(\langle \boldsymbol{p}_0, \boldsymbol{n}_0 \rangle, \langle \boldsymbol{p}_1, \boldsymbol{n}_1 \rangle) = \tag{2.1}$$
$$\begin{cases} \text{True} & (\parallel \boldsymbol{p}_0 - \boldsymbol{p}_1 \parallel \leq \delta_d) \wedge (\boldsymbol{n}_0 \cdot \boldsymbol{n}_1 \geq \cos\theta_n) \\ \text{False} & \text{otherwise} \end{cases},$$

where δ_d is the maximum allowed distance and θ_n is the maximum allowed difference in normal directions. In Figure 2.3, $\langle \boldsymbol{p}_0, \boldsymbol{n}_0 \rangle$ and $\langle \boldsymbol{p}_1, \boldsymbol{n}_1 \rangle$ are the same surface; however, $\langle \boldsymbol{p}_0, \boldsymbol{n}_0 \rangle$ and $\langle \boldsymbol{p}_2, \boldsymbol{n}_2 \rangle$ are not regarded as the same.

Next, the algorithm uses the weighted voting scheme to discard isolated surfaces. It computes the confidence ω by summing up the confidence of each surface which is regarded

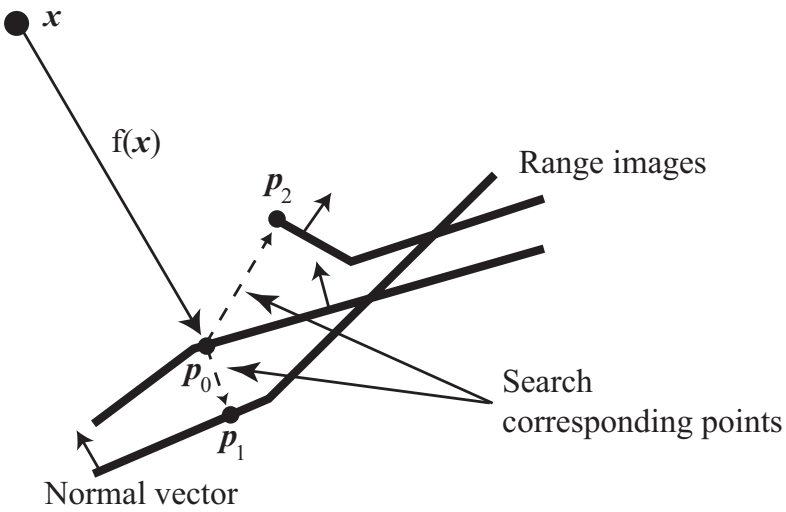


Figure 2.3: Taking the consensus of range images: First, we find the nearest neighbor point from the center of a voxel (the solid arrow). Second, we find the nearest neighbor points of the other range images from the first nearest neighbor point (the dotted arrows).

as the same. If ω is larger than the threshold θ_{quorum}, the surfaces have consensus; and the averaged surface is called "consensus surface". The confidence of the surface of a range image is proposed to be computed by $\boldsymbol{v} \cdot \boldsymbol{n}$ in [97], where $\boldsymbol{v}$ is the viewing direction. However, since the confidence of the surface of a range image depends on the range finder, we simply use the number of the same surfaces as ω.

Figure 2.4 shows that the consensus surface algorithm is applied to the same situation with Figure 2.2. It computes three signed distances from $\boldsymbol{x}$, which are already averaged among the same surfaces. In Figure 2.4, the consensus surfaces are A and B; the surface C does not have consensus. Thus, the final signed distance from $\boldsymbol{x}$ is chosen from A or B. Since the magnitude of A is smaller than that of B, A is chosen in this case. We can discard the isolated observation C. Therefore, $f(\boldsymbol{x})$ becomes positive and the implicit surface is successfully generated.

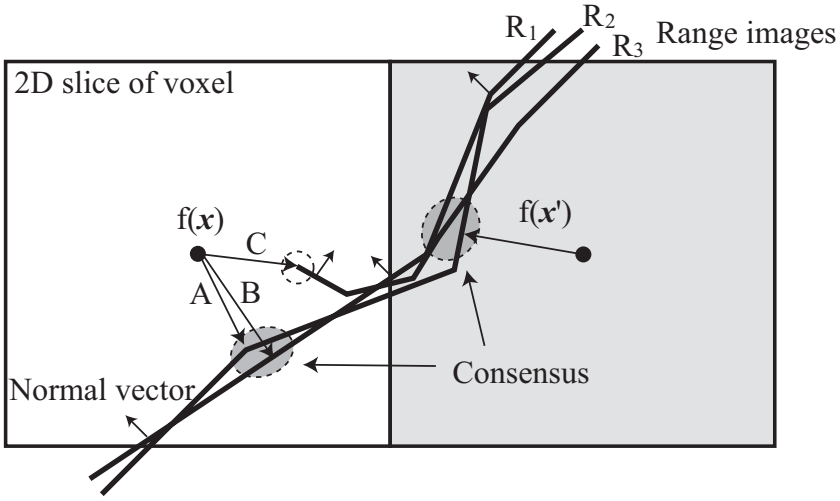


Figure 2.4: Consensus surface algorithm: The signed distance is chosen from the consensus surfaces inside the gray circle.

2.1.3 Subdividing Voxels Based on Octree

To find out where the implicit surface is, we have to compute the signed distances of all voxels around the zero level of the implicit function. It is costly to compute the signed distances of all voxels, since the computational cost is $O(n^3)$ if the volume of interest is uniformly divided into $n \times n \times n$ voxels along each axis. Curless and Levoy [17] efficiently traversed the volume by resampling range images along scanlines of voxels, since their method finds corresponding points on the screen space by projecting voxels and a range image. Level-set methods [98, 103, 85] use the narrow-band method to reduce the computational cost, which updates the finite band of voxels on either side of zero level. Wheeler [97] proposed the strategy of computing signed distances by subdividing the volume of interest recursively in an octree manner. It starts the entire volume as a voxel for computing the signed distance; it subdivides the voxel if the signed distance satisfies the following inequality,

$$|f(\boldsymbol{x})| < \frac{3\sqrt{3}}{2}w, \tag{2.2}$$

where w is the width of the voxel of interest. If (2.2) is satisfied, the implicit surface can exist inside the voxel or the neighbor voxels. It stops subdividing if the voxel becomes the

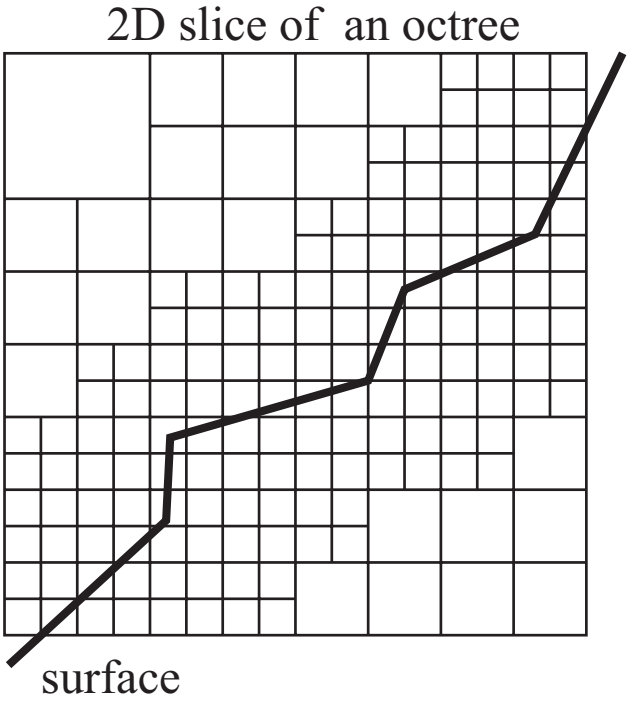


Figure 2.5: An example of a 2D slice of an octree subdividing volume. The resolution is high around the surface and low elsewhere.

finest resolution. Since the width of voxels which contains the implicit surface is the same (see Figure 2.5), MC [53] can be straightforwardly applied to the voxels subdivided in an octree manner. Subdividing voxels in an octree manner practically reduces the computational cost to $O(n^2)$, because the finest resolution voxels exist only near the surface.

2.1.4 Marching Cubes Algorithm

Though a volumetric representation such as SDF can be visualized by volume rendering [29], a mesh model is suitable for our goal, which is geometric modeling and the analysis of objects. Lorensen and Cline [53] proposed a method which converts the volumetric representation to a mesh model. The algorithm is called marching-cubes algorithm. The marching-cubes algorithm constructs a surface mesh by "marching" around the cubes which contain the zero level of the implicit surface $f(\boldsymbol{x}) = 0$. In Figure 2.6, the white and

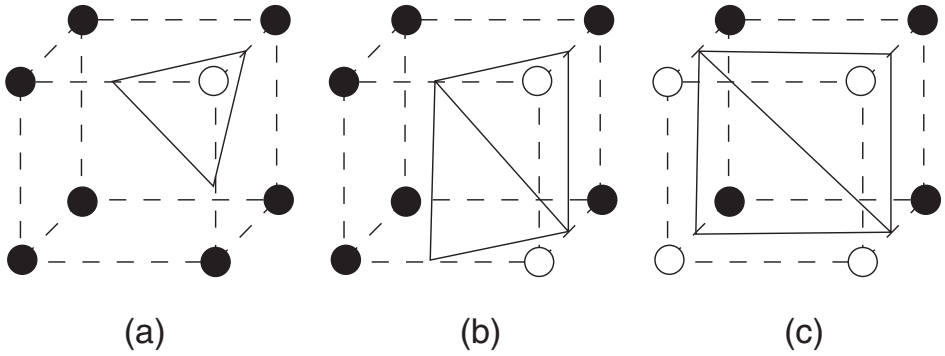


Figure 2.6: Marching Cubes: An implicit surface is approximated by triangles. ○: voxels of outside surface. ●: voxels of inside surface.

black circles are the center points of the voxels. The white ones mean the outside surface; and the signed distance is more than 0. Also, the black ones mean the inside surface; and the signed distance is less than 0. The marching-cubes algorithm is applied to each cube, which consists of 8 adjacent voxels. The surface triangles are generated to intersect between a black circle and a white one. The position of the vertex of a triangle $\boldsymbol{p}$ is on the edge of the cube; and it is interpolated by the following equation:

$$\boldsymbol{p} = \boldsymbol{x}_1 + \frac{-d_1}{d_2 - d_1}(\boldsymbol{x}_2 - \boldsymbol{x}_1), \tag{2.3}$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the center points of adjacent voxels, and d_1 and d_2 are signed distances of these voxels. The marching cubes algorithm uses a lookup table to determine how triangles are generated. Since the original algorithm [53] has ambiguity in the algorithm of generating triangles, Nielson and Hamann [65] proposed a method to resolve ambiguous cases.

2.1.5 Summary of Consensus Surface Algorithm

We summarize the consensus surface algorithm. First, Algorithm 2.1 shows the pseudo code of the algorithm to find the nearest surface from an arbitrary point $\boldsymbol{x}$ with the taking of a consensus of range images R_{set}. It searches the nearest surface of a range image by **SearchNearestNeighbor**($\boldsymbol{x}, R$), which is accomplished using a k-d tree [30]. The k-d tree is a structure for finding the nearest neighbor point in k dimensional space. If the

confidence ω of the found surface $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$ is larger than θ_{quorum}, let it into the set of consensus surfaces C_{set}; otherwise, let it into O_{set}. If $C_{\text{set}} \neq 0$, the nearest surface of C_{set} is chosen; otherwise, the most confident one of O_{set} is chosen.

Second, Algorithm 2.2 shows the algorithm to compute the signed distance of a node of octree N as the voxel of interest. The **TraverseOctree**$(N, d_{\text{max}}, R_{\text{set}})$ is recursively called with subdividing the current node of the octree. It starts the entire volume as a voxel and stops if $|v| < \frac{3\sqrt{3}}{2}w$ or the depth of the current node reaches the maximum depth d_{max}. At the maximum depth, the entire volume is divided into $2^{d}\text{max} \times 2^{d}\text{max} \times 2^{d}\text{max}$ voxels along each axis.

Algorithm 2.1 ConsensusSurface($\boldsymbol{x}, R_{\text{set}}$)

Input: Point: $\boldsymbol{x}$

Input: Set of Range Images: R_{set}

Local: Point: $\boldsymbol{p}, \boldsymbol{p}_1, \boldsymbol{p}_2$

Local: Normal Vector: $\boldsymbol{n}, \boldsymbol{n}_1, \boldsymbol{n}_2$

Local: Weight: $\omega, \omega_1, \omega_2$

Local: Set of $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$: $C_{\text{set}}, O_{\text{set}}$

Output: Tuple of Point, Normal and Weight: $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$

```
for all R ∈ R_set do
    p ← n ← ω ← 0
    ⟨p_1, n_1, ω_1⟩ ← SearchNearestNeighbor(x, R)
    for all R' ∈ R_set do
        ⟨p_2, n_2, ω_2⟩ ← SearchNearestNeighbor(p_1, R')
        if SameSurface(⟨p_1, n_1⟩, ⟨p_2, n_2⟩) then
            p ← p + ω_2 p_2
            n ← n + ω_2 n_2
            ω ← ω + ω_2
        end if
    end for
    p ← (1/ω) p
    n ← n / ||n||
    if ω ≥ θ_quorum then
        C_set ← C_set ∪ ⟨p, n, ω⟩
    else
        O_set ← O_set ∪ ⟨p, n, ω⟩
    end if
end for
if C_set ≠ 0 then
    ⟨p, n, ω⟩ ← arg min_{⟨p,n,ω⟩ ∈ C_set} || x − p ||
else
    ⟨p, n, ω⟩ ← arg max_{⟨p,n,ω⟩ ∈ O_set} ω
end if
return ⟨p, n, ω⟩
```

Algorithm 2.2 TraverseOctree($N, d_{\max}, R_{\text{set}}$)

Input: Current Node of Octree: N

Input: Maximum Depth of Octree: $d_{\max}$

Input: Set of Range Images: R_{set}

Local: Center of N: $\boldsymbol{x}$

Local: Octree Depth of N: d

Local: Width of N: w

Local: Tuple of Point, Normal and Weight: $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$

Output: Signed Distance of N: v

$\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle \leftarrow \mathbf{ConsensusSurface}(\boldsymbol{x}, R_{\text{set}})$

if $(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n} > 0$ **then**

$v \leftarrow \| \boldsymbol{x} - \boldsymbol{p} \|$

else

$v \leftarrow - \| \boldsymbol{x} - \boldsymbol{p} \|$

end if

if $|v| < \frac{3\sqrt{3}}{2} w \wedge d < d_{\max}$ **then**

for all children $N_i (i = 0, \ldots, 7)$ of N **do**

TraverseOctree($N_i, d_{\max}, R_{\text{set}}$)

end for

end if

2.2 Parallel Computation of Signed Distance

We scanned a cultural heritage object, the statue of the Great Buddha of Kamakura, in Japan. When we tried to merge the range images obtained, we found that the size of the range images was too large to be process[ed by a single computer, because a greater memory capacity was needed, and the computational time was very long. Thus, we propose a parallel merging method to accomplish the merging of a huge size of range images [77]. We constructed a PC cluster for parallel processing and developed a parallel algorithm which consists of the following two methods:

- Distributed allocation of range images and the parallel computation of the nearest neighbor search
- Parallel subdivision and traversal of an octree

2.2.1 Distributed Allocation of Range Images

In Algorithm 2.1, **ConsensusSurface**($\boldsymbol{x}, R_{\text{set}}$) computes the nearest surface $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$ of each range image R from the center point $\boldsymbol{x}$ of a voxel by **SearchNearestNeighbor**($\boldsymbol{x}, R$). Since a range image includes a lot of vertices and triangles, the computational cost of searching the nearest surface is very high; it occupies most of the cost of the consensus surface algorithm. Moreover, we have to handle many range images when the target is a cultural heritage object. Thus, it is difficult for a single computer to allocate all of the range images. We propose a new method, which distributes the images to multiple PCs to allocate them, and computes the signed distances in parallel.

ConsensusSurface($\boldsymbol{x}, R_{\text{set}}$) executes **SearchNearestNeighbor**($\boldsymbol{x}, R$) for each range image sequentially. Since each search is independent from the other searches, we can execute them in parallel. Figure 2.7 shows the parallel computation of finding the nearest surface in the same situation of Figure 2.4. If range images R_1, R_2 and R_3 are allocated by PC1, PC2 and PC3, respectively, PC1 computes the nearest neighbor points A and A' only for R_1. Similarly, PC2 finds only B, B' of R_1; and PC3 finds only C, C' of R_2.

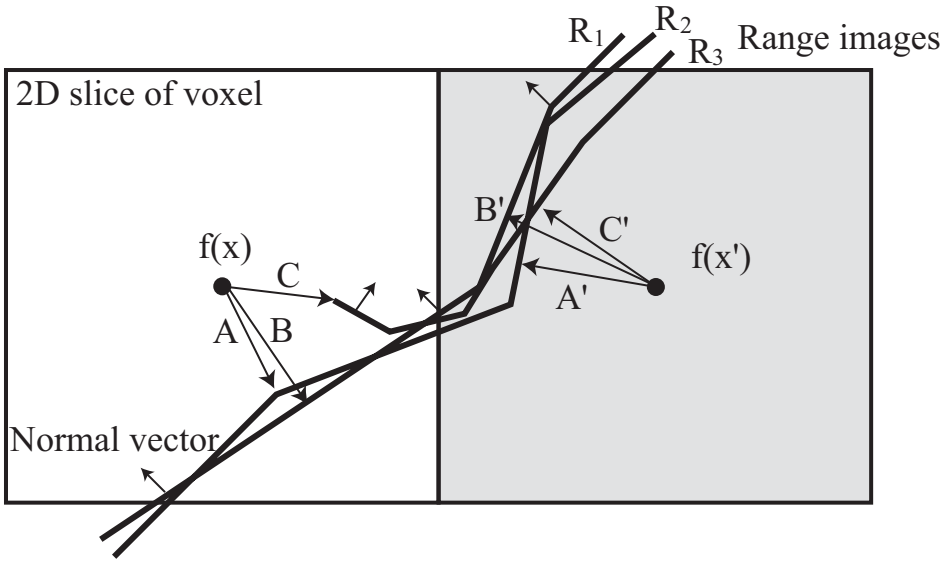


Figure 2.7: Distributed allocation of range images and parallel computation of signed distances

2.2.2 Parallel Subdivision and Traversal of Octree

Algorithm 2.2 computes the signed distance of each node of an octree sequentially by subdividing a voxel and traversing its children. Since the traversal of one of its children is independent from the traversals of the other children, we can traverse subdivided children in parallel. We assigned a partial space of the entire volume to each PC, as shown in Figure 2.8. A partial space is represented by a subtree of the octree. Since no synchronization is necessary while traversing subtrees and the results of computing signed distances are collected after the traversal of every subtree is completed, we can make full use of the power of the PCs. Thus, the performance of the parallel traversal algorithm increases almost linearly according to the number of PCs used.

2.2.3 Combining Two Parallel Algorithms

Since the computation of signed distances occurs independently and asynchronously for each traversal of octree, each traversal consists of a group of processes, a process of traversing the octree, and several processes of computing signed distance (see Figure 2.9).

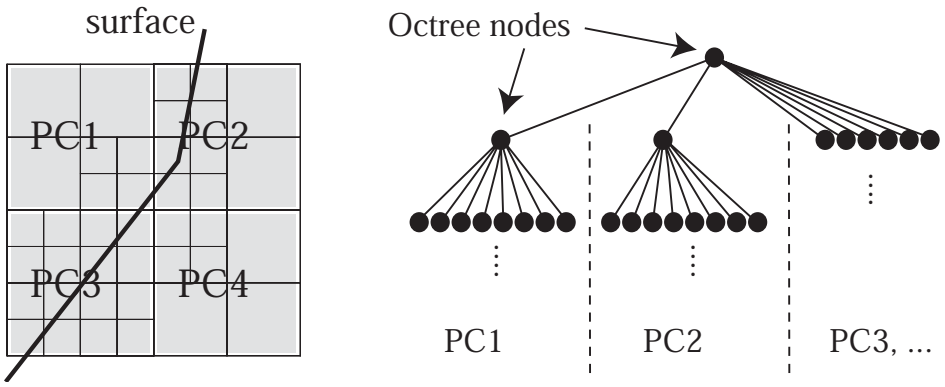


Figure 2.8: Assignment of partial space of the octree to each PC and parallel traversal of subtrees

The number of the processes of the computing signed distances depends on the data size of the range images. In our implementation, since the maximum memory size of a process is restricted by the 32bit memory of a PC, it is about 2GB. If the data size is larger than 2GB, we increased the number of traversing processes to reduce the size allocated for each process to be less than 2GB. Moreover, if the allocated size was larger than the size of the physical memory, we had to manage the working set of the data. We accomplished this management by using the memory mapping function of the operating system.

The pseudo code of the first method, the distributed allocation of range images, is shown in Algorithm 2.3. The change from Algorithm 2.1 is indicated by gray boxes. The sequential execution of **SearchNearestNeighbor** is replaced by **ParallelSearchNearestNeighbor**. Since the computational cost of **ParallelSearchNearestNeighbor** is expected to be equal to **SearchNearestNeighbor**, the computational cost of Algorithm 2.3 becomes $O(m)$, while that of Algorithm 2.1 is $O(m^2)$, where m is the number of range images. However, the computational cost of Algorithm 2.3 is actually worse than $O(m)$, because synchronization and communication is necessary after searching the nearest neighbor point in parallel.

Algorithm 2.4 shows the pseudo code of the second method, parallel subdivision and traversal of an octree. It assigns a subtree of the octree to an idling process M, until sending tasks to all processes. The depth of the distribution d_{dist} is set to be that 8^{d}dist is larger than the number of M_{all} in order to balance the load of each process; for example,

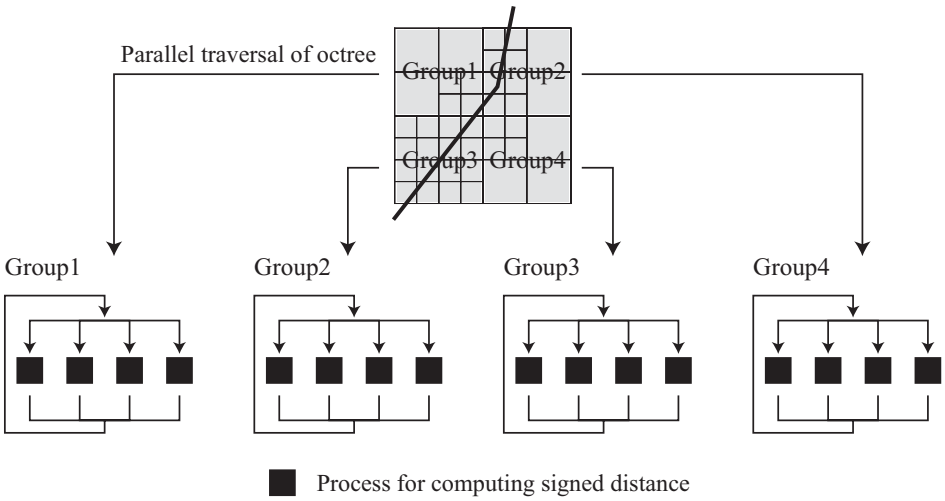


Figure 2.9: Combination of parallel computation of signed distances and parallel traversal of partial trees

we used $d_{\text{dist}} = 4$ when the number of $M_{\text{all}} = 16$ and $d_{\text{max}} = 10$. For the node of the octree whose depth is less than d_{dist}, we computed the signed distances by a single process or by calling **ParallelTraverseOctree**($M_{\text{all}}, d_{\text{dist}} - 1, 1, R_{\text{set}}$). After finishing the computing signed distances for all subtrees, the result was the creation of a whole octree.

Algorithm 2.3 ParallelConsensusSurface($\boldsymbol{x}, R_{\text{set}}$)

Input: Point: $\boldsymbol{x}$

Input: Set of Range Images: R_{set}

Local: Point: $\boldsymbol{p}, \boldsymbol{p}_1, \boldsymbol{p}_2$

Local: Normal Vector: $\boldsymbol{n}, \boldsymbol{n}_1, \boldsymbol{n}_2$

Local: Weight: $\omega, \omega_1, \omega_2$

Local: Set of $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$: $C_{\text{set}}, O_{\text{set}}, P_{\text{set}}, Q_{\text{set}}$

Output: Tuple of Point, Normal and Weight: $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$

$P_{\text{set}} \leftarrow$ **ParallelSearchNearestNeighbor**($\boldsymbol{x}, R_{\text{set}}$)
for all $\langle \boldsymbol{p}_1, \boldsymbol{n}_1, \omega_1 \rangle \in P_{\text{set}}$ **do**
$\boldsymbol{p} \leftarrow \boldsymbol{n} \leftarrow \omega \leftarrow 0$
$Q_{\text{set}} \leftarrow$ **ParallelSearchNearestNeighbor**($\boldsymbol{p}_1, R_{\text{set}}$)
for all $\langle \boldsymbol{p}_2, \boldsymbol{n}_2, \omega_2 \rangle \in Q_{\text{set}}$ **do**
if SameSurface($\langle \boldsymbol{p}_1, \boldsymbol{n}_1 \rangle, \langle \boldsymbol{p}_2, \boldsymbol{n}_2 \rangle$) **then**
$\boldsymbol{p} \leftarrow \boldsymbol{p} + \omega_2 \boldsymbol{p}_2$
$\boldsymbol{n} \leftarrow \boldsymbol{n} + \omega_2 \boldsymbol{n}_2$
$\omega \leftarrow \omega + \omega_2$
end if
end for
$\boldsymbol{p} \leftarrow \frac{1}{\omega} \boldsymbol{p}$
$\boldsymbol{n} \leftarrow \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|}$
if $\omega \geq \theta_{\text{quorum}}$ **then**
$C_{\text{set}} \leftarrow C_{\text{set}} \cup \langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$
else
$O_{\text{set}} \leftarrow O_{\text{set}} \cup \langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$
end if
end for
if $C_{\text{set}} \neq 0$ **then**
$\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle \leftarrow \arg\min_{\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle \in C_{\text{set}}} \| \boldsymbol{x} - \boldsymbol{p} \|$
else
$\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle \leftarrow \arg\max_{\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle \in O_{\text{set}}} \omega$
end if
return $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$

Algorithm 2.4 ParallelTraverseOctree($M_{\text{all}}, d_{\text{max}}, d_{\text{dist}}, R_{\text{set}}$)

Input: Set of Processes for Parallel Traversing: M_{all}

Input: Maximum Depth of Octree: d_{max}

Input: Depth of Octree for Distribution: d_{dist}

Input: Set of Range Images: R_{set}

Local: Set of Idling Processes: M_{idle}

Local: Set of Octree Nodes: N_{set}

```
M_idle ← M_all
N_set ← all nodes of octree at depth d_dist
for all N ∈ N_set do
    M ∈ M_idle
    M_idle ← M_idle ∩ M̄
    execute TraverseOctree(N, d_max, R_set) by process M in parallel
    if M_idle = 0 then
        Wait finishing one of TraverseOctree
        M_idle ← finished process M
    end if
end for
repeat
    Wait finishing one of TraverseOctree
    M_idle ← M_idle∪ finished process M
until M_idle = M_all
```

Figure 2.10: The Great Buddha of Kamakura

Figure 2.11: Range Images of the Great Buddha of Kamakura

2.2.4 Evaluation

To evaluate the performance of the proposed method we constructed a PC cluster, which consisted of eight PCs. Each PC has two 800MHz PentiumIII processors and 1GB physical memory. They were connected by a 100BASE-TX ethernet. The target object was the statue of the Great Buddha of Kamakura, which has a height of about 11.3m (see Figure 2.10). We acquired 16 range images of the Buddha by Cyrax2400 [18]. The scanning range of the sensor was about 1.5m to 50m. Each range image had about 0.3 million vertices and 0.6 million triangles (see Figure 2.11). Figure 2.12 shows the merging result of

Figure 2.12: Merging result of the Great Buddha of Kamakura

Table 2.1: Results of computation time of merging with different numbers of traversals

# of Traversals	Computational Time
1	945 min.
2	450 min.
4	227 min.
8	116 min.
16	61 min.

those range images, using the proposed parallel merging algorithm. The volume is divided to $1024 \times 1024 \times 1024 (= 2^10)$ voxel in the finest resolution, and the width of the finest voxel is about 1.4cm. The merged model consists of 3.0 million vertices and 6.0 million triangles. The mean difference between the merged model and a range image is 2.7mm. It is appropriate compared the maximum error of Cyrax2400, which is about 7–8mm.

We tested our merging algorithm by changing the number of traversals. Table 2.1 shows the result of the computational time. Since the time was almost inversely proportional to the number of traversals (see Figure 2.13), we proved that our algorithm efficiently uses the computational resources.

