Automatic Three-dimensional Modeling from Reality

Daniel F. Huber

CMU-RI-TR-02-35

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

July 23, 2002

Copyright © 2002 by Daniel F. Huber. All rights reserved.

Abstract

In this dissertation, we develop techniques to fully automate the process of constructing a digital, three-dimensional (3D) model from a set of 3D views of a static scene obtained from unknown viewpoints. Since 3D sensors only capture the scene structure as seen from a single viewpoint, multiple views must be combined to create a complete model. Current modeling methods are limited in that they require measurement of the sensor viewpoints or manual registration of the views. We have developed an alternative approach that fully automates the modeling process without without any knowledge the sensor viewpoints and without manual intervention.

Given a set of 3D views of a scene, the challenge is to align all the views in a common coordinate system without knowing the original sensor viewpoints or even which views overlap one another. This problem, which we call multi-view surface matching, is analogous to assembling a jigsaw puzzle in 3D. The views are the puzzle pieces, and the problem is to correctly assemble the pieces without even knowing what the puzzle is supposed to look like. To solve the multi-view surface matching problem, we begin by using pair-wise surface matching to align pairs of views. The resulting relative pose estimates (matches), some of which may be incorrect, are stored as edges in a graph data structure called the local registration model graph (G_{LR}). Unfortunately, it is generally not possible to distinguish all the correct matches from incorrect ones at the local (pair-wise) level; however, by considering the surface consistency constraints of a sequence of matches, we can identify incorrect, but locally consistent matches.

G_{LR} contains a node for each view and an edge for each match from pair-wise surface matching. The sub-graphs of G_{LR} represent the set of discrete model hypotheses – potential solutions to the multi-view surface matching problem. If we can find a model hypothesis that connects all the views and contains only correct matches, we can use the relative poses to place the views in a common coordinate system, thereby correctly solving this multi-view surface matching instance. This discrete search space also contains a continuous aspect, since the view poses we are estimating are continuous-valued. We therefore formulate this search as a mixed discrete/continuous optimization over the discrete hypothesis space and continuous pose parameters.

With this formulation, we have reduced the multi-view surface matching problem to two fundamental questions: 1) What constitutes a "good" model hypothesis? and 2) How do we robustly and efficiently search for the best model hypothesis in an exponentially large hypothesis space? We answer the first question with a method for learning a probabilistic model of the local quality of pair-wise matches and the global quality of model hypotheses. The method provides a principled way to combine multiple, disparate measures of registration quality. We answer the second question with two classes of search algorithms. One class searches deterministically for a globally consistent model hypothesis by repeatedly adding edges from G_{LR} to an initially empty hypothesis. The second class uses a stochastic edge swapping approach to search for the best hypothesis according to the given global quality measure.

Our multi-view surface matching framework is sensor independent and can be applied to 3D data at any scale. We show results using three different range scanners to automatically model scenes varying in size from from 20 centimeters to 200 meters. We model a variety of scene types, including small, "desktop" objects, building interiors, and terrain. For small objects, we have developed an application called hand-held modeling, in which the user holds an object and scans it from various viewpoints. This is an easy modeling method, requiring no specialized hardware, minimal training, and only a few minutes to model an average object.

Acknowledgments

I dedicate this dissertation to my wife, Celia. Her love, encouragement, and, most importantly, her understanding have made it possible for me to continue my education through eight of the ten years of our marriage.

I am grateful to my advisor, Martial Hébert, who, in addition to being invaluable in guiding me through my graduate career, is officially recognized as the coolest person in the Robotics Institute. His mastery of all topics of computer vision is an inspiration to us all. Thanks also to my committee (Katsushi Ikeuchi, Yanxi Liu, and Steve Seitz), each of whom helped shape my research in their own way. This work has benefitted immensely from their questions, insights, and suggestions. This work has also benefitted immensely from the support of Larry Ray and the Eastman Kodak Company, without which the research would not have been possible. The Kodak Fellows Program, which funded my work, is a tremendous service to the research community.

My fellow researchers at the Robotics Institute also deserve recognition. In particular, the longstanding members of the MISC reading group (Matt Deans, Frank Dellaert, Owen Carmichael, Sanjiv Kumar, and Shyjan Mahamud) have taught me more about computer vision than I could have ever learned on my own. I sincerely appreciate their helpful comments on my work and their tolerance of my never-ending questions. I am indebted to the Robotics faculty members who have kept their doors open for me: Takeo Kanade, Illah Nourbakhsh, Jianbo Shi, Henry Schneiderman, Andrew Moore, Jonas August, Geoff Gordon, and Paul Heckbert. Thanks also to the students and staff whose friendship, advice, and/or assistance with data collection made my years as a graduate student a lot of fun: Dave LaRose, Parag Batavia, Sundar Vedula, Nicolas Vandapel, Stewart Moorehead, Ryan Miller, Daniel Morris, Suzanne Crow, Tim Doebler, and Marce Zaragoza.

Finally, I would like to thank my colleagues who have made their software available to me. Andrew Johnson (spin-image surface matching), Brian Curless (VRIP surface reconstruction), Michael Garland (quadric mesh simplification), and Andrew Willmott (VL math library). Their patience in helping me with their software is much appreciated. I also am thankful to those who have made their 3D data available for public use, including Cyberware, the Stanford Graphic Lab, and the Georgia Institute of Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this dissertation, we develop techniques to fully automate the process of constructing a digital, three-dimensional (3D) model from a set of 3D views of a static scene obtained from unknown viewpoints. We begin by introducing the modeling from reality problem (section 1.1) and describing how existing, non-automated systems solve the problem using range sensors (section 1.2). The limitations of these systems motivate the need to automate modeling from reality. Section 1.3 defines the multi-view surface matching problem, which must be solved in order to automate modeling from reality, and describes our framework for solving it. In section 1.4, we list a variety of modeling from reality applications that illustrate the wide applicability of automatic modeling from reality, and in section 1.5, we describe alternative modeling from reality approaches that do not rely on range sensing. We conclude the chapter with an outline of the remainder of the dissertation (section 1.6).

1.1 Modeling from reality

Three-dimensional modeling from reality is the process of creating 3D digital models of real-world objects and environments. We use the generic term *scene* to refer to objects and environments when the distinction is unimportant. Modeling from reality finds application in a surprising number of domains, ranging from archaeology to virtual reality. For example,

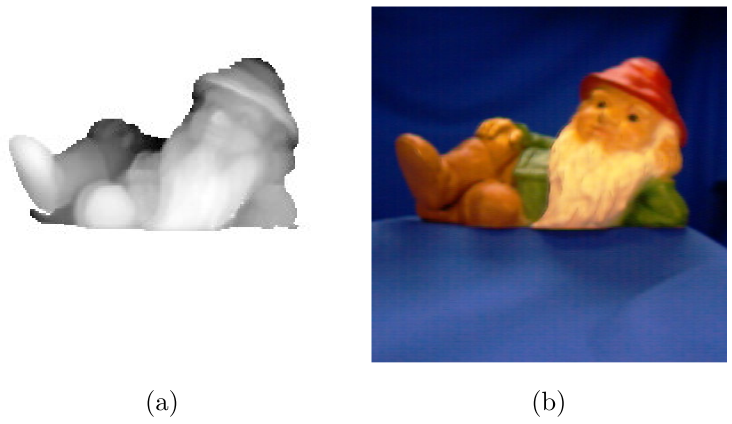


Figure 1.1: a) An example range image obtained with the Minolta Vivid 700 laser scanner. b) The corresponding co-registered color image.

consider the Great Buddha of Kamakura. Exposure to the elements is slowly destroying this national treasure of Japan. Ikeuchi and a team of researchers created a digital model of the Buddha in order to preserve it for all time [35][51]. As a second example, imagine taking a virtual tour of the Roman Colosseum. Although photographs are interesting, a 3D model of the landmark would allow for user interaction, enabling visitors to explore the Colosseum without traveling half-way around the world. If the experience is to be believable, modeling from reality should be employed to create a realistic 3D model.

Depending on the application, different aspects of a digital model may be important. With some applications, such as quality control of manufactured parts, geometric accuracy is paramount. In other cases, such as the Roman Colosseum virtual tour, precise geometry is not as critical as photometric accuracy (i.e., surface appearance). In our work, we focus on geometric accuracy. Knowing the geometry of a scene can simplify the estimation of its photometric attributes [55]. Research has shown that range scanners are an effective sensing modality for creating geometrically accurate models [35][46][4][1].

