

Stereo Vision and Range Image Techniques for Generating 3D Computer Models of Real Objects

A Dissertation Presented

by

Soon-Yong Park

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

State University of New York

at

Stony Brook

May 2003

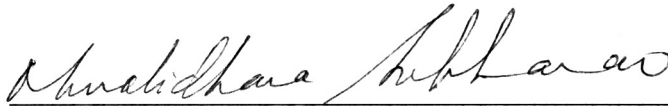
Copyright © by
Soon-Yong Park
2003

STATE UNIVERSITY OF NEW YORK
AT STONY BROOK

THE GRADUATE SCHOOL

Soon-Yong Park

We the dissertation committee for the above candidate for the Ph.D. degree, hereby recommend acceptance of this dissertation



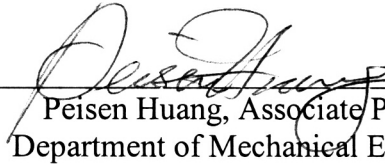
Murali Subbarao, Advisor of Dissertation
Professor, Department of Electrical & Computer Engineering



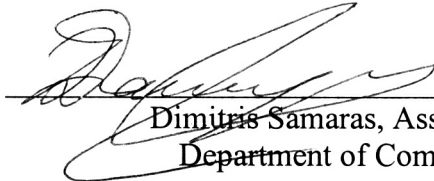
Petar Djuric, Chairperson of Defense
Professor, Department of Electrical & Computer Engineering



Gene Gindi, Associate Professor
Department of Electrical & Computer Engineering



Peisen Huang, Associate Professor
Department of Mechanical Engineering



Dimitris Samaras, Assistant Professor
Department of Computer Science

This dissertation is accepted by the Graduate School

Graduate School

Abstract of the Dissertation

**Stereo Vision and Range Image Techniques for
Generating 3D Computer Models of Real Objects**

by

Soon-Yong Park

Doctor of Philosophy

in

Electrical Engineering

State University of New York at Stony Brook

2003

One topic of research interest today in three-dimensional (3D) model reconstruction is the generation of a complete and photorealistic 3D model from multiple views of an object. This dissertation addresses the problem of generating 3D computer models of real-world objects. We present *Stereo Vision* systems and *Computer Vision* techniques for complete 3D model reconstruction through a sequence of steps: (1) Multi-view range image acquisition (2) Registration and integration of multi-view range images (3) Pose estimation of 3D models (4) Integration of two-pose 3D models and (5) Photorealistic texture mapping.

We present two stereo vision systems to obtain multi-view range images and photometric textures of an object. Each system consists of a stereo camera and a motion control stage to change the view of the object. Calibrations of both stereo cameras and motion control stages are presented. Range images obtained from multiple views of an object are registered to a common coordinate system through the calibrations of the vision systems.

In order to refine the registration of multi-view range images, we introduce a novel registration refinement technique. The proposed technique combines *Point-to-Tangent Plane* and *Point-to-Projection* approaches for accurate and fast refinement.

In order to merge registered range images, we present two different integration techniques. A mesh-based technique integrates range images through merging of multiple contours on a cross section of a volumetric representation of the object. A slice-by-slice integration on all cross sections reconstructs a complete 3D model represented by a set of closed contours. We also present a volumetric multi-view integration technique. In order to remove erroneous points outside of the visual hull of an object, *Shape-from-Silhouettes* technique is combined. A 3D grid of voxels is classified into several sub-regions based on the signed-distances of a voxel to overlapping range images. The iso-surface of the object is reconstructed by a class-dependent technique of averaging the signed distances. *Marching Cubes* algorithm then converts the iso-surface representation of the object to a 3D mesh model.

For many real objects, using a single pose yields only a partial 3D model because some surfaces of the object remain hidden from a range sensor due to occlusions or concavities. In order to obtain a complete and closed 3D model, we generate two 3D models of the object, register and integrate the 3D models into a single 3D model. By placing the object in different suitable poses and sensing the visible surfaces, we reconstruct two partial 3D models. We then merge the partial 3D models by novel pose registration and integration techniques. Registration of two pose models consists of two steps, coarse registration, and its refinement. A pose estimation technique between two 3D models is presented to determine coarse registration parameters. The pose estimation technique finds a stable tangent plane (STP) on a 3D model which can be transformed to the base tangent plane (BTP) of the other model and vice versa. After pose estimation, the two pose models are integrated to obtain a complete 3D model through a volumetric pose integration technique. The integration technique merges two iso-surfaces of the corresponding partial 3D models. Texture mapping finally generates photorealistic 3D models of real-world objects.

To my family

Contents

Abstract	iii
List of Figures	xii
List of Tables	xiii
Acknowledgements	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Literature Review	4
1.2.1 Range Image Acquisition	5
1.2.2 Multi-view Registration	6
1.2.3 Multi-view Integration	7
1.2.4 Pose Estimation	8
1.2.5 Pose Integration	9
1.3 Dissertation Overview	9
2 Parallel Stereo Vision System (SVIS-2)	12
2.1 Introduction	12
2.2 System Overview	12
2.3 Vision System Calibration	13
2.3.1 Focus Calibration	13
2.3.2 Stereo Camera Calibration	15
2.3.3 Rotation Stage Calibration	16
2.3.4 Calibration Test	18
2.4 Partial Shapes Acquisition	19
2.4.1 Multi-resolution Stereo Matching	19
2.4.2 Experimental Results	22
2.5 Conclusions	22

3	Vergence Stereo Vision System (SVIS-3)	24
3.1	Introduction	24
3.2	Stereo Camera Calibration	25
3.3	Stereo Rectification	26
3.3.1	Pinhole Camera Model	26
3.3.2	Epipolar Geometry	28
3.4	Rectification of Camera Matrices	29
3.5	Experimental Results	30
3.5.1	Stereo Camera	30
3.5.2	Camera Calibration	30
3.5.3	Stereo Rectification	33
3.6	3D Reconstruction	34
3.6.1	Simple Triangulation	35
3.6.2	Solution of a Linear Equation	35
3.6.3	Comparison of Reconstruction Methods	37
3.7	Turntable Calibration	39
3.7.1	Rotation Stage Calibration	39
3.7.2	Experimental Results	43
3.8	Conclusions	44
4	Surface-based 3D Reconstruction	46
4.1	Introduction	46
4.2	Multi-view Registration	47
4.2.1	Coarse Registration	47
4.2.2	Partial Shape Representation	47
4.3	Registration Refinement	49
4.3.1	Epipolar Geometry of Multiple Views	50
4.3.2	Pairwise Refinement	52
4.4	Multi-view Integration	56
4.4.1	Partial Shape Integration	56
4.4.2	Mesh Generation	58
4.5	Experimental Results	59
4.6	Conclusions	60

5	Volumetric 3D Reconstruction	65
5.1	Introduction	65
5.2	Registration of Multiple Range Images	66
5.2.1	Multi-view Registration	66
5.2.2	Calibration Parameter Optimization	70
5.3	Multi-view Integration	72
5.3.1	Surface Representation	72
5.3.2	Volume Intersection	75
5.3.3	Partial Shapes Integration	76
5.4	Experimental Results	80
5.4.1	Registration Results	81
5.4.2	Integration Results	82
5.5	Error Analysis	91
5.6	Conclusions	92
6	Pose Estimation	96
6.1	Introduction	96
6.2	Methodology of Pose Estimation	97
6.2.1	Base Tangent Plane	97
6.2.2	Stability Constraints	100
6.3	Pose Estimation	100
6.3.1	Finding Tangent Planes	101
6.3.2	Finding Stable Tangent Planes	103
6.3.3	Matching Tangent Planes	105
6.4	Experimental Results	107
6.5	Conclusions	108
7	Pose Integration for Complete 3D Reconstruction	114
7.1	Introduction	114
7.2	Pose Registration	114
7.3	Pose Integration	115
7.4	Texture Mapping	118
7.4.1	Texture Blending	118
7.4.2	Textures in Occlusion Area	121
7.5	Experimental Results	122
7.6	Conclusions	124

8	An Efficient Point-to-Plane Registration Technique	130
8.1	Introduction	130
8.2	Comparison of Registration Techniques	131
8.2.1	Description of Three Registration Techniques	131
8.3	Contractive Projection Point (CPP) Technique	132
8.3.1	Combining Point-to-Plane and Point-to-Projection Techniques	132
8.3.2	Contraction Mapping Property of CPP	134
8.3.3	Convergence Conditions	136
8.3.4	CPP Algorithm	137
8.4	Experimental Results	139
8.4.1	Test Objects	139
8.4.2	Registration Error with respect to Iteration of Projections	140
8.4.3	Pair-wise Registration	140
8.4.4	Multi-view Registration	143
8.5	Conclusions	145
9	Conclusions	147
9.1	Summary	147
9.2	Future Work	148
	Bibliography	151

List of Figures

1.1	Flowchart of complete 3D model reconstruction	3
2.1	Diagram of SVIS-2 vision system	14
2.2	Picture of SVIS-2 vision system	14
2.3	Plot of object distance vs. focus step of C-3030 digital camera	15
2.4	Parallel stereo geometry of SVIS-2 system	17
2.5	Multi-view geometry of SVIS-2 system	18
2.6	Calibration patterns.	20
2.7	Result of registration of two control point sets	20
2.8	An example of object segmentation and stereo matching	22
2.9	Experimental result of stereo matching	23
3.1	Epipolar geometry of a vergence stereo camera	28
3.2	Picture of SVIS-3 stereo vision system	31
3.3	Checkerboard pattern for camera calibration	32
3.4	Stereo images of the checkerboard pattern	33
3.5	Rectified stereo images	35
3.6	Experimental results of stereo matching	36
3.7	Checkerboard patterns for turntable calibration	38
3.8	Rotation axis calibration with respect to the world coordinate system	41
3.9	The rotation center is one of the intersections of two circles c_1 and c_2	42
3.10	World coordinate system on the checkerboard patterns	44
4.1	Representation of a volumetric workspace	48
4.2	Hierarchical structure of contour segments	49
4.3	The structure of a segment linked list	50
4.4	Epipolar geometry of \mathcal{V}_0 and \mathcal{V}_1	51
4.5	Finding closest point on epipolar line	54
4.6	Connection of two contour segments	57
4.7	Connection of two linked lists	58
4.8	Mesh generation between two slice planes.	59
4.9	Test objects for experiments	60

4.10	Four partial views of 'Duck' object	61
4.11	Pairwise merging of two partial shapes	62
4.12	Integrated model of 'Duck' object	62
4.13	Integrated model of 'Head' object	63
4.14	Integrated model of 'Potatohead' object	63
4.15	Translation errors in the X and Z directions for objects	64
5.1	SVIS-2 vision system geometry	67
5.2	Registration of two partial surfaces	68
5.3	Multi-view registration strategy	70
5.4	Rotation center refinement	71
5.5	Signed distance computation for a voxel	73
5.6	Overlapping surfaces from a voxel \mathbf{P}	74
5.7	Selecting valid partial surface for signed-distance averaging	75
5.8	Reconstruction using <i>Shape-from-silhouettes</i> technique	77
5.9	Binary voxel code for 8 multi-views	78
5.10	Examples of combinations of two bits voxel code.	79
5.11	Voxel classification	79
5.12	Test objects for volumetric 3D model reconstruction	83
5.13	Partial shapes of 8 views of 'Monkey'	84
5.14	Registered partial shapes of 'Monkey'	84
5.15	Registration errors of 'Monkey' Object	85
5.16	Registration errors of objects 'Pokemon' and 'Nikon775'	86
5.17	Registration errors of objects 'Polarbear1', 'Polarbear2', 'Potatohead', and 'Dog'	87
5.18	Reconstructed 3D models of 'Monkey' with different voxel size	88
5.19	Mesh and shading results for test objects-1	89
5.20	Mesh and shading results for test objects-2	90
5.21	Test object 'Cubes' for error analysis	91
5.22	Test object 'Cylinder' for error analysis	92
5.23	RMS registration error between the ground truth and the reconstructed 3D models	93
5.24	Points clouds of registered 'Cylinder' model	95
6.1	Schematic diagram of our 3D modeling	98
6.2	Uniqueness of tangent plane matching	99
6.3	Initializing tangent plane and its supporting vertices	102
6.4	Construction of a tangent plane on a 3D model	104
6.5	An unstable tangent plane	105
6.6	Finding STPs on an object based on geometric constraints)	106

6.7	Pose estimation results of 'Monkey' object	110
6.8	Pose estimation results of 'Pokemon' object	111
6.9	Pose estimation results of 'Nikon775' object	112
6.10	Pose estimation results of 'PolarBear' object	113
7.1	A schematic diagram of overlapping regions between two scans	115
7.2	Signed distance in a concave region	117
7.3	Errors of mesh growing in a concave region	118
7.4	Artifacts in a concave region due to a reconstruction error	119
7.5	Texture blending on a triangle mesh	120
7.6	Barycentric coordinates for texture blending	121
7.7	Visibility test for texture mapping	122
7.8	Pose integration and texture mapping results of 'Monkey'	124
7.9	Pose integration and texture mapping results of 'Pokemon'	125
7.10	Pose integration and texture mapping results of 'Nikon775'	126
7.11	Pose integration and texture mapping results of 'Polarbear'	127
7.12	Pose integration results of 'Potatohead'	128
7.13	Texture mapping results of 'Potatohead'	129
8.1	Three common registration techniques	133
8.2	Finding the intersecting point \mathbf{Q}_s from \mathbf{P}_0 by the proposed algorithm.	135
8.3	Contraction mapping property of a new 2D coordinate system.	136
8.4	Three cases of projections to a normal vector	138
8.5	Point clouds models of test objects	139
8.6	Comparison of convergence and translation errors with respect to different number of projections	141
8.7	RMS error comparison of different registration techniques	144
8.8	Results of multi-view registration for 'Potatohead' object after 50 iterations	146

List of Tables

2.1	Rotation calibration error analysis	19
3.1	Results of the width of the test pattern	38
3.2	Registration error of turntable calibration in <i>mm</i>	45
5.1	Statistics for the results of 'Monkey'	82
5.2	RMS and maximum errors of 'Cubes'	94
5.3	RMS and maximum errors of 'Cylinder'	94
6.1	Number of tangent planes on 3D models	108
6.2	Pose estimation error and estimation time	109
7.1	Processing time in <i>sec</i> for pose integration	123
7.2	Number of hidden surfaces after pose integration	126
8.1	Test objects for IPP registration	140
8.2	Registration error and processing time after 50 iterations	143
8.3	Multi-view registration error and processing time after 50 iterations	145

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor Professor Muralidhara Subbarao for offering me the opportunity to do my research, for giving me valuable suggestions and advice, and for letting me have all freedom to pursue my interests. Four years ago, when I decided to start my PhD study, he gave me a great opportunity to join the Computer Vision Lab in this school. He has always been a great advisor and friend for the past four years. He also provided valuable comments and suggestions for improvement of many research papers and this thesis. I also sincerely thank Professors Peter Djuric, Gene Gindi, Peisen Huang, and Dimitris Samaras for serving on my thesis committee and for their valuable comments.

My special thanks go to Professor Sung-Il Chien of Kyungpook National University, Korea, who introduced me to the world of Computer Vision. Since my Master's study, he has always been a source of valuable advice and encouragement.

I must acknowledge my debt to my family - parents, parents-in-law, and my lovely son and daughter, Sang-Wook and Sang-Ah, for their encouragement through these years. My final and warmest thanks must go to my wife Dr. Jae-Kyoung Moon. I would have not finished my study without her support, patience, and understanding.

The support of this research in part by the Olympus Optical Corporation is gratefully acknowledged.

Chapter 1

Introduction

Automatic reconstruction of photo-realistic three-dimensional (3D) models of real world objects has been a topic of much interest for applications such as *E-Commerce*, *Human-Computer Interaction*, *Reverse Engineering*, and *Augmented Reality*. Recently, there has been considerable progress in automatic 3D reconstruction techniques that build 3D models through merging of multi-view range images of objects. This dissertation addresses issues of automatic 3D reconstruction by developing techniques of obtaining multi-view range images, merging range images into 3D models, estimating pose of 3D models, merging multi-pose models, and finally displaying photo-realistic images of real objects.

This chapter begins by describing a set of problems in the frameworks of automatic 3D reconstruction. We then discuss some relevant previous work and conclude with an overview of the dissertation.

1.1 Motivation

This dissertation presents automatic 3D reconstruction techniques to reconstruct complete 3D models of real-world objects. The research on 3D reconstruction has been one of the interesting topics in *Computer Vision* and *Computer Graphics*. It also has a variety of applications such as *E-commerce*, *Reverse Engineering*, *Computer Animation*, and *Augmented Reality*, etc. In *Computer Vision* and *Computer Graphics*, the outer surfaces of a rigid-body object, which is in 3D space, may be represented by a finite number of manifold polygons (usually by a set of triangles). Each polygon is a representation of a surface patch on the real object's outer surfaces. The overall goal of this dissertation is automatic 3D reconstruction of real world objects through complete representation of the objects using manifold polygons.

Let us consider a rigid-body object in 3D space. We call a 3D surface model of the object is complete (or closed) when its outer surfaces are exact representations of geometric and photometric structures of the real object's all visible surfaces. In

order to reconstruct a complete 3D model (a representation of the object), the 3D model should be reconstructed such that all manifolds (or surfaces) have the same geometric and photometric structures as the real surfaces of the object.

A 3D surface model of a real object may be reconstructed by either *passive* or *active* technique. Active technique uses a projected signal which scans over the scene of interest. The most common techniques, in *Computer Vision*, are structured-light and laser projection. They are the most reliable methods of acquiring 3D images. Passive technique means that a vision system works only on naturally occurring images produced by reflected light from the scene or object. *Depth from Stereo* and *Depth from Motion* are two most obvious examples.

Range image is a 2D depth image of a scene of interest, which is obtained from a single view direction of a range sensor. It represents the geometric 3D structure of the scene from the given view point. Most range sensors produce the range image of a scene in terms of a 2D image, where its intensity is the measure of depth from the sensor to the scene. Therefore, strictly speaking, range image is a 2.5D image rather than a 3D image. The 3D surface model of the range image is then obtained by polygonizing it into a set of triangles or rectangles. However, both a range image and its 3D surface model are often called *Range Image* without distinction. In this dissertation, we also call both a 2.5D range image and its 3D surface model as a range image.

Let us consider the problem of reconstructing a complete 3D model of a real object. If we obtain a range image of the object from a single view point, it is only a *partial surface* of the object's whole structure. Recently, there has been many investigations on complete 3D reconstruction which produces full 3D structures of the object. The central technique of complete 3D reconstruction is merging multi-view range images into a single and closed 3D model. Multi-view range images are obtained from multiple view points in order to collect geometric structures of all visible surfaces of the object. And they are merged into a complete 3D model through two important steps, *registration* and *integration*.

Figure 1.1 shows a flowchart of our 3D model reconstruction steps. Let us consider the reconstruction of the first pose of an object. Range images obtained from multiple views of the object must be brought into a common coordinate system so that their geometrical and photometrical structures are aligned. This requires determining rigid transformations from the all multiple view points to the common coordinate system. We call this process as *Multi-view Registration*. Given registered multi-view range images (or partial shapes), it is again required to merge all partial surfaces into a single surface to reconstruct a complete 3D model. The process which merges all partial shapes into a single surface is called *Multi-view Integration*. When an integrated 3D model has complete structures on its all outer surfaces, without any hole, we call the 3D model is complete (or closed).

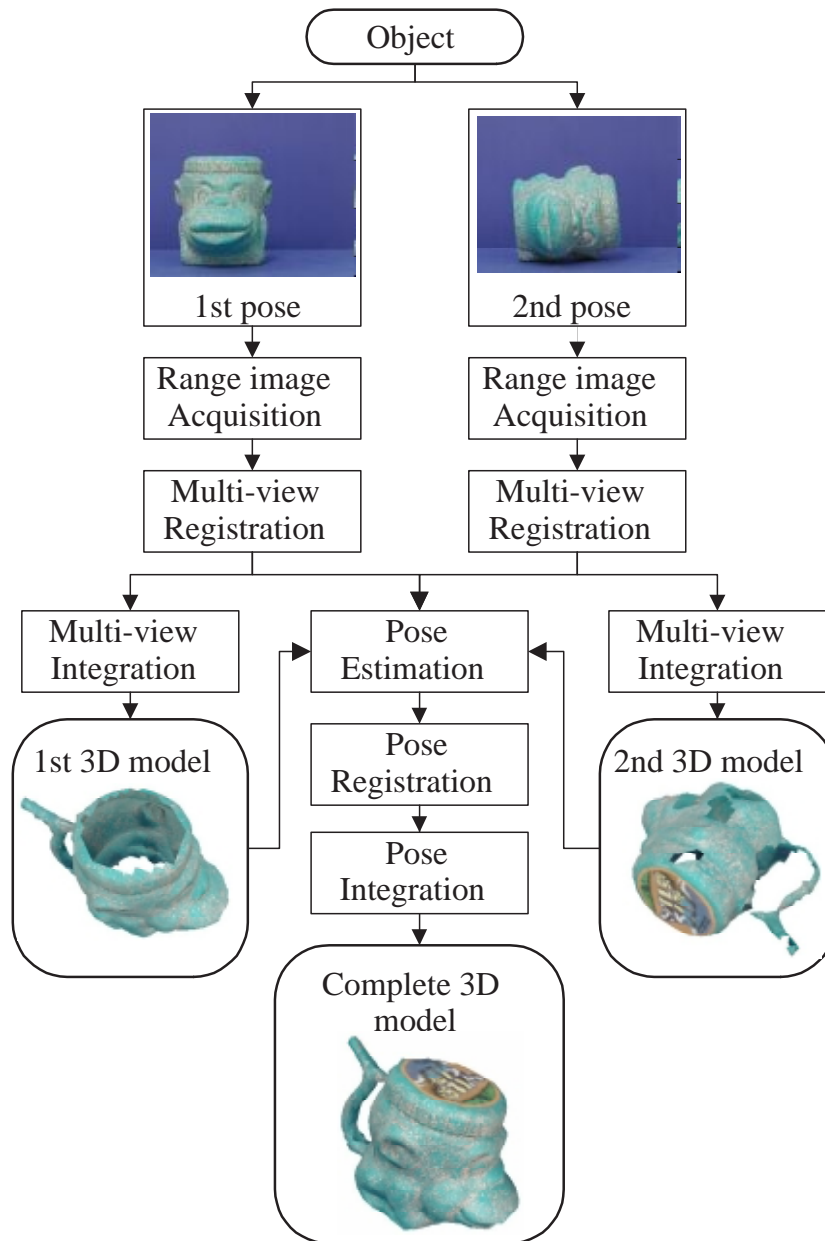


Figure 1.1: Flowchart of complete 3D model reconstruction

Most investigations on such multi-view 3D reconstruction have been limited to using a single pose of an object. However, for many real objects, using a single pose yields only a partial 3D model because some surfaces of the object remain hidden from the range sensor for any given pose due to occlusion, concavities, etc. An example of incomplete reconstruction is shown for the first pose of the object in Figure 1.1. The 3D model experiences some holes on its outer surface due to occlusions. This kind of incompleteness may happen on most real objects. For example, a tea cup placed upright on a turntable hides the bottom surface of the cup from the range sensor. A 3D model of such hidden surfaces could be reconstructed by placing the object in a different suitable pose (e.g. by placing the tea cup on its side) and sensing the visible surfaces. This yields a second partial 3D model for the new pose, for example the second pose model in Figure 1.1. In order to obtain a complete 3D model as shown in the figure, the two partial 3D models for the two different poses need to be registered and integrated. However, since there is no a priori information of the rigid transformation between two different poses, it also needs to estimate the pose between the 3D models before registration. For this reason, only a few researchers have considered *Pose Registration* and *Pose Integration* problems.

In this dissertation, we address some problems of automatic and complete 3D reconstruction. We address the problems by presenting techniques from building a range image acquisition system to mapping textures on reconstructed 3D models. The problems addressed in this dissertation are:

- How can we obtain an object’s surface geometry using a digital stereo camera?
- How can we accurately register and integrate multi-view range images into a single seamless 3D surface model?
- How can we obtain all outer surfaces of an object placed on a planar turntable?
- How can we automatically reconstruct a complete and closed 3D model of the object by integrating two 3D models obtained with different poses?
- How can we estimate a rigid transformation matrix between two 3D models?

1.2 Literature Review

The technique of 3D reconstruction from multi-view range images has been addressed by many researchers. In this section, we summarize the previous work in areas of range image acquisition, multi-view registration, multi-view integration, and pose estimation and integration.

1.2.1 Range Image Acquisition

Range image (or partial shape) is a 3D structure of a scene of interest acquired from a given view point. Many vision techniques have been investigated for obtaining range images (or partial shape) of objects. Some popular techniques are *stereo image analysis*, *laser range imaging*, *structured light*, *shape from focus and defocus*, etc. Among them, *laser range imaging* and *structured light* techniques are the most widely used *active* techniques. In contrast, *stereo image analysis* (SIA) and *shape from focus and defocus* are the most widely used *passive* techniques [41, 48, 92, 91, 93]. Many commercial products based on *active* techniques are already available, but they are usually very expensive. In contrast, *passive* techniques are generally less expensive than *active* techniques. For example, SIA technique produces range images using disparity between the coordinates of two image points, which are projections of the same 3D point to stereo image planes.

Laser range imaging and structured light techniques are the most common active techniques. These techniques project special light patterns onto the surface of a real object to measure the depth to the surface by a simple triangulation technique [19]. Some common patterns for the structured light are a single line pattern [53], a multi-line pattern [75], a colored pattern [109], and a space-time coded pattern [82]. Advantages of using the active techniques are accuracy and speed of depth acquisition. Some recent investigations on this technique show that they can reconstruct a complete 3D model very accurately in real-time [109, 82]. However, the cost of active techniques is still expensive than that of passive techniques.

In contrast, the *passive* techniques work only on naturally formed images produced by reflected light from an object. A common technique is Stereo Image Analysis (SIA). The task of searching correspondence in a pair of stereo images is called *stereo matching*. In Computer Vision research, stereo matching has been an important problem in the last thirty years [21, 51, 68]. However, due to inherent stereo problems (mis-matching, occlusion), it is still considered a difficult problem. One of the simple and efficient techniques in SIA is a correlation-based technique [29, 33]. Correlation-based technique finds stereo correspondence in a pair of stereo images which maximizes a cross-correlation function. It is easy to implement and robust, but computationally expensive. With the growing computing power, some approaches have been investigated for fast stereo matching by employing multi-resolution techniques [94, 102]. Gaussian pyramids are used for generating multi-resolution stereo images, and coarse-to-fine stereo matching techniques reduce computation time [13, 46].

1.2.2 Multi-view Registration

Obtaining range images of objects from a fixed range sensor results in some occluded regions due to the object’s hidden surfaces. In order to view the occluded parts of an object, either the object or the camera has to be moved. Then multi-view range images can be obtained from different view directions to view such hidden surfaces. Because all range images are constructed with reference to different camera coordinate systems corresponding to different directions of view, it is necessary to register reconstructed partial surfaces. This requires determining the relative position and orientation between each camera coordinate system and a reference (common) coordinate system.

Registration of multiple partial shapes generally consists of two steps, *coarse registration* and *registration refinement*. Coarse registration determines approximate rigid transformation parameters between each camera coordinate system to a common coordinate system. In a calibrated vision system, those parameters are already known from the calibration of the vision system. Registration refinement is a process of refining the parameters to minimize registration errors between all overlapping partial shapes. Without the refinement step, an integrated 3D model may include some geometric artifacts on its surface or photometric texture mismatch between different views.

There are mainly three approaches for multi-view registration: *point-to-point*, *point-to-tangent plane*, and *point-to-projection* approaches. A popular method in *point-to-point* approach is the *Iterative Closest Point* (ICP) algorithm [9, 81]. In case of two overlapping shapes, for example, the ICP algorithm iteratively minimizes overlapping errors between two shapes by estimating transformation parameters using two closest point sets on both shapes. The *point-to-tangent plane* approach finds the control point sets such that a point on a tangent plane at a destination partial shape is intersected from a control point on a source partial shape [17, 7, 22, 30]. The Euclidean distance from the source point to the tangent plane is measured as a cost function of registration error between two views. The *point-to-tangent plane* technique is a more stable technique because it uses geometric properties of the overlapping shapes [7, 70]. *Point-to-projection* approach finds the correspondence of a source control point by projecting the source point onto the destination surface from the point of view of the destination [10, 64].

Many researchers have investigated the registration problem for calibrated and uncalibrated vision systems. When a vision system is not calibrated, an auto calibration technique in projective space is used [97, 107], or motion parameters between different views are estimated [24, 105]. In contrast, when the system is calibrated, an initial estimate of calibration parameters is usually computed at the beginning, and refined later during the registration step [17, 76, 101]. Our vision system is initially

calibrated by the Tsai’s method and refined in the registration step.

1.2.3 Multi-view Integration

After the registration step, multiple partial shapes have to be integrated into a complete 3D model. There are two common approaches for multi-view modeling techniques. The first one is merging multiple range images into a complete 3D model [8, 27, 87, 34, 18, 103]. The other approach is based on processing photographic images using volumetric modeling technique, such as *Voxel Coloring* or *Shape-from-Silhouettes* [12, 26, 86]. In this dissertation, we present a 3D model reconstruction technique based on merging of multiple range images of an object. Multi-view (or N -view) range images are acquired from different views of the object. Viewing direction can be changed by either moving the sensor or the object. A moving object on a turntable [18, 26] with a fixed sensor, or a moving sensor with a fixed object [2, 65] have been used by researchers, in addition to other variations [36, 37, 50].

Mesh-based integration and volumetric integration are popular techniques of merging range images [73, 85, 88, 99]. One of the mesh-based integration techniques is based on stitching neighborhood meshes from one view after segmenting meshes according to a ‘Region of Construction’ algorithm [54]. Mesh ‘zippering’ and mesh ‘growing’ are other techniques investigated by Turk et al. [99] and Rutishauser et al. [83], respectively. However, it is not easy to find connecting points between two surfaces in mesh-based integration. A technique for mitigating this problem is to integrate meshes slice-by-slice by considering cross-sections of the object [69].

In volumetric integration approach, implicit representation of object’s surfaces is commonly used [18, 23, 34, 35, 77, 79, 70, 103]. Hoppe et al. [35] have reconstructed an implicit surface model from unorganized points. Their approach assumes that all points are on the physical object surface, which may not be true in a practical system. Curless and Levoy [18] have used multiple range images obtained from a laser range finder to construct an object surface. Hilton et al. [34] also used multiple laser-range images and assumed noise-free partial shapes. Because they have all used laser range finders to acquire partial shapes, errors in partial shapes are negligible. An approach to handle the noise problem in partial shapes has been investigated recently by Wheeler et al. [103]. They use a ‘Consensus-Surface’ algorithm to find the points which have similar geometric conditions with each other to get the ‘Consensus-Surface’. However, he also uses a range finder and many overlapping partial shapes for one surface point to minimize the effect of errors. Some other approaches use multiple video sequences to generate in real-time a 3D model of an object for virtual reality applications [43, 62, 61, 79, 84, 100].

Most researches on 3D modeling have been done using range images obtained from high-quality laser ranging systems or structured light systems. However, rela-

tively less research has been done on using passive techniques, such as Stereo Image Analysis (SIA) or Depth from Focus, for complete 3D modeling. This is mainly due to the inherent problems (e.g. mismatching and occlusion) of stereo matching [25, 54, 106]. Chen [16] has used a stereo camera to get partial shapes and integrated them using a volumetric method. His approach looks similar to ours, but he used a 3D volume mainly for finding a disparity surface, not for integrating partial shapes. Rander et al. [79] and Vedular et al. [100] also used stereo algorithm to create 3D models of a dynamic scene for Virtual Reality applications. They use a large number of cameras and a Multiple Baseline (MB) stereo matching technique. They mitigate stereo correspondence problem using 'shape from silhouettes' technique as a post-processing operation to the stereo algorithm. But a simple space carving algorithm can also remove the object area due to errors in registration step. There are also some investigations to mitigate these problems by integrating Image Focus Analysis (IFA) and Image Defocus Analysis (IDA) with SIA [106, 54]. IFA and IDA provide an approximate 3D shape of the object which simplifies the stereo matching problem in SIA. The approximate 3D shape is refined by SIA to obtain a more accurate 3D shape.

1.2.4 Pose Estimation

In this dissertation, one of the interesting research issues is pose estimation between 3D models. Because our approach of complete 3D model reconstruction is merging two partial 3D models into a single 3D model, we have to register two 3D models by estimating the pose between them. Registration of two pose models consists of two steps, coarse registration and its refinement. Coarse registration is estimating rigid transformation between two 3D models to approximately register the models. It can be considered a pose estimation problem of the models [14, 57]. Pose estimation is a very interesting problem in computer vision, and there has been many investigations [14, 42, 44, 95, 108].

Pose estimation between multiple 3D models can be done based on either geometric features or photometric features of the models. Barequet [5] employs volumetric structures of 3D models to estimate the pose between models. Cyr [20] introduces a pose estimation technique using geometric features of 2D projection images of a 3D model. Similar technique is also investigated by Winkler [104]. There are also considerable investigations on the pose estimation using the (Extended) Gaussian Image of a 3D model [38, 39, 45, 55]. In this dissertation, we introduce a pose estimation technique of two 3D models to determine coarse registration parameters [70, 71]. The pose estimation technique finds a stable tangent plane (STP) on a 3D model which can be transformed to the base tangent plane (BTP) of the other model and *vice versa*.

1.2.5 Pose Integration

Registration and integration of partial 3D models into a single 3D model is also a very difficult problem. For this reason, only a few researchers have considered this problem. P. Allen and R. Yang [2] stitch a bottom surface of an object by matching edge features of the object’s 3D model with an edge image of the bottom. They acquire multi-view range images of a fixed object using a moving range finder. After reconstructing a 3D model, they acquire a partial shape of the bottom surface and stitch the shape and texture of the bottom to the 3D model using a feature matching technique. K. Wong and R. Cipolla [105] employ a shape-from-silhouettes technique for 3D modeling and combine a *structure-from-motion* technique to estimate registration parameters of multiple views. But they manually register the top and the bottom surfaces of the object. W. Niem [65] also reconstructs complete 3D models using a *shape-from-silhouettes* technique, registers, and integrates the top and the bottom surfaces manually. H. Lensch et al. [52] and Y. Iwakiri and T. Kaneko [40] use silhouette matching techniques for registration and stitching of textures to a 3D model. However, their investigations consider only the registration and stitching of texture properties of an object, not the geometric reconstruction of the object. D. Huber [36] also presents a 3D reconstruction technique using an unconstrained registration of n-view partial shapes. He registers the partial shapes using *Spin* images and a graph searching technique.

1.3 Dissertation Overview

This dissertation is organized as follows. In Chapter 2 and Chapter 3, we introduce two stereo vision systems, SVIS-2 and SVIS-3. The SVIS-2 system consists of a parallel-axis stereo camera and a turntable stage. We use a simple camera calibration technique because we know the system parameters such as the baseline of the stereo camera and the rotation angle of the turntable. We have calibrated the transformation parameters of the turntable axis with respect to the camera coordinate system. We also present a multi-resolution stereo matching technique to obtain a range image from a pair of stereo images.

Chapter 3 introduces another 3D vision system called SVIS-3. The system’s configuration is similar to the SVIS-2 system, but we use a toed-in stereo camera to more effectively accommodate the size of an object. Instead of fixing the position of the stereo camera, we freely move the stereo camera on the planar table top to change the distance from an object depending on the size of the object. We accurately calibrate the stereo camera and rectify a pair of images to make the epipolar line to be on the same horizontal line on the images. We also calibrate the turntable coordinate

system with respect to the camera coordinate system.

In this dissertation, both mesh-based and volume-based integration techniques are presented to integrate multiple range images. In Chapter 4, we present a mesh-based 3D model reconstruction technique. In this chapter, a new registration algorithm based on an epipolar line constraint between two view directions is proposed. In order to integrate multiple partial shapes, we divide the partial shapes into many slices, segment a partial shape into several contours, and integrate multi-view contours after removing some erroneous segments. After integration, we polygonize all slices of contours into a 3D mesh model.

In Chapter 5, we present another 3D reconstruction technique based on a volumetric integration of multiple range images. The proposed technique registers all range images using a point-to-plane registration technique. After refinement, we compute the implicit surface function of the object in a volumetric space and convert it to a 3D mesh model using the Marching Cubes algorithm [56]. We introduce a new volumetric integration technique by classifying the volume into multiple regions based on the signed distance of a voxel. We also test the visibility of the voxel from all view points from its simple visibility code, which is called *voxel code*. In order to remove some erroneous voxels, we combine the *Shape-from-Silhouettes* technique [3, 49, 59, 65, 66, 89, 96].

A novel pose estimation and an integration techniques are presented in Chapter 6 and Chapter 7. These techniques facilitate integration of two 3D models into a complete and closed 3D model. In Chapter 6, we present a novel pose estimation technique to determine coarse registration parameters between two 3D models of the object. We introduce a simple pose estimation technique based on matching tangent planes of a 3D model with the base tangent plane (BTP) which is invariant for a vision system. By matching the BTP of one 3D model to a STP of the other model, we derive a pose transformation matrix and register the models to a common coordinate system. Chapter 7 addresses a pose integration problem in order to merge two partial 3D models into a complete and closed 3D model. The novelty of our pose integration technique is merging two incomplete iso-surfaces which represent two different models. Since the two 3D models are already reconstructed through integration of n -view range images in each pose, we integrate two iso-surfaces rather than $2 \times n$ -view range images.

In Chapter 8, we address a registration refinement problem and present an accurate and fast *Point-to-(Tangent) Plane* registration technique. We introduce a novel *Point-to-Plane* registration technique by incorporating the high-speed advantage of the *Point-to-Projection* approach. In order to find the intersection point fast and accurately, we forward-project the source control point to the destination surface and reproject the projection point to the normal vector of the source point. Therefore, we call our technique as *Contractive Projection Point* (CPP) algorithm. Finally, in

Chapter 9, we conclude the dissertation by summarizing presented 3D reconstruction systems and techniques. Future research topics are also presented.

Chapter 2

Parallel Stereo Vision System (SVIS-2)

This dissertation introduces two stereo vision systems for range image acquisition. This chapter presents one of them, which is named SVIS-2 (Stony Brook Vision System-2). Calibration of the vision system and acquisition of multi-view range images are presented.

2.1 Introduction

This chapter presents a stereo vision system for acquisition of multi-view range images. This system is fixed on the top of a planar table while obtaining range images. In order to reconstruct a complete 3D model of an object, we need multiple range images obtained from different views of the object. A rotation stage (turntable) is employed to change the object's view for multiple range image acquisition.

We calibrate the stereo camera and the rotation stage. The calibration of rotation stage facilitates registration of multi-view range images into a common coordinate system. We use the Tsai's calibration technique using a calibration pattern which contains several control points [98]. A focus calibration with respect to multiple distances of an object is also presented.

2.2 System Overview

The SVIS-2 vision system consists of five components. A digital still camera, a translation stage to implement a parallel stereo geometry, a rotation stage to change view direction of an object, a slide projector to project a random dot pattern on the object surface, and a personal computer system to control all motion components.

For photometric image acquisition, we use a Olympus C3030-Zoom digital camera which interfaces with a host computer via an Universal Serial Bus (USB). We have written a Windows-based program to communicate the camera with the host

computer using an Active-X control. Most properties and functions of the camera can be controlled by the host computer, which include focus, zoom, aperture, picture resolution, etc. Focus distance of the camera can be controlled from 200 mm to ∞ distance, and focal length can be controlled from 6.5 mm (Wide mode) to 19.35mm (Tele mode). For stereo image acquisition, focus is set to the best focus position for a given object distance and zoom is set to the tele mode of the lens.

The translation stage moves the camera in the horizontal direction along the baseline to implement a parallel stereo geometry. We assume that the stereo baseline is parallel to the x axis of the camera coordinate system. The rotation stage rotates the object to change viewing directions from the camera to acquire multiple stereo pictures around the object. Both the translation and the rotation stages have stepper motors, which are controlled by the host computer through a stepper motor controller. The rotation stage rotates 360° in 2400 motor steps and the translation stage horizontally moves 76 mm in 600 steps. The accuracies of the movements in the translation stage and the rotation stage are 0.127 mm and 0.1 degree respectively per motor step.

In order to introduce contrast on the surface of a low-contrast object, we use a slide projector to project a random dot pattern onto it. A slide film with black and white random dots pattern was taken from a computer screen with 1024×768 resolution. The resulting contrast facilitates the use of stereo matching. We named the vision system as SVIS-2 (Stony Brook Vision System-2). Figure 2.1 shows a diagram of SVIS-2 and Figure 2.2 shows a picture of the system.

2.3 Vision System Calibration

2.3.1 Focus Calibration

Focus changes with respect to the distance of an object. Calibration of the focus lens is finding the best lens position which focuses the object most sharply. Later in this chapter, we calibrate the rotation stage for multi-view modeling. An object is placed on the rotation stage and it is necessary to sharply focus the object to obtain high-quality images of the object from multiple viewing directions.

To find the exact focus step position of the lens with respect to the object distance, a calibration pattern is set up in front of the camera, and its distance is changed from 300 mm to 1010 mm with 50 mm intervals. The calibration pattern has high contrast textures so that a focus-measuring method can easily measure the level of focusing on the target. At each distance of the calibration pattern, we take several pictures of it by changing focus step number, and find the lens position corresponding to the maximum focus measure. Measuring how much the object is focused on the

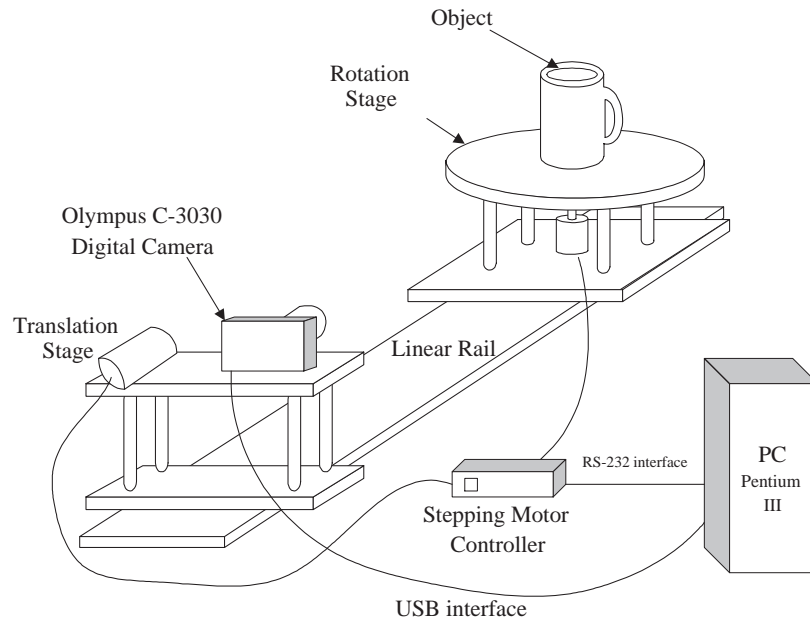


Figure 2.1: Diagram of SVIS-2 vision system

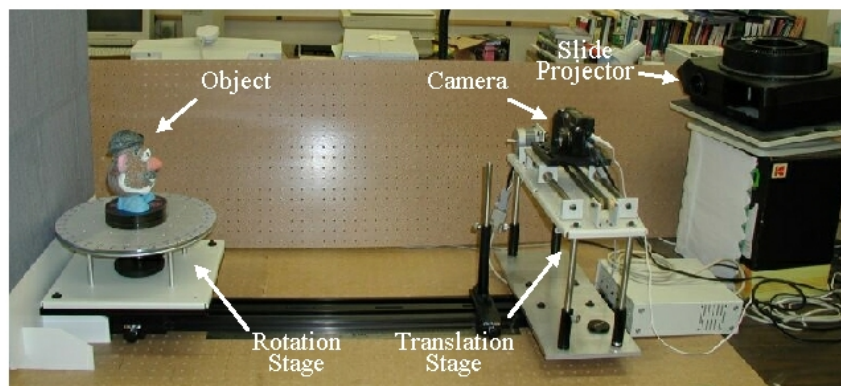


Figure 2.2: Picture of SVIS-2 vision system

image plane is done by computing a focus measure. The sum of the energy of the Laplacian filtered image within 16×16 blocks is used as the focus measure [90]. Step number of best focus with respect to object distance is plotted in Figure 2.3. The horizontal axis of the figure is focus step number of the camera, and the vertical axis is the corresponding focusing distance. We put the rotation stage about 0.8 m from

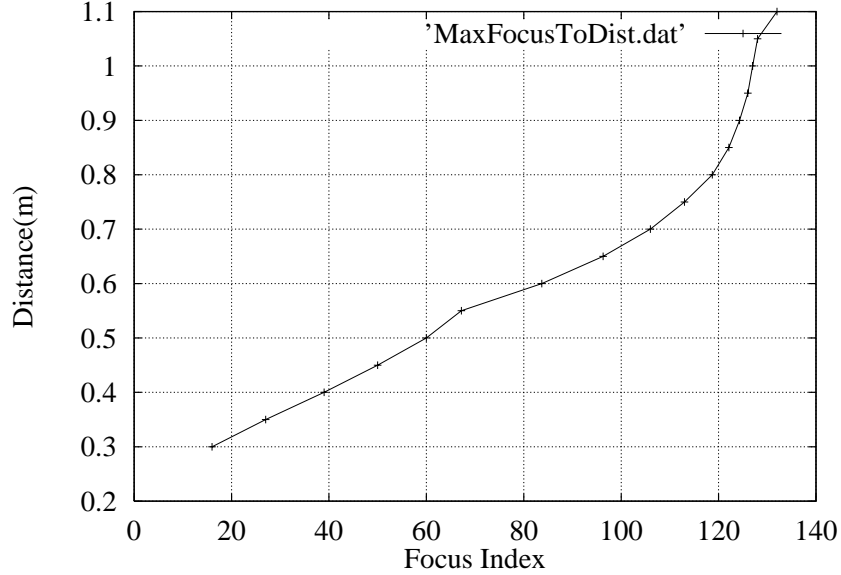


Figure 2.3: Plot of object distance vs. focus step of C-3030 digital camera

the camera and set the focus of the camera based on the calibration table. According to Figure 2.3, focus step number 120 is corresponds to about 0.8 m distance from the camera.

2.3.2 Stereo Camera Calibration

A parallel stereo geometry is used in SVIS-2. A digital camera is installed on the translation stage to implement the parallel stereo. Calibration of the stereo camera is done by Tsai's algorithm [98]. Consider a point \mathbf{P} in Figure 2.4 in 3D space, its representations \mathbf{P}_l and \mathbf{P}_r in the left and the right camera coordinate systems. Then the transformation between two coordinate systems is

$$\mathbf{P}_r = \mathbf{R}_l^r (\mathbf{P}_l - \mathbf{t}_l^r). \quad (2.1)$$

Where, \mathbf{R}_l^r and \mathbf{t}_l^r are rotation and translation matrices from the left to the right camera coordinate system, respectively. We assume that the translation stage moves

the camera in horizontal direction— x axis in camera coordinate system. Therefore, there is little error on epipolar line so that we can search stereo correspondence on epipolar line in horizontal direction. When there is an error in vertical direction— y axis in camera coordinate, we can manually adjust the level of the translation stage to minimize it. Figure 2.4 shows the geometry of the stereo camera of SVIS-2 system. The baseline of the stereo camera B is 76 mm, it corresponds to 600 steps of the stepper motor on the translation stage, and the focal length of the camera f is 19.35 mm, at tele mode of the lens. The focal length was calibrated using Tsai’s non-coplanar camera calibration technique.

Consider again a 3D point \mathbf{P} and its projection point on the right CCD plane, \mathbf{p}_r as shown in Figure 2.4. If we find another point \mathbf{p}_l in the left image plane, which is the corresponding point of \mathbf{p}_r computed by a stereo matching algorithm, we can get the vector \mathbf{P} in 3D space,

$$\mathbf{P} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{Bf}{x_{pl} - x_{pr}} \\ \frac{zx_{pr}}{f} \\ \frac{zy_{pr}}{f} \end{pmatrix} \quad (2.2)$$

and x_{pl} and x_{pr} are the x coordinates of the points \mathbf{p}_l and \mathbf{p}_r respectively. Here, we assign the right camera position as the reference of the camera coordinate system.

2.3.3 Rotation Stage Calibration

Rotation stage calibration is important for registering multiple range images. This calibration gives transformation parameters between the camera coordinate system to the rotation coordinate system. These parameters are later used to register all partial 3D shapes into a common coordinate system and to integrate them. An accurate calibration of the rotation stage gives better performance for registration and integration of multiple range images.

A calibration pattern which has several control points is used for calibrating the stage. Since several stereo images of an object are taken at every θ degree interval around an object, two consecutive images of calibration pattern are also taken with θ angle difference. By estimating calibration parameters which register two sets of 3D control points as close as possible, we can also register the partial shapes of the object into a common coordinate system.

Calibration for each view coordinate system is done using Tsai’s algorithm. But we modified the algorithm to fix the focal length of the camera to get more accurate parameters. A checkerboard calibration pattern is placed on the rotation stage. Two

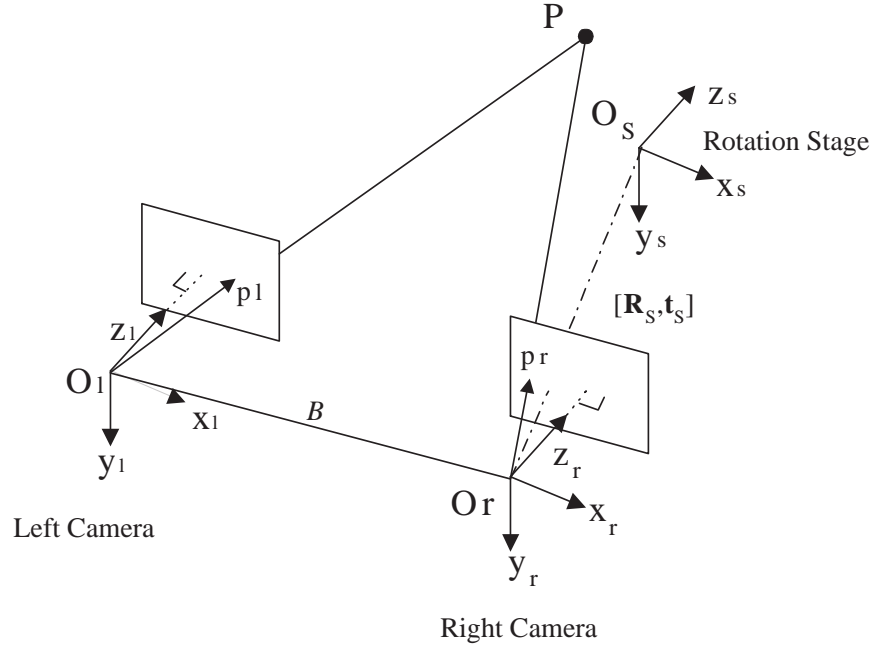


Figure 2.4: Parallel stereo geometry of SVIS-2 system

sets of stereo images are taken with θ degree angle difference. Let \mathcal{V}_0 and \mathcal{V}_1 denote the camera coordinate systems of the two view directions and \mathcal{V}_w denote the world coordinate system. Let a control point in \mathcal{V}_k be \mathbf{P}_k , where $k = 0$ and 1, and \mathbf{P}_w be the same point represented by the world coordinate system. Then transformations \mathbf{T}_w^k from the world coordinate system to each camera coordinate systems are

$$\begin{aligned}
 \mathbf{P}_0 &= \mathbf{T}_w^0 \mathbf{P}_w \\
 \mathbf{P}_1 &= \mathbf{T}_w^1 \mathbf{P}_w, \\
 \text{and} \\
 \mathbf{P}_0 &= \mathbf{T}_w^0 (\mathbf{T}_w^1)^{-1} \mathbf{P}_1 \\
 &= \mathbf{R}_1^0 \mathbf{P}_1 + \mathbf{t}_1^0.
 \end{aligned} \tag{2.3}$$

where, \mathbf{R}_1^0 and \mathbf{t}_1^0 are rotation and translation matrices between two coordinate systems as shown in Figure 2.5.