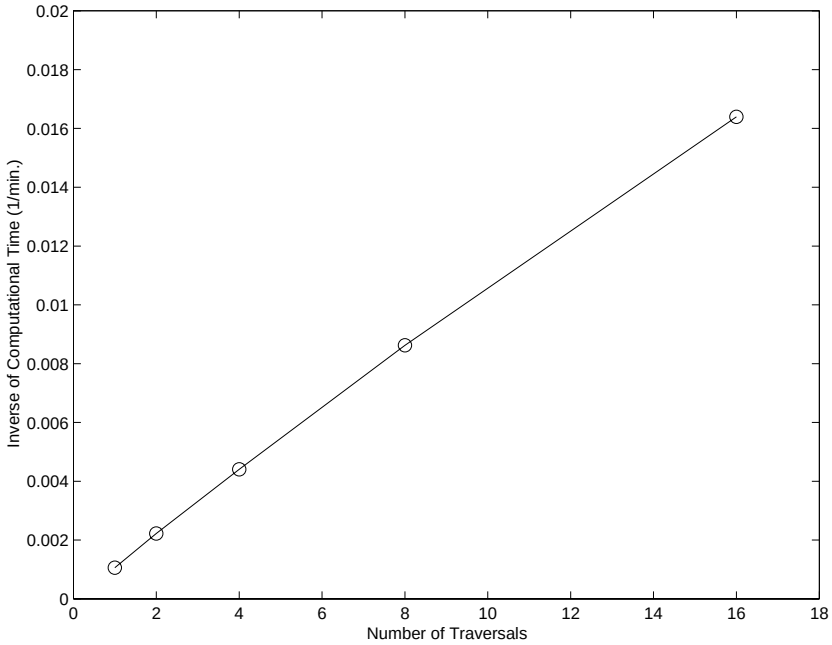


Figure 2.13: Relationship of the number of traversals and computational time

2.3 Summary

We have proposed a new method to merge a huge amount of range images by parallel computation of the signed distance using a PC cluster, which is extended from the consensus surface algorithm. It consists of the following two components:

1. Distributed allocation of range images to multiple PCs

2. Parallel traversal of subtrees of octree

We handled a huge amount of range images, which were larger than the size of the physical memory, by the first technique. The second technique makes full use of many CPUs in a PC cluster, and reduces the computational time.

Chapter 3

Effective Nearest Neighbor Search

The nearest neighbor problem in multidimensional space is a major issue in many applications. Many methods have been developed to search for the nearest neighbor of a query. A simple exhaustive search computes the distance from a query to every point. Its computational cost is $O(n)$. This approach is clearly inefficient. Hashing and indexing[99, 8] search in constant time, these methods require a large space to store the index table. Some hierarchical structures have been proposed to access multidimensional data, such as k-d tree[4, 30], quadtree[79], k-d-B tree[74], hB-tree[52] and R-tree[35]. These trees differ in structure, but their searching algorithms are similar. For details, refer to [32].

The k-d tree[4, 30] is one of the most widely used structures for searching for nearest neighbors. It is a kind of binary tree that partitions space using hyperplanes that are perpendicular to the coordinate axes. If a k-d tree consists of n records, the k-d tree requires $O(n \log_2 n)$ operations to construct and $O(\log_2 n)$ to search. In this chapter, we analyze the search algorithm using the k-d tree and propose a novel method of pruning branches to reduce the computational cost.

A case in which a search finishes in $O(\log_2 n)$ is an ideal case, in that only a leaf node is examined, and the nearest neighbor belongs to it. However, a search using a k-d tree does not actually finish in logarithmic time. In many cases, a search needs to examine several leaf nodes before finding the nearest neighbor point. In the worst case scenario, all leaf nodes must be examined. When the nearest neighbor is far from a query, the number of leaf nodes is apt to increase (see 3.1). Therefore, we introduced a novel test that takes place during the traversing of the k-d tree. This test compares the distance from a query to

the nearest neighbor with a threshold defined by the user (See 3.2).
Since the threshold depends on the application, we tested our new method in the following three applications: aligning range images, merging range images, and retrieving similar color images. The first two applications were three-dimensional cases, in which 3-D models were compared. For aligning range images, several methods have been proposed to speed up the search of a k-d tree by caching the closest points [88, 34]. The third application is a high-dimensional case, and we analyzed the effectiveness of our method for high-dimensional applications. Nene and Nayar[62] proposed a method to find the nearest neighbor point within a distance ϵ in high-dimensional space. However, this method cannot find any point outside the distance ϵ, and it has to re-create the data structure if ϵ changes.

3.1 Basic Search Algorithm using k-d Tree

First, we explain the basic algorithm by which the k-d tree searches for the nearest neighbor. Figure 3.1 shows a 2-D example of a k-d tree that consists of four leaf nodes labeled A, B, C and D. We do not describe how to construct a k-d tree in this paper; please refer to [4, 30] for more details.

3.1.1 Finding the Nearest Neighbor Point

Now we will find the nearest neighbor point from a query point $\boldsymbol{p}$. In the searching algorithm, we will start at the root node and traverse down to the leaf node that contains the query point. In Figure 3.1, the leaf node A contains $\boldsymbol{p}$, and we compute the distances from $\boldsymbol{p}$ to the records of A.
To avoid examining all leaf nodes, the algorithm prune branches by the Bounds-Overlap-Ball (BOB) test[30]. After node A is examined, the distance from $\boldsymbol{p}$ to the nearest neighbor is d. We examine B if d satisfies the following BOB test:

$$d > d_B, \tag{3.1}$$

where d_B is the distance from the query point $\boldsymbol{p}$ to the boundary of A and B. Similarly, we compare d with d_C and d_D to decide whether or not we will examine C and D. In this case, d satisfies (3.1) for B, C and D. Thus, we have to examine all nodes.

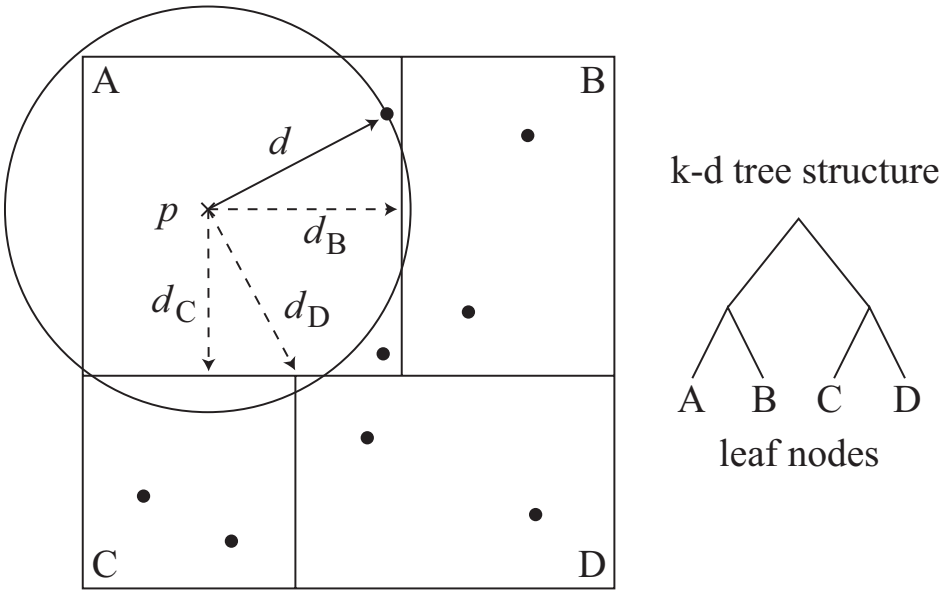


Figure 3.1: A 2-D example of a k-d tree

The basic search algorithm is represented by a recursive function (Algorithm 3.1). N is the node interested in. $\boldsymbol{p}$ is the query point. d is the distance of the current nearest neighbor. rightson(N) and leftson(N) mean the sons of node N. $d_{\text{rightson}(N)}$ and $d_{\text{leftson}(N)}$ are the distance from the query to the boundary of the right/left son of N.

3.1.2 Estimating Computational Cost

If a k-d tree contains n records, the depth of the tree is $O(\log_2 n)$. Because of this, Friedman et al.[30] says that the computational cost of searching for the nearest neighbor using the k-d tree is $O(\log_2 n)$. However, this is only for a case in which the leaf node that contains the query is examined, and all other branches are pruned by the BOB test.
Actually, in the worst case, such as Figure 3.1, no branch can be pruned, and the computational cost becomes $O(n)$[1]. In particular, when the distance d from the query to the nearest neighbor is large in comparison with the distribution of records in the k-d tree, almost all boundaries can be inside the ball.

[1]This is a rough estimation. A more accurate estimation is $O(\sum_{k=0}^{\log_2 n} 2^k)$

Algorithm 3.1 SearchNearestNeighbor(N)

Input: Node N

```
if N is leaf node then
  Examine records of N
else
  if p is inside leftson(N) then
    Search(leftson(N))
    if d > d_rightson(N) then
      Search(rightson(N))
    end if
  else
    Search(rightson(N))
    if d > d_leftson(N) then
      Search(leftson(N))
    end if
  end if
end if
```

3.2 Bounds-Overlap-Threshold Test

In this section, we propose a new method to reduce the computational cost of searching for the nearest neighbor points using the k-d tree. Many applications need nearest neighbor points that are actually near queries. Thus, when the records are far from the queries, they are either not used or contribute minimally. However, as we estimated the computational cost in the previous section, if all records in a k-d tree are far from a query, the computational cost is larger than it is when the nearest neighbor is close.

If we can assume that it is not important if the nearest neighbor is far from a query, it is sufficient that we find out that there are no records near the query. Then, we propose a new method to reduce the computational cost when the nearest neighbor is far from a query. We introduce the Bounds-Overlap-Threshold (BOT) test to the searching algorithm.

Now, we assume that it is not important if the nearest points are farther than δ. Figure 3.2 shows the same situation with Figure 3.1. When we decide whether to examine node B or

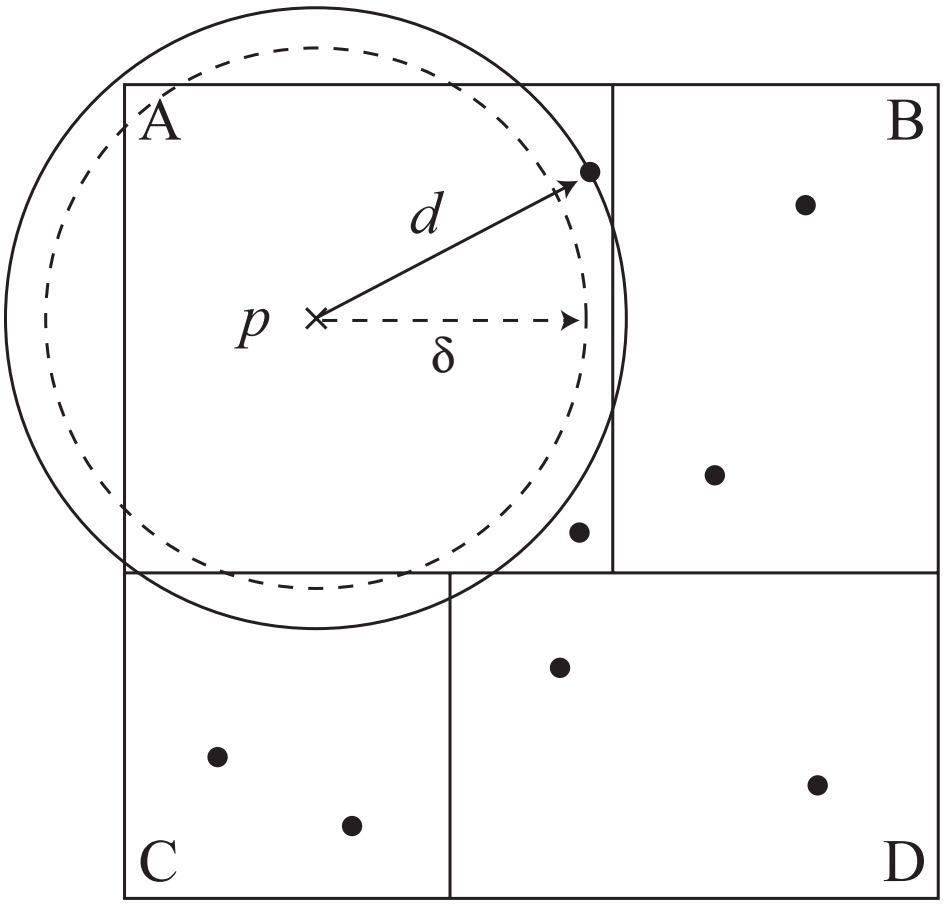


Figure 3.2: The Bounds-Overlap-Threshold (BOT) test

not, we define the BOT test as follows:

$$\delta > d_B. \tag{3.2}$$

We prune a branch if the BOB test or the BOT test fails. Since the BOT test failed in the cases of B and D, we did not examine them. However, we examined C, since $d > d_C$ and $\delta > d_C$.

If $d \leq \delta$, the algorithm is completely the same as that before the BOT test is introduced. If $d > \delta$, the new algorithm may not find the true nearest neighbor point. The distance d_{true} from the query to the true nearest neighbor is

$$\delta < d_N < d_{true} \leq d, \tag{3.3}$$

Algorithm 3.2 SearchNearestNeighborBOT(N)

Input: Node N

```
if N is leaf node then
    Examine records of N
else
    if p is inside leftson(N) then
        Search(leftson(N))
        if d > d_rightson(N)  ∧  δ > d_rightson(N) then
            Search(rightson(N))
        end if
    else
        Search(rightson(N))
        if d > d_leftson(N)  ∧  δ > d_leftson(N) then
            Search(leftson(N))
        end if
    end if
end if
```

where d_N is the smallest distance from the query to the boundary of node N, which is larger than δ.

If the maximum distance of two points in a k-d tree is $2D$, the volume of the hypersphere, which contains all points of a k-d tree, is proportional to D^k. The volume of the hypersphere of radius δ is also proportional to δ^k. Therefore, if the points in a k-d tree have uniform distribution, our new method reduces the computational cost of the worst case from $O(n)$ to $O((\frac{\delta}{D})^k n)$.

The search algorithm with the BOT test is represented by a recursive function (Algorithm 3.2). N is the node interested in. $\boldsymbol{p}$ is the query point. d is the distance of the current nearest neighbor. rightson(N) and leftson(N) mean the sons of node N. $d_{\text{rightson}(N)}$ and $d_{\text{leftson}(N)}$ are the distance from the query to the boundary of the right/left son of N. The difference from the basic algorithm is illustrated by gray boxes.

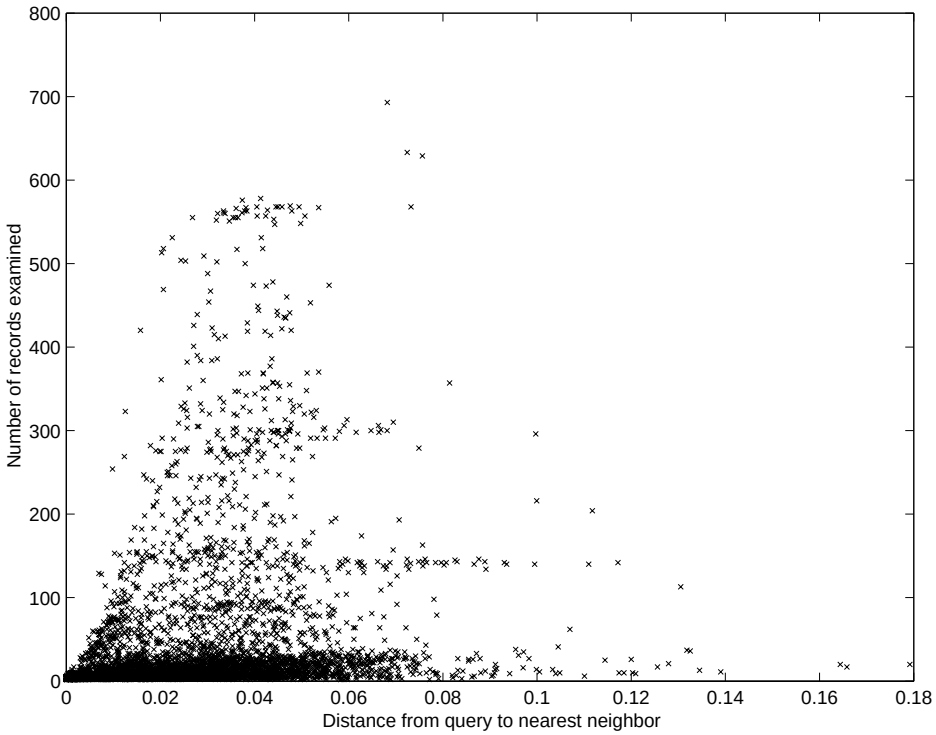


Figure 3.3: Relationship between distance from a query to the nearest neighbor and the number of records examined using the basic search algorithm for merging range images.

3.3 Applying to Consensus Surface

As described in Chapter 2, to merge range images, we convert the images to volumetric representation by computing the signed distance field (SDF) from the multiple range images. We create a mesh model of the whole object by converting from the SDF by using the marching cubes algorithm[53]. If the distance from a voxel to the range images is larger than $\frac{\sqrt{3}}{2}w$, where w is the interval of voxels, there is no surface around the voxel. Thus, it is enough for us to find that no point in the k-d tree is closer than $\frac{\sqrt{3}}{2}w$, and we set $\delta = \frac{\sqrt{3}}{2}w$.

Our merging method, which is based on Wheeler's method[97], reduces the computation of the SDF in an octree manner. Therefore, the voxel width w varies according to the depth of octree subdivision to which the current voxel belongs. We change the threshold δ as well as the voxel width w.

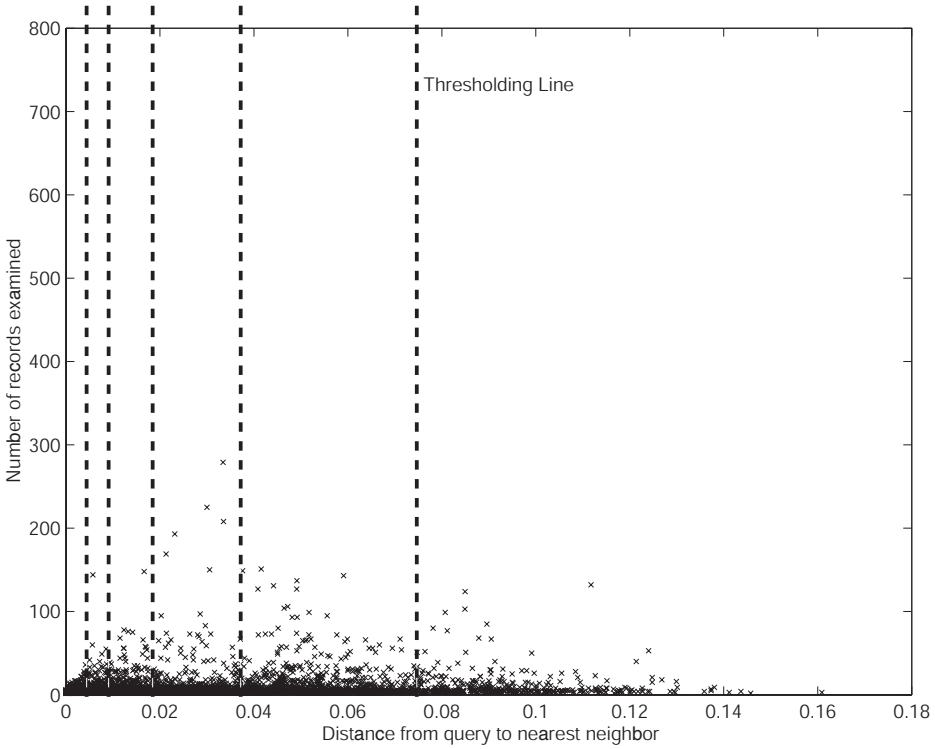


Figure 3.4: Relationship between distance from a query to the nearest neighbor and the number of records examined with the BOT test for merging range images.

Figure 3.3 and Figure 3.4 shows an example of the distribution of the number of records examined during the search for a nearest neighbor point in the merging application. When we search for the nearest neighbor points using the BOT test, the number of records examined gets closer to 1 at any distance from the query. This is because we adjust threshold δ according to the voxel width. In this example, the total numbers of records examined are 11,844,253 without the BOT test and 2,716,475 with the BOT test. Specifically, the computational cost of searching the nearest neighbor points is reduced to 22.9% of that of the basic search algorithm.

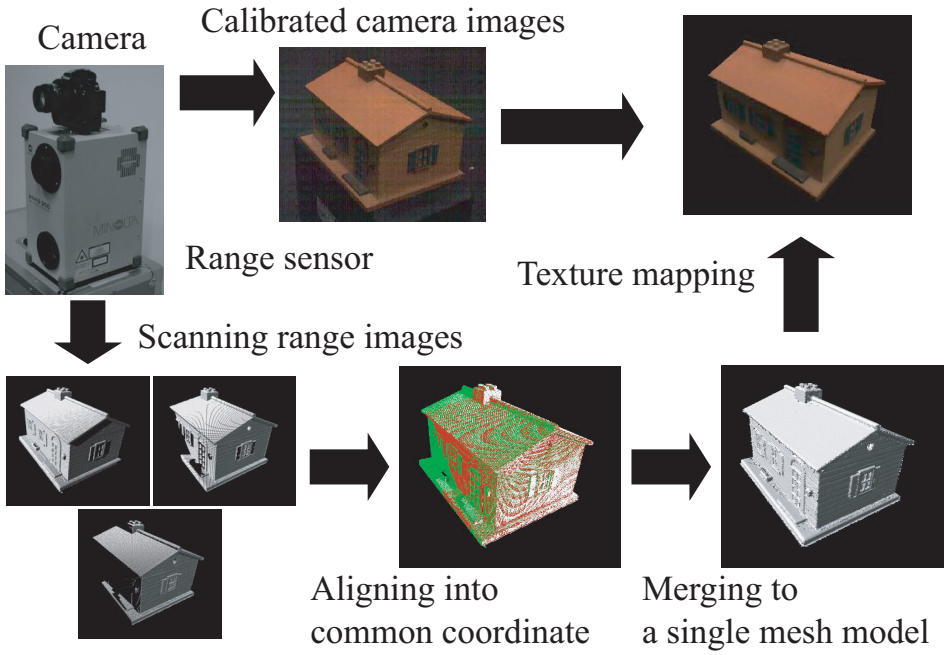


Figure 3.5: Steps of geometric and photometric modeling of an object

3.4 Other Applications

To test our new method, we considered two other applications that use nearest neighbors in a multidimensional space: aligning range images and retrieving similar images. The former application is for 3-D cases, and the latter is for a higher-dimensional case.

3.4.1 Aligning Range Images

Figure 3.5 shows the steps in modeling an object's shape (same with Figure 1.2). We can acquire the shape of an object from various viewpoints using range finders[18, 58]. A range image acquired by the range finders contains part of the shape of an object. Thus, we capture range images from various viewpoints to acquire the whole shape of the object. We do not know the mutual relationship of the range images a priori. Therefore, we align them into a common coordinate system using iterative registration techniques [6, 97, 72, 55]. To find corresponding points between two range images, we use the nearest neighbor points by searching a k-d tree that we construct from each range image.

In aligning range images, some methods remove wrong correspondences by thresholding or M-estimation [72, 97, 55]. Thus, in computing the posture of range images, we can assume that the presence of corresponding points farther than the threshold distance is not important. In this experiment, by considering the distribution of the error of the laser range finder, we set the threshold $\delta = 1.0$cm. Figure 3.6 shows the distribution of the number of records examined during the search for a nearest neighbor point in the aligning application.

When we use the basic search algorithm, the number of records examined grows according to the distance from the queries. On the other hand, when we search for nearest neighbor points using the BOT test, we can drastically reduce the number of records examined in the area where the distance from the query is larger than d. (See Figure 3.7). The total number of records examined are 1,842,640 without the BOT test, and 470,300 with the BOT test. The computational cost of searching for the nearest neighbor points is regarded as 25.5% of that of the basic search algorithm.

3.4.2 Retrieving Similar Images

The third application is image retrieval from the image database by comparing pixels of color images. The difference between the first two applications and this one is the number of dimensions. Since a vector stored in a k-d tree is a vector of RGB values converted from a color image by arranging them in a raster scan manner, the dimension is very high. The test image set we used is the Columbia-Utrecht reflectance and texture database[19]. We used 100×100 pixels of the center of the database images, which are originally 640×480 pixels. Thus, the dimensionality is 3(RGB) $\times 100 \times 100 = 30,000$. For the criterion for comparing vectors, we simply used

$$\begin{aligned} C_{12} &= \sum_{i,j} D^2(i,j) \\ D^2(i,j) &= \sum^{R,G,B} (I_1(i,j) - I_2(i,j))^2, \end{aligned} \tag{3.4}$$

where $I(i,j)$ is the intensity of each RGB element. Each RGB element is normalized to 0.0-1.0 in the k-d tree.

We constructed a k-d tree that contained 2501 images, which is 20% of the original image database. The number of reference images for the sake of comparison is 2135; these

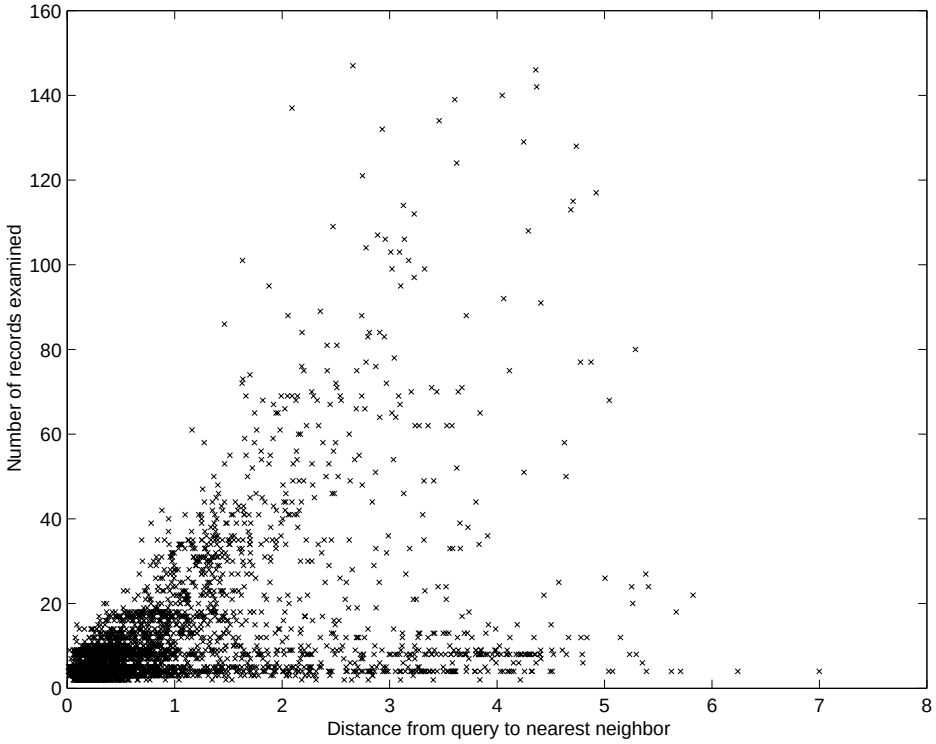


Figure 3.6: An example of the relationship between the distance from a query to the nearest neighbor and the number of records examined, using the basic search algorithm for aligning range images.

images were randomly sampled from the original database and blurred by a Gaussian filter. That is, the original image of a reference image is not always contained in the k-d tree.

Figure 3.9 shows the distribution of the number of records examined without the BOT test. Because the dimensionality is very high, the BOB test does not work effectively. Consequently, many image searches will examine all records of the k-d tree. We estimated that a typical distance between a blurred image and its original image is less than 1000. Thus, we set the threshold δ to 1000. Figure 3.10 shows the result with the BOT test of $delta = 1000$. We see that the BOT test is less effective than low-dimensional cases, and is also less effective than the BOB test. Because the distance from a query to the nearest neighbor is summed up over all dimensions, we cannot set δ to a smaller value, even when

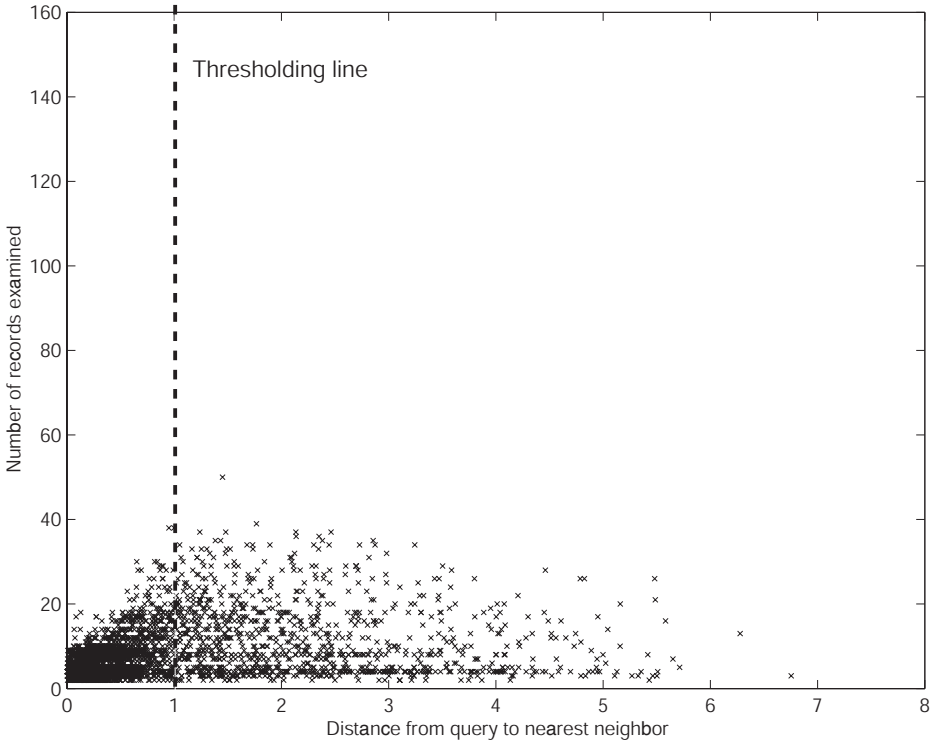


Figure 3.7: Relationship between distance from a query to the nearest neighbor and the number of records examined, using the BOT test for aligning range images.

each distance in a dimension is quite small.