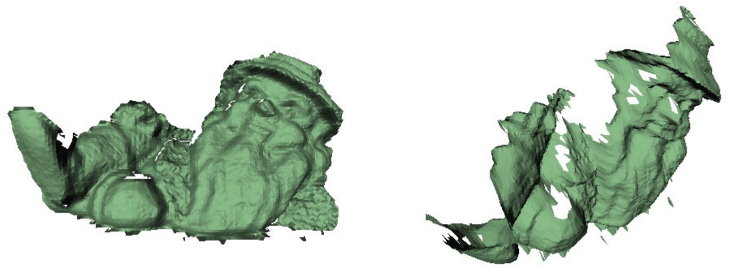
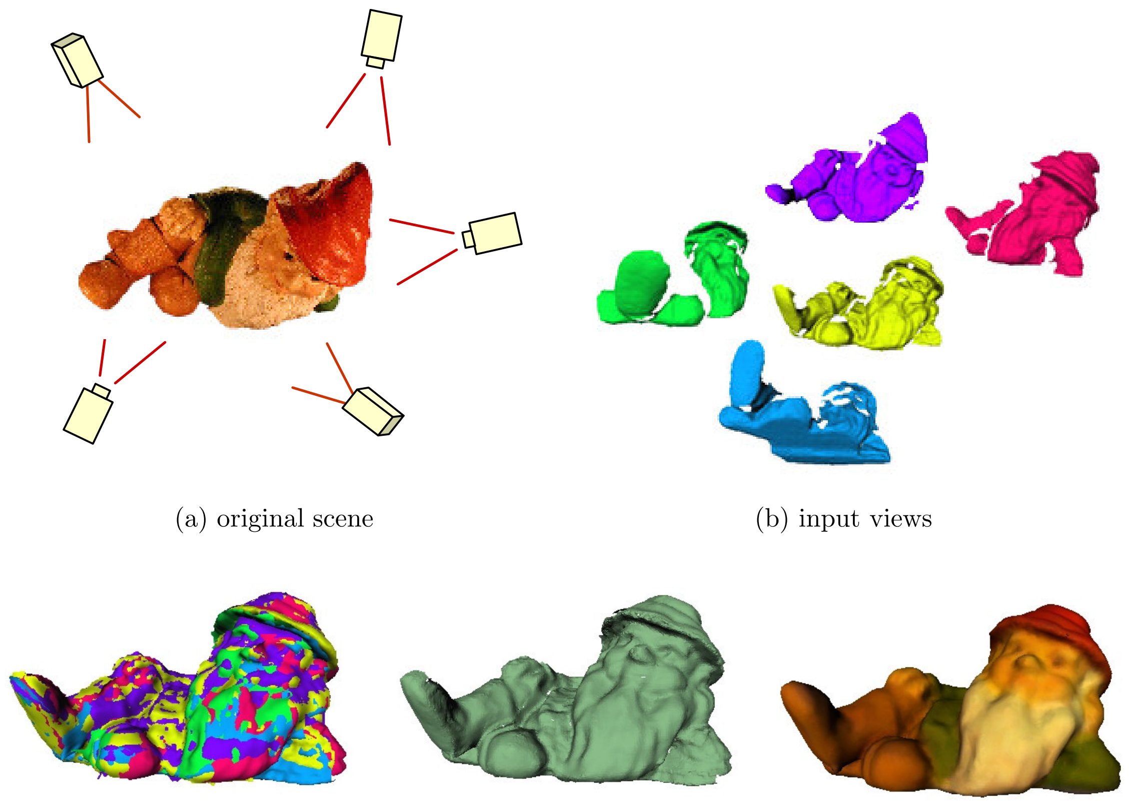


Figure 1.2: An example 3D surface derived from the range image in figure 1.1. While the 3D surface looks fine when seen from the original viewpoint, observing it from a different viewpoint reveals the missing regions.

1.2 Modeling using range scanners

Range scanners densely measure the distance from the sensor to points in the scene, typically in a regular grid pattern. This grid pattern can be visualized as a *range image* – an image in which pixel intensity is a function of range (figure 1.1a). A range scanner can be thought of as a camera that records range images rather than the images captured by a conventional camera. Range scanners can measure distances up to a hundred meters or more with centimeter-level accuracy. At shorter ranges, on the order of a few meters, millimeter-level accuracy is possible.

A range scanner measures the 3D structure of a scene from a single viewpoint. Generally, some parts of the scene will be unobservable from any given position, either due to occlusion or limitations in the sensor's field of view. When seen from a slightly different viewpoint, the missing data in unobserved regions is readily apparent (figure 1.2). To form a complete model of a scene, the range data from multiple viewpoints must be combined. Given a collection of single viewpoint 3D data sets (*views*) of a scene, the modeling from reality process involves two main phases: registration and integration (figure 1.3). In the registration phase, the views are all aligned in a common coordinate system. In the integration phase, the registered views are combined into a unified representation, a process generally called surface reconstruction. Depending on the application, additional processing



(a) original scene

(b) input views

(c) registered views

(d) integrated model

(e) texture mapped model

Figure 1.3: The modeling from reality process. a) The scene to be digitized is scanned from various viewpoints. b) The resulting input views. c) The registration phase aligns the views in a common coordinate system. d) The integration phase reconstructs the original surface. e) Optional texture mapping adds realism to the reconstructed scene.

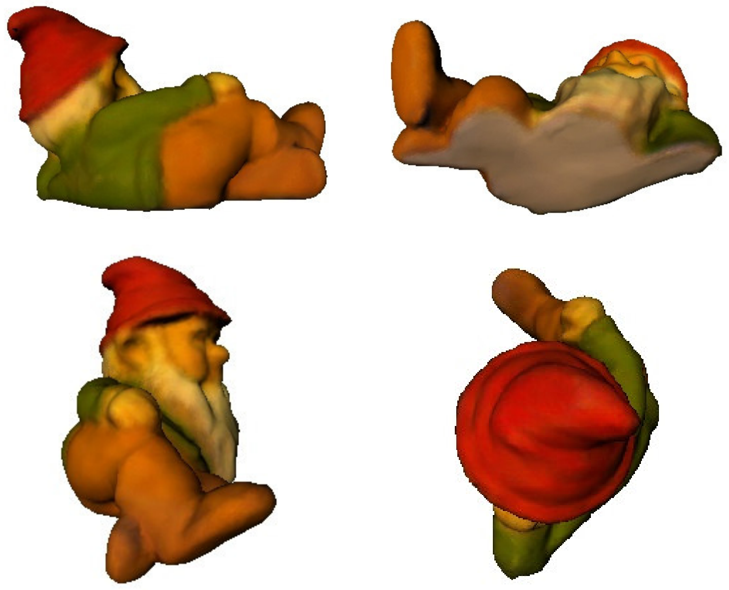


Figure 1.4: Result of modeling from reality. The output model can be examined from any viewpoint. This model (and all the digitized models in this dissertation) was created by our automatic modeling from reality system (chapter 8).

may be performed. For example, texture maps, which can be generated from color images obtained during data collection, may be added to improve the realism of the geometric model (figure 1.4).

Our goal is to fully automate the modeling from reality process. We focus on the registration phase because that is where the central automation issues lie. With a range scanner, the position of the 3D data with respect to the sensor is known. Consequently, registering the input views is equivalent to estimating the pose from which each view was obtained. A pose estimate is a rigid body transform (e.g., three rotations and three translations) and can be specified for a single view in world coordinates (*absolute pose*) or with respect to a pair of views (*relative pose*).

Existing modeling from reality systems register the input views using a variety of techniques, including calibrated pose measurement, manual registration/verification, and environmental modification. Calibrated pose measurement systems measure the absolute poses directly through mechanical means. The mechanism involved depends on the scene size. For small objects, the camera can be mounted on a robot arm [83], or it can be held fixed while the object moves on a calibrated platform such as a turntable [79]. With larger scenes, accurate measurement of camera pose becomes considerably more complex. For example, a custom-made gantry system was required for scanning statues in the Digital Michelangelo project [47].