In order to register all partial shapes to a common view coordinate system, it is necessary to find the transformation between the common view coordinate system—view 0 (\mathcal{V}_0) in this dissertation—and the rotation stage coordinate system. Let \mathbf{R}_s and

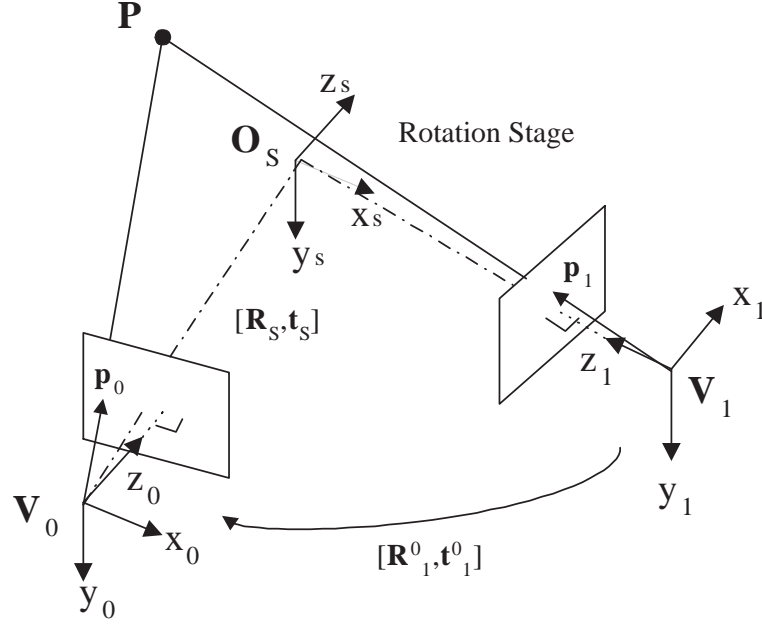


Figure 2.5: Multi-view geometry of SVIS-2 system

\mathbf{t}_s be the rotation and the translation matrices between the rotation stage coordinate system and the camera coordinate system. The transformation between two view directions is also expressed as

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{R}_\theta \mathbf{R}_s (\mathbf{P}_1 - \mathbf{t}_s) + \mathbf{t}_s. \\ &= \mathbf{R}_\theta \mathbf{R}_s \mathbf{P}_1 + \mathbf{t}_s (\mathbf{I} - \mathbf{R}_\theta \mathbf{R}_s). \end{aligned} \quad (2.4)$$

From equations (2.3) and (2.4),

$$\begin{aligned} \mathbf{R}_s &= \mathbf{R}_\theta^{-1} \mathbf{R}_1^0, \\ \mathbf{t}_s &= (\mathbf{I} - (\mathbf{R}_1^0)^{-1}) \mathbf{t}_1^0. \end{aligned}$$

2.3.4 Calibration Test

A picture of the calibration pattern with 48 control points is shown in Figure 2.6. From one viewing direction, a pair of stereo image of the calibration pattern is taken and the 3D positions of all control points $\{\mathbf{P}_0\}$ are obtained. After rotating the pattern, another set of control points $\{\mathbf{P}_1\}$ are obtained again in the same way.

We set the rotation angle $\theta = 90^\circ$. Parameters of the rotation \mathbf{R}_1^0 and the translation \mathbf{t}_1^0 are computed by Tsai's algorithm,

$$\begin{aligned} \mathbf{R}_1^0 &= \begin{pmatrix} 0.0144 & 0.0022 & -0.9998 \\ -0.0163 & 0.9998 & 0.0020 \\ 0.9997 & 0.0163 & 0.0145 \end{pmatrix} \\ \mathbf{t}_1^0 &= (793.37 \quad -2.862 \quad 856.56)^T. \end{aligned} \quad (2.5)$$

And, the resulting calibration parameters for the system is

$$\begin{aligned} \mathbf{R}_S &= \begin{pmatrix} 0.9980 & 0.01966 & -0.0586 \\ -0.0188 & 0.9997 & 0.0147 \\ 0.0588 & -0.0136 & 0.9980 \end{pmatrix} \\ \mathbf{t}_S &= (-38.19 \quad 0.0 \quad 833.38)^T. \end{aligned} \quad (2.6)$$

Figure 2.7 shows an initial calibration result. One set of control points is registered to the coordinate of the other point set. Table 2.1 shows errors in calibration.

Table 2.1: Rotation calibration error analysis

No. of control points	48
Max. distance error	3.60mm
Average distance error	1.90mm

2.4 Partial Shapes Acquisition

This section describes the reconstruction of 3D partial shapes for a single viewing direction. A 3D partial shape of an object is reconstructed from a stereo matching technique. A multi-resolution stereo matching technique based on Sum of Squared Difference (SSD) algorithm is applied to a pair of stereo pictures.

2.4.1 Multi-resolution Stereo Matching

We employ a multi-resolution stereo matching algorithm using a Gaussian Pyramid [13]. Multi-resolution approach in stereo matching gives many advantages. It

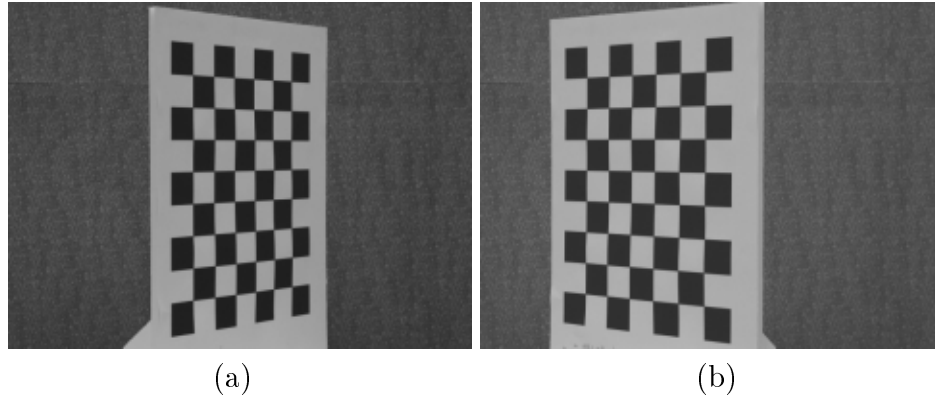


Figure 2.6: Calibration patterns.(a) 0 degree (b) 90 degree

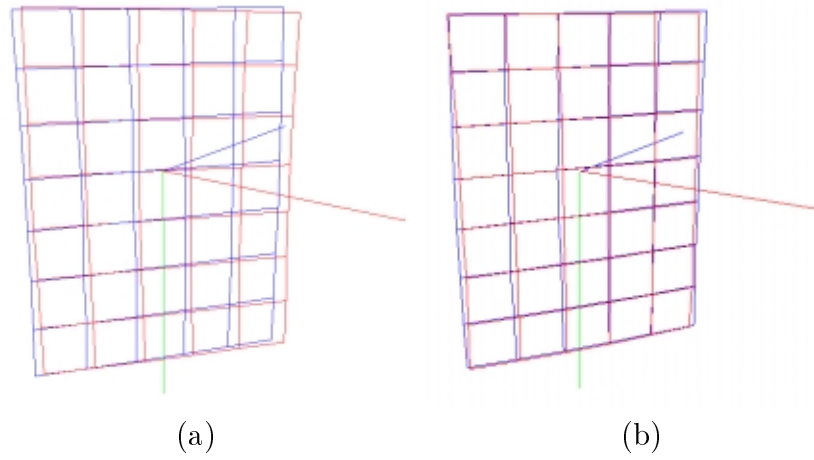


Figure 2.7: Result of registration of two control point sets (a) Before calibration (b) After calibration

can restrict search range from the result of matching in lower resolution image and thus reduce computation time for searching stereo correspondences.

A Gaussian pyramid for an image I is a sequence of copies of I , where each successive copy has half the resolution and sample rate. The levels of a Gaussian pyramid for a given image I is calculated as

$$\begin{aligned} g_k(i, j) &= \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) g_{k-1}(2i + m, 2j + n) \\ g_0(i, j) &= I(i, j). \end{aligned} \quad (2.7)$$

where $w(m, n)$ is a 5×5 size of Gaussian kernel. Because this kernel is separable, we use one-dimensional Gaussian kernel $w(m)$ whose length is 5. The weights of Gaussian kernel are

$$\begin{aligned} w(0) &= 0.4, \\ w(1) &= w(-1) = 0.25, \\ w(2) &= w(-2) = 0.05. \end{aligned}$$

Three levels of Gaussian pyramid are used from level 0 to level 2. Image of level 0 corresponds to the original image and level 2 corresponds to the smallest image. Original image size is 1280×960 and the image size at level 2 is 320×240 . We use a variable size of matching block for stereo matching. It is $(15-2k) \times (15-2k)$ for the k th pyramid level. A sample picture, a segmented image, and its range image, are displayed in Figure 2.8. Normalized Gaussian low-pass filter is also applied to the range image to obtain sub-pixel accuracy.

Object's silhouettes in the stereo image are segmented by a *blue screen* technique. A binary morphological closing and opening operation is used to remove noise in image segmentation. The matching algorithm finds stereo correspondence only on the object areas in left and right images. Object's silhouettes are also used later for volume intersections in multi-view integration processing.

SSD based stereo matching is done at each level of the Gaussian pyramid from low resolution to high resolution. At the first level of the stereo matching, initial search range of stereo disparity SR_0 at level 0 is set to $[0, sr_0]$. Then, at the lowest level of pyramid for $k = 2$, initial stereo disparity SR_2 becomes $[0, sr_0/(2k)]$. At successive levels of pyramid, the result of stereo disparity at lower level decides the search range of the respective level. If the disparity at lower level is D_i , then the search range of current level SR_i is restricted within $[2 * D_i - 2, 2 * D_i + 2]$ so that the stereo matching algorithm can correct possible mismatches in previous level. When there is a pair of stereo image, $g_k^{(l)}$ and $g_k^{(r)}$ for left and right images, which are at

level k of the pyramid, $SSD(i,j)$ at image coordinate (i,j) is

$$SSD(i,j) = \sum_{k=-m}^m \sum_{l=-m}^m \{g_k^{(l)}(i,j) - g_k^{(r)}(i+k,j+l)\} \quad (2.8)$$

where, $2m + 1$ is the size of a matching block.



Figure 2.8: An example of object segmentation and stereo matching (a)An image of a stereo pair (b)Segmented object (c) Stereo matching result on segmented region

2.4.2 Experimental Results

A pair of stereo image of an object 'Monkey' is obtained with 1280×960 image size. The SSD-based stereo matching is done within a multi-sized matching block. Figure 2.9 (a) and (b) show a input pair of stereo image and Figure 2.9 (c) and (d) show the corresponding pair of segmented images of the object from the background. Figure 2.9 (e) shows results of 3D shape reconstruction from the multi-resolution stereo matching. Shapes are represented as point clouds, where each point represents a range data point on the object's surface.

2.5 Conclusions

The SVIS-2 system is developed to obtain range images of an object from multiple view points. We have calibrated the stereo camera and the turntable stages. Since the stereo camera has a parallel stereo geometry, we use a mechanical measurement for the baseline of the stereo camera. The focus of the camera is also calibrated by measuring focusing of the lens as a function of the distance of an object. The calibration parameters of the turntable will facilitate registration of the multiple views into a common coordinate system. Multi-view registration and integration will be presented in the Chapter 4 and 5.

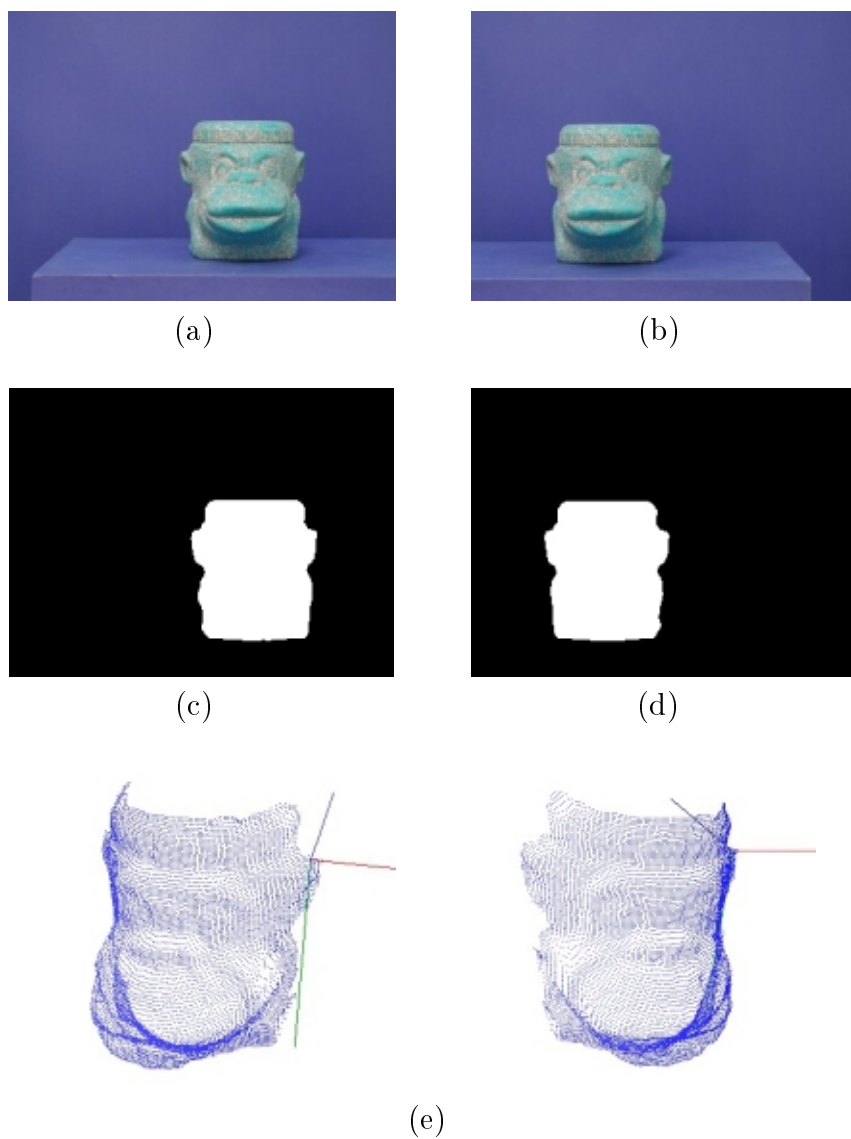


Figure 2.9: Experimental result of stereo matching: (a) Left image (b) Right image (c) Left segmentation (d) Right segmentation (e) Point clouds of the reconstructed 3D shape

Chapter 3

Vergence Stereo Vision System (SVIS-3)

This chapter presents another stereo vision system called SVIS-3 (Stony Brook Vision System-3). The configuration of SVIS-3 is similar to that of SVIS-2. However, the system adopts a toed-in stereo camera to obtain range images of an object more effectively. We calibrate the stereo camera to rectify stereo image pairs. In contrast to the SVIS-2 system, the stereo camera in SVIS-3 can be placed on the table top arbitrarily to accommodate to the size of an object. Therefore, we precisely calibrate a rotation stage with respect to the stereo camera so that the system can reconstruct a complete 3D model from an arbitrary view point.

3.1 Introduction

Calibration and rectification techniques of another stereo vision system (SVIS-3) are presented. The vision system consists of a digital stereo camera, a turning table, and a personal computer. We calibrate the stereo camera and the turntable, rectify stereo images, acquire range images of an object from multiple view points, and reconstruct a complete 3D model. Stereo rectification determines a transformation of each image plane such that pairs of conjugate epipolar lines become parallel to the horizontal image axes [28, 32]. The rectified images can be considered as new images acquired by a new parallel stereo camera, obtained by rotating the original camera. An important advantage of rectification is for computing stereo correspondences. Because each pair of epipolar line is parallel to the horizontal image axis, finding correspondence between two stereo images are done on the same horizontal image axis.

In order to rectify stereo images, we calibrate the stereo camera. Some useful calibration techniques for vision systems are already available, which determine external and internal calibration parameters. However, we do not use the techniques, but obtain only projection matrices of the left and the right cameras for rectification. Projection matrix is a homogeneous transform matrix which maps a 3D point in space

into a 2D point in the image plane of a camera. Estimating the projection matrix is the solution of simple and overdetermined linear system equations, and we solve it from Singular Value Decomposition (SVD) of the linear system.

Given calibration parameters, we rectify stereo images by using a rectification technique investigated by A. Fusiello, etc [28]. Their algorithm determines a transformation of each stereo image plane by assuming that a new stereo camera is obtained by rotating an original stereo camera. We adopt this technique because it is simple and accurate.

3.2 Stereo Camera Calibration

In this section, we describe a calibration technique of a vision camera. It can be considered as an estimation of a projective transformation matrix from the world coordinate system to the camera's image coordinate system. If we have the coordinates $(X_i^w, Y_i^w, Z_i^w, 1)$ of a 3D point in space and the coordinates $(x_c, y_c, 1)$ of its projection on the 2D image plane, a 3×4 projection matrix \mathbf{M} can be written according to the equation

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \mathbf{M} \begin{bmatrix} X_i^w \\ Y_i^w \\ Z_i^w \\ 1 \end{bmatrix}, \quad (3.1)$$

with

$$\begin{aligned} x_c &= \frac{u_i}{w_i} = \frac{m_{11}X_i^w + m_{12}Y_i^w + m_{13}Z_i^w + m_{14}}{m_{31}X_i^w + m_{32}Y_i^w + m_{33}Z_i^w + m_{34}} \\ y_c &= \frac{v_i}{w_i} = \frac{m_{21}X_i^w + m_{22}Y_i^w + m_{23}Z_i^w + m_{24}}{m_{31}X_i^w + m_{32}Y_i^w + m_{33}Z_i^w + m_{34}} \end{aligned} \quad (3.2)$$

The matrix \mathbf{M} is defined up to an arbitrary scale factor and has only 11 independent entries. Therefore we need at least 6 world image points and their matching points in the image plane. If we use a calibration pattern, for example a checkerboard pattern, we have more correspondences and \mathbf{M} can be estimated through least squares techniques. If we assume we are given N matches for the homogeneous linear system like Equation (3.2), we have

$$\mathbf{A}\mathbf{m} = 0, \quad (3.3)$$

with

$$\mathbf{A} = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1 X_1 & -x_1 Y_1 & -x_1 Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1 X_1 & -y_1 Y_1 & -y_1 Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2 X_2 & -x_2 Y_2 & -x_2 Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2 X_2 & -y_2 Y_2 & -y_2 Z_2 & -y_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_N X_N & -x_N Y_N & -x_N Z_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_N X_N & -y_N Y_N & -y_N Z_N & -y_N \end{bmatrix}$$

and $\mathbf{m} = [m_{11}, m_{12}, \dots, m_{33}, m_{34}]^T$.

Since \mathbf{A} has rank 11, the vector \mathbf{m} (or \mathbf{M}) can be recovered from SVD related techniques as the column of \mathbf{V} corresponding to the smallest singular value of \mathbf{A} , with $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$. For more information see reference [97].

3.3 Stereo Rectification

Suppose we have two cameras whose optical centers are C_1 and C_2 , respectively. Let a 3D point in space be W and its projection to the left camera's image plane be p_1 . Then we can find its correspondence p_2 on the right image plane on the epipolar line u_2 , which is the intersection of the right image plane and the epipolar plane of a triangle WC_1C_2 . If two image planes are collinear, the epipolar line u_2 will be collinear with the epipolar line u_1 on the left image plane. However most stereo cameras, which have a toed-in angle between left and right cameras, their conjugate epipolar lines are not collinear.

Stereo rectification determines a transformation of each image plane such that pairs of conjugate epipolar lines become parallel to the horizontal image axes. This gives an advantage for computing stereo correspondences. Because each pair of epipolar line is parallel to the horizontal image axis, finding correspondence between two stereo images are constrained on the same horizontal image axis.

3.3.1 Pinhole Camera Model

A pinhole camera is modeled by its optical center C and its image plane \mathcal{R} . A 3D point W is projected into an image point P , which is an intersection of \mathcal{R} with the line containing C and W . Let $\mathbf{W} = [x \ y \ z]^T$ be the coordinates of W in the world coordinate system, $\mathbf{p} = [u \ v]^T$ the coordinates of P in the image plane (CCDs), and $\mathbf{p}' = [u' \ v']^T$ the coordinates of P in the picture plane (pixels). The mapping from 3D coordinates to 2D coordinates is the *perspective projection*, which is represented by a linear transformation in *homogeneous coordinates*. Let $\tilde{\mathbf{p}} = [u \ v \ 1]^T$, $\tilde{\mathbf{p}}' = [u' \ v' \ 1]^T$,

and $\tilde{\mathbf{W}} = [x \ y \ z \ 1]^T$ be the homogeneous coordinates of \mathbf{p} , \mathbf{p}' , and \mathbf{W} , respectively. Then the perspective transformation is given by the matrix $\tilde{\mathbf{M}}$:

$$\tilde{\mathbf{p}}' = \mathbf{K}\tilde{\mathbf{p}} \cong \mathbf{K}\tilde{\mathbf{M}}\tilde{\mathbf{W}}, \quad (3.4)$$

where \cong means equal up to a scale factor. The camera is therefore modeled by a scaling and translating matrix \mathbf{K} and its perspective transformation matrix (PPM) $\tilde{\mathbf{M}}$ which can be decomposed, using the QR factorization, into the product

$$\tilde{\mathbf{M}} = \mathbf{A}[\mathbf{R}|\mathbf{t}]. \quad (3.5)$$

The matrices \mathbf{K} and \mathbf{A} depend on the intrinsic parameters only, and have the following forms:

$$\mathbf{K} = \begin{bmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.6)$$

where, f_u, f_v are the focal lengths in the horizontal and the vertical directions, k_u, k_v are the scaling factors from the CCD plane to the picture plane, (u_0, v_0) are the coordinates of the principal point in picture plane, (u_i, v_i) are the coordinates of the offset of the principal point in image plane, and γ is a skew factor. The camera position and orientation (extrinsic parameters) are represented by a 3×3 rotation matrix \mathbf{R} and a translation vector \mathbf{t} , representing a rigid transformation that brings the camera coordinate system onto the world coordinate system.

The PPM can be also written as

$$\tilde{\mathbf{M}} = \left[\begin{array}{c|c} \mathbf{q}_1^T & q_{14} \\ \mathbf{q}_2^T & q_{24} \\ \mathbf{q}_3^T & q_{34} \end{array} \right] = [\mathbf{Q}|\tilde{\mathbf{q}}]. \quad (3.7)$$

The focal plane is the plane parallel to the image plane that contains the optical center C , and the projection of \mathbf{W} to the plane is 0. Therefore, the coordinates \mathbf{C} of C is given by

$$\mathbf{C} = -\mathbf{Q}^{-1}\tilde{\mathbf{q}}. \quad (3.8)$$

Therefore, $\tilde{\mathbf{M}}$ can be written as

$$\tilde{\mathbf{M}} = [\mathbf{Q} | -\mathbf{Q}\mathbf{C}]. \quad (3.9)$$

The optical ray associated to an image point P is the line PC , that means the set of 3D points $\mathbf{W} : \tilde{\mathbf{p}} \cong \tilde{\mathbf{M}}\tilde{\mathbf{W}}$. In parametric form:

$$\mathbf{W} = \mathbf{C} + \lambda\mathbf{Q}^{-1}\tilde{\mathbf{p}}, \lambda \in \mathbb{R}. \quad (3.10)$$

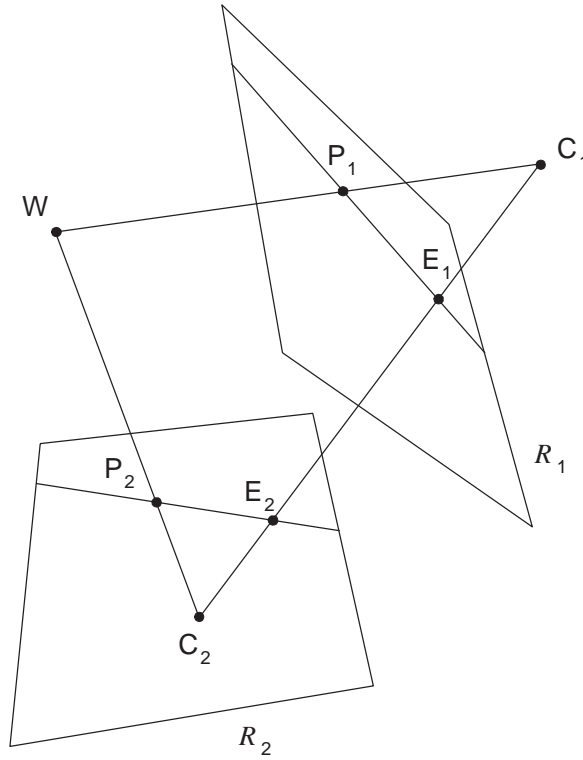


Figure 3.1: Epipolar geometry of a vergence stereo camera

3.3.2 Epipolar Geometry

Let us consider a stereo vision system composed by two pinhole cameras as shown in Figure 3.1. Let C_1 and C_2 be the optical centers of the left and the right cameras, respectively. A 3D point W is projected onto both image planes, to points P_1 and P_2 , which are corresponding points between two images. Given a point P_1 in the left image plane, its corresponding point in the right image plane is constrained to lie on a line called the epipolar line. All the epipolar lines in one image planes pass through a common point, E_1 and E_2 respectively, and it is called the epipole.

A very special case is when both epipoles are at infinity. It happens when the line C_1C_2 (the baseline) is constrained in both focal planes, where the image planes are parallel to the baseline. Therefore, any stereo images can be transformed so that epipolar lines are parallel and horizontal in each image. This procedure is called *rectification*.

3.4 Rectification of Camera Matrices

We assume that the stereo system is calibrated, that means the original PPMs $\tilde{\mathbf{M}}_{o1}$ and $\tilde{\mathbf{M}}_{o2}$ are known for both cameras. Rectification estimates two new PPMs $\tilde{\mathbf{M}}_{n1}$ and $\tilde{\mathbf{M}}_{n2}$ by rotating the old matrices around their optical centers until focal planes become coplanar.

In order to have horizontal epipolar lines, the baseline must be parallel to the new X axis of both cameras. In addition, corresponding points must have the same vertical coordinate (Y axis). Consequently, the positions of new optical centers are the same as what in the old cameras, the new matrices differ from the old ones by suitable rotations, and intrinsic parameters are the same for both cameras. Therefore, the new PPMs will differ only in their optical centers.

Let us write the new PPMs in terms of their factorization,

$$\begin{aligned}\tilde{\mathbf{M}}_{n1} &= \mathbf{A}[\mathbf{R} \mid -\mathbf{R} \mathbf{C}_1], \\ \tilde{\mathbf{M}}_{n2} &= \mathbf{A}[\mathbf{R} \mid -\mathbf{R} \mathbf{C}_2].\end{aligned}\tag{3.11}$$

The intrinsic parameters matrix \mathbf{A} is the same for both new PPMs and computed arbitrary in this report as, $A = \frac{A_1+A_2}{2}$. The rotation matrix \mathbf{R} is also the same for both PPMs. It can be specified in terms of its row vectors

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix}\tag{3.12}$$

, which are the X , Y , and Z axes of the camera coordinate system.

According to the previous descriptions, each axis is computed as:

1. The new X axis is parallel to the baseline: $\mathbf{r}_1 = \frac{(\mathbf{c}_1 - \mathbf{c}_2)}{\|\mathbf{c}_1 - \mathbf{c}_2\|}$.
2. The new Y axis is orthogonal to X and to \mathbf{k} : $\mathbf{r}_2 = \mathbf{k} \wedge \mathbf{r}_1$.
3. The new Z axis is orthogonal to XY plane: $\mathbf{r}_3 = \mathbf{r}_1 \wedge \mathbf{r}_2$.

In number 2, \mathbf{k} is an arbitrary unit vector, but we take it equal to the Z unit vector of the old left camera coordinate system. Then, we constrain the new Y axis to be orthogonal to both the new X and the old left Z .

In order to rectify the left and the right image, we need to compute transformations mapping of the image plane $\tilde{\mathbf{M}}_{oi} = [\mathbf{Q}_{oi} \mid \tilde{\mathbf{q}}_{oi}]$ onto the image plane $\tilde{\mathbf{M}}_{ni} = [\mathbf{Q}_{ni} \mid \tilde{\mathbf{q}}_{ni}]$.

For any 3D point \mathbf{W} , we can write

$$\begin{aligned}\tilde{\mathbf{p}}_{oi} &\cong \tilde{\mathbf{M}}_{oi} \tilde{\mathbf{W}} \\ \tilde{\mathbf{p}}_{ni} &\cong \tilde{\mathbf{M}}_{ni} \tilde{\mathbf{W}}.\end{aligned}$$

According to Equation 3.10, the equations of the optical rays are the following:

$$\begin{aligned}\mathbf{W} &= \mathbf{C}_i + \lambda_o \mathbf{Q}_{oi}^{-1} \tilde{\mathbf{p}}_{oi}, \quad \lambda_o \in \mathbb{R} \\ \mathbf{W} &= \mathbf{C}_i + \lambda_n \mathbf{Q}_{ni}^{-1} \tilde{\mathbf{p}}_{ni}, \quad \lambda_n \in \mathbb{R}\end{aligned}$$

hence,

$$\begin{aligned}\tilde{\mathbf{p}}_{ni} &= \lambda \mathbf{Q}_{ni} \mathbf{Q}_{oi}^{-1} \tilde{\mathbf{p}}_{oi}, \quad \lambda \in \mathbb{R}, \\ \tilde{\mathbf{p}}_{ni} &= \lambda \mathbf{T}_i \tilde{\mathbf{p}}_{oi}.\end{aligned}\tag{3.13}$$

The transformation $\mathbf{T}_i = \mathbf{Q}_{ni} \mathbf{Q}_{oi}^{-1}$ is then applied to the original stereo images to produce rectified images. Because the pixels of the rectified image correspond to non-integer positions on the original image planes, we compute new pixel positions by using bilinear interpolation.

3.5 Experimental Results

3.5.1 Stereo Camera

The stereo camera consists of two identical digital still cameras, which is *Olympus C-3020 Zoom*. The two cameras are installed on a vertical stereo mount. We fix the cameras on the mount with an arbitrary toed-in angle so that the optical axes of the cameras converge at the distance about 800 mm from the camera. Two digital cameras are connected to a personal computer, Intel Pentium 1.8GHz, through two USB communication ports. We use an Active-X driver software, called *CRye Control*, to communicate and control the stereo camera. Figure 3.2 shows a picture of the stereo camera.

3.5.2 Camera Calibration

We use a checkerboard pattern to calibrate the stereo camera. It has two planes which are parallel to the XY and the YZ planes of the world coordinate systems. On each plane there are 48 control points which compose a set of 3D world coordinate points. Figure 3.3 shows a diagram of the calibration pattern. The specifications of the pattern is as follows:

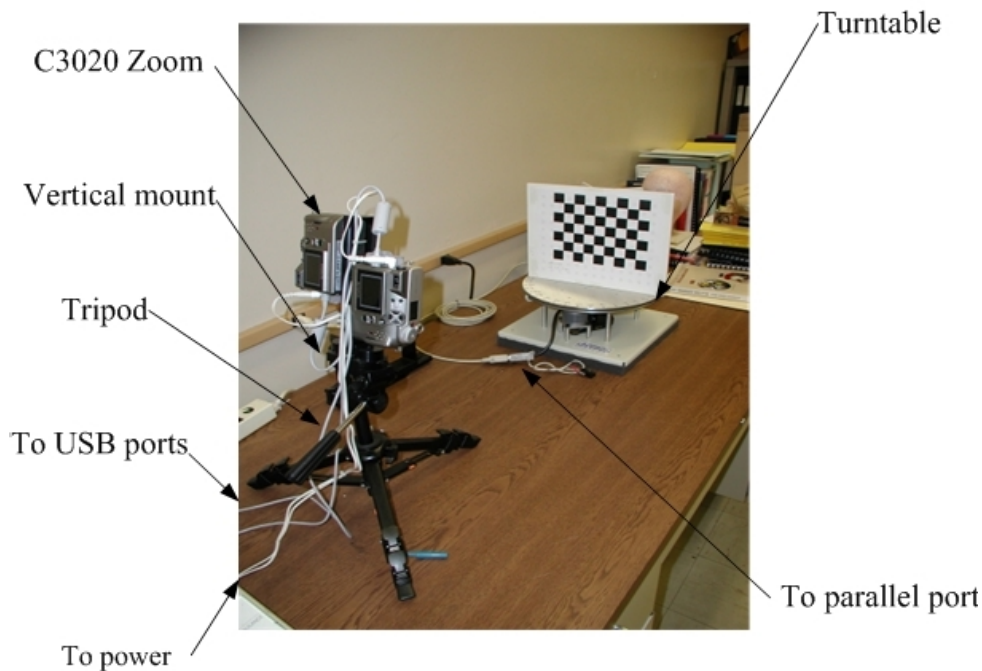


Figure 3.2: Picture of SVIS-3 stereo vision system

- The size of a black square in the x direction, $x_d = 17.2 \text{ mm}$.
- The size of a black square in the y direction, $y_d = 17.18 \text{ mm}$.
- Offset to the YZ plane in the x direction from the origin, $x_f = 6 \text{ mm}$, which means $x = -6 \text{ mm}$ on the YZ plane.
- Offset to the edge of the right-most square on the XY plane in the z direction, $z_f = 11.5 \text{ mm}$.

Stereo pictures of the calibration pattern are shown in Figure 3.4(a) and 3.4(b), which are the left and the right image, respectively. Each images' pixel resolution is 960×1280 for (x, y) axes, but they are obtained originally with 1280×960 and rotated to the new resolution because the cameras are mounted vertically.

All control points on the calibration pattern are acquired by a corner detection algorithm. The world coordinates w_i and the image coordinates p_i of the inner 96 corner points on both calibration planes are used to compute the projective matrix of each camera. Projection matrix is computed as described in an earlier section. The

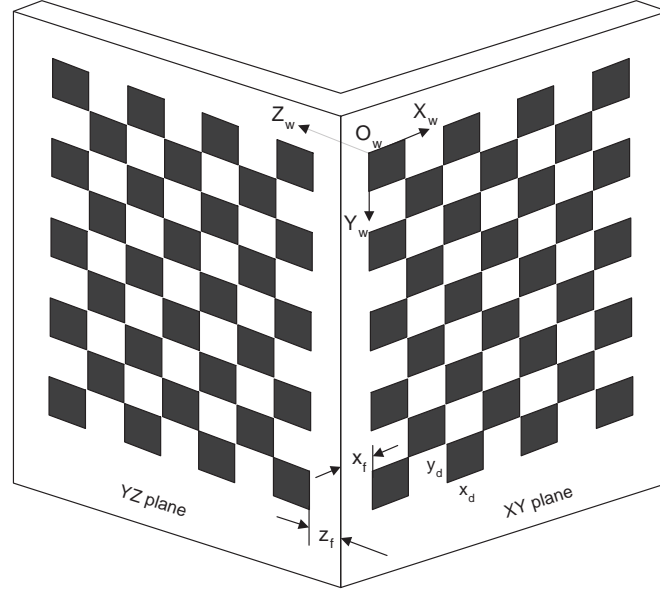


Figure 3.3: Checkerboard pattern for camera calibration

left and the right projection matrices, \mathbf{M}_{o1} and \mathbf{M}_{o2} , which bring a 3D world point to the corresponding image planes are

$$\mathbf{M}_{o1} = \begin{bmatrix} 1.4465 \cdot 10^{-2} & -4.2325 \cdot 10^{-4} & -1.4824 \cdot 10^{-2} & 5.0865 \cdot 10^{-1} \\ -2.9278 \cdot 10^{-3} & 2.0158 \cdot 10^{-2} & -3.4859 \cdot 10^{-3} & -1.5406 \cdot 10^0 \\ 1.2290 \cdot 10^{-3} & 3.7926 \cdot 10^{-4} & 1.2175 \cdot 10^{-3} & 1.0000 \cdot 10^0 \end{bmatrix} \quad (3.14)$$

$$\mathbf{M}_{o2} = \begin{bmatrix} 1.6169 \cdot 10^{-2} & 3.8523 \cdot 10^{-4} & -1.2728 \cdot 10^{-2} & 5.3614 \cdot 10^{-1} \\ -2.9701 \cdot 10^{-3} & 2.0061 \cdot 10^{-2} & -3.2199 \cdot 10^{-3} & -1.4531 \cdot 10^0 \\ 1.0727 \cdot 10^{-3} & 3.8318 \cdot 10^{-4} & 1.3537 \cdot 10^{-3} & 1.0000 \cdot 10^0 \end{bmatrix}. \quad (3.15)$$

Coordinates of the optical centers of the two cameras are also calibrated as

$$\mathbf{C}_1 = \begin{bmatrix} -423.81 \\ -50.45 \\ -377.78 \end{bmatrix}, \quad \mathbf{C}_2 = \begin{bmatrix} -370.69 \\ -51.52 \\ -430.36 \end{bmatrix}. \quad (3.16)$$

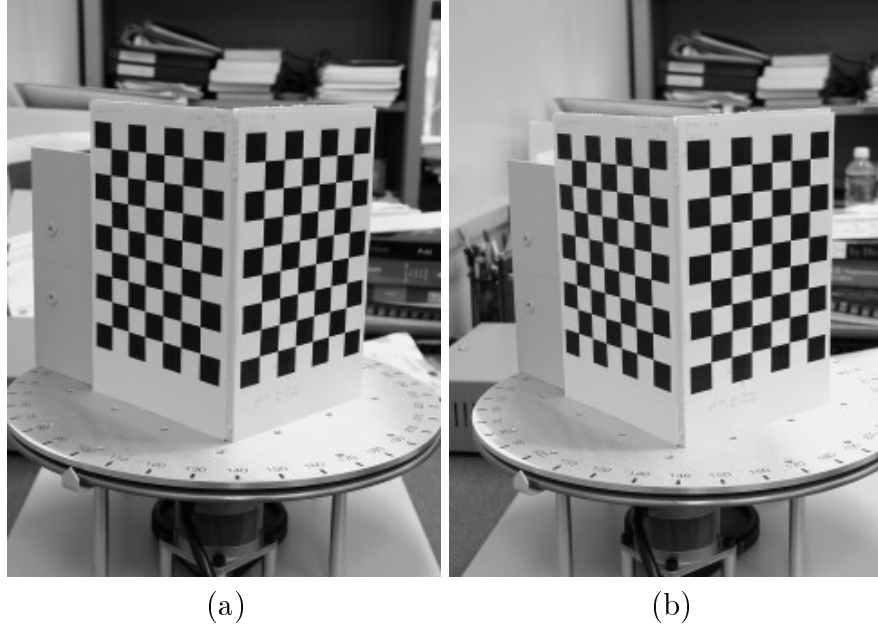


Figure 3.4: Stereo images of the checkerboard pattern: (a) Left image (b) Right image

3.5.3 Stereo Rectification

Given projection matrix \mathbf{M}_{oi} for the stereo camera, new projection matrix \mathbf{M}_{ni} is computed as described in the Section 3.4:

$$\mathbf{M}_{n1} = \begin{bmatrix} 1.4653 \cdot 10^{-2} & -3.0417 \cdot 10^{-4} & -1.4570 \cdot 10^{-2} & 6.9056 \cdot 10^{-1} \\ 2.8936 \cdot 10^{-3} & -2.0141 \cdot 10^{-2} & 3.3305 \cdot 10^{-3} & 1.4683 \cdot 10^0 \\ 1.2213 \cdot 10^{-3} & 3.7957 \cdot 10^{-4} & 1.2261 \cdot 10^{-3} & 1.0000 \cdot 10^0 \end{bmatrix}, \quad (3.17)$$

$$\mathbf{M}_{n2} = \begin{bmatrix} 1.4653 \cdot 10^{-2} & -3.0417 \cdot 10^{-4} & -1.4571 \cdot 10^{-2} & -8.5421 \cdot 10^{-1} \\ 2.8936 \cdot 10^{-3} & -2.0141 \cdot 10^{-2} & 3.3306 \cdot 10^{-3} & 1.4683 \cdot 10^0 \\ 1.2213 \cdot 10^{-3} & 3.7958 \cdot 10^{-4} & 1.226 \cdot 10^{-3} & 1.0000 \cdot 10^0 \end{bmatrix}. \quad (3.18)$$

Transformation matrix \mathbf{T}_i to rectify the stereo pair in the image planes is then estimated as $\mathbf{T}_i = \mathbf{Q}_{ni}\mathbf{Q}_{oi}^{-1}$. For a pixel \mathbf{p}_o in the original image plane and its homogeneous coordinates $\tilde{\mathbf{p}}_o$, a new pixel position $\tilde{\mathbf{p}}_n$ is estimated as

$$\tilde{\mathbf{p}}_n = \mathbf{T}_i \tilde{\mathbf{p}}_o. \quad (3.19)$$

The picture coordinates $\mathbf{p}'_n = (\mathbf{u}', \mathbf{v}')$ of the image point \mathbf{p}_n is then obtained by

multiplying the scaling and translating matrix \mathbf{K} to the image coordinates:

$$\tilde{\mathbf{p}}'_n = \begin{bmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{p}}_n. \quad (3.20)$$

However when we save a rectified image to a 2D array of picture frame, we need to consider a translation of the principal point. Otherwise, we may lose some portion of the image outside of the original picture frame. It is because an offset between the original principal point (u_{o0}, v_{o0}) and the new principal point (u_{n0}, v_{n0}) , which is due to the rotation of the optical axis of the camera. In order to translate the rectified image back into the picture frame, we compute the new principal point (u_{n0}, v_{n0}) by adding the offset to the old principal point. The offset of the principal points can be computed by mapping the origin of the retinal plane onto the new retinal plane:

$$\tilde{\mathbf{o}}_n = \mathbf{T} \begin{bmatrix} u_{o0} \\ v_{o0} \\ 1 \end{bmatrix}, \quad (3.21)$$

and the new retinal coordinates are

$$\tilde{\mathbf{p}}'_n = \mathbf{K}(\tilde{\mathbf{p}}_n - \tilde{\mathbf{o}}_n). \quad (3.22)$$

We consider the offset only in x direction because rectifying transformation rotate the image plane around y axis. Offsets of the principal points on the left and the right retinal planes are $(0.0742, 0)$ and $(-1.299, 0)$ respectively.

Figure 3.5 (a) and (b) show the rectified images of the calibration pattern. For each image, we compute a rectifying transformation matrix, transform a new pixel position to an old pixel position, and obtain the gray value of the new pixel position using bilinear interpolation. Rectified images are also shifted into the picture frame according to the offsets of principal points. However, we take into account the offsets when we compute the actual depth of a 3D object from stereo disparity.

3.6 3D Reconstruction

Using a rectified stereo image pair, we acquire a range image of a 3D object using a multi-resolution stereo matching technique. The stereo matching technique uses the Gaussian pyramid to obtain a multi-resolution image pyramid. A normalized correlation technique is used to find the stereo correspondences. Figure 3.6 (a) shows a pair of rectified stereo image and 3.6(b) and (c) show the results of stereo matching. Depth to a 3D point is measured using the disparity between projected points in stereo images. The offsets in the picture plane are also taken into account for the depth measure. We can use two methods to compute depth from stereo disparity.

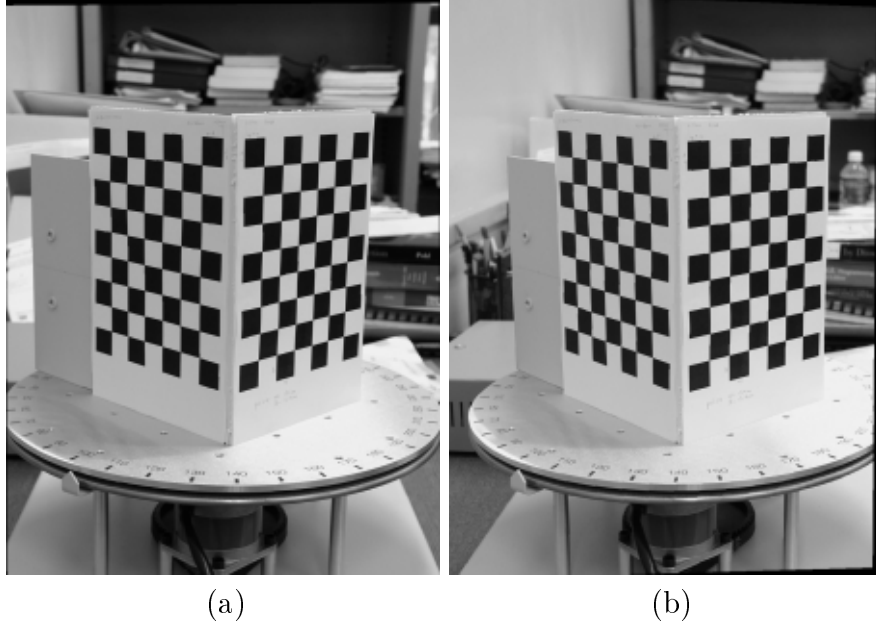


Figure 3.5: Rectified stereo images (a) left (b) right

3.6.1 Simple Triangulation

Because stereo images are rectified already, depth computation uses a simple equation for a parallel stereo camera. When there is a disparity d'_u in x direction of the picture plane, the depth D to a 3D point from the stereo camera is

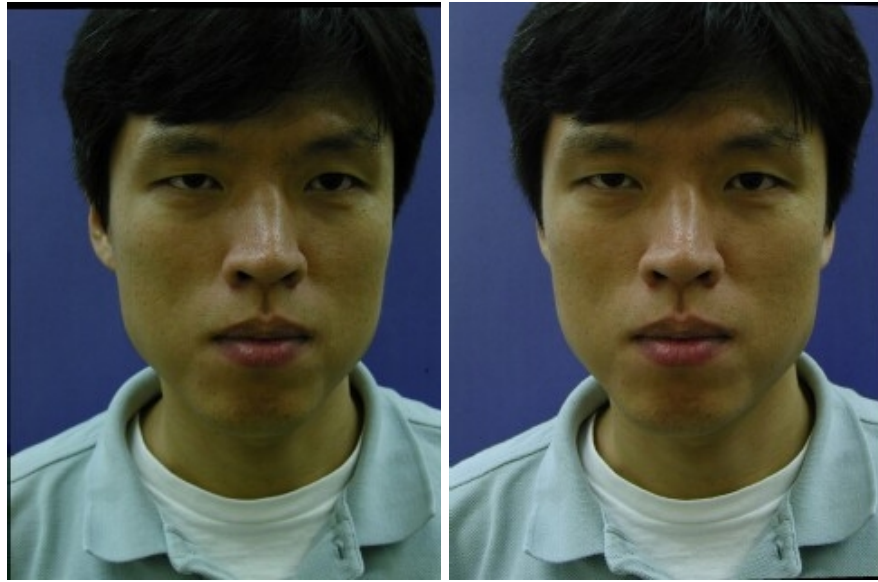
$$D = \frac{f \cdot B}{d'_u/k_u + (\tilde{u}_{n1} - \tilde{u}_{n2})}, \quad (3.23)$$

where $B = \|\mathbf{C}_1 - \mathbf{C}_2\|$ is the length of the baseline of the stereo camera and it is 74.25 mm in our system. For the focal length f of the camera we average the calibration results for both left and right focal lengths f_u, f_v , and it is 11.65 mm.

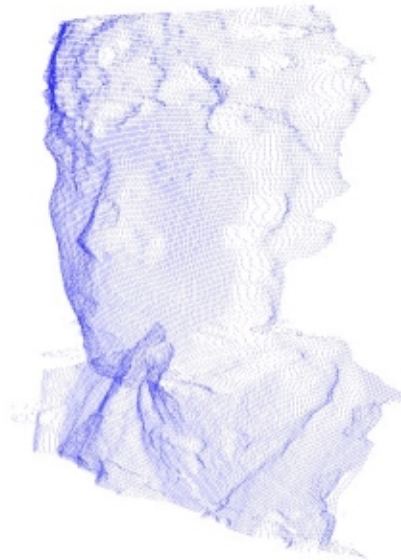
3.6.2 Solution of a Linear Equation

The range for a pair of conjugate points is also reconstructed by using Equation 3.1. Given two conjugate points $\tilde{\mathbf{p}}_1 = (u_1, v_1, 1)$ and $\tilde{\mathbf{p}}_2 = (u_2, v_2, 1)$ and the two projection matrices $\tilde{\mathbf{M}}_{n1}$ and $\tilde{\mathbf{M}}_{n2}$, we can write a overconstrained linear system

$$\mathbf{A}\mathbf{W} = \mathbf{y}, \quad (3.24)$$



(a)



(b)



(c)

Figure 3.6: Experimental results of stereo matching: (a) Rectified left and right stereo images of an human face (b) Point clouds of the measured 3D shape (c) Texture mapped result

where

$$\mathbf{A} = \begin{bmatrix} (\mathbf{a}_1 - u_1 \mathbf{a}_3)^T \\ (\mathbf{a}_2 - v_1 \mathbf{a}_3)^T \\ (\mathbf{b}_1 - u_2 \mathbf{b}_3)^T \\ (\mathbf{b}_2 - v_2 \mathbf{b}_3)^T \end{bmatrix} \mathbf{y} = \begin{bmatrix} -a_{14} + u_1 a_{34} \\ -a_{24} + v_1 a_{34} \\ -b_{14} + u_2 b_{34} \\ -b_{24} + v_2 b_{34} \end{bmatrix}. \quad (3.25)$$

Then \mathbf{w} gives the position of the 3D point projected to the conjugate points. Column vectors \mathbf{a}_i and \mathbf{b}_i are entry vectors of the new left and the right projection matrices, respectively. Sometimes, the rectified image can be reflected along the vertical or the horizontal axis. This can be detected by checking the ordering between the two diagonal corners of the image. If a reflection occurs, the image should be reflected back to keep the original ordering.

The 3D point \mathbf{w} is represented with respect to the world coordinate system. Therefore we need to transform the point in order to represent it with respect to the reference camera coordinates. Suppose we let the right camera's coordinate system be the reference. Then we can transform the point by simply using the external calibration parameters $[\mathbf{R}|\mathbf{t}]$ of the right cameras.

However, two transformations can be considered, one is to the old right camera's coordinates before rectification, and the other is to the new right camera's coordinates after rectification. By taking into account a multi-view registration, which we will present in the next section, we transform the point to the old right camera's coordinate system by

$$\mathbf{P}_w = [\mathbf{R}_{o2}|\mathbf{t}] \mathbf{W}, \quad (3.26)$$

where $[\mathbf{R}_{o2}|\mathbf{t}]$ is the old external calibration parameters of the right camera. Because it needs to calibrate the turntable for registration of multi-view range images, we represent all range images with respect to the old right camera's coordinate system.