Although the effectiveness of the BOT test is reduced in high-dimensional applications, this test improves the efficiency of searching for nearest neighbors. The total numbers of records examined are 4,666,740 without the BOT test and 1,746,480 with it. The computational cost of searching the nearest neighbor points is reduced to 37.4% of that of the basic search algorithm.

3.5 Discussion

The performance of the BOT test depends on the application. In this section, we consider the condition of the application in which our method works best. First, the most important requirement in applying our method is that the threshold δ can be determined in the appli-

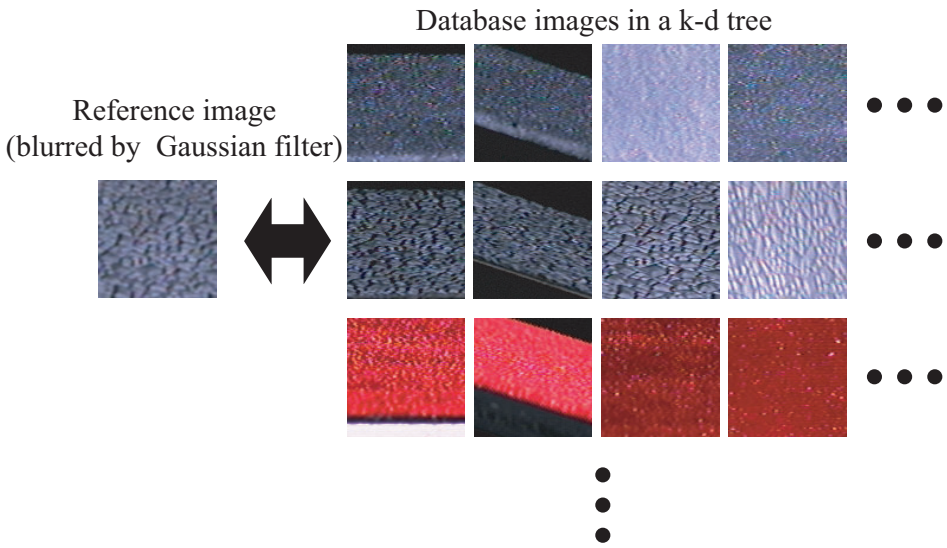


Figure 3.8: Image retrieval: Images are stored in a k-d tree. The k-d tree contains 20 % of the original image database. A reference image is blurred by a Gaussian filter from one of the images in the original database.

cation. After determining δ, the performance of the BOT test depends on the distribution of distances from queries to nearest neighbor points. Our method works best when the portion of the number of nearest neighbor points that are farther than δ becomes larger.
As shown in merging range images, our method can be applied with the variable threshold δ without re-creating the structure of a k-d tree. Therefore, our method is efficient in cases in which the appropriate threshold varies according to the situation.
In a high-dimensional case such as image retrieval, the BOT test is less effective than it is in a low-dimensional case. However, since the performance improves drastically, as we see in 3.4.2, our method can be applied to high-dimensional applications.
From the viewpoint of the correctness of the nearest neighbor point, if the distance is smaller than δ, the found result is correct. If the distance is larger than δ, the distance to the correct nearest neighbor is in the range given by (3.3). Thus, (3.3) gives us the estimation for the distance of the true nearest neighbor. However, we cannot obtain a good estimation for the vector of the true nearest neighbor. Therefore, we have to set δ larger than the minimum distance of the vector of the nearest neighbor that we need to obtain.

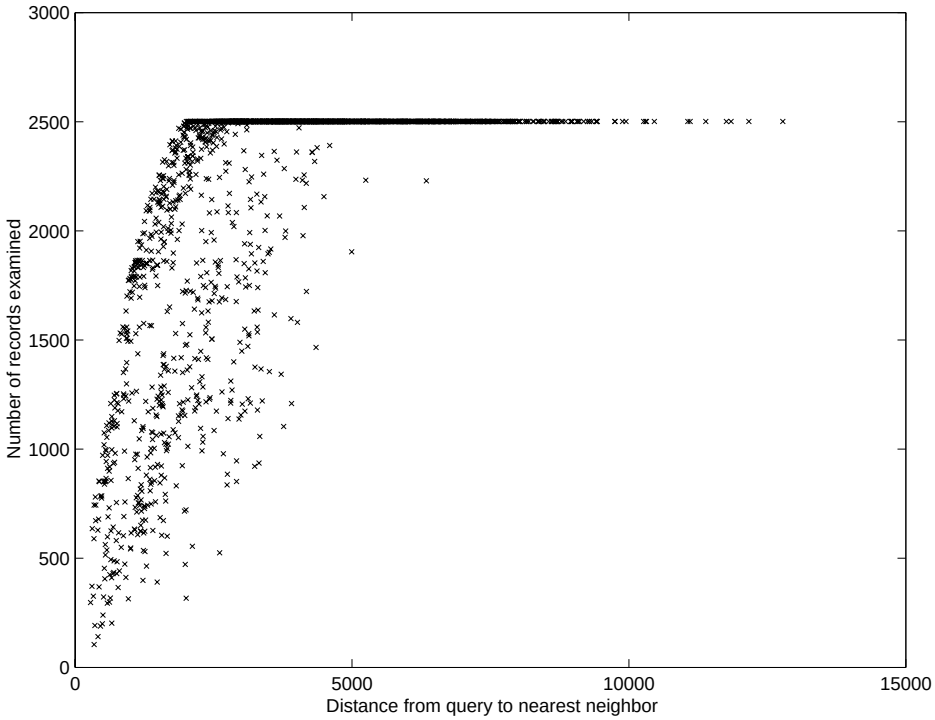


Figure 3.9: The relationship between the distance from a query to the nearest neighbor and the number of records examined when the basic search algorithm is used to retrieve similar color images.

3.6 Summary

In this chapter, we proposed a new algorithm for searching for the nearest neighbor using the k-d tree. If the nearest neighbor point is far from a query, it is not an important nearest neighbor in many applications. Thus, we propose the Bounds-Overlap-Threshold test, which does not search strictly by pruning branches if the nearest neighbor point is beyond a threshold. This technique drastically reduces the computational cost if the nearest neighbor is far from a query. Since the threshold of the BOT test can be changed without re-creating a k-d tree, it is suitable for applications in which variable threshold is considered. Finally, we discuss the performance, which depends on the distribution of the distance from a query to the nearest neighbor.

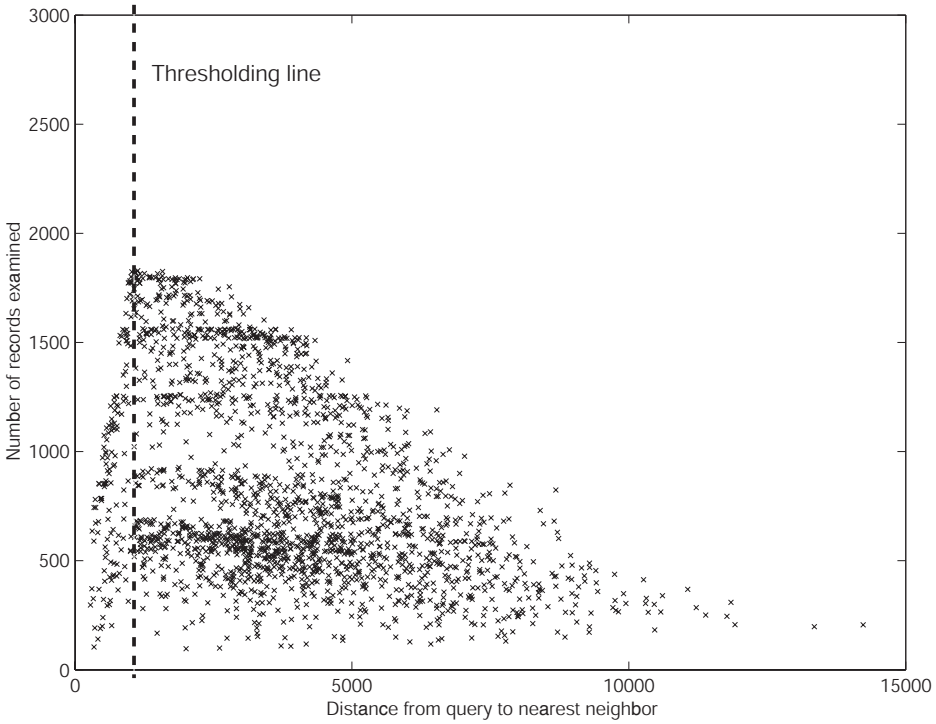


Figure 3.10: Relationship between the distance from a query to the nearest neighbor and the number of records examined when the BOT test is used to retrieve similar color images.

Chapter 4

Adaptive Merging Algorithm

The third approach to merging a huge amount of range images is an adaptive algorithm of merging range images according to the shape of the object. The original algorithm produces a mesh model of the finest resolution everywhere; however, the dense sampling for generating a mesh model is not necessary where the shape of the object is near planar. Thus, we propose two algorithms to construct the 3D model in an efficient resolution.
We consider two methods to compute the curvature of surface. The first one is the curvature of an implicit surface. Since we compute the signed distance field (SDF) using range images in our merging process, the curvature of the surface can be computed by SDF. If the curvature around a voxel is near planar, we use the larger voxel of the octree to convert to a mesh model. Thus, we can reduce the data size of the created mesh model.
The second method computes the curvature of a surface from the curvature of range images. Since the first method reduces the data size after computing SDF in the finest resolution everywhere of the volume of interest, it cannot reduce the computational cost of computing the SDF. However, the second one reduces both the computational cost and the data size of the mesh model, since it computes SDF only in high curvature areas.

4.1 Shrinking Octree Based on Curvature of Implicit Surface

To reduce the data size of a mesh model which is generated by the marching cubes algorithm (MC), we shrink the octree created in the merging process by removing voxels which are subdivided in an octree manner. If the curvature of SDF in a voxel is small and

Algorithm 4.1 ImplicitCurvature(N)

Input: Current Node of Octree: N

Output: Curvature of SDF around N: κ

```
compute κ by (4.1)
if N is terminal then
    return |κ|
else
    for all children N_i (i = 0, ..., 7) of N do
        κ_i = ImplicitCurvature(N_i)
    end for
    return max(|κ|, |κ_i|)(i = 0, ..., 7)
end if
```

the surface is near planar, we do not use the voxel to generate a mesh model by MC. The mean curvature κ of SDF is computed by the following equation [85]:

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} = \frac{\left\{ \begin{array}{l} (\phi_{yy}+\phi_{zz})\phi_x^2 + (\phi_{xx}+\phi_{zz})\phi_y^2 + (\phi_{xx}+\phi_{yy})\phi_z^2 \\ -2\phi_x\phi_y\phi_{xy} - 2\phi_y\phi_z\phi_{yz} - 2\phi_z\phi_x\phi_{zx} \end{array} \right\}}{(\phi_x^2+\phi_y^2+\phi_z^2)^{3/2}}, \qquad (4.1)$$

where ϕ is the SDF of the volume. ϕ_x means $\frac{\partial\phi}{\partial x}$ and other subscripted notations similarly mean partial derivatives. In the numerical implementation, the derivatives are calculated by centralized differences.

The curvature of SDF is computed by Algorithm 4.1. We shrink an octree by removing voxels whose curvature κ are less than a threshold κ_θ. Therefore, if the surface is near planar, the voxels of finer resolution are removed from the octree, and the resolution of the new octree becomes adaptive according to the curvature of the surface. Algorithm 4.2 shows the algorithm for removing voxels from the octree based on the curvature.

This approach need to generate a fine SDF and shrink it as a post process. Thus, the computational cost becomes larger. Moreover, since this process is a simpilification of a model, it can be replaced by the techniques of mesh simplification [33, 41, 42] after generating a mesh model. Then, in the next section, we describe a method to reduce the computional cost of generating a SDF.

Algorithm 4.2 ShrinkOctree(N)

Input: Current Node of Octree: N

```
κ = ImplicitCurvature(N)
if κ < κ_θ then
    remove N and its descendants from the octree
    return
end if
if N is nonterminal then
    for all children N_i(i = 0, ..., 7) of N do
        ShrinkOctree(N_i)
    end for
end if
```

4.2 Subdividing Voxels Based on the Geometric Attributes of Range Images

As the second method, we determined the sampling interval of the signed distance depending on the variation of geometric attributes to efficiently represent the final mesh model. As an example, we used the surface curvature. Depending on the change in surface curvature, the proposed method coarsely samples in planar areas, consequently reducing the amount of data and computation, while creating a finer model of an intricate object by efficiently utilizing computation power.

Our method determines the variation of surface curvature comparing surface normals of range images. We compare the normal $\boldsymbol{n}_i$ of each 3D point of all range images inside the voxel in interest and the normal $\bar{\boldsymbol{n}}$ of the approximated plane (see Figure 4.1), which can be estimated by applying principal component analysis (PCA) to all point data in the voxel. If the angle between the data point normals $\boldsymbol{n}_i$ and approximate normal $\bar{\boldsymbol{n}}$ satisfies

$$\max_i(\arccos(\boldsymbol{n}_i \cdot \bar{\boldsymbol{n}})) < \delta_n, \tag{4.2}$$

where δ_n is the threshold of the angle, the sampling interval is fine enough, and no further voxel splitting is required.

To avoid erroneous subdivisions of voxels by the influence of noise included in each range image, our method takes a consensus between range images on the decision of voxel subdivision. Now, N_n is the number of range images which satisfies (4.2). Our method does

not subdivide the voxel if

$$N_n > T_n, \tag{4.3}$$

where T_n is the threshold of consensus for normal vectors.

Another approach to avoid erroneous subdivisions of voxels is taking the consensus of range images before the merging process. The algorithm for taking consensus described in Algorithm 2.1 requires an arbitrary point $\boldsymbol{x}$ and a set of range images R_{set} as input. Thus, we can compute the consensus of each vertex of range images. By computing the consensus of vertices a priori, we only use vertices which have sufficient consensus for computing (4.2). We propose a novel method to computing the consensus of range images in Chapter 6, which is applied before the merging process; therefore, the method can be used to avoid the erroneous subdivisions of voxels.

The algorithm of traversing an octree with adaptive voxel subdivision is represented as Algorithm 4.3. The changes from Algorithm 2.2 are indicated by gray boxes. To determine if we subdivide the current voxel N or not, we consider the curvature of range images inside the voxel by **LocalCurvature**(N, R_{set}) (see Algorithm 4.4). **LocalCurvature** returns the number of range images which satisfies (4.2). Moreover, since we subdivide the voxels adaptively, the voxels attain sufficient resolution even if the threshold value of the magnitude of a signed distance is reduced to $\frac{\sqrt{3}}{2}w$.

4.3 Marching Cubes for Adaptive Octree

The original marching cubes algorithm (MC) can be applied only to voxels that have the same resolution (size of voxels). We extend the algorithm for the triangulation of voxels in adaptive resolution generated from our method.

For voxels that are surrounded by voxels with the same resolution, the vertices of a cube to march are the central points of 8 adjacent voxels. In a similar manner, voxels surrounded by different size voxels will have a set of connected voxels in a form of quadratic pyramid or other special forms to march. Figure 4.2 shows the edges connecting adjacent voxels in an adaptive octree. The 3D space is partitioned into cubes or quadratic pyramids, etc. Since these forms can be considered to be degenerated and transformed cubes, the original marching cube algorithm can be applied to these irregular forms without creating new tables of mesh generation for each form. When we use voxels of fixed resolution (grids

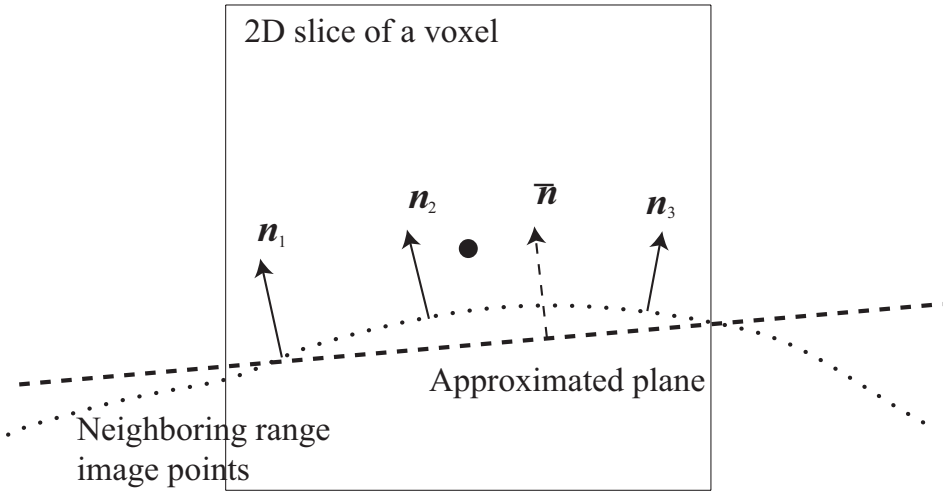


Figure 4.1: Comparison of the normal vector of each vertex and the approximate normal $\bar{n}$ by PCA

of gray lines), a mesh model of the dotted line is generated, and its vertices are on the edge of cubes. When we use adaptively subdivided voxels (grids of black lines), the mesh model of a solid line is generated, and its vertices are on the edge of transformed cubes. If we subdivide the high curvature area into small voxels, we can effectively create a precise mesh model. Since a transformed cube becomes a skewed rectangle or a triangle in a 2D slice of the volume, as shown in Figure 4.2, the vertices of the mesh model generated by MC are on those edges. The detailed algorithm used to determine the voxels to be used to MC is described in Appendix A.

Figure 4.3 shows examples of a transformed cube in a 3D isometric view. Figure 4.3(b) is a pyramid, since the upper four vertices are not subdivided. We can regard that (a) becomes (b) by moving the upper four vertices of (a) to the same position, which have the same signed distance. Thus, we can generate the isosurface of (b) by applying MC to (a), which has the same signed distance with (b). MC uses a table to generate an isosurface, which is classified by the state of signed distances of its eight vertices. Since we regard a voxel of the adaptive octree as a transformed cube, we can apply MC without creating a new table. In the case of Figure 4.3(b) and (c), two triangles are generated. However, the number of triangles are reduced in the case of Figure 4.3(d), because the number of edges

Algorithm 4.3 AdaptiveTraverseOctree$(N, d_{\max}, R_{\text{set}})$

Input: Current Node of Octree: N

Input: Maximum Depth of Octree: $d_{\max}$

Input: Set of Range Images: R_{set}

Local: Center of N: $\boldsymbol{x}$

Local: Octree Depth of N: d

Local: Width of N: w

Local: Tuple of Point, Normal and Weight: $\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle$

Output: Signed Distance of N: v

$\langle \boldsymbol{p}, \boldsymbol{n}, \omega \rangle \leftarrow \mathbf{ConsensusSurface}(\boldsymbol{x}, R_{\text{set}})$

if $(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n} > 0$ **then**

$v \leftarrow \| \boldsymbol{x} - \boldsymbol{p} \|$

else

$v \leftarrow - \| \boldsymbol{x} - \boldsymbol{p} \|$

end if

if $|v| < \frac{\sqrt{3}}{2} w \wedge d < d_{\max} \wedge$ **LocalCurvature**$(N, R_{\text{set}}) > T_n$ **then**

for all children $N_i (i = 0, \ldots, 7)$ of N **do**

AdaptiveTraverseOctree$(N_i, d_{\max}, R_{\text{set}})$

end for

end if

of the transformed cube are reduced. Therefore, we removed the redundant vertices of the mesh model after generating it by applying MC.

4.4 Experiment of Adaptive Merging

We experimented with the adaptive merging algorithm using range images of the Great Buddha of Kamakura. The specification of the range images was the same with 2.2.4; we acquired 16 range images by Cyrax2400 [18], which contained about 0.3 million vertices and 0.6 million triangles in each range image. Figure 4.4 shows one of the range images (column A), the merging result without adaptive subdivision (column B), and the merging result with adaptive subdivision (column C). The statistics of the merging process are

Algorithm 4.4 LocalCurvature(N, R_{set})

Input: Current Node of Octree: N

Input: Set of Range Images: R_{set}

Local: Set of Range Images: R_{curved}

```
Compute the approximate normal vector n̄
R_curved ← 0
for all R ∈ R_set do
  for all normal vector n_i of vertices inside N do
    if arccos(n_i · n̄) < δ_n then
      R_curved ← R_curved ∩ R
    end if
  end for
end for
return Number of elements of R_curved
```

Table 4.1: Statistics of models of the Great Buddha of Kamakura

	number of points	Time for Merging	Mean Difference
(B)	3.0 million	61 min.	N/A
(C)	1.4 million	25 min.	0.99 mm

described in Table 4.1. The adaptive merging algorithm reduces the amount of data and computation time required using the original merging method. We compared the difference between (B) and (C) using Metro [13]. The mean difference (0.99mm) was quite small compared to the height of the Buddha (11.3m tall).

The figures in the top row are rendered using triangle faces. The result of the adaptive merging seems (C) completely the same as the result of the fixed resolution (B). However, if they are rendered by a wireframe, as shown in the middle row, we can see that our adaptive merging algorithm generates larger faces in planar areas. Thus, the size of the result of the adaptive merging is reduced to less than 50% of the result of the fixed resolution. Consequently, the time for merging is also reduced to less than 50%. The figures in the bottom row are zoom-ups of the forehead of the Buddha.

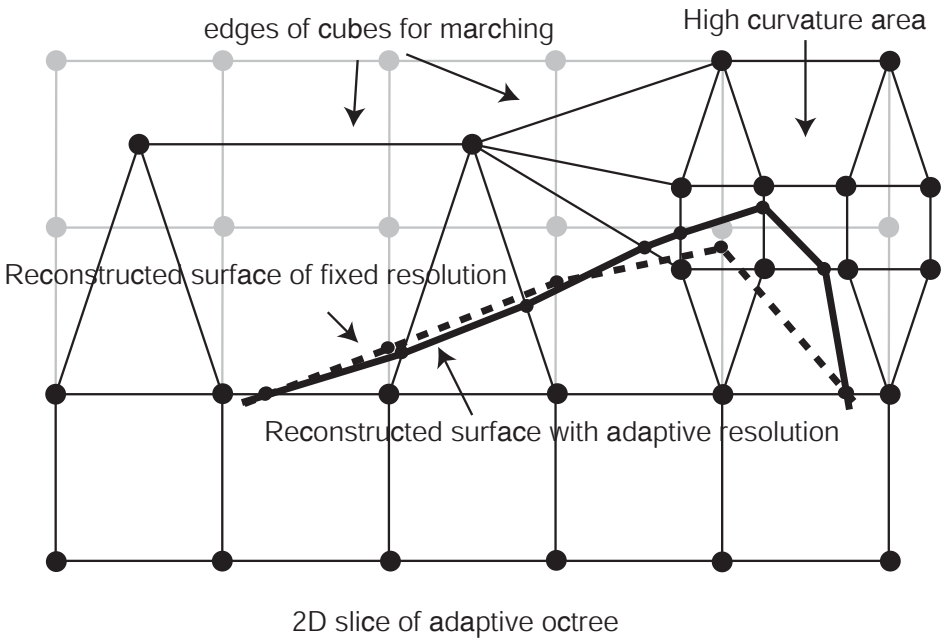


Figure 4.2: Edges connecting adjacent voxels in an adaptive octree and the generate mesh model by MC

4.5 Summary

We have proposed an algorithm for constructing a 3D model in an efficient representation. Considering the surface curvature, we constructed 3D models that have higher detail in surface areas that contain high curvature. We have proposed two approaches, one which considers the curvature of the implicit surface of SDF after the merging process, and the other, which is an adaptive merging technique based on the curvature of the range images. We can efficiently use the computational resources by these methods.

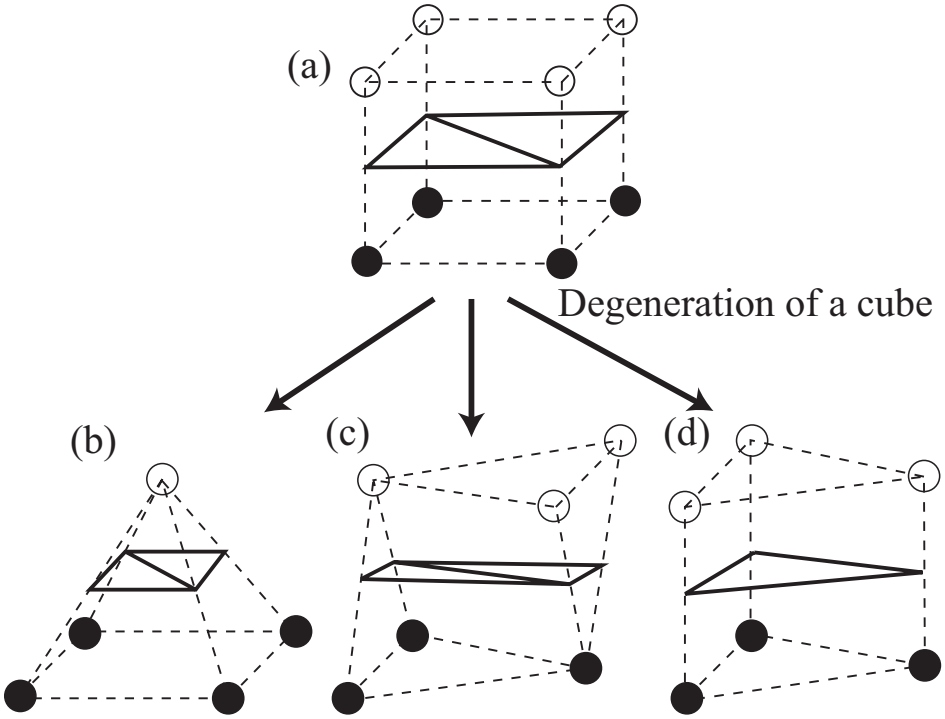


Figure 4.3: Examples of degenerated cubes and the surfaces generated by MC

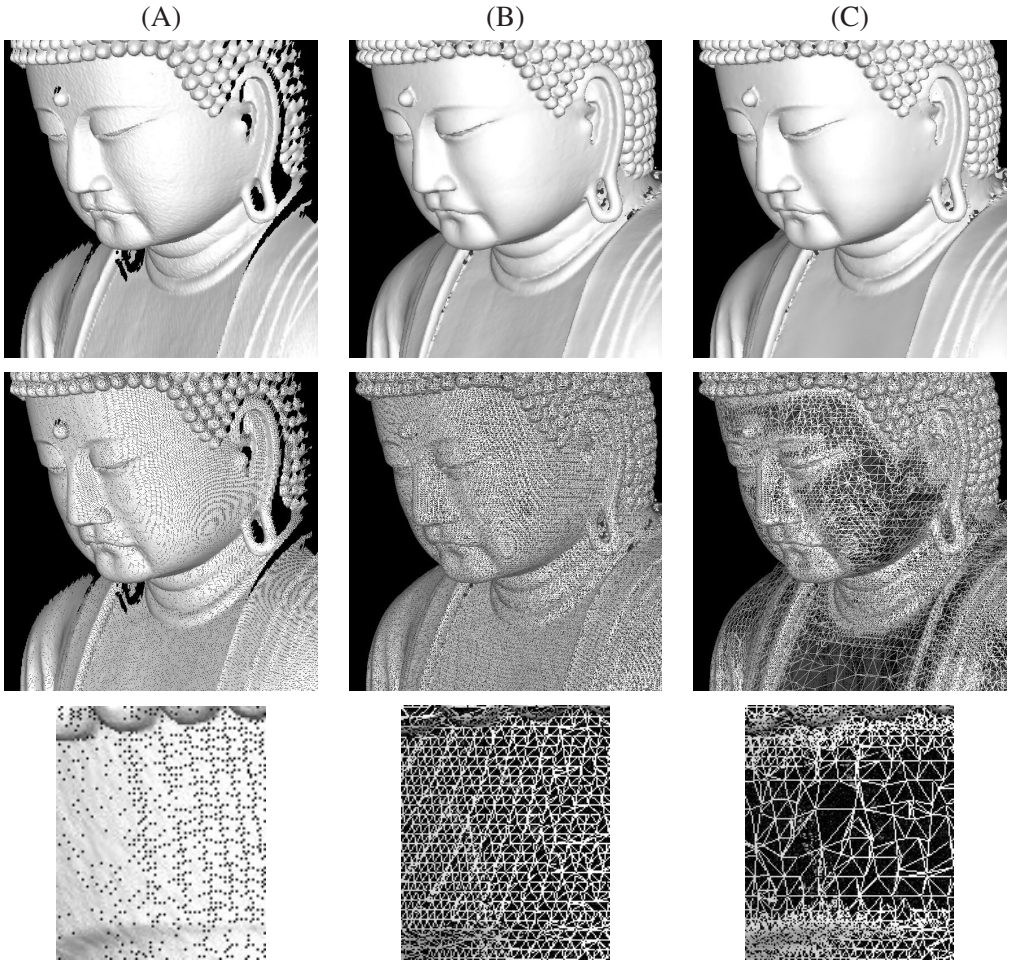


Figure 4.4: Adaptive Merging Results of the Great Buddha of Kamakura

Chapter 5

Merging of Photometric Attributes of Range Images

In this chapter, we propose a novel method of merging photometric attributes attached to range images. In applications utilizing geometric models, e.g., 3D object recognition/localization, non-rigid appearance variation plays a crucial role in the accuracy and robustness. In particular,, specular reflection causes the appearance to change non-rigidly, consequently making the whole process difficult. Thus, to date, most object recognition and object localization algorithms simply neglect specular reflection as outliers, and assume Lambertian surfaces for the target scene and model. To cooperate with this basic assumption on photometric properties, it is highly desirable to construct the geometric model to be used in such applications with Lambertian reflection properties. If the 3D model is represented with photometric attributes that are rigid against changes in illumination and viewing directions, a higher accuracy and robustness can be expected. Furthermore, if the illumination directions and viewing directions can be pre-estimated when processing recognition algorithms, a non-rigid appearance variation such as specular reflection can be predicted and added to the 3D model appearance to further elevate the accuracy.
We accomplish this photometrically rigid 3D model construction in our range image integration framework. As examples of photometric attributes attached to range images, we consider two different attributes: laser reflectance strength and intensity/color.