With manual registration, the user may specify corresponding feature points in view pairs, from which relative poses can be estimated [53]. In some systems, corresponding feature points are automatically detected and then manually verified for correctness [21]. Alternatively, the 3D data can be aligned directly by manually rotating and translating the raw data [62]. In more advanced approaches, the user indicates only which views to register, the system registers view pairs with no initial estimate of relative pose, and the user manually verifies the results [36][1]. If the motion between successive scans is small,[1] simpler registration algorithms that rely on greedy, local search can be used instead. This technique was used in the Great Buddha project [54] and in Rusinkiewicz' real-time modeling system [67].

Environmental modification generally takes the form of markers (known as fiducials). The markers may be physical objects added to the scene, such as reflective tape or Styrofoam balls, or they may be patterns of light projected onto the scene. In either case, the markers serve as feature points to aid in manual registration. For example, in the Pietà project [4], researchers found that manual registration was much simplified by projecting a pattern of dots onto the scene.

While many commercial 3D modeling systems claim to have an "automatic modeling" capability, to our knowledge, all existing systems (commercial or otherwise) use some combination of these three registration techniques. These methods have significant limitations. Calibrated pose measurement systems add cost and complexity to the sensor and limit the scale of scenes that can be modeled. For example, it is impossible to model a building

[1]By small, we mean rotation on the order of 10° and translation on the order of 10% of the scene size.

exterior using a turntable system. With turntable systems, parts of the scene – such as an object's top and bottom – cannot be seen in a single rotation, so the object must be moved and scanned a second time and the results manually registered. Manual registration is tedious and time-consuming, which limits the scalability and commercial viability of modeling from reality. We offer a third alternative: automatically register the views without resorting to pose measurements or manual intervention. We call this technique *multi-view surface matching*, so named because it can be seen as an extension of surface matching to more than two views, as we will show later.

1.3 Automatic modeling from reality

By combining multi-view surface matching with surface reconstruction, we can fully automate the modeling from reality process. More precisely, we define the automatic modeling from reality problem as follows:

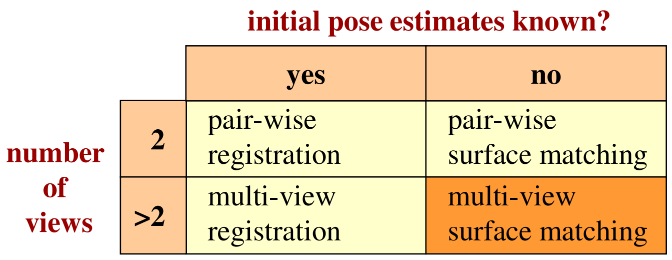
> **The automatic modeling from reality problem**
> Given an unordered set of 3D views of a static scene, automatically construct a geometrically accurate 3D digital representation of the scene.

The multi-view surface matching problem is defined similarly, since it is a sub-problem of automatic modeling from reality.

> **The multi-view surface matching problem**
> Given an unordered set of 3D views of a static scene, recover the original sensor viewpoints,[2] thereby aligning the 3D data in a common coordinate system.

We do not assume any prior knowledge of the original sensor viewpoints or even which views contain overlapping scene regions (*overlaps*). Furthermore, the views are unordered, meaning that consecutive views are not necessarily close together spatially. This problem

[2]The original sensor poses can be determined only up to a rigid body transform. In practice, we express the poses with respect to an arbitrarily selected input view, which we call the *base view*.



initial pose estimates known?

number of views	yes	no
2	pair-wise registration	pair-wise surface matching
>2	multi-view registration	multi-view surface matching

Figure 1.5: A simple registration taxonomy showing the relationship between the four types of registration used in our automatic modeling framework. This dissertation contributes the first algorithms in the multi-view surface matching category.

is analogous to assembling a jigsaw puzzle in 3D. The views are the puzzle pieces, and the problem is to correctly put the pieces together without even knowing what the puzzle is supposed to look like.

The relationship between multi-view surface matching and existing 3D registration problems is illustrated by the simple taxonomy in figure 1.5. The taxonomy contains two axes: the number of input views and whether initial pose estimates are known. When a good initial estimate of the relative pose is known, aligning two views is called *pair-wise registration*. The most popular pair-wise registration algorithm is the Iterative Closest Point (ICP) algorithm [5]. When more than two views are involved and initial pose estimates are given, the process is called *multi-view registration*. Multi-view registration is often used in modeling from reality systems to distribute pair-wise registration errors evenly over an entire model. With unknown pose estimates, aligning a pair of views is called *(pair-wise) surface matching*. Surface matching is often used in 3D object recognition systems to locate models within a 3D scene. Finally, multi-view surface matching occupies the fourth corner of this taxonomy, extending pair-wise surface matching to more than two views. To our knowledge, our algorithms are the first in this category. The problems in the first three categories are interesting research topics on their own, but they are particularly relevant here because we use algorithms from each category as components of our multi-view surface matching framework. Rather than reinvent these components, we build on existing algorithms in each category. For pair-wise surface matching, we use a modified version of Johnson's spin-image

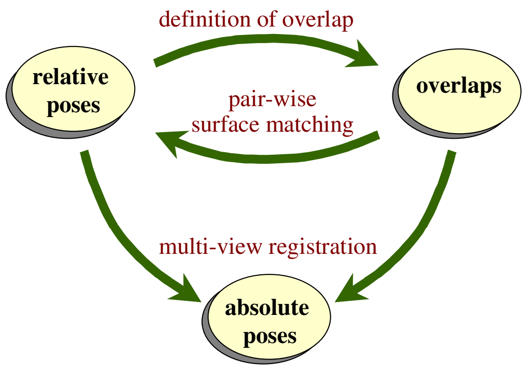


Figure 1.6: The multi-view surface matching problem is challenging due to the inter-relationship between the unknown relative poses, overlaps, and absolute poses.

surface matching engine [36][38], and for pair-wise and multi-view registration, we employ Neugebauer's algorithm [53]. Since these are modular components in our framework, it is easy to replace them with alternative algorithms if better-performing ones are developed.

1.3.1 Difficulties of multi-view surface matching

Multi-view surface matching is a challenging problem, and it is not immediately clear that it can be solved at all. To begin with, the problem is ill-posed. Depending on the scene shape and the choice of input views, there may be an infinite number of incorrect, yet entirely reasonable, ways to arrange the 3D views. For example, consider a set of views of a ping-pong ball. Since the ball is a featureless sphere, we cannot determine the exact position of any view on the sphere or even whether the original object was a sphere, a hemisphere, or something in between. Fortunately, the real world usually contains sufficient irregularity that we can solve the multi-view surface matching problem for a variety of scenes.

Even without such insidious input data, multi-view surface matching is still difficult because it involves solving three interrelated sub-problems (figure 1.6): 1) determining which views overlap; 2) determining the relative pose between each pair of overlapping views; and 3) determining the absolute poses of the views – the output of the algorithm. The difficulty lies in the mutual dependency between the overlaps and relative poses. If the relative poses

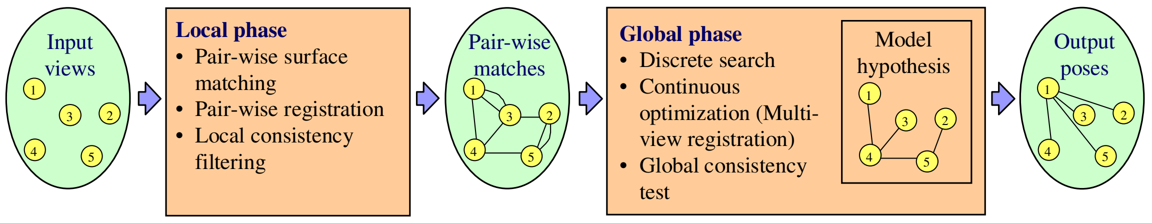

Input views
Local phase
• Pair-wise surface matching
• Pair-wise registration
• Local consistency filtering
Pair-wise matches
Global phase
• Discrete search
• Continuous optimization (Multi-view registration)
• Global consistency test
Model hypothesis
Output poses
1
2
3
4
5

Figure 1.7: Block diagram for our multi-view surface matching framework. See text for details.

are known, the overlapping views can be determined by applying a suitable definition of overlap to the registered pairs. Conversely, if the overlaps are known, the relative poses can be found using a surface matching algorithm and manually verifying the results. Once the overlaps and relative poses are known, the absolute poses can be determined by a suitable multi-view registration algorithm. For multi-view surface matching, both the overlaps and the relative poses are unknown, which makes the problem considerably harder.