3.6.3 Comparison of Reconstruction Methods

In this section, we compare the accuracy of the two 3D reconstruction methods. In order to compare the results with the ground truth, we use a checkerboard pattern to compute the 3D positions of all control points. The pattern is placed on the table and rotated by 0 degree and 45 degree. We take a pair of stereo images at each angle, detect all corners, and compute the depth of corners using the disparity between their conjugates. We assume that the stereo camera is calibrated.

As shown in Figure 3.7, we measure the horizontal length between two control points which are at the upper-left and the upper-right corners on the pattern. In order to minimize noise effect in measurement, we also average all 8 horizontal lengths between left-most and the right-most corners. The triangulation method using Equation (3.23) shows small reconstruction error. As shown in Table 3.1, there is also a

difference between two lengths measured at 0 and 45 degrees. This difference could cause a serious problem when some multi-view models are registered and integrated to a single 3D model. In fact, when we use this method to integrate multiple range images, a geometric distortion occurs on the 3D model. In contrast, using Equation (3.24), we can reconstruct more accurate 3D model. Table 3.1 shows that the reconstruction results are more accurate than the triangulation method.

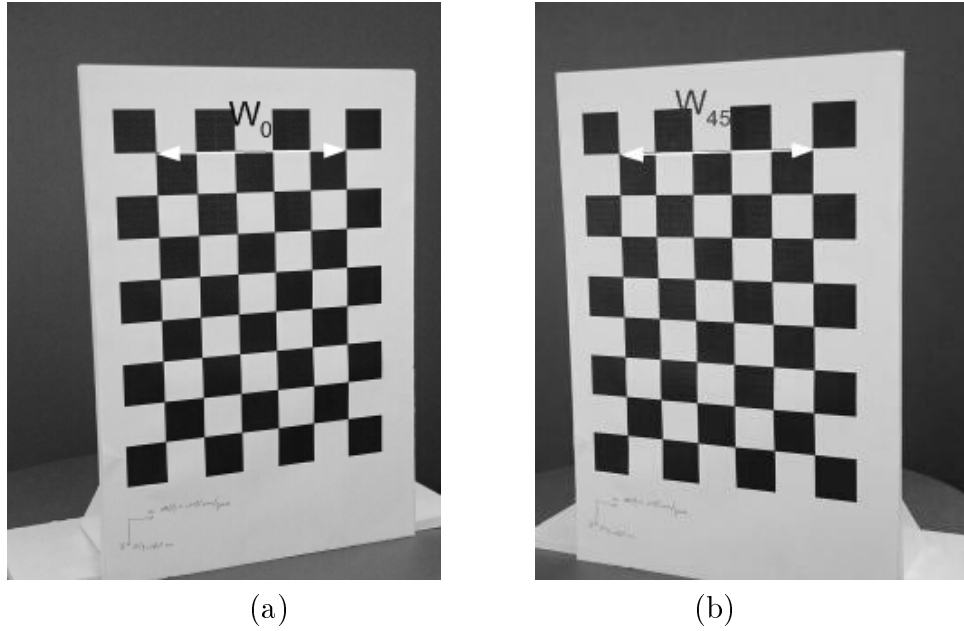


Figure 3.7: Checkerboard patterns for turntable calibration at (a) 0 degree (b) 45 degree

Table 3.1: Results of the width of the test pattern (Ground truth is 100 *mm*)

Method	w_0	w_{45}
Linear eq. (mm)	99.6	99.2
Triangulation (mm)	104.1	99.7

3.7 Turntable Calibration

3.7.1 Rotation Stage Calibration

The SVIS-3 system employs a turntable to change the viewing direction of an object. In order to merge multiple range images, we need to know the rigid transformation of each image with respect to a common coordinate system. Suppose there are N viewing directions for the object and view0 is the reference view point. When there is a 3D point \mathbf{P}_i^i which is obtained and represented by the i th view point, we can register it to a new point \mathbf{P}_i^0 in the reference view as follows:

$$\mathbf{P}_i^0 = \mathbf{T}_s^c \mathbf{R}_i \mathbf{T}_s^{c-1} \mathbf{P}_i^i. \quad (3.27)$$

where, \mathbf{R}_i is the rotational transformation from view i to view0, and \mathbf{T}_s^c is the transformation from the turntable coordinate system to the camera coordinate system, which is represented by

$$\mathbf{T}_s^c = [\mathbf{R}_s^c | \mathbf{t}_s^c]. \quad (3.28)$$

Let us define two coordinate systems in 3D space, the world coordinate and the turntable coordinate systems, whose origins are O_w and O_s respectively. Suppose we know the transformation \mathbf{T}_w^c , from the world coordinate system O_w to the camera coordinates O_c . If we know another transformation \mathbf{T}_s^w which is from the turntable O_s to the world coordinate O_w , then we can derive the transformation

$$\mathbf{T}_s^c = \mathbf{T}_w^c \mathbf{T}_s^w \quad (3.29)$$

$$= [\mathbf{R}_w^c | \mathbf{t}_w^c] [\mathbf{R}_s^w | \mathbf{t}_s^w] \quad (3.30)$$

Suppose there is a world coordinate system in 3D space with its origin at O_w as shown in Figure 3.8. In the figure, \mathbf{P}_0 is the origin of the world coordinate system (but it is not necessary) and \mathbf{P}'_0 is the same point after being rotated by θ angle along the \mathbf{Y}_s axis of the turntable. Given two 3D points and the rotation axis \mathbf{Y}_s , we can define a plane Π as shown in the figure. Then we know the vector product

$$(\mathbf{P}'_0 - \mathbf{P}_0) \cdot \mathbf{Y}_s = 0. \quad (3.31)$$

In other words,

$$\begin{bmatrix} p'_{0x} - p_{0x} \\ p'_{0y} - p_{0y} \\ p'_{0z} - p_{0z} \end{bmatrix}^T \begin{bmatrix} Y_{sx} \\ Y_{sy} \\ Y_{sz} \end{bmatrix} = 0. \quad (3.32)$$

When we have at least 3 points or more in world coordinates, we can solve a overdetermined linear equation

$$\mathbf{A}\mathbf{Y} = \begin{bmatrix} p'_{0x} - p_{0x} & p'_{0y} - p_{0y} & p'_{0z} - p_{0z} \\ p'_{1x} - p_{1x} & p'_{1y} - p_{1y} & p'_{1z} - p_{1z} \\ \dots \\ p'_{Nx} - p_{Nx} & p'_{Ny} - p_{Ny} & p'_{Nz} - p_{Nz} \end{bmatrix} \begin{bmatrix} Y_{sx} \\ Y_{sy} \\ Y_{sz} \end{bmatrix} = 0. \quad (3.33)$$

using the SVD technique. When the matrix \mathbf{A} is decomposed such that $\mathbf{A} = (\mathbf{U}\mathbf{D}\mathbf{V}^T)$, the solution of the equation is a column vector of \mathbf{V} which corresponds to the column of the least eigenvalue in \mathbf{D} matrix. We then normalize the \mathbf{Y}_s vector to $\hat{\mathbf{Y}}_s$. If the computed Y_{sy} is negative, then we change the direction of the axis to have the axis be the same direction with \mathbf{Y}_w axis of the world coordinate system.

In order to compute the \mathbf{X}_s and \mathbf{Z}_s axes of the turntable coordinate system, we apply the following computations. Let us initialize the \mathbf{X}_s axis to $(1.0, X_{sy}, 1.0)$. Then

$$\begin{aligned} \mathbf{X}_s \cdot \mathbf{Y}_s &= 0, \\ X_{sy} &= (-X_{sx}Y_{sx} - X_{sz}Y_{sz})/Y_{sy}, \\ \hat{\mathbf{X}}_s &= \mathbf{X}_s/\|\mathbf{X}_s\|, \\ \text{and } \hat{\mathbf{Z}}_s &= \hat{\mathbf{X}}_s \times \hat{\mathbf{Y}}_s. \end{aligned}$$

Finally, the rotation matrix from the turntable to the world coordinate system is defined as

$$\mathbf{R}_{ws} = \begin{bmatrix} (\hat{\mathbf{X}}_s)^T \\ (\hat{\mathbf{Y}}_s)^T \\ (\hat{\mathbf{Z}}_s)^T \end{bmatrix}^T. \quad (3.34)$$

Let us now consider a translation from the origin of the turntable coordinate system to the origin of the world coordinate system. The origin of the turntable coordinate system is defined as the intersection of the axis \mathbf{Y}_s and the plane Π . If we transform two 3D points \mathbf{P}_0 and \mathbf{P}'_0 using the rotation in Equation (3.34), transformed points are on the xz plane of the turntable coordinate system.

Suppose two points \mathbf{P}_0 and \mathbf{P}'_0 are transformed to the turntable coordinates to new points \mathbf{P}_0^s and \mathbf{P}'_0^s , respectively. Then the three points $\mathbf{O}_s, \mathbf{P}_0^s$, and \mathbf{P}'_0^s are on the plane Π and forms an isosceles triangle. Therefore using a vector

$$\begin{aligned} \mathbf{b} &= \mathbf{P}'_0^s - \mathbf{P}_0^s, \\ \text{where, } (\mathbf{P}_0^s, \mathbf{P}'_0^s) &= (\mathbf{R}_s^w)^T(\mathbf{P}_0, \mathbf{P}'_0) \end{aligned}$$

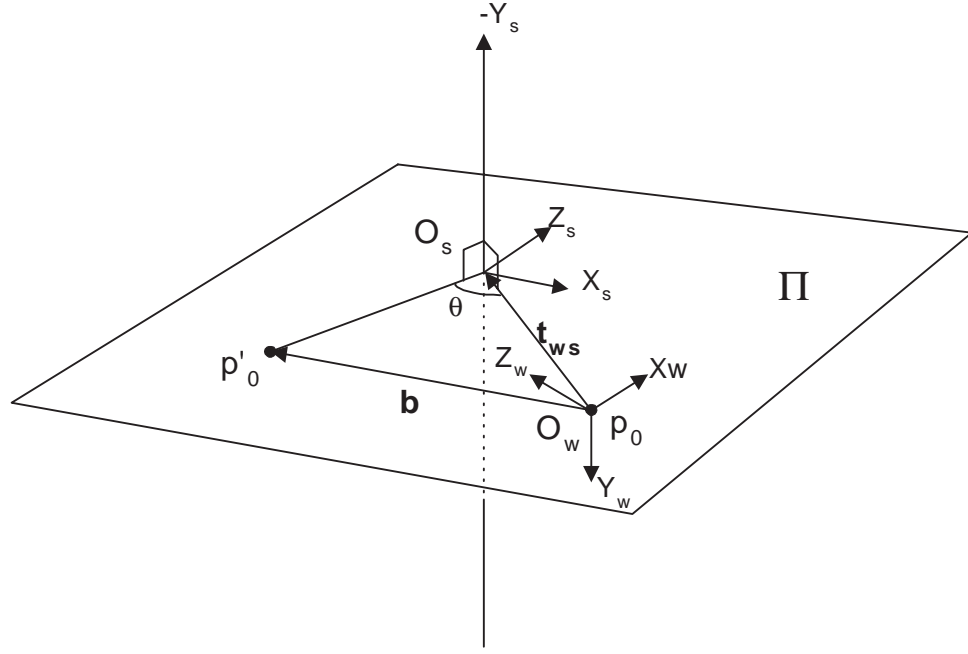


Figure 3.8: Rotation axis calibration with respect to the world coordinate system

and the rotation angle θ , we can compute a translation vector \mathbf{t}_s^w from \mathbf{P}_0^s to the origin O_s .

Let us consider the plane Π on which the origin is moved to \mathbf{P}_0^s and the y component is zero as shown in Figure 3.9. Then the center of rotation intersects with Π at a 3D point $\mathbf{t}_i = (\mathbf{x}, \mathbf{0}, \mathbf{z})$. Because the isosceles triangle is also on the plane, the origin \mathbf{t}_i is one of the intersection points of two circles c_1 and c_2 as shown in the figure. On the plane Π , the center of c_1 is at $(0.0, 0.0, 0.0)$ and its diameter is $\|\mathbf{t}_i\|$. Similarly, the center of c_2 is at $(b_x, 0.0, b_z)$ and its diameter is also $\|\mathbf{t}_i\|$. Let $r = \|\mathbf{t}_i\|$ and $b = \|\mathbf{b}\|$, then we derive two circle's equations

$$x^2 + z^2 = r^2 \quad (3.35)$$

$$(x_2 - b_x)^2 + (z_2 - b_z)^2 = r^2. \quad (3.36)$$

Using the equations, we get

$$z = \frac{b_x^2 + b_z^2 - 2b_x x}{2b_z}. \quad (3.37)$$

From equation (3.35), we also get

$$\begin{aligned}
 r^2 &= x^2 + \frac{(b^2 - 2b_x x)^2}{4b_z^2}, \\
 0 &= 4b^2 x^2 - 4b_x b^2 x + (b^4 - 4r^2 b_z^2), \\
 \text{where } r &= \frac{b/2}{\sin(\theta/2)}.
 \end{aligned} \tag{3.38}$$

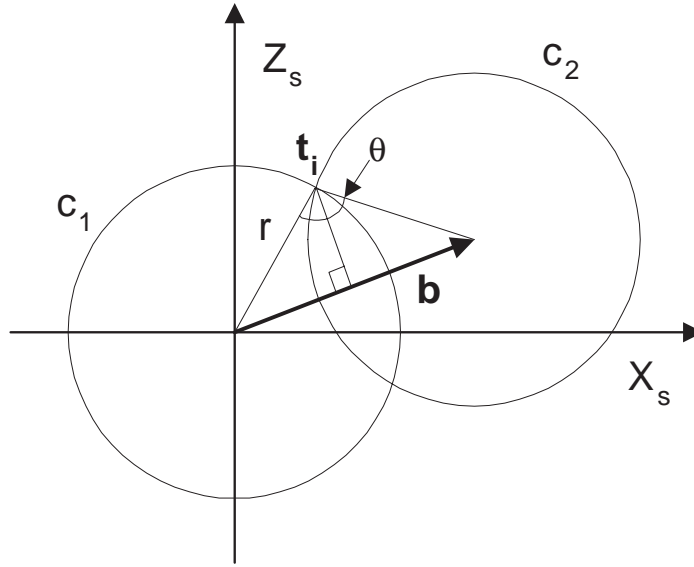


Figure 3.9: The rotation center is one of the intersections of two circles c_1 and c_2

Therefore, the x coordinates of the two intersection points are the solution of a second order binomial Equation (3.38). And the z coordinate is computed by Equation (3.37). Given two intersection points, only one of them is the real intersection point. If the intersection point is computed as $\mathbf{t}_i = (\mathbf{x}, \mathbf{0}, \mathbf{z})$ on the Π plane, it should have a property such that

$$\begin{aligned}
 \mathbf{a} &= \mathbf{b} \times \mathbf{t}_i, \\
 \text{and } a_y &> 0,
 \end{aligned}$$

because we rotate the point \mathbf{P}_0 by a positive angle θ along the Y_s axis of the turntable coordinates.

Now let us derive the transformation matrix from the turntable coordinate system to the camera coordinate system. Because we shift the origin O_s to point \mathbf{P}_0^s to find

the point \mathbf{t}_i , the translation from O_w to O_s becomes $-(\mathbf{t}_i + \mathbf{P}_0^s)$ with respect to the turntable coordinate system, and $-\mathbf{R}_s^w(\mathbf{t}_i + \mathbf{P}_0^s)$ with respect to the world coordinate system. Finally the transformation from turntable to the camera coordinate system is computed as

$$\begin{aligned} \mathbf{T}_s^c &= \mathbf{T}_s^w \mathbf{T}_w^c \\ \text{where, } \mathbf{T}_s^w &= [\mathbf{R}_s^w | \mathbf{t}_s^w] = [\mathbf{R}_s^w | -\mathbf{R}_s^w(\mathbf{t}_i + \mathbf{P}_0^s)]. \end{aligned} \quad (3.39)$$

In order to reduce noise effect on computing \mathbf{t}_i , we average results of the vectors using a number of world points.

3.7.2 Experimental Results

We use a checkerboard calibration pattern to estimate the turntable coordinate system. First we put the pattern on the table so that the xy plane of the pattern faces the camera, leaving the rotation axis behind. Using the stereo camera in the SVIS-3 system, we take two pairs of stereo pictures at 0 and θ degrees. Then we detect all 48 corner points in each picture. Using conjugate points in a pair of stereo picture, we compute the 3D position of the corner points with respect to the right-side camera's coordinate system.

Computing the transformation from the world coordinate system to the camera coordinate system is done as follows. As shown in Figure 3.10, translation \mathbf{t}_w^c is a vector from the camera to the upper-left corner point \mathbf{P}_{ul} . The three axes of the world coordinate system with respect to the camera system is computed as

$$\hat{\mathbf{r}}_{wx} = \mathbf{P}_{ur} - \mathbf{P}_{ul} / \|\mathbf{P}_{ur} - \mathbf{P}_{ul}\|, \quad (3.40)$$

$$\hat{\mathbf{r}}_{wz} = \mathbf{P}_{ll} - \mathbf{P}_{ul} / \|\mathbf{P}_{ll} - \mathbf{P}_{ul}\|, \quad (3.41)$$

$$\hat{\mathbf{r}}_{wy} = \hat{\mathbf{r}}_{wx} \times \hat{\mathbf{r}}_{wz}, \quad (3.42)$$

$$\text{and } \mathbf{R}_w^c = \begin{bmatrix} \hat{\mathbf{r}}_{wx}^T \\ \hat{\mathbf{r}}_{wy}^T \\ \hat{\mathbf{r}}_{wz}^T \end{bmatrix}^T \quad (3.43)$$

A transformation matrix from the turntable coordinate system to the camera coordinate system is computed as

$$\mathbf{T}_s^c = \begin{bmatrix} 0.504629 & 0.013110 & -0.863265 & -11.003747 \\ -0.123068 & 0.990635 & -0.058155 & -72.315765 \\ 0.854513 & 0.135868 & 0.501416 & 487.191485 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 \end{bmatrix} \quad (3.44)$$

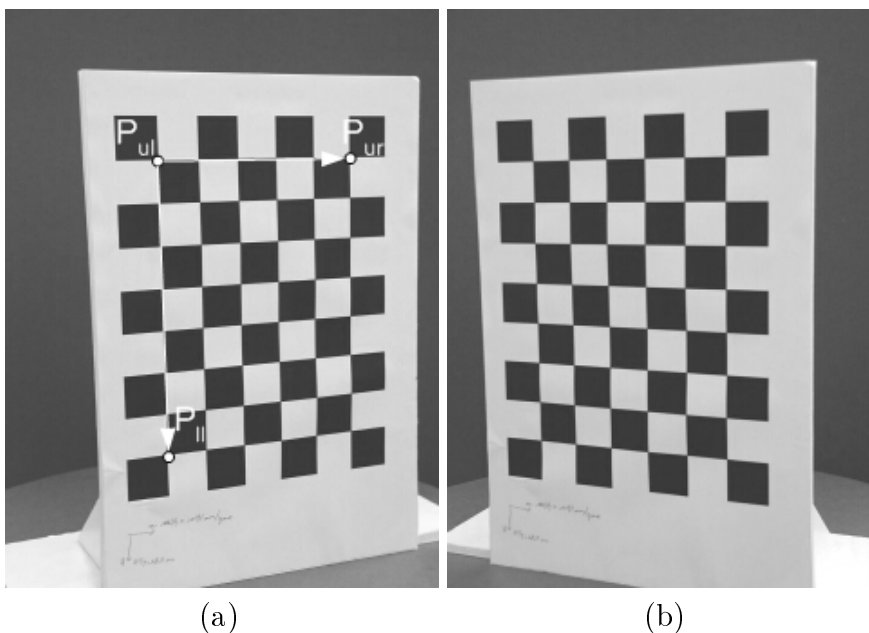


Figure 3.10: World coordinate system on the checkerboard patterns (a) 0 degree (b) 45 degree

We test our calibration algorithm at several positions of the stereo camera. Table 3.2 shows registration error between two control point sets, at 0 degree and 45 degree, on the checkerboard pattern. The z value of the translation \mathbf{t}_s^c shows distance from the camera to the turntable coordinates.

3.8 Conclusions

Calibration and rectification techniques of a new stereo-vision system (SVIS-3) are presented. The new camera has a toed-in vergence angle and an arbitrary focal length. External and internal calibration parameters are estimated through a camera calibration technique. Stereo images of an object are rectified according to the calibration parameters and a range image of the object is obtained using a matching technique. In order to facilitate registration of multi-view range images, we also calibrate a turntable. The new stereo camera system can be used for obtaining multi-view range images and for reconstructing a complete 3D shape of real objects.

Table 3.2: Registration error of turntable calibration in *mm*

\mathbf{t}_s^c			Mean error	Max. error
x	y	z		
-41.9	-64.6	393.7	0.21	0.53
-41.0	-69.3	420.6	0.21	0.53
-18.2	-80.8	488.9	0.15	0.39
-15.4	-93.3	569.4	0.11	0.20
4.93	-105.3	638.1	0.11	0.25
14.2	-120.4	727.1	0.18	0.35
21.9	-140.1	825.5	0.27	0.50

Chapter 4

Surface-based 3D Reconstruction

This chapter and the next chapter present two different 3D model reconstruction techniques. The two techniques both address the problem of merging multi-view range images into a complete 3D model. In general, there are two approaches for merging range images. One is a surface-based approach which stitches multiple partial surfaces into a single surface. The other is a volumetric approach which finds the iso-surface of an object in a grid of voxels. This chapter presents a surface-based merging technique.

4.1 Introduction

We present a surface-based reconstruction technique which exploits the epipolar constraint for the multi-view geometry. Partial 3D shapes of an object from multiple viewing directions are obtained using the SVIS-2 vision system presented in Chapter 2. The vision system is calibrated as described in the same chapter, to obtain initial transformation matrices for both the stereo geometry and the multi-view geometry. The partial 3D shapes are registered coarsely using the initial transformation matrices. They are then refined iteratively until registration errors between overlapping surfaces minimize close to zero. At this step, a modified Iterative Closest Point (ICP) algorithm is used to find correspondence between two different views. A 3D point in one view is projected to another view using the transformation between them and a search is made for a closest point in the other view that lies on the epipolar line. A similar idea is used during partial surface integration step to obtain improved results. Partial surfaces are represented as linked lists of segments and integrated segment by segment. Experimental results are presented to show the effectiveness of the technique.

4.2 Multi-view Registration

4.2.1 Coarse Registration

We acquire stereo images of an object from N viewing directions by rotating the object with $360/N$ angle. However, we model this data acquisition by assuming the object to be stationary but the camera system being rotated around the rotation axis by the same angle. For each of the rotation position of the camera system, we associate a coordinate system \mathcal{V}_i for $i = 0, \dots, N$. Stereo image analysis, which was presented in Chapter 2, is used to compute the partial 3D shapes of the object with respect to \mathcal{V}_i . In our experiments, \mathcal{V}_i is the same as the right camera coordinate system of the stereo rig.

All partial 3D shapes computed with respect to \mathcal{V}_i have to be registered with respect to a common coordinate system, which is \mathcal{V}_0 of the first view. An accurate knowledge of the position and orientation of the rotation axis of the turntable with respect to \mathcal{V}_0 is needed for registration of the partial 3D shapes. Let us assume the orientation of the rotation axis is almost parallel to the y -axis of the camera coordinate system and the rotation angle of the turntable is very accurately controlled by a computer. The translation of the rotation axis from \mathcal{V}_0 is defined by a vector \mathbf{t}_s . An initial estimate of \mathbf{t}_s is obtained through stereo camera calibration which is made by Tsai's technique [98]. This estimate will be refined iteratively during registration refinement as described later.

Let \mathbf{P}_i^i denote a 3D point obtained and represented with respect to \mathcal{V}_i . Similarly, let \mathbf{P}_i^j denote the same 3D point which is represented with respect to \mathcal{V}_j by registering \mathbf{P}_i^i to \mathcal{V}_j . Also let \mathbf{R}_j^i be the rotation transformation matrix from \mathcal{V}_j to \mathcal{V}_i . In order to register the point \mathbf{P}_j^j to the common coordinate system \mathcal{V}_0 to obtain \mathbf{P}_j^0 , we use a rigid transformation,

$$\mathbf{P}_j^0 = \mathbf{T}_j^0 \mathbf{P}_j^j = \mathbf{R}_j^0 (\mathbf{P}_j^j - \mathbf{t}_s) + \mathbf{t}_s \quad (4.1)$$

where, \mathbf{t}_s is the translation vector defined earlier.

4.2.2 Partial Shape Representation

The partial 3D shapes are initially registered by \mathbf{R}_j^0 and \mathbf{t}_s , which are estimated from the system calibration. The registered shapes are then represented by the coordinates of the first view \mathcal{V}_0 with its y -axis along the rotation axis. Suppose the partial shapes are registered to a 3D workspace which contains the object, and the workspace is assumed to be a cube with its volume $W \times H \times D$ (mm^3). Now let us consider intersections of the partial 3D shapes with N_H horizontal planes which are d_H mm apart and parallel to the $x - z$ plane. Then the intersections are horizontal contours as shown in Figure 4.1, where each contour consists of a sequence of points.

These points are represented by the 2D coordinates on the horizontal slice containing them, where each slice corresponds to a constant vertical (y) coordinate in 3D space.

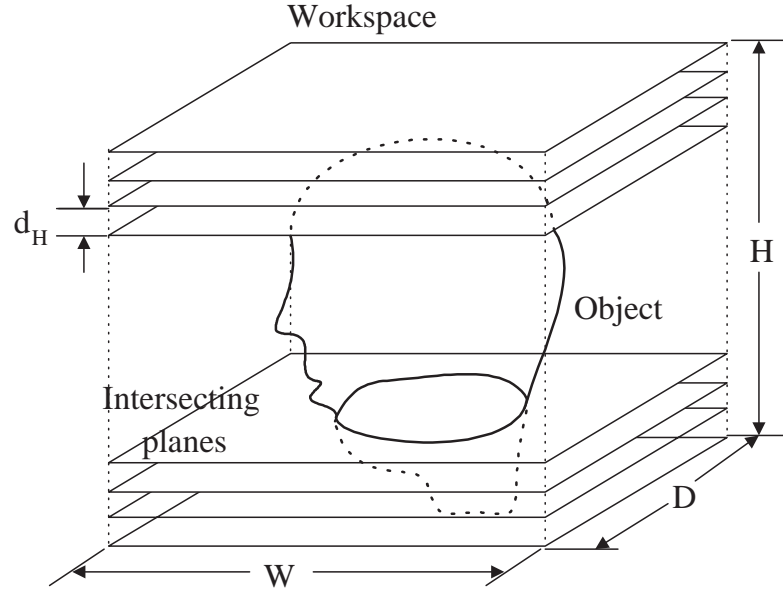


Figure 4.1: Representation of a volumetric workspace: The object workspace is a stack of slices on which a 3D object is represented by segmented contours.

Now suppose a slice from the stack as shown in Figure 4.2. On this 2D slice, we establish linked lists of all surface points and sort them with respect to x and z coordinates. The sorting is done because we want the linked list to represent the points in order on a continuous contour on the object's surface. For each slice plane, we generate one linked list for each viewing direction, thereby generating N linked lists. Figure 4.2 shows an example when $N = 4$. Let \mathcal{LV}_i be the i th linked list of points for \mathcal{V}_i . Integration of all linked lists for $i = 0, \dots, N - 1$ is merging the lists in such a way that the resulting list represents a closed contour that is an accurate cross-section of the object.

There will be some erroneous points on the list representing short, irregular, false contours that do not belong to the object. Also, some points on the object's surface could be missing due to gaps in contours. These errors are introduced by stereo mismatching due to occlusion, low contrast, or noise. Therefore, we segment the linked lists to detect and reduce these errors. This is important because refining the registration matrix should be done using only the correct object points.

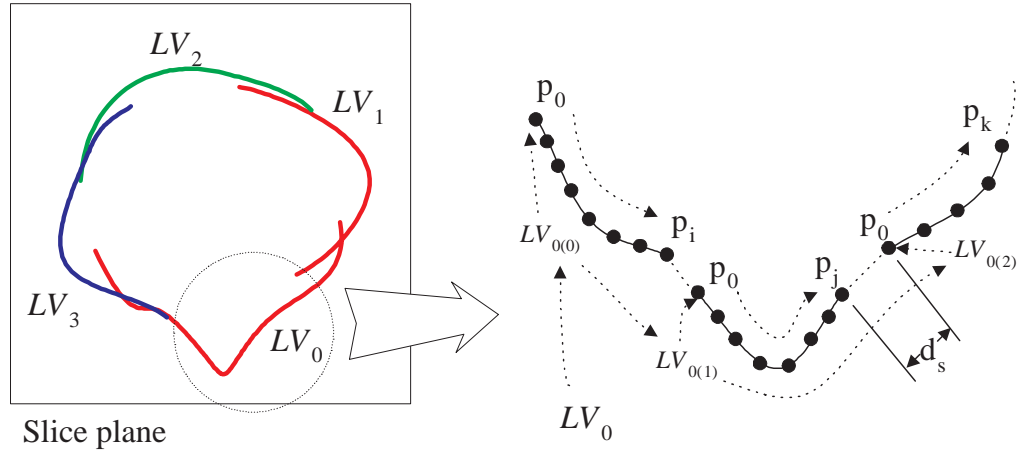


Figure 4.2: Hierarchical structure of contour segments

Initially we have only one linked list of points \mathcal{LV}_i for the i th view. Then, we split the linked lists \mathcal{LV}_i into multiple sublists $\mathcal{LV}_{i(j)}$ where each sublist represents one connected contour segment. This splitting is done based on thresholding the distance between two consecutive points on a list. If the distance $d_s = |\mathbf{P}_{(i)} - \mathbf{P}_{(i+1)}|$ is larger than a threshold, we divide the linked list into two separate lists under the assumption that the object surface is locally linear and continuous. As a result, a list \mathcal{LV}_i for i th view is split into multiple lists $\mathcal{LV}_{i(j)}$ for $m = 0, \dots, M$ where M is the number of segments on the list of i th view. After generating linked lists of segments for all views, we remove a linked list if its length d_l is too short, as it is very likely to be due to errors in stereo mismatching. When there are K points on a segment list, the length of a segment list is computed as

$$d_l = \sum_{k=0}^{K-1} |p_{(k)} - p_{(k+1)}| \quad (4.2)$$

Figure 4.3 shows a tree structure of the linked list of segments and voxels on a slice of one view.

4.3 Registration Refinement

In Chapter 2, we presented the calibration of the SVIS-2 vision system. Assuming that each rotation interval of the turntable is exactly $(360/N)$ degree, we calibrate the

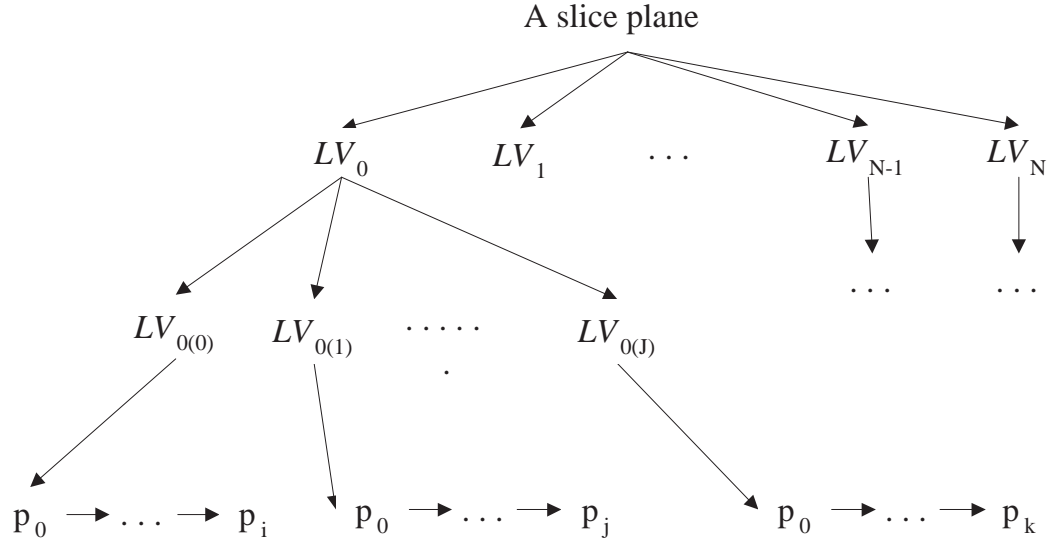


Figure 4.3: The structure of a segment linked list

rotation and the translation matrices \mathbf{R}_s and \mathbf{t}_s from the origin of the common camera coordinates to the center of rotation coordinates \mathbf{O}_s . In previous section, all partial shapes are registered to the first view's coordinate system according to the calibration parameters. However, due to inherent errors on system calibration, we need to refine the parameters before integration of shapes. Before describing refinement procedures, let us present a epipolar geometry between different viewing points.

4.3.1 Epipolar Geometry of Multiple Views

In order to refine the registration matrix between a pair of partial shapes, we introduce a new technique called EICP (Epipolar Iterative Closest Point) method. Our technique exploits the epipolar geometry to reduce errors and computation complexity in searching the closest point. Given two segment lists, we compare spatial information of two segments in the image space as well as in the object space. Comparing in the image space is based on the reprojection of one segment to the other's image space. To find out the geometrical information of two segments in the image space, we transform the coordinate of one segment to the other's coordinate system.

Consider two segment lists \mathcal{LV}_0 and \mathcal{LV}_1 , which are reconstructed from \mathcal{V}_0 and \mathcal{V}_1 , respectively. Each list consists of many sublists and each sublist represents a

contour segment. Therefore we can write

$$\begin{aligned}\{\mathcal{LV}_0\} &= \{\mathcal{LV}_{0(j)} \mid j = 0, \dots, J-1\} \\ \{\mathcal{LV}_1\} &= \{\mathcal{LV}_{1(k)} \mid k = 0, \dots, K-1\}\end{aligned}$$

, where M and Q are the number of sublists at the 0 and 1 view, and \mathcal{LV}_{ij} is the j th sublist on \mathcal{LV}_i . Let \mathbf{P}_0^0 be a point in a list of \mathcal{LV}_0 , the projection of this point onto the image plane of \mathcal{V}_0 be \mathbf{p}_0^0 , and another projection onto the image plane of \mathcal{V}_1 be \mathbf{p}_0^1 .

Figure 4.4 shows the geometry of \mathcal{V}_0 and \mathcal{V}_1 . The three points—origin of \mathcal{V}_0 , origin of \mathcal{V}_1 , and the point \mathbf{P}_0^0 —together determine the epipolar plane between two different views. A point \mathbf{P}_0^i in \mathcal{V}_0 is denoted by \mathbf{P}_0^i in \mathcal{V}_i . They are related by

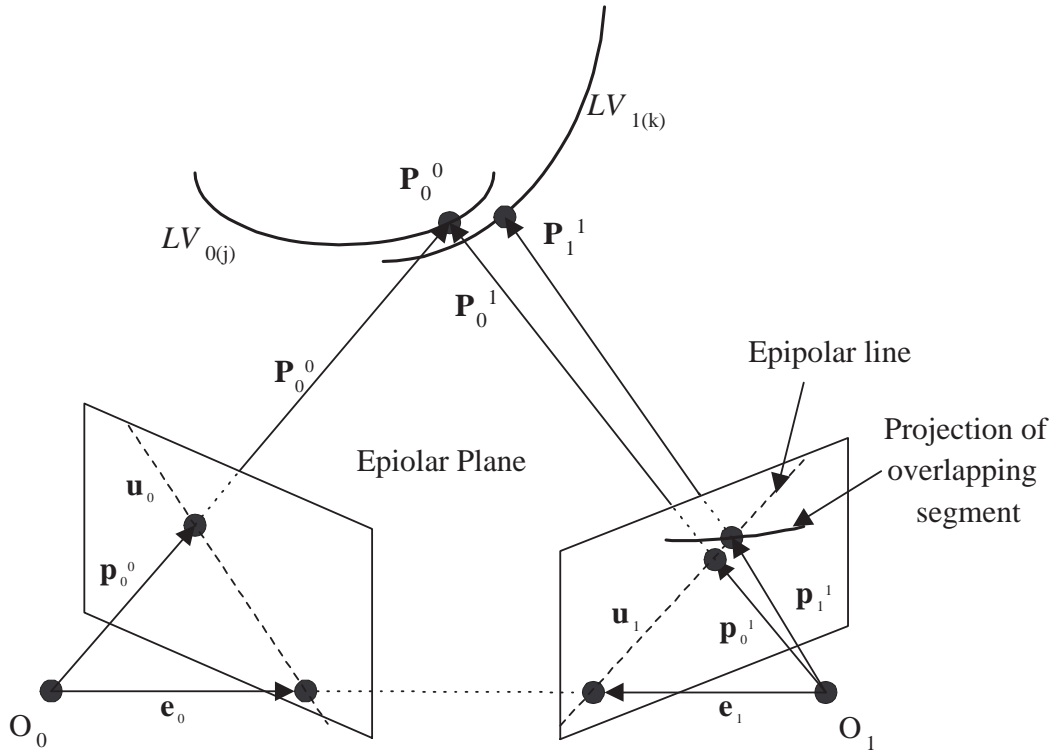


Figure 4.4: Epipolar geometry of \mathcal{V}_0 and \mathcal{V}_1

$$\mathbf{P}_0^i = \mathbf{R}_0^i(\mathbf{P}_0^0 - \mathbf{t}_s) + \mathbf{t}_s. \quad (4.3)$$

The projection of \mathbf{P}_0^i onto the image plane in \mathcal{V}_i is given by the perspective transformation relation:

$$\mathbf{p}_0^i = \left(f \frac{P_{0x}^i}{P_{0z}^i}, f \frac{P_{0y}^i}{P_{0z}^i}, f \right), \quad (4.4)$$

Setting $i = 1$ in the above equations, we can get the epipolar line \mathbf{u}_1 on the image plane in \mathcal{V}_1 by transforming the image vector \mathbf{p}_0^0 as

$$\mathbf{u}_1 = \mathbf{E} \mathbf{p}_0^0 \quad (4.5)$$

where, E is a essential matrix and it is represented as

$$\mathbf{E} = \mathbf{R}_0^1 \mathbf{S}, \quad (4.6)$$

and

$$\mathbf{S} = \begin{pmatrix} 0 & -t_{0z}^1 & t_{0y}^1 \\ t_{0z}^1 & 0 & -t_{0x}^1 \\ -t_{0y}^1 & t_{0x}^1 & 0 \end{pmatrix}$$

, where \mathbf{t}_0^1 is a translation vector ($V_1 - V_2$). If two points \mathbf{P}_0^0 and \mathbf{P}_1^1 correspond to the same physical point on the object, the projections of the points onto the same image plane should have the same coordinate. Therefore \mathbf{p}_0^0 and \mathbf{p}_1^1 should be the same vectors.

However, because of registration error due to errors in initial calibration caused by stereo mismatching, distortion of the camera lens, and noise, there will be an error between two vectors. To minimize the registration error, we introduce an iterative minimization technique based on the epipolar geometry of the two directions of view. In Figure 4.4, we can expect the vector \mathbf{p}_1^1 to be on the epipolar line \mathbf{u}_1 if two vectors are different representations of the same point, because the epipolar line is the transformation of the vector \mathbf{p}_0^0 to the \mathcal{V}_1 image plane.

4.3.2 Pairwise Refinement

In order to refine the translation matrix, we search for two closely overlapping segments from segment lists of two different views and compute translation error between them. The errors \mathbf{t}_ϵ is computed for all possible overlapping segments from adjacent pairs of views and averaged. This error is assumed to have a Gaussian distribution. The average error is iteratively minimized until it converges close to zero.

Let us consider registration of two partial segments $\mathcal{L}\mathcal{V}_0$ and $\mathcal{L}\mathcal{V}_1$, which are reconstructed from \mathcal{V}_0 and \mathcal{V}_1 , respectively. Suppose there is a 3D point \mathbf{P}_0^0 on $\mathcal{L}\mathcal{V}_0$, a transformed point \mathbf{P}_0^1 to \mathcal{V}_1 , its projection \mathbf{p}_0^1 onto the image plane of \mathcal{V}_1 , and

the corresponding epipolar line \mathbf{u}_1 . Projection of the point \mathbf{P}_0^1 to \mathbf{p}_0^1 is done by a projection matrix \mathbf{M}_1 such that

$$\mathbf{p}_0^1 \cong \mathbf{M}_1 \mathbf{P}_0^1 = \mathbf{A}[\mathbf{I}|\mathbf{0}]\mathbf{P}_0^1, \quad (4.7)$$

$$\text{where } \mathbf{A} = \begin{bmatrix} f_u & \gamma & u_i \\ 0 & f_v & v_i \\ 0 & 0 & 1 \end{bmatrix}.$$

Given a segment in $\mathcal{L}\mathcal{V}_{0(j)}$ that contains \mathbf{P}_0^0 , we find a closely overlapping segment $\mathcal{L}\mathcal{V}_{1(k)}$ in \mathcal{V}_1 as shown in Figure 4.5. We consider a contour segment $\mathcal{L}\mathcal{V}_{1(k)}$ is overlapping with \mathbf{P}_0^0 when the distance from any point on the segment to the \mathbf{P}_0^0 is shorter than a threshold d_{TH} . In Figure 4.5, black-colored segment on $\mathcal{L}\mathcal{V}_{1(k)}$ is overlapping with the point \mathbf{P}_0^0 .

Now let us project the segment onto the image plane of \mathcal{V}_1 to plot a projected contour lv_{1k} . As presented in this chapter, the projection of the corresponding point \mathbf{P}_0^1 should be on the epipolar line \mathbf{u}_1 , if there is no registration error. However, because of the error, we need to find an image point on lv_{1k} which is the closest to the epipolar line. Rather than comparing projection distances of all points on lv_{1k} to \mathbf{u}_1 , we fit the segment into a linear line lv'_{1k} to get the intersection \mathbf{p}_1^1 with the epipolar line. Then we select an image point \mathbf{p}_1^1 , which is the closest to \mathbf{p}_1^1 , as the matching image point. This point is back projected onto the segment $\mathcal{L}\mathcal{V}_{1(k)}$ to determine the point \mathbf{P}_1^1 in 3D space.

If we assume that two vectors \mathbf{P}_0^0 and \mathbf{P}_1^1 in 3D space correspond to the same physical point on the object, we can consider the difference of the two vectors as a translation error between two view coordinates. Let $\mathbf{t}_{\epsilon|0}^1$ be a translation error between two views, then

$$\mathbf{t}_{\epsilon|0}^1 = \mathbf{P}_0^1 - \mathbf{P}_1^1 \quad (4.8)$$

The translation vector \mathbf{t}_0^1 between two object points from \mathbf{P}_0^0 to \mathbf{P}_1^1 is computed as follows. Let \mathbf{t}_s' be a refined translation matrix between the first view coordinate system \mathcal{V}_0 and the rotation center. Then

$$\mathbf{t}_s = \mathbf{t}_s' + \mathbf{t}_\epsilon \quad (4.9)$$

,where \mathbf{t}_s is the translation matrix which is calibrated by Tsai's calibration method and \mathbf{t}_ϵ is a translation error in \mathbf{t}_s . To find the translation between two camera coordinate system, we use transformation between two vectors

$$\begin{aligned} \mathbf{P}_1^1 &= \mathbf{R}_0^1(\mathbf{P}_0^0 - \mathbf{t}_s) + \mathbf{t}_s \\ &= \mathbf{R}_0^1\mathbf{P}_0^0 + (\mathbf{I} - \mathbf{R}_0^1)\mathbf{t}_s \\ &= \mathbf{R}_0^1\mathbf{P}_0^0 + \mathbf{t}_0^1 \end{aligned} \quad (4.10)$$

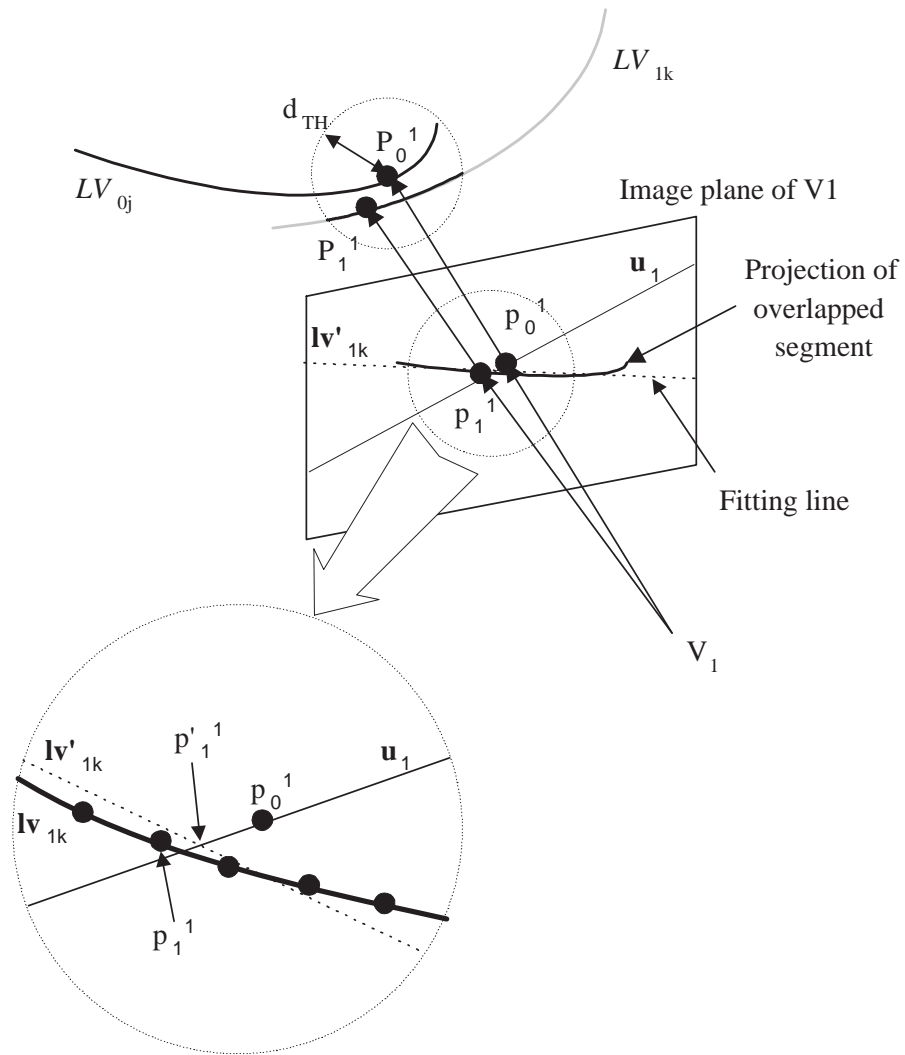


Figure 4.5: Finding closest point on epipolar line

, where \mathbf{t}_0^1 is the translation from \mathbf{P}_0^0 to \mathbf{P}_1^1 and can be expressed as

$$\begin{aligned} \mathbf{t}_0^1 &= \mathbf{t}_0^{1'} + \mathbf{t}_{\epsilon|0}^1 \\ \text{and} \\ \mathbf{t}_{\epsilon|0}^1 &= (\mathbf{I} - \mathbf{R}_0^1)\mathbf{t}_{\epsilon} \\ \mathbf{t}_{\epsilon} &= (\mathbf{I} - (\mathbf{R}_0^1)^T) \mathbf{t}_{\epsilon|0}^1. \end{aligned} \quad (4.11)$$

Where, $(\mathbf{R}_0^1)^T$ is the transpose of \mathbf{R}_0^1 and $\mathbf{t}_{\epsilon|0}^1$ is a translation error in the matrix \mathbf{t}_0^1 .

Using Equation (4.11), we find the matrix \mathbf{t}_{ϵ} from the epipolar geometry of the overlapping segments, and compute the translation matrix \mathbf{t}_s by Equation (4.9). This procedure is repeated to all horizontal slice planes and the average of the translation error $\hat{\mathbf{t}}_{\epsilon}$ is computed. After getting the matrix \mathbf{t}_s , we re-register all partial shape to the common coordinate system based on the new translation matrix. The matrix is computed again iteratively until the error \mathbf{t}_{ϵ} converges close to zero. An algorithm for refining the translation matrix is summarized below.

Pseudo-codes for registration refinement

do

for j = 0 to m : number of segments on \mathcal{LS}_0

for each point on a segment $\mathcal{LV}_{0(j)}$

for k = 0 to n : number of segments on \mathcal{LS}_1

for each point on a segment $\mathcal{LV}_{1(k)}$

if Overlapping segment($\mathbf{p}_0^1, \mathcal{LV}_{1(k)}$)

 find intersecting point \mathbf{p}_1^1

$\mathbf{t}_{\epsilon|0}^1 \leftarrow \mathbf{P}_0^1 - \mathbf{P}_1^1$

$\mathbf{t}_{\epsilon} \leftarrow (\mathbf{I} - (\mathbf{R}_0^1)^T) \mathbf{t}_{\epsilon|0}^1$

$\hat{\mathbf{t}}_{\epsilon} \leftarrow$ averaged \mathbf{t}_{ϵ}

$\mathbf{t}_s^{(t+1)} \leftarrow \mathbf{t}_s^{(t)} + \hat{\mathbf{t}}_{\epsilon}$:update (t + 1)th iteration

 Register again all segment lists of V_0 and V_1

while ($\hat{\mathbf{t}}_{\epsilon}$ not close to zero).

4.4 Multi-view Integration

4.4.1 Partial Shape Integration

After finding the \mathbf{t}_s matrix that minimizes the registration error for all views, we integrate partial shapes to obtain a complete 3D shape. The basic ideas used in this step are similar to those in the registration step. Integration is also done slice by slice by merging partial shapes of views \mathcal{V}_0 and \mathcal{V}_1 first, \mathcal{V}_1 and \mathcal{V}_2 next, etc. Consider the integration of partial shapes of views \mathcal{V}_0 and \mathcal{V}_1 . we have two linked lists $\mathcal{L}\mathcal{V}_{0(j)}$ and $\mathcal{L}\mathcal{V}_{1(k)}$, where i and j are the index of segments for each view. For a point \mathbf{P} in the linked list $\mathcal{L}\mathcal{V}_{0(j)}$, we find points on the $\mathcal{L}\mathcal{V}_{1(k)}$, which are close to the point \mathbf{P} . As before, let \mathbf{P}_0^0 be a point vector in \mathcal{V}_0 which becomes \mathbf{P}_0^1 after transforming to \mathcal{V}_1 , and let \mathbf{p}_0^1 be its projection onto the image plane in \mathcal{V}_1 .