Figure 5.1: Two images of the Great Buddha in Kamakura, using LRS values as pixel values.

5.1 Laser Reflectance Strength Attached to Range Images

Laser range finders measure distance by shooting a laser and receiving its reflection from the target object. The distance to a particular point on the target object is computed by measuring the time duration between the laser shot and received back time in time-of-flight range finders, by measuring the phase difference in phase-transition based range finders, or by optical triangulation of the illuminant, surface, and optical sensors. In either case, the ratio of the discharged laser strength and the reflected laser strength can be returned per each 3D point. We will refer to this additional attribute of range images obtained from laser range finders as laser reflectance strength (LRS). As the laser can be considered as light with a very narrow wavelength distribution, almost a single value, the behavior of the reflected laser on the target surface can be considered the same as the general light reflection. Namely, almost isotropic reflection analogous to diffuse reflection and sharp reflection distributed around the perfect mirror direction analogous to specular reflection occurs. Since the laser reflected in the perfect mirror direction will not be observed from the range finder direction, the portion of the laser that is reflected back can be considered to be caused by this diffusive reflection. Figure 5.1 depicts two images using the LRS values attached to each 3D point as pixel values, rendered from the view point of the laser range finder.

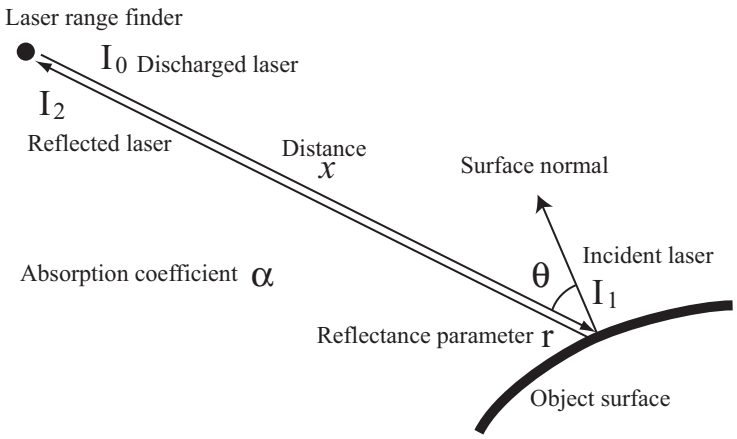


Figure 5.2: Reflection model of a laser light

LRS values are considered to depend on the characteristics of the surface, the incident angle of laser light, and the distance from the sensor. The LRS value which we obtain by a laser range finder is the ratio of the discharged laser strength and the reflected laser strength. If we assume that the LRS value depends only on the diffusive reflection, the relationship of the LRS value and the other parameters are represented by the following equation:

$$I_1 = I_0 e^{-\alpha x} \tag{5.1}$$

$$I_2 = r I_1 e^{-\alpha x} \cos\theta, \tag{5.2}$$

where I_0 is the discharged laser strength, I_1 is the incident laser strength on the surface, and I_2 is the reflected laser strength. As for the other parameters, x is the distance from the laser range finder; α is the absorption coefficient of the laser; r is the reflectance parameter of the surface; and θ is the incident angle of the laser (see Figure 5.2). The LRS value which we obtain by a laser range finder is I_2/I_0. Then, (5.1) and (5.2) become

$$\frac{I_2}{I_0} = r e^{-2\alpha x} \cos\theta. \tag{5.3}$$

Since the reflectance parameter r is peculiar to the surface, we want to obtain r by using several observations from various viewpoints.

Since we can obtain I_2/I_0, x and θ for each vertex of range images, the unknown variables

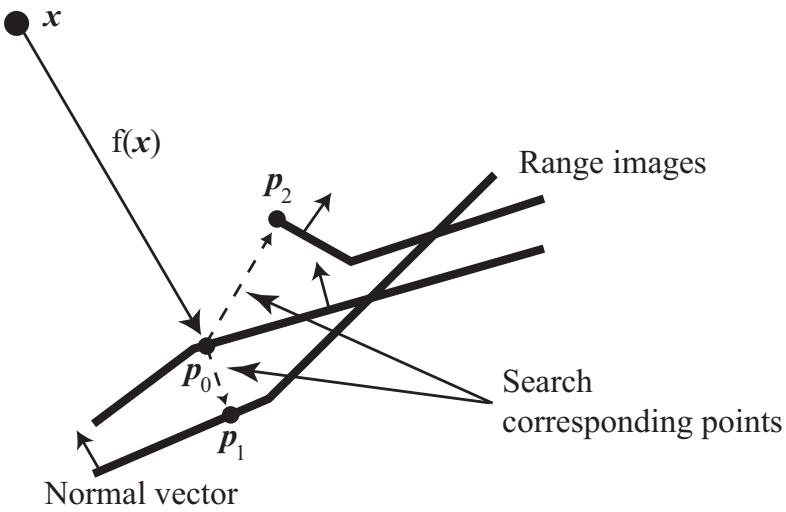


Figure 5.3: Finding corresponding points in taking a consensus of the range images

are r and α. The logarithm of (5.3) becomes

$$\log \frac{I_2}{I_0} = \log r - 2\alpha x + \log \cos \theta. \tag{5.4}$$

Therefore, the system becomes a linear equation with two unknowns. Since we find corresponding points of the range images in taking a consensus of range images, as shown in Figure 5.3 (same with Figure 2.3), we can solve the system if more than two corresponding points are found. If we have more than three equations, we can solve the system by the least square method.

Another method to estimate the reflectance parameter r is calibrating the absorption coefficient α before scanning a target. Since the absorption coefficient α depends on the atmosphere around the environment of the target, α can be assumed constant to all points of range images which are acquired at a time. If we measure the same point from a fixed direction with varying distances, we can estimate α by fitting α to the following equation:

$$y = -2\alpha x + c, \tag{5.5}$$

where $y = \log I_2/I_0$ and $c = \log r + \log \cos \theta$. Then, we compute r by using $I_2/I_0, x, \theta$ and α for each point of the range images. In the merging process, we take a consensus of r of the corresponding points of the range images. The method used to take the consensus

of r is the same as that used with the color/intensity of range images. We will describe it in 5.2.

5.2 Color/Intensity Images Attached to Range Images

Some range sensors can acquire a color/intensity image simultaneously with a range image. For example, laser range sensors which measure distance by triangulation, such as Minolta VIVID [58], use CCD for sensing laser lights emitted from the illuminant; thus, they can acquire a color/intensity image while scanning an object. Also, structured-light range finders [81, 36] and stereo range finders [69, 15] use cameras for sensing. When we use those sensors, we obtain color/intensity values as photometric attributes for each point of range images. Otherwise, we can use 2D images taken separately by aligning to the range images obtained by the laser range finders [49, 89].

In the case of color/intensity values attached to range images, the light sources vary, including, for example, the sun and electric lights. As is well-known, the color variation on the object surface is composed of two reflection components: the diffuse reflection and the specular reflection. While the diffuse reflection is almost independent of the viewing direction and its strength varies depending on the illumination direction, the specular reflection changes its strength drastically depending on the viewing direction and illumination direction. For simplicity, we considered a situation where the color images of the target object were taken under a static illumination environment, with only the viewing direction varying, while the object stays static. This assumption can be made naturally when scanning objects with laser range finders, especially when the target object is large, and also when scanning small objects to build 3D models for use in object recognition tasks in indoor scenes. Though some researchers [21, 80, 68] have estimated the radiance and positions of light sources to analyze the reflectance properties of the surface, we simply computed the reflectance property by taking a consensus of the color/intensity values attached to the range images without estimating the light sources.

As in the case of LRS, natural light has an almost isotropic reflection analogous to diffuse reflection and sharp reflection distributed around the perfect mirror direction analogous to specular reflection. Since the light reflected in the perfect mirror direction will not be observed from the range finder direction, the portion of the light that is reflected back can

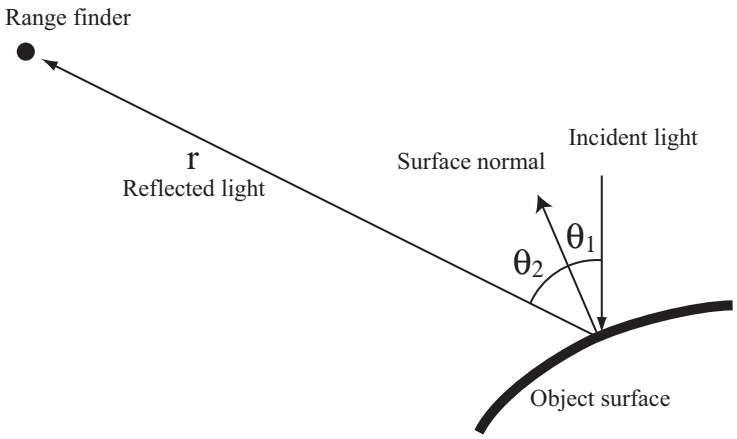


Figure 5.4: Reflection model of natural light

be considered to be caused by this diffusive reflection. If we assume that the radiance and position of light sources are static during scanning, the color/intensity values are almost view-independently constant. Namely, θ_1 is constant of all range images and θ_2 varies according to the positions of the range finders in Figure 5.4. Since they do vary slightly depending on the scanned direction [61] besides the occasional specular reflection, we computed the parameter of the diffuse reflection under a static illumination environment by taking a consensus of the color/intensity values r attached to the range images.

As the illumination direction can be considered to be static for all color images, the intensity variation of each 3D point on the target object surface should have a DC component because of the invariant diffuse reflection with a sharp peak caused by specular reflection added to it, which can be observed from a narrow viewing direction. Thus, if each 3D point is observed from a sufficient number of viewing directions, the histogram of the intensity values should have a sharp peak at the diffuse reflection value, with some distribution around it due to image capturing noise. Figure 5.5 depicts an example of this from real data. Based on this consideration, by taking the the median from multiple range images inside each voxel and assigning it to each voxel, we can determine the color/intensity values to be attached to the resulting 3D model.

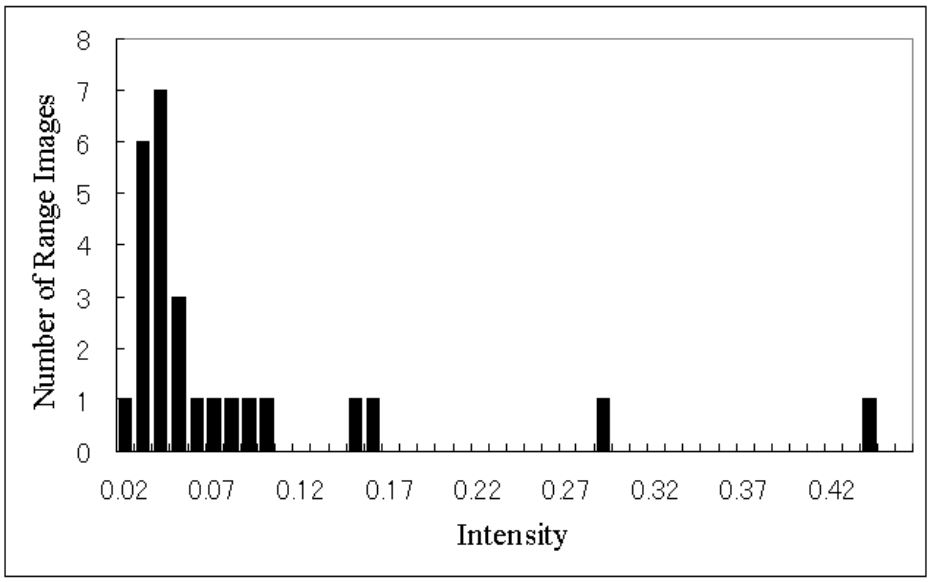


Figure 5.5: An example of the histogram of the intensity values of consensus points. Some outliers due to specular reflection are observed. In this case, the median value is 0.04.

5.3 Subdividing Voxels Based on the Photometric Attributes of Range Images

We have introduced a new criterion of voxel subdivision based on the geometric attributes of the surface for the adaptive merging method in 4.2. Voxel subdivision based on the variation of photometric attributes can be accomplished in a similar manner. When the voxel is subdivided depending on geometric attributes only or without consideration of any attributes, the appearance of the resulting object will be significantly smoothed out. Since ordinary shaders such as smooth shading and phong shading will simply interpolate the intensity values attached to each vertex, this smoothing is unavoidable. However, if we can triangulate the 3D mesh model with regards to the appearance variation, i.e., fine around appearance boundaries and having each triangular patch contain almost the same texture color, simple shading will work dramatically well. Also, the 3D models tessellated with regards to the texture variation of the models are useful to accomplish further texture analysis and synthesis. For instance, view-dependent texture mapping like [67] can achieve higher compression, since global texture compression stacking triangular

patches with a similar texture can be applied.

In a similar manner to subdividing by the curvature of the surface, our method computes the variation of photometric attributes of 3D points inside the voxel of interest. Now, c_i, c_j are the photometric attributes of neighbor points included in a range image. If the maximum difference satisfies

$$\max_{i,j}(\text{Distance}(c_i, c_j)) < \delta_c, \tag{5.6}$$

where δ_c is the threshold and Distance(c_i, c_j) is the function which computes the difference of two photometric attributes, the sampling interval is fine enough for the range image.

Our method also takes a consensus while considering the photometric attributes. Similar to (4.3), our method does not subdivide the voxel if

$$N_c > T_c, \tag{5.7}$$

where N_c is the number of range images which satisfy (5.6), and T_n is the threshold of consensus for the photometric attributes. Similar with subdivision based on the geometric attributes, we can take another approach to avoid erroneous subdivisions of voxels by taking a consensus of range images before the merging process. By computing the consensus of vertices a priori, we only use vertices which have sufficient consensus for computing (5.6).

5.4 Experiment of Photometric Merging

Figure 5.6 shows the merging results of the Great Buddha of Kamakura with the LRS values as photometric attributes. Figure 5.6(a) is one of the range images with LRS. By taking a consensus of the LRS values, we constructed a 3D model of the Buddha with LRS (Figure 5.6(b)). Figure 5.6(c) and (d) are the result of the adaptive merging technique which we have proposed in Chapter 4, which subdivides voxels only based on the geometric attributes. Thus, the texture of LRS is smoothed out compared with Figure 5.6(b). Figure 5.6(e) and (f) are the result of adaptive merging with voxel subdivision based on the photometric attributes proposed in 5.3. Figure 5.6(g) and (h) are zoom-ups of the forehead of Figure 5.6(c) and (d). Similarly, Figure 5.6(i) and (j) are zoom-ups of the forehead of

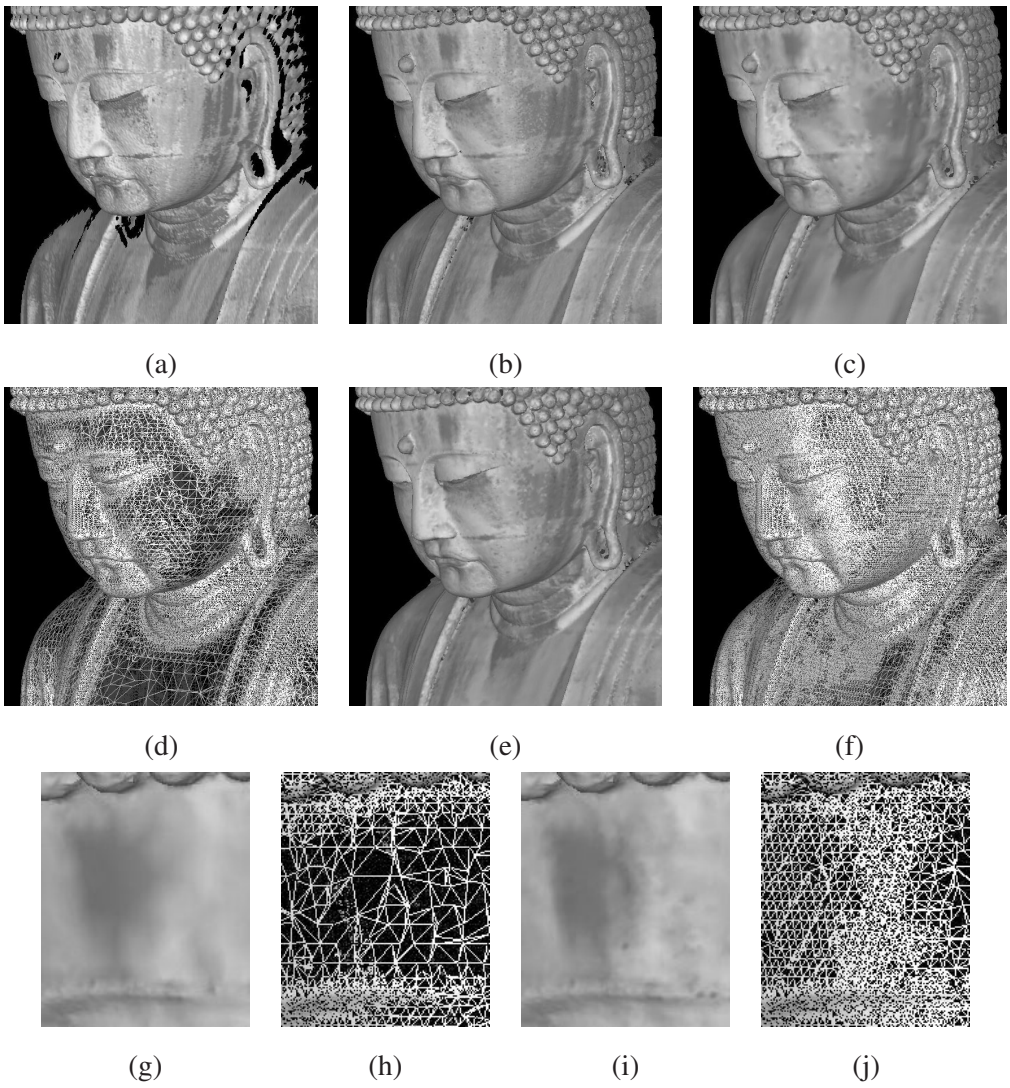


Figure 5.6: Merging results of the Great Buddha of Kamakura with the LRS values

Figure 5.6(e) and (f). By considering the photometric attributes as the criterion of voxel subdivision, the sharp edges due to the variation in LRS values are well preserved. The statistics of the merging results with photometric attributes are shown in Table 5.1. Adaptive merging based on the photometric attributes successfully reduces the amount of data and the computational time, while well preserving the edges of the LRS values.

Table 5.1: Statistics of models of the Great Buddha of Kamakura with photometric attributes

	Number of points	Time for Merging	Mean Difference
(b)	3.0 million	61 min.	N/A
(c),(d)	1.4 million	25 min.	0.99 mm
(e),(f)	1.7 million	30 min.	0.44 mm

5.5 Applications

As an example of applications utilizing the photometric attributes of the attached 3D model, a robust 2D-3D registration can be considered [49]. Though in this paper we mainly presented the results of our studies on the geometric modeling of objects, the next step of modeling is the photometric modeling of objects. As a method to acquire photometric information, texture acquisition using a camera is considered; which requires knowing the posture of the camera at the time that it acquired a texture image, to combine with the geometric model of the object.

A laser range finder is often much heavier than a camera; for example, Cyrax 2500 [18] is about 20kg. Therefore, the viewing positions of laser range finders are restricted, because the finders cannot be set up at an arbitrary position. If we calibrate a camera with the range finder a priori, the viewing position of the camera also become restricted. However, if we consider view-dependent photometric information such as BRDF [64], many images acquired from various directions are necessary.

Thus, it is useful to estimate the posture of a camera using a texture image and a geometric model which have already been acquired. Wheeler [96] have proposed a technique for aligning a 2D image and a 3D model which compares the edges of the intensity values of the 2D image and the occluding edges which are obtained when the 3D model is rendered from a viewpoint. Kurazume et. al. [49] have extended the technique to use photometric attributes attached to a 3D model.

Figure 5.7 shows an example of aligning a 2D image and a 3D model of the Kamakura Buddha. Figure 5.7(b) is the edges of the color values extracted from Figure 5.7(a) using a Canny filter [9]. Figure 5.7(c) is the geometric model to be aligned to the image. The

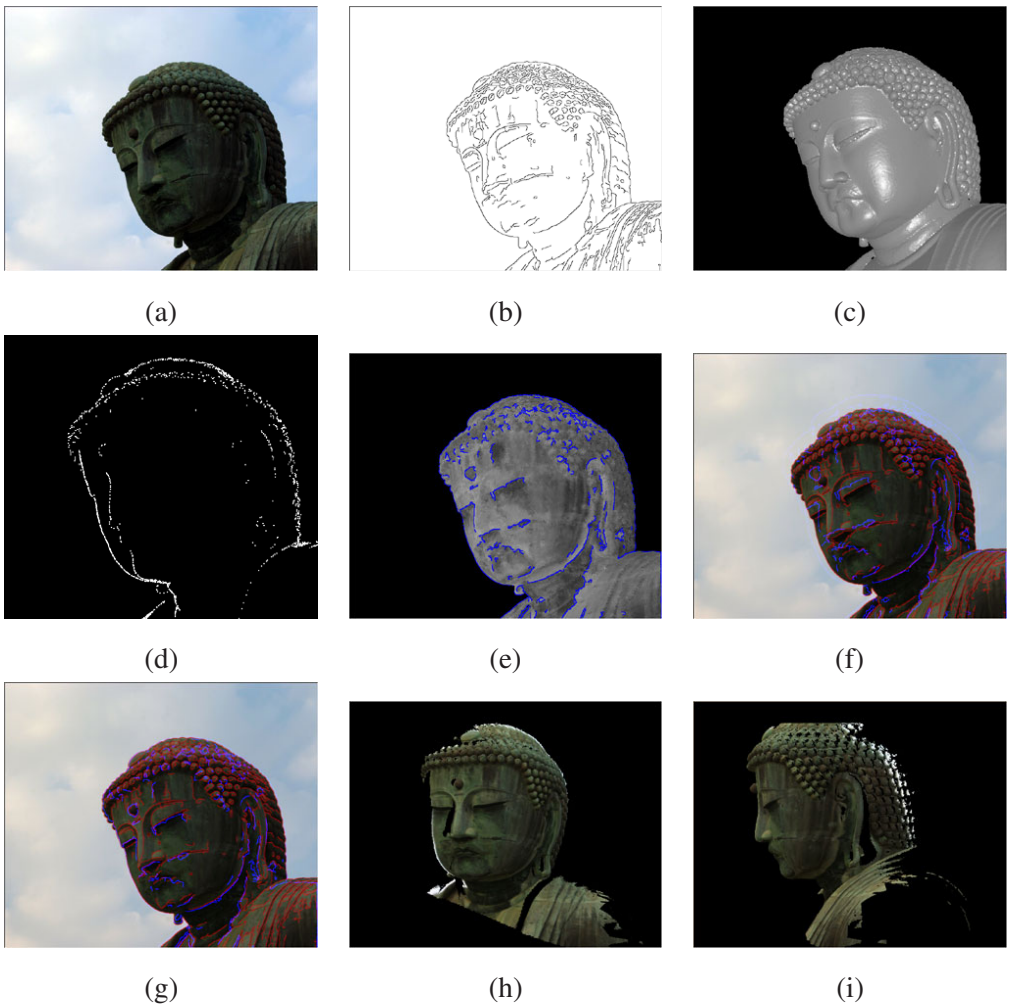


Figure 5.7: Aligning a 2D image with a 3D model of the Kamakura Buddha using the photometric attributes of the 3D model

original algorithm used occluding edges such as Figure 5.7(d). Figure 5.7(d) is the edges of the LRS values extracted from the 3D model(Figure 5.7(e)). The method estimates the posture of the camera by taking matching edges of 2D image (red lines) and 3D model (blue lines). Figure 5.7(f) is the initial posture of camera before iterative computation and the posture converges to Figure 5.7(g). Finally, texture mapping is accomplished using estimated camera parameter(Figure 5.7(h) and (i)).

5.6 Summary

We have proposed an extension of the merging framework, which consists of the merging of photometric attributes which are attached with range images. By taking a consensus of the appearance changes of the target object from multiple range images, we reconstructed a 3D model with an appearance which successfully discards outliers due to noise. Also, we were able to provide a model with Lambertian reflected light values by discarding specular reflection as outliers. The photometric attributes of the model can be used for aligning with 2D color images.

Chapter 6

Refining Range Images

The errors of the final 3D model are considered to come from the following factors:

(a) A measurement error on the range images

(b) A matching error on the aligning process

(c) A quantizing error on the merging process

The merging algorithm described in Chapter 2 and Chapter 4 reduces the error using multiple range images by taking a consensus of them. However, since the signed distance is calculated along the normal direction of the surface, it is weak if range images have high frequent errors, and, in such a case, the normal directions are not reliable.
The merging process consists of two components:

- De-noising range images
- Generating a unified mesh model

The former component accomplishes the improvement of the accuracy of the measurement. In this chapter, we propose a novel method [78] to improve the accuracy of the measurement in place of taking a consensus of the range images in the merging process. The error distribution of a range image is anisotropic according to the line of sight of the scanning. If we observe a surface from various directions, we can improve the precision of the measurement by considering the multiple error distributions of different directions.

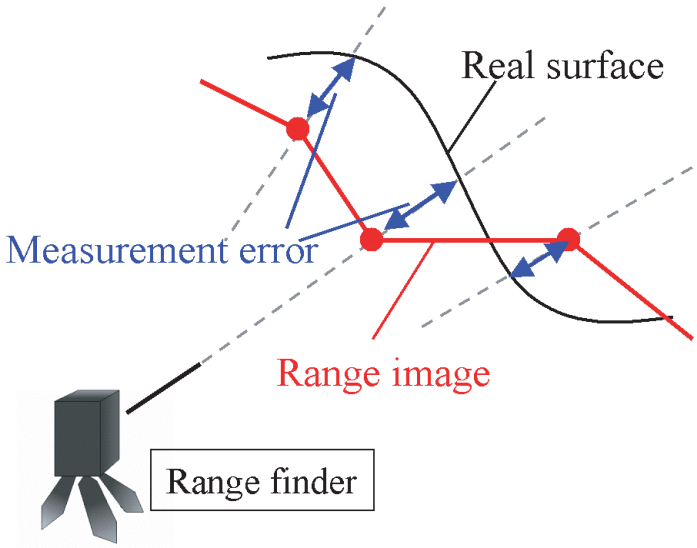


Figure 6.1: Measurement error

6.1 Error model of Range Measurement

Laser range finders measure distance by shooting a laser and receiving its reflection from the target object. The 3D position of the point of reflection is computed by the distance and the ray vector. The error of the 3D position mainly depends on the error of the distance. The error of the vertical direction to the ray vector, which is caused by the mechanism of the range finder, is much smaller than the error of the distance. Thus, we assume the error of the range measurement by a laser range finder is anisotropic and exists only along the ray vector (Figure 6.1).

6.2 Algorithm Overview

Our method corrects errors by iterative computation similar to registration techniques like ICP[6, 63]. Let us call the base range image the 'model' and the others the 'scenes.' We first search the corresponding points on all scenes of each vertex of the model. Then, we move every vertex of the model respectively to reduce the distance of each correspondence. Our method continues this process until the distances become sufficiently small.

Algorithm 6.1 RefineRangeImages(R_{set})

Input: Set of Range Images: R_{set}

```
while e > θ do
  for all R_i ∈ R_set do
    for all v_ik ∈ R_i do
      V_set ← 0
      for all R_j ∈ R_set do
        if R_i ≠ R_j then
          v ← CorrespondenceSearch(v_ik, R_j)
          V_set ← V_set ∪ v
        end if
      end for
      v'_ik ← ComputeNewPosition(v_ik, V_set)
      Compute error e_ik between v_ik and V_set
    end for
  end for
  Update vertices v_ik → v'_ik for all i, k
  Compute average e of e_ik for all i, k
end while
```

Algorithm 6.1 shows the proposed algorithm. R_{set} is the set of input range images. For each vertex v_{ik} of the model range image R_i, we find the corresponding points of other scene range images R_j. Then, we compute the new vertex position v'_{ik} using v_{ik} and V_{set}. After computing the new position of all vertices, we update all vertices of the range images. We iterate this computation until the average error e becomes smaller than a user defined threshold θ.

6.3 Correspondence Search

Since we assume that error exists only along the ray vector and that range images are completely aligned, our method searches corresponding points along the ray vector. Now, $\vec{x}$ is the vector from the center of the sensor to the vertex of the model, and $\vec{y}$ is the vector

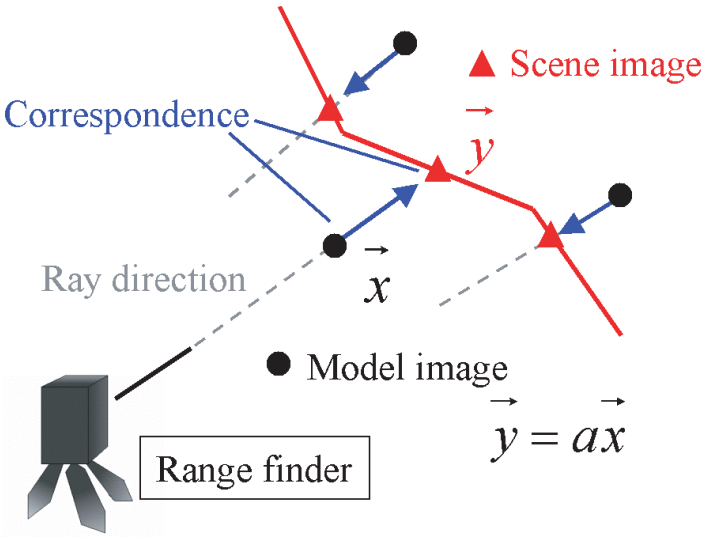


Figure 6.2: Search correspondence

from the center to the corresponding point of the scene. Then,

$$\vec{y} = a\vec{x} \tag{6.1}$$

where a is the coefficient. Thus, these points are on the same line (Figure 6.2).
To eliminate wrong correspondences, if the distance of the corresponding points is larger than a threshold, we remove the scene point from the correspondence. We use the maximum error of the range finder as the threshold. This correspondence search is computed for every combination of range images.