1.3.2 Overview of approach

Our framework for multi-view surface matching involves two phases (figure 1.7) – a local phase, which operates only on pairs of views, and a global phase, which considers sets of three or more views. In the local phase, we perform pair-wise surface matching on all pairs of views, generating a graph of relative pose estimates (*matches*), some of which may be incorrect. The sub-graphs of this graph represent the set of possible model hypotheses for the given input views. In the global phase, we search this graph for a model hypothesis containing only correct matches, distinguishing incorrect matches from correct ones by exploiting global consistency constraints. The algorithm outputs a model hypothesis that partitions the input views into one or more sets (*parts*) and computes the absolute poses for the views within each part.

We now explain the process in more detail and illustrate it with a 2D toy example (figure 1.8). First, we perform surface matching between all pairs of views. Each match is then refined using pair-wise registration to obtain the best possible pair-wise alignment. If a pair of views contains overlapping scene regions, pair-wise surface matching often finds

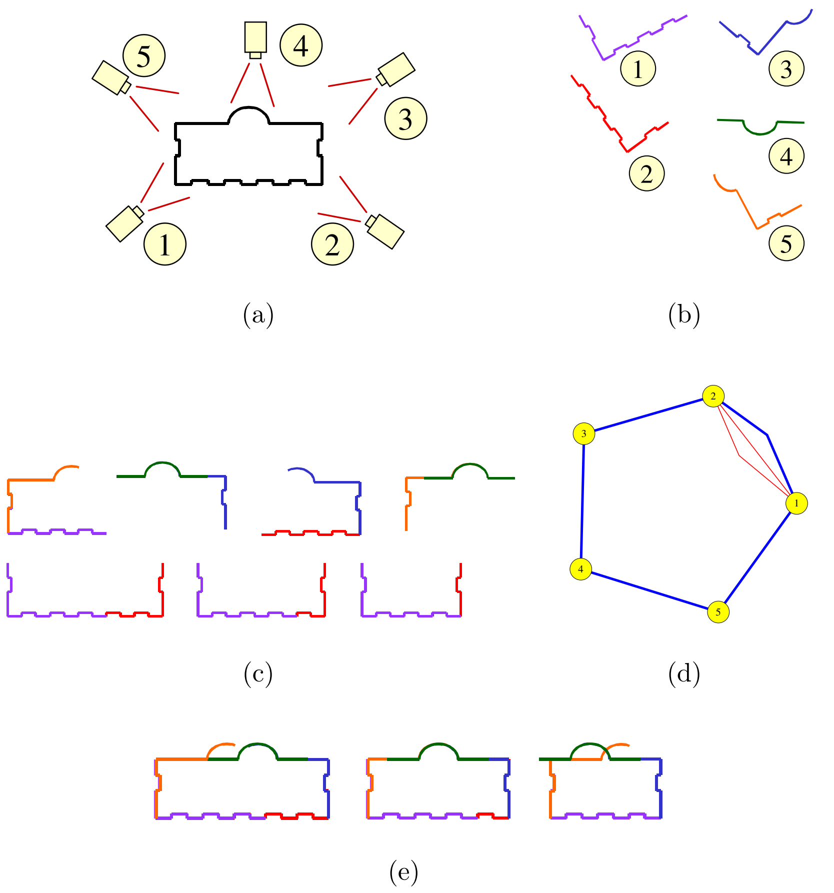


Figure 1.8: A 2D example of the multi-view surface matching process. a) Five views of a building are obtained (shown from the top). b) The resulting views. c) Pair-wise surface matching finds a set of candidate matches, some of which are incorrect but undetectable at the pair-wise level. d) The pair-wise matches are inserted into a model graph. The graph has been hand labeled for illustration – thick blue lines for correct matches and thin red lines for incorrect ones. d) A globally consistent model is constructed by searching the model graph. Depending on which match is used between views 1 and 2, there are three possible outcomes. Only the center one is globally consistent.

the correct relative pose, but it may fail for a number of data-dependent reasons (e.g., not enough overlap or insufficient complexity of the surfaces). Even if the views do not overlap, surface matching may find one or more plausible, but incorrect, matches. We remove the worst matches from consideration using a local consistency test. Multi-view surface matching would be greatly simplified if this test could determine with certainty which pair-wise matches are correct and which are incorrect. However, this classification cannot be accomplished just by looking at pairs of views; two views could have zero registration error but still be an incorrect match. Instead, we must consider the constraints imposed by of an entire network of views. This idea, which we call *topological inference*, is the key to our approach. By assembling several views, incorrect matches can be identified by detecting surface inconsistencies between multiple views (figure 1.8e). Using the jigsaw puzzle analogy, sometimes a piece can be inserted incorrectly in a puzzle because it fits well with another piece. Later, when the remaining surrounding pieces are added, the mistake is discovered because the piece does not fit on all sides.

The aforementioned network of views is called a *model graph*; it contains a node for each input view. We insert the locally consistent matches from pair-wise surface matching as edges into a model graph G_{LR} (LR stands for local registration) (figure 1.8d). If we can find a connected sub-graph of G_{LR} that contains only correct matches, it is straightforward to convert the relative poses associated with the matches into absolute poses for each view. We formulate this search as a mixed continuous and discrete optimization problem. The discrete optimization performs a combinatorial search over the space of connected sub-graphs of the model graph, using a global consistency measure to detect and avoid incorrect, but locally consistent matches. The continuous optimization, which is essentially a multi-view registration algorithm, adjusts the absolute pose parameters to minimize the registration error between all overlapping surfaces, distributing the pair-wise registration errors in a principled way.

Unfortunately, a connected sub-graph containing only correct matches may not exist within G_{LR}. This could happen if one of the input views is corrupted or if some subset of the input views does not overlap sufficiently with any of the remaining views (e.g., front views and back views of an object, but no side views). We have developed a second class of search algorithms that handle these situations gracefully by searching the space of *all* sub-graphs of G_{LR} for the *best* model hypothesis rather than searching only for globally consistent connected sub-graphs.

With this framework, we have reduced the multi-view surface matching problem to two fundamental questions: 1) What constitutes a "good" model hypothesis? and 2) How do we robustly and efficiently search for the best model hypothesis in an extremely large hypothesis space? We answer the first question in chapter 3 where we develop a framework for evaluating the local quality (Q_{L}) of pair-wise matches and the global quality (Q_{G}) of model hypotheses. For local quality, we use maximum likelihood estimation to learn a probabilistic model of registration success. This method offers several advantages over traditional registration metrics, including superior discriminability, automatic determination of model parameters (i.e., no thresholds to tweak), and a principled way to combine multiple, disparate measures of registration quality. We then show how the local quality measures can be used to define global quality measures and global consistency tests (C_{G}) for model hypotheses. In chapter 4, we answer the second question, describing two classes of search algorithms. One class, which we call iterative addition algorithms, searches deterministically for a globally consistent model hypothesis by repeatedly adding edges to an initially empty model graph, and the second class, which we call iterative swapping algorithms, uses a stochastic edge swapping approach to robustly search for the best model hypothesis according to the given global quality measure.

In chapter 7, we demonstrate that our multi-view surface matching algorithms work independently of the scene size, scene type, and type of range sensor used. We show results using three different range sensors: the Minolta Vivid 700 (Vivid), the Zoller and Fröhlich LARA 25200 (Z+F), and the K2T Ben Franklin 2 (BF2). Details of these sensors are given in appendix A. Using these sensors, we have automatically modeled scenes varying in scale from from 20 centimeters to 200 meters. We have modeled a variety of scene types, including

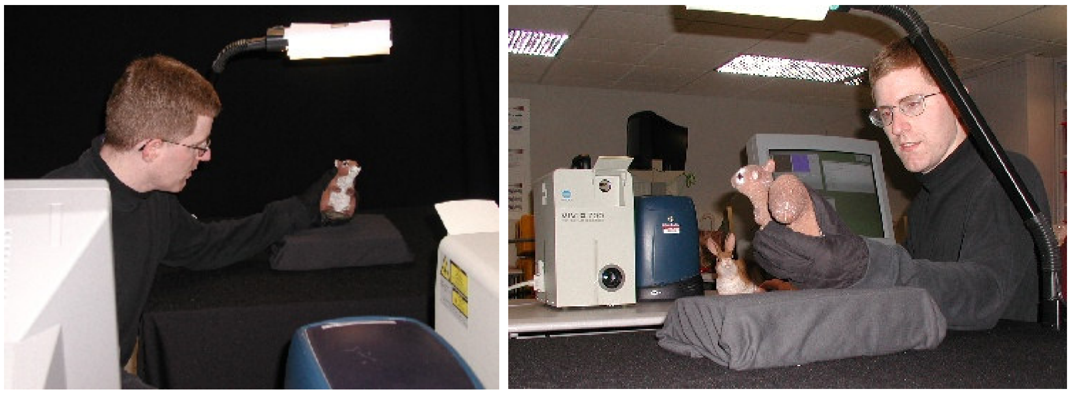


Figure 1.9: The hand-held modeling application. The object is held before the scanner while range images are obtained from various viewpoints (black clothes simplify the problem of background removal). The FAMUS system takes the range images and automatically produces a 3D model.

small, "desktop" objects, building interiors, and terrain. While some existing modeling from reality methods may be capable of automating one or two of the scenarios described here (e.g., turntable systems can model small objects with the Vivid), our approach is the only one capable of automating all of these scenarios within a single framework.