Finding points close to \mathbf{p}_0^1 is similar to the method in the registration step. It is based on two criteria. We find the epipolar line \mathbf{u}_1 corresponding to \mathbf{p}_0^1 , and find a segment from the list of second view that intersects the epipolar line close to \mathbf{p}_0^1 . As the first criterion, we use c_1 given by a scalar product of two vectors

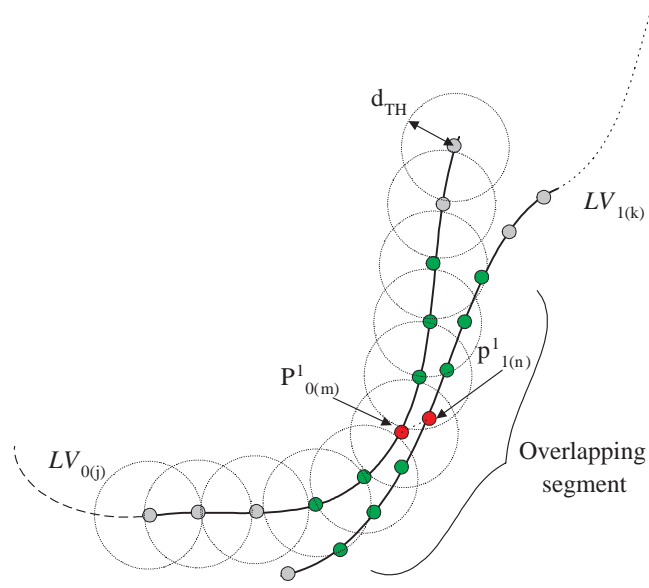
$$c_1 = \mathbf{u}_1 \cdot \mathbf{p}_1^1. \quad (4.12)$$

We find a set of close points \mathbf{p}_1^1 for which c_1 is smaller than a pre-defined threshold. As the second criterion, we use c_2 which is the Euclidean distance between two points in 3D space,

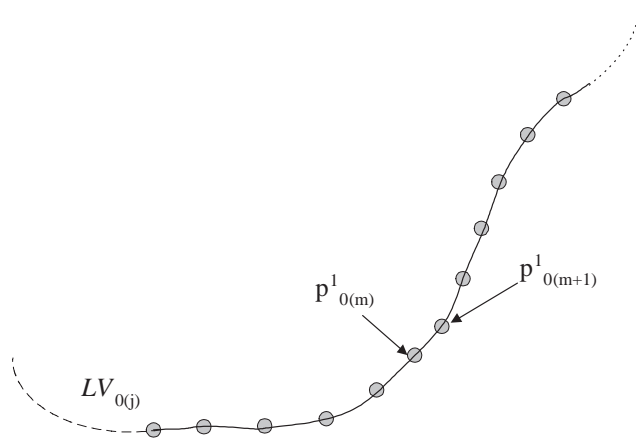
$$c_2 = \|\mathbf{P}_0^1 - \mathbf{P}_1^1\|. \quad (4.13)$$

The distance has to be less than the threshold d_{TH} . Both criteria have to be satisfied for merging the two segment lists. If they are satisfied, the linked list $\mathcal{L}\mathcal{V}_{0(j)}$ is connected to the list $\mathcal{L}\mathcal{V}_{1(k)}$ as shown in Figure 4.6. In the figure, a point $\mathbf{P}_{0(m)}^1$ on a segment $\mathcal{L}\mathcal{V}_{0(j)}$ has the closest point $\mathbf{p}_{1(q)}^1$ on another segment $\mathcal{L}\mathcal{V}_{1(k)}$. Therefore, the point $\mathbf{P}_{0(m)}^1$ is connected to a new $\mathbf{P}_{0(m+1)}^1$ which was $\mathbf{p}_{1(q)}^1$ on $\mathcal{L}\mathcal{V}_{1(k)}$. The connected segment $\mathcal{L}\mathcal{V}_{0(j)}$ also has a new data structure on its linked list. As shown in Figure 4.7, two linked lists are connected to a single linked list in a similar way with the connection of two contours. The number of lists of the new structure is also updated by counting all its elements. After integration, a linked list $\mathcal{L}\mathcal{V}_{1(k)}$ is removed.

We compare all linked lists between two views $\mathcal{L}\mathcal{V}_{0(j)}$ and $\mathcal{L}\mathcal{V}_{1(k)}$ for $j = 0, \dots, J$ and $k = 0, \dots, K$ one by one to find possible connections. After connecting the linked lists of two views, we check the next segment lists between \mathcal{V}_1 and \mathcal{V}_2 , and so on. Final step is connecting segments between \mathcal{V}_{N-1} and \mathcal{V}_0 on the same slice. After the last connection, we check the distance between the starting point of \mathcal{V}_0 and the ending point of \mathcal{V}_{N-1} . If it is small, they are connected to close the segment lists. This results in a closed contour representing the horizontal cross-section of the



(a)



(b)

Figure 4.6: Connection of two contour segments (a) Point $p_{0(m)}^1$ on $\mathcal{LV}_{0(j)}$ has the closest point $p_{1(n)}^1$ on $\mathcal{LV}_{1(k)}$ (b) Point $p_{0(m)}^1$ is connected to a new $p_{0(m+1)}^1$ which was $p_{1(n)}^1$ on $\mathcal{LV}_{1(k)}$.

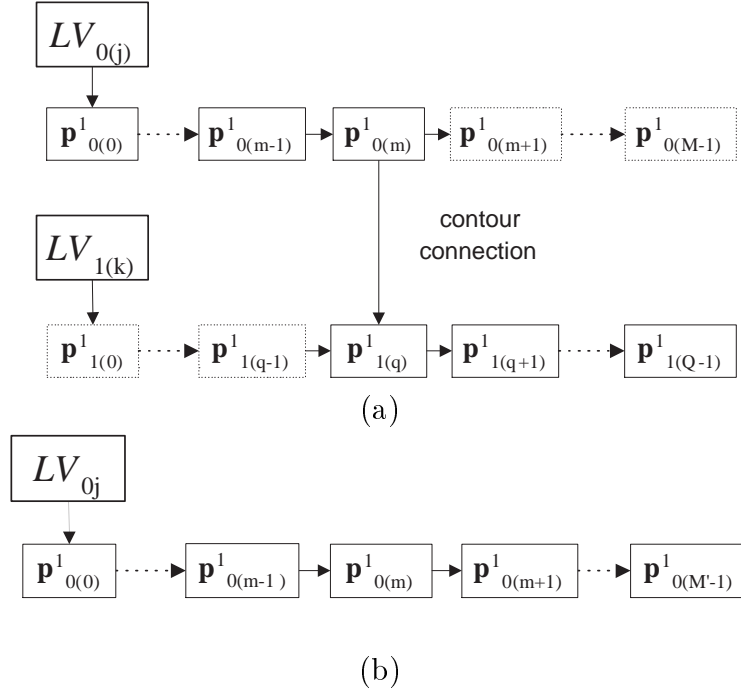


Figure 4.7: Connection of two linked lists (a) Point $\mathbf{p}_{0(m)}^1$ in the list of $\mathcal{LV}_{0(j)}$ is connected to $\mathbf{p}_{1(q)}^1$ in $\mathcal{LV}_{1(k)}$ (b) List $\mathcal{LV}_{0(j)}$ has a new linked list after integration.

object. After connecting all segment lists together, we remove again some short-length segments. A Gaussian low pass filter is applied to the connected contours to get a smoothed 3D shape after integration.

4.4.2 Mesh Generation

After integration of contours between different view points, a slice plane consists of several connected contours. In other word, all slices in object space consists of stacks of contours which are integrated from all slice planes. Surface meshes of integrated contours are generated between two adjacent slices from top to bottom. Given two slices which are facing each other, we generate a mesh of triangles by connecting all contours.

Given a point on the upper slice, the closest point on the lower slice is connected as an edge of a triangle. Then the next point in the same linked list is also connected with the upper point. For example in Figure 4.8, a point \mathbf{P}_k on list $\mathcal{LV}_{0(0)}^{(u)}$ is connected

to the closest point \mathbf{P}_m on another list $\mathcal{LV}_{0(0)}^{(l)}$ on the lower slice. The next point \mathbf{P}_{m+1} is then connected to \mathbf{P}_k to form a triangle $\mathbf{P}_k\mathbf{P}_m\mathbf{P}_{m+1}$. Other triangles are generated in the same way until there is a gap on either contour. If there is a disconnection, an edge of triangle can't be generated if its length is too long, for example between \mathbf{P}_0 on $\mathcal{LV}_{0(1)}^{(u)}$ and \mathbf{P}_1 on $\mathcal{LV}_{11}^{(l)}$. In case that there is a new contour on either slice, an edge of triangle can be generated even though its length is not the shortest among all candidates. For example in the figure, from a point \mathbf{P}_0 on \mathcal{LV}_{01} has the shortest point \mathbf{P}_3 on \mathcal{LV}_{11} . However, it is better to select the previous point \mathbf{P}_2 to form an edge $\mathbf{P}_0\mathbf{P}_2$, if the length of the edge is not too long.

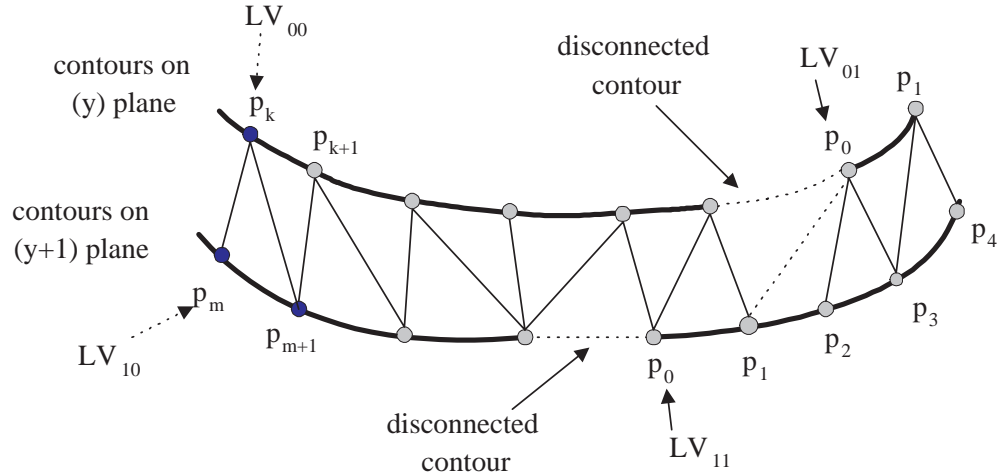


Figure 4.8: Mesh generation between two slice planes.

4.5 Experimental Results

The presented surface-based 3D reconstruction technique is tested on three real objects. Each object is placed on the turntable and 4 stereo image pairs are pictured from different view points by rotating the object by 90 degree. Each stereo image is obtained with TIFF format with 1280×960 size. Range images of the object is obtained by the technique described in Chapter 2. In order to introduce contrast on the object's surface, a slide projector is used to project a random dot pattern.

Figure 4.9 shows three objects used in this experiments. For an object such as 'Potatohead' in the figure, we do not use the slide projector because it already has contrast on its surface. Figure 4.10 shows partial shapes of 4 different views, all

partial shapes are represented by segmented contours along the $x - z$ plane of the camera coordinate system. Integration of the sets of contours between different views are accomplished slide by slide.

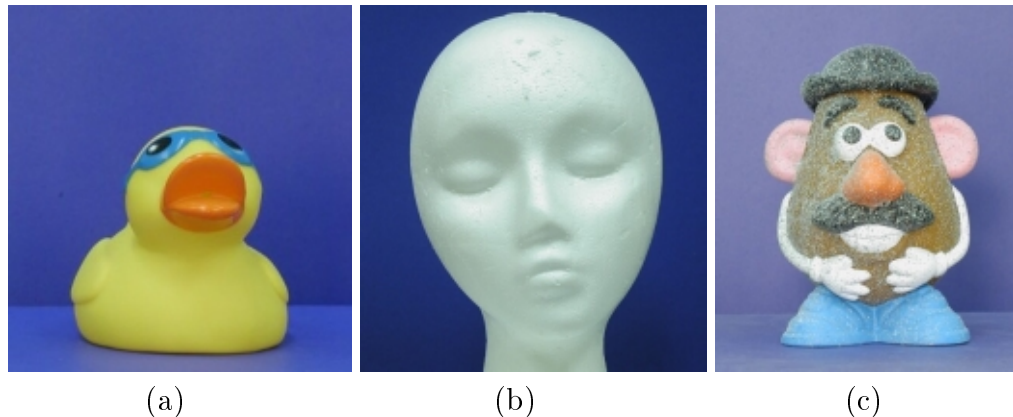


Figure 4.9: Test objects for experiments: (a) Duck (b) Head (c) Potatohead

Figure 4.11 shows an example of merging two adjacent partial shapes. Figure 4.11(a) is the shape from the front view and (b) is that from the left-side view. On each plane of slices, two views are merged to reconstructed the result shown in Figure 4.11(c). Merging of all 4 partial shapes is shown in Figure 4.12(a). In this figure, the object is represented by a stack of contours which are coplanar with $x - z$ plane. A mesh model of the object is generated and some novel views of it are displayed on Figure 4.12(b) and (c). Because there are no information on the top and the bottom views, there are some holes on those surfaces. Figure 4.13 and 4.14 show the results of other objects 'Head' and 'Potatohead'. In Figure 4.14 (c), there is an error on surface on object's left foot due to a stereo mismatching.

Translation errors from the center of the camera coordinate system to that of the rotation stage is measured by the technique described in the previous section. The results are plotted in Figure 4.15. In the figure all objects are tested for 30 iterations to show convergence of the error. However, registration procedure stops after 5 or 6 iterations in real applications. All results shows that the error reduces close to zero after 10 iterations.

4.6 Conclusions

This chapter presents a surface-based 3D reconstruction technique using multiple partial shapes acquired from a digital stereo camera. We have introduced new tech-

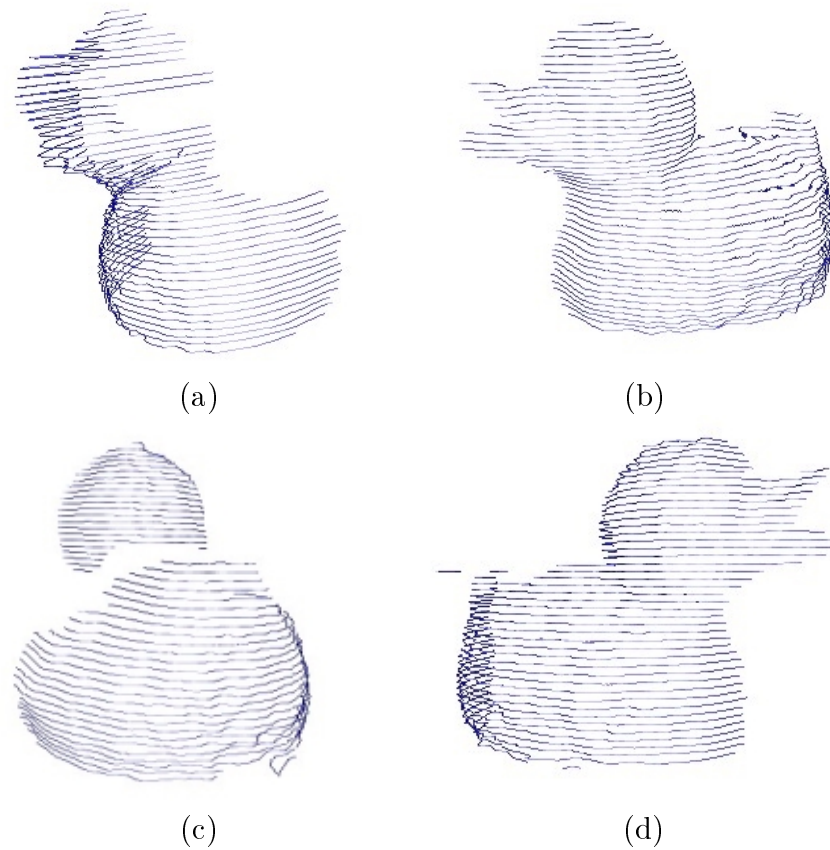


Figure 4.10: Four partial views of the object 'Duck'. Shapes are represented by a set of horizontal contours: (a) 0 degree (b) 90 degree (c) 180 degree (d) 270 degree

niques for registration and integration of partial 3D models. The algorithms exploit the epipolar geometry between different views. Correspondence between 3D points are established by projecting overlapping contour segments from one view to another view and finding the closest point on the epipolar line.

Integration is done slice by slice using linked lists representing points and segments of contours of object's cross-sections. The linked lists for different views are merged based on two closeness criteria to obtain closed contours representing complete cross-sections of the object. Meshes are created from the connected contours to generate a complete model of the object. Experimental results for three real objects are presented.

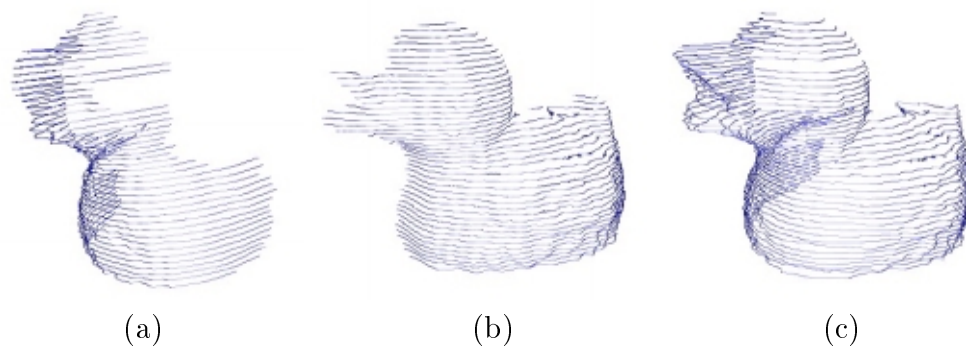


Figure 4.11: Pairwise merging of two partial shapes: (a) Shape from 0 degree (b) Shape from 90 degree (c) Merged contours

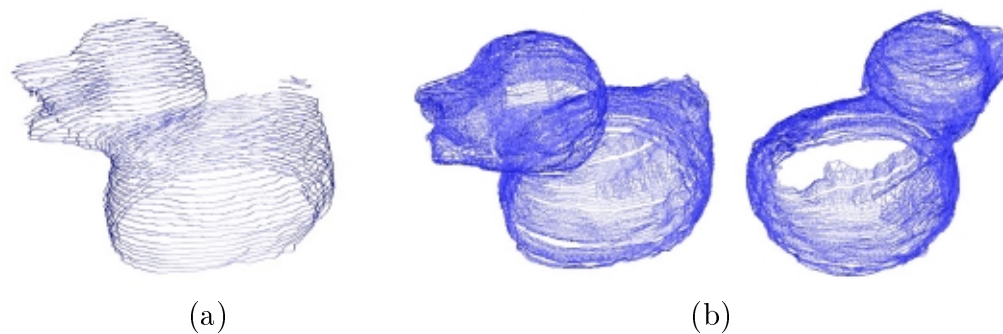


Figure 4.12: Integrated model of the object 'Duck': (a) Contour model (b) Integrated mesh models

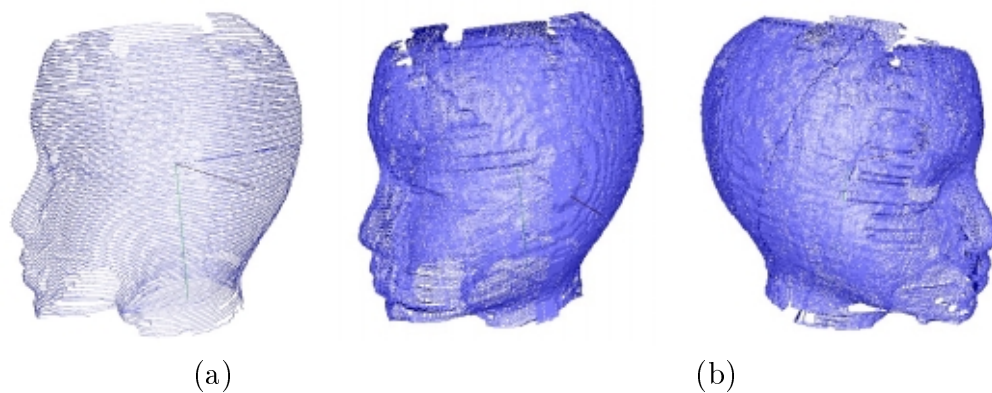


Figure 4.13: Integrated model of the object 'Head': (a) Contour model (b) Integrated mesh models

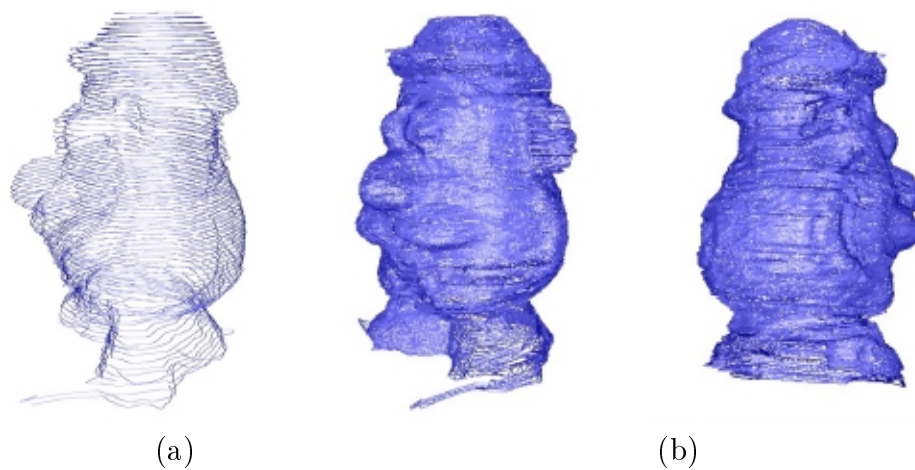
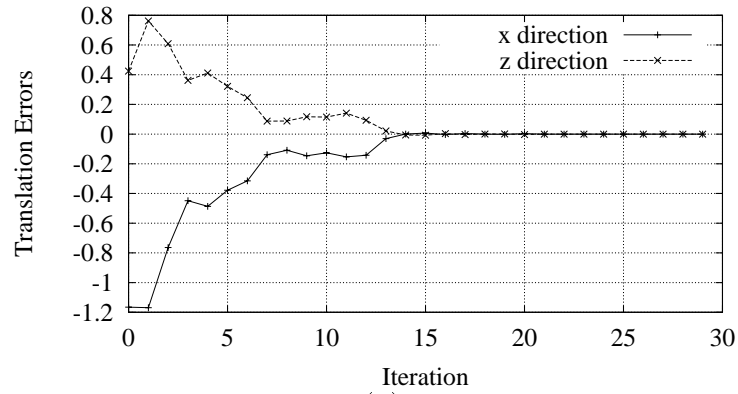
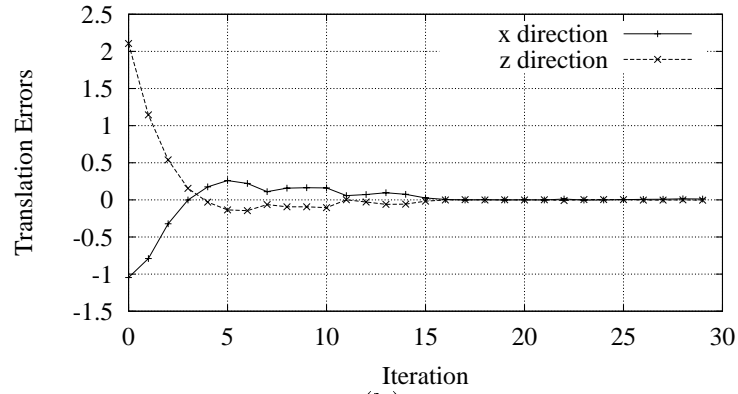


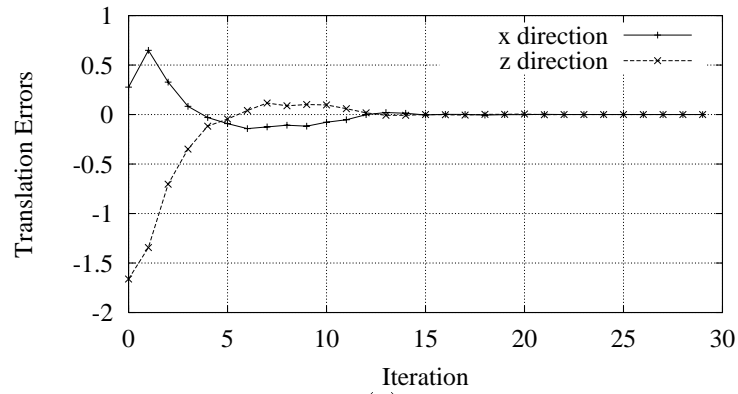
Figure 4.14: Integrated model of the object 'Potatohead': (a) Contour model (b) Integrated mesh models



(a)



(b)



(c)

Figure 4.15: Translation errors in the X and Z directions for objects (a) Duck (b) Head (c) Potatohead

Chapter 5

Volumetric 3D Reconstruction

This chapter presents a volumetric 3D reconstruction technique for registration and integration of multi-view range images. This technique is based on estimation of the signed distance of a voxel element to the object's outer surfaces. This estimation provides an implicit representation of the object's surfaces, which can be polygonized to a mesh model by the Marching Cubes algorithm.

5.1 Introduction

Volumetric 3D reconstruction has been a common technique for generating 3D models in *Computer Vision* and *Computer Graphics*. In contrast with surface-based technique, in volumetric technique it is topologically easy to reconstruct complex objects. In computer graphics, the term *volumetric* means a representation of both the outer surfaces and the enclosed space of a 3D model in a sampled 3D space. Examples are Space Carving and Voxel Coloring. In contrast, in computer vision volumetric is used to represent outer surfaces of the reconstructed model. Octree [15] and Rectangular Parallelepiped [1] are two examples.

However, in this thesis, the term is used only to refer to a 3D reconstruction technique which samples a 3D space on a grid of voxels to obtain an implicit representation of the object. Implicit representation is a very efficient technique to represent a 3D model in a volumetric space. The surfaces of an object are represented implicitly by the signed distances of all voxels to the object's surfaces. A mesh model of the object is then easily reconstructed by a well-known polygonization technique such as Marching Cubes algorithm [11].

In this chapter, multi-view range images are registered to a common coordinate system and their registration parameters are refined by a point-to-tangent plane refinement technique. After the registration, signed distances of voxels which are close to object's surfaces are computed by incorporating range images and their projections on image plane. The signed distance of a voxel is computed by the weighted average

of all signed distances to overlapping surfaces with which the voxel is associated. The Marching Cubes algorithm polygonizes the implicit space into a single mesh to represent a 3D model of the object. Voxel space is classified into four different regions to reduce computation error of signed distance.

Shape-from-silhouettes technique is also incorporated to carve the volumetric 3D space from silhouettes of the multiple view points. Therefore, signed distance of a voxel is computed only when it is inside of the Visual Hull of the object. Experimental results for several real objects are presented to analyze registration errors and to show mesh representations of the objects. In the final section, we analyze reconstruction error of our volumetric technique by generating two real ground-truth objects.

5.2 Registration of Multiple Range Images

Multiple range images obtained by rotating an object should be registered into a common coordinate system before integration. Calibration parameters of the vision system are used for an initial registration of the range images. Registered range image is represented by a triangle mesh by polygonizing the range image. After initial registration, it needs to be refined to reconstruct an accurate 3D model.

As mentioned earlier, there are two main approaches to registration, *point-to-point* and *point-to-tangent plane* approaches. This chapter adopts a *point-to-tangent plane* approach to minimize the registration error because it is known to be more accurate than *point-to-point* technique. We also estimate optimal calibration parameters for our vision system based on the refined parameters.

5.2.1 Multi-view Registration

The camera calibration and center of rotation parameters are used for an initial registration. The center of rotation is calibrated by Tsai's algorithm as presented in Chapter 2. Figure 5.1 shows the geometry of the SVIS-2 vision system. We consider the right camera coordinate system of i th view as the corresponding view point \mathcal{V}_i and the first view point \mathcal{V}_0 as the world and common coordinate system. Given a stereo correspondence \mathbf{p}_l and \mathbf{p}_r in stereo image planes, a 3D point $\mathbf{P} = (P_x, P_y, P_z)^T$ is represented as

$$\mathbf{P} = \left(\frac{p_{rx}}{f} P_z, \frac{p_{ry}}{f} P_z, f \frac{B}{(p_l - p_r)} \right)^T \quad (5.1)$$

where p_{rx} and p_{ry} are x and y coordinates of the point \mathbf{p}_r in the right image plane. All undistorted image points are calibrated by Tsai's algorithm. A partial shape of i th view, \mathcal{S}_i and $i = 1, \dots, N$, is registered to the common coordinate system \mathcal{V}_0 using

the calibration parameters. If there is a point \mathbf{P}_i from \mathcal{V}_i , it is registered to \mathcal{V}_0 with a new point \mathbf{P}_i^0 as

$$\mathbf{P}_i^0 = \mathbf{R}_i^0(\mathbf{P}_i - \mathbf{t}_s) + \mathbf{t}_s, \quad (5.2)$$

$$= \mathbf{T}_i^0 \mathbf{P}_i. \quad (5.3)$$

where \mathbf{R}_i^0 is the rotation matrix from \mathcal{V}_i to \mathcal{V}_0 , and \mathbf{t}_s is the translation vector from the origin of \mathcal{V}_0 to the center of the rotation stage coordinate system.

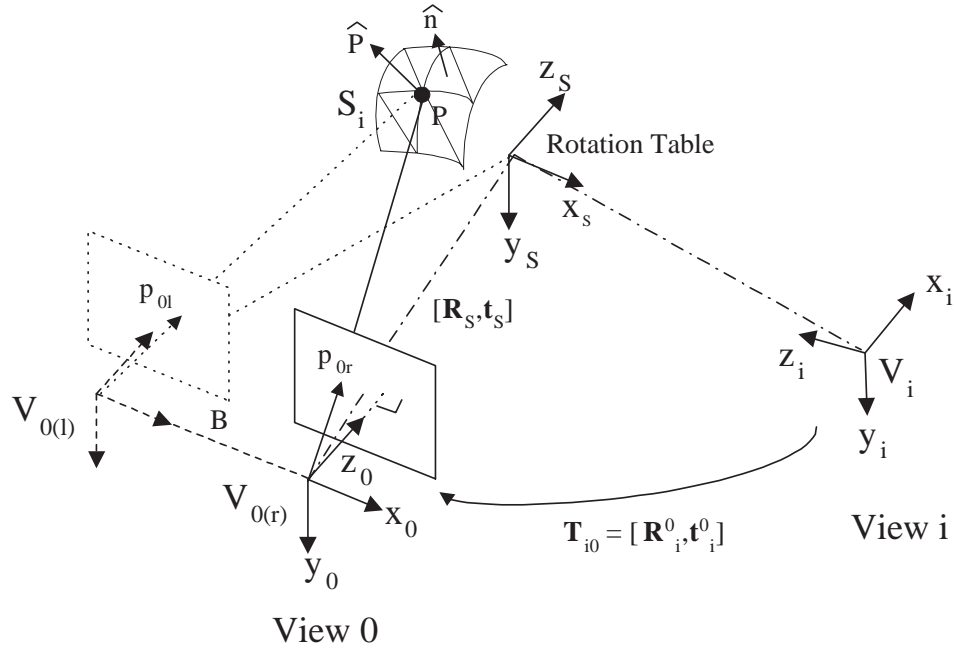


Figure 5.1: SVIS-2 vision system geometry. The coordinates of the right camera of the first view, $\mathcal{V}_{0(r)}$ is defined as the common coordinate system

Registration refinement minimizes registration error due to inherent errors in initial calibration parameters. A refinement technique finds correspondences between two or more overlapping partial shapes to minimize the transformation error between the correspondences. In general, geometric features on partial shapes are used to decide the correspondences between views. There are also techniques using image features on object's texture images, but we describe geometric feature matching on this section.

Let us consider a problem of refining two partial shapes, which is called as pairwise registration. If correspondences between two partial shapes are decided, we

call the corresponding points as control points. Suppose we have two control point sets $\{\mathbf{P}_i\}$ on a partial shape \mathcal{S}_i , and $\{\mathbf{Q}_j\}$ on another partial shape \mathcal{S}_j . Then the refinement procedure computes a rigid transformation matrix \mathbf{T}_j^i which minimizes overlapping errors. The transformation matrix \mathbf{T}_j^i is computed by using a least-squares-minimization technique.

Chen and Medioni [17] proposed a point-to-tangent plane registration technique. They find an intersecting tangent plane on a target surface from a control point from a source surface. Bergevin et al.[7] modify Chen and Medioni’s algorithm to find an intersecting point by interpolation of grid points. In addition, they propose a multi-view registration algorithm. It simultaneously minimizes registration errors in all overlapping regions.

Our approach is similar to the Bergevin’s algorithm. Suppose there are two partial shapes \mathcal{S}_i and \mathcal{S}_j as shown in Figure 5.2. To register two partial shapes, we first establish a set of control points $\{\mathbf{P}_i\}$ on the surface \mathcal{S}_i by sampling vertices on surface and testing their reliability. Let us consider a point $\mathbf{P}_i \in \{\mathbf{P}_i\}$. From its normal vector $\hat{\mathbf{P}}_i$, an intersecting point \mathbf{Q}'_j is decided by interpolating three vertices of an intersecting triangle on surface \mathcal{S}_j . Then we find another point \mathbf{Q}_j on the tangent plane at \mathbf{Q}'_j , which is the projection of \mathbf{P}_i to the plane. This point is saved as the corresponding control point. By repeating this searching for all points in $\{\mathbf{P}_i\}$, we get its corresponding point set $\{\mathbf{Q}_i\}$.

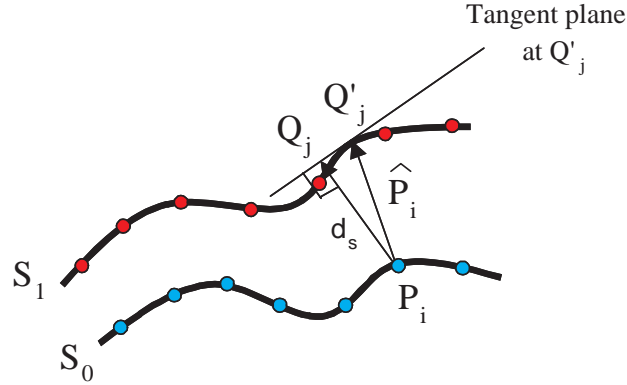


Figure 5.2: Registration of two partial shapes. From a control point \mathbf{P}_i , its corresponding control point \mathbf{Q}_j on the other surface is decided on the tangent plane at \mathbf{Q}'_j

In order to select reliable control points in the first view, we sample vertices and

check them according to following conditions. Suppose the first view index is i and the second view index is j , then a vertex \mathbf{P}_i should satisfy the following conditions.

- Angle between vertex normal $\hat{\mathbf{P}}_i$ and view normal $\hat{\mathbf{z}}_i$ is less than $\frac{\pi}{3}$.
- Angles between vertex normal $\hat{\mathbf{P}}_i$ and face normal vectors of K surrounding triangles $\hat{\mathbf{n}}_k$ with $k=0,\dots,K$, are less than $\frac{\pi}{3}$.

Finding an intersecting triangle and an intersecting point with a vertex normal $\hat{\mathbf{P}}_i$ requires a lot of computations. In order to reduce computation complexity, we reject triangles on the second view during search procedures, which satisfy the following conditions.

- Angle between face normal $\hat{\mathbf{n}}_j$ and view normal $\hat{\mathbf{z}}_i$ is greater than $\frac{\pi}{2}$.
- Angle between $\hat{\mathbf{n}}_j$ and view normal $\hat{\mathbf{z}}_j$ is greater than $\frac{\pi}{3}$.
- Angles between vertex normal $\hat{\mathbf{P}}_i$ and face normal $\hat{\mathbf{n}}_j$ is greater than $\frac{\pi}{2}$.
- Distance from \mathbf{P}_i to a vertex of the second view \mathbf{Q}_j is greater than a predefined threshold, d_{TH} .

Two control points sets $\{\mathbf{P}_i\}$ and $\{\mathbf{Q}_j\}$ are used for estimating the rigid transformation matrix between two surfaces. After finding all control points, we compute the transformation matrix \mathbf{R} and \mathbf{t} using a least-squares-minimization technique through the singular value decomposition (SVD) [4]. SVD finds a least-squares solution for \mathbf{R} and \mathbf{t} , which minimizes an overlapping error ϵ such that

$$\epsilon = \sum_{k=1}^K \|\mathbf{Q}_k - (\hat{\mathbf{R}}\mathbf{P}_k + \hat{\mathbf{T}})\|^2. \quad (5.4)$$

Registration error ϵ is measured every iteration to check the convergence of two surfaces. In real experiments we define two thresholds for each rotation and translation errors, ϵ_R and ϵ_t respectively. If the errors are less than the defined thresholds, we stop the refinement.

It is well-known that registration errors accumulate if multi-view partial shapes are registered in pairwise. Therefore, it is necessary to register all multiple partial shapes together in order to evenly distribute the registration error in all overlapping regions. We set the first view \mathcal{V}_0 as the reference frame of multi-view registration [7]. Bergevin et al. search all overlapping partial shapes to find intersecting planes from a control point. In our approach we do not search all partial shapes, because of computation time, but search adjacent partial shapes. For example, in order to

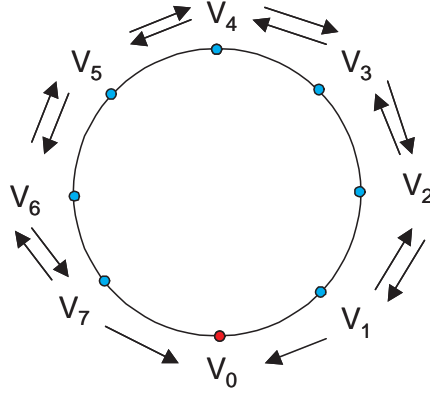


Figure 5.3: Multi-view registration strategy. From a control point in view point \mathcal{V}_i , we search only surfaces of views \mathcal{V}_{i-1} and \mathcal{V}_{i+1}

register a partial shape \mathcal{S}_i , we search intersecting points only on the partial shapes \mathcal{S}_{i-1} and \mathcal{S}_{i+1} as shown in Figure 5.3. This approach works fine because a view point \mathcal{V}_i has most overlappings with its adjacent views \mathcal{V}_{i-1} and \mathcal{V}_{i+1} .

Registration errors ϵ between all overlapping shapes decrease while the multi-view registration refines all partial shapes iteratively. In our experiments, after about 5 or 6 iterations, registration errors reduce below of the predefined thresholds for ϵ_R and ϵ_t . Then we stop the registration refinement.

5.2.2 Calibration Parameter Optimization

As mentioned earlier, we initially register all partial shapes into the common coordinate system \mathcal{V}_0 according to their calibrated transformations. Therefore, the calibration parameters are very important for initial registration. If the initial parameters are very close to the optimal values, refinement procedure will run fast.

Equation 5.2 shows that there are two transformation matrices for partial shape registration. \mathbf{R}_{i0} is the rotation matrix from \mathcal{V}_i to \mathcal{V}_0 . Because rotation angle of the turntable is accurately controllable, we can consider the initial parameters of $\mathbf{R}_{\mathbf{S}}$ are reliable. However let us consider the translation vector $\mathbf{t}_{\mathbf{s}}$ from the camera's origin to the rotation center. We know the step number of the linear translation stage, which corresponds to the position of aligning the origin of the camera coordinate system with the origin of the rotation stage coordinates. However, it was determined by aligning the center of image coordinates with the rotation center manually. And we

know that this parameters \mathbf{t}_s is the main source of registration error. Therefore, we refine the translation matrix \mathbf{t}_s to find the optimum.

Figure 5.4 shows a 2D plot of partial shapes between \mathcal{V}_0 and \mathcal{V}_1 . Let \mathbf{t}_s be a calibrated vector from the origin of \mathcal{V}_0 to the rotation center. If this is the ideal vector to the rotation center, a registered vector to \mathcal{V}_1 , $\mathbf{t}_s^{(1)}$ should be the same as \mathbf{t}_s with respect to the common coordinate system. However, in real situation, there may be an error between \mathbf{t}_s and $\mathbf{t}_s^{(1)}$. If a vector \mathbf{t}_s' is the ideal vector to the center of rotation, then

$$\begin{aligned} \mathbf{t}_s^{(1)} &= \mathbf{R}_1^0(\mathbf{t}_s - \mathbf{t}_s') + \mathbf{t}_s' \\ \mathbf{t}_s' &= (\mathbf{I}_D - \mathbf{R}_0^1)^{-1}(\mathbf{t}_s^{(1)} - \mathbf{R}_0^1), \\ &\text{where, } \mathbf{t}_s^{(1)} = \mathbf{T}_0^1 \mathbf{t}_s, \end{aligned} \quad (5.5)$$

and I_D is the identity matrix. Using this equation, we estimate \mathbf{t}_s' between \mathcal{V}_0 and \mathcal{V}_1 . For the other views, we also estimate \mathbf{t}_s' and average all of them for the next iteration. After several experiments, we estimate the optimal value of the \mathbf{t}_s and use it in all other experiments.

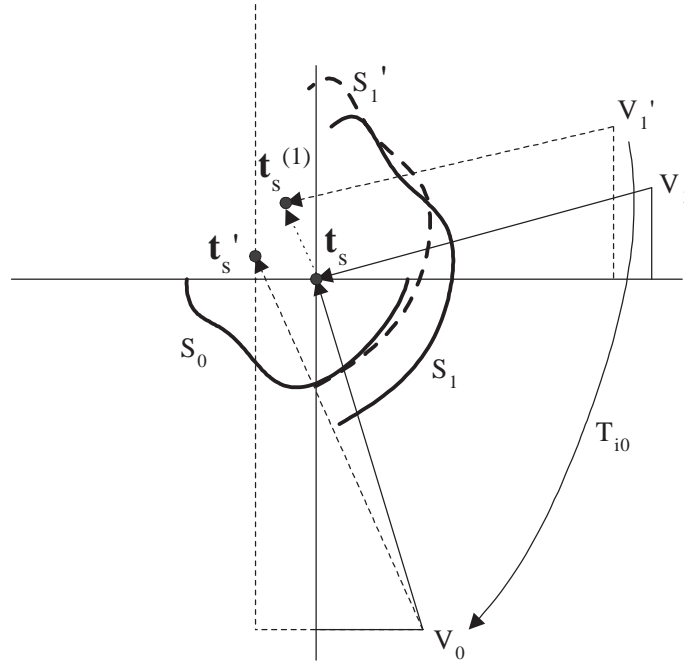


Figure 5.4: Rotation center refinement

5.3 Multi-view Integration

A volumetric integration technique is employed to reconstruct a 3D model of an object. Volumetric technique for shape integration is widely used in 3D modeling. It is topologically simple and robust compared to surface-based integration techniques. Converting volumetric space to a surface model is implemented by a surface polygonizing algorithm. Given multiple partial shapes which are registered as described in previous section, we find the iso-surface of the object in a grid of voxels, and convert it to a 3D mesh model using the Marching Cubes algorithm [11, 56]. In order to save memory space, we introduce a continuation approach of the algorithm through a classification of the voxel space into multiple regions based on signed distances from a voxel to its overlapping surfaces.

As presented in an earlier chapter, range images of an object are acquired by a stereo vision technique. However, stereo vision has inherent problems such as stereo mismatching and occlusion. In order to remove and reduce the distortion caused by erroneous points, we combine a *shape-from-silhouettes* technique. *Shape-from-silhouettes* technique is used to remove erroneous points outside of the object’s *visual hull*, because silhouettes are very dominant image features and are reliable.

5.3.1 Surface Representation

Implicit surface representation is widely used for volumetric shape reconstruction [18, 34, 103]. An object is represented by the signed distances from a grid of voxels to the object’s surface. Suppose a voxel \mathbf{P} in 3D space as shown in Figure 5.5, the signed distance $f(\mathbf{P})$ from the voxel to the object’s surface is computed by taking the dot product of two vectors

$$f(\mathbf{P}) = (\mathbf{P} - \mathbf{P}_s) \cdot \hat{\mathbf{n}} \tag{5.6}$$

where \mathbf{P}_s is the range of the voxel from the origin of a view point \mathcal{V}_i to a 3D point on the object’s surface \mathcal{S} . Here, the point is on the same projection line with the vector \mathbf{P} . And $\hat{\mathbf{n}}$ is a unit normal vector of the object surface at \mathbf{P}_s . According to Equation (5.6), the sign of a voxel is positive when it is outside of an object, but it is negative otherwise.

In order to get the vector \mathbf{P}_s , we use a range image which is obtained from i th view point. However, because the range image is a discrete range of the object surface, we have to use an interpolation technique to get an accurate range to \mathbf{P}_s . For example in this figure, we project back the point \mathbf{P} to an image point \mathbf{p}_s on 2D range image by using the perspective projection matrix \mathbf{M} of the camera system such that

$$\mathbf{p}_s \cong \mathbf{M}\mathbf{P}. \tag{5.7}$$

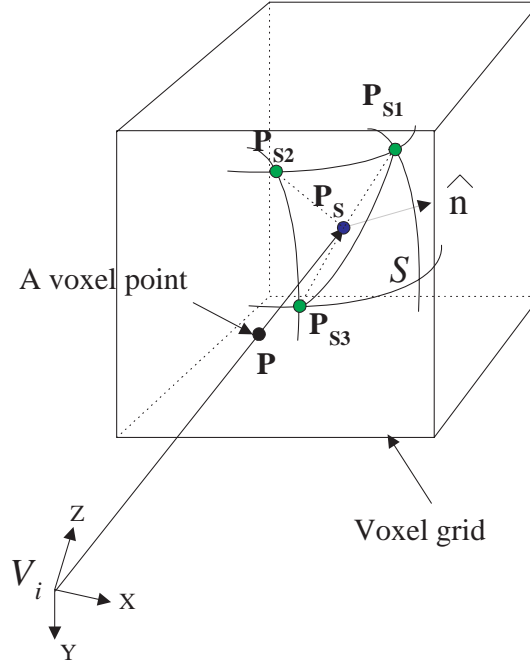


Figure 5.5: Signed distance from a voxel \mathbf{P} to a object's surface point \mathbf{P}_s is computed by the dot product of $(\mathbf{P} - \mathbf{P}_s)$ and the surface normal $\hat{\mathbf{n}}$

The image point \mathbf{p}_s should be in a triangle which encloses the point. Let us the three vertices be $\mathbf{p}_{s1}, \mathbf{p}_{s2}$, and \mathbf{p}_{s3} . And Let us three vectors associated with the vertices be $\mathbf{P}_{s1}, \mathbf{P}_{s2}$, and \mathbf{P}_{s3} . Then we can compute the vector \mathbf{P}_s by interpolating the other vectors such that

$$\mathbf{P}_s = d_1 \mathbf{P}_{s1} + d_2 \mathbf{P}_{s2} + d_3 \mathbf{P}_{s3}, \quad (5.8)$$

$$\text{where } d_i = \frac{\|\mathbf{p}_{si} - \mathbf{p}_s\|}{\sum_j \|\mathbf{p}_{sj} - \mathbf{p}_s\|}.$$

For integration of partial shapes, the signed distance from a voxel to the object's surface is estimated by averaging weighted signed distances to all overlapping shapes. If there are N_o overlapping surfaces from the voxel, N_o signed distances from the voxel to all overlapping surfaces are measured and the weighted signed distance is estimated by averaging all of them. Suppose there is a voxel \mathbf{P} in a 3D grid of voxels as in Figure 5.6. From the voxel, if three surfaces are very close in distance, the weight of depth measure for S_i surface is defined as $w_i(\mathbf{P}) = \hat{\mathbf{P}}_i \cdot \hat{\mathbf{n}}_i$, where \mathbf{P}_i is the representation of the voxel \mathbf{P} by the coordinate system of view \mathcal{V}_i and $\hat{\mathbf{P}}_i$ is its

normal. And the signed distance of the voxel is

$$F(\mathbf{P}) = \frac{\sum w_i(\mathbf{P})f_i(\mathbf{P})}{\sum w_i(\mathbf{P})} \quad (5.9)$$

where $f_i(\mathbf{P})$ is the signed distance from the voxel \mathbf{P} to the i th partial shape.

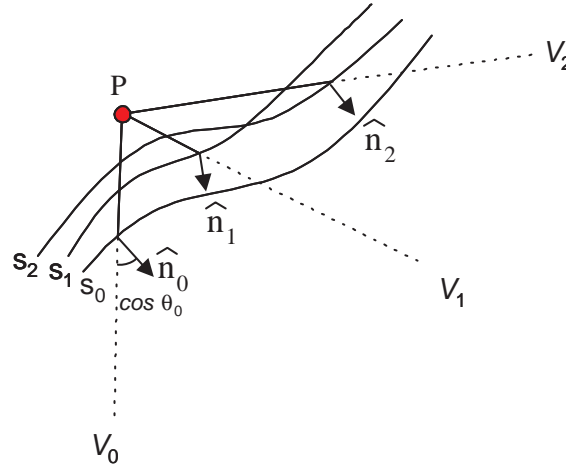


Figure 5.6: Overlapping surfaces from a voxel \mathbf{P} . The signed distance of the voxel is computed by a weighted averaging of the distances to all surfaces.

If a vision system has N multiple views, N signed distances from a voxel to all partial shapes can be measured and the weighted signed distance is estimated by averaging all of them. However, if the signed distances from the voxel to some surfaces are too long or if there are inconsistency of the signs of the voxel, the distances to those surfaces should not be involved for computing the weighted average. A graphical example is shown in Figure 5.7. In this figure a voxel \mathbf{P} has some neighboring surfaces. However, distances to some surfaces, for example d_2 and d_i in the figure, are too long for the surfaces to be considered as overlapping ones. This means those surfaces should not contribute to distance averaging.

When a voxel is in an overlapping area where some surfaces overlap each other, the magnitude of signed distances from the voxel to the overlapping surfaces are usually very small. But it is not to other surfaces which are far from the voxel. Therefore, it is necessary to threshold signed distance to decide overlapping surfaces from the voxel. As in Figure 5.7, we set a threshold d_{TH} to compute the final signed distance $F(\mathbf{P})$ only using the overlapping shapes. However, if there are some erroneous points

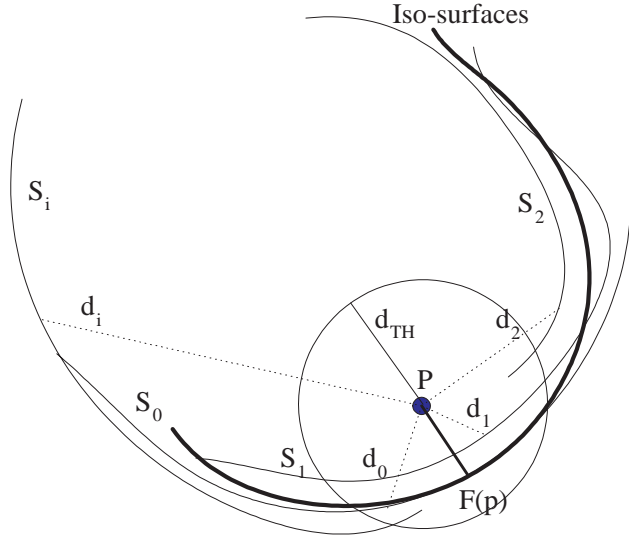


Figure 5.7: Selecting valid partial surface for signed-distance averaging. From a voxel \mathbf{P} , some surfaces to which the signed distance should not be involved for distance averaging, if their distance is too long.

on partial surfaces, computation of the signed distance may not be an accurate distance to the 3D model. An effective method to avoid this problem is to apply a *shape from silhouettes* technique to remove erroneous voxels by volume intersections.