6.4 Computing New Vertex Position

Errors are corrected by moving each vertex to the new position, which is estimated from the corresponding points. Since the direction of error of each range image is different, some correspondences are not accurate. If the number of overlapped range images is small, it is difficult to estimate an accurate point. Thus, we move each vertex to the weighted average point of the correspondence in order to gradually converge the error.

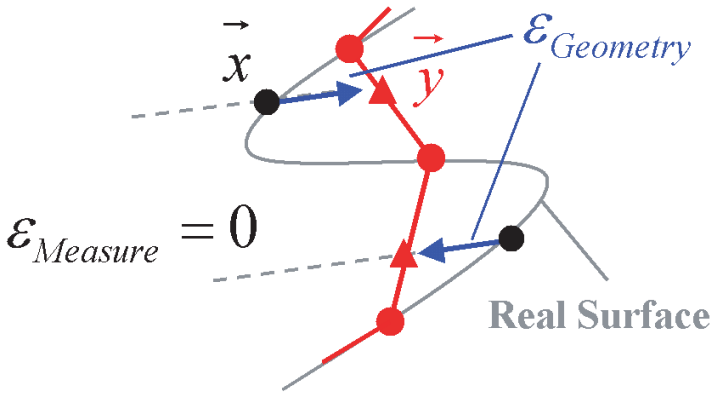


Figure 6.3: Error by resolution and geometry

The kth vertex of the ith range image $\vec{x}_{ik}$ is moved to the weighted average point

$$\vec{x}'_{ik} = (1 - w) \cdot \vec{x}_{ik} + w \cdot \frac{1}{n_{ik} - 1} \sum_{i \neq j} \vec{y}_{jk} \tag{6.2}$$

where n_{ik} is the number of corresponding points, and w is the weight. In this paper, we use $w = 0.5$. This process is applied to all vertices of each range image. We reiterate it until the error of correspondence converges sufficiently.

6.5 Discussion

The error of corresponding points ϵ depends on the error of measurement $\epsilon_{Measure}$ and the error by sparse sampling of a range image $\epsilon_{Geometry}$.

$$\epsilon = \epsilon_{Measure} + \epsilon_{Geometry} \tag{6.3}$$

$\epsilon_{Measure}$ is corrected by iterative computation. However, $\epsilon_{Geometry}$ is caused by the curvature of the surface and the sampling interval of a range image.

In Figure 6.3, the range measurement is noise-free and the vertices of the range images are on the real surface (namely $\epsilon_{Measure} = 0$); however, the error exists between $\vec{x}$ and $\vec{y}$. Thus, $\epsilon = 0$ only if the surface is planar.

$$\epsilon \begin{cases} = 0 & \text{planar area} \\ > 0 & \text{otherwise} \end{cases} \tag{6.4}$$

Figure 6.4 shows a 2D example of range images. For simplicity, we assume the new positions of x_2, y_1, y_2 after an iteration is computed as

$$\begin{aligned} x_2' &= (1-w)x_2 + w((1-\alpha)y_1 + \alpha y_2) \\ y_1' &= (1-w)y_1 + w((1-\beta)x_1 + \beta x_2) \\ y_2' &= (1-w)y_2 + w((1-\gamma)x_2 + \gamma x_3) \end{aligned} \tag{6.5}$$

where w, α, β, γ are coefficients. Coefficients α, β, γ are determined by the correspondence of x_1, x_2, x_3, y_1 and y_2. After one more iteration, x_2' moves to

$$\begin{aligned} x_2'' &= (1-w)x_2' + w((1-\alpha')y_1' + \alpha' y_2') \\ &= w^2(1-\alpha')(1-\beta)x_1 + w^2\alpha'\gamma x_3 + \\ &\quad ((1-w)^2 + w^2(1-\alpha')\beta + w^2\alpha'(1-\gamma))x_2 + \\ &\quad w(1-w)(2-\alpha-\alpha')y_1 + \\ &\quad w(1-w)(\alpha+\alpha')y_2 \end{aligned} \tag{6.6}$$

where α' is a coefficient. Since the equation of x_2'' includes the neighbor vertices x_1, x_3, the smoothing effect occurs during an iteration similar to the smoothing filter. However, the weight of x_1 and x_3 in (6.6) is small compared with that of the smoothing filter, for example,

$$x_2' = \alpha x_1 + \beta x_2 + (1-\alpha-\beta)x_3, \tag{6.7}$$

which does not include y_1, y_2. Thus, the propagation of the smoothing effect of our method is slower than that of the smoothing filter. In our present implementation, we determine the number of iterations by estimating manually whether the iteration is sufficient.

With regard to the computation cost, most of the computation cost is due to searching for correspondences. Our algorithm searches for correspondences of vertices by rendering all range images from the viewpoint of the reference range image. Thus, the order of computation is $O(MN^2)$ for an iteration of refining all range images, where N is the number of range images, and M is the number of vertices of a range image.

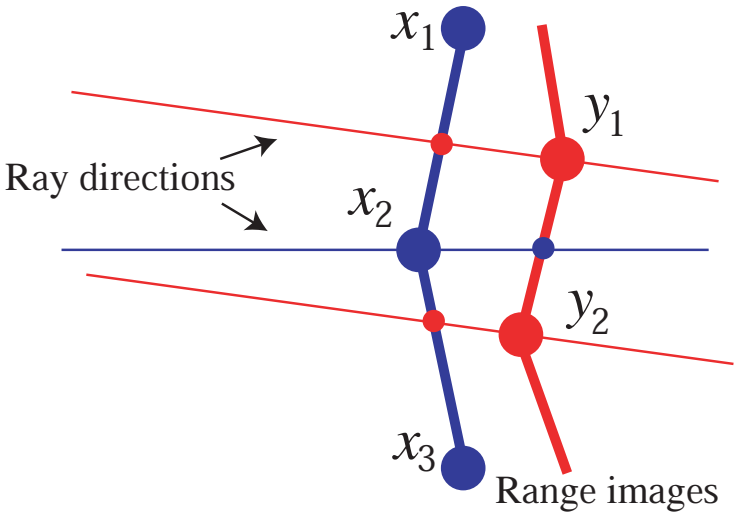


Figure 6.4: Smoothing effect by iteration

6.6 Experiment

6.6.1 Error Distribution of Laser Range Finder

Among the types of laser range finders, time-of-flight range finders are useful for measuring a distant object with high accuracy. We use a laser range finder of the time-of-flight type, Cyrax 2500 [18] made by Cyra Technologies, Inc. To estimate the error distribution of the Cyrax 2500, we set the range finder in front of a concrete wall, and measured the distance to the wall many times. We tested three configurations of different distances, the far range (67m), middle range (20m), and near range (2m) The recommended range of Cyrax 2500 is 1.5–50m. Figure 6.5 shows the results of the measurements, and the average, variance, and standard deviation are depicted in Table 6.1. The error distribution becomes wide in the near range; however, it can be regarded as a normal distribution with about 3mm standard deviation. The maximum error is about 7–8mm, which is a little larger than the 6mm (at 50m range) of the catalog specification.

We did not test the error distribution of the vertical direction to the ray vector. According to the catalog, it is 0.25mm at a 50m range (0.0003 degree), which is drastically smaller than that of the ray direction. Thus, the error distribution of the range image by Cyrax

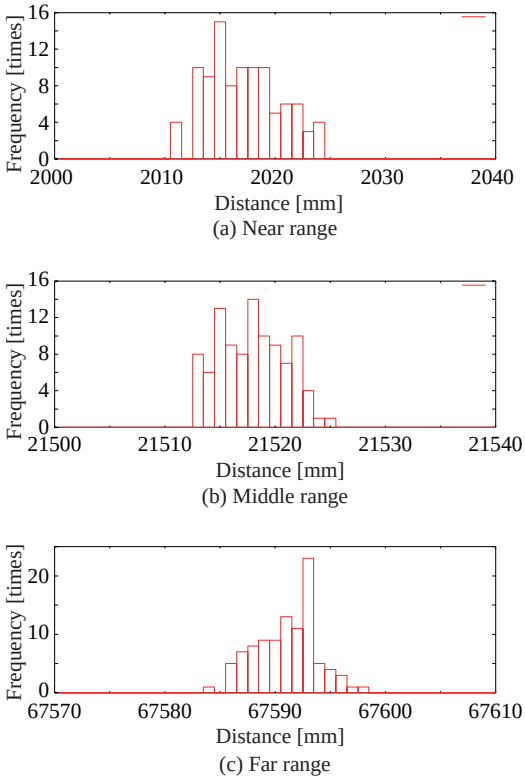


Figure 6.5: Distribution of errors of Cyrax2500

2500 depends on the ray direction.

6.6.2 Error Correction of Artificial Data

First, we created artificial range images with random noise, and experimented with error correction. Figure 6.6(a) shows the model without noise. Its width and height are 40cm, its depth is 20cm, and it consists of 100×100 points. The range image with noise, of which the maximum error is 6mm, is Figure 6.6(b). We created 10 range images, to which were added noises coming from different directions. The result of error correction is shown in Figure 6.7(a). Figure 6.7(b) is one of the range images filtered by the Gaussian filter. It can be seen that our method corrects error sufficiently and preserves edges more accurately than the Gaussian filter. Figure 6.8 compares these two results.

We used a PC with a PentiumIII 866MHz processor and a NVIDIA GeForce3 graphics card. Since our method searches correspondences by rendering range images, it can be ac-

Table 6.1: Distance measurement error of Cyrax 2500

Average distance $[mm]$	Var. $[mm^2]$	STD. $[mm]$
2017.2 (near)	11.0	3.3
21518.0 (middle)	9.1	3.0
67591.1 (far)	7.7	2.8

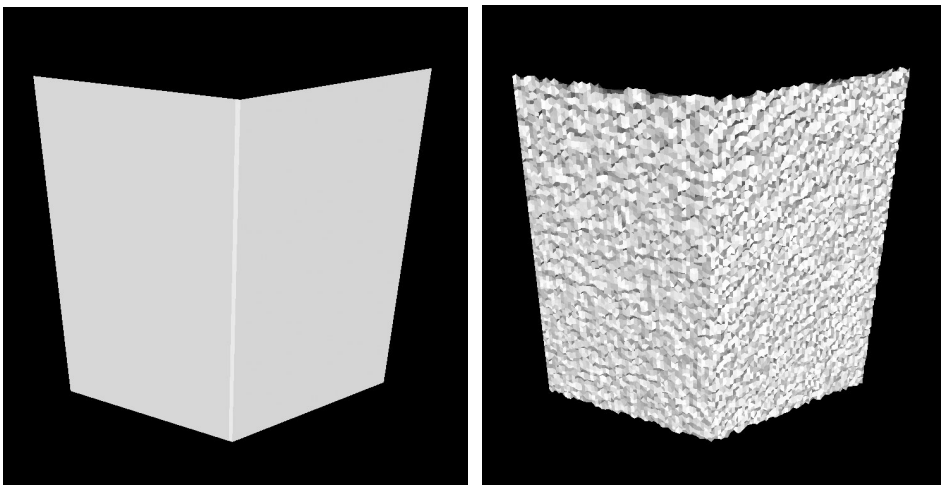


Figure 6.6: Artificially created model

celerated by the use of graphics hardware. The computation time required for 20 iterations is 413 seconds.

6.6.3 Error Correction of Real Data

Next, we experimented on the error correction of range images acquired by a laser range finder Cyrax 2400, whose accuracy is the same as that of the Cyrax 2500.
The object observed was the Asuka Great Buddha in Nara, which is considered to be the oldest statue of Buddha in Japan. The height of the statue is about 2.7m. Cyrax 2400 is a range finder used for long range, but it is not suitable for measuring objects of the Asuka Buddha's size. Because the surroundings of this Buddha prevented us from getting close

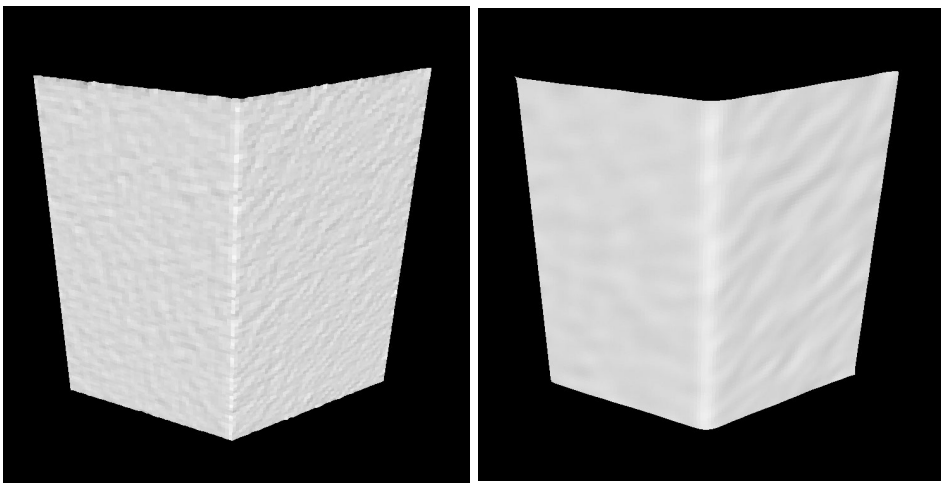


Figure 6.7: Refined model

to it, we were not able to use a more accurate range finder for near range distances. Thus, we measured the Buddha from a short distance away by using the Cyrax 2400.

We acquired 9 range images of the front of the Buddha. We aligned these range images simultaneously by using a robust registration method [66] (see Figure 6.10). Since the object was relatively small and the range was near, the noise of the range image can be obviously seen in the range images.

Figure 6.11 shows the result of the error correction by our method. The noise was removed from the model, and its surface is smooth. In spite of that, the edges are well preserved, because our method, unlike the Gaussian filter, is not a smoothing operation. In Figure 6.12, the error converges to 0 as the number of iterations increases. The computation time for 20 iterations was 1510 seconds.

Figure 6.13 shows the results of merging range images using both the original range images and the refined range images. We used the merging technique created by Sagawa [76]. Since the method takes a consensus of range images using the distance and normal direction [97], it can remove errors caused by measurement and registration. However, in this experiment, it was difficult to correct the errors, because the range images were too noisy and the normal direction of each vertex of the range images cannot be relied upon. An

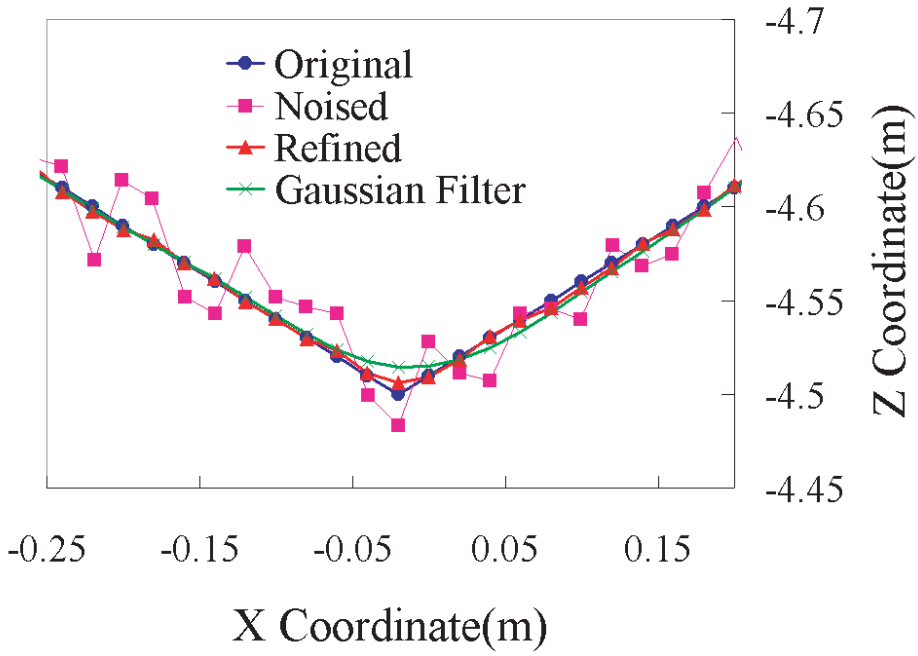


Figure 6.8: Compare our method and the Gaussian filter

error remained in the merging result when the original range images in Figure 6.13 were used. However, an accurate model was reconstructed in the merging result with the refined range images. In the area where range images were not overlapped, such as the side of the head, the error was not removed.

We compared our method with the model filtered by the Gaussian filter. Figure 6.14 shows the part of the model filtered by the Gaussian filter after merging, and the model that was merged from the refined range images. The model with the Gaussian filter was smoothed out; however, our method removed noise and preserved the edge of the surface.

Finally, we considered whether our method can be applied to other range finders, for example, a stereo range finding system. We constructed a multi-baseline stereo system [69], which consisted of 9 cameras. Figure 6.15 shows one of the camera images and a stereo range image. Since the multi-baseline stereo generated a range image for each camera by taking matching images with the other 8 cameras, we generated 9 stereo range images. These range images were pre-aligned by stereo calibration. Thus, our refining process

Figure 6.9: Great Buddha at Asuka Temple

could be applied directly to the 9 range images. In the raw stereo range image (Figure 6.16(a)), we can see the step-shaped error caused by the quantization of the images. Figure 6.16(b) is the refined model after iteration was performed 10 times. The step-shaped error was removed after refinement. Also, Figure 6.16(a) contained a lot of debris, due to mismatching. Since the refining process is an estimation of the confidence of the range data, we can regard the vertices of the range image which cannot be refined as unreliable vertices. Figure 6.16(c) is the range image after the unreliable vertices were removed.

6.7 Summary

In this paper, we have proposed an efficient range image refining method, taking into consideration the unique error distributions of multiple range images. We described how we applied this method to model artificial test objects and actual cultural heritage objects from

Figure 6.10: Original range image of Asuka Buddha

the images acquired by a time-of-flight range sensor. Finally, we applied our method to the range images which were generated by the multi-baseline stereo system. The experimental result shows the validity of this method compared with that of the existing filter-based methods.

Figure 6.11: Refined range image of Asuka Buddha

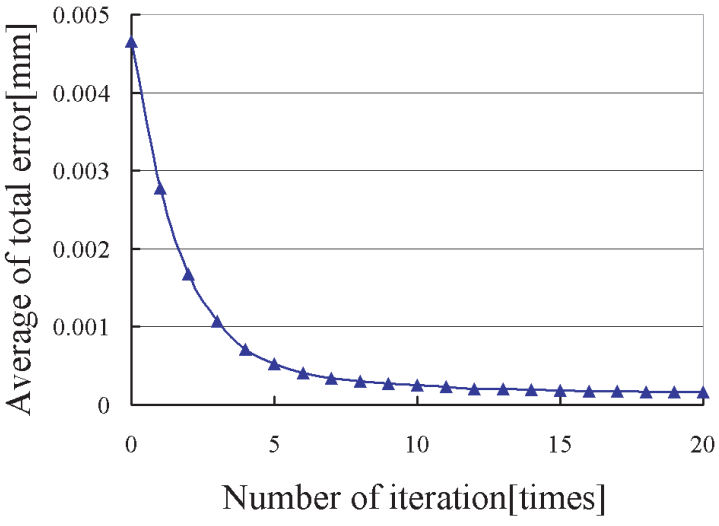


Figure 6.12: Convergence of error

Figure 6.13: Results of merging of Asuka Buddha

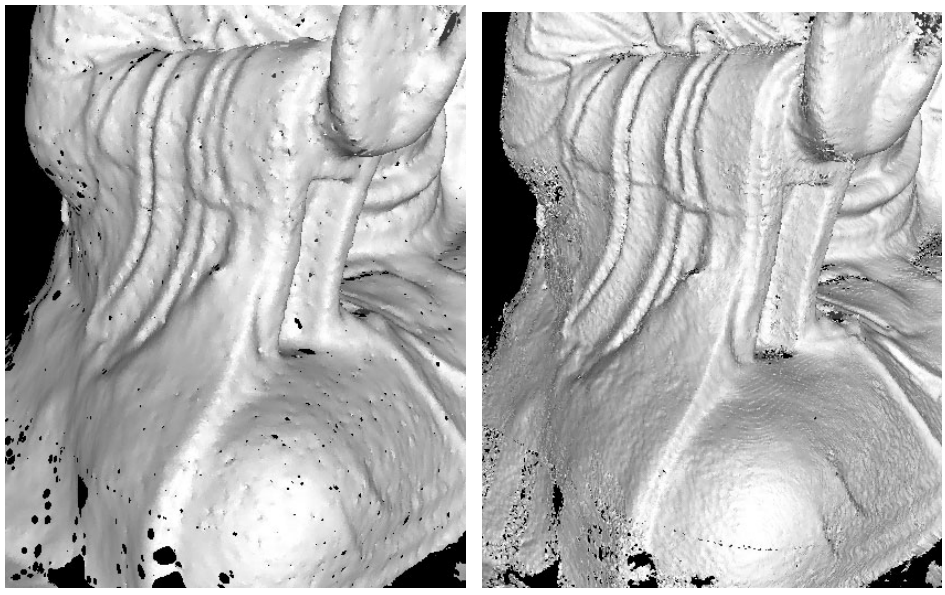
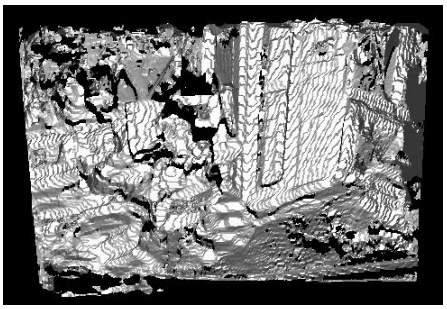


Figure 6.14: Comparison with Gaussian filter

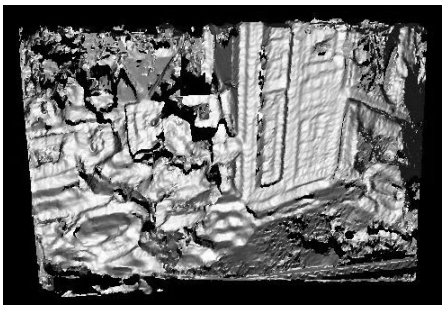
Original camera image

Stereo range image

Figure 6.15: Range measurement by stereo



(a)Raw model by stereo



(b)Refined model

(c)Remove unreliable vertices

Figure 6.16: Refining range image by stereo

Chapter 7

Flying Stereo Range Sensor

Our target objects were mainly cultural heritage objects. Since some cultural heritage objects are very large, it is difficult to completely cover the surface of such objects using the usual range finders. The reasons why the surfaces cannot be observed from the sensors were considered to be due to the following two reasons. First, the surface is self-occluded or occluded by other objects from any viewpoints. Because most cultural heritage objects are intricately shaped, the line of sight for a laser range finder for some points of the surface cannot be obtained. To acquire a range image of intricately-shaped surfaces, it was necessary to develop a new device which was completely different from the usual range finders.

Second, if the line of sight for a range finder can be obtained, it is difficult in some cases to settle a sensor stably at the best point of view. The laser range finders we mainly used to acquire the range images of these objects were set up on the ground. Thus, the large part of the unobservable surfaces of those sensors was the upper side of the object surface. To acquire a range image of the upper surface, it was necessary to develop a new method to acquire the range image from the air. In this paper, we report on our development of a new range finder which is loaded on a balloon, and which acquires range images from the air.

7.1 Overview of Range Sensing System Sent Aloft by Balloon

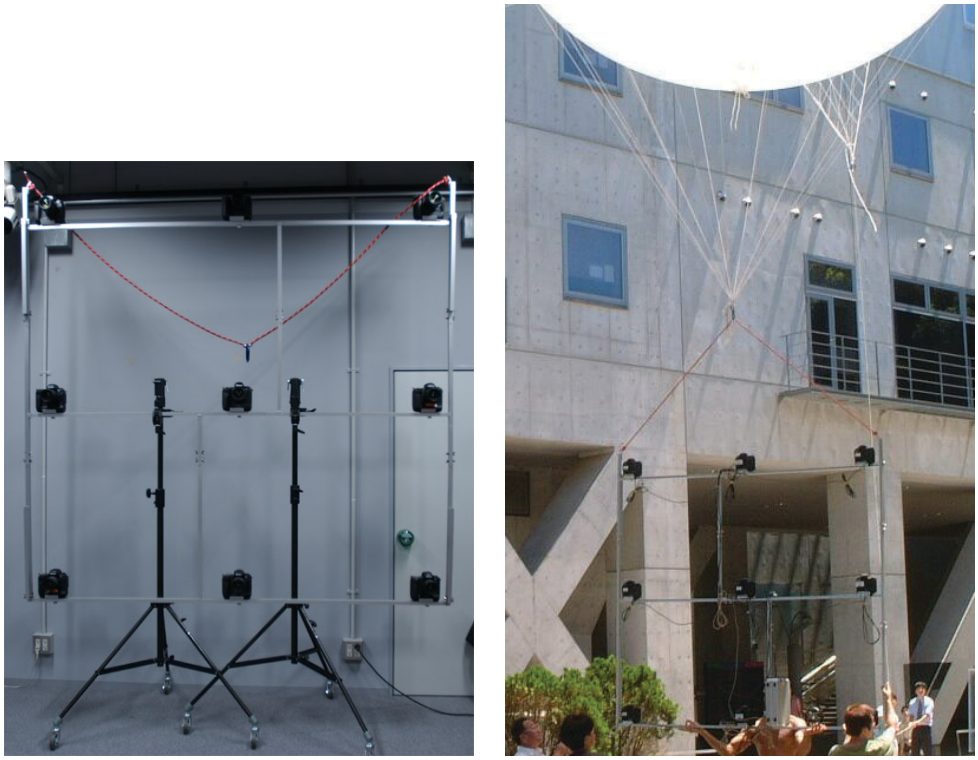
The problem with mounting a range finder on a balloon and sending it aloft is that the range finder cannot remain stationary, and moves due to wind. A typical laser range finder requires about 10 seconds to several minutes for each scan. Since it is necessary for the sensors to remain stationary while scanning, we cannot load send them aloft by balloon. To solve this problem, the following two approaches were considered:

- Measuring the motion of the sensor during scanning
- Scanning during a moment in which the motion of the sensor can be ignored

The first approach measures the motion of the sensor during scanning and computes the laser direction. If we know the laser direction of each range measuring, we can construct a range image from a moving sensor. To measure the motion of the sensor, several methods were considered, such as GPS, gyroscope, and visual markers. Miller [57] developed a helicopter whose position is measured by differential GPS and by an inertial measurement unit (IMU), which scans the terrain using a laser range sensor which is loaded on it. This approach necessitates using those additional sensors to construct a system.
In this paper, we used the second approach. Since laser range finders require at least several seconds to scan, we developed a new range finder which measures distance by stereo matching. The precision of this stereo range finder is lower than that of a laser range finder; however, it has a great advantage in that acquisition is completed in less than a second. Also, since other devices are not required to augment this approach, we were able to construct a simpler system. We call our new stereo range finder, which is loaded on a balloon, a "flying stereo range finder" (FSRF).

7.1.1 System of the Flying Stereo Range Finder

Though stereo matching can be accomplished by only 2 cameras, the multi-baseline stereo matching method [69] improves the precision of range sensing, which takes matching by multiple pairs of cameras using more than 3 cameras. In the case in which we used more than 3 images captured from different viewpoints for multi-baseline stereo, two approaches were considered: motion stereo by a single camera [70, 82] and fixed multiple cameras [73, 47]. Since we need to capture various viewpoints by moving the balloon, in



(a) (b)

Figure 7.1: FSRF which consists of 9 cameras

order to cover the entire surface of the object, and because it is costly to move the balloon to arbitrary viewpoints, we developed a stereo range finder which consists of 9 cameras which generate a dense range image by the multi-baseline stereo matching method. This reduces the cost to move cameras.

Figure 7.1 shows our new stereo range finder. We placed 9 cameras as 3×3 matrix on the frame. We used a digital still camera, the Nikon D1, for each camera of the system; it acquires an image of 2000×1312 pixels. Since we need high resolution images to measure the distance accurately by stereo matching, we used digital still cameras rather than video cameras.