1.4 Applications of automatic modeling from reality

We have implemented our multi-view surface matching framework in a system called Fully Automatic Modeling of Unstructured Scenes (FAMUS). The FAMUS system joins our multi-view surface matching algorithms with an existing surface reconstruction algorithm [15] to form a fully automated modeling from reality system. This system enables new and unique applications such as hand-held modeling (figure 1.9). With hand-held modeling, the object to be digitized is held before a laser scanner while range images are obtained from various viewpoints. This is an easy data collection method, requiring no additional hardware, minimal training, and only a few minutes to scan an average object. Once data collection is complete, our system automatically generates a digital 3D model of the original scene.

Hand-held modeling offers several advantages over modeling systems that use turntables or some other type of movable stage. An object can be placed in an arbitrary pose in order to scan hard-to-see regions of complex objects that would not be observable otherwise. Many range scanners produce more accurate measurements for frontally viewed surfaces. With hand-held modeling, the object can be scanned from all directions, minimizing the error in regions that would only be seen obliquely by a turntable-based scanner.

Rather than holding an object before a stationary range scanner, the user can instead move a portable range scanner around to scan a static scene. In this way, building interiors, exteriors, and terrain are easily modeled. We envision that people will eventually use 3D cameras to create models like they use ordinary cameras to create photo-mosaics today. They would simply snap a few 3D pictures and automatically create a photo-realistic 3D model in just a few minutes.

Finally, automatic modeling from reality also has the potential to greatly simplify existing modeling from reality applications. As previously mentioned, modeling from reality is used in a variety of domains, ranging from archaeology to virtual reality. Here, we describe applications in a few of these areas.

- **Archaeology** – Many historical sites are slowly deteriorating due to exposure to the elements. Although remediation efforts can reduce the rate of destruction, a digital model of the site will preserve it indefinitely. Examples of this type of work include the Great Buddha project [51], the Digital Michelangelo project [47], and the Pietà project [4]. Models of historical artifacts also allow scientists to study the objects in new ways. For example, archaeologists can measure artifacts in a non-contact fashion, which is useful for fragile objects. Also, digital models allow objects to be studied remotely, saving time and travel expenses and enabling more archaeologists to study them.

- **Civil engineering** – Modeling from reality offers an efficient alternative to surveying. For example, bridges need to be surveyed periodically to determine whether significant settling or other movement has occurred. A 3D model of the site allows engineers to make the same measurements with equivalent accuracy in a fraction of the time [88].

- **Special effects** – Hollywood's special effects companies often use 3D models of movie sets to realistically integrate computer generated (CG) effects with live action. In the movie Starship Troopers, a model of a cave set was created using a range scanner, allowing CG aliens to crawl on the walls and attack the human actors [88].

- **Reverse engineering and rapid prototyping** – Modeling from reality can be used to reverse engineer manufactured parts for which a CAD model is unavailable or never existed. For example, a design engineer at an auto manufacturer might create a hand-carved model of a prototype car. Modeling from reality "imports" the prototype into a computer, creating a digital model that can be edited with a CAD program.

- **Architecture and construction** – Architects frequently need to determine the "as built" plans of buildings or other structures such as industrial plants, bridges, and tunnels. A 3D model of a site can be used to verify that it was constructed according to specifications. When preparing for a renovation or a plant upgrade, the original architectural plans might be inaccurate, if they exist at all. A 3D model allows architects to plan a renovation and to test out various construction options [89].

- **Robot navigation** – Robots navigating over rough terrain benefit from 3D models for route planning and obstacle avoidance. In the Mars Pathfinder mission, 3D models of the area around the base station allowed scientists to visualize the environment and direct the rover's daily travels. Automated 3D modeling methods would have significantly reduced the man-hours required for this task [87].

- **Virtual reality** – Instead of constructing virtual reality environments by hand, as is common practice today, real environments can be modeled. For example, 3D models of houses would allow virtual walkthroughs for the real-estate market. Museums and historical sites could create virtual tours, which would be educational as well as entertaining (e.g., the Roman Colosseum example in section 1.1).

1.5 Alternative methods for modeling from reality

Although the focus of this work is on 3D modeling using range scanners, there are two primary alternative approaches to 3D modeling — contact sensing and modeling from images — each of which has advantages and disadvantages when compared to the range scanner method. Three-dimensional models can also be constructed from volumetric sensing techniques, including computed tomography (CT), magnetic resonance imaging (MRI), and sonar (e.g., sonograms); however, approaches using these modalities are beyond the scope of this work.

1.5.1 Contact sensing

Contact sensing is the traditional method for digitizing 3D objects in industry. With contact sensing, the 3D position of surface points is measured with a physical probe, which can be automated or hand-held. The automated devices are called coordinate measurement machines (CMMs) and are primarily used for high-precision verification of manufactured parts. The hand-held devices consist of a pencil-like stylus, which the user touches to the object surface, measuring one point at a time. The pose of the stylus is measured by optical, electro-magnetic, sonic, or mechanical means.

Contact sensing can be extremely accurate (on the order of a few micro-meters). A CMM can probe surfaces that are difficult to sense with a range scanner, including transparent and mirror-like objects. On the other hand, they cannot probe soft or fragile objects. Contact sensing is also much slower than range scanners (1 point/second versus 120,000 points/second) and is limited to objects less than a few meters in size.

1.5.2 Image-based methods

A variety of 3D modeling techniques have been developed that only rely on individual images or video sequences of the scene. Some of these methods (e.g, structure from motion) are beginning to be introduced into the commercial domain for targeted applications such as video editing.

Multi-camera systems

Multi-camera systems provide a bridge between range-image and intensity-image techniques. Stereo vision can provide a dense, but relatively noisy range-image of a scene. Using several cameras (i.e., multi-baseline stereo) can reduce the errors by integrating multiple independent range estimates [41]. In the extreme case, a scene can be entirely surrounded by cameras as in Carnegie Mellon University's (CMU's) virtualized reality room [63][68]. A 3D model can be created by combining the range images generated using multi-baseline stereo on many subsets of the cameras. The range images are automatically registered since the absolute poses of the cameras are precisely calibrated in advance.

Some multi-camera systems capture images in real-time, allowing moving scenes to be modeled [80]. On the downside, these systems are highly complex, non-portable, and require significant computational power. Furthermore, the range accuracy of stereo is worse than that of range scanners and untextured surfaces are difficult or impossible to model.

Structure from motion

Structure from motion (SFM) relies on the ability to locate and track stable features (e.g., corner points) over a sequence of images. Given a sufficient number of tracked 2D features, the 3D location of the features and the camera poses can be recovered simultaneously. Tomasi and Kanade developed the factorization method [77], a batch algorithm for solving the SFM problem, that helped popularize the problem in the computer vision community. However, SFM has its roots in the field of photogrammetry, where it is known as bundle adjustment [78]. Recent work extends SFM to handle unknown correspondences [17] and multiple moving rigid objects [25].

Structure from motion systems require only a single video camera and are therefore low-cost and portable. The algorithms compute 3D structure with lower accuracy than with range sensors and only at the relatively sparse feature points. Dense 3D structure can be estimated in a separate step [58], but untextured regions are still a problem. Correspondence matching between non-consecutive images can be difficult in sequences with complex camera motions. For example, if the sequence completely circles an object, the features from the first and last frames should be matched.