5.3.2 Volume Intersection

Volume intersection technique using multiple silhouettes of an object is very useful for removing erroneous voxels in volumetric reconstruction. This is because silhouettes are often dominant image features and reliable. An example of silhouette generation is in Figure 5.8(a). From a camera’s view point, the silhouette of the view is a 2D binary image which consists of two regions, object area and background. If all multiple view points are outside of the convex hull of the object \mathbf{O} , intersections of all cones of views generate the object’s visual hull $\mathbf{VH}(\mathbf{O})$ as shown in Figure 5.8(b). In order to reconstruct $\mathbf{VH}(\mathbf{O})$ as accurate as possible to the object \mathbf{O} , it usually needs a lot of object’s silhouettes. However, because we have a limited number of

silhouettes, the reconstructed visual hull contains not only the object \mathbf{O} but also $\bar{\mathbf{O}}|_{\mathbf{VH}}$ such that

$$\bar{\mathbf{O}}|_{\mathbf{VH}} = \{\mathbf{p} | \mathbf{p} \in \mathbf{VH}(\mathbf{O}) \cap \bar{\mathbf{O}}\}, \quad (5.10)$$

where $\bar{\mathbf{O}}$ is space outside of the object \mathbf{O} . In our integration step, all partial surfaces are considered to be within overlapping range L as shown in the figure. It is inside of the visual hull $\mathbf{VH}(\mathbf{O})$, however it contains both voxels $\mathbf{P} \in \bar{\mathbf{O}}$ and $\mathbf{P} \in \mathbf{O}$. Therefore, it is necessary to find the exact object's surface O by integrating the partial shapes. In this thesis, volume intersection is mainly used for removing erroneous surface due to stereo mismatches which lie outside of $\mathbf{VH}(\mathbf{O})$. This technique is also used for the integration of two 3D pose models of the object, which will be described later.

5.3.3 Partial Shapes Integration

Integration of partial shapes begins with finding a seed voxel which is a starting point of the Marching Cubes (MC) algorithm [56]. From the seed voxel, the algorithm computes the weighted signed distances of all voxels which are close to the model's surface, approximately in the region L in Figure 5.8(b). The algorithm automatically converts a grid of voxels to triangle meshes based on signed distances. In this thesis, we suppose that voxels outside of object have (+) sign and inside voxels have (-) sign. If a voxel is outside of visual hull, we assign an arbitrary positive number as its signed distance. Therefore, it only needs to compute the signed distance of voxels inside of $\mathbf{VH}(\mathbf{O})$.

Signed distance of a voxel is computed by averaging weighted signed distances to all overlapping shapes from the voxel. Determining overlapping shapes from the voxel is based on the distances to the shapes. If there are N partial shapes, we can measure the signed distances of the voxel for all N surfaces. However, only some of them contribute to computing the signed distance. In order to select partial shapes to compute the signed distance of a voxel, we introduce a *voxel code* which is consistent and reliable with the system geometry. Let's first define the maximum number of overlapping shapes and the corresponding views. Because we use a rotation stage which rotates an object by $360/N$ angle, we set the number of the maximum overlapping shapes to $N_o = N/2$.

Each voxel which is inside of $\mathbf{VH}(\mathbf{O})$ is coded by a N bit binary code as shown in Figure 5.9. We assign one bit to each view. If a signed distance $f(\mathbf{P})$ of a voxel \mathbf{P} to \mathcal{S}_i is negative, we assign 0 to the corresponding bit, otherwise, we assign 1. Therefore, 0 at the i th bit means the voxel is behind \mathcal{S}_i and 1 means it is between \mathcal{S}_i and \mathcal{V}_i . According to the *voxel code* $VC(\mathbf{P})$ of a voxel \mathbf{P} , we determine N_0 number of signed distances to compute the weighted signed distance of the voxel. Because

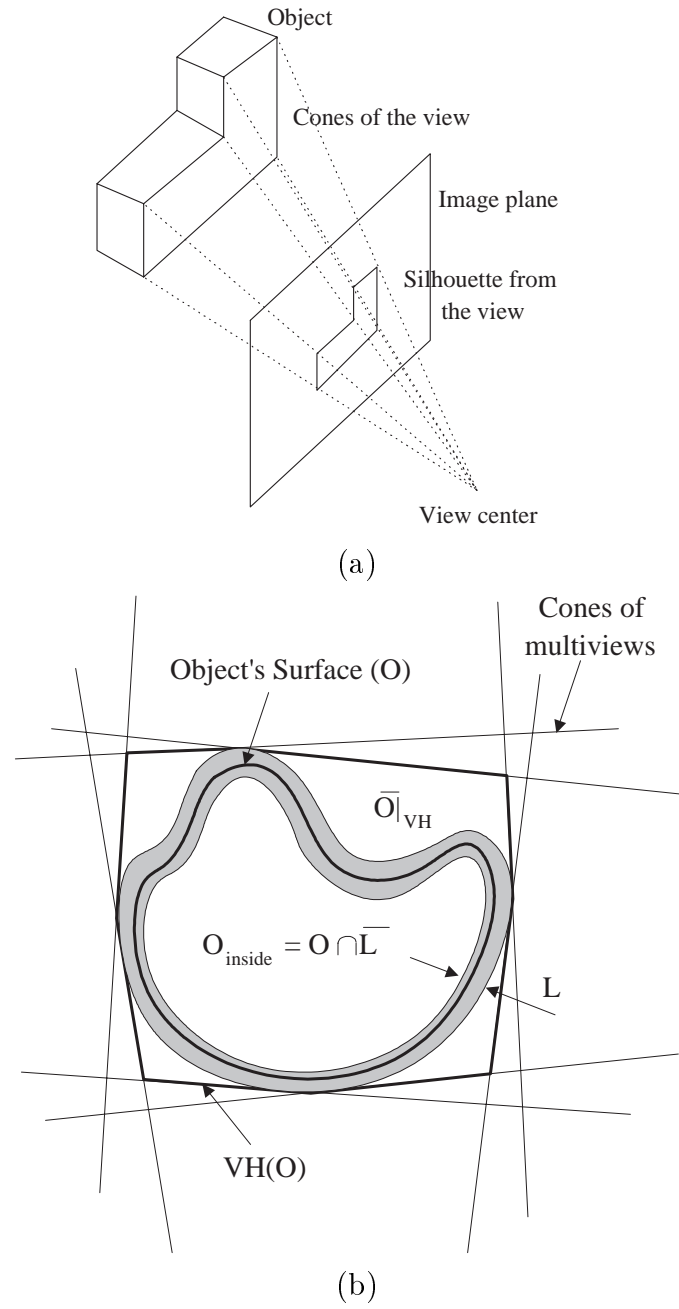


Figure 5.8: Reconstruction using *Shape-from-silhouettes* technique. (a) An example of a silhouette from a single view. (b) Visual hull of an object reconstructed by volume intersections.

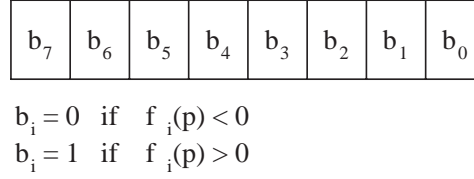


Figure 5.9: Binary voxel code for 8 multi-views

when we estimate the weighted distance from the voxel to object's surface, we have to reject some distances which are extensions of other surfaces.

Selecting reliable signed distance from a voxel is based on the voxel code of \mathbf{P} . Let's define reliable signed distance $d_i(\mathbf{P})$, where $i = 0 \cdots N_0 - 1$. Since two distances $f_i(\mathbf{P})$ and $f_{i+N_0}(\mathbf{P})$ are measured always from two opposite viewing directions, selecting one of them as a signed distance is very reliable. Therefore, distance $d_i(\mathbf{P})$ is selected from one of them according to the condition of two bits b_i and b_{i+N_0} . The two bits can have total 4 conditions as shown in Figure 5.10. The figure shows 4 different conditions of two bits of voxel code. According to the figure, we make three strategies for selecting the reliable distance.

- Both b_i and b_{i+N_0} are 1: select the shorter distance.
- Either b_i or b_{i+N_0} is 1: select the distance with positive sign, because positive distance is more reliable than negative one.
- Both b_i and b_{i+N_0} are 0: select the shorter distance.

When either b_i or b_{i+N_0} is 1, we select the positive distance, because positive distance to surface means that it is seen directly from the view point, but not for negative one. For negative distance, sometimes it is due to occlusion of a surface. Therefore, positive distance is more reliable.

Let us have N_0 reliable signed distances of the voxel by the technique above. Next step is computing the weighted signed distance of the voxel. In order to compute the weighted distance, we classify the voxel space into four regions, as shown in Figure 5.11, based on distances $d(\mathbf{P})$ from a voxel \mathbf{P} to partial shapes.

- If at least two distances $|d_i(\mathbf{P})|$ are shorter than d_{TH} , where $i = 0, \dots, N_0 - 1$, a voxel \mathbf{P} is considered to be in an overlapping region L , and $\mathbf{P} \in P_{overlap}$.
- If all $d_i(\mathbf{P})$ have positive sign and $|d_i(\mathbf{P})| > \mathbf{d}_{TH}$ for N_0 distances, the voxel is totally outside of object, and $\mathbf{P} \in P_{outside}$.

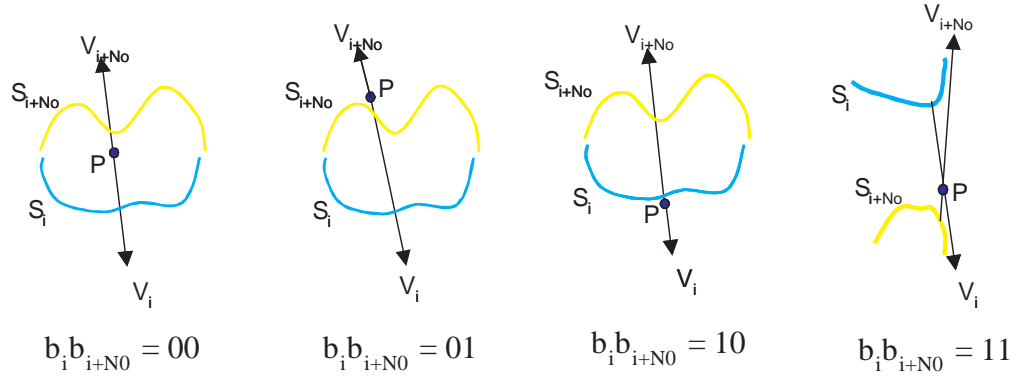


Figure 5.10: Examples of combinations of two bits voxel code.

- If all $d_i(\mathbf{P})$ have negative sign and $|d_i(\mathbf{P})| > \mathbf{d}_{\text{TH}}$ for N_0 distances, the voxel is totally inside of object, and $\mathbf{P} \in P_{\text{inside}}$.
- Else, we assume that the voxel is in non-overlapping area, and $\mathbf{P} \in P_{\text{nonoverlap}}$.

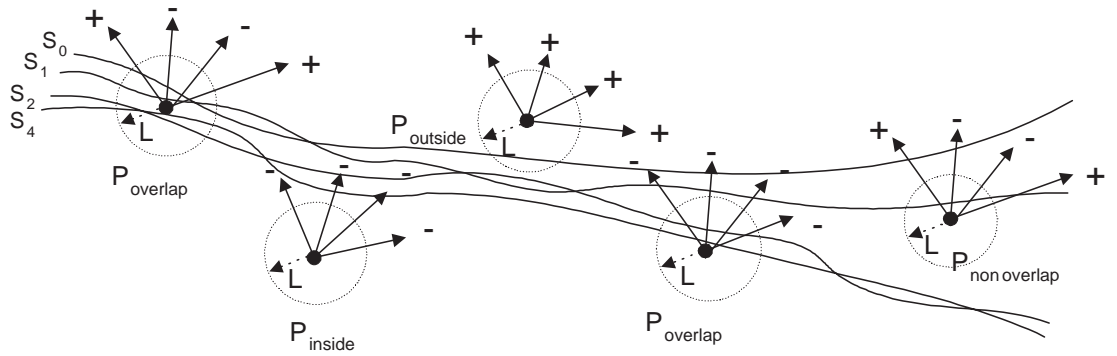


Figure 5.11: Voxel classification. Arrows from each voxel correspond to vectors from the voxel to visible view origins

Most voxels which are close to model's surface are in P_{overlap} . Therefore, they can have all possible signs as their distances. On the contrary, P_{inside} can have only negative sign and P_{outside} can have only positive sign as shown in Figure 5.11. Continuation approach of the MC algorithm computes weighted distances of voxels,

which are usually in $P_{overlap}$. However, if there are some occlusions or errors on partial shapes, a voxel can be classified as P_{inside} or $P_{nonoverlap}$. Based on the description above, we compute the weighted signed distance $D(\mathbf{P})$ of a voxel \mathbf{P} . Below is a pseudo-code of the algorithm. where, $wd_i(\mathbf{P})$ is the weight value corresponding to

Pseudo-codes for registration refinement

```

for  $i = 0$  to  $N_0 - 1$ ;
  if ( $b_i \parallel b_{i+N_0}$ )
    if ( $b_i$ )  $d_i \leftarrow f_i$ ;
    else  $d_i \leftarrow f_{i+N_0}$ ;
  else
    if ( $|f_i| < |f_{i+N_0}|$ )  $d_i \leftarrow f_i$ ;
    else  $d_i \leftarrow f_{i+N_0}$ ;
for  $i = 0$  to  $N_0 - 1$ ;
  if ( $d_i < d_{TH}$ )
     $W(\mathbf{P}) += wd_i(\mathbf{P})$ ;
     $D(\mathbf{P}) += wd_i(\mathbf{P})d_i(\mathbf{P})$ ;
     $overlap\_cnt ++$ ;
  if ( $\mathbf{P} \in P_{overlapping}$ )
     $D(\mathbf{P}) = \frac{D(\mathbf{P})}{W(\mathbf{P})}$ 
  else if (( $\mathbf{P} \in P_{inside}$ ) or ( $\mathbf{P} \in P_{outside}$ ))
     $D(\mathbf{P}) = \min_i \{|d_i(\mathbf{P})|\}$ 
  else
     $D(\mathbf{P}) = \min_i \{|d_i(\mathbf{P})|\}$  , if  $wd_i(\mathbf{P}) > 0.4$ 

```

the distance $d_i(\mathbf{P})$ and d_{TH} is the threshold of the magnitude of signed distance. When \mathbf{P} is neither in $P_{overlapping}$, P_{inside} , nor $P_{outside}$, partial shapes do not form a surface. Therefore, we set a threshold for weight value to get a more reliable surface. After the integration, we get a 3D model of the object which is represented by a closed triangle mesh.

5.4 Experimental Results

We have tested our registration and integration techniques on several real and complex objects. Each object is placed on a turntable and $N = 8$ different range images are obtained by the SVIS-2 vision system. We use a Pentium-III 1GHz processor.

Range image has 320×240 resolution and it is converted to a triangle-mesh model by a simple polygonization algorithm. For every unit square region in the range image, which consists of 4 pixels on its corners, 2 triangles are reconstructed by dividing the square. The square is divided by a common edge that connects two diagonal pixels. By comparing length of two possible edges in the square, two triangles are reconstructed by the shorter edge.

Figure 5.12 shows all test objects in this chapter. The actual size of image is 1280×960 in TIFF image format. If an object has high contrast texture on its surface, we use texture images for stereo matching. Otherwise, we picture another set of images for stereo matching while projecting a random dot pattern on the surface to introduce high contrast texture.

5.4.1 Registration Results

The multi-view registration technique described in the earlier section is applied to register all objects. Translation error is measured by averaging distance d_s as in Figure 5.2 for all control points. Control points are selected uniformly on a partial shape and their validities are checked as described in section 5.2.1.

Figure 5.13 shows all 8 partial views of the 'Monkey' object. There are some errors on each surface due to stereo mismatching. This test shows 20 iterations of registration, but usually we stop the registration when it is less than 10 iterations. The main problem of registration is computation complexity. From a control point on a partial surface, we find an intersecting triangle on another surface by testing all triangles on the surface. Without any fast-searching technique, 10 iterations take about 2 minutes for a pairwise registration and about 15 minutes for multi-view registration. Registration step takes most time in our 3D model reconstruction. To reduce the time for 3D reconstruction, we introduce a fast registration technique in Chapter 8.

Figure 5.15 shows registration errors in 'Monkey' object. Translation error is average of the distance d_s between control point sets \mathbf{P} and \mathbf{Q} . Rotation error is measured by summation of each elements of the matrix $\mathbf{I} - \mathbf{R}$, where \mathbf{I} is the identity matrix. After 7 or 8 iteration, all average errors on multi-views surfaces become close to zero.

Results for all other objects are also shown in Figure 5.16 and 5.17. In all objects, partial surfaces are fairly registered to a common coordinate system. The translation and the rotation errors show the robustness of our registration algorithm. Practically speaking, we stop the iteration of registration when the translation error is smaller than 0.3 mm and the rotation error is smaller than 1×10^{-4} . Otherwise, we stop the registration if the number of iterations is more than 10.

5.4.2 Integration Results

Integration of partial surfaces into a complete 3D models is accomplished by computing the signed distance of objects and using Marching Cubes algorithm. Depending on the size of voxel, output 3D mesh models have different resolutions. Figure 5.18 shows the results of meshes for different sizes of voxel. If its size is too large, we lose some details of the objects, or if the size is too small, the number of vertices and triangles of the results are too many. We usually reconstruct 3D models for voxel size 3 or 4 mm.

Table 5.1 shows the number of vertices and triangles on each mesh model. When voxel size is 4 mm, it takes about 1 minute to integrate the 'Monkey' object into a single mesh. Results of all other objects are shown in Figure 5.19 and 5.20.

Table 5.1: Statistics for the results of 'Monkey'

Voxel size (mm)	2	3	4	6	8
Vertices	17348	7406	4162	1804	978
Triangles	34624	14804	8324	3608	1940
Time (sec)	315	134	76	33	18

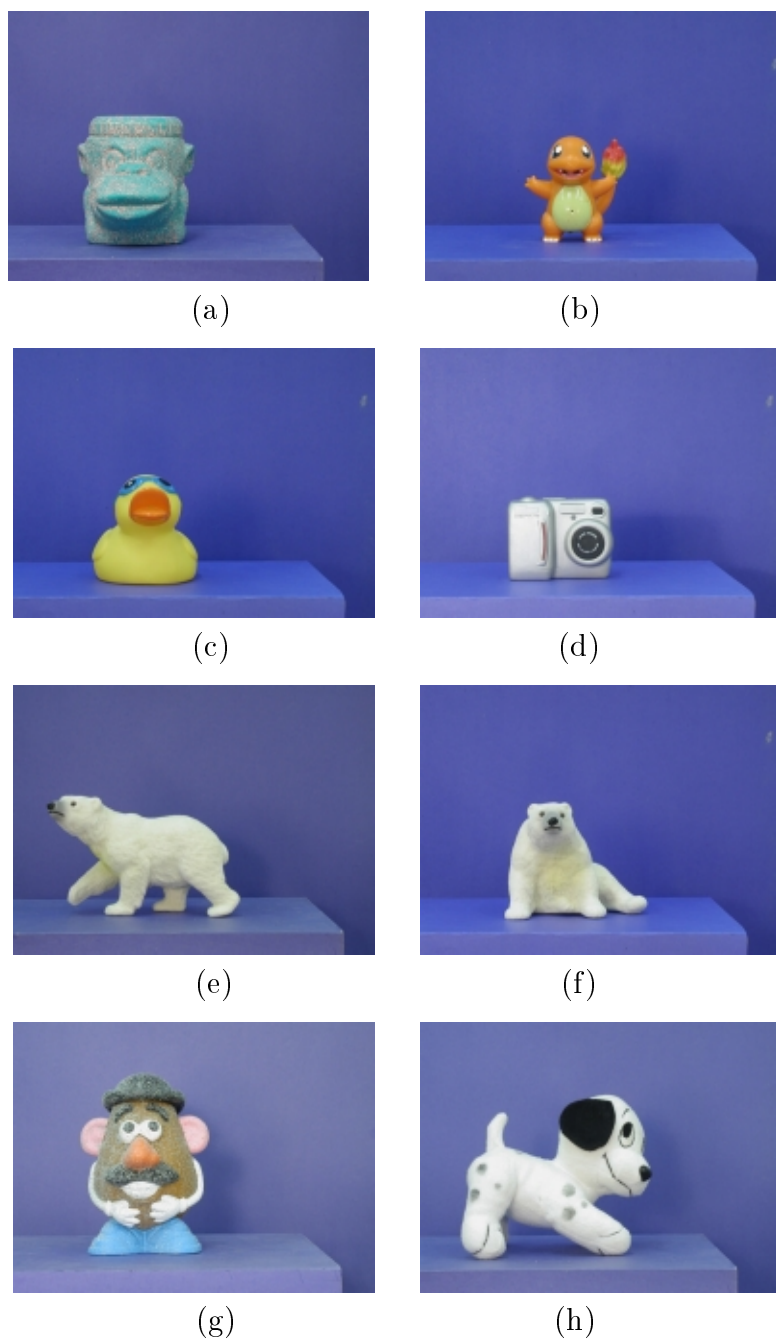


Figure 5.12: Test objects for volumetric 3D model reconstruction. (a) Monkey (b) Pokemon (c) Duck (d) Nikon775 (e) Polarbear1 (f) Polarbear2 (g) Potatohead (h) Dog

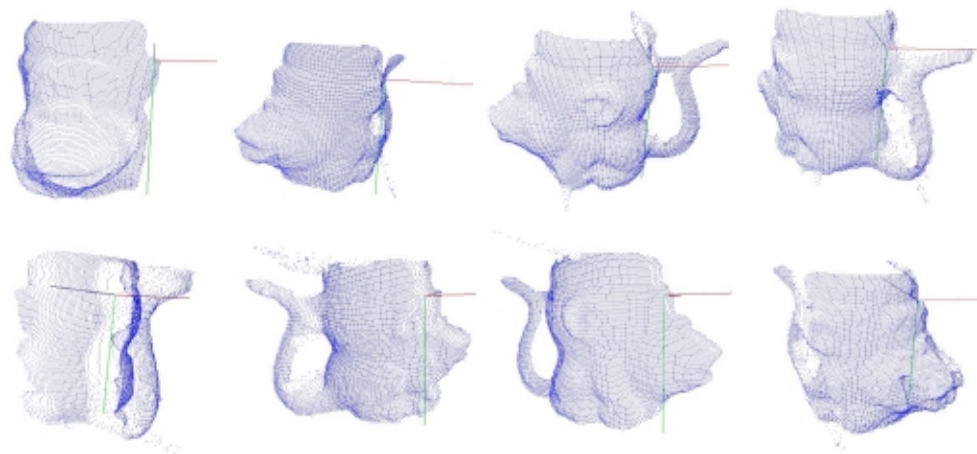


Figure 5.13: Partial shapes of 8 views of 'Monkey'. From the upper left, they are obtained at $0, 45, \dots$, and 315 degree.

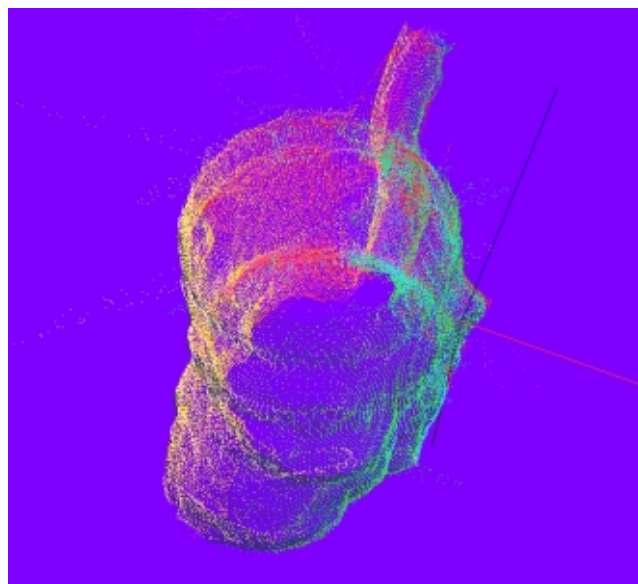


Figure 5.14: Registered partial shapes of 'Monkey'. All shapes are represented with different colors of point clouds.

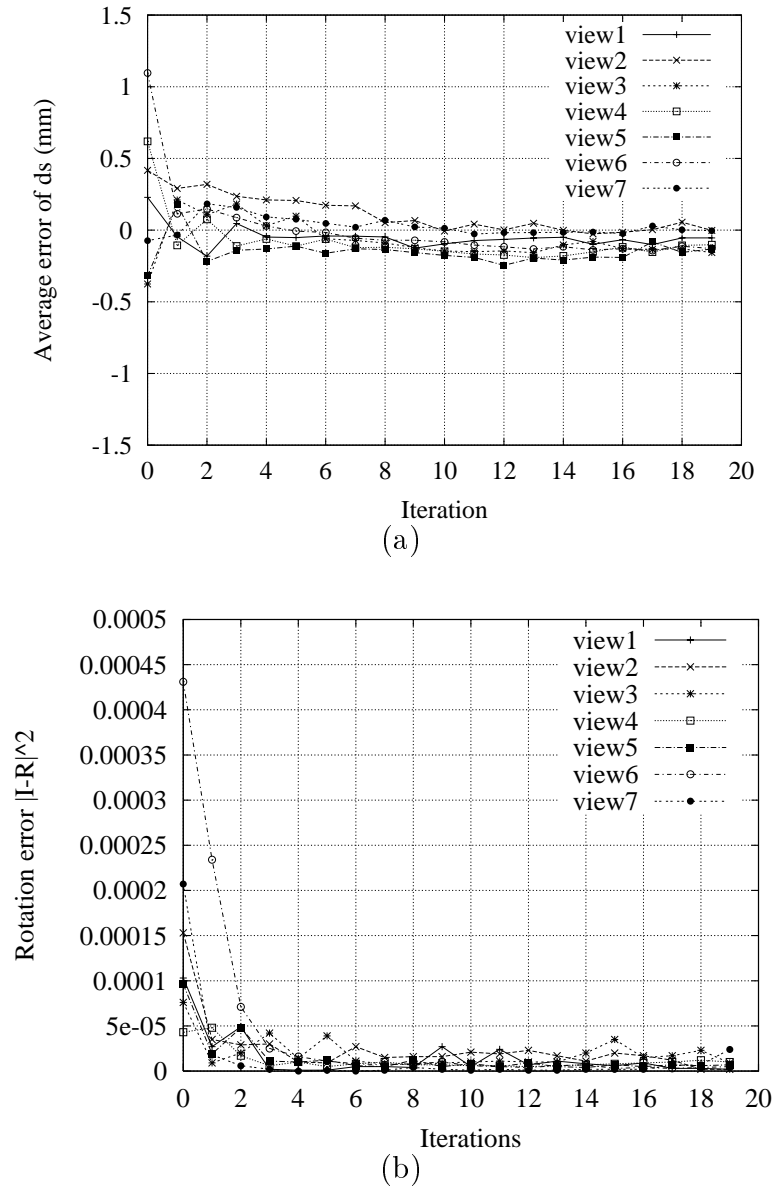


Figure 5.15: Registration errors of 'Monkey' Object (a) Average of translation error
(b) Average of rotation error

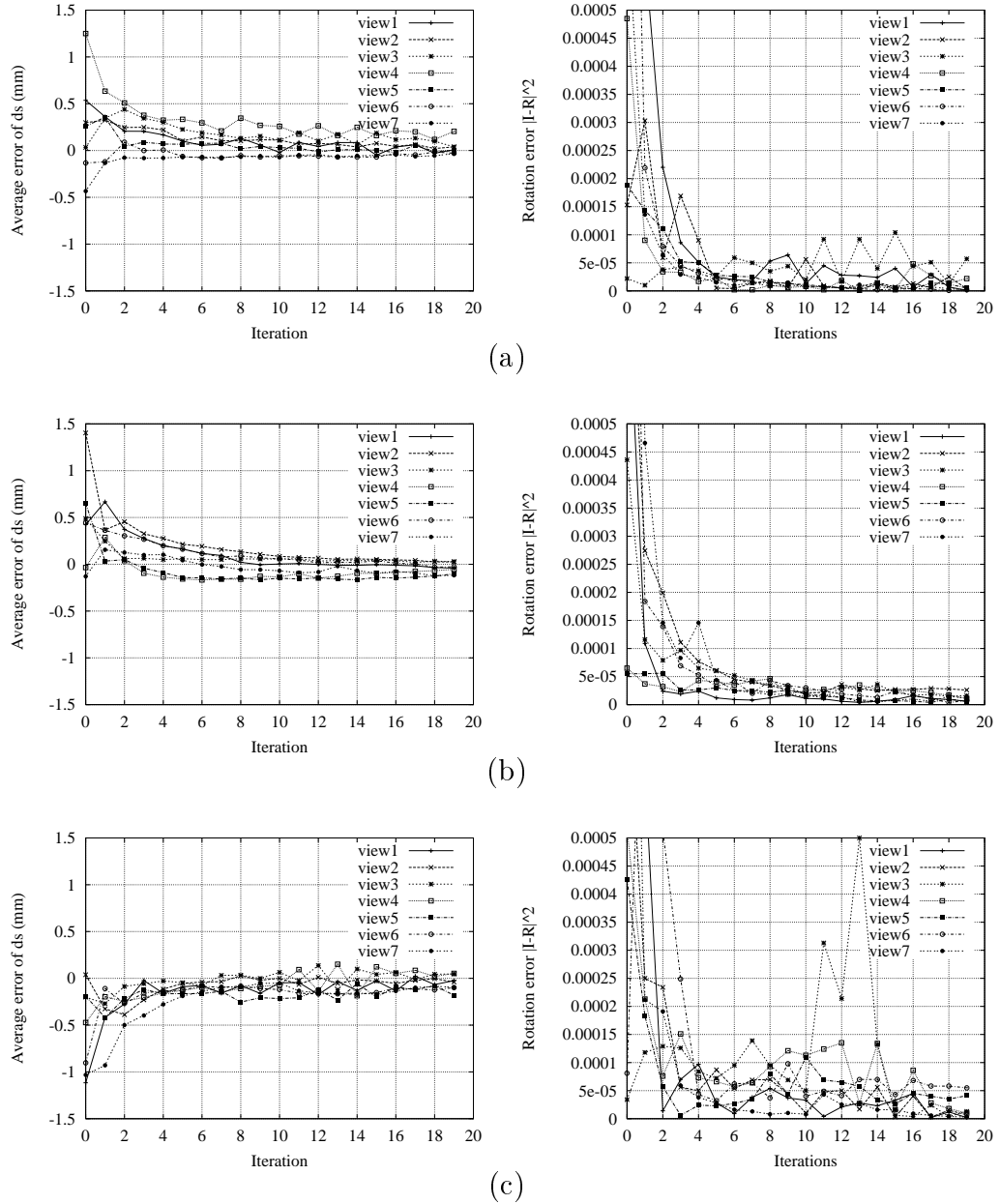


Figure 5.16: Registration errors (translation and rotation) of objects 'Pokemon' and 'Nikon775' (a) Pokemon (b) Duck (c) Nikon775

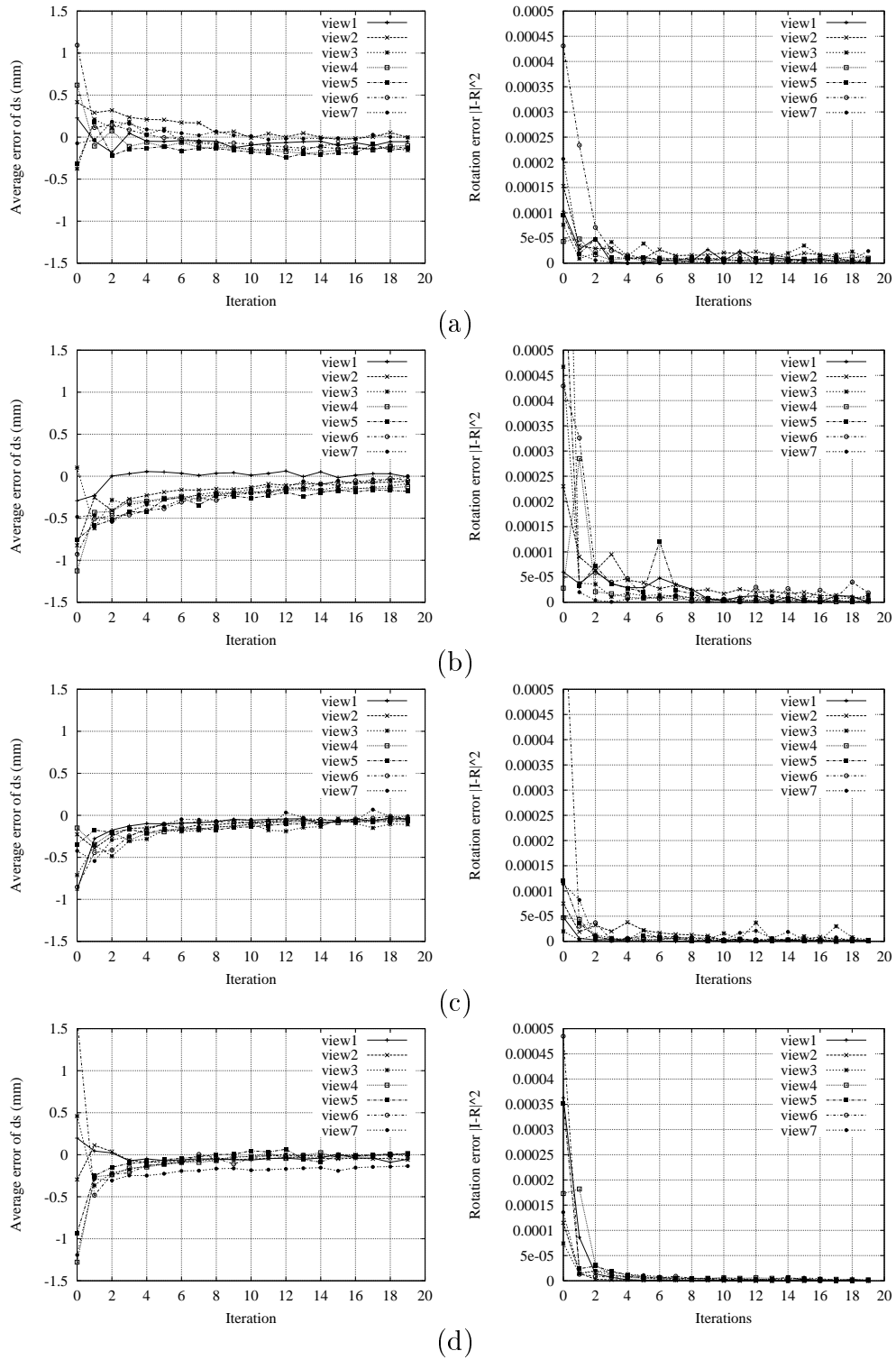


Figure 5.17: Registration (translation and rotation) errors of objects (a) Polarbear1 (b) Polarbear2 (c) Potatohead (d) Dog

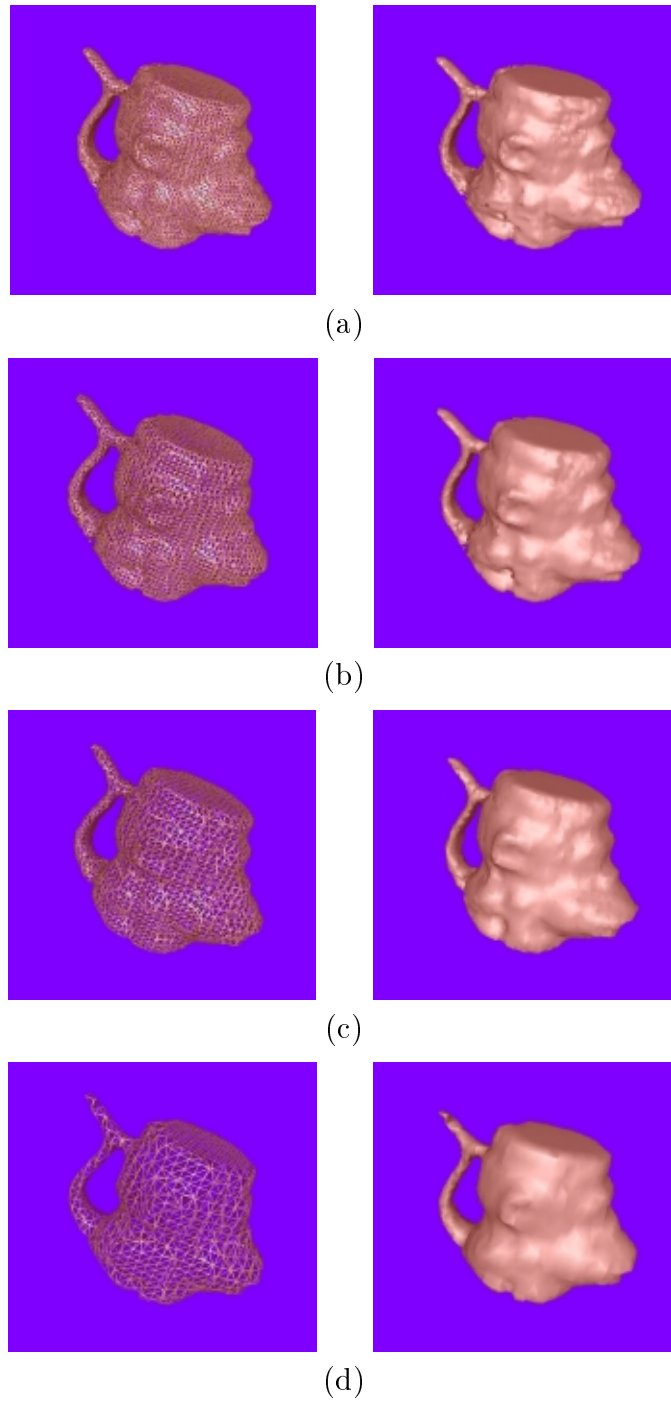


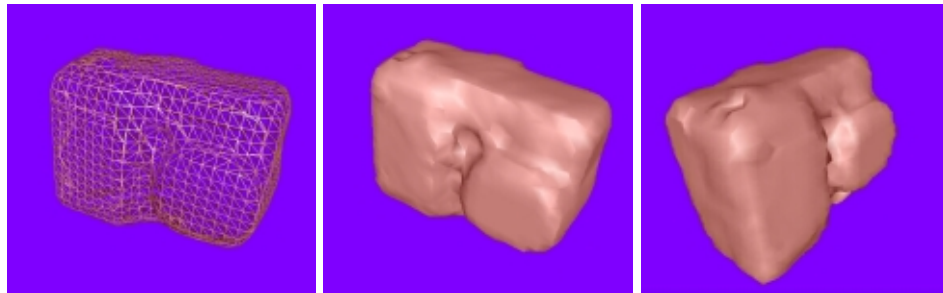
Figure 5.18: Reconstructed 3D models of 'Monkey' with different voxel size. (a) 2mm (b) 3mm (c) 4mm (d) 6mm



(a)



(b)



(c)



(d)

Figure 5.19: Mesh and shading results for test objects (a) Pokemon (b) Duck (c) Nikon775 (d) Polarbear1

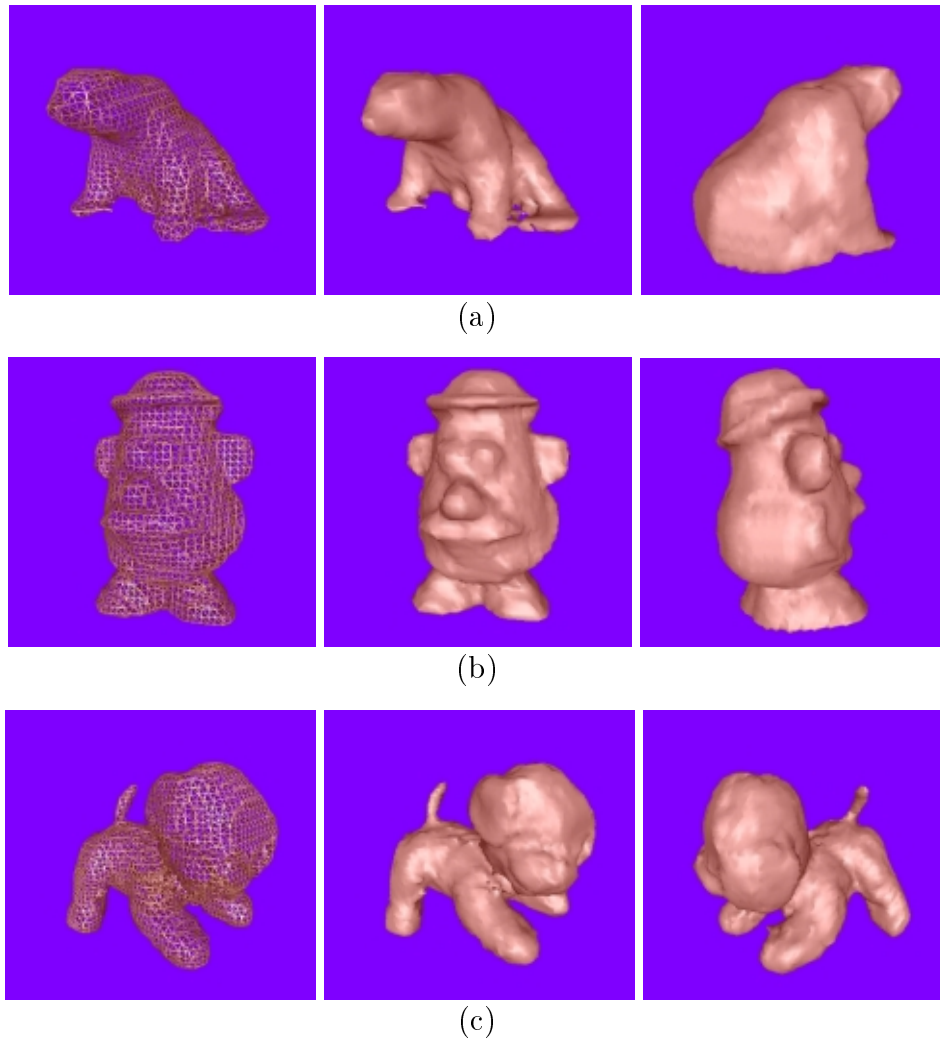


Figure 5.20: Mesh and shading results for test objects (a) Polarbear2 (b) Potatohead (c) Dog

5.5 Error Analysis

In order to analyze the accuracy of our reconstruction technique, we reconstruct 3D models of some simple objects. We reconstruct two test objects as shown in Figure 5.21 and Figure 5.22. One object is a rectangular parallelepiped and the other one is a cylinder. We reconstruct 3D models of the objects and measure dimensions of the models to compare with those of the real objects. We choose these two objects because it is easy to measure their dimensions.

The dimension of the first object 'Cubes' is $60 \times 60 \times 90$ (mm^3) for $W \times D \times H$. The radius R of the second object 'Cylinder' is 50.04 mm and the height H is 138.2 mm . We reconstructed the 3D models of the object under the same conditions used in experiments in the previous sections.

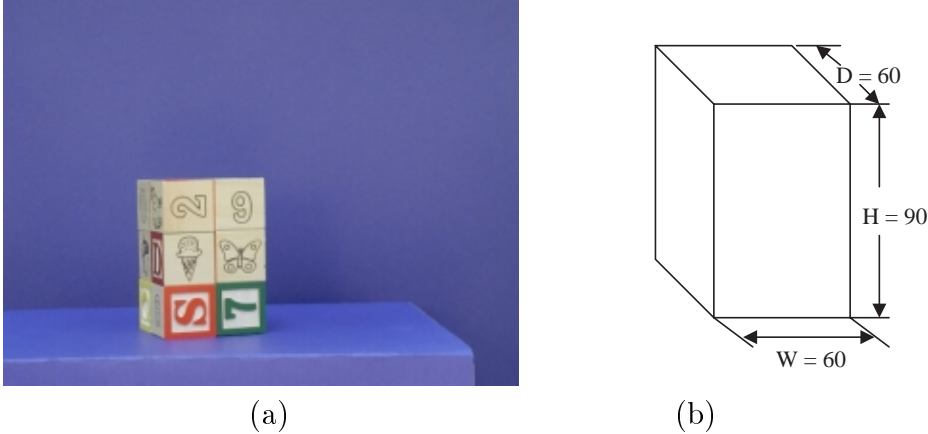


Figure 5.21: Test object 'Cubes' for error analysis(a) Picture (b) Dimension in mm

In order to measure errors of the reconstructed 3D model, we use an ICP-based registration technique to first register point clouds of reconstructed 3D model to that of the ground truth. Then dimensional errors are measured between all points on the model and their closest conjugates on the ground truth. Figure 5.24(a) shows all points on the ground truth of the object 'Cubes'. Figure 5.24(b) shows overlapped points of the ground truth and the reconstructed model after the registration. The blue-colored (dark-grey) points are on ground truth model and the green-colored (light-grey) points on the reconstructed model. The small red-colored line segments are errors between two control point sets. Figure 5.24(c) and (d) also show point clouds of all vertices on the 'Cylinder' object.

We iteratively register a 3D model to its ground truth until an error metric between the 3D model and the ground truth converge to a constant value. The error

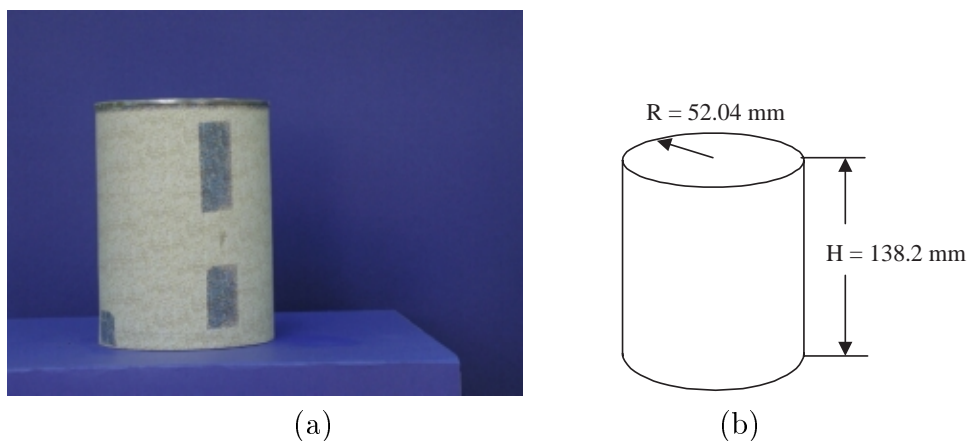


Figure 5.22: Test object 'Cylinder' for error analysis (a) Picture (b) Dimension

between two models is measured by the RMS error between two control point sets of the models. Figure 5.23 shows the results of registration of two objects. We use 392 control points for the 'Cubes' object and 104 points for 'Cylinder'.

After two 3D models – the reconstructed model and the ground truth model – are registered, we measure dimensional errors on the reconstructed model with respect to the ground truth. For the 'Cubes' object, the RMS errors in W , H and D dimensions are measured for all vertices on corresponding planes– for example the top and the bottom planes for H dimension– with respect to the closest vertices on the ground truth. Similarly for the 'Cylinder' object, errors in R and H dimensions are measured using points on side surfaces, and top and bottom surfaces, respectively. Table 5.2 shows RMS and maximum errors in all dimensions of the 'Cubes' object. We also measure the volume V of the reconstructed model using a volume measuring technique described in a reference paper [60]. With respect to the volume of the ground truth, the reconstructed model has 2.04 % error as shown in the table. Table 5.3 shows the results of 'Cylinder' object.

5.6 Conclusions

Complete 3D model reconstruction through a volumetric integration of multiple range images is presented. Multiple partial shapes are acquired by stereo vision systems presented in the Chapter 2 or Chapter 3. The partial shapes are then registered to a common coordinate system according to the system calibration parameters. And a multi-view registration technique refines and minimizes registration error in

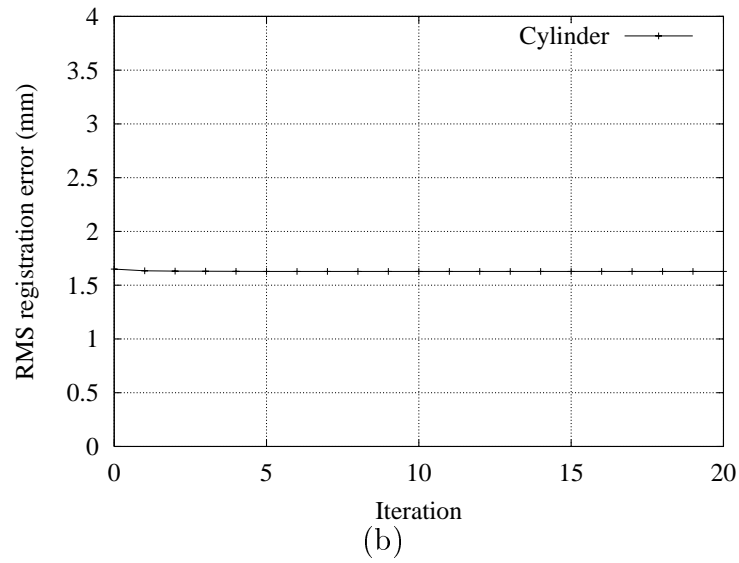
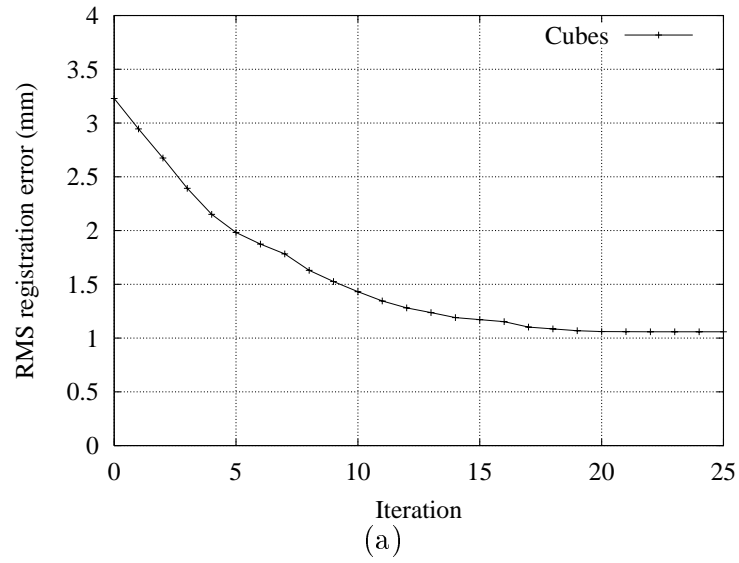


Figure 5.23: RMS registration error between the ground truth and the reconstructed 3D models. (a) Cubes (b) Cylinder

Table 5.2: RMS and maximum errors of 'Cubes'

Dimension	W (<i>mm</i>)	D (<i>mm</i>)	H (<i>mm</i>)	V (<i>mm</i> ³)
Size	60	60	90	324000
RMS error	1.06	0.90	0.85	330542
MAX error	2.99	3.03	2.17	
(%) error (RMS)	1.76	1.50	0.94	2.04

Table 5.3: RMS and maximum errors of 'Cylinder'

Dimension	H (<i>mm</i>)	R (<i>mm</i>)	V (<i>mm</i> ³)
Size	138.2	52.04	1175797.4
RMS error	1.54	1.20	1179466.7
MAX error	4.56	4.42	
(%) error (RMS)	1.11	2.29	0.31

overlapping regions.

We combine *shape from silhouettes* technique and *voxel coding* technique to remove erroneous data points due to stereo mismatches. A novel voxel coding technique is introduced to select reliable partial shapes to compute the signed distance of a voxel. Voxel coding not only removes erroneous voxels but also increases the accuracy of reconstruction. We show the accuracy of our 3D reconstruction technique through error analysis of ground truth objects. In the following chapters, we will introduce a novel pose estimation and integration technique to reconstruct all visible surfaces of an object. In this technique, we will also incorporate the voxel coding technique to integrate two different 3D models.