Figure 7.2 shows the system of FSRF. The FSRF is lifted up by a balloon, whose loading capacity is about 50kg. In our current system, the weight of each camera is 1.5kg and that of the frame, 10kg; the total weight is about 25kg. The balloon is controlled manually by

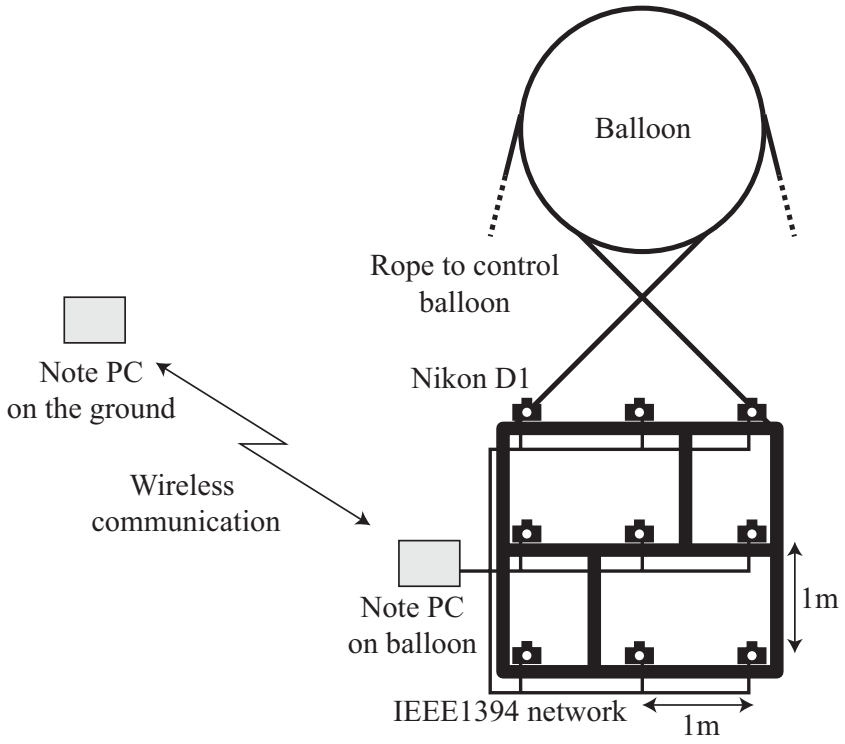


Figure 7.2: System of FSRF

rope. Though other ways to lift up the system can be considered, such as a a helicopter, we chose a balloon, since it is important to be able to control the system. The frame of the system is suspended by the balloon, and cameras are fixed to the frame. The cameras are fixed almost parallel, at an interval of 1m. A notebook PC was also loaded on the balloon, and this PC controls the cameras through an IEEE1394 network. We were able to change the parameters of the image acquisition of the cameras from the ground. We had two ways to release the cameras, by the IEEE1394 network, and by remote release cable. We controlled the system from the ground through wireless communication. The images acquired by the cameras were kept in the memory cards of the cameras; the system can also send images to the computer on the ground by IEEE1394 and the wireless network.

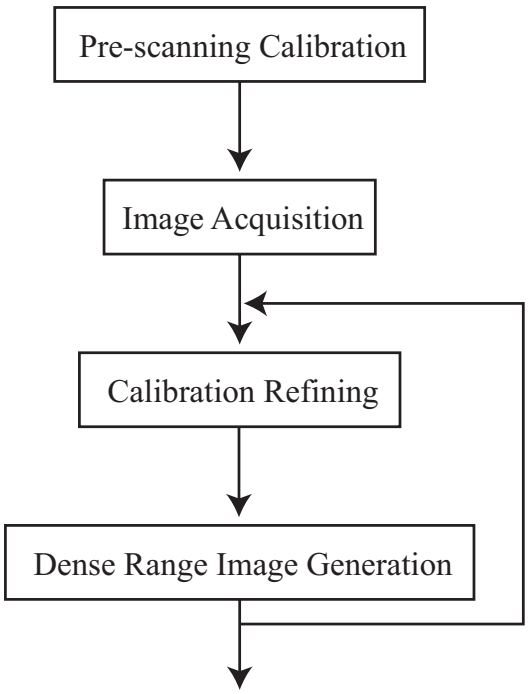


Figure 7.3: Scanning Steps of FSRF

7.1.2 Scanning Steps of FSRF

Several steps are required to generate a range image using the FSRF. Figure 7.3 shows the scanning steps of FSRF. To take matching of multiple images, we first calibrated the intrinsic and extrinsic parameters of the cameras. Since the cameras of the FSRF are fixed on a frame, we calibrated them before acquiring the images, using a visual marker (see 7.2.1).

At the time of scanning, we controlled the position of the cameras on the balloon. Since the balloon has no mechanism to move itself, it was controlled manually by rope from the ground. The images were acquired at the same time, and the acquisition was completed at a moment at which we could ignore the motion of the system. The distance to the object was typically about 10m in our experiments.

Though it is necessary to use a marker which is a similar size to the target object in calibrating the extrinsic parameters of the cameras, our target can be larger than 10m. Also, it is impossible to prepare such a large marker to calibrate the parameters. The error of

calibration becomes large if the size of the target object is much larger than that of the marker. Thus, we refined the parameters of the cameras using the image of the target itself after acquiring the images (see 7.2.2). If we computed a range image using the image sequences which were obtained by the smooth motion of the camera, the distance of the corresponding points in the images was small; however, the distance of the corresponding points when we used the FSRF became larger, since the baseline of the FSRF is wider than that of the image sequences. Therefore, we used the range images which were computed using the current parameters to find corresponding points, which restricts the area for searching corresponding points. We fed the refined parameters back to the dense range image generation recursively.

After acquiring the images by multiple cameras, we generated a dense range image by stereo matching. We used a multi-baseline algorithm [69] to generate a range image from 9 images. To accelerate the matching process, we used the algorithm of recursive correlation calculation [27] and graphics hardware accelerators to rectify the image pairs.

7.2 Calibration of Multiple Cameras

This section describes the calibration of the camera parameters of the FSRF. The method consists of pre-scanning calibration and refining calibration after image acquisition.

7.2.1 Pre-scanning calibration

Before scanning the target object, we calibrated the intrinsic and extrinsic parameters of the cameras using a visual marker. Since the error of calibration becomes large if the size of the target object is much larger than that of the marker, we used a marker which was larger than usual to scan a large-scale object. Figure 7.4 shows the marker we used, which had a cubic frame of 1.2m width. We acquired images of the cube by all cameras at the same time. We used the 8 corners of the cube as features by finding them manually, and measuring their correspondence. If we had more than 6 corresponding points, we were able to calibrate the camera parameters [81].

Figure 7.4: Cubic frame for calibration

7.2.2 Refining Calibration

To generate a precise range image, we had to refine the parameters of the cameras because the position and size of the target objects differed from those of the calibration marker that was used for pre-scanning calibration. Therefore, we refined the parameters of the cameras using the image of the target itself after acquiring the images. The steps to refining the parameters consisted of two components:

1. Matching features by utilizing a range image by the current parameters
2. Refining the parameters of all cameras by bundle adjustment [7, 93]

Restricting the Search Area using A Range Image

When we find features of an image pair using feature extraction techniques such as Harris corner detector [37], When we find features of an image pair, we have to search the entire image if we have no knowledge of camera parameters. Since this is extremely exhaustive,

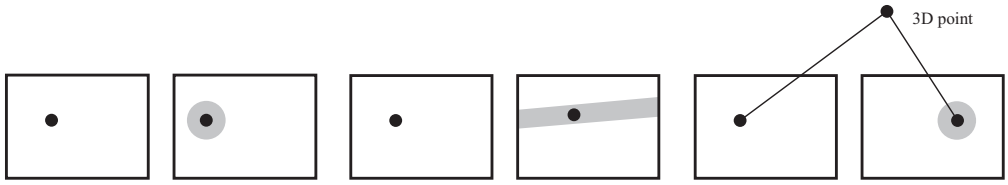


Figure 7.5: Constraints for searching corresponding feature points

several constraints were considered to restrict the search area. Figure 7.5 shows three constraints for searching corresponding feature points [3]. If we use an image sequence by the smooth motion of a camera, the corresponding feature points exist at a similar position to the reference image. Thus, we search the neighborhood of the feature (Figure 7.5(a)). However, with the FSRF, we cannot use this constraint, because the baseline of the FSRF is wide. If we know a rough estimation of the epipolar geometry, we can restrict the search area to the epipolar line of the feature (Figure 7.5(b)). Since we have already computed the camera parameters, we can apply this constraint. After computing a dense range image, the position of the corresponding feature is computed by the 3D position of the feature. We can restrict the search area to the predicted position of the feature (Figure 7.5(c)).

Bundle Adjustment

After taking the correspondence of features, we used bundle adjustment [7, 93] to refine the camera parameters. This method minimizes the error of camera parameters of multiple cameras globally by the maximum likelihood estimation. First, we computed the 3D position of feature points by triangulation. Since corresponding rays do not completely intersect in the 3D point due to the errors of camera parameters, we found the optimal point by estimating the error using a polynomial of degree 6 [38] and a midpoint of the corresponding rays [3].

If we have m images and n feature points, we minimize the following criterion:

$$\min_{P_i, M_j} \sum_{i=1}^{m} \sum_{j=1}^{n} D(m_{ij}, P_i M_j), \tag{7.1}$$

where P_i is the projection matrix from the world coordinate to the image coordinate of the

i-th image, M_j is a 3D point which is computed by triangulation of the j-th corresponding feature points, m_{ij} is a 2D point of the j-th feature point in the i-th image, and $D(m, m')$ means the Euclidean image distance. The minimization is accomplished by Levenberg-Marquardt minimization [71]. Since the number of feature points can be larger than a thousand, a straightforward computation is difficult. Thus, a special technique [93, 70] was developed by utilizing the sparse structure of Jacobian matrix which is used in Levenberg-Marquardt minimization.

In many cases, the correspondences of the feature points contain the wrong matches of features. Since the minimization of the bundle adjustment is a least-square minimization, the estimation P_i and M_j is affected, and fails by those outliers. Robust estimation methods such as RANSAC (RANdom SAmpling Consensus) [28] and LMedS (Least Median Square) are used to detect wrong matches for computing the fundamental matrix of the epipolar geometry [92, 101]. In our implementation, we introduced the RANSAC approach to bundle adjustment to detect and remove wrong matches.

Algorithm 7.1 shows bundle adjustment with RANSAC. First, it generates sample sets F_{set} from the set of corresponding feature points F_{all}. The number of the sample sets m is determined by

$$\Gamma = 1 - (1 - (1 - \epsilon)^p)^m, \tag{7.2}$$

where Γ is a probability that a good subsample is selected, ϵ is the rate of contaminated data and p is the number of feature points in each sample (Refer [75] for detail). We compute a new camera parameters C_{new} for each sample and estimate C_{new} by counting the feature points which are near the epipolar line with C_{new}. After testing all samples, we select the best solution of C_{new}, which has the maximum number of inliers. We refine C_{new} using all feature points which are consistent with C_{new}. Finally, we refine C_{new} one more time.

7.3 Range Sensing by Stereo Matching

We generated a dense range image by taking a matching of the window areas of multiple images based on the multi-baseline stereo algorithm [69]. Since we computed the correlation of 8 stereo pairs using 9 images and each image was high resolution, we accelerated the computation by the algorithm of recursive correlation calculation [27] and graphics

Algorithm 7.1 BundleAdjustmentWithRANSAC(C, F_{all})

Input: Camera Parameters: C

Input: Corresponding Feature Points: F_{all}

Output: New Camera Parameters: C_{new}

Generate random sample set F_{set} of corresponding feature points from F_{all}

for all $F \in F_{\text{set}}$ **do**

$C_{\text{new}} \leftarrow$ **BundleAdjustment**(C, F)

Estimate solution by counting inliers consistent with C_{new}

end for

$F \leftarrow$ feature points consistent with best solution of C_{new}

$C_{\text{new}} \leftarrow$ **BundleAdjustment**(C, F)

$F \leftarrow$ feature points consistent with C_{new}

$C_{\text{new}} \leftarrow$ **BundleAdjustment**(C, F)

hardware accelerators to rectify the image pairs. Though the image rectification by special hardware [14] exists, we use common graphics hardware.

7.3.1 Hardware Acceleration of Rectification

The 3D point M of the corresponding feature points are computed as follows:

$$M = P_1 m_1 = P_2 m_2, \tag{7.3}$$

where P_1 and P_2 are the projection matrices from the world coordinate to the image coordinate, and m_1 and m_2 are the image coordinates of the feature points. Therefore, the position of a corresponding point is computed by

$$m_2 = P_2^{-1} P_1 m_1. \tag{7.4}$$

If an image has $N \times N$ pixels and we examine D disparities for each pixel, the computational cost of (7.4) for every pixel of an image is $O(N^2 D)$. This forms a major part of the computational cost of stereo matching. Since the computation is the transformation of coordinates, it can be accelerated by using graphics hardware. When we compute the correlation of a disparity, we transform the reference image as a rectangle with texture. The transformation by (7.4) is computed for four corners of the image explicitly; and the

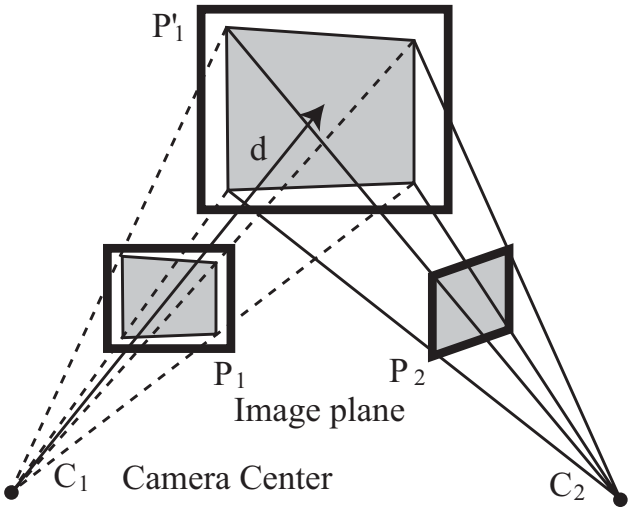


Figure 7.6: Rectifying image pair by reprojection

transformation of the internal areas is interpolated by using texture mapping of the reference image. Since those operations are supported by recent graphics hardware, we can compute them efficiently.

Figure 7.6 shows rectification of the points at depth d. The plane P_1' is parallel to the image plane P_1 and the distance from the camera center C_1 to P_1' is d. First, we compute the positions of four corners on P_1' which are projection of the four corners of the image plane P_2. Second, we render the quadrangle with mapping the image of P_2 as texture. The corresponding point is reprojected to the same position. We iterate this reprojection for each depth d to search different disparities.

7.3.2 Recursive Computation of Correlation

We used the sum of the sum of absolute distance (SSAD) as the criterion for stereo matching, which is represented by

$$N(m) = \sum_{i}^{W \times W} \sum_{k}^{n} |I_0(m_i) - I_k(m_i')|, \tag{7.5}$$

where m is a point of the reference image, m_i is a point of the reference image in a window of size W and m_i' is the corresponding point of m_i. Namely $m_i' = P_k^{-1} P_0 m_i$, where

P_0 and P_k are the projection matrices of the reference image and k-th image. The computational cost is $O(N^2W^2D)$. It can be reduced to $O(N^2D)$ by recursive computation of correlation [27]. The pointwise correlation of each pixel for depth d is computed by

$$P(x, y, d) = \sum_{k}^{n} |I_0(x, y) - I_k(P_k^{-1} P_0(x, y, d)^T)|. \tag{7.6}$$

Then, (7.5) is rewritten as

$$N(x, y, d) = \sum_{i=0}^{W-1} \sum_{j=0}^{W-1} P(x + i, y + j, d). \tag{7.7}$$

Now, we define $Q(x, y, d)$ as

$$Q(x, y, d) = \sum_{j=0}^{W-1} P(x, y + j, d). \tag{7.8}$$

$N(x, y, d)$ can be computed recursively by

$$N(x + 1, y, d) = N(x, y, d) + Q(x + W, y, d) - Q(x, y, d). \tag{7.9}$$

as shown in Figure 7.7. $Q(x, y, d)$ is also computed recursively.

7.3.3 Matching by Recursive Correlation

The computation of disparity by (7.5) is represented by

$$\begin{aligned} O(x, y) &= \min_d \{N(x, y, d)\} \\ N(x, y, d) &= \textstyle\sum_i^W \sum_k^n |I_0(x, y) - I_k(P_k^{-1} P_0(x, y, d)^T))| \end{aligned} \tag{7.10}$$

The computation can be reduced by recursive computation of $N(x, y, d)$:

$$\begin{aligned} P(x, y, d) &= \textstyle\sum_k^n |I_0(x, y) - I_k(P_k^{-1} P_0(x, y, d)^T)| \\ Q(x, 0, d) &= \textstyle\sum_j P(x, j, d) \\ Q(x, y + 1, d) &= Q(x, y, d) + P(x, y + W, d) - P(x, y, d) \\ N(0, y, d) &= \textstyle\sum_i Q(i, y, d) \\ N(x + 1, y, d) &= N(x, y, d) + Q(x + W, y, d) - Q(x, y, d) \end{aligned} \tag{7.11}$$

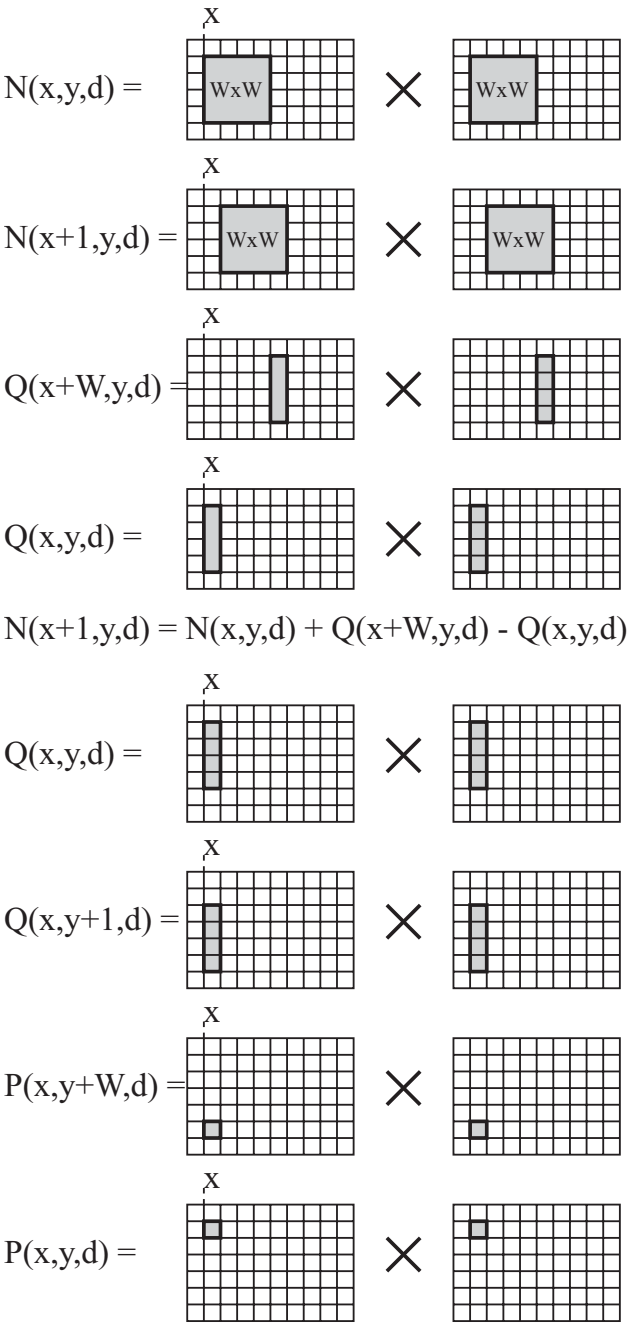
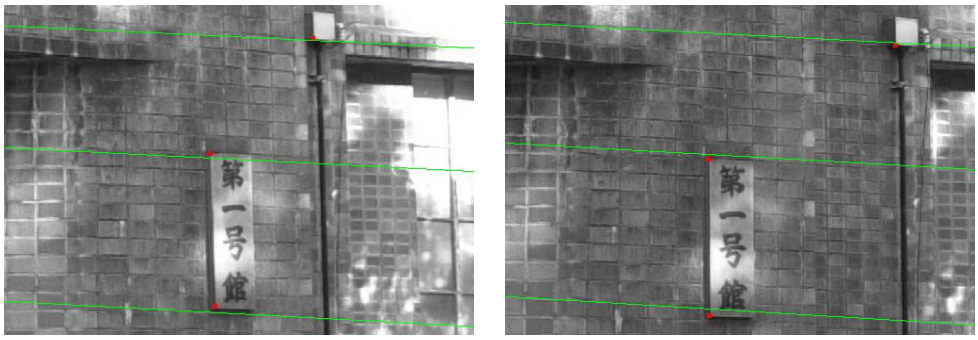
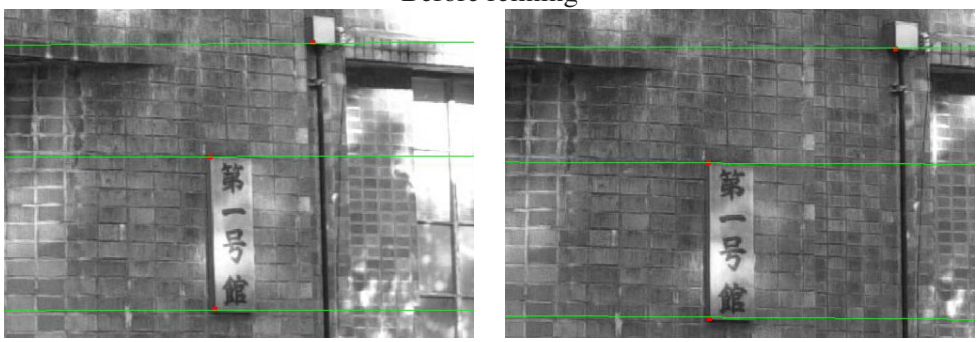


Figure 7.7: Recursive correlation calculation



Before refining



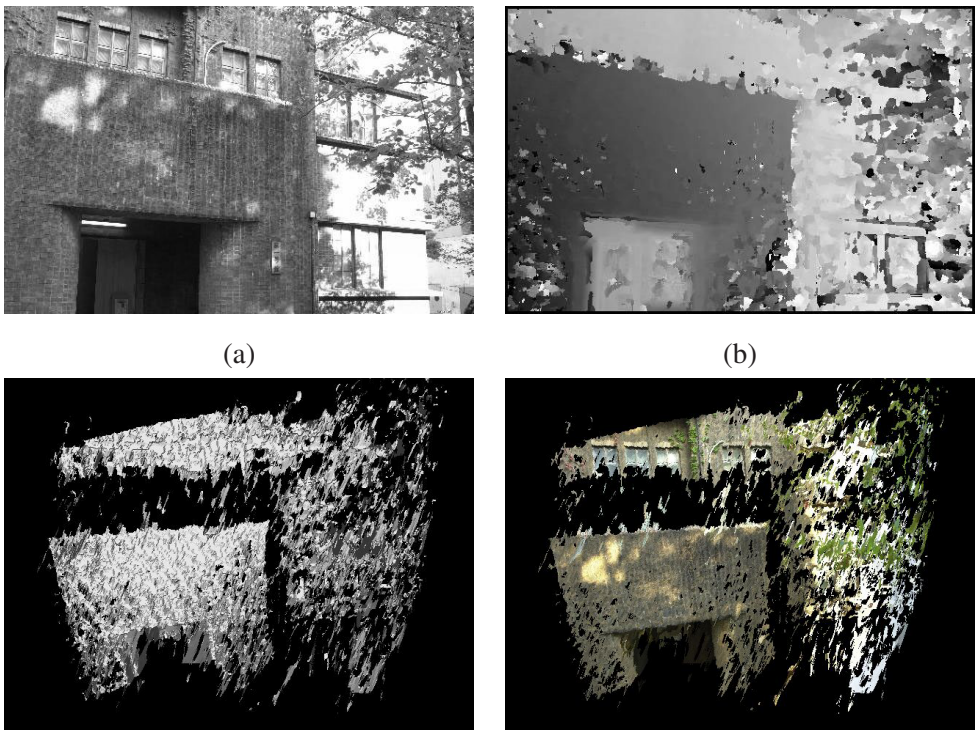
After refining

Figure 7.8: Refining camera parameters by bundle adjustment

7.4 Experiments

First, we calibrate camera parameters using a marker shown in Figure 7.4. Next, we refine the camera parameters by bundle adjustment. Figure 7.8 shows the result of refining camera parameters. 2 images of each row consist a pair of camera images. Red points are extracted feature points and green lines means epipolar lines. Since we know rough epipolar geometry, these feature points can be matched automatically. Before applying bundle adjustment, feature points are not accurately on the line of corresponding epipolar lines. After refining camera parameters by minimizing the distances between feature points, feature points becomes on the line of corresponding epipolar lines.

Figure 7.9 shows generating a range image by stereo matching. Figure 7.9(a) is one of 9 camera images of FSRF. Figure 7.9(b) is the result of stereo matching. Dark pixels are near the camera and light pixels are far. Figure 7.9(c) is the 3D model constructed from the range image and Figure 7.9(d) is the model with mapping the camera image as texture.



(a) (b)

(c) (d)

Figure 7.9: Generating a range image by stereo matching

7.5 Summary

We developed a new range scanning system to cover unobservable surfaces from the ground. This system obtains a dense range image by stereo matching. It is loaded on a balloon and scans a large-scale object from the air. Since it moves by wind, the system acquires multiple images using 9 cameras at the same time, to avoid the difficulty of calibration. To construct a model with high resolution, we use digital steel cameras, which can obtain high resolution images. Also, since there are difficulties when in calibrating a camera when we scan a large-scale object, we developed a calibration technique using range images. For efficient computation of stereo matching, our algorithm uses rectification accelerated by graphics hardware and recursive correlation calculation.

Chapter 8

Complement of Unobservable Surface

Even after scanning an object using various kinds of sensors, unobserved surfaces may exist in many cases. In this chapter, we propose an approach to fill those holes. This approach complements the geometric and photometric model of the object by estimating the neighborhood area of the holes. In our merging framework, we have already computed the signed distance field (SDF). Since we separate the entire volume into two manifolds by SDF, we can generate a closed surface by converting SDF to a mesh model. However, the sign of SDF becomes sensitive around the holes. Thus, we cannot interpolate the holes by a suitable surface by applying the merging method straightforwardly. In this chapter, we thus propose a novel method to interpolate the holes of range images by taking a consensus of the sign of SDF with those of the neighbor voxels.

8.1 Convert from Range Images to SDF

We have already described the method "consensus surface algorithm" [97, 77] to convert multiple range images to SDF in Chapter 2. In this section, we explain the algorithm in brief again. Figure 8.1 shows that the consensus surface algorithm is applied to a situation in which there are three range images which are intersecting between two neighboring voxels. First, it finds the nearest point of each range image from the the center of the voxel $\boldsymbol{x}$. Next, it searches the nearest points of the other range images from them. If the found points are similar in terms of their positions and normal vectors, we say that the points have consensus and we regard them as the same surface. By averaging the positions and

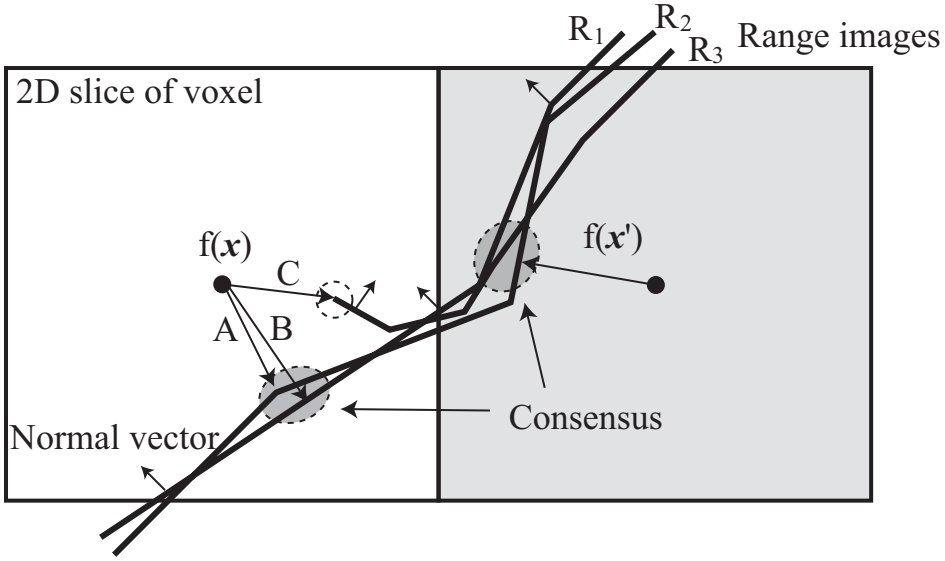


Figure 8.1: Compute signed distance by considering consensus of range images.

normal vectors of the same surfaces, we compute three signed distances $f(\boldsymbol{x})$ from $\boldsymbol{x}$. In Figure 8.1, the consensus surfaces are A and B; the surface C does not have consensus. Thus, the final signed distance from $\boldsymbol{x}$ is chosen from A or B. Since the magnitude of A is smaller than that of B, A is chosen in this case. We can discard the isolated observation C. Finally, since $f(\boldsymbol{x})$ is positive and $f(\boldsymbol{x}')$ is negative, the isosurface $f(\boldsymbol{x}) = 0$ is successfully generated to intersect between $\boldsymbol{x}$ and $\boldsymbol{x}'$ by the marching cubes algorithm (MC) [53]. Since the computation of the signed distance is necessary near the surface, our algorithm computes signed distances by subdividing the volume of interest recursively in an octree manner to reduce the cost of computation.

The consensus surface algorithm robustly computes the signed distance by discarding the outliers of range images. Since MC generates a closed surface by converting SDF to a mesh model, we can interpolate the holes of the surface which any range image does not cover if we can compute signed distances of voxels around holes. However, the computation of SDF becomes sensitive to noise around the holes; Thus, it is actually difficult to fill the holes by applying the merging method directly. We analyze the reasons why the computation of SDF becomes sensitive to noise in the next section.