Shape from silhouette and space carving

Given a sequence of images of an object obtained from known camera positions, a 3D model can be constructed by removing parts of the viewing volume that are exterior to the object. The resulting shape is known as the visual hull [45], and its calculation traditionally required complex volume intersection operations. Recent work has shown that the visual hull can be efficiently computed entirely in the image domain [22]. While visual hulls exploit the constraints imposed by the background regions, Kutulakos and Seitz' space carving algorithm [44] exploits photo-consistency constraints of the object itself. Their algorithm begins with a solid volume and removes voxels that do not have consistent appearance (i.e., color) in all images in which they are observed. Recent work extends the framework by modeling voxel occupancy probabilistically [8]. These methods require known camera positions and work best with a controlled background. Shape from silhouette methods cannot model certain concavities, such as the inside of a bowl, while space carving methods can. Neither method can guarantee geometric accuracy, since there is an equivalence class of shapes that will produce the observed images.

Geometric primitives

For regularly-shaped environments such as buildings, it is possible to create 3D models using a small set of geometric primitives such as rectangular blocks, pyramids, and cones. Debevec's Façade system [16] exemplifies this approach. With Façade, an approximate model is hand-constructed from the set of geometric primitives, and an optimization algorithm finds the parameters of the primitives that minimize the disparity between the observed image edges and the model edges projected into the images. More recent work fully automates the process and eliminates Façade's requirement of initial camera pose estimates [81]. The main drawback to these approaches is the limited class of scenes to which they can be applied.

1.6 Dissertation outline

The rest of this dissertation is organized as follows: First, in chapter 2, we formally define the terms used throughout this document. Chapters 3 and 4 contain the core theoretical contributions of this work. In chapter 3, we present our maximum likelihood framework for learning and evaluating the local quality of pair-wise matches and the global quality of model hypotheses, and in chapter 4 we describe our multi-view surface matching search algorithms and give examples of each algorithm operating on real data. Chapter 5 provides the details of the component algorithms used in our framework (pair-wise surface matching, pair-wise registration, and multi-view registration). In chapter 6, we explain our methodology for evaluating the success or failure of multi-view surface matching. Chapter 7 contains experiments that demonstrate the capabilities of our multi-view surface matching algorithms and the component algorithms embedded within. This chapter also shows a large number of example scenes that were automatically modeled using our algorithms. Next, chapter 8 describes our automatic modeling system, including the preprocessing and postprocessing operations involved and various implementation details. Finally, the dissertation concludes in chapter 9 with a summary of contributions and directions for future work.

Chapter 2

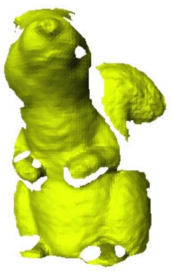
Definitions and background

This chapter covers the background material for our multi-view surface matching framework. We begin by formally defining the terms used informally in the introduction (sections 2.1 through 2.3). Next, we describe the model graph and its associated terminology (section 2.4). Then, in section 2.5, we revisit the component algorithms introduced in section 1.3, giving a high-level description and example output for each class of algorithms: pair-wise surface matching, pair-wise registration, and multi-view registration. Finally, in section 2.6, we discuss topological inference. Throughout the next few chapters, we use data from a real object[1] for demonstration (figure 2.1).

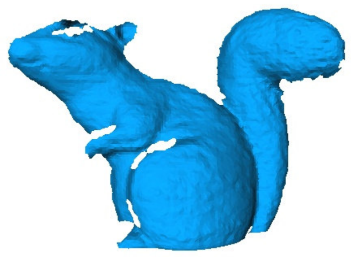
2.1 Views and surfaces

A single scan from a range scanner is called a *view*. A view V consists of a 3D surface S and the intrinsic parameters $\mathcal{I}$ (e.g., focal length, field of view, etc.) under which the scan was obtained. The surface S is represented in the sensor's local coordinate system and can be computed from a range image using the intrinsic parameters, which are assumed to be known. The representation of S is relevant primarily for implementation purposes. We use triangular meshes, a representation well-suited for describing the types of surfaces captured

[1]The data is a subset of views taken from the squirrel1 test (see chapter 7).

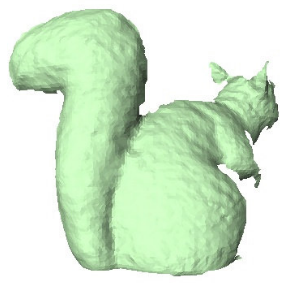


(a) view 1

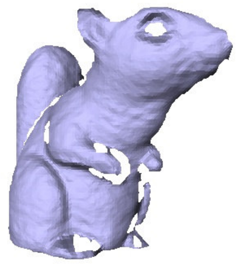


(b) view 3

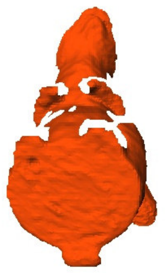
(c) view 5



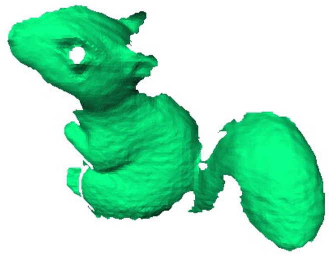
(a) view 7



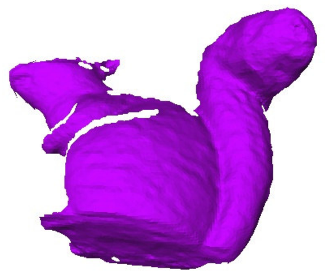
(b) view 9



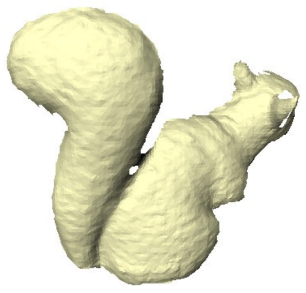
(c) view 11



(a) view 13



(b) view 15



(c) view 17

Figure 2.1: Input views for a simple demonstration scene (a squirrel model). This is a subset of views from a larger test set.

by range cameras – piecewise smooth surfaces with numerous holes and discontinuities. Other representations are possible, including range images, point clouds, and parametric surfaces (e.g., NURBS). We define the *size* of a view to be the diagonal length of the axis-aligned bounding box enclosing its surface. At times, we use the term "view" to refer to the surface associated with the view. The meaning should be obvious from the context.

2.2 Relative and absolute poses

A *pose* is a six degree of freedom rigid body transform T. A *relative pose*, T_{ij}, is a transform between two views V_i and V_j. The transform T_{ij} is a linear operator that takes 3D points from the coordinate system of V_j to that of V_i (i.e., $\boldsymbol{p}_i = T_{ij}\boldsymbol{p}_j$). Similarly, an *absolute pose*, T_i, is a transform from the coordinate system of view V_i to the world coordinate system (i.e., $\boldsymbol{p}_w = T_i\boldsymbol{p}_i$). This distinction between relative and absolute poses is somewhat superficial because the absolute pose T_i can be written as the relative pose, T_{wi}, where V_w is a dummy view in the world coordinate system. However, the separate terms are helpful in understanding the multi-view surface matching problem.

2.3 Closest points and the definition of overlap

A *point* is a vector in $\mathcal{R}_3$ ($\boldsymbol{p} = (p_x, p_y, p_z)^{\mathrm{T}}$). An *oriented point* $\boldsymbol{p}_o = (\boldsymbol{b}, \boldsymbol{l})$ consists of a base point $\boldsymbol{b}$ and a direction vector $\boldsymbol{l} = (l_x, l_y, l_z)^{\mathrm{T}}$. Typically, an oriented point is sampled from a surface, in which case, $\boldsymbol{b}$ is the surface point and $\boldsymbol{l}$ is the surface normal at $\boldsymbol{b}$. We call the minimum Euclidean distance between a point $\boldsymbol{p}$ and a surface S the *closest point distance*:

$$D_{\mathrm{CP}}(\boldsymbol{p}, S) = \min_{\boldsymbol{q} \in S} \|\boldsymbol{p} - \boldsymbol{q}\| \tag{2.1}$$

The point $\boldsymbol{q}$ is called the *closest point*.