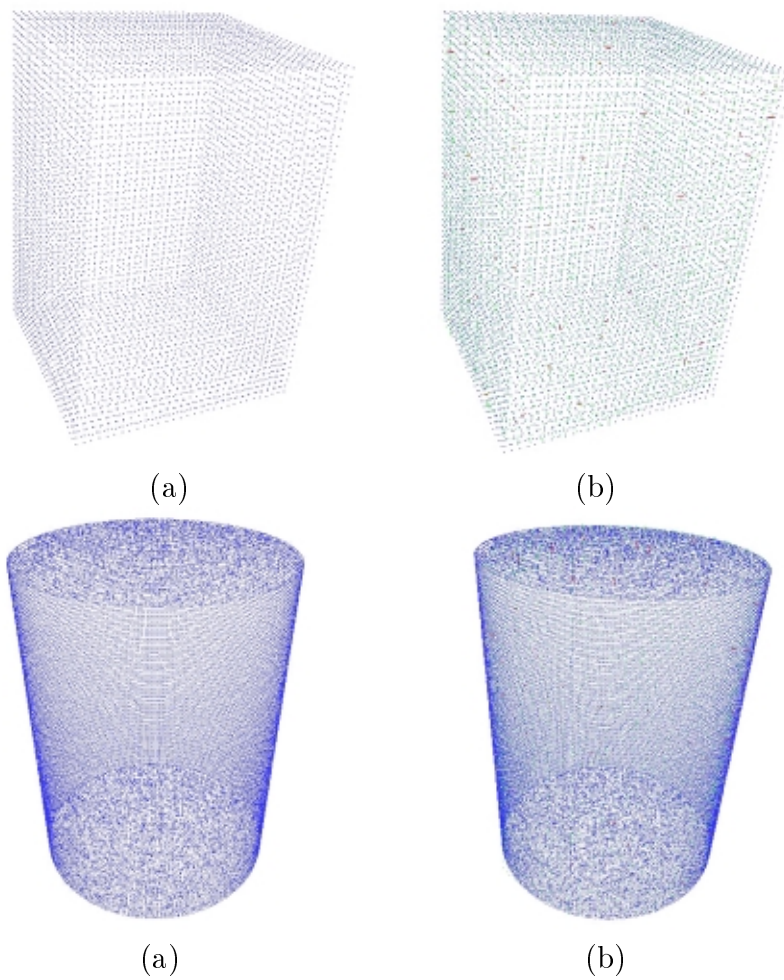


Figure 5.24: Points clouds of registered 'Cylinder' model (a) Ground Truth (b) Overlapping with reconstructed model

Chapter 6

Pose Estimation

This chapter presents a pose estimation technique to determine coarse registration parameters between two 3D models of an object. The models are reconstructed by merging multi-view range images of two different poses of the object, which is arbitrarily placed on a turntable. The result of pose estimation will facilitate registration refinement and integration of two 3D models. We introduce a simple pose estimation technique based on matching tangent planes of a 3D model with the Base Tangent Plane (BTP) which is invariant for a vision system. In order to reduce computation time, we employ geometric constraints to find consistent and Stable Tangent Planes (STP). A STP is a plane on which the object can rest in a stable state on a turntable. By matching the base tangent plane of one 3D model to a stable tangent plane of the other model, we derive a pose transformation matrix and register the models to a common coordinate system. We find the best-matching STPs which minimize a pose error between two models.

6.1 Introduction

Most investigations on 3D model reconstruction are limited to using a single pose of an object. However, for many real objects, using a single pose yields only a partial 3D model because some surfaces of the object remain hidden from the range sensor for any given pose due to occlusion, concavities, etc. For example, a tea cup placed upright on a turntable hides the bottom surface of the cup from the range sensor. A 3D model of such hidden surfaces could be reconstructed by placing the object in a different suitable pose (e.g. by placing the tea cup on its side) and sensing the visible shape. This yields a second partial 3D model of the object for the new pose. In order to obtain a complete 3D model, the two partial 3D models for the two different poses need to be registered and integrated. However, registration and integration of two partial 3D models for two different poses of an object is a very difficult problem. For this reason, only a few researchers have considered this problem.

P. Allen and R. Yang [2] stitch a bottom surface of an object by matching edge features of the object’s 3D model with an edge image of the bottom. They acquire multi-view range images of a fixed object using a moving range finder. After reconstructing a 3D model, they acquire a partial shape of the bottom surface and stitch the shape and texture of the bottom to the 3D model using a feature matching technique. K. Wong and R. Cipolla [105] employ a shape-from-silhouettes technique for 3D modeling and combine a *structure-from-motion* technique to estimate registration parameters of multiple views. But they manually register the top and the bottom surfaces of the object. W. Niem [65] also reconstructs complete 3D models using a *shape-from-silhouettes* technique, registers, and integrates the top and the bottom surfaces manually. H. Lensch et al. [52] and Y. Iwakiri and T. Kaneko [40] use silhouette matching techniques for registration and stitching of textures to a 3D model. However, their investigations consider only registration and stitching of texture properties of an object, not geometric reconstruction of the object. D. Huber [36, 37] also presents a 3D reconstruction technique using an unconstrained registration of n-view partial shapes. He registers partial shapes using *Spin* images and a graph searching technique.

A schematic diagram of our 3D modeling system is shown in Figure 6.1. The 3D model reconstruction for a single pose of an object is based on a volumetric modeling technique which is presented in the previous chapter. Registration of two pose models consists of two steps, coarse registration and its refinement. Because we place an object in two arbitrary poses, determining coarse registration parameters is very difficult without *a priori* knowledge of the transformation between two poses. We use a novel pose estimation technique of two 3D models to determine coarse registration parameters [70]. The pose estimation technique finds a stable tangent plane (STP) on a 3D model which can be transformed to the base tangent plane (BTP) of the other model and *vice versa*. For a rigid object, the BTP is a tangent plane of the object’s outer surface. Therefore, we match the BTP of the first pose to a STP of the second pose, and simultaneously match the BTP of the second pose to a STP of the first pose. The best matching plane from each model is used to estimate the transformation matrix.

6.2 Methodology of Pose Estimation

6.2.1 Base Tangent Plane

There have been many investigations on pose estimation for multiple 3D models. A useful survey on pose estimation techniques is presented in [14]. We employ a novel pose estimation technique based on geometric constraints of our vision system

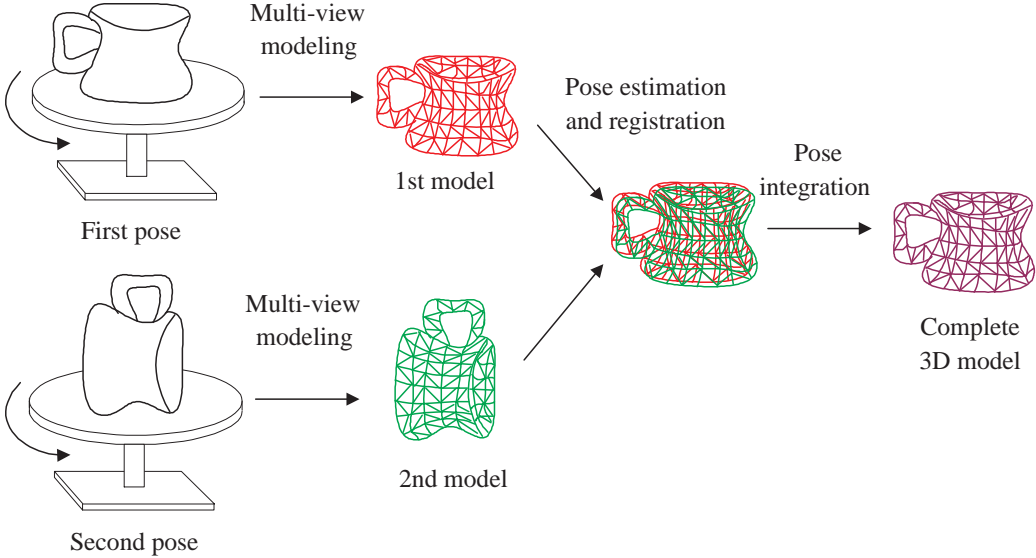


Figure 6.1: Schematic diagram of our 3D modeling

[70]. This technique finds a stable tangent plane (STP) on a 3D model which can be matched to the base tangent plane (BTP) of the other model. When we place a rigid object on the flat (planar) top of a turntable, the object rests with its outer surface touching the table top. The planar table top will be a tangent plane of the object’s surface. We call the planar table top the BTP of the turntable. The BTP is a global tangent plane in the sense that it will not intersect the object’s volume anywhere (in contrast, a local tangent plane may intersect the object’s volume at a point far from the point of tangency). The BTP is invariant with respect to the object’s pose and the world coordinate system. The following are some characteristics useful in pose estimation.

1. The base tangent plane (BTP) of the turntable is a global tangent plane of the object.
2. There exists a unique tangent plane of the first pose model which corresponds to the BTP of the second pose, which is also a tangent plane of the second model.

Suppose an object is placed on the turntable with two different poses, $Pose1$ and $Pose2$ as shown in Figure 6.2. Then there is a unique tangent plane \mathcal{T}_1 (its normal

vector is $\hat{\mathbf{n}}_{T1}$) in the first pose which matches the base tangent plane B ($\hat{\mathbf{n}}_B$) in the second pose. Similarly, there is a unique plane \mathcal{T}_2 ($\hat{\mathbf{n}}_{T2}$) in *Pose2*, which matches B ($\hat{\mathbf{n}}_B$) in *Pose1*. Because $\hat{\mathbf{n}}_B$ is a common and invariant vector in the vision system, we can estimate a rotation matrix using $\hat{\mathbf{n}}_{T1}$ and $\hat{\mathbf{n}}_{T2}$.

Let \mathbf{Q}_2 be an initial transformation matrix from *Pose2* to *Pose1* which aligns $\hat{\mathbf{n}}_{T2}$ with $\hat{\mathbf{n}}_B$. After the alignment, two models are registered except one degree of freedom along the axis of $\hat{\mathbf{n}}_B$. Then, by aligning the transformed vector $\mathbf{Q}_2\hat{\mathbf{n}}_B$ with $\hat{\mathbf{n}}_{T1}$, we estimate the transformation matrix \mathbf{Q}_2^1 . Translation between two models is estimated by Center of Mass (CoM) of the models. We assign new coordinate systems \mathbf{T}_1 and \mathbf{T}_2 , for tangent plane \mathcal{T}_1 and \mathcal{T}_2 respectively. It will be described in the next section. The rotation matrix is estimated by transforming the coordinate system \mathbf{T}_1 and \mathbf{T}_2 to the common coordinate system of the base B .

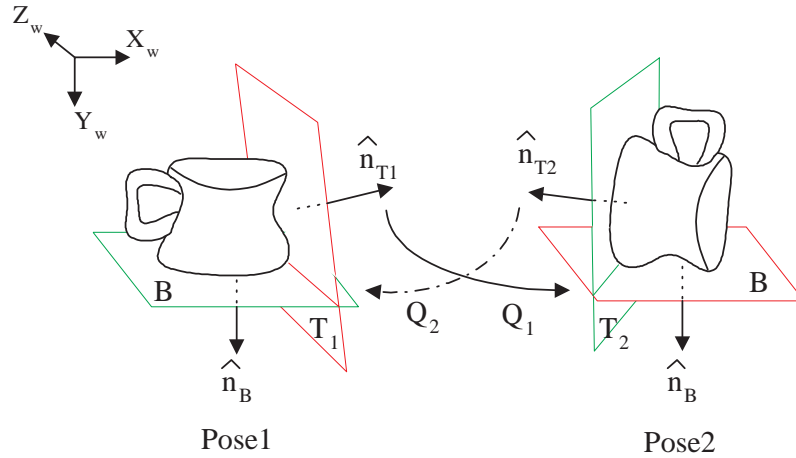


Figure 6.2: A tangent plane of the first pose $\hat{\mathbf{n}}_{T1}$ uniquely matches the BTP of the second pose $\hat{\mathbf{n}}_B$, and *vice versa*.

Consequently, our technique finds tangent plane coordinates \mathbf{T}_1 and \mathbf{T}_2 , one from each model, which minimize a cost function, or a volumetric pose error, between two models. The pose error is estimated in terms of SSD (Sum of Square Difference) error between two models. Suppose a vertex \mathbf{P}_{1i} on *Pose1* corresponds to another vertex \mathbf{P}_{2i} on *Pose2*. If \mathbf{Q}_2^1 is a transformation from *Pose2* to *Pose1*, then the pose error is measured by

$$\sum^2 = \sum_{i=0}^{K-1} \|\mathbf{P}_{1(i)} - \mathbf{Q}_2^1 \mathbf{P}_{2(i)}\|^2, \quad (6.1)$$

where, K is the number of vertices in $Pose1$ or $Pose2$, but it is usually down-sampled to increase computation speed.

6.2.2 Stability Constraints

The surfaces of the object are represented by a finite number of triangle meshes [56]. Therefore, there will be a finite number of tangent planes on each mesh model. Let a set of tangent planes of $Pose1$ be \mathcal{T}_1 , and \mathcal{T}_2 be another set of tangent planes of $Pose2$. If we assign a tangent plane for every vertex, however, measuring the cost function for all combinations of two sets \mathcal{T}_1 and \mathcal{T}_2 is computationally expensive.

In order to reduce computational complexity, we remove some local or unstable tangent planes by employing geometric constraints as the followings. A key idea for employing the constraints is that the object is placed on the turntable in a stable pose and the turntable is horizontal.

1. *Base plane constraint* : An object is placed on the BTP which is one of the global tangent planes of the object. This BTP does not intersect the object's volume.
2. *Stability constraint* : The BTP of the turntable is horizontal and the object is in a stable pose. Therefore, the projection of the CoM to a STP is always inside the convex hull of the projections of its supporting vertices.
3. *Height constraint* : If two pose models are correctly registered, their heights will be very similar (It may not be the same, because of noise).

Based on the constraints above, we consider only stable tangent planes (STPs) on the model. These constraints greatly reduce the number of tangent planes and the computation time for pose matching.

6.3 Pose Estimation

First we list the steps in our pose estimation algorithm and then provide additional details.

1. For each pose model, obtain an Extended Gaussian Image (EGI) using vertex normal vectors of the 3D model.

2. For each face of the tessellated sphere, determine whether the tangent plane corresponding to the face is a candidate for matching with the base plane of the other pose. The tangent plane is a valid candidate in all cases except the following. If the orientation histogram count of the face is zero, or the angle between the face normal and the vertical axis is smaller than a threshold, then the tangent plane is not a candidate. The latter condition is due to the assumption that the relative rotation between the two poses is greater than a threshold angle.
3. Reject any tangent plane intersecting its own volume by checking the signed distance from each voxel to the plane.
4. Reject any tangent plane \mathcal{T} for which the projection of the CoM of the model is outside the convex hull of its supporting vertices.
5. Reject any tangent plane \mathcal{T} if the height difference between the model transformed by \mathbf{Q}_2^1 and the fixed model is greater than a threshold δ_H .
6. Find two matching planes from both models by measuring the pose error between the two registered models.

6.3.1 Finding Tangent Planes

In the first step, an EGI of a pose model is constructed and tangent planes on a tessellated Gaussian sphere are obtained [39, 45]. Suppose $\hat{\mathbf{n}}_f$ is the normal vector of a tessellated polygon and $\hat{\mathbf{n}}_T$ is the normal vector of an associated tangent plane, where $\mathbf{T}(\mathbf{P}) = Ax + By + Cz + D$. Because of the first constraint, we assume that the tangent plane passes through a vertex whose projection distance to $\hat{\mathbf{n}}_f$ is the maximum. We call this vertex \mathbf{P}_m a *frontier vertex*. Then we initialize the plane $\mathbf{T}(\mathbf{P}_m)$ using the *frontier vertex* and the face normal.

However, because we search for a STP which can support the object as a base plane, we refine the tangent plane \mathbf{T} by employing some supporting vertices. Ideally speaking, supporting vertices \mathbf{P}_i should be on the tangent plane \mathbf{T} , and it is necessary that

$$\mathbf{T}(\mathbf{P}_i) = 0 \tag{6.2}$$

$$\hat{\mathbf{n}}_f \cdot \hat{\mathbf{n}}_p = 1, \tag{6.3}$$

for the stability of the object [47]. However, our 3D model is not a physical object, but a digitized and polygonized 3D model which has inherent noise on its surface. Therefore, it is necessary to set thresholds for selecting supporting vertices. The

normal vector of a supporting vertex must closely coincide with the normal of the tangent plane. In addition, the vertices should be close to the plane. Therefore we find some vertices whose normal vectors $\hat{\mathbf{n}}_p$ and their dot product, $\hat{\mathbf{n}}_p \cdot \hat{\mathbf{n}}_f$ are less than a threshold $\cos(\theta_G)$ as shown in Figure 6.3. And instead of searching vertices on the tangent plane, we search for supporting vertices which are close to the plane and refine the plane \mathbf{T} .

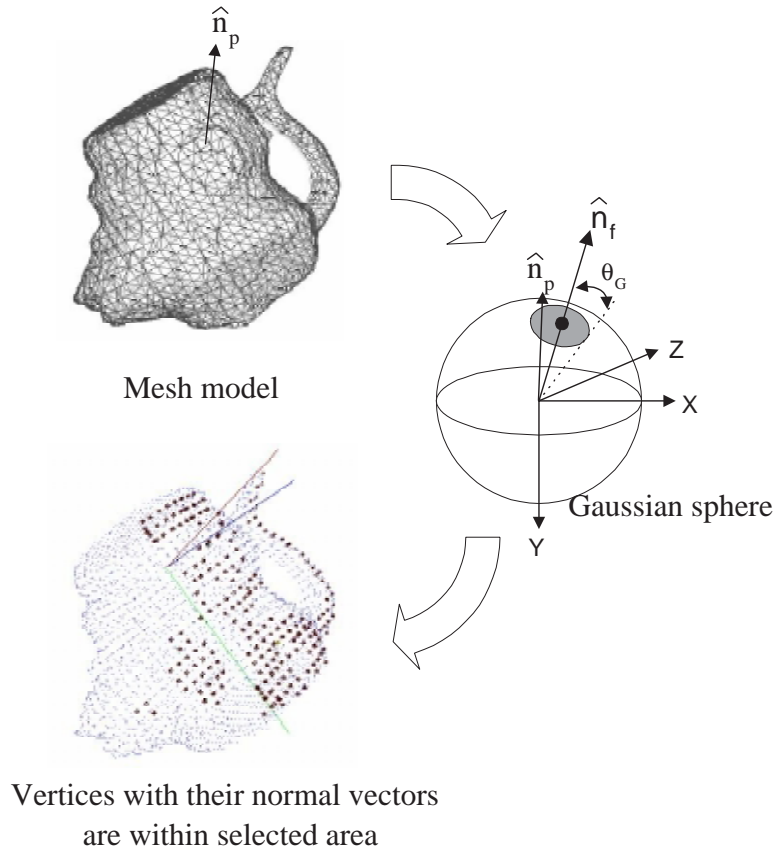


Figure 6.3: Initializing tangent plane and its supporting vertices

As shown in Figure 6.4(a), we select a vertex \mathbf{P}_i as one of the supporting vertices, if its relative distance to the plane, D_i/D_{mi} , is less than a threshold. Here, D_i is the distance from \mathbf{P}_i to the plane and D_{mi} is the distance from the *frontier vertex* to \mathbf{P}_i . In order to avoid selecting vertices only in a very small region (it may produce an unstable plane), we pick only one vertex from a triangle face when multiple vertices of the face meet the condition. After finding a set of supporting vertices \mathcal{P}_s , we move

the origin of the plane to \mathbf{P}_c , the centroid of \mathcal{P}_s , and refine the normal vector $\hat{\mathbf{n}}_T$ by averaging normal vectors of \mathcal{P}_s .

A new coordinate system is then generated for the tangent plane in order to obtain a transformation matrix to the reference coordinate system. We set the normal vector $\hat{\mathbf{n}}_T$ (let \mathbf{T} denote a refined tangent plane from now on) to the Y axis as shown in Figure 6.4(b), because it is convenient to match with the Y_W of the reference coordinate system. Y_W is also the vector normal of the base plane. For X axis, a vector from \mathbf{P}_c to the projection of \mathbf{P}_m to \mathbf{T} is normalized. And the Z axis is set according to the right hand coordinate system. In Figure 6.4(b), the coordinate system of a tangent plane \mathbf{T} is shown. A 3D model is represented as point clouds, the green (light-grey) dots are vertices whose normal vectors are within angle θ_G from $\hat{\mathbf{n}}_f$, and the red (dark-grey) dots are all supporting vertices selected from green vertices.

6.3.2 Finding Stable Tangent Planes

For two 3D models, we find all tangent planes using their EGIs. Each tangent plane \mathbf{T} consists of supporting vertices and has its own coordinate system. Using tangent planes from each 3D model, we can estimate a pose transformation between two models by finding the best matching planes from two models. However, there will be a finite, but large number of tangent planes for each 3D mesh model. Let a set of tangent planes of *Pose1* be \mathcal{T}_1 , and \mathcal{T}_2 be another tangent plane set for *Pose2*. If we consider every tangent plane of a pose to be a candidate for matching the BTP of the other pose, the matching becomes a computationally expensive step. In order to reduce computation, we remove some local or unstable tangent planes from further consideration by employing three geometric constraints.

The first constraint is *Base plane constraint*. The BTP is a global tangent plane in the sense that it will not intersect the object’s volume anywhere. Therefore we check all vertices to determine if a tangent plane intersects the volume. If the dot product of any vertex p with the plane normal $\hat{\mathbf{n}}_T$ is greater than the parameter D of the plane, we remove the plane. Due to noise on model’s surface, we use a threshold $D + \delta_I$ instead of D .

Next constraint is *Stability constraint*. The BTP of the turntable is horizontal and the object is in a stable pose. Therefore, given the center of mass (CoM) of a 3D model, its projection to a STP, CoM_T is always inside the convex hull of the projections of all supporting vertices. The object will be unstable and fall over if CoM_T is outside the convex hull as shown in Figure 6.5.

The last constraint is *Height constraint*. It is a weak constraint. If two pose models are correctly registered, their heights will be very similar (It may not be the same, because of noise). We reject any tangent plane when the height difference is greater than a threshold δ_H . Figure 6.6 shows an example of removing inconsistent

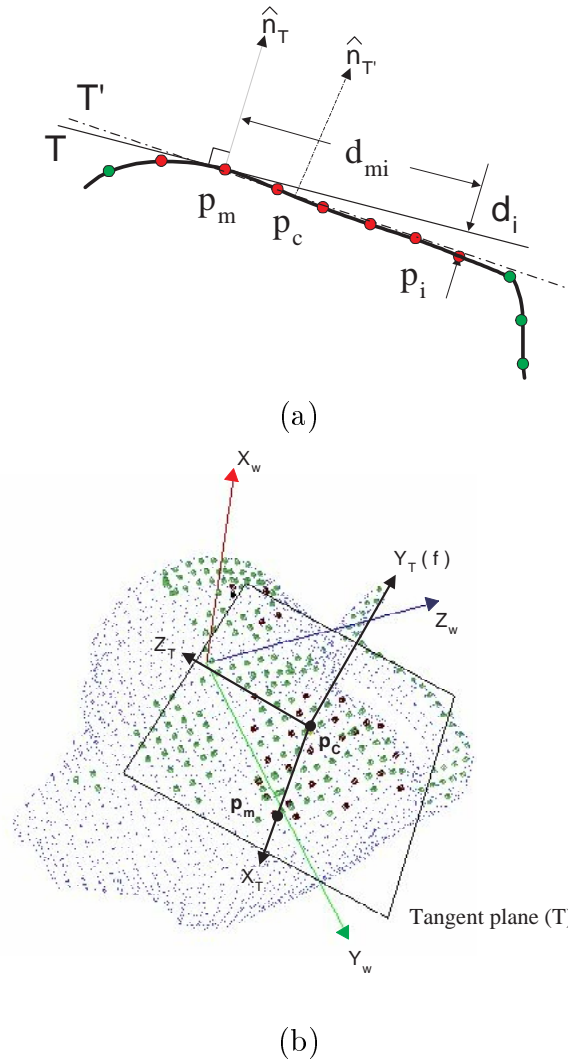


Figure 6.4: Construction of a tangent plane on a 3D model. (a) Tangent plane \mathbf{T} is refined by their supporting vertices. (b) Green (light grey) dots are all vertices with their normals within a threshold angle θ_G . But only red dots (dark grey) consist of supporting points on the tangent plane.

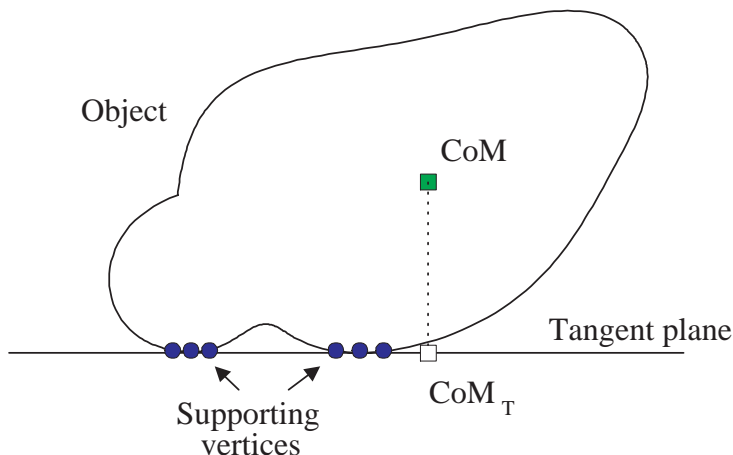


Figure 6.5: An unstable tangent plane. The projection of CoM to the plane is outside the convex hull of supporting vertices.

and unstable tangent planes. A 3D model is represented by a point clouds model and a tangent plane is represented by a square-shaped polygon. Figure 6.6(a) shows all tangent planes without using any constraint. Clouds of black-colored line show overlapping of all squares. Figure 6.6(b) is the result after rejecting volume-intersecting planes, 6.6(c) is after rejecting unstable planes, and 6.6(d) is the result after height comparison. It is clear that the number of candidates for the matching tangent plane is greatly reduced by the three constraints.

6.3.3 Matching Tangent Planes

Rejection of unstable and local tangent planes significantly reduces the number of tangent planes. The last step in pose estimation is finding two tangent planes, one from each pose, which registers two 3D models with a minimum pose error. For every STP in \mathcal{T}_1 , we derive a transformation matrix \mathbf{Q}_2^1 using every STP in \mathcal{T}_2 , measure the pose error, and find two STPs which yield the best-matching. Let a STP in \mathcal{T}_1 be \mathbf{T}_1 and another STP in \mathcal{T}_2 be \mathbf{T}_2 . The transformation matrix of the second pose (vertex set \mathcal{P}_2) to the first pose (vertex set \mathcal{P}_1) is estimated by using \mathbf{T}_2 and \mathbf{T}_1 . The transformation matrix first aligns \mathbf{T}_2 with the coordinates of BTP \mathbf{B}

$$\mathbf{B}' = \mathbf{T}_2^{-1}\mathbf{B}, \quad (6.4)$$

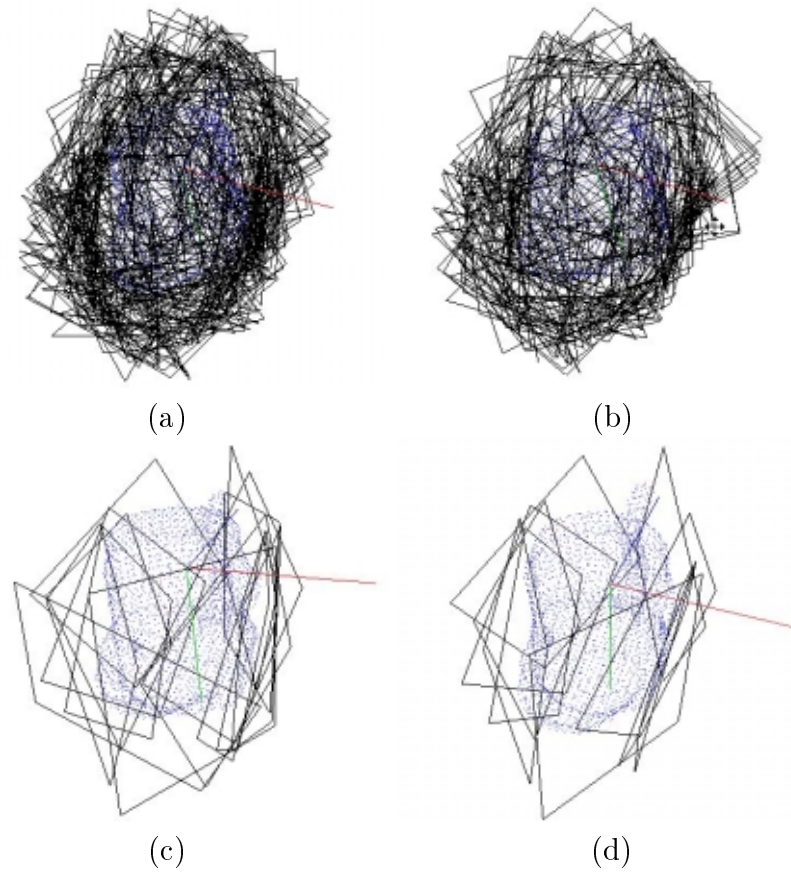


Figure 6.6: Finding STPs based on geometric constraints (*Pose1* of 'Monkey'). (a) Initial tangent planes (186 planes) (b) After removing volume-intersecting planes ($\delta_I = 3$ mm, 141 planes) (c) After removing unstable planes (18 planes) (d) After height comparison ($\delta_H = 3$ mm, 9 planes)

and \mathbf{T}_1 with \mathbf{B}' by rotating the model along the Y axis

$$\mathbf{T}_1 = \mathbf{R}_y^{-1} \mathbf{B}'. \quad (6.5)$$

The coordinate system of \mathbf{B} is the same as the world (or common) coordinate system. A rotation matrix \mathbf{R}_y aligns \mathbf{B}' with the coordinate system \mathbf{T}_1 . It is a rotation along the Y axis and computed by

$$\mathbf{R}_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where, $\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} = \begin{pmatrix} B'_x + B'_z & B'_z - B'_x \\ B'_x - B'_z & B'_x + B'_z \end{pmatrix}^{-1} \begin{pmatrix} T_{1x} + T_{1z} \\ T_{1x} - T_{1z} \end{pmatrix}$.

Center of mass of a 3D model is computed by a mass-computation technique [60]. Let the translation from the origin of the common coordinate system to each *CoM* be \mathbf{M}_1 and \mathbf{M}_2 . Then the pose transformation matrix \mathbf{Q}_{21} from \mathcal{P}_2 to \mathcal{P}_1 is computed by

$$\mathcal{P}'_2 = \mathbf{M}_1 \mathbf{R}_y^{-1} \mathbf{T}_2^{-1} \mathbf{M}_2^{-1} \mathcal{P}_2 \quad (6.6)$$

$$= \mathbf{Q}_2^1 \mathcal{P}_2. \quad (6.7)$$

The best matching pose transformation \mathbf{Q}_2^1 is estimated by minimizing a cost function between two models,

$$\min_{\{\mathbf{T}_1 \in \mathcal{T}_1, \mathbf{T}_2 \in \mathcal{T}_2\}} \left\{ \sum \|\mathcal{P}_1 - \mathbf{Q}_2^1 \mathcal{P}_2\|^2 \right\}. \quad (6.8)$$

6.4 Experimental Results

We test the pose estimation technique on a Pentium III 1GHz personal computer. We estimate the pose of several real and complex objects. Each object is placed on a turntable and 8 stereo image pairs are taken for each pose. We place the object in a normal upright pose for the first 3D model, and on its side for the second 3D model.

Figure 6.7(a) and 6.7(b) show images of two poses of an object “Monkey”. Each image is the first view of its respective multi-view images. Pose between two 3D models is estimated by our technique and a matching STP on each 3D model is shown in 6.7(c) and 6.7(d). The squares represent the matching planes. XYZ-axes of a common coordinate system are shown with RGB-colored lines. Overlapping of the

two models in their original poses are shown in Figure 6.7(e). After pose estimation, the two models are coarsely registered and the result is shown in Figure 6.7(f).

The number of tangent planes at every rejection step in pose estimation is shown in Table 6.1. The “Monkey” object is polygonized with a voxel size of 4mm [56]. Table 6.2 shows threshold angle θ_G , threshold distances δ_H and δ_I , average pose error between two models, number of vertices, and total estimation time. For measuring pose error, we use Equation 6.1. From a vertex $\mathbf{P}_{1(i)}$ in \mathcal{P}_1 , we find the closest vertex $\mathbf{P}_{2(i)}$ in \mathcal{P}_2 as the corresponding one. We sampled 50 vertices from each 3D model for computing the error measure.

The second object is a small toy, “Pokemon”. It’s height is only 8 cm. Figure 6.8(c) and (d) show matched STPs on the object’s surface. Figure 6.8(e) and (f) show two 3D models before and after the pose estimation. Figure 6.9 shows results of another object which is a plastic replica of a digital camera “Nikon775”. Pose estimation results are also shown in Figure 6.9(c), (d), (e), and (f). The last object is “PolarBear” in Figure 6.10. Pose estimation time for all objects is shown in Table 6.2. We can estimate pose of each object in about 2 seconds.

Table 6.1: Number of tangent planes on 3D models. (a)Initial planes (b) Intersection constraint (c) Stability constraint (d) Height constraint

Constraints		(a)	(b)	(c)	(d)
Object					
Monkey	pose1	186	141	18	8
	pose2	171	154	21	4
PolarBear	pose1	180	150	18	7
	pose2	167	144	11	4
Pokemon	pose1	205	173	8	2
	pose2	185	173	17	9

6.5 Conclusions

Pose estimation between two 3D models is estimated by matching global tangent planes with base planes. Two 3D models are reconstructed by multi-view modeling of two different poses of an object. In order to reconstruct a complete 3D model by registering and integrating two pose models, we estimate the pose between two models. Because the object always stands on one of its global tangent planes, we find

Table 6.2: Pose estimation error and estimation time ($D_i/D_{mi} = 0.1$)

Object	Monkey	Polarbear	Pokemon
Voxel size (<i>mm</i>)	4	4	2
$\cos(\theta_G)$	0.8	0.5	0.5
δ_I, δ_H (<i>mm</i>)	3.0	3.0	3.0
Avg. error (<i>mm</i>)	2.46	1.95	2.44
No. vertices	3294	5106	6180
Time (<i>sec</i>)	1.85	1.56	1.4

a tangent plane from the first model, which matches the base plane of the second pose and *vice versa*.

Matching tangent plane with the BTP gives constraints that reject inconsistent and unstable tangent planes. We employ three constraints- base plane constraint, stability constraint, and object height constraint, to reject such tangent planes. Experimental results show a great decrease in the number of tangent planes and matching complexity. In the next chapter, we will use our pose estimation technique for an initial registration of two 3D models, which will be integrated to a complete 3D model.

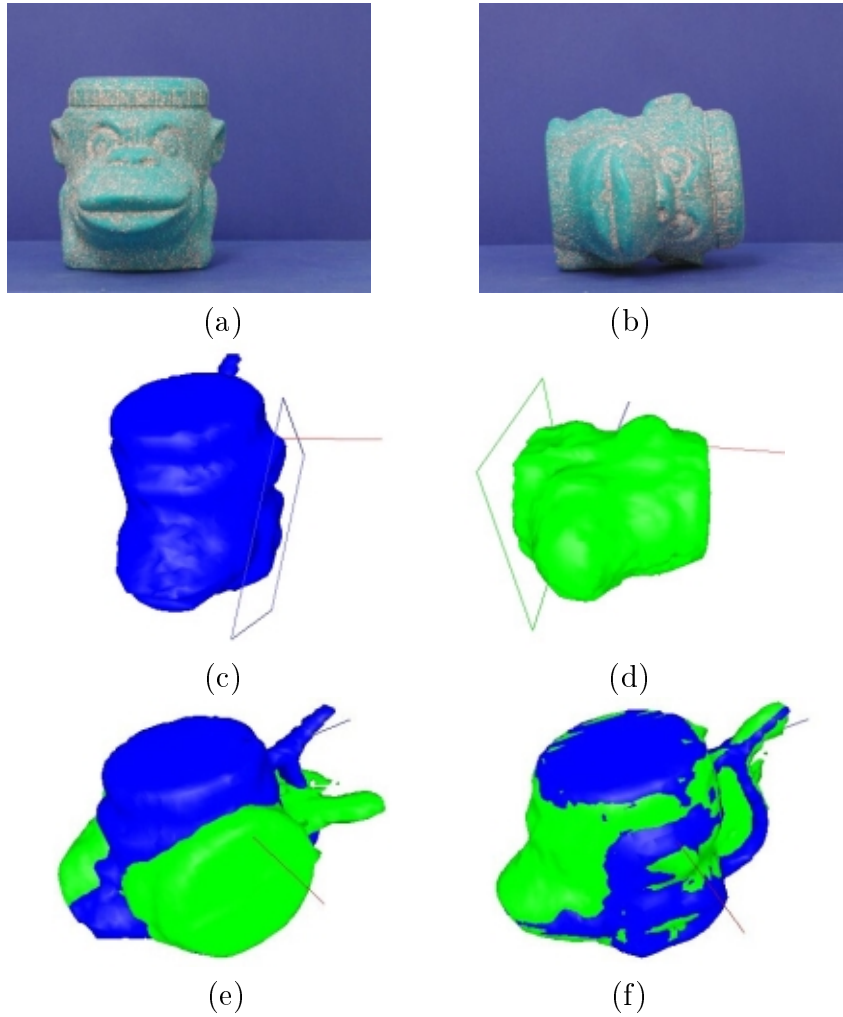


Figure 6.7: Pose estimation results of “Monkey” object. The object is placed on the table in two poses shown in (a) Pose1 (b) Pose2. (c) Pose1 model and its matching tangent plane to Pose2 (d) Pose2 model and its matching tangent plane to Pose1 (e) Original poses of the two models (f) Coarsely registered models

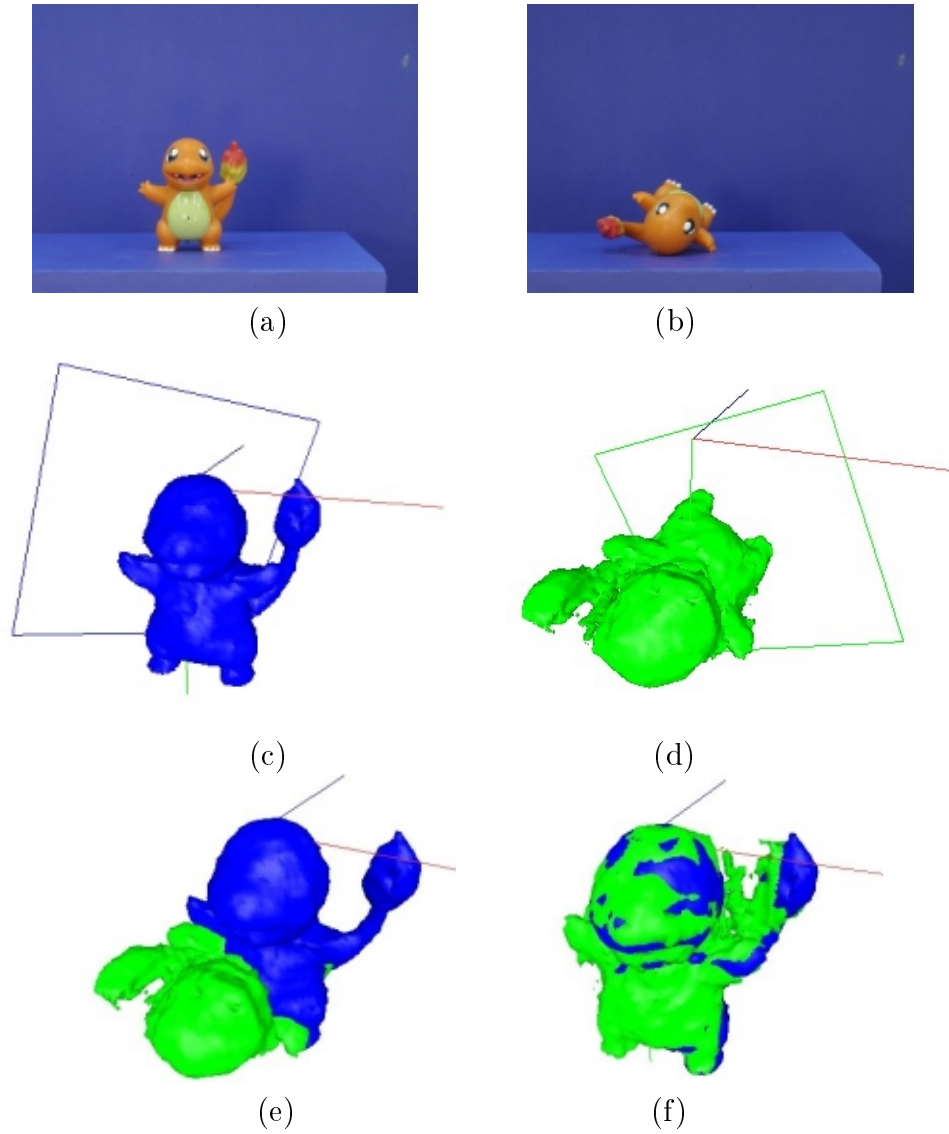


Figure 6.8: Pose estimation result of “Pokemon” object. (a) Pose1 (b) Pose2 (c) Pose1 model and its matching tangent plane to Pose2 (d) Pose2 model and its matching tangent plane to Pose1 (e) Original poses of the two models (f) Coarsely registered models

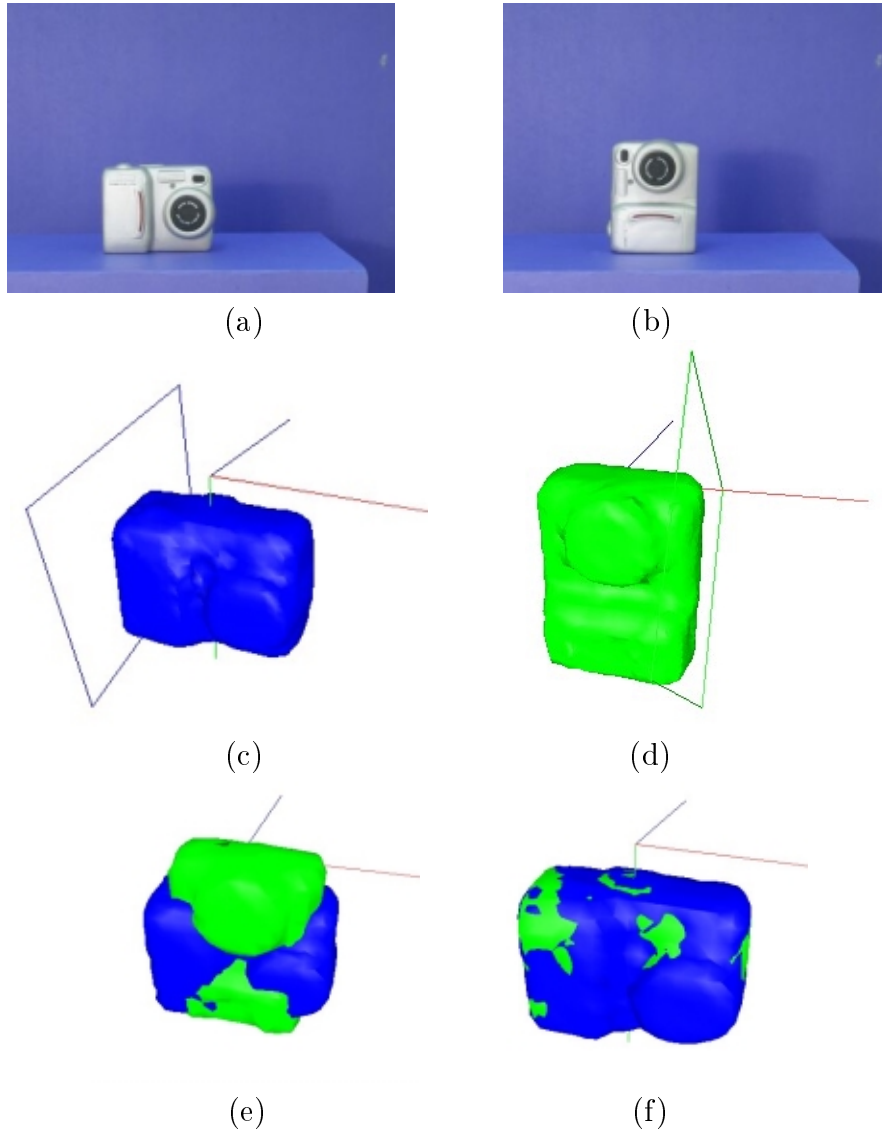


Figure 6.9: Pose estimation result of “Nikon775” object (a) Pose1 (b) Pose2 (c) Pose1 model and its matching tangent plane to Pose2 (d) Pose2 model and its matching tangent plane to Pose1 (e) Original poses of the two models (f) Coarsely registered models

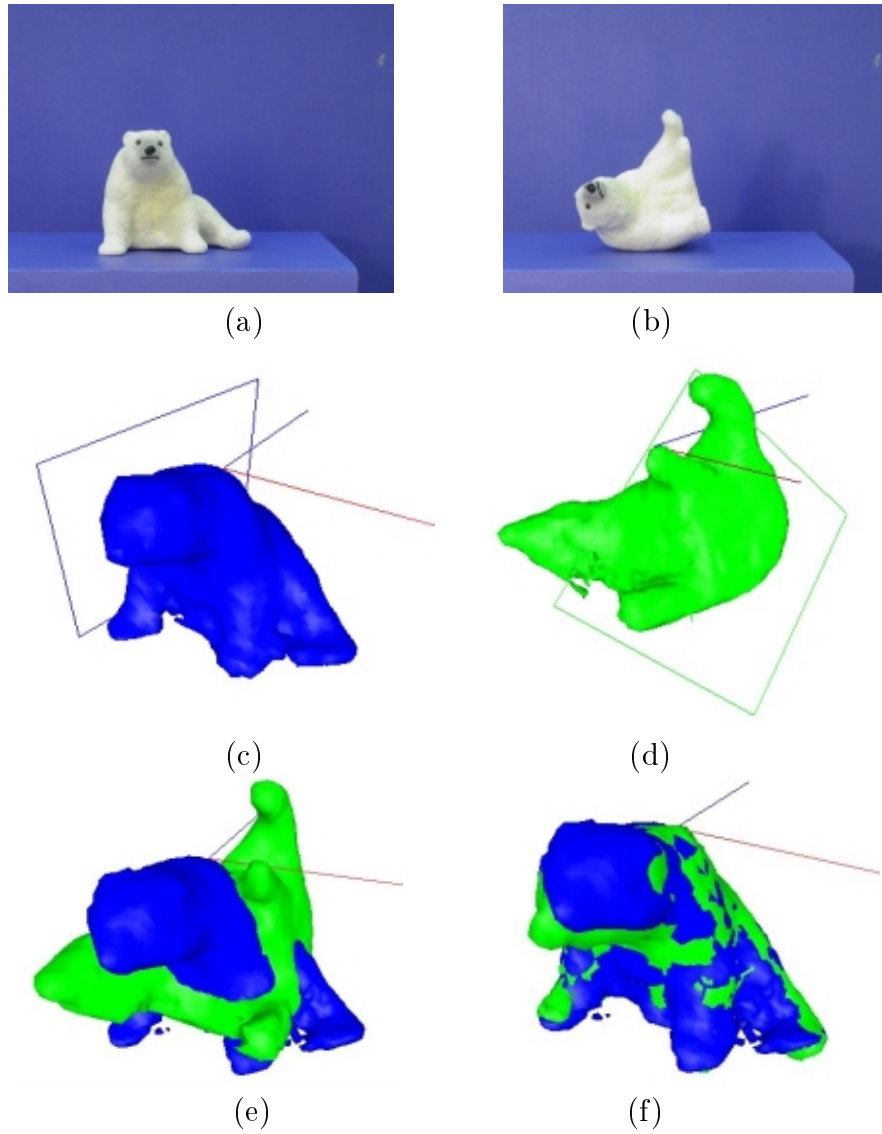


Figure 6.10: Pose estimation result of “PolarBear” object (a) Pose1 (b) Pose2 (c) Pose1 model and its matching tangent plane to Pose2 (d) Pose2 model and its matching tangent plane to Pose1 (e) Original poses of the two models (f) Coarsely registered models

Chapter 7

Pose Integration for Complete 3D Reconstruction

7.1 Introduction

In this chapter, we present a pose integration technique in order to merge two partial 3D models into a complete and closed 3D model. The novelty of the pose integration technique is merging two incomplete iso-surfaces which represent two different models. Since the two 3D models are reconstructed through integration of n -view range images in each pose, we integrate two iso-surfaces rather than $2 \times n$ -view range images. Voxel classification introduced in n -view integration presented in Chapter 5 is also employed. To solve a hidden surface problem, pose integration algorithm combines signed distance and class of a voxel in each pose. Texture mapping for photo-realistic 3D model is also presented for several real objects.

7.2 Pose Registration

Based on the estimated pose \mathbf{Q}_2^1 described in Chapter 6, we register and refine two 3D models before integration. We fix the first pose model and bring the second pose model (range image set) to the first. Refinement algorithm is based on a geometric registration technique which is similar to a multi-view registration [7]. Because registration for multi-view range images in a single pose is already refined, we refine the pose between two range image sets. Because the two range image sets are scanned from two different poses of an object, their scans have overlapping areas between *Pose1* (horizontal) scan and *Pose2* (vertical) scan as shown in Figure 7.1. From all partial surfaces in *Pose2*, we sample some of the vertices as control points, and find intersecting points from partial surfaces in *Pose1*. *Pose1* is set to the reference pose and *Pose2* is registered iteratively until registration error (translation and rotation between two control point sets) is less than a pre-defined threshold.

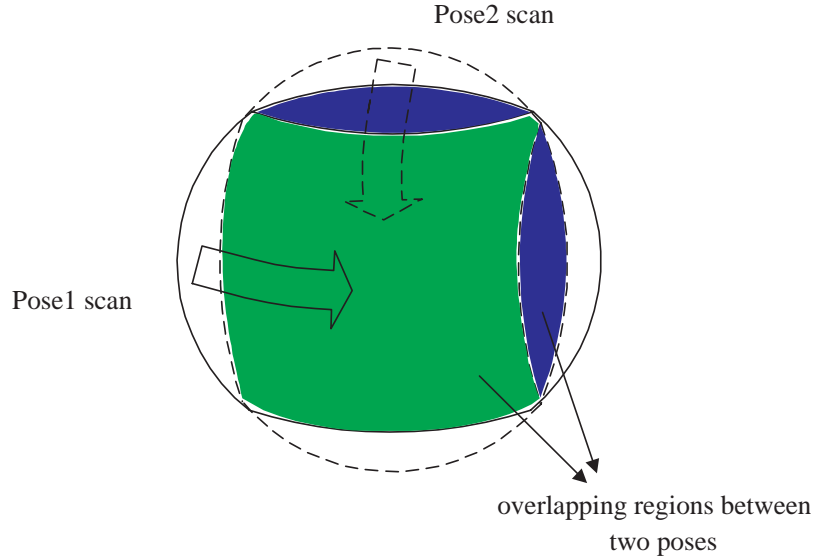


Figure 7.1: A schematic diagram of overlapping regions between two scans. Range image sets are scanned in vertical (solid line) and horizontal (dotted line) directions. Green (Light grey) and blue (dark grey) regions are overlapping.