8.2 Sensitiveness of Computing Sign of SDF

In this section, we point out the problem of the merging algorithm. It often make a mistake in computing the sign of a signed distance. This especially happens where the surface of the object is highly curved. Figure 8.2 shows an example of computing signed distances when there is high curvature surface of an object. The gray line means the real surface of the object. The black solid lines are the range images. In this situation, we regard every point of the range images as having consensus. Each arrow is the vector from the center of a voxel to the nearest point. If a voxel is inside the object by considering normal vectors of range images, the sign is negative, and the voxel is filled by the color gray in Figure 8.2. In a high curvature area, it is difficult to cover both sides of a corner by a single range image. Thus, we need several range images to cover the entire corner; and to align them into a common coordinate system. Though the lower left area is obviously outside in Figure 8.2, the signs of those voxels are negative because they are considered inside by the normal vector of the nearest point. Consequently, the surface generated by MC corrupts, as represented by dotted lines. The signs of the signed distances become sensitive to noise around a sharp corner, which are caused by the following four reasons:

1. Their relative positions after aligning contains errors caused by the errors of measuring.

2. It is difficult for range images to overlap each other at the point of sharp corners.

3. The sampling interval of the points of a range image can be too coarse to represent a sharp corner.

4. There are holes of the surface which any range image does not cover.

First, since the signs of SDF can change sensitively due to a small error of the position of a range image, the shape of the generated mesh model is greatly affected by change of signs of SDF. Second, because a single range image cannot cover both sides of a sharp corner, the sign of a voxel changes according to range images which are used to compute SDF. Thus, it is difficult to determine that the voxel is inside or outside the surface. Third, if the sampling interval of a range image is coarse, the shape of the corner can be smoothed out. Fourth, it is often difficult to acquire range images to cover a sharp corner, and holes in

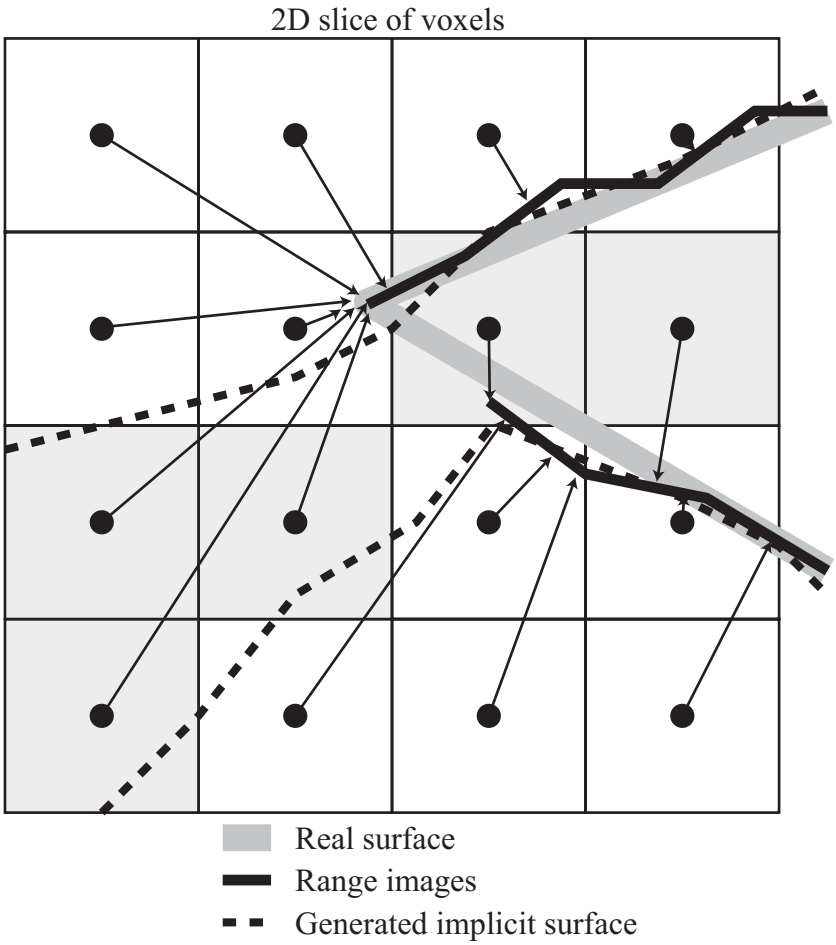


Figure 8.2: Corruption of the surface caused by wrong sign of SDF

range images can be found in many cases. If a hole exists, the sign of SDF can be sensitive around the hole.

Figure 8.3(a) shows the range images of an object which has sharp corners. Figure 8.3(d) is a zoom-up of one of the sharp corners. Due to the above reasons, it is hard to determine whether a voxel is inside or outside near the corner, even if we take a consensus of the signed distances using the position and normal vector of points of the range images. Consequently, the result of converting SDF to a mesh model by MC is shown in Figure 8.3(b), and its zoom-up is Figure 8.3(e). Since the signs of SDF of some voxels are wrong, many vertices and triangles are generated outside the object.

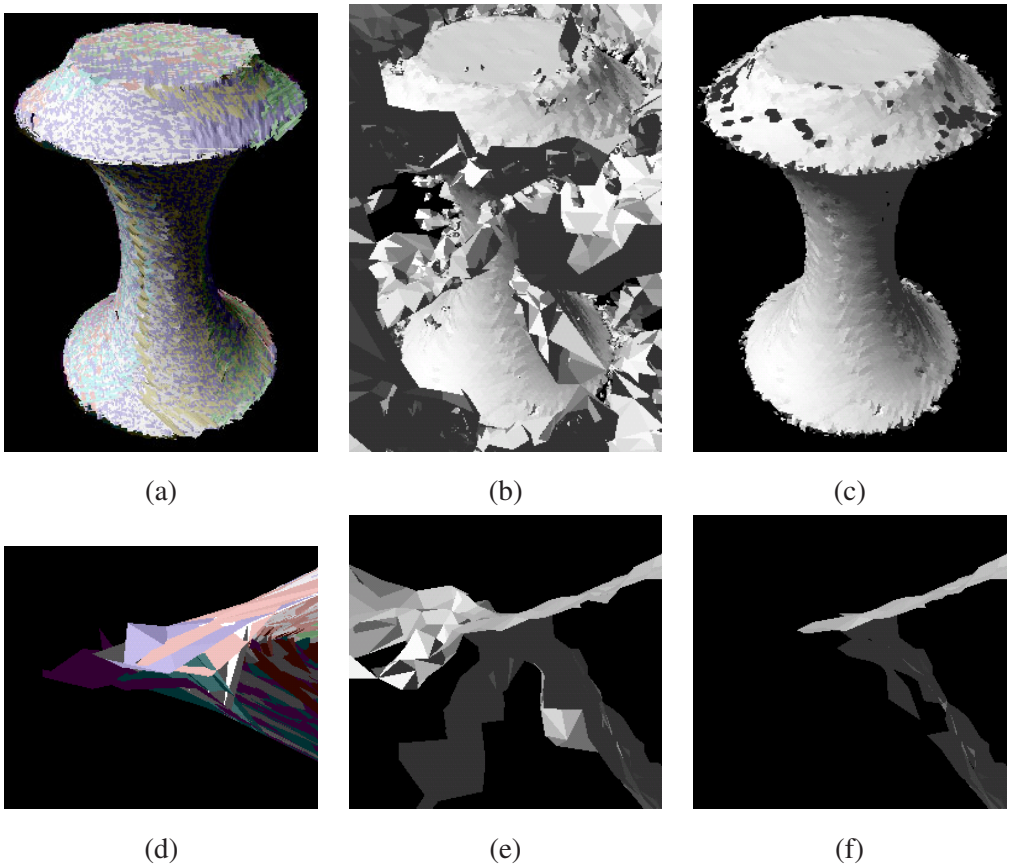
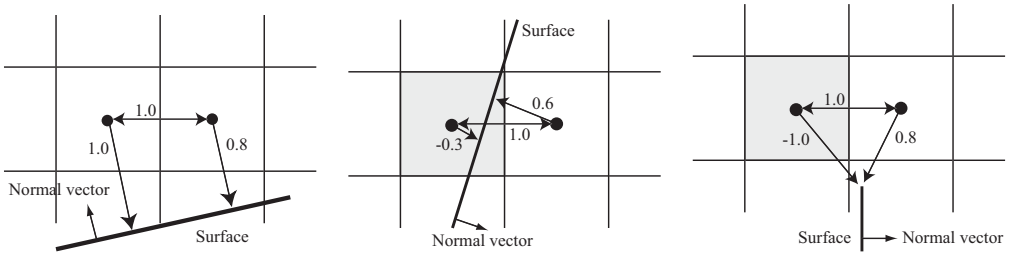


Figure 8.3: Merging result of a sharp corner

If the magnitude of a signed distance is smaller than the width of the voxel, the nearest point is inside the voxel. Thus, the generated vertices and triangles are not far from the real surface. Therefore, if we restrict the magnitude of SDF to apply the SDF to MC comparing the magnitude with the width of the voxels, erroneous meshes are not generated. Namely, if the signed distance of a voxel is d and that of its neighbor voxel is d', we can remove erroneous meshes by applying the voxel to MC, only if the signed distances satisfy

$$|d - d'| < W \tag{8.1}$$

for all voxels which consist of a cube, where W is the width of the voxels. After restricting by (8.1), the merging result becomes Figure 8.3(c) and (f). Erroneous meshes are removed; however, holes in the surface remain where the magnitude of the signed distances was



(a) (b) (c)

Figure 8.4: Three examples of signed distances of adjacent voxels

larger than the width of voxels. To interpolate holes of the surface, a method is needed to determine the signs of SDF which are resistant to errors caused by the above reasons.

8.3 Consistency of the SDF

Our merging algorithm takes a consensus of range images for outlier detection; however, it does not consider a consensus of the signed distances of adjacent voxels. In this section, we introduce a new criterion to estimate whether the signed distances of adjacent voxels are consistent with each other or not.

Figure 8.4 shows three examples of the signed distances of adjacent voxels. The width of the voxels was 1.0. In Figure 8.4(a), both voxels are outside an object, whose signed distances are 1.0 and 0.8. If the sign of the adjacent voxels are the same, the isosurface does not exist between them. Since the arrows from the center of the voxel to the surface represents the nearest neighbor points and the normal vector looks outside, the situation that both signed distances are same is possible, as shown in Figure 8.4(a). Thus, they are considered consistent with each other.

In Figure 8.4(b), the signs of the adjacent voxels are different. The signed distances are -0.3 and 0.6. The gray voxel means that it has a negative signed distance. An isosurface exists between two voxels. Since it is possible to compute the situation from a surface as shown in Figure 8.4(b), the signed distances of the two voxels are considered consistent.

In Figure 8.4(c), the sign of the adjacent voxels are different, similarly with Figure 8.4(b). The difference from Figure 8.4(b) is that the signed distances are -1.0 and 0.8. By considering the SDF of these two voxels, an isosurface exists between them; however, if the

original surface exists between them, the sum of the magnitude of the two signed distances is smaller than the width of the voxel. Thus, the original surface does not exist between the two voxels, as shown in Figure 8.4(c), and the situation is considered inconsistent to generate meshes between them. The inconsistent situation such as Figure 8.4(c) occurs by the reasons described in 8.2. If the signed distances of two adjacent voxels satisfy the following inequality, we define the situation as inconsistent:

$$|d - d'| > W, \tag{8.2}$$

where d and d' are the signed distances of two adjacent voxels. Consequently, if there is an inconsistency of SDF, the sign of the signed distance is doubtful, and we examine the situation to generate a mesh model which is consistent with the original surface.

8.4 Flip Sign of SDF

We propose a novel method to examine whether the signed distance of a voxel is appropriate or not. Since we find the nearest neighbor point of a range image using a k-d tree [30], the magnitude of each signed distance is computed by Euclidean distance. Thus, its magnitude is appropriate as an element of SDF to generate an isosurface. The factor we have to examine is the sign of each signed distance to generate a mesh model which is consistent with the original surface.

The merging algorithm described in Chapter 2 takes a consensus of range images by searching the nearest neighbor points. However, it does not consider the consistency of the signed distances of adjacent voxels. Thus, our new method takes a consensus of the signed distances of adjacent voxels. We have already introduced a criterion by (8.2). We compared a voxel with its adjacent voxels. Since our merging method uses voxels of adaptive resolution, as described in Chapter 4, (8.2) is modified to

$$|d - d'| > \alpha \frac{W + W'}{2}, \tag{8.3}$$

where W is the width of the current voxel, W' is the width of the adjacent voxel and α is a parameter defined by the user.

Figure 8.5 shows a 2D slice of adjacent voxels to take the consistency of the signs of SDF. The number of adjacent voxels in a 3D space is 26 if the resolution of the voxels is

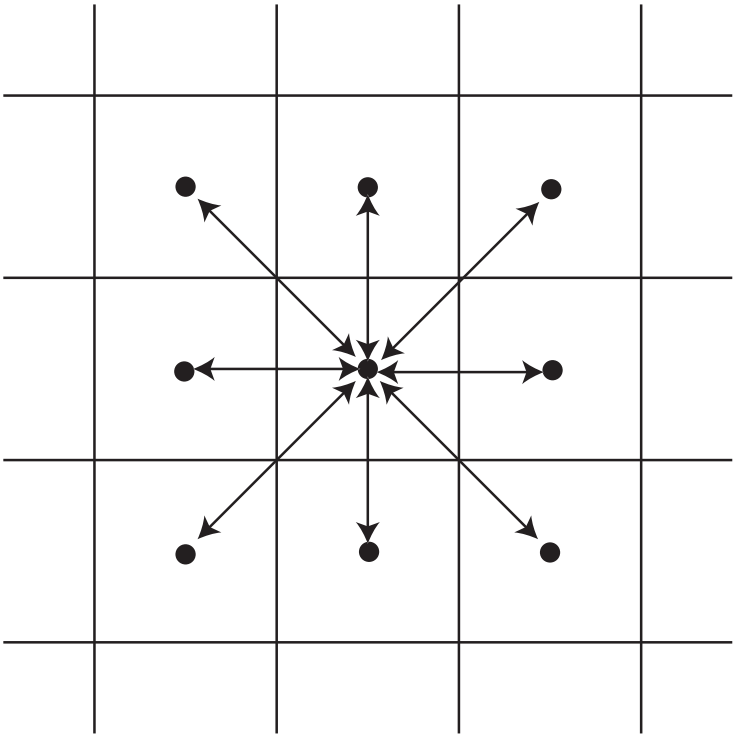


Figure 8.5: Adjacent voxels to take consistency of the signs of SDF

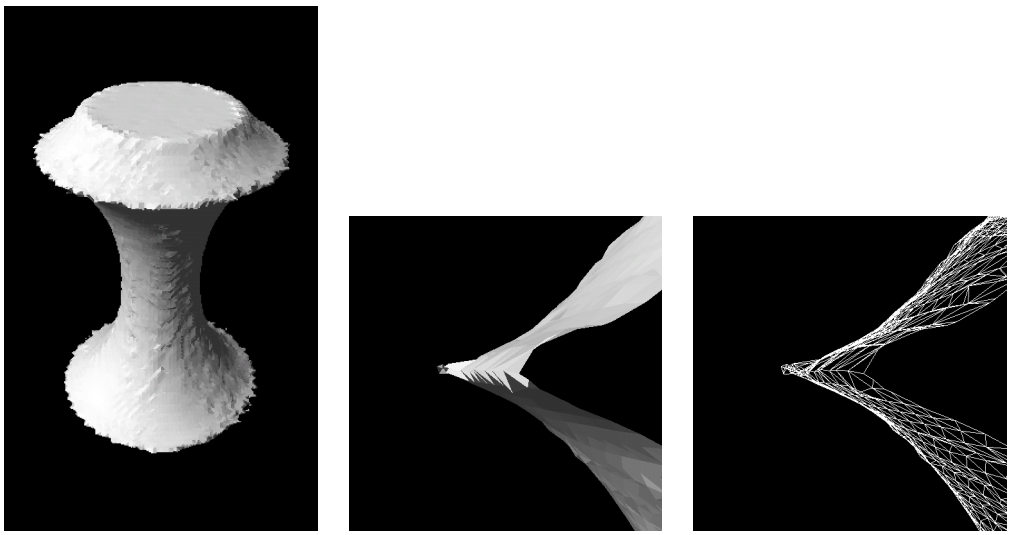
constant. We compared a voxel with all adjacent voxels. If the number of voxels which satisfy (8.3) is larger than $n/2$, where n is the number of adjacent voxels, we consider the sign of the signed distance of the voxel to be inconsistent. Then, we flip its sign to make it consistent with the adjacent voxels. We compute the consistency as for all voxels, and flip its sign if it is inconsistent with its adjacent voxels. We iterate the process until all voxels are consistent.

The algorithm used to take the consistency of adjacent voxels becomes Algorithm 8.1. d, d' and d_o are the signed distance of the voxels N, N' and N_o, respectively. If a voxel is determined as an inconsistent voxel by computing the consistency with all adjacent voxels, we flip the sign of the signed distance and add the voxel N_o to output octree O. If it is consistent, we add the new voxel N_o without changing the signed distance. We tested all voxels by traversing the current octree. We iterated **FlipSign**(N) until the number of voxel whose signed distance was flipped became 0 (see Algorithm 8.2). Since the signs of the

flipping voxels may oscillate and do not converge to 0, we increased α and relaxed the condition during iteration if the convergence became slow.
After one iteration, we can restrict the voxels to examine their signs. If the signs of a voxel and its adjacent voxels did not flip in the previous iteration, we did not have to examine it in the current iteration. Thus, we recorded the voxels whose signs were flipped, and used the database to determine whether we needed to examine a voxel or not.

8.5 Experiments

We first tested our algorithm to take the consistency of signed distances with adjacent voxels using a small object. In Figure 8.3, the signed distances around sharp corners are sensitive to noises. We applied our new method to the SDF of the object, and obtained the model shown in Figure 8.6. Since we took the consistency of the signed distances of the adjacent voxels, we removed erroneous vertices and triangles from the generated mesh model at the same time as we filled holes on the original surface. Figure 8.7 shows the rendering results of SDF directly by volume rendering [29]. The red spaces mean voxels which have positive signed distances; and the blue spaces are voxels which have negative ones. The green faces are the isosurface of zero level of the SDF. In Figure 8.7(a), there are blue spaces outside the object. However, after taking the consistency of SDF, our method removes them, as shown in Figure 8.7(b), and successfully extracts the isosurface of zero level.
We iterated the traversal of an octree 32 times; the number of flipped voxels changed, as shown in Figure 8.8. In this experiment, α was fixed to 1. The number of flipped voxels was reduced quickly at the beginning. Since the total number of voxels in the octree was 130521, the number of flipped voxels were much smaller than the total number of voxels. Thus, the reduction of computation by using the database of flipped voxels worked effectively.
Next, we complemented the unobserved surface of the Great Buddha of Kamakura. After merging range images of the Buddha, we obtained the model shown in Figure 8.9(a). Since the mesh model was generated with restriction by (8.1), the model has holes around the unobserved areas, such as the top of the head, the shoulders, and the arms. By taking the consistency of SDF, we complemented those unobserved surfaces, as shown in Figure



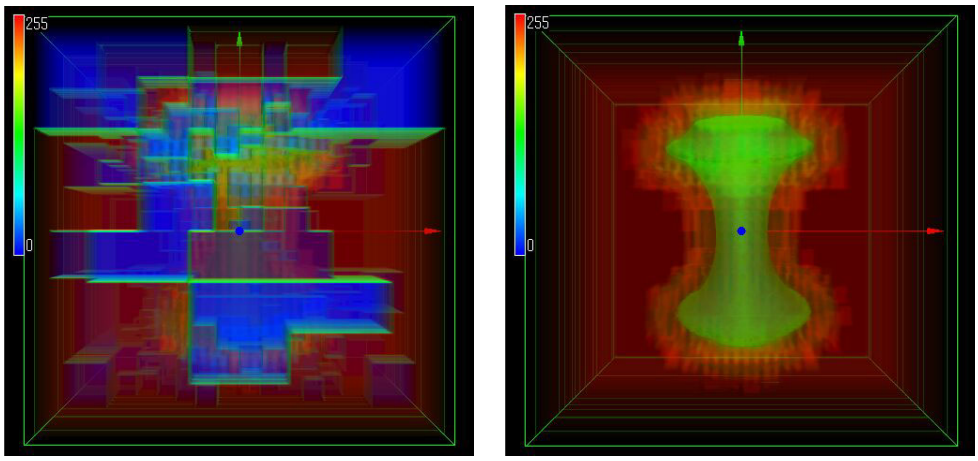
(a) (b) (c)

Figure 8.6: Result of taking consistency of the SDF

8.9(b). We were able to successfully generate a plausible model of the unobserved surfaces. Figure 8.10 shows the SDF by volume rendering. There are many voxels whose signs are doubtful, which was caused by sensitive computation of the signed distance in Figure 8.10(a). After taking the consistency of SDF, we were able to remove those doubtful voxels and extract a robust isosurface.

8.6 Summary

We have proposed a novel method to complement such holes of surfaces. Since the signed distance field, which is computed during the merging process, has inconsistencies for the signs of neighbor voxels because it becomes sensitive around the holes of range images, we efficiently complemented the unobserved surfaces by taking a consensus of their signs.



(a) (b)

Figure 8.7: SDF by volume renderer

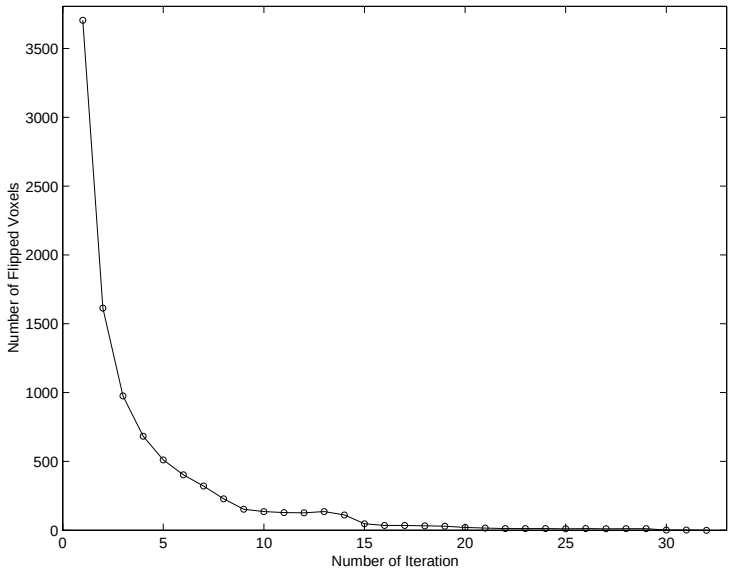


Figure 8.8: Number of voxels whose signs are flipped

Algorithm 8.1 FlipSign(N)

Input: Current Node of Octree: N

Local: Node of Octree: N', N_o

Local: Signed Distance: d, d', d_o

Local: Width of N, N': W, W'

Local: Number of Voxels: n, n_{i}

Output: Output Octree: O

```
N_o ← N
if need to examine N then
    n_i ← 0, n ← the number of adjacent voxels
    for all adjacent voxels N' of N do
        if |d − d'| > α (W+W')/2 then
            n_i ← n_i + 1
        end if
    end for
    if n_i > n/2 then
        d_o ← −d
    else
        d_o ← d
    end if
end if
add N_o to output octree O
if N is nonterminal then
    for all children N_i (i = 0, ..., 7) of N do
        FlipSign(N_i)
    end for
end if
```

Algorithm 8.2 IterateFlipSign(O)

Input: Current Octree: O

Local: New Octree: O'

Local: Root Voxel of Octree: N_{root}

Local: Number of Flipped Voxels: n, n_{prev}

$n \leftarrow \infty$

while $n > 0$ **do**

$N_{\text{root}} \leftarrow$ root node of octree O

$O' \leftarrow 0$

$O' \leftarrow$ **FlipSign**(N_{root})

$n_{\text{prev}} \leftarrow n$

$n \leftarrow$ number of voxels whose signed distance is flipped

if $n_{\text{prev}} - n < \epsilon$ **then**

increase α

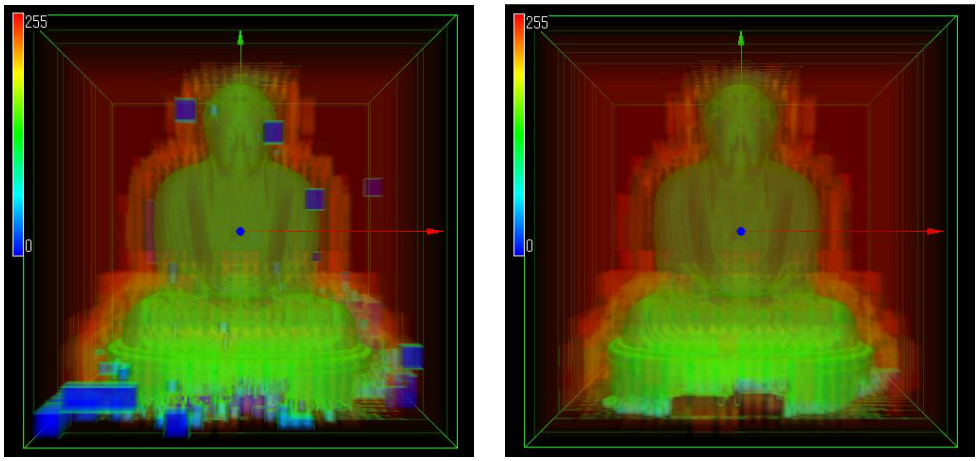
end if

$O \leftarrow O'$

end while

(a) (b)

Figure 8.9: Complementing the unobserved surfaces of the Buddha



(a) (b)

Figure 8.10: SDF of Kamakura Buddha rendered by volume renderer

Chapter 9

Conclusion

In this thesis, we have considered the geometric and photometric modeling of large-scale and intricately shaped objects, such as cultural heritage objects. In modeling such objects, the following new issues occurred during the modeling steps, which have not been considered in previous research on the modeling of small, indoor objects:

1. Creating a detailed model from a huge amount of data
2. Merging of photometric attributes of range images
3. Constructing a precise model using noisy data
4. Covering of the whole surface of an object
5. Complementing unobservable surfaces of an object

Concerning the first issue, we have proposed three approaches, the parallel processing of merging range images, an effective nearest neighbor search, and an adaptive algorithm of merging range images. In addition, we proposed a new technique to merge photometric attributes attached with range images in our merging framework. If the obtained range images are noisy, we have proposed a novel method to refine the range images using multiple observations. To obtain range images and color images of the areas of the object which cannot be observed from the ground, we have developed a new stereo range finder which is loaded on a balloon, and which acquires images from the air. Since unobservable surfaces still exist after making full use of various kinds of sensors, we have proposed a new method to complement the holes of surfaces which cannot be observed by any scans.

9.1 Parallel Merging of Range Images

A huge amount of data is needed to create a detailed model of a large-scale object such as a cultural heritage object. Thus, we have proposed a new method to merge a huge amount of range images by parallel computation of signed distances using a PC cluster, which is extended from the consensus surface algorithm [97]. It consists of the following two components:

1. Distributed allocation of range images to multiple PCs
2. Parallel traversal of subtrees of an octree

We handled a huge amount of range images, which were larger than the size of the physical memory of a single PC, by the first technique. The second technique made full use of many CPUs belonging to a PC cluster, and reduced the computational time.

9.2 Effective Nearest Neighbor Search

As the second approach to handling a huge amount of data, we have proposed a new algorithm for searching for the nearest neighbor using a k-d tree. If the nearest neighbor point is far from a query, it is not as important as the nearest neighbor in many applications. Thus, we have proposed the Bounds-Overlap-Threshold test, which does not search strictly by pruning branches if the nearest neighbor point is beyond a threshold. This technique drastically reduces the computational cost if the nearest neighbor is far from a query. Since the threshold of the BOT test can be changed without re-creating a k-d tree, it is suitable for applications in which variable threshold is considered. We discussed the performance, which depends on the distribution of the distance from a query to the nearest neighbor.

9.3 Adaptive Merging Algorithm

The third approach to handling a huge amount of data is the adaptive algorithm of merging range images. We have proposed an algorithm for constructing a 3D model in an efficient representation. Considering the surface curvature and the photometric attributes, we constructed 3D models that have a higher detail in surface areas that contain either high

curvature or a significant variation of appearance. We have proposed two approaches: considering the curvature of the implicit surface of SDF after the merging process, and an adaptive merging technique based on the curvature of range images. We can efficiently use the computational resources by these methods.

9.4 Merging Photometric Attributes of Range Images

Another extension of the merging framework is the merging of photometric attributes which are attached with range images. By taking a consensus of the appearance changes of the target object from multiple range images, we reconstructed a 3D model with an appearance which successfully discards outliers due to noise. Also, we provided a model with Lambertian reflected light values by discarding specular reflection as outliers. The photometric attributes of the model can be used for aligning with 2D color images.

9.5 Refining Range Images using Multiple Observations

We have proposed an efficient method of refining range images under the consideration of unique error distributions of multiple range images. We described how we applied this method for the modeling of the artificial test object and the actual cultural heritage object from the images acquired by the time-of-flight range sensor. We also applied our method to the range images which were generated by a multi-baseline stereo system. The experimental result shows the validity of this method compared with that of the existing filter-based methods.

9.6 Flying Stereo Range Finder

To cover unobservable surfaces from the ground, we developed a new range scanning system, which obtains a dense range image by stereo matching. It is loaded on a balloon, and scans a large-scale object from the air. Since it moves by wind, the system acquires multiple images using 9 cameras at the same time to avoid difficulty of calibration. To construct a high resolution model, we used digital steel cameras, which can obtain high resolution images. Also, since it is difficult to calibrate cameras when scanning a large-scale object, we developed a calibration technique that uses range images. For efficient

computation of stereo matching, our algorithm uses rectification accelerated by graphics hardware and recursive correlation calculation.

9.7 Complementing Unobservable Surfaces

After scanning the surface of an object using various kinds of sensors, unobserved surfaces often exist. We have proposed a novel method to complement such holes of surfaces. Since the signed distance field, which is computed during the merging process, has inconsistencies with regards to the signs of neighbor voxels because it becomes sensitive around the holes in the range images, we efficiently complemented the unobserved surfaces by taking a consensus of their signs.