The overlapping region of two surfaces is the area of each surface that corresponds to the same physical surface. In our case, the ground truth correspondence between the two surfaces is unknown, so an algorithmic definition of overlap requires some subtlety.

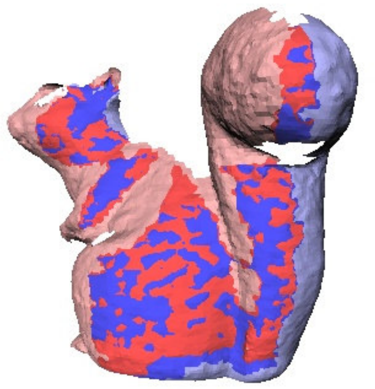


Figure 2.2: An example of the overlapping region for two registered surfaces, one red and one blue. The dark red and dark blue patches indicate the overlapping region.

Intuitively, two surfaces overlap wherever they are sufficiently close together and have similar orientations. The following definition captures this notion of overlap.

A point $\boldsymbol{p}$ *overlaps* surface S if three conditions hold: 1) the closest point distance $D_{\mathrm{CP}}(\boldsymbol{p}, S)$ is less than a threshold t_D; 2) the closest point $\boldsymbol{q}$ on S is an interior (non-boundary) point of S; and 3) the angle between the surface normal at $\boldsymbol{p}$ and the surface normal at $\boldsymbol{q}$ is less than a threshold t_θ. The thresholds t_D and t_θ can be set based on the measurement uncertainty for a given range camera.[2] The set of points $\mathcal{P}$ on surface S_i that overlap surface S_j is called the *overlapping region* of S_i (figure 2.2). We say two surfaces/views *overlap* if their overlapping region is non-null.

2.4 Model graphs and model hypotheses

A model graph is an attributed undirected graph $G = (N, E, A, B)$ that encodes the topological relationship between overlapping views (figure 2.3). G contains a node N_i for each input view V_i. We usually refer to nodes by their corresponding view (i.e., instead of saying "the view associated with N_i," we simply say "V_i"). The attributes A_i for node N_i consist

[2]In our implementation, we estimate t_D from the registration residual as described in section 5.3.2 and t_θ is set to 45°.

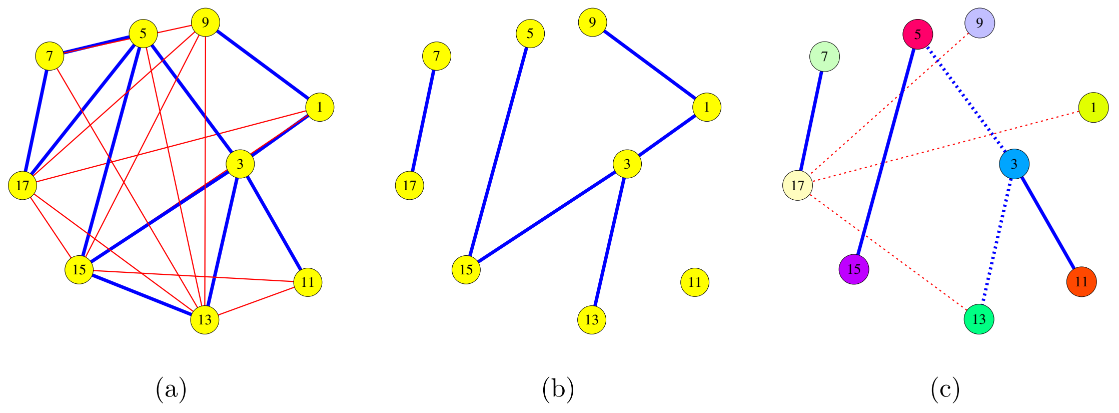


Figure 2.3: Model graphs and model hypotheses. a) An example model graph for the views in figure 2.1. Matches are labeled as correct (thick/blue lines) or incorrect (thin/red lines) for illustration. b) A model hypothesis for the iterative addition class of algorithms. The hypothesis corresponds to a model with three parts, one for each connected component. c) A model hypothesis for the iterative swapping class of algorithms. The hypothesis corresponds to a model with three parts. Hidden edges (dashed lines) indicate divisions between parts, which are connected by visible edges (solid lines).

of the absolute pose T_i for view V_i. The attributes B_{ij} for edge E_{ij} include the relative pose, T_{ij}, between V_i and V_j, and the local registration quality Q_{L} for the view pair. An edge E_{ij} in G indicates that S_j, transformed by T_{ij}, overlaps S_i. Note that our model graph figures show the topological relationship but not the geometric relationship between views. The position of a node in such figures does not correspond to the physical location of the original sensor viewpoints.

We use the following notation from basic graph theory [13]. Two views V_i and V_j are *adjacent* in G iff $E_{ij} \in G$. The adjacency function $\texttt{adj}(V_i)$ outputs the set of views $\mathcal{V}_{\mathrm{adj}}$ that are adjacent to V_i in G. A *path* of length k from V_i to V_j in G is a sequence of edges from G $\langle(s_0, s_1)(s_1, s_2)\ldots(s_{k-1}, s_k)\rangle$ such that $s_0 = i$ and $s_k = j$. A *cycle* in G is a path where $s_0 = s_k$. Two views are *connected* if there is a path between them. A *connected graph* contains a path between every pair of views. If a graph is not connected, each maximally large[3] connected sub-graph is called a *connected component*. Each component represents a

[3]By maximally large, we mean that the sub-graph, G_s, contains all edges $E_{ij} \in G$ such that $V_i \in G_s$ and $V_j \in G_s$ and there are no edges $E_{ij} \in G$ such that $V_i \in G_s$ and $V_j \notin G_s$.

separate part in the corresponding 3D model.

Using model graphs, we can convert between absolute and relative poses and vice versa. If the absolute poses are known, we can compute the relative pose between any two views V_i and V_j directly: $T_{ij} = T_i^{-1} T_j$. Therefore, converting absolute poses to relative poses is straightforward. When the absolute poses are unknown but the relative poses are known, the relative pose between connected (but not necessarily adjacent) views can be computed by composing the relative poses along a path between the two views. For example, if G contains a path $\langle(1,2),(2,3),(3,4)\rangle$, the relative pose $T_{1,4} = T_{1,2}T_{2,3}T_{3,4}$. We say a model graph is *pose consistent* if the relative pose between any two views is independent of the path used in the calculation. Another way of saying this is that the composition of transforms around any cycle must be the identity transform, T_{I}. If a connected model graph is pose consistent, we can unambiguously compute absolute poses (with respect to a base view V_b) from relative poses by setting the absolute pose of V_b to T_{I} and the absolute poses of the remaining views V_i to T_{bi}. An acyclic connected model graph (i.e., a spanning tree of the views) is guaranteed to be pose consistent since there is only one path between any two views. In the next section, we will see that multi-view registration can be used to convert relative to absolute poses for model graphs that are not pose consistent. If a model graph is not connected, we cannot compute absolute poses for the entire graph, but we can choose a base view for each connected component and compute absolute poses for each component. In summary, converting absolute poses to relative poses is straightforward, as is converting relative poses to absolute poses, provided that the model graph is pose consistent.

A model hypothesis is a model graph that is a potential solution to the multi-view surface matching problem. We use two representations for model hypotheses, one for each class of multi-view surface matching search algorithms. The first representation, which is used for the iterative addition algorithms, is an acyclic sub-graph (i.e., a spanning tree or its sub-graph) of G_{LR} (figure 2.3b). The second representation, which is used in the iterative swapping algorithms, is a spanning tree of G_{LR} with an additional edge attribute called visibility (figure 2.3c). If an edge is hidden (visibility = false), it separates the model into two parts at that point. Visible edges connect the views within each part. The hidden edges are essentially used for bookkeeping in our iterative swapping algorithms and are removed when evaluating the quality of a hypothesis and when outputting the final result. When hidden edges are removed, this hypothesis representation covers the space of all acyclic sub-graphs of G_{LR}, just as the first hypothesis representation does.

Restricting our model hypotheses to be acyclic sub-graphs of G_{LR} has several advantages. First, an acyclic hypothesis is guaranteed to be pose consistent, since there is at most one path between any two views. Pose consistency simplifies the computation of global quality, global consistency, and the initial poses used in multi-view registration. Second, an acyclic hypothesis contains the minimum number of matches necessary to specify the absolute poses, minimizing the chances of including an incorrect match in the solution. For example, a data set with N input views requires $N - 1$ matches to specify a single part model. Finally, the acyclic restriction reduces the size of the hypothesis space without eliminating any correct solutions from the representation. We will discuss this point in section 2.6.