7.3 Pose Integration

After pose registration, two 3D models are integrated into a complete 3D model. The final 3D model is represented as an implicit surface in a 3D volumetric space and converted to a triangular mesh model using Marching Cubes (MC) algorithm. Therefore, the basic algorithm of the pose integration is similar to multi-view integration. Pose integration computes the final signed distance $D_f(\mathbf{P})$ of a voxel, which is a function of signed distances in two poses,

$$D_f(\mathbf{P}) = f(D_1(\mathbf{P}), D_2(\mathbf{P})), \quad (7.1)$$

where, $D_i(\mathbf{P}) = f(d_0^i(\mathbf{P}), \dots, d_{N_o^i}^i(\mathbf{P}))$ for $i = 1, 2$,

and $D_i(\mathbf{P})$ is weighted signed distance of a voxel \mathbf{P} in pose i , $d_j^i(\mathbf{P})$ is the signed distance in pose i and view j , and N_o^i is the number of overlapping shapes in pose i . Computing $D_f(\mathbf{P})$ of the voxel \mathbf{P} is based on average of signed distance $D_i(\mathbf{P})$. If a voxel \mathbf{P} is seen from both poses, $D_f(\mathbf{P})$ is computed as a weighted average. However, if there is any occlusion or concavity in either pose, $D_f(\mathbf{P})$ is estimated based on the class of the voxel in both poses.

Suppose there is a concavity on the object's surface as shown in Figure 7.2. As shown in Figure 7.2(a), $O_{\mathbf{P}_1}$ is a common coordinate system of the first pose and all view points are almost on the XZ plane of the coordinate system. We see that no view in the first pose can observe the concavity. But in Figure 7.2(b), some of the views in the second pose can see the concavity. If a voxel \mathbf{P} is inside of unseen surface region as shown in Figure 7.2(a), it is classified as $\mathbf{P} \in P_{inside}$ at the first pose. And a continuation approach of the MC algorithm closes a mesh model by following the visual hull $VH_1(O)$ of the multi-view frustum as shown in the figure. This means there is a voxel \mathbf{P} which is close to the visual hull and classified as P_{inside} in the first pose. However, because it is seen from the second pose, there is no possibility that the MC algorithm marches on the same voxel. Instead, the algorithm follows the object's surface, because $\mathbf{P} \in P_{overlap}$ as in Figure 7.2(b).

The proposed pose integration technique is based on integration of two implicit models. If there are two implicit models, \mathcal{A} and \mathcal{B} , the merged model \mathcal{C} can be represented simply as $\mathcal{C} = \mathcal{A} \cap \mathcal{B}$. In this case, a final signed distance $D_f(\mathbf{p})$ can be represented as $D_f(\mathbf{P}) = \min\{D_1(\mathbf{P}), D_2(\mathbf{P})\}$, where $D_i(\mathbf{P})$ is a signed distance of \mathbf{P} in the pose i . But, in a real system, selecting the shorter distance may shrink the volume of the final model. Rather than selecting the minimum, we average two weighted signed distances, when $\mathbf{P} \in P_{overlap}^i$ for $j=1, 2$. Otherwise, we heuristically select one of the distances or the shorter one. Consequently, we select either the shorter one D_{shr} , the average D_{avg} , or one of the distances $D_i(\mathbf{P})$ according to the results of multi-view integration as follows.

- $D_f(\mathbf{P}) = D_{avg}(\mathbf{P}) = \frac{\sum W_i(\mathbf{P})D_i(\mathbf{P})}{\sum W_i(\mathbf{P})}$, if $\mathbf{P} \in P_{overlap}^1$ and $\mathbf{P} \in P_{overlap}^2$.
- $D_f(\mathbf{P}) = D_{shr} = \min\{D_1(\mathbf{P}), D_2(\mathbf{P})\}$, if $\mathbf{P} \in P_{inside}^1$ and $\mathbf{P} \in P_{inside}^2$.
- $D_f(\mathbf{P}) = D_1(\mathbf{P})$, if $\mathbf{P} \in P_{inside}^2$, or $D_f(\mathbf{P}) = D_2(\mathbf{P})$, if $\mathbf{P} \in P_{inside}^1$.

However, in some situations, a voxel can be in both $P_{nonoverlap}^1$ and $P_{nonoverlap}^2$. A voxel in a concave region may be assigned in these classes. Consider an example shown in Figure 7.3. In the figure, the signed distance from voxel \mathbf{P} to two pose models are $D_1(\mathbf{P})$ and $D_2(\mathbf{P})$ and $\mathbf{P} \in P_{nonoverlap}^1$ and $\mathbf{P} \in P_{nonoverlap}^2$. As shown in the figure, $D_1(\mathbf{P})$ has minus sign, even though the voxel is outside the object, because it is not visible from *pose1*. If $|D_1(\mathbf{P})| < |D_2(\mathbf{P})|$, the MC algorithm may choose the shorter one, then the voxel is considered to be inside the object. Typically, such mistakes may happen near the visual hull and an example of some artifacts of on the object's surface is shown in Figure 7.4(a).

Rather than selecting the shorter of the two distances in the two poses, we compare visibility of the voxel from available views in each pose. In other words,

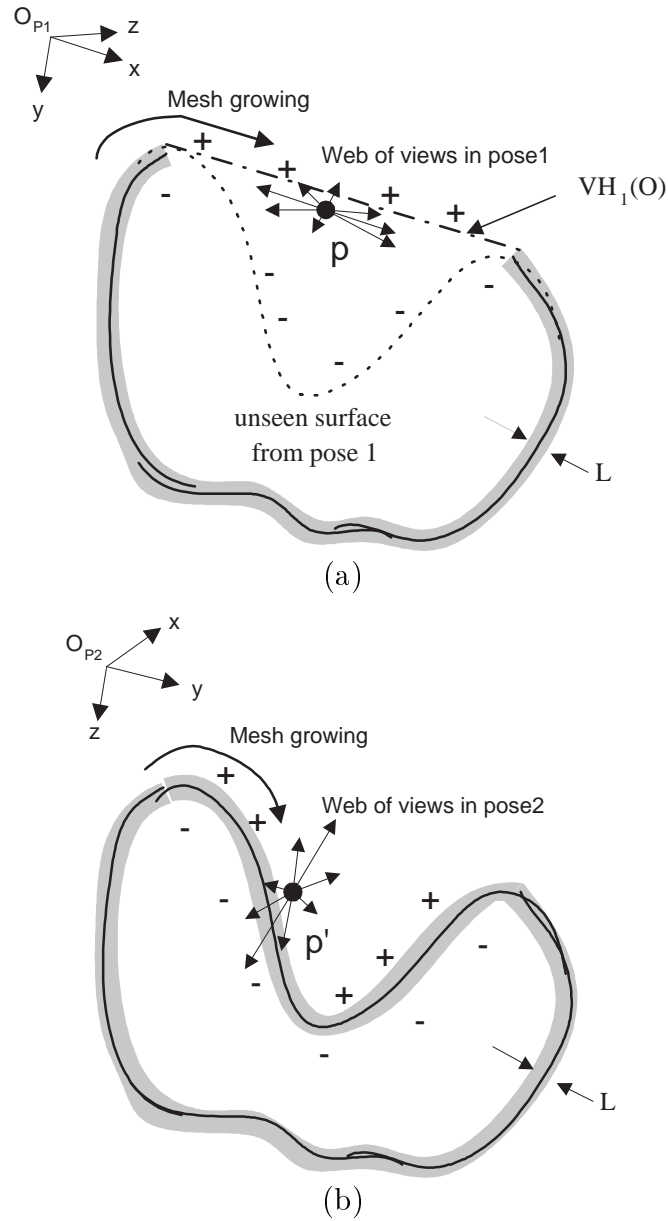


Figure 7.2: Signed distance in a concave region: (a) MC algorithm follows the visual hull $VH_1(O)$ to close the mesh model. (b) MC algorithm follows the real surface of the object.

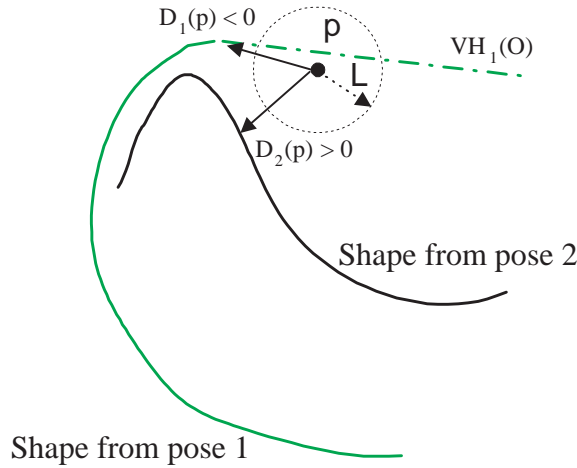


Figure 7.3: Errors of mesh growing in a concave region. If $|D_1(\mathbf{P})| < |D_2(\mathbf{P})|$, voxel \mathbf{P} has a wrong distance sign.

we check if the voxel is in $P_{nonoverlap}$ in both poses. If the voxel has many positive implicit distances for the views, it implies high visibility (our implicit representation assigns (+) sign when a voxel is outside an object). The visibility of the voxel can be considered to be the number of positive distances in each pose. The number can be easily determined by counting the number of positive distances of $d_j^i(\mathbf{P})$. Therefore, we define more conditions for computing $D_f(\mathbf{P})$ in $P_{nonoverlap}$. If a voxel $\mathbf{P} \in P_{nonoverlap}^1$ and $\mathbf{P} \in P_{nonoverlap}^2$,

- $D_f(\mathbf{P}) = D_1(\mathbf{P})$ if $count^+(d_j^1(\mathbf{P})) > count^+(d_j^2(\mathbf{P}))$.
- $D_f(\mathbf{P}) = D_2(\mathbf{P})$ if $count^+(d_j^2(\mathbf{P})) > count^+(d_j^1(\mathbf{P}))$.

where, $count^+(\cdot)$ is the count of the number of positive distances in $d_j^i(\mathbf{P})$, where $j = 0, \dots, N_o^i$. After applying these conditions for distance selection, we obtain an accurate reconstruction of the concave object's surface shown in Figure 7.4(b).

7.4 Texture Mapping

7.4.1 Texture Blending

Texture mapping is the last step of our 3D reconstruction. The reconstructed 3D mesh model consists of numerous triangles on its surfaces. We map image textures on

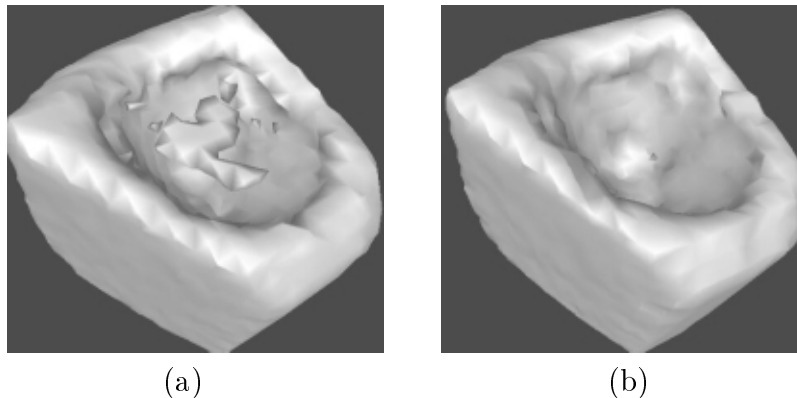


Figure 7.4: Artifacts in a concave region due to a reconstruction error. (a) Artifacts are near the visual hull. (b) Correct pose integration.

the surfaces of the model. There are several view-dependent and view-independent techniques for texture mapping [52, 58, 78]. We apply a general *view-independent texture mapping* technique. For each vertex on the object's surface, we decide the best-viewing point from all available views. The best view is selected based on the cosine angle between the vertex normal and viewing vectors. Suppose there is a vertex \mathbf{P} on the surface, then the best view of the vertex V_p^B is decided as

$$V_p^B = \underset{V_j^i}{\operatorname{argmin}} \{ \hat{\mathbf{V}}_j^i \cdot \hat{\mathbf{P}} \} \quad (7.2)$$

, where $0 \leq j \leq N^i - 1, i \in \{1, 2\}$. If the three vertices of a triangle map to the same view point, we texture the triangle using the image of the view. However, if they map to different view points, it is necessary to blend textures from different views to reduce the effect of texture blocking. Texture blocking may happen between two or more adjacent triangles, when they are textured by different view points. Because there can be difference of brightness or reflection between different views, texture blocking results in seams on the texture of a 3D model's surface. In order to reduce texture blocking, we interpolate textures on the triangle using the barycentric coordinate system [31, 80].

Suppose there is a triangle \mathcal{T} on a mesh \mathcal{M} in 3D space as shown in Figure 7.5. The triangle has three vertices, $\mathbf{P}_a, \mathbf{P}_b$, and \mathbf{P}_c . Decided by the best-view criterion, the vertices are supposed to map to the image of V_0, V_1 , and V_2 , respectively. Then we interpolate the texture inside the triangle \mathcal{T} using the barycentric coordinate system which consists of the images of three vertices.

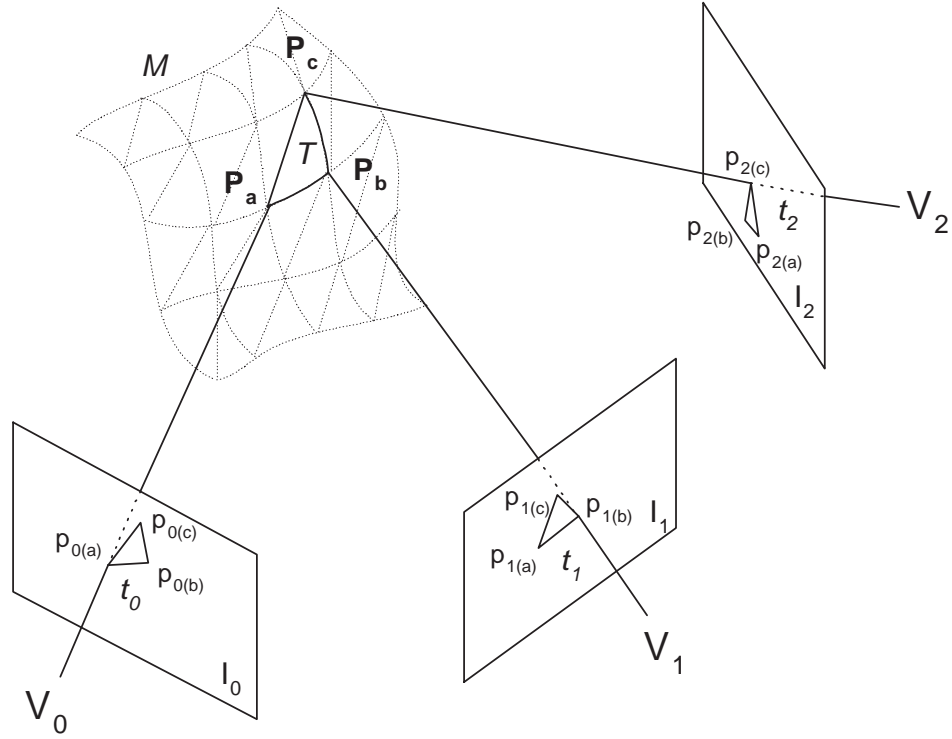


Figure 7.5: Texture blending on a triangle mesh

Suppose the image of a vertex \mathbf{P}_a is $\mathbf{p}_{0(a)}$ on the image plane I_0 of V_0 . Similarly, $\mathbf{p}_{1(b)}$ is the image of \mathbf{P}_b and $\mathbf{p}_{2(c)}$ is the image of \mathbf{P}_c , on image plane I_1 and I_2 , respectively. Then the color of a point \mathbf{P}_x inside of the triangle \mathcal{T} is determined by the colors of the three vertices and the barycentric coordinates

$$C(\mathbf{P}_x) = w_a C(\mathbf{p}_{0(a)}) + w_b C(\mathbf{p}_{1(b)}) + w_c C(\mathbf{p}_{2(c)}), \quad (7.3)$$

$$\text{where, } w_a = \frac{\Delta \mathbf{P}_x \mathbf{P}_b \mathbf{P}_c}{\Delta \mathbf{P}_a \mathbf{P}_b \mathbf{P}_c},$$

$$w_b = \frac{\Delta \mathbf{P}_x \mathbf{P}_a \mathbf{P}_c}{\Delta \mathbf{P}_a \mathbf{P}_b \mathbf{P}_c},$$

$$\text{and } w_c = \frac{\Delta \mathbf{P}_x \mathbf{P}_a \mathbf{P}_b}{\Delta \mathbf{P}_a \mathbf{P}_b \mathbf{P}_c}.$$

In the equation, $C(\mathbf{p})$ is the RGB color at a 2D point \mathbf{p} .

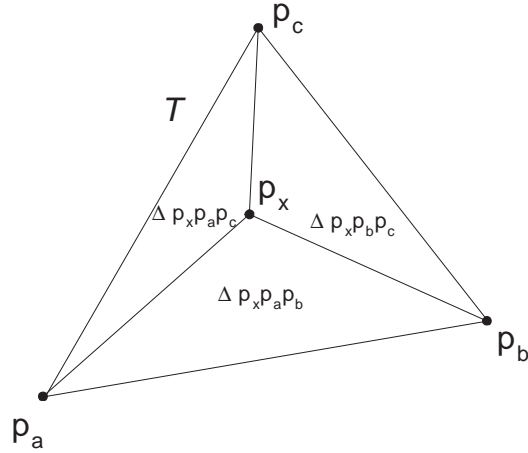


Figure 7.6: Barycentric coordinates for texture blending

7.4.2 Textures in Occlusion Area

The best-view criterion to decide the visibility of a vertex assumes that the vertex is seen from all view points. The visibility of the vertex to a view point is measured by the cosine angle between the vertex normal and the viewing vector. However, although an angle between a vertex normal and a viewing vector is the smallest of all available view points, there can be an occlusion from the vertex to the view point by some object's surfaces. An example of occlusion is shown in Figure 7.7. For three available views, V_0 , V_1 , and V_2 , the normal vector of a vertex \mathbf{P}_a has the smallest angle to the view point V_1 . However, it is also occluded by another surface. In this case we have to select the next best view point to map the texture. Therefore, when we decide the best-view from a vertex, we also need to check if the vertex is occluded from any view point.

Suppose we have all range images for all view points. Then we can easily decide if a vertex is occluded or not. For example, in Figure 7.7, let us consider the projection of a vertex \mathbf{P}_a to the image plane I_0 of the view V_0 . Then we obtain a 2D image point $\mathbf{p}_{0(a)}$ and we know the back projected 3D points from the 2D point is

$$d(\mathbf{p}_{0(a)}) \approx \mathbf{P}_a, \quad (7.4)$$

where, $d(\mathbf{p})$ is the range of a 2D point \mathbf{p} . Because the object's surface is the weighted average of all overlapping surfaces, $d(\mathbf{p}_{0(a)})$ will not be the same with the \mathbf{P}_a , but very close to it. In contrast, consider another projection to the image plane I_1 . Then

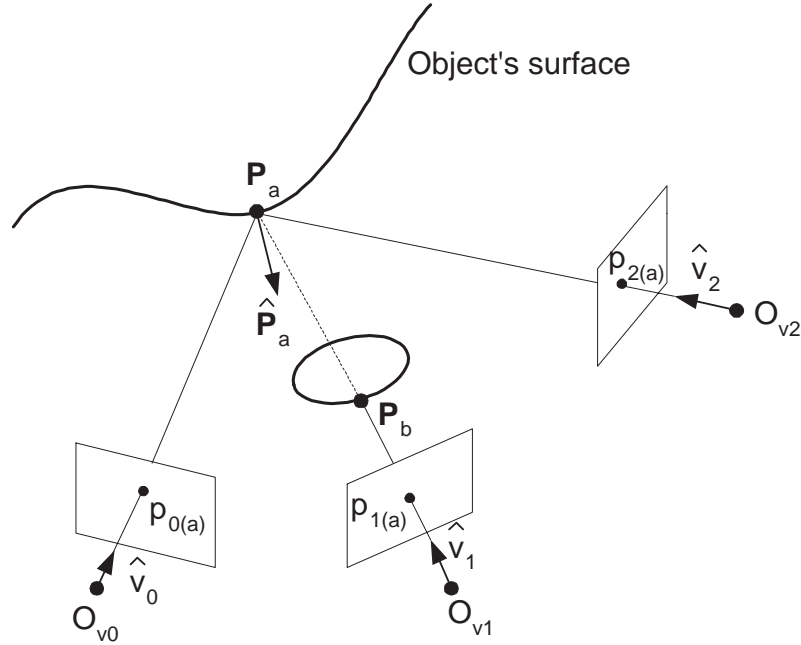


Figure 7.7: Visibility test for texture mapping

the range of the projected 2D point $\mathbf{p}_{1(a)}$ will be

$$d(\mathbf{p}_{1(a)}) \approx \mathbf{P}_b. \quad (7.5)$$

Because there is an occlusion between the view point V_1 and the point \mathbf{P}_a , the distance from \mathbf{P}_a to $d(\mathbf{p}_{1(a)})$ should be greater than a threshold such that

$$\|d(\mathbf{p}'_{a1}) - \mathbf{P}_a\| > \epsilon_{occ}. \quad (7.6)$$

By measuring the distance from a vertex and a 3D point which is inverse-projection of the image of the vertex, we decide the visibility of the vertex. The texture-mapped 3D model is saved to a file using Apple Computer's *3dmf* file format. The output file is displayed using a 3D graphic file viewer called *Quickdraw 3D Viewer*.

7.5 Experimental Results

We generate 3D models of the real objects presented in the previous chapter. Figure 7.8(a) and 7.8(b) are two 3D models of the “Monkey” object, reconstructed in

Table 7.1: Processing time in *sec* for pose integration

Object	Monkey	Polar Bear	Nikon775	Pokemon
Voxel size (mm)	4	4	4	3
No. of triangles	8096	5224	3128	5216
Time (sec)	7.0	6.0	5.1	6.2

two different poses. Figure 7.8(c) shows the result of integration of the two models. All surfaces of the object are reconstructed including the bottom surface as shown in the figure. Figure 7.8(d) shows some novel views of the integrated 3D model. The texture mapping results show accurate reconstruction on the surface including the top and the bottom.

Figure 7.9(a), (b), and (c) show 3D models of two poses and after integration for the “Pokemon” object. There are some occlusions in the second pose as shown in Figure 7.9(b), but the integrated 3D model in Figure 7.9(c) shows an accurate reconstruction. Texture mapping results in Figure 7.9(d) show novel views of the object. Figure 7.10(a), (b), and (c) show 3D models of the “Nikon775”. Texture mapping results are also shown in Figure 7.10(d). Figure 7.12 and Figure 7.13 are results of another object “Potatohead”. Figure 7.12 shows pictures of the first and the second poses of the object, their mesh models, and an integrated model. Figure 7.13 shows the texture mapped 3D model from novel view points. The figure shows that our techniques are able to generate complete and photo-realistic 3D models.

Integration time depends on the number of vertices and triangles. When there are about 4000 vertices and 8000 triangles on a 3D model, approximate computation time is about 5 minutes for pose registration refinement and 7 seconds for pose integration. Table 7.1 shows computation time for pose integration of the objects. Table 7.2 shows the number of hidden triangles after pose integration. Even after we integrate two 3D models for complete 3D model generation, there may be still hidden surfaces from all available views. When we consider a triangle is hidden when cosine angle of its surface normal is less than 0.0 degree with all viewing directions, for example, 4 triangles are still hidden in the “Pokemon” object. If we increase this angle for the decision of hidden surface, the number also increases. We may employ a new technique to reconstruct these hidden surfaces in the future.

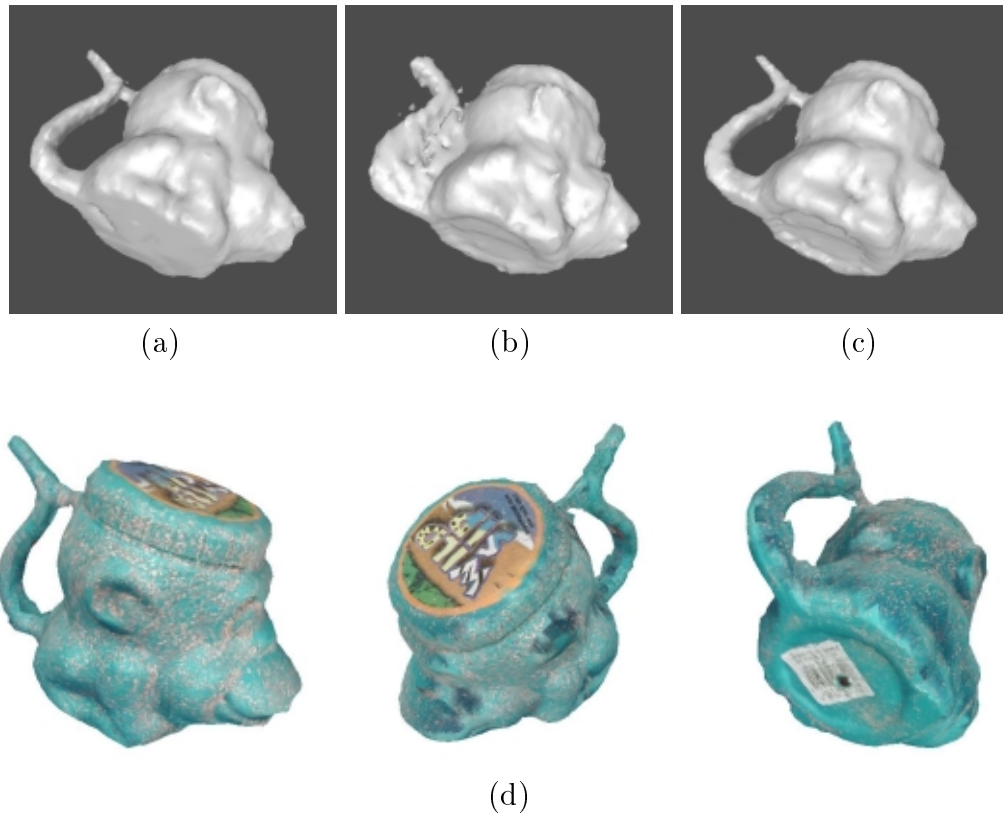


Figure 7.8: Pose integration and texture mapping results of 'Monkey' (a) Pose1 mesh model (b) Pose2 mesh model (c) Integrated model (d) Novel views of the object

7.6 Conclusions

A pose integration technique is presented for automatic and complete 3D model reconstruction. In order to avoid hidden surfaces that arise when only one pose is used, two poses are used. We introduce a novel technique for integrating two partial 3D models for two different poses. The voxel classification technique used in n -view integration shown in Chapter 5 is also employed to reconstruct accurate surface reconstruction. Because there can be a hidden surface from one view but not from another, integration algorithm combines the signed distance and the class of a voxel from each pose. Texture mapped 3D models of several real objects are presented. We also blend textures between mesh triangles whose vertices map to different view points according to the best-view criterion. Texture blending reduces blocking effect of brightness difference between different view points. Experimental

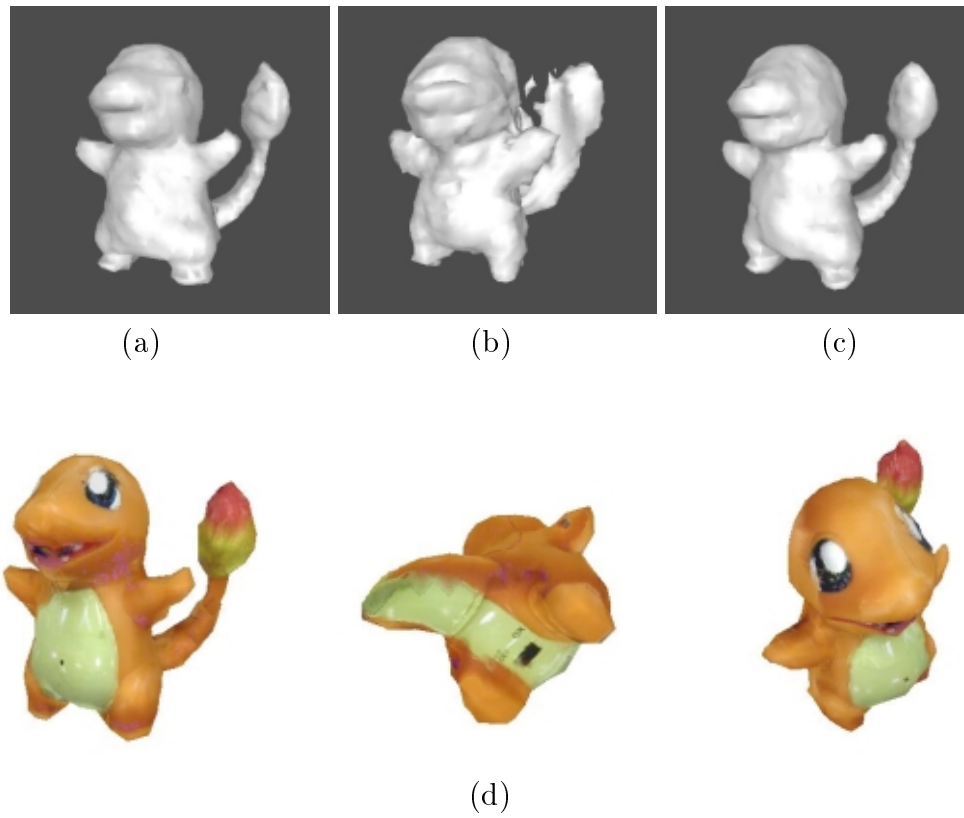


Figure 7.9: Pose integration and texture mapping results of 'Pokemon' (a) Pose1 mesh model (b) Pose2 mesh model (c) Integrated model (d) Novel views of the object

results for several real objects show our pose integration technique is very effective for 3D model reconstruction.

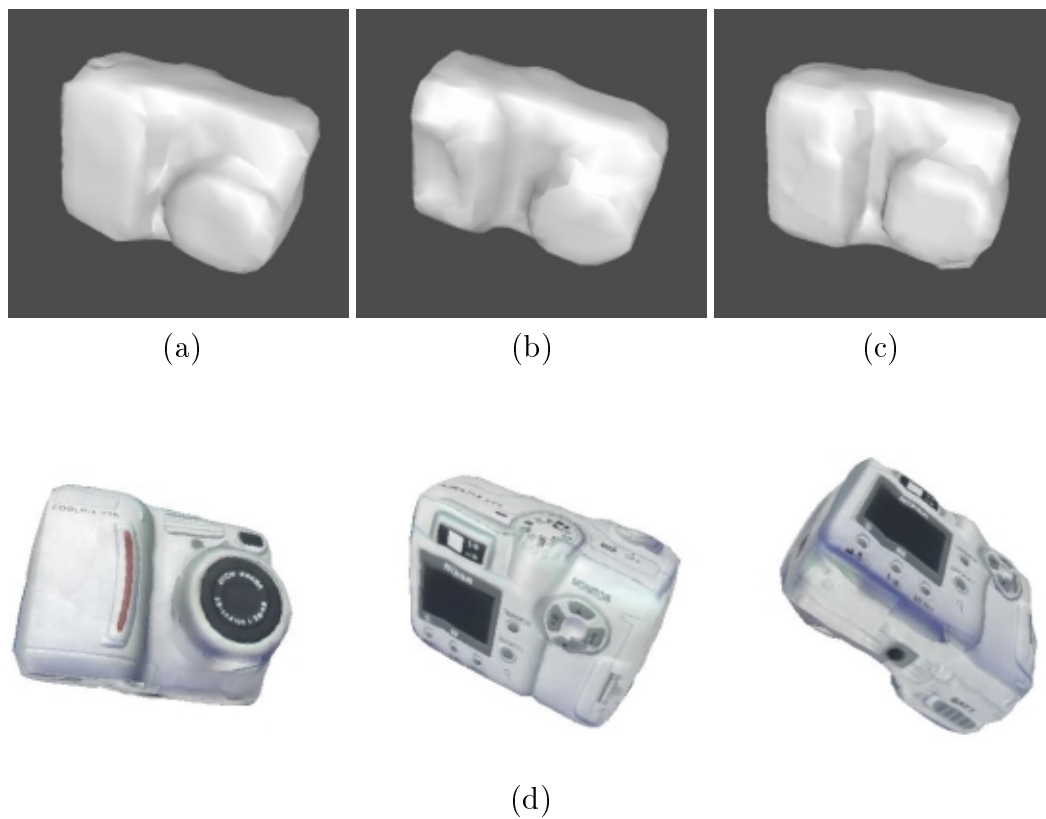


Figure 7.10: Pose integration and texture mapping results of 'Nikon775' (a) Pose1 mesh model (b) Pose2 mesh model (c) Integrated model (d) Novel views of the object

Table 7.2: Number of hidden surfaces after pose integration

Surface normal constraint for hidden surface	Monkey	Polar Bear	Nikon775	Pokemon
No. of triangles	8096	5224	3128	5216
Hidden if $\cos(\theta) < 0.0$	0	4	0	4
Hidden if $\cos(\theta) < 0.1$	46	13	0	15
Hidden if $\cos(\theta) < 0.2$	84	13	0	22

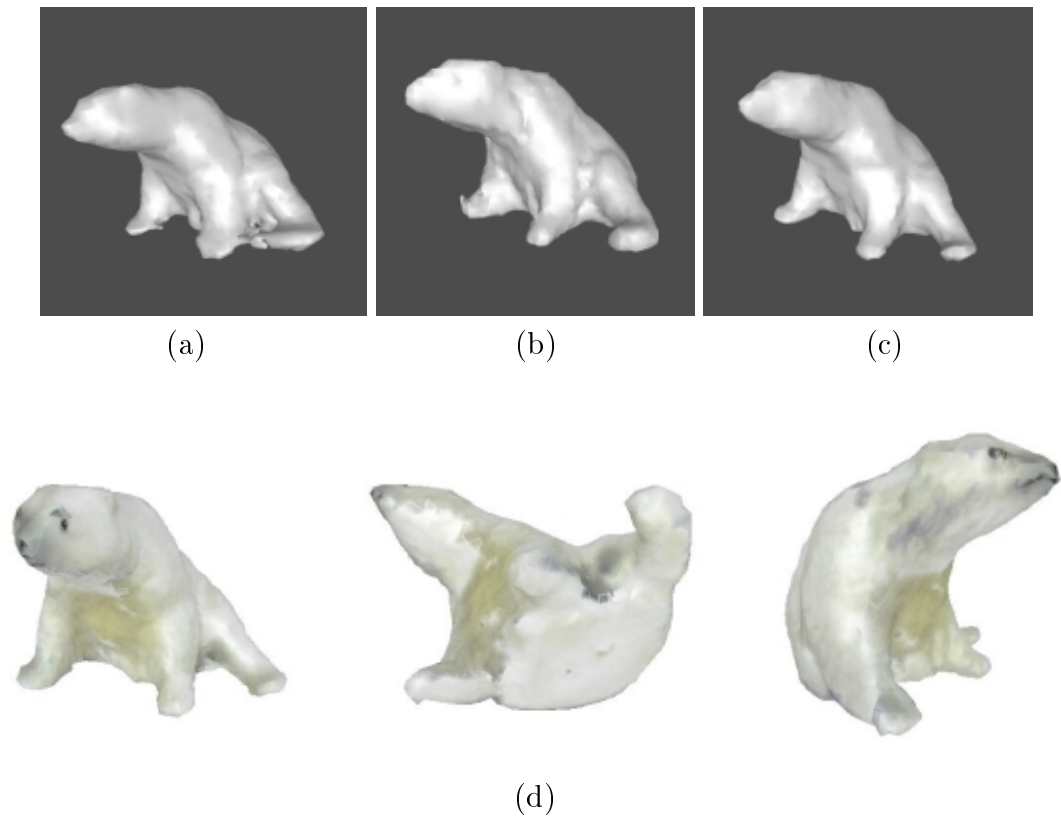


Figure 7.11: Pose integration and texture mapping results of 'PolarBear' (a) Pose1 mesh model (b) Pose2 mesh model (c) Integrated model (d) Novel views of the object

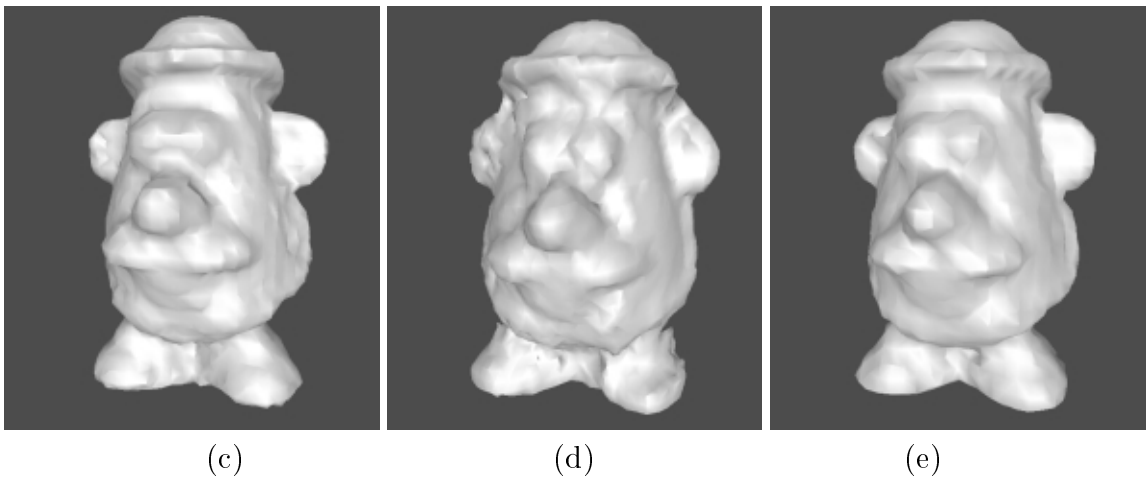
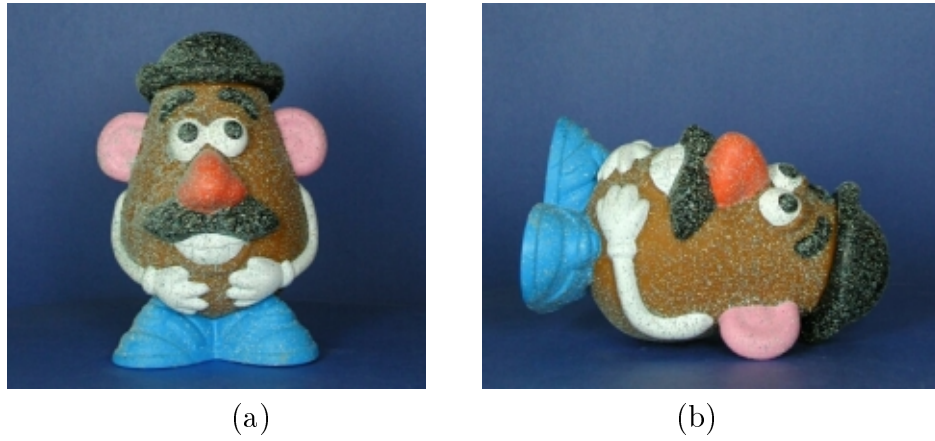


Figure 7.12: Pose integration results of 'PolarBear' (a) Pose1 of the object (b) Pose2 of the object (c) Pose1 mesh model (d) Pose2 mesh model (e) Integrated model



Figure 7.13: Texture mapped results of 'Potatohead'

Chapter 8

An Efficient Point-to-Plane Registration Technique

In this chapter, we address a registration refinement problem and present an accurate and fast *Point-to-(Tangent) Plane* registration technique. We introduce a novel *Point-to-Plane* registration technique by combining fast *Point-to-Projection* approach.

8.1 Introduction

Registration refinement of multiple range images is an essential step in multi-view 3D modeling. When the range images are coarsely registered by *a priori* knowledge of registration parameters, registration refines the rigid transformation parameters to minimize alignment error between overlapping range surfaces. There have been many investigations on the refinement problem [14, 81]. For a pair-wise registration, the registration problem can be considered to be an error minimization of the rigid body transformation between two control point sets [4]. Control point sets are determined by matching either geometric [9], photometric [52, 101], or geometric and photometric [8, 45] structures of the range surfaces. In this section, we address a registration problem in the first category, which considers only geometric structure for control point matching.

Based on the method of control point matching, three approaches can be considered in general. *Point-to-Point*, e.g. Iterative Closest Point (ICP) algorithm [9], *Point-to-(Tangent) Plane* [17, 7], and *Point-to-Projection* [10] techniques are very common. The ICP algorithm is one of the common techniques for refinement of partial 3D surfaces (or models) and many variant techniques have been investigated. However, searching the closest point for the ICP algorithm is a computationally expensive task. In order to accelerate the speed of closest point searching, some searching techniques are typically employed, for example kd-tree searching, z-buffering, or closest-point caching [67, 6].

In contrast, *Point-to-Projection* approach finds the correspondence of a source control point by projecting the source point onto a destination surface from the point of view of the destination [10, 64]. The destination control point is a forward projection of a 2D point, which is an image of the source point to the image plane of the destination surface. This approach makes registration very fast, because it does not involve any step for correspondence searching. However, one disadvantage is that the result of registration is not as accurate as those of the others [81].

Among the three approaches, *Point-to-(Tangent) Plane* technique is known to be the most accurate [75, 81]. From a source control point, the matching control point is the projection of the source point onto the tangent plane at a destination surface point which is the intersection of the normal vector of the source point [7, 17, 30]. However, finding the intersection on the destination surface is also computationally expensive. One of the acceleration techniques is first searching the closest point, and finding the intersecting surface (or the triangle) from its neighboring triangles [81]. Gagnon [30] tries to find the intersection on a 2D grid of range images along the projection of the normal vector of the source point.

In this section, we address the registration problem to refine a pair of range surfaces as well as multi-view range surfaces. We propose an accurate and fast *point-to-plane* registration technique. We combine advantages of *point-to-plane* and *point-to-projection* techniques simultaneously. We employ accuracy of *point-to-plane* technique and speed of *point-to-projection* technique. In order to find the intersection control point, we project a source point to the destination surface, re-project the projection point to the normal vector of the source point. We present that iterative normal projections converge to the intersection point. By assuming that the destination surface is a monotonic function in a new 2D coordinate system, we show a contraction mapping property of our registration technique [63]. Therefore, we call our technique as *Contractive Projection Point* (CPP) algorithm. Experimental results for several 3D models are presented for many single pair registrations as well as a multi-view registration.

8.2 Comparison of Registration Techniques

8.2.1 Description of Three Registration Techniques

In this section, we briefly describe three registration refinement techniques. There have been many variants, but we present basic principle of each technique. Suppose there is a control point set $\{\mathbf{P}\}$ on the source surface \mathcal{S}_p , and another control point set $\{\mathbf{Q}\}$ on the destination surface \mathcal{S}_d . If we have K control points on each surface and $k = 0, \dots, K - 1$, then the registration problem is estimating a rigid body

transformation $\mathbf{T} = [\mathbf{R}|\mathbf{t}]$ which minimizes an alignment error ϵ such that

$$\epsilon = \sum_{k=0}^{K-1} \|\mathbf{Q}_k - (\mathbf{R}\mathbf{P}_k + \mathbf{t})\|^2. \quad (8.1)$$

In general, the source control point set $\{\mathbf{P}\}$ are selected by sampling the source surface-randomly or uniformly, and filtered by some constraints to delete unreliable control points. The destination control point set $\{\mathbf{Q}\}$ is then the conjugate of the source point set, which is determined by a matching criterion.

Point-to-point technique is the most common technique, and ICP is a well-known registration algorithm. From a source control point \mathbf{P} on the source surface, ICP algorithm searches the closest point \mathbf{Q} on the destination surface. Figure 8.1(a) shows a basic diagram of ICP algorithm. Error metric for the ICP algorithm is the distance d_s between two control points. The most time-consuming task in ICP algorithm is searching the closest point. Suppose there are total N_P and N_Q points on the source and the destination surfaces, respectively. If there are K source control points on the source surface, a brute force searching requires $O(KN_Q)$ computations. In order to reduce the computation complexity, some high-speed searching techniques are employed such as kd-tree searching [81, 67]. Using a kd-tree searching, the computation complexity can be reduced to $O(K \log N_Q)$.

Point-to-plane registration is another common technique. It searches the intersection on the destination surface from the normal vector of the source point. As shown in Figure 8.1(b), the destination control point \mathbf{q}' is the projection of \mathbf{p} onto the tangent plane at \mathbf{q} which is the intersection from the normal of \mathbf{p} .

Point-to-projection approach is known to be a fast registration technique. As shown in Figure 8.1(c), this approach determines a point \mathbf{q} which is the conjugate of a source point \mathbf{p} , by forward-projecting \mathbf{p} from the point of view of the destination \mathbf{O}_Q . In order to determine the projection point, \mathbf{p} is first backward-projected to a 2D point \mathbf{p}_Q on the range image plane of the destination surface, and then \mathbf{p}_Q is forward-projected to the destination surface to get \mathbf{q} . This algorithm is very fast because it does not include any correspondence searching step. However, one of its disadvantages is that the result of registration is not as accurate as those of the others.

8.3 Contractive Projection Point (CPP) Technique

8.3.1 Combinig Point-to-Plane and Point-to-Projection Techniques

In this section, we propose a novel *point-to-plane* registration technique by combining fast searching property of *point-to-projection* technique. Suppose there are two

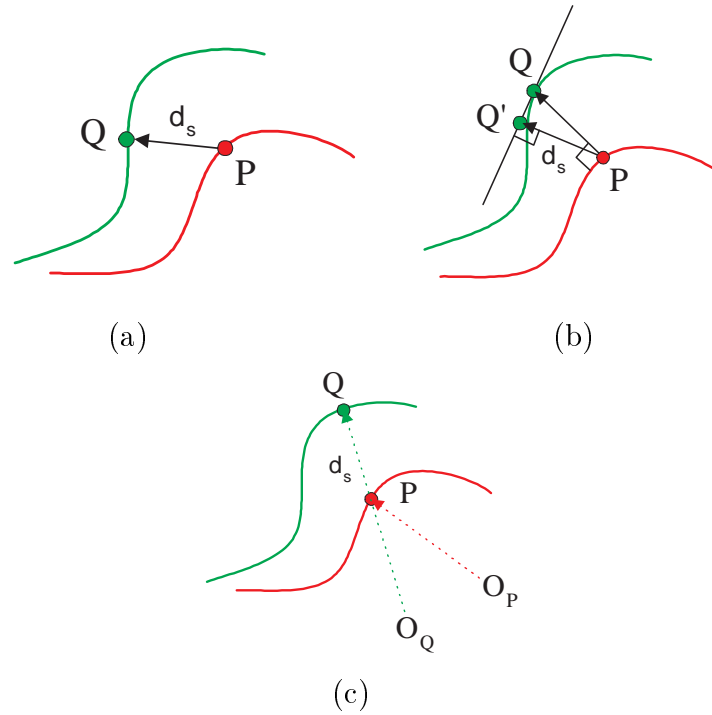


Figure 8.1: Three common registration techniques. (a) Point-to-point (b) Point-to-plane (c) Point-to-projection

partial surfaces \mathcal{S}_P and \mathcal{S}_Q as shown in Figure 8.2. We assume that they are acquired from two different views P and Q of an object and coarsely registered to the world coordinate system by using the transformation matrices \mathbf{T}_P and \mathbf{T}_Q , respectively. Let us also suppose there is a source control point \mathbf{P}_0 on the surface \mathcal{S}_P . Then the problem in the *point-to-plane* registration is finding the intersecting point \mathbf{Q}_s on the surface \mathcal{S}_Q as shown in the figure. The point \mathbf{Q}_s is an intersection on the surface \mathcal{S}_Q by the normal vector $\hat{\mathbf{p}}$ of \mathbf{P}_0 .

One of the typical methods of searching the intersection is first finding a triangle (when the surface consists of triangles) which is intersected by the normal vector $\hat{\mathbf{p}}$. Then the intersection point \mathbf{Q}_s can be interpolated by three vertices on the triangle. With a brute force searching for example, it takes about tens of seconds to find the intersecting point among tens of thousands triangles using a typical personal computer. In addition, if there are hundreds of control points on the source surface, it will take about tens of minutes to find the all matching points. We may also use a

fast searching algorithm to first find the closest point, and then to try its neighborhood triangles to find the intersection triangle. However, it does not guarantee that there is an intersection triangle on the neighborhoods.

We present a new *point-to-plane* registration technique by employing advantages from both techniques. The core idea is using *iterative point-to-projections* to estimate the position of the intersection on the destination surface as shown in Figure 8.2. Let us backward-project the point \mathbf{P}_0 to a 2D image point

$$\mathbf{p}_q = \mathbf{M}_Q \mathbf{T}_Q^{-1} \mathbf{P}_0, \quad (8.2)$$

where \mathbf{M}_Q is the perspective projection matrix of view Q to the image plane I_Q , and \mathbf{T}_Q is the transformation matrix from the camera coordinate system of view Q to the world coordinate system. Then let us forward-project \mathbf{p}_q to a new 3D point \mathbf{Q}_{p0} . Forward-projection \mathbf{Q}_{p0} is the range at the destination image point \mathbf{p}_q . The point \mathbf{Q}_{p0} is computed by interpolating a grid of destination image plane and transformed back to the world coordinate system. This point is then the matching point of \mathbf{P}_0 in a typical *point-to-projection* technique. Now let us consider another projection of \mathbf{Q}_{p0} to the normal vector $\hat{\mathbf{p}}$ at \mathbf{P}_0 . Then we obtain a new 3D point \mathbf{P}_1 , such that

$$\begin{aligned} \alpha &= (\mathbf{Q}_{p0} - \mathbf{P}_0) \cdot \hat{\mathbf{p}}, \\ \mathbf{P}_1 &= \mathbf{P}_0 + \alpha \hat{\mathbf{p}}. \end{aligned} \quad (8.3)$$

If we iterate the same projections above using the new control point \mathbf{P}_1 , then we get the next source point \mathbf{P}_2 , and so on. If there is an intersection on the surface \mathcal{S}_Q by the normal vector $\hat{\mathbf{p}}$, then the point \mathbf{Q}_{pi} (or \mathbf{P}_i) for the i th projection converges to \mathbf{Q}_s when i goes to infinity, such that

$$\lim_{i \rightarrow \infty} \|\mathbf{P}_i - \mathbf{Q}_{pi}\| \rightarrow 0. \quad (8.4)$$

However in real situation, only small number of projections can make a convergence measure

$$\epsilon_c = \|\mathbf{P}_i - \mathbf{Q}_{pi}\| \quad (8.5)$$

become close to zero. If we find all convergence points for corresponding source control points, we can find destination control points on their tangent planes to run the *point-to-plane* registration technique.