9.8 Contributions

We would like to list the contributions of this thesis:

- Parallel processing algorithm of computation of signed distances: we have analyzed the algorithm of computation of signed distances and developed a parallel computation algorithm.
- A novel algorithm for searching for the nearest neighbor using a k-d tree: we have proposed an efficient search algorithm to reduce the computational cost by pruning branches if the nearest neighbor point is beyond a threshold.
- Adaptive merging algorithm using octree: we constructed a model in adaptive resolution which depends on curvature and photometric variation. Consequently, we can efficiently use the computational resources available.
- Merging photometric attributes of range images: we have proposed a new technique to integrate photometric attributes of range images in our merging framework. We can utilize them for aligning color images.
- Refining range images using multiple observations from various viewpoints: we refined range images during iterative computation by assuming that the distribution of their error is anisotropic according to the viewing direction.

- Flying Stereo Range Finder: this sensor obtains a dense range image by stereo matching while it is loaded on a balloon and moving in the air.
- Complementing unobserved surfaces which are not observed from any scans: we have proposed a new technique which fills holes of the surface by taking a consensus of the signed distances of neighbor voxels.

9.9 Future Work

We have proposed a novel method of merging range images in adaptive resolution. The main goal of the adaptive merging method is to reduce the computational cost and the size of generated model. Though the method works well, the quality of the result as a mesh model is lower than a model generated by a mesh simplification technique because our method generates irregular meshes based on marching cubes. We will consider the quality of the resulting mesh model as several research [100, 23] do.

We have developed a method of refining range images. So far, we have tested the method using several kinds of range finders, for example, laser range finders and a stereo range finder, whose ray directions are perspective. We plan to test range images which are obtained by other types of range finders, such as by an airborne laser scanner [48]. Since the range image obtained by an airborne laser scanning is obtained from an airplane in flight, it does not give perspective but rather a pushbroom panorama [83]. We will experiment with our refining technique by applying it to range images obtained by other types of range finders.

In our current implementation, we were able to refine range images after aligning range images. The refining method can be considered as a problem of estimating the parameters which are attached to each vertex of range images. Since the aligning of range images is a problem of estimating the parameters of rotation and translation, we will consider the relationship of refining and aligning range images.

We have developed a stereo range finder which consists of 9 cameras. To generate a more accurate range image, we plan a new sensor which has more than 9 cameras or which utilizes structure-from-motion.

Though we have developed the FSRF for filling holes of the surface of objects which

cannot be observed from the ground, unobservable surfaces still exist, such as concave surfaces in a narrow space. We plan to develop a new sensor to observe those types of unobservable surfaces.

Appendix A

Determine Adjacent Voxels of Octree

A.1 Determine Adjacent Voxels

To apply the marching cubes algorithm (MC) to an adaptive octree, we have to determine 8 voxels which consists 8 vertices of a cube. Since these 8 voxels are adjacent to each other, we must first determine the adjacent voxels of a voxel. To identify a node in the entire volume, we define the ID L of a node N of the octree, which corresponds to the voxel. L is a list of codes which represent the relative position compared with the parent node. Namely, if N is at depth d of the octree, L of the node N represented a list of code:

$$L = \{L_1, L_2, \ldots, L_d\}, \tag{A.1}$$

where L_i is the code at depth i. Figure A.1 shows the coding of child nodes relative to the parent node. If a node at depth i is larger side of the axis as to x axis among brothers of the node, we set $\delta_x = 1$; otherwise, $\delta_x = 0$. Similarly, we also set δ_y and δ_z. Then, the code L_i relative to the parent is

$$L_i = \delta_x + 2\delta_y + 4\delta_z. \tag{A.2}$$

A node has 6 adjacent nodes along to the x, y and z axes. We compute the adjacent node by Algorithm A.1. If we compute the ID of the adjacent node which is the larger side of N along the x axis, we set $v = 1$. However, if we compute the one which is the smaller side along the x axis, we set $v = -1$. Similarly, we set $v = 2$ or $v = -2$ for the adjacent node along the y axis, and we set $v = 4$ or $v = -4$ for the adjacent node along the z axis. In Algorithm A.1, $\otimes$ means bit-by-bit logical AND.

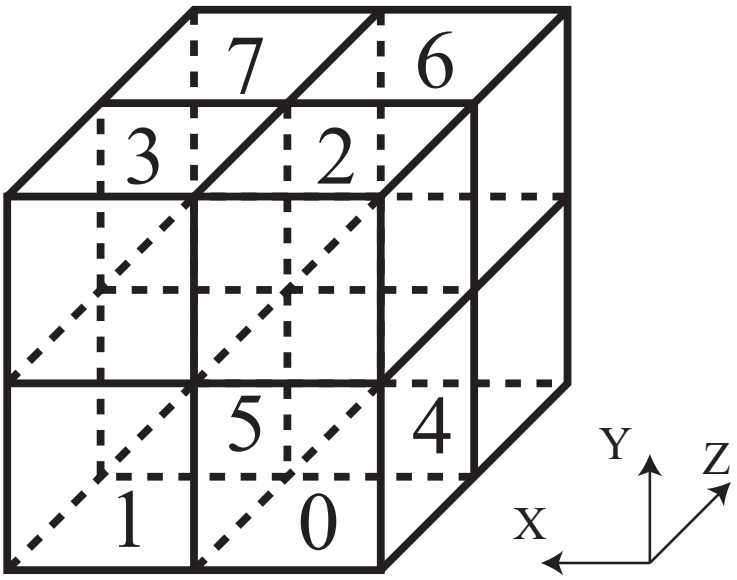


Figure A.1: Coding of 8 child nodes relative to the parent node

Figure A.2 shows one dimensional view of an example of computing the ID of the adjacent node. The ID of node a is $\{0, 1, 1\}$. We compute the adjacent node b of larger side as to x axis, namely $v = 1$. We obtain that the ID of b is $\{1, 0, 0\}$ by **AdjacentNode**$(a, 1)$.

A.2 Determining 8 Voxels of A Cube

The next step is to determine the 8 voxels which consist of a cube. Since the 8 voxels are adjacent to each other, we can determine those ID by applying Algorithm A.1 in order. However, since the depth of a node in the octree varies according to the distance from the surface of the object, there is curvature of the surface and variation of the photometric attributes of the surface. Thus, the depth of the node which is computed by Algorithm A.1 may be less than d; or the node may have child nodes. We have to use terminal nodes, which have no children, for a cube to be applied to MC, because we want to generate a mesh model which is as fine as possible.

Therefore, we have to determine if the computed 8 voxels are suitable for a cube to be applied to MC. Algorithm A.2 shows the algorithm to determine the cube which voxel N belongs to. Since voxel N may belong to 8 cubes, the combination of v_x, v_y and v_z is 8

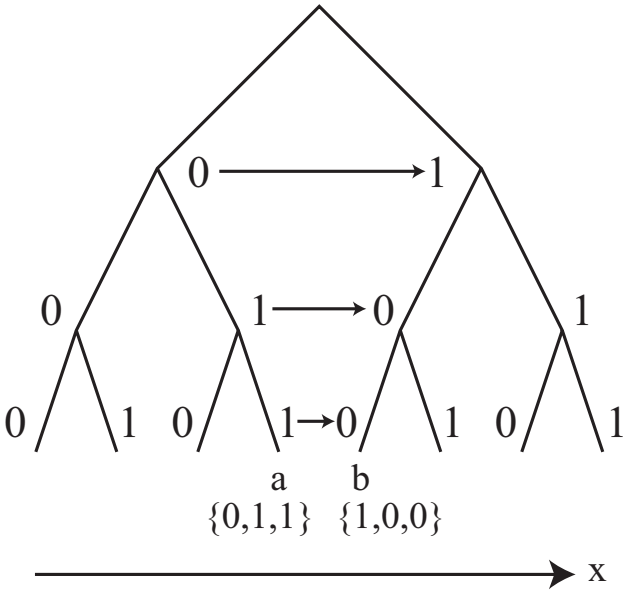


Figure A.2: An example of computing the ID of the adjancent node

cases: {1, 2, 4}, {-1, 2, 4}, {1, -2, 4}, {-1, -2, 4}, {1, 2, -4}, {-1, 2, -4}, {1, -2, -4}, {-1, -2, -4}. If v_x, v_y and v_z are negative, we sort these voxels by swapping them.

If the depth of the node which is computed by Algorithm A.1 may be less than d, some nodes of cube C become the same nodes. In such cases, C is not a hexahedron. However, we can apply it to MC, as described in 4.3. If we apply all nodes to MC when we generate a mesh model, a cube can be repeatedly applied to MC. Thus, the same vertices and triangles are generated redundantly. We can then remove redundant vertices and triangles after generating a mesh model.

Algorithm A.1 AdjacentNode(N, v)

Input: Node of Octree: N

Input: Direction of Adjacent Node: $v(= -4, -2, -1, 1, 2, 4)$

Local: ID of a Node: L, L'

Local: Depth of L: d

Output: Adjacent Node: N'

```
L' ← L
for i = d to 0 do
  if i = 0 then
    adjacent node does not exist
    return
  end if
  if (v > 0 ∧ L'_i ⊗ v > 0) ∨ (v < 0 ∧ L'_i ⊗ (−v) = 0) then
    L'_i ← L'_i − v
  else
    L'_i ← L'_i + v
    exit loop
  end if
end for
N' ← root of octree
for i = 1 to d do
  if N' has children then
    N' ← N'_{L_i}
  else
    return N'
  end if
end for
return N'
```

Algorithm A.2 ComputeCube(N, v_x, v_y, v_z)

Input: Node of Octree: N

Input: Direction of Adjacent Nodes: v_x, v_y, v_z

Output: 8 Voxels of Cube: $C = \{N_i | i = 0, \ldots, 7\}$

$N_0 \leftarrow N$

$N_1 \leftarrow$ **AdjacentNode**(N, v_x)

$N_2 \leftarrow$ **AdjacentNode**(N, v_y)

$N_4 \leftarrow$ **AdjacentNode**(N, v_z)

$N_3 \leftarrow$ **AdjacentNode**(N_1, v_y)

$N_5 \leftarrow$ **AdjacentNode**(N_1, v_z)

$N_6 \leftarrow$ **AdjacentNode**(N_2, v_z)

$N_7 \leftarrow$ **AdjacentNode**(N_3, v_z)

for $i = 0$ to 7 **do**

  if N_i has children **then**

    this cube is not suitable to be applied to MC

    return

  end if

end for

if $v_x < 0$ **then**

  $N_0 \leftrightarrow N_1, N_2 \leftrightarrow N_3, N_4 \leftrightarrow N_5, N_6 \leftrightarrow N_7$

end if

if $v_y < 0$ **then**

  $N_0 \leftrightarrow N_2, N_1 \leftrightarrow N_3, N_4 \leftrightarrow N_6, N_5 \leftrightarrow N_7$

end if

if $v_z < 0$ **then**

  $N_0 \leftrightarrow N_4, N_1 \leftrightarrow N_5, N_2 \leftrightarrow N_6, N_3 \leftrightarrow N_7$

end if

return C

Appendix B

Modeling Results of Cultural Heritage Objects

We now present some results of modeling cultural heritage objects which we have scanned so far. Figure B.1 shows some examples of the objects we have scanned. The two statues of the top row are the Great Buddha of Kamakura and the Great Buddha of Kamakura in Japan. They are also famous big statues of the Buddha. It is outside as we can see and the height of the Great Buddha of Kamakura is 11.3m. We constructed scaffolds to obtain the top view of the statue. However, we cannot see the top of the head and shoulder. We fill holes of the model by the technique described in Chapter 8. The result is shown in Figure B.2. We obtain 16 range images of the Buddha using Cyrax2400 [18]. Each range image has about 0.3 million vertices and 0.6 million triangles The model has about 3.0 million vertices and 6.0 million triangles. We took 61 minutes to merge range images.

The height of the Great Buddha of Nara is 14.98m and it is inside of the hall. We brought up to the ceiling and scanned it from the upper side. We took 90 range images of the Buddha and the total size of range images is about 1.3GB. Figure B.3 shows the merging result. The model has about 7.2 million points and 14.0 million triangles. The computational time is about 31 hours and 44 minutes.

The middle row of Figure B.1 is the stone statue of the Buddha made in China It is smaller than the former two statues. We obtain 40 range images using VIVID [58], which can acquire a range image with RGB values for each pixel of a range image. 7-9 range images overlap at each point of the Buddha. We merge range images with RGB values by the technique described in Chapter 5. Figure B.4 shows the merging result of the Buddha.

The two photographs of the bottom row in Figure B.1 are the Great Buddha in Sukhothai,

Thailand. The statue is inside the wall and we obtain range images from the top of the wall. We have acquired 176 range images (including 68 images of the Buddha and 108 images of the wall). The total size of range image is about 2.5GB (Buddha: 790MB, wall: 1.7GB). Figure B.5 shows the merging result of the Buddha. The model only including the Buddha has about 1.2 million vertices and 4.8 million triangles. The computational time is about 6 hours and 58 minutes.

Figure B.1: Top row: Kamakura Buddha(left), Nara Buddha(right). Middle row: China Buddha. Bottom row: Thai Buddha

Figure B.2: The merging result of the Kamakura Buddha

Figure B.3: The merging result of the Nara Buddha

Figure B.4: The model of China Buddha with RGB values as photometric attributes

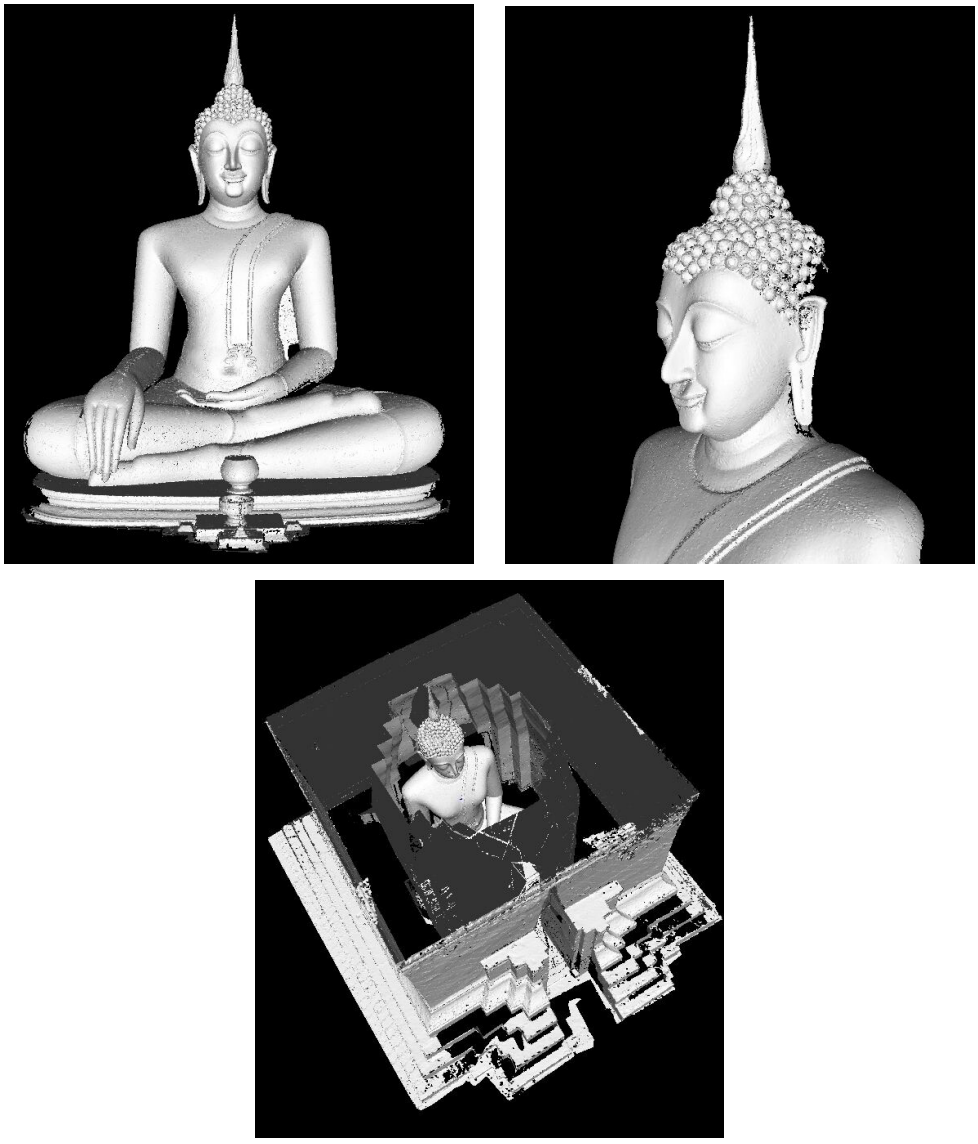


Figure B.5: The model of Thai Buddha and the wall

References

[1] E. Baltsavias, M. Pateraki, and L. Zhang. Radiometric and geometric evaluation of ikonos geo images and their use for 3d building modelling. In *Proc. Joint ISPRS Workshop "High Resolution Mapping from Space 2001"*, Hannover, Germany, 19-21 September. 2001.

[2] D. Bartz and W. Straßer. Parallel construction and isosurface extraction of recursive tree structures. In *Proceedings of WSCG'98*, volume III, Plzen, 1998.

[3] P.A. Beardsley, A. Zisserman, and D.W. Murray. Sequential updating of projective and affine structure from motion. *International Journal of Computer Vision*, 23:235–259, 1997.

[4] J. L Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multi-view registration technique. *IEEE Trans. Patt. Anal. Machine Intell.*, 18(5):540–547, May 1996.

[6] 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, Feb 1992.

[7] D. Brown. The bundle adjustment - progress and prospect. In *XIII Congress of the ISPRS*, Helsinki, 1976.

[8] A. Califano and R. Mohan. Multidimensional indexing for recognizing visual shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16:373–392, 1994.

[9] F.J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 8(6):679–698, 1986.

[10] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. SIGGRAPH 2001*, pages 67–76. ACM, 2001.

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

[12] Y. Chen and G. Medioni. Description of complex objects from multiple range images using an inflating balloon model. *Computer Vision and Image Understanding: CVIU*, 61(3):325–334, 1995.

[13] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, June 1998.

[14] P. Courtney, N.A. Thacker, and C.R. Brown. A hardware architecture for image rectification and ground plane obstacle detection. In *Proc. International Conference on Pattern Recognition*, volume IV, pages 23–26, 1992.

[15] I.J. Cox, S.L. Hingorani, S.B. Rao, and B.M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, May 1996.

[16] W.B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, number 1883 in LNCS, pages 100–115. Springer-Verlag, sep 1999.

[17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH'96*, pages 303–312. ACM, 1996.

[18] Cyra Technologies, Inc. http://www.cyra.com.

[19] K.J. Dana, B. van Ginneken, S.K. Nayar, and J. J. Koenderink. Reflectance and texture of real world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[20] J. Davis, S.R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Proc. First International Symposium on 3D Data Processing, Visualization, and Transmission*, 2002.

[21] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Computer Graphics Proceedings, ACM SIGGRAPH '98*, pages 189–198, July 1998.

[22] H. Delingette, M. Hebert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. *Image and vision computing*, 10(3):132–144, April 1992.

[23] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics*, 33(Annual Conference Series):317–324, 1999.

[24] C. Dorai, G. Wang, A.K. Jain, and C. Mercer. From images to models: Automatic 3d object model construction from multiple views. In *Proc. of the 13th IAPR International Conference on Pattern Recognition*, pages 770–774, 1996.

[25] D.W. Eggert, A.W. Fitzgibbon, and R.B. Fisher. Simultaneous registration of multiple range views for use in reverse engineering. Technical Report 804, Dept. of Artificial Intelligence, University of Edinburgh, 1996.

[26] C. Elachi. *Introduction to the physics and techniques of remote sensing*. New York : Wiley, 1987.

[27] O. Faugeras, B. Hots, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real Time Correlation-Based Stereo: Algorithm, Implementations and Applications. Technical Report N°2013, INRIA, 1993.

[28] M. Fischler and R. Bolles. Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[29] J.D. Foley, A. van Dam, S.K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice in C*. Addison Wesley Professional, 2nd edition, 1995. ISBN:0-201-84840-6.

[30] J.H. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[31] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proc. SIGGRAPH2000*, pages 249–254. ACM, July 2000.

[32] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[33] M. Garland and P.S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proc. IEEE Visualization 1998*, 1998.

[34] M. Greenspan and G. Godin. A nearest neighbor method for efficient ICP. In *Proc. 3DIM*, pages 161–168, 2001.

[35] A. Guttman. R-trees: A dynamic index struc- ture for spatial searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 47–54, 1984.

[36] O. Hall-Holt and S. Rusinkiewicz. Stripe boundary codes for Real-Time Structured-Light range scanning of moving objects. In *Proc. International Conference on Computer Vision*, pages 359–366, 2001.

[37] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conference*, pages 147–151, 1988.

[38] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[39] Tim Hawkins, Jonathan Cohen, and Paul Debevec. A photometric approach to digitizing cultural artifacts. In *2nd International Symposium on Virtual Reality, Archaeology, and Cultural Heritage*, Glyfada, Greece, November 2001.

[40] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Proceedings of European Conference on Computer Vision*, pages 117–126, Springer-Verlag, 1996.

[41] H. Hoppe. Progressive meshes. In *Computer Graphics (SIGGRAPH 1996 Proceedings)*, pages 99–108, 1996.

[42] H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Proc. IEEE Visualization 1999*, pages 59–66, 1999.

[43] H. Hoppe, T. DeRose, T. Duchamp, J.A. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. SIGGRAPH'92*, pages 71–78. ACM, 1992.

[44] K. Ikeuchi, Y. Sato, K. Nishino, R. Sagawa, T. Nishikawa, T. Oishi, I. Sato, J. Takamatsu, and D. Miyazaki. Modeling cultural heritage through observation. In *Proc. of IEEE First Pacific-Rim Conference on Multimedia*, Dec. 2000.

[45] A. Johnson and M. Hebert. Surface registration by matching oriented points. In *Proc. Int. Conf. On Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, May 1997.

[46] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[47] I. Kitahara, H. Saito, S. Akimichi, T. Ono, Y. Ohta, and T. Kanade. Large-scale virtualized reality. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2001) Technical Sketches*, 2001.

[48] LTD KOKUSAI KOGYO CO. Remote airborne mapping & systems. http://www.ramse3d.com/.

[49] R. Kurazume, K. Nishino, Zhengyou Zhang, and Katsushi Ikeuchi. Simultaneous 2d images and 3d geometric model registration for texture mapping utilizing reflectance attribute. In *Proc. The 5th Asian Conference on Computer Vision*, volume 1, pages 99–106, January 2002.

[50] K.N. Kutulakos and S.M. Seitz. What Do N Photographs Tell Us about 3D Shape? Technical Report 680, Computer Science Dept. U. Rochester, Jan. 1998.

[51] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proc. SIGGRAPH 2000*, pages 131–144, 2000.

[52] D. B. Lomet and B. Salzberg. The hb-tree: A multiattribute indexing method with good guaranteed performance. *ACM Trans. Database Systems*, 15(4):625–658, 1990.

[53] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. In *Proc. SIGGRAPH'87*, pages 163–170. ACM, 1987.

[54] P. Mackerras. A fast parallel marching-cubes implementation on the fujitsu ap1000. Technical report, Australian National University, TR-CS-92-10, 1992.

[55] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-d model constuction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 879–883, June 1996.

[56] E.M. Mikhail, J.S. Bethel, and J.C. McGlone. *Introduction to modern photogrammetry*. New York : J. Wiley, 2001.

[57] J.R. Miller. *A 3D Color Terrain Modeling System for Small Autonomous Helicopters*. PhD thesis, Robotics Institute, Carnegie Mellon University, February 2002. CMU-RI-TR-02-07.

[58] MINOLTA Co. Ltd. Vivid 900 non-contact digitizer. http://www.minoltausa.com/vivid/.

[59] Daisuke Miyazaki, Takeshi Ooishi, Taku Nishikawa, Ryusuke Sagawa, Ko Nishino, Takashi Tomomatsu, Yutaka Takase, and Katsushi Ikeuchi. The great buddha project: Modelling cultural heritage through observation. In *Proceedings of 6th International Conference on Virtual Systems and MultiMedia*, pages 138–145, Gifu, 2000.

[60] Myricom, Inc. http://www.myri.com/.

[61] S.K. Nayar and M. Oren. Generalization of the lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14:227–251, 1995.

[62] S.A. Nene and S.K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(9):989–1003, 1997.

[63] P. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *Proc. Int. Conf. on Shape Modeling and Application*, pages 130–139, Mar 1997.

[64] F.E. Nicodemus, J.C. Richmond, J.J. Hsia, I.W. Ginsberg, , and T. Limperis. Geometrical considerations and nomenclature for reflectance. NBS Monograph 160, National Bureau of Standards (US), 1977.

[65] G.M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of Visualization'91*, pages 83–91. IEEE, 1991.

[66] K. Nishino and K. Ikeuchi. Robust simultaneous registration of multiple range images. In *Proc. Fifth Asian Conference on Computer Vision ACCV '02*, pages 454–461, Jan. 2002.

[67] K. Nishino, Y. Sato, and K. Ikeuchi. Eigen-texture method: Appearance compression based on 3d model. In *Proc. of Computer Vision and Pattern Recognition '99*, volume 1, pages 618–624, Jun. 1999.

[68] K. Nishino, Z. Zhang, and K. Ikeuchi. Determining reflectance parameters and illumination distribution from a sparse set of images for view-dependent image synthesis. In *Proc. of Eighth IEEE International Conference on Computer Vision ICCV '01*, volume 1, pages 599–606, July 2001.

[69] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[70] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, ESAT-PSI, K.U.Leuven, 1999.

[71] W. Press, S. Teukolsky, and W. Vetterling. *Numerical recipes in C: the art of scientific computing*. Cambridge university press, 1992.

[72] K. Pulli. Multiview registration for large data sets. In *Second Int. Conf. on 3D Digital Imaging and Modeling*, pages 160–168, Oct 1999.

[73] P. Rander. *A Multi-Camera Method for 3D Digitization of Dynamic, Real-World Event*. PhD thesis, The Robotics Institute Carnegie Mellon University, December 1998. CMU-RI-TR-98-12.

[74] J. T. Robinson. The k-d-b tree: A search structure for large multidimensional dynamic indexes. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 10–18, 1984.

[75] P.J. Rousseeuw and A.M. Leroy. *Robust regression and outlier detection*. Wiley, New York, 1987.

[76] R. Sagawa, K. Nishino, and K. Ikeuchi. Robust and adaptive integration of multiple range images with photometric attributes. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2001*, volume 2, pages 172–179, December 2001.

[77] R. Sagawa, K. Nishino, M.D. Wheeler, and K. Ikeuchi. Parallel processing of range data merging. In *Proc. 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 577–583, Oct. 2001.

[78] R. Sagawa, T. Oishi, A. Nakazawa, R. Kurazume, and K. Ikeuchi. Iterative refinement of range images with anisotropic error distribution. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 79–85, October 2002.

[79] H Samet. The quadtree and related hierarchical data structure. *ACM Computing Surveys*, 16(2):187–260, 1984.

[80] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Trans. on Visualization and Computer Graphics*, 5(1):1–12, 1999.

[81] K. Sato and S. Inokuchi. Range-imaging system utilizing nematic liquid crystal mask. In *Proc. International Conference on Computer Vision*, pages 657–661, 1987.

[82] T. Sato, M. Kanbara, N. Yokoya, and H. Takemura. Dense 3-d reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera. *International Journal of Computer Vision*, 47(1-3):119–129, April 2002.

[83] S. M. Seitz. The space of all stereo images. In *Proc. Eighth International Conference on Computer Vision (ICCV)*, pages 307–314, 2001.

[84] S.M. Seitz and C.R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. In *Proc. of Computer Vision and Pattern Recognition '97*, pages 28–34, 1997.

[85] J.A. Sethian. *Level Set Methods*. Cambridge University Press, 1996.

[86] R. Shekhar, E. Fayyad, R. Yagel, and J. Cornhill. Octree-based decimation of marching cubes surfaces. In *Proc. Visualization'96*, pages 335–342, 1996.

[87] R. Shu, Z. Chen, and M.S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11:202–217, 1995.

[88] D. A. Simon, M. Hebert, and T. Kanade. Realtime 3- d pose estimation using a high speed range sensor. In *Proc. ICRA*, pages 2235–2241, 1994.

[89] I. Stamos and P.K. Allen. Registration of 3d with 2d imagery in urban environments. In *Proc. the Eighth International Conference on Computer Vision, to appear*, Vancouver, Canada, 2001.

[90] F. Stein and G. Medioni. Structural indexing: efficient 3-d object recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):125–145, 1992.

[91] Gabriel Taubin. A signal processing approach to fair surface design. *Computer Graphics*, 29(Annual Conference Series):351–358, 1995.

[92] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *Int Journal of Computer Vision*, 24(3):271–300, 1997.

[93] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[94] R. Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[95] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proc. SIGGRAPH'94*, pages 311–318, Jul 1994.

[96] Mark D. Wheeler. *Automatic Modeling and Localization for Object Recognition*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1996.

[97] M.D. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *Proc. International Conference on Computer Vision*, January 1998.

[98] R.T Whitaker. A level-set approach to 3d reconstruction from range data. *International Journal of Computer Vision*, 29(3):203–231, October 1998.

[99] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science & Engineering*, 4(4):10–21, 1997.

[100] Z.J. Wood, P. Schroder, D.E. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *IEEE Visualization*, pages 275–282, 2000.

[101] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–198, 1998.

[102] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[103] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proc. First IEEE Workshop on Variational and Level Set Methods, in conjunction with Proc. ICCV '01*, pages 194–202. IEEE, 2001.