8.3.2 Contraction Mapping Property of CPP

In order to show the validity of the algorithm presented in the previous section, we present contraction mapping properties of the algorithm. When a source control

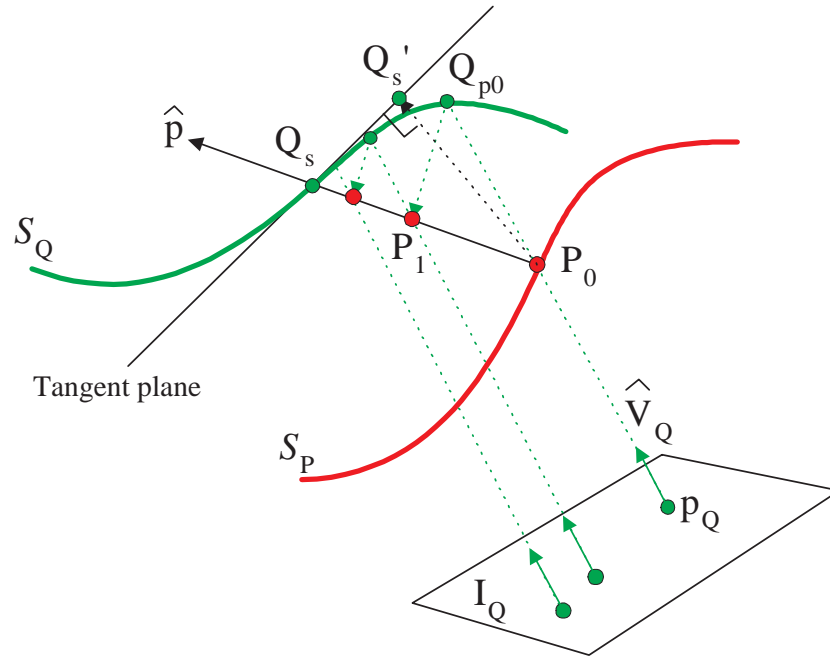


Figure 8.2: Finding the intersecting point Q_s from P_0 by the proposed algorithm.

point converges, the proposed iterative projection has contraction mapping properties such that the convergence measure ϵ_c becomes close to zero after a couple of iterations. Because we employ the contractive point-to-projection technique to estimate the intersection, we call the proposed technique as *Contractive Projection Point* (CPP) algorithm.

The definition of the contraction mapping is as the followings.

Definition 1 Let (X, d) be a metric space and $f : X \rightarrow X$. We say a contraction mapping, if there is a real number $k, 0 \leq k < 1$, such that

$$d(f(x), f(y)) \leq kd(x, y)$$

for all x and y in X .

Let us consider a 2D coordinate system as shown in Figure 8.3. When there are two different view points P and Q , the new axes consist of the viewing vector \hat{V}_Q of Q and the normal vector \hat{p} of the source control point P_0 . And the point P_0 becomes the origin of the coordinate system. Then surface S_Q becomes a contour on the 2D plane, which is the intersection of the surface with the 2D plane. As shown in

the figure, the proposed algorithm projects the source point \mathbf{P}_i to a new point \mathbf{P}_{i+1} , iteratively until the convergence measure in Equation (8.5) becomes close to zero.

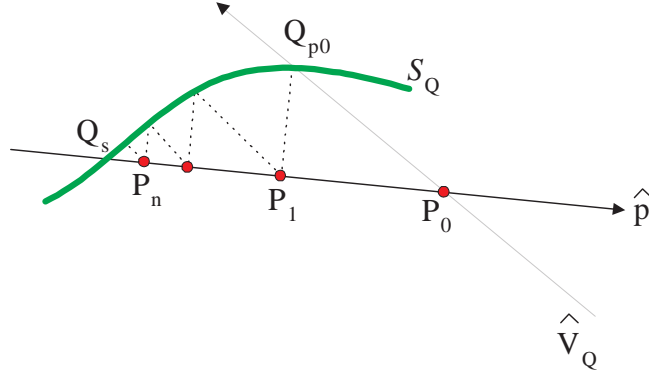


Figure 8.3: Contraction mapping property of a new 2D coordinate system.

In real case, distance from \mathbf{p}_0 to the contour is very small, because two surfaces are coarsely registered before the refinement. Therefore we consider that all forward-projecting lines from the view origin of Q to all control point \mathbf{p}_i are almost parallel. Let us then present the contraction mapping properties of the proposed algorithm by the following theorem.

Theorem 1

1. A surface \mathcal{S}_Q is a monotonic function with respect to the viewing vector $\hat{\mathbf{V}}_Q$.
2. A vector product $\hat{\mathbf{V}}_Q \cdot \hat{\mathbf{P}}_0 < 0$, because we only consider a source point which is seen from the view Q . Therefore there is always a projection of $\mathbf{Q}_{\mathbf{p}_i}$ to the normal vector $\hat{\mathbf{P}}_0$.
3. If there is an intersection \mathbf{Q}_s , a function $f : \mathbf{P}_i \rightarrow \mathbf{P}_{i+1}$ (or $\mathbf{Q}_{\mathbf{p}_i} \rightarrow \mathbf{Q}_{\mathbf{p}_{i+1}}$) is a contraction mapping when $\|\mathbf{P}_{i+1}, \mathbf{P}_{i+2}\| < k\|\mathbf{P}_i, \mathbf{P}_{i+1}\|$, where $0 \leq k < 1$.
4. Then, there is a 3D point \mathbf{Q}_s on the surface such that $f(\mathbf{Q}_s) = \mathbf{Q}_s$.

8.3.3 Convergence Conditions

In an ideal situation, the CPP algorithm makes the source point always converge to a convergence point. However, in a real situation, it may diverge or enter into a non-convergent cycle. Therefore we need to find out convergence condition according to the contraction coefficient k . Three possible cases are shown in Figure 8.4. In

Figure 8.4(a), a source control point converges and the coefficient is $0 \leq k < 1$. In this case, we can use the convergence point as a matching control point of its conjugate.

However, as shown in Figure 8.4(b), the control point could diverge if the coefficient is $k > 1$. This could happen when the tangent normal at a point on surface \mathcal{S}_Q has a high angle with respect to $\hat{\mathbf{V}}_Q$. However, we can reduce the effect of diverging control points by restricting some source points which also have high angle with respect to $\hat{\mathbf{V}}_Q$. However, if the convergence measure ϵ_c is greater than the initial measure $\|\mathbf{Q}_{p0} - \mathbf{P}_0\|$, then we discard the destination point. The last case is that the projective mapping enters into a non-convergent cycle when $k = 0$, as shown in Figure 8.4(c). This case could happen when a segment of the destination curve is overlapping with the projection line from \mathbf{Q}_{p_i} to \mathbf{P}_{i+1} . We also solve this problem by checking the error measure over some iterations. If the measure is repeating for some iterations, we consider the mapping enters into a non-convergent cycle and stop the searching.

8.3.4 CPP Algorithm

The proposed CPP algorithm can be easily implemented by a recursive searching program. A pseudo code of the CPP algorithm is written as follos. The *GetIntersection()* function searches a 3D point \mathbf{Q}_s , the intersection of a control point \mathbf{P} on the destination surface *Dest*. The *Dest* surface has view transformation matrix $Dest \rightarrow \mathbf{T}$ and the rotation $Dest \rightarrow \mathbf{R}$ with respect to a common coordinate system. The *RecursiveProjection()* function is a recursive searching program to find the intersection. The N_c is the maximum iterations and D_c is the threshold for the convergence error ϵ_c .

```

Int GetIntersection(Vertex * P, Vec3d &Qs, Mesh * Dest)
{
    P0 = (Dest → T)-1 (P → coord);
    p̂ = (Dest → R)-1 (P → normal);
    Vec3d Pc = RecursiveProjection(P0, p̂, Qs, Dest, 0);
    if (Pc == NULL) return 0; //Nointersection;
    εc = ‖Pc - Qs‖;
    if (εc > ε0) return -1; //Diverge
    Qs = (Dest → T) Qs; //Intersection
    return 1;
}

Vec3d RecursiveProjection(Vec3d &P1, Vec3d &p̂, Vec3d &Qp1, Mesh * Dest, int cnt)
{
    if (cnt > Nc) return P1;

```

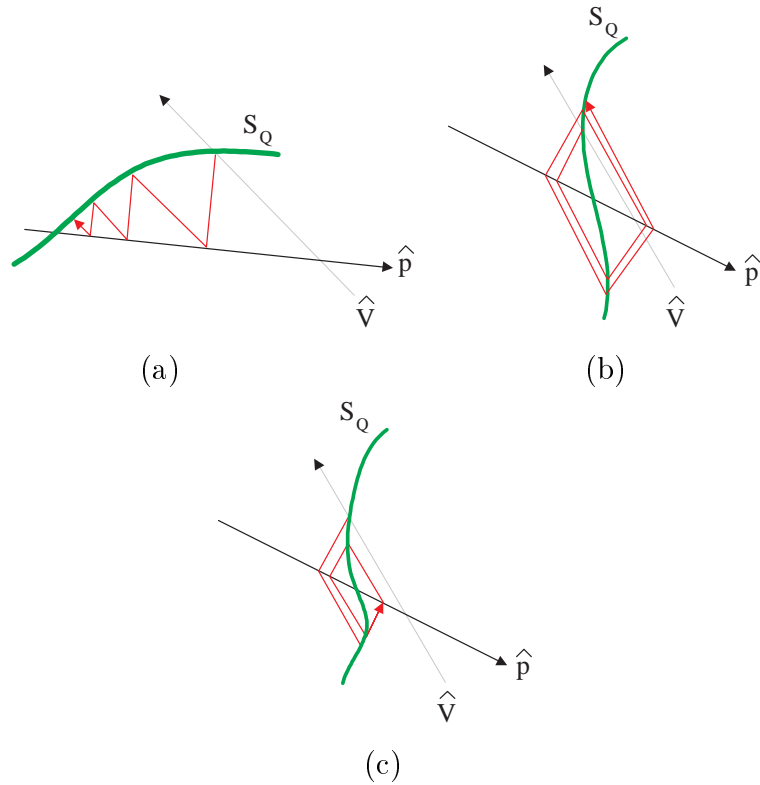



Figure 8.4: Three cases of projections to a normal vector. (a) Converge ($0 \leq k < 1$) (b) Diverge ($k > 1$) (c) Infinity loop ($k = 1$)

```

pq = MqP1;
flag = OnObject(pq, P1, Dest, Qp1);
if (!flag) return NULL;
α = (Qp1 - P1) · p-hat;
P2 = P1 + αp-hat;
εc' = ||P2 - Qp1||;
if (cnt == 0) ε0 = εc';
if (εc' < Dc) return P2;
return RecursiveProjection(P2, p-hat, Qp1, Dest, cnt + 1);
}

```

After finding all matching control points, we use two control point sets \mathbf{P}_0 and \mathbf{Q}_s for estimating the rigid transformation between two surfaces. The transformation

$\mathbf{T} = [\mathbf{R}|\mathbf{t}]$ is computed by a SVD (Singular Value Decomposition) technique [4].

8.4 Experimental Results

8.4.1 Test Objects

We have tested the proposed algorithm for three objects. The three test objects are shown in Figure 8.5. The *Wave* object is a pair of 3D sinusoidal surfaces, which is synthesized with 320×240 image resolution. Two surfaces have 10 degree of phase difference in the XY image plane (Z axis is along the height of the object) and 10 mm of translation along the Z axis. The object's height is 50 mm and we add random noise to every image point with maximum 10 % of the height. The second object *Angel* is a pair of range images obtained from a laser range finder. This object is one of the models in the Ohio State University's Range Image Database. The two images are obtained by rotating the object 20 degrees. Because two images are obtained from a range sensor, it has little noise on the surfaces and little distortion between overlapping part of surfaces. The third object *Potatohead* consists of multiple range images. We obtain the range images from 8 different views by rotating a real object by 45 degrees as described in a previous chapter. We use a stereo camera and a stereo matching technique to obtain the range images. Due to the inherent matching problem, surfaces of the object show some high-peak noises in Figure 8.5(c). We have 8 partial surfaces for the object, but Figure 8.5(c) shows only the first two partial views. The number of points and triangles on the test objects are shown in the Table 8.1. They are the number of points and triangles on the surface of the first view.

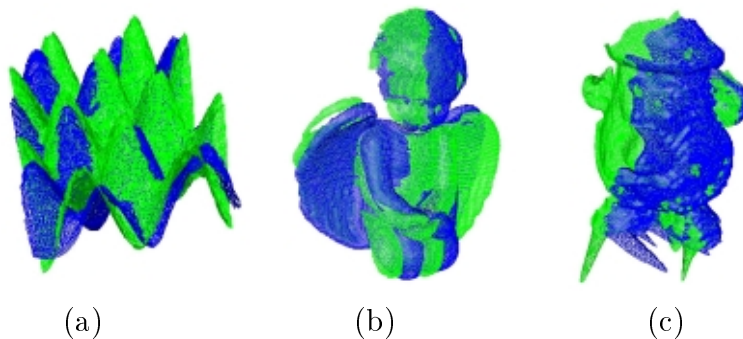


Figure 8.5: Point clouds models of test objects. (a) Wave (b) Angel (c) Potatohead

Table 8.1: Test objects for IPP registration

Objects	Wave	Angel	Potatohead
Num points (\mathcal{S}_0)	22500	14089	13827
Num triangles (\mathcal{S}_0)	44402	27213	27144
Ranging sensor	Synthetic	Laser Ranging	Stereo camera
Noise	10 %	little	Very noisy
Distortion	No	No (Negligible)	Yes
No. of views	2	2	8

8.4.2 Registration Error with respect to Iteration of Projections

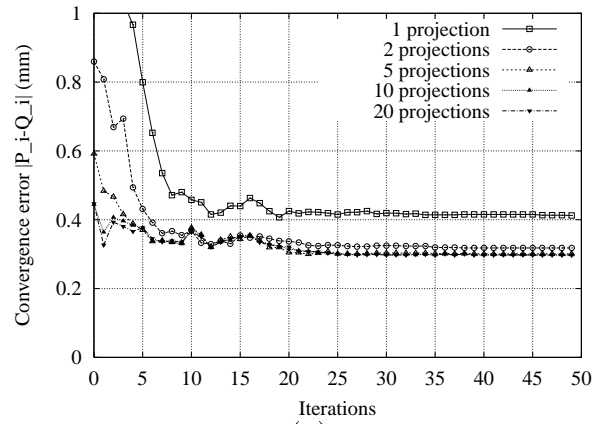
We apply our CPP registration technique to the *Wave* object to plot convergence error ϵ_c with respect to different numbers of normal projections. This number is the count of projections from $\mathbf{Q}_{\mathbf{p}_i}$ to $\mathbf{Q}_{\mathbf{i}+1}$ – to the line of point normal $\hat{\mathbf{p}}_0$ – in Equation (8.5). It can be expected that the more projections result in the smaller convergence error, because the error converges closer to zero with more projections. In Figure 8.6(a), convergence error is plotted for different numbers of projections. We run the CPP for 50 iterations. At each iteration, we plot the RMS convergence error of all control points, where we use about 300 pairs. As shown in the figure, the convergence error decreases when the number of projections increases. However, the figures shows that the error does not decrease any more when the projection number is greater than 5. This means that, in real situations, more projection numbers do not always produce better results.

RMS error of translation between control point sets are plotted in Figure 8.6(b) and (c). In Figure 8.6(b), we use the *Wave* object without adding random noise. As shown in the figure, even with small number of projections, two partial shapes are registered very fast and accurately. In Figure 8.6(c), the translation error is plotted for noisy *Wave* object. In this figure, we find that only small number of projections is enough to register the surfaces accurately. For the rest of experiments, we use the CPP algorithm with 5 projections.

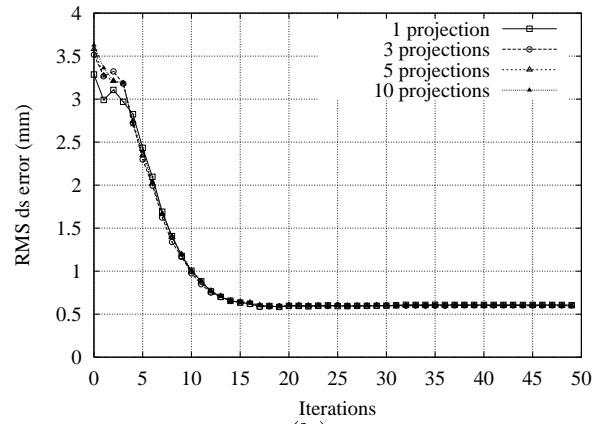
8.4.3 Pair-wise Registration

Test Environment

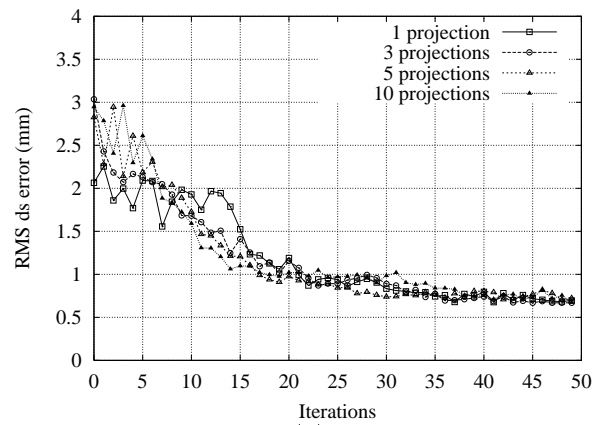
We test our CPP algorithm for a pair of range image in each test object. We also test the *Point-to-Projection* and the *Point-to-Point* (ICP with kd-tree searching) approaches to the same objects to compare their results with ours. We test all three



(a)



(b)



(c)

Figure 8.6: Comparison of convergence and translation errors with respect to different number of projections. (*Wave* object) (a) Convergence error (b) RMS translation error without noise (c) RMS translation error with 10% noise

approaches in the same registration condition using a Pentium III 1GHz personal computer. In order to pick control points on both source and destination surfaces, we

- uniformly sample a source surface to pick about 200 source control points,
- discard a source control point when the vector product of its normal and the viewing vector of the source is greater than (-0.3) – the point normal is opposite to the viewing vector,
- discard a destination control point when the magnitude of the vector product of its normal and the viewing vector is greater than (-0.3),
- and discard a destination control point when the distance to its conjugate point is greater than 10.0 mm.

Error Metric

Registration error of the test objects is measured by the distance error of d_s shown in Figure 8.1. RMS (Root Mean Squares) error of all control points is plotted at every iteration. However, registration error in each approach can not be directly compared. For example, the error metric of *point-to-point* approach is different with that of *point-to-plane*. In addition, because we use range images of real objects, we don't know true matching pairs of the control points. In order to compare registration error between all three approaches, we normalize the error using the initial error of each approach. Because the initial condition of all approaches is the same, we normalize the RMS errors to watch the decrease rate of them. We divide RMS error at each iteration by the initial one.

Results

Results of registration refinement for the *Wave* object are shown in Figure 8.7. We register the second view's surface to the first view for 50 iterations. The RMS error of d_s , distance between control points, for the *Wave* object is plotted in Figure 8.7(a). *Point-to-projection* approach shows the worst result. The proposed registration technique shows better result than others.

Figure 8.7(b) shows the results for the object *Angel*. *Point-to-projection* result also shows a bad registration result because the initial condition (20 degree rotation) is too much for this approach to register the surfaces. The proposed technique also shows better result than other techniques. Figure 8.7(c) is the result for *Potatohead*. As mentioned in an earlier section, the range images are obtained from a stereo camera. This result only shows the registration of a pair of images. We use the first

and the second views of object out of 8 multiple views. In this figure, we also plot a result of original point-to-plane technique, which use a brute force searching to find matching control points. The result of the proposed technique is almost the same with that of the original technique.

Table 8.2 shows the results of registration error and processing time for 50 iterations. As shown in the previous figure, our CPP technique has the smallest registration error after refinement. As far as computation time is concerned, the point-to-projection technique is faster than other approaches. The proposed technique and the point-to-point technique show similar results but the results depend on types of objects. This is because we have some computations for finding the matching control point as mentioned in the previous section.

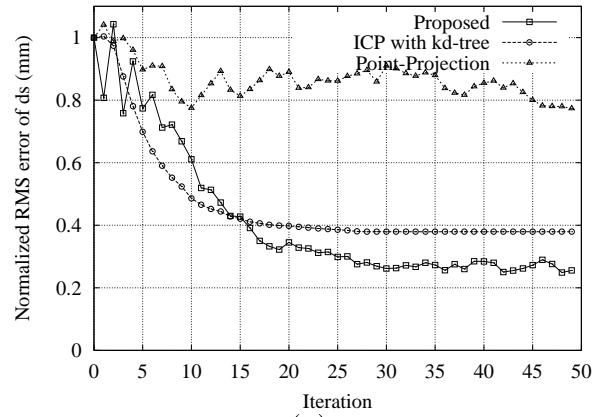
Table 8.2: Registration error and processing time after 50 iterations ($N_c = 5, D_c = 0.1mm$)

Method		Point- Projection	ICP with kd-tree	Proposed
Wave	Normalized d_s (mm)	0.77	0.38	0.26
	Time/Itr (ms)	38.4	40.6	38.4
Angel	Normalized d_s (mm)	0.73	0.34	0.27
	Time/Itr (ms)	28.4	35.0	32.8
Potatohead	Normalized d_s (mm)	0.63	0.63	0.47
	Time/Itr (ms)	24.2	27.6	28.4

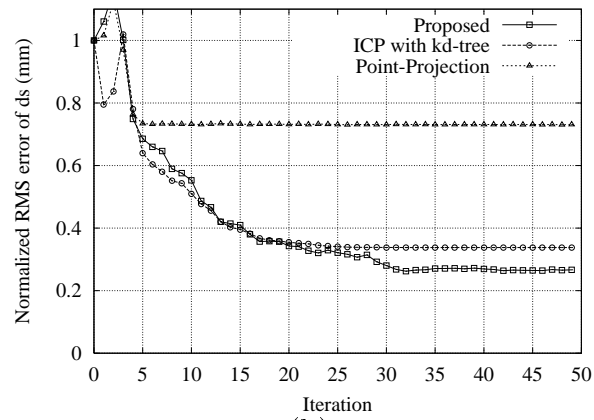
8.4.4 Multi-view Registration

Multi-view registration is a more difficult problem than a pair-wise registration, because registration error should be evenly distributed on all overlapping surfaces. The multi-view registration problem is also an important research topic but we use Bergevin’s technique [7]. We set the first view of the *Potatohead* object as a reference view and register all the other views to the reference view. In order to find the matching control points from a source view, we search on its neighboring view’s surfaces. For example, if the i th view is concerned, we search the matching control points on its neighborhoods, $(i+1)$ th view and $(i-1)$ th view. After finding all matching points for every view point, except the first view, we compute rigid transformations and refine all surfaces simultaneously.

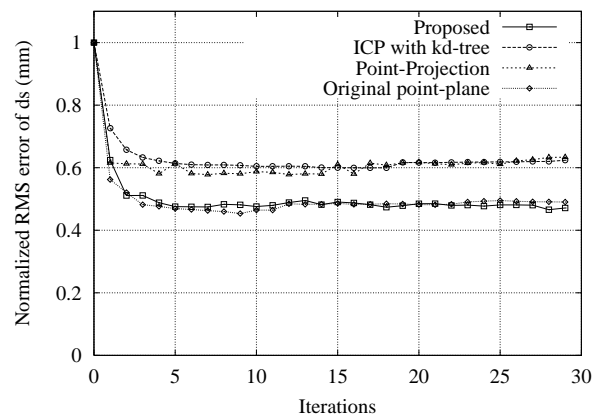
We test a multi-view registration for the 8 views of the *Potatohead* object. The Figure 8.8 plots their results. In this figure we do not normalize the errors, because their results are very clear for comparison. Both *point-to-projection* and ICP



(a)



(b)



(c)

Figure 8.7: RMS error comparison of different registration techniques (a) *Wave* with noise (b) *Angel* (c) *Potatohead*

techniques show bad registration results. Both of them diverge or do not properly register the multi-view images. In contrast, our CPP technique shows better results. The table 8.3 shows registration error and processing time for this experiment. In this multi-registration, ICP (with kd-tree) technique takes more time, because it needs to refine kd-trees of all surfaces (except the reference surface) for every iteration. As a result, this table shows that our CPP technique has advantages over ICP and *Point-to-projection* techniques for registration of multiple range images.

Table 8.3: Multi-view registration error and processing time after 50 iterations. ($N_c = 5, D_c = 0.1 \text{ mm}$)

Method	Point-Projection	ICP with kd-tree	Proposed
Average d_s (mm)	3.45	1.88	0.74
Time (sec)	10.05	47.29	16.04
Time/Itr (sec)	0.201	0.946	0.321

8.5 Conclusions

We address a registration refinement problem and present an accurate and fast *Point-to-(Tangent) Plane* registration technique. In order to find an intersection point on a destination surface, we project a source control point to the destination surface, re-project the projection point to the normal vector of the source point. We show that iterative projections of the projected destination point to the normal vector converge to the intersection point. By assuming the destination surface to be a monotonic function in a new 2D coordinate system, we show contraction mapping properties of our *Contractive Projection Point* technique. In an alternative technique, finding convergence point could be implemented by another method such as the Newton-Raphson method. Instead of projecting the destination control point to the vector normal, the tangent vector at the destination point can be used to find the intersection with the vector normal. Experimental results show that our approach is very accurate and fast for both pair-wise registration and multi-view registration problems.

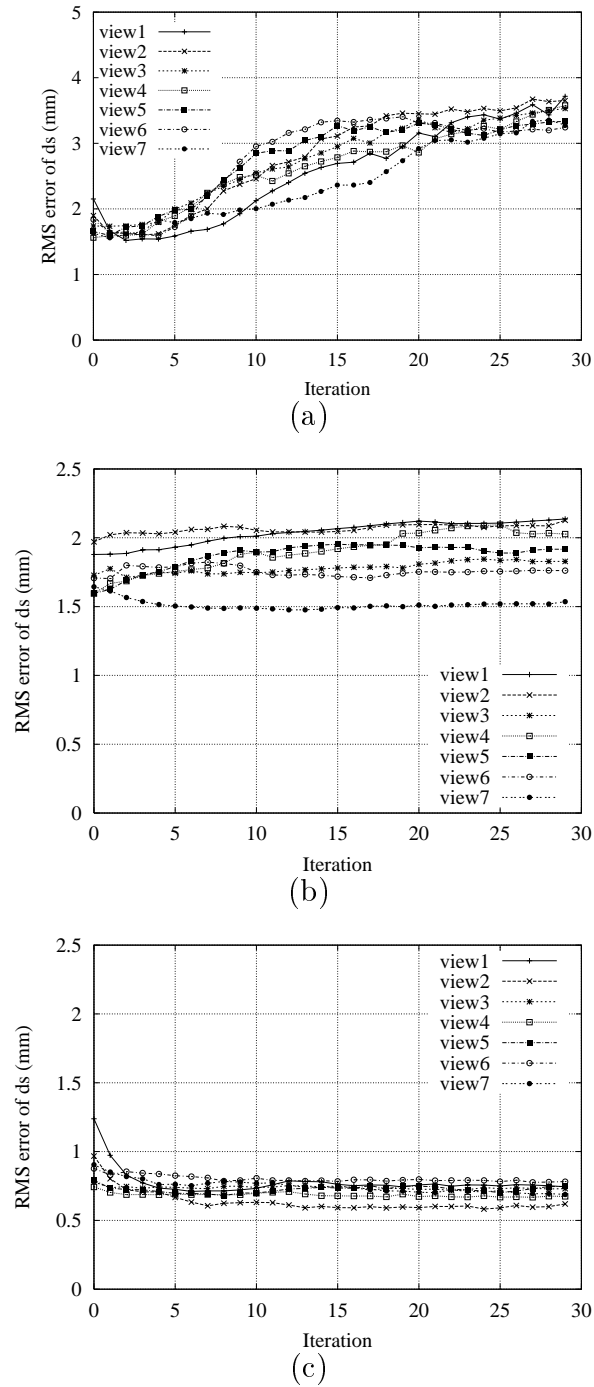


Figure 8.8: Results of multi-view registration for 'Potatohead' object after 50 iterations. (a) Point-to-projection (b) ICP with kd-tree (c) Proposed (CPP)

Chapter 9

Conclusions

9.1 Summary

In this dissertation, we have presented stereo vision systems and computer vision techniques for complete and photo-realistic 3D model generation. A complete 3D model of a real object is reconstructed by merging multiple range images and two pose models of the object. Multiple range images of the object are obtained by either SVIS-2 or SVIS-3 stereo vision system. The SVIS-2 vision system consists of a parallel stereo camera and a motion control stage to acquire range images and to change the view of the object. The SVIS-3 system consists of a vergence stereo camera and a motion stage. The two vision systems are calibrated to facilitate registration of multi-view range images into a common coordinate system. Multiple pairs of stereo images are acquired from different views of the object and rectified according to the calibration results of the camera systems. Corresponding range images are then obtained by a stereo vision technique.

In Chapter 4, we presented new computer vision techniques for registration and integration of multiple range images. The techniques exploit the epipolar geometry between different view points. Correspondence between 3D points is established by projecting overlapping contour segments from one view to another view and finding the closest point on the epipolar line. A surface-based integration technique merges the contours slice by slice using linked lists which represent points and segments of the contours of an object's cross-sections. The linked lists for different views are merged based on two closeness criteria to obtain closed contours representing complete cross-sections of the object.

Another multi-view integration technique is presented in Chapter 5. Integration of range images is accomplished by estimating the iso-surface of an object in a volumetric 3D space. In order to remove erroneous points outside the visual hull of the object, shape-from-silhouettes technique is used. A grid of 3D voxels is classified into several sub-regions according to the signed-distances of a voxel to adjacent range

images. The iso-surface of the object is reconstructed by applying a novel averaging technique based on the class of a voxel. Then the iso-surface representation of the object is converted into a 3D mesh model by applying the Marching Cubes algorithm.

In order to avoid hidden surfaces that arise when only one pose of an object is used, we introduce a novel technique of integrating two 3D models which are reconstructed in two different poses of the object. Two 3D models are reconstructed by the proposed multi-view modeling technique by placing the object on a planar surface in two different poses. Pose between two 3D models is estimated by matching stable tangent plane (STB) with base tangent plane (BTP). Because the object always stands on one of its stable tangent planes, we find a STP from the first model, which matches the BTP of the second plane and *vice versa* to estimate the pose.

A pose integration technique is presented in Chapter 7. We introduce a novel technique to integrate two partial 3D models of an object's two different poses. The voxel classification technique employed in multi-view integration in Chapter 5 is also employed to reconstruct a closed 3D model. Pose integration algorithm combines the signed distance and the class of a voxel from each pose. Photorealistic texture mapping of 3D models of several real objects are presented. We blend textures on mesh triangles whose vertices project onto different view points according to the best-view criterion. Texture blending reduces blocking effect of brightness difference between different view points.

In the last chapter, we addressed a registration refinement problem and presented an accurate and fast *Point-to-(Tangent) Plane* registration technique. In order to find control point sets fast and accurately, we project a source control point to the destination surface, re-project the projection point to the normal vector of the source point. We show that iterative projections of the projected destination point onto the normal vector converge to an intersection control point. By assuming the destination surface to be a monotonic function in a new 2D coordinate system, we show contraction mapping properties of our *Contractive Projection Point* technique.

9.2 Future Work

This research can be extended and improved further. Some possible topics are listed below as future research work.

Range image acquisition

In this dissertation, a complete 3D model of an object is reconstructed by merging multiple range images. Therefore, the accuracy of our 3D reconstruction directly depends on the accuracy of the range images. We have employed stereo camera

systems to obtain multiple range images. This means that there could be some erroneous points on range images due to inherent stereo matching problems. We can improve the performance of our 3D reconstruction by adapting new stereo matching techniques, such as multi-baseline stereo or rotational stereo matching technique.

Image-based registration

Registration of multiple range images is a very important step of our 3D reconstruction. Unless range images of an object are accurately registered to a common coordinate system, a 3D model merged from the range images may experience geometric distortion or photographic texture blocking on its surfaces. One approach that can overcome this problem is employing an image-based registration technique. By matching features on texture images between different views of an object, the range images can be aligned so that their photometric structures between the different views are consistent.

Real-time 3D reconstruction

Real-time 3D reconstruction of real-world objects is another extension of this research. The computational complexity of reconstructing a 3D model of a real object is heavy. Using our techniques, for example, we can reconstruct a 3D model of a simple object in a few of minutes. However, because the computation cost is becoming cheaper, we may reconstruct it in a few seconds using high-speed techniques or graphic accelerators. The major problems for real-time reconstruction are real-time range image acquisition, real-time registration of multiple range images, and real-time integration and texture mapping. We have already shown in Chapter 8 that the proposed point-to-plane registration technique is very fast and accurate for multi-view modeling. If we have a fast range image acquisition technique or system, we may reconstruct the 3D model in a very short time.

Unstructured 3D reconstruction

This dissertation presents calibration techniques of stereo vision systems to facilitate registration of multiple range images. However, in case that a vision system cannot be calibrated, we need another technique to register them without a priori knowledge of calibration parameters. This is a pose estimation problem between multi-view range images. We may suggest two approaches to solve the problem. One is employing an image-based feature matching technique that could find feature correspondences between different views. The other approach is employing a geometry-based pose estimation technique. When a range sensor changes its viewing

direction to an object, the object's pose, from the point of the sensor's view, is also changed. If we estimate the object's pose between different views, using geometric features and constraints, we could solve the rigid transformation to register 3D model of the object.

Bibliography

- [1] J. K. Aggarwal and Y. C. Kim, "Rectangular Parallelepiped Coding: A Volumetric Representation of Three Dimensional Objects," *IEEE Journal of Robotics and Automation*, Vol. 2, No. 3, pp.127-134, Sep. 1986.
- [2] P. Allen and R. Yang, "Registering, Integrating, and Building CAD Models from Range Data," *IEEE International Conference on Robotics and Automation*, pp. 3115-3120, 1998.
- [3] N. Ahuja , J. E. Veenstra, "Generating Octrees from Object Silhouettes in Orthographic Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 2, pp. 137-149, Feb. 1989.
- [4] K.S. Arun, T.S. Huang, and S.D. Blostein, "Least-Squares Fitting of Two 3D Point Sets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9, No. 5, Sep. 1987.
- [5] G. Barequet and M. Sharir, "Partial Surface and Volume Matching in Three Dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 19, No. 9, Sep. 1997.
- [6] R. Benjemaa and F. Schmitt, "Fast Global Registration of 3D Sampled Surface using a Multi-Z-buffer Technique," *Image and Vision Computing*, Vol. 17, pp. 113-123, 1999.
- [7] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau, "Toward a General Multi-view Registration Technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 5, 1996.
- [8] F. Bernadini, I. M. Martin, and H. Rushmeier, "High-quality Texture Reconstruction from Multiple Scans," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 7, No. 4, pp. 318-332, 2001.
- [9] P. Besl and H. McKay, "A Method for Registration of 3D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, pp. 239-256, 1992.

- [10] G. Blais and M. Levine, "Registering Multiview Range Data to Create 3D Computer Object," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No.8, 1995.
- [11] J. Bloomenthal, *An Implicit Surface Polygonizer*, Graphics Gems IV, pp.324-349. Academic Press, 1994.
- [12] J. S. Bonet and P. Viola, "Roxels: Responsibility Weighted 3D Volume Reconstruction", *Proceedings of International Conference on Computer Vision*, pp. 418-425, 1999.
- [13] P.J. Burt, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, Vol. 31, No. 4, pp. 532-540, 1983.
- [14] R. J. Campbell and P. J. Flynn, "A Survey of Free-form Object Representation and Recognition Techniques," *Computer Vision and Image Understanding*, vol. 81, pp. 166-210, 2001.
- [15] H. H. Chen and T. S. Huang, "A Survey of Construction and Manipulation of Octrees," *Computer Vision, Graphics, and Image Processing*, Vol. 43, pp. 409-431, 1988.
- [16] Q. Chen and G. Medioni, "A Volumetric Stereo Matching Method: Applications to Image-based Modeling," *Proceedings of Computer Vision and Pattern Recognition*, 1999.
- [17] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," *Image and Vision Computing*, vol. 10, No. 3, pp.145-155, 1992.
- [18] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," *Proceedings of SIGGRAPH*, pp. 303-312, 1996.
- [19] B. Curless and M. Levoy, "Better Optical Triangulation through Spacetime Analysis," *IEEE International Conference on Computer Vision*, pp. 987-994, June 1995.
- [20] C. M. Cyr, T. Sebastian, and B. Kimia, "2D-3D Registration Based on Shape Matching," *IEEE Workshop on Mathematical Methods in Biomedical Image*, pp.198-203, 2000.
- [21] U.R. Dhond and J.K. Aggarwal, "Structure from Stereo-A Review", *IEEE Transactions on System, Man, and Cybernetics*, Vol. 19, No. 6, Nov. 1989.

- [22] C. Dorai, J. Weng, and A. Jain, "Optimal Registration of Object Views Using Range Data," *IEEE Transactions Pattern Analysis and Machine Intelligence*, Vol. 19, No. 10, pp. 1131-1138, Oct. 1997.
- [23] C. Dyer, "Volumetric Scene Reconstruction from Multiple Views," *Foundations of Image Understanding*, L. S. Davis, ed., Kluwer, Boston, pp. 469-489, 2001.
- [24] P. Eisert, E. Steinbach, and B. Girod, "Automatic Reconstruction of Stationary 3-D Objects from Multiple Uncalibrated Camera Views," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 2, pp. 261-277, Mar. 2000.
- [25] O. Faugeras and R. Keriven, "Complete Dense Stereovision using Level Set Methods," *Proceedings of the 5th European Conference on Computer Vision*, Vol. 1406 of Lecture Notes on Computer Science, pp. 379-393. 1998.
- [26] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D Model Construction for Turn-table Sequences", *European Workshop SMILE'98, Lecture Notes in Computer Science 1506*, pp. 155-170, 1998.
- [27] F.P. Ferrie and M.D. Levine, "Integrating Information from Multiple Views," *Proceedings of the IEEE Workshop on Computer Vision*, pp. 117-122, 1987.
- [28] A. Fusiello, E. Trucco, and A. Verri, "A Compact Algorithm for Rectification of Stereo Pairs," *Machine Vision and Applications*, Vol. 12, pp. 16-22, 2000.
- [29] A. Fusiello, V. Roberto, and E. Trucco, "Efficient Stereo with Multiple Windowing," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 858-863, 1997.
- [30] H. Gagnon, M. Soucy, R. Bergevin, and D. Laurendeau, "Registration of Multiple Range Views for Automatic 3D Model Building," *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, pp. 581-586, June 1994.
- [31] F. Gerald, *Curves and Surfaces for Computer aided Geometric Design*, Academic Press, 1993.
- [32] J. Gluckman, S Nayar, "Rectifying Transformations that Minimize Resampling Effects," *Proceedings of International Conference on Computer Vision and Pattern Recognition*, Vol.1, pp. 111-117, 2001.
- [33] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, 1992.

- [34] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt, "Reliable Surface Reconstruction from Multiple Range Images," *Proceedings of the European conference on Computer Vision*, Springer-Verlag, pp. 117-126, 1996.
- [35] H. Hoppe, T. DeRose, and T. Duchamp, "Surface Reconstruction from Unorganized Points," *Computer Graphics*, Vol. 26, No. 2, July 1992.
- [36] D. F. Huber, "Automatic 3D Modeling Using Range Images Obtained from Unknown Viewpoints," *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, IEEE Computer Society, pp. 153-160, 2001.
- [37] D. Huber and M. Hebert, "Fully Automatic Registration of Multiple 3D Data Sets," *IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum(CVBVS 2001)*, Dec. 2001.
- [38] K. Ikeuchi, "Recognition of 3D Objects using the Extended Gaussian Image," *7th International Conference on Artificial Intelligence*, pp. 595-600, 1995.
- [39] K. Ikeuchi and M. Hebert, "Spherical Representations: from EGI to SAI," *Technical report CMU-CS-95-197*, 1995.
- [40] Y. Iwakiri and T. Kaneko, "PC-based Real-time Texture Painting on Real World Objects," *Computer Graphics Forum*, Vol. 20, No. 3, 2001.
- [41] Jarvis, R.A., "A Perspective on Range Finding Techniques for Computer Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, No.2, pp.122-139, 1983.
- [42] A. Johnson and H. Hebert, "Surface Registration by Matching Oriented Points," *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 121-128, May, 1997.
- [43] A. Johnson and S. Kang, "Registration and Integration of Textured 3-D Data," *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 234-241, May, 1997.
- [44] A. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 5, pp. 433 - 449, May, 1999.
- [45] S. B. Kang and K. Ikeuchi, "3D Object Pose Determination using Complex EGI," *Technical report CMU-RI-TR-90-18*, 1990.

- [46] A. Koschan and V. Rodehorst, "Dense Depth Map by Active Color Illumination and Image Pyramids," *Advances in Computer Vision*, Springer, pp.137-148, 1997.
- [47] D. J. Kriegman, "Computing Stable Poses of Piecewise Smooth Objects," *Image Understanding*, Vol. 55, No. 2, pp. 109-118, 1992.
- [48] Krotkov, E., "Focusing", *International Journal of Computer Vision*, Vol. 1, pp. 223-237, 1987.
- [49] K. Kutulakos and S. Seitz, "A Theory of Shape by Space Carving," *Technical Report TR692*, Computer Science Dept., U. Rochester, 1998.
- [50] P. Lander, *A Multi-camera Method for 3D Digitization of Dynamic, Real-world Events*, Doctoral Dissertation, Carnegie Mellon University, 1998.
- [51] C. Zitnick and T. Kanade, "A Volumetric Iterative Approach to Stereo Matching and Occlusion Detection," *Technical Report CMU-RI-TR-98-30*, Robotics Institute, Carnegie Mellon University, Dec. 1998.
- [52] H. Lensch, W. Heidrich, and H. Seidel, "Automated Texture Registration and Stitching for Real World Models," *Proceedings of Pacific Graphics 2000*, IEEE Computer Society Press, pp. 317-327, 2000.
- [53] M. Levoy et al, "The Digital Michelangelo Project: 3D Scanning of Large Statues," *Proceedings of SIGGRAPH*, 2000.
- [54] H.Y. Lin and M. Subbarao, "Three-Dimensional Model Acquisition using Rotational Stereo and Image Focus Analysis," *Proceedings of SPIE*, Oct. 2000.
- [55] J. Little, "Determining Object Attitude from Extended Gaussian Images," *9th International Conference on Artificial Intelligence*, pp. 960-963, 1985.
- [56] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol. 21, No. 4, 1987.
- [57] A. Lorusso, D. Eggert, and R. Fisher, "A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations," *Proceedings of 6th British Machine Vision Conference*, pp. 237-246, 1995.
- [58] K. Matsushita and T. Kaneko, "Efficient and Handy Texture Mapping on 3D Surfaces," *Computer Graphics Forum*, Vol. 18, No. 3, Sep. 1999.
- [59] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based Visual Hulls," *Proceedings of SIGGRAPH*, pp. 369-374, 2000.

- [60] B. Mirtich, "Fast and Accurate Computation of Polyhedral Mass Properties," *Journal of Graphics Tools*, Vol. 1, No. 2, pp. 31-50, 1996.
- [61] S. Moezzi, L. Tai, and P. Gerard, "Virtual View Generation for 3D Digital Video," *IEEE Multimedia*, pp. 18-26, 1997.
- [62] P. Narayanan and T. Kanade, "Virtual Worlds using Computer Vision," *ATR Workshop on Computer Vision for Virtual Reality Based Human Communications*, pp. 2-13, Jan. 1998.
- [63] A. Naylor and G. Sell, *Linear Operator Theory in Engineering and Science*, Springer-Verlag, New York, 1982.
- [64] P. Neugebauer, "Geometrical Cloning 3D Objects via Simultaneous Registration of Multiview Range Images," *IEEE Conference on Shape Modeling and Applications*, pp. 130-139, 1997.
- [65] W. Niem, "Automatic Reconstruction of 3D Objects using a Mobile Camera," *Image and Vision Computing*, Vol. 17, pp. 125-134, 1999.
- [66] W. Niem, R. Buschmann, "Automatic Modeling of 3D Natural Objects from Multiple Views," *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, pp. 23.-24. Nov. 1994.
- [67] K. Nishino and K. Ikeuchi, "Robust Simultaneous Registration of Multiple Range Images," *The 5th Asian Conference on Computer Vision (ACCV2002)*, pp.23-25, 2002.
- [68] M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 4, Apr. 1993.
- [69] S. Park and M. Subbarao, "A New Technique for Registration and Integration of Partial 3D Models," *Proceedings of SPIE*, Vol. 4567, Oct. 2001.
- [70] S. Park and M. Subbarao, "Automatic 3D Model Reconstruction using Voxel Coding and Pose Integration," *Proceedings of International Conference on Image Processing*, pp. 533-536, Sep. 2002.
- [71] S. Park and M. Subbarao, "Pose Estimation of Two-pose 3D models using the Base Tangent Plane and Stability Constraints," *Proceedings of Vision, Modeling, and Visualization*, 2002.
- [72] S. Park and M. Subbarao, "Pose Estimation and Integration for Complete 3D Model Reconstruction," *IEEE Workshop on Application of Computer Vision (WACV2002)*, pp.143-147, Dec. 2002.

- [73] R. Pito, "Mesh Integration Based on Co-measurements," *Proceedings of International Conference on Image Processing*, pp. 397-400, 1996.
- [74] P.C. Pritchett and A.P. Zisserman, "Wide Baseline Stereo Matching," *Proceedings of 6th International Conference on Computer Vision*, Bombay, January 1998.
- [75] K. Pulli, H. Abi-Rached, T. Duchamp, L. Shapiro, and W. Stuetzle, "Acquisition and Visualization of Colored 3D Objects," *IEEE International Conference on Pattern Recognition*, pp. 11-15, Aug. 1998.
- [76] K. Pulli, "Multiview Registration for Large Data Sets," *Proceedings of 2nd International Conference on 3D Digital Imaging and Modeling*, pp. 160-168, 1999.
- [77] K. Pulli et al., "Robust Meshes from Range Maps," *Proceedings of International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 205-211, May 1997.
- [78] K. Pulli et al., "View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data," *Proceedings of 8th Eurographics Workshop on Rendering*, June 1997.
- [79] P.W. Rander, P.J. Narayanan, and T. Kanade, "Recovery of Dynamic Scene Structure from Multiple Image Sequences," *Proceedings of the 1996 International Conference on Multisensor Fusion and Integration for Intelligence Systems*, pp.305-312, Dec. 1996.
- [80] C. Rocchini, P. Cignoni, and C. Montani, "Multiple Textures Stitching and Blending on 3D Objects," *Eurographics Rendering Workshop '99*, June 1999.
- [81] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," *Proceedings of 3D Digital Imaging and Modeling*, pp. 145-152, 2001.
- [82] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-Time 3D Model Acquisition," *Proceedings of SIGGRAPH 2002*.
- [83] M. Rutishauser, M. Stricker, and M. Trobina, "Merging Range Images of Arbitrarily Shaped Objects," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 348-353, 1992.
- [84] H. Saito and T. Kanadem "Shape Reconstruction in Projective Grid Space from Large Number of Images," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June, 1999.

- [85] A. Sarti and S. Tubaro, "A Multiresolution Level-Set Approach to Surface Fusion," *Proceedings of International Conference on Image Processing (ICIP 2001)*, Vol. 2, pp. 181-184, 2001.
- [86] S. M. Seitz and C. R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1067-1073, 1997.
- [87] I. Söderkvist, "Introductory Overview of Surface Reconstruction Methods," *Lulea University of Technology, Department of Mathematics, Research Report 10*, 1999.
- [88] M. Soucy and D. Laurendeau, "Multiresolution Surface Modeling from Multiple Range Views," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 348-353, 1992.
- [89] J. Starck, A. Hilton, and J. Illingworth, "Human Shape Estimation in a Multi-Camera Studio," *12th British Machine Vision Conference (BMVC)*, Vol. 2, pp. 573-582, Manchester, Sep. 2001.
- [90] M. Subbarao and J. K. Tyan, "Selecting the Optimal Focus Measure for Autofocusing and Depth-from-Focus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 8, pp. 864-870, Aug. 1998.
- [91] M. Subbarao, "Parallel Depth Recovery by Changing Camera Parameters", *Second International Conference on Computer Vision*, Florida, USA, pp.149-155, Dec. 1988.
- [92] M. Subbarao and T. Choi, "Focusing Techniques", *Proceedings of SPIE*, Boston, Massachusetts, Vol 1823, pp. 163-174, Nov. 1992.
- [93] M. Subbarao and G. Surya, "Application of Spatial-Domain Convolution/Deconvolution Transform for Determining Distance from Image Defocus", *Proceedings SPIE*, Boston, Massachusetts, Vol 1822, pp. 159-167, Nov. 1992.
- [94] C. Sun, "A Fast Stereo Matching Method," *Digital Image Computing: Techniques and Applications*, pp. 95-100, Massey University, Auckland, New Zealand, Dec. 1997.
- [95] Y. Sun and M. Abidi, "Surface Matching by 3D Point's Fingerprint," *Proceedings of IEEE International Conference on Computer Vision*, pp.263-269, 2001.

- [96] S. Tosovic, R. Sablatnig, "3D Modeling of Archaeological Vessels using Shape from Silhouette," *Proceedings of 3rd IEEE International Conference on 3-D Digital Imaging and Modeling*, pp. 51-58, 2001.
- [97] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer Vision*, New Jersey, Prentice-Hall, 1998.
- [98] R. Y. Tsai, "A Versatile Camera Calibration technique for High-accuracy 3D Machine Vision Metrology using Off-the-shelf TV Camera and Lenses," *IEEE Journal of Robotics and Automation*, Vol. 3, No. 4, pp. 323-344, 1987.
- [99] G. Turk and M. Levoy, "Zippered Polygon Meshes from Range Images," *Computer Graphics Proceedings in SIGGRAPH'94*, pp. 311-318, 1994.
- [100] S. Vedula, P. Rander, H. Saito, and T. Kanade. "Modeling, Combining, and Rendering Dynamic Real-World Events From Image Sequences," *Proceedings 4th Conference Virtual Systems and Multimedia*, Vol.1, pp.326-332, 1998.
- [101] S. Weik, "Registration of 3D Partial Surface Models Using Luminance and Depth Information", *In Proceedings International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 93-100, May 1997.
- [102] J. Weng, N. Ahuja, and T. S. Huang, "Matching Two Perspective Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, pp. 806-825, Aug.1992
- [103] M. D. Wheeler, Y. Sato, and K. Ikenuchi, "Consensus Surfaces for Modeling 3D Objects from Multiple Range Images", *IEEE Conference on Computer Vision*, pp. 917-924, 1998.
- [104] S. Winkler, P. Wunsch, and G. Hirzinger, "A Feature Map Approach to Real-time 3D Object Pose Estimation from Single 2-D Perspective Views," *Proceedings of 19th DAGM Symposium*, pp. 129-136, Sep. 1997.
- [105] K. Wong and R. Cipolla, "Structure and Motion from Silhouettes," *Proceedings 8th IEEE International Conference on Computer Vision (ICCV01)*, Vol. 2, Vancouver, Canada, pp. 217-222, 2001.
- [106] T. Yuan and M. Subbarao, "Integration of Multiple-baseline Color Stereo Vision with Focus and Defocus Analysis for 3D Shape Measurement," *Proceedings of SPIE*, Vol. 3520, pp.44-51, Nov. 1998.

- [107] A. Zisserman, F. Schaffalitzky, T. Werner, and A. Fitzgibbon, "Automatic Reconstruction from Multiple Photographs," *IEEE Conference on Image Processing (ICIP2002)*, 2002.
- [108] D. Zhang and M. Hebert, "Harmonic Maps and Their Applications in Surface Matching," *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, 1999.
- [109] L. Zhang, B. Curless, and S. Seitz, "Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming," *1st International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 24-36, June, 2002.