2.5 Component algorithms revisited

With the model graph terminology in hand, we now return to the registration taxonomy introduced in section 1.3. As mentioned previously, these algorithms are used as modular components in our multi-view surface matching framework. Here, we treat the algorithms as black boxes, describing their inputs, outputs, high-level behavior, and effect on a model graph. The low-level details of each algorithm are deferred until chapter 5.

2.5.1 Pair-wise surface matching

Pair-wise surface matching aligns two surfaces when the relative pose between them is unknown, searching globally over the entire six-dimensional space of relative pose parameters. A surface matching algorithm takes two views as input and outputs the relative pose that best aligns the two surfaces (or no pose if the surfaces could not be registered). We call the relative pose estimate output by a surface matching algorithm a *match*. Some algorithms

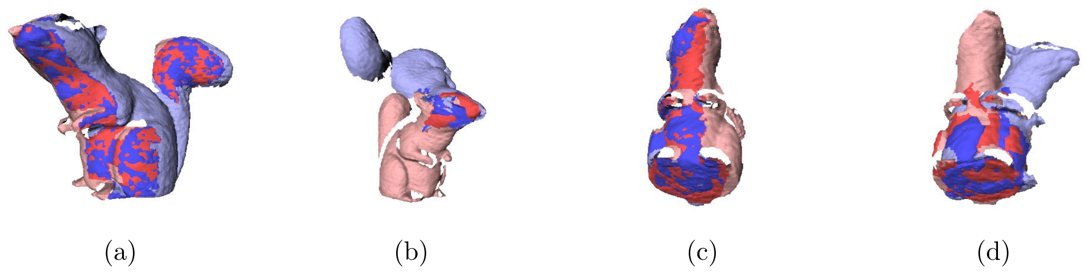


Figure 2.4: Sample pair-wise surface matching results. a) A correct match. b) An incorrect match. c/d) Multiple matches for a single pair.

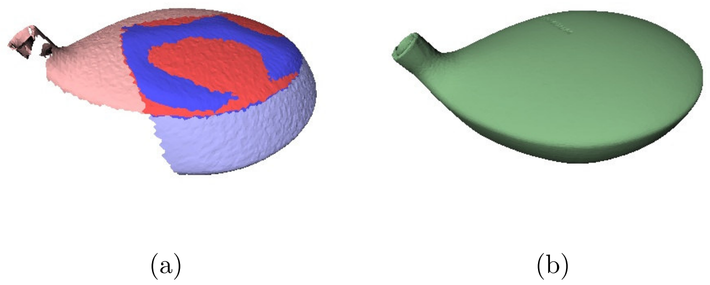


Figure 2.5: a) Example of a differentially ambiguous match from the club data set. b) The original model (shown for reference).

output a ranked list of the K best matches, rather than just the best one. Figure 2.4 illustrates pair-wise surface matching between several pairs of views.

In terms of the model graph, pair-wise surface matching adds edges to the graph. For example, the match between views 1 and 3 shown in figure 2.4a is inserted into the graph as an edge between nodes 1 and 3 in the graph and is annotated with the relative pose and registration quality of the match. Multiple matches between two views appear as multiple edges between two nodes. For some surfaces, such as a plane or a sphere, the number of possible alignments could be infinite. We call such an underconstrained match *differentially ambiguous* (figure 2.5). Currently, we do not explicitly detect differentially ambiguous matches or handle them specially, but the problem is discussed further in section 9.2.1.

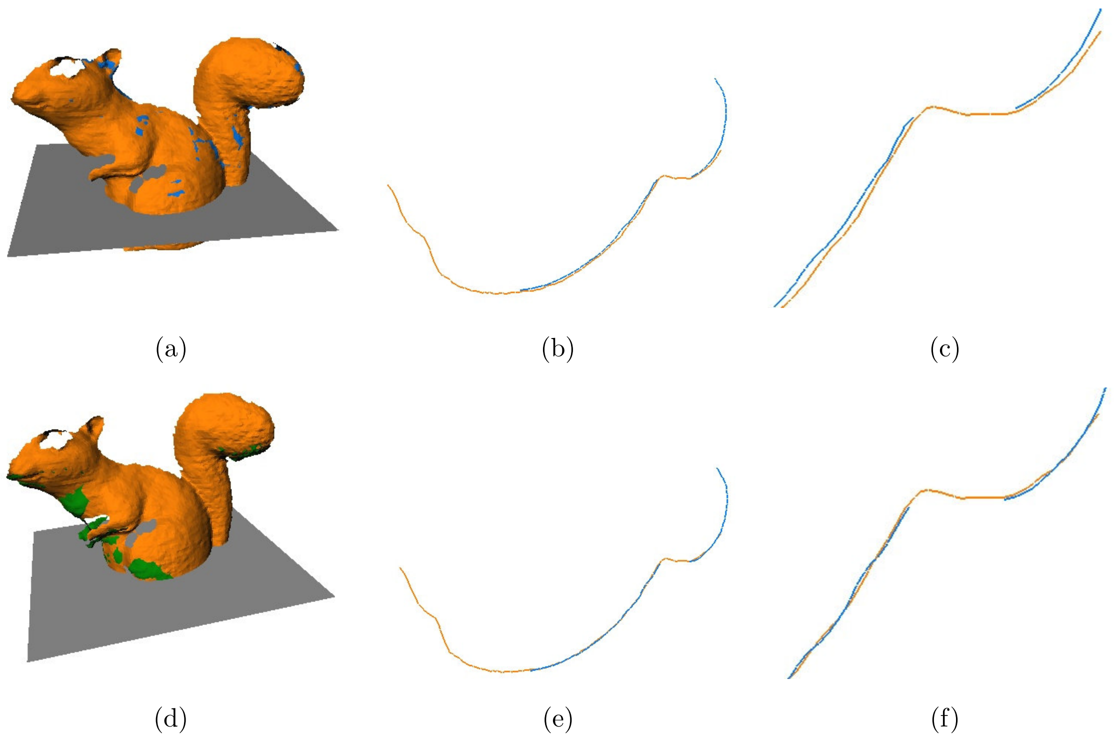
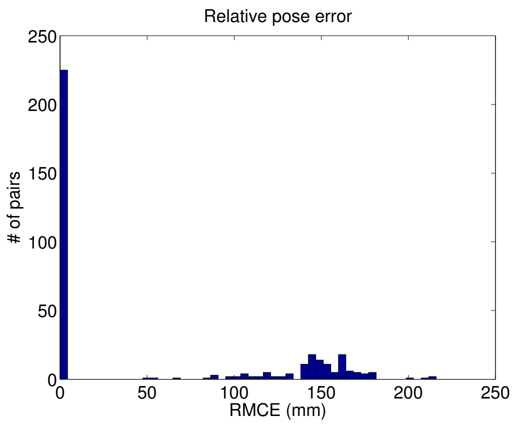


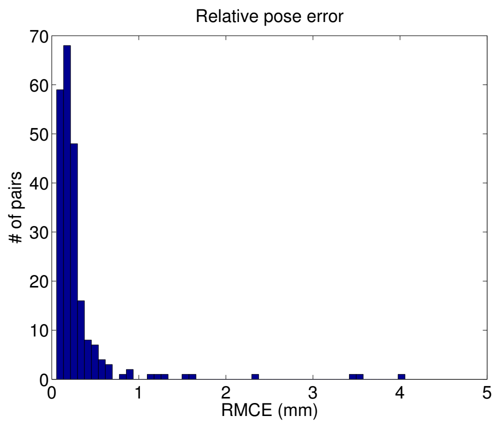
Figure 2.6: A pair of views before (top row) and after (bottom row) pair-wise registration. a/d) 3D visualization of the views. The plane shows the location of the cross-section in (b/e). c/f) A close-up of the cross-section in (b/e) that clearly shows the improvement in alignment.

2.5.2 Pair-wise registration

Pair-wise registration aligns two surfaces provided that a good initial estimate of the relative pose is known. In contrast to pair-wise surface matching, which performs a global search, pair-wise registration only searches locally from its initial state, typically in a greedy fashion. A pair-wise registration algorithm takes two views and a relative pose as input and outputs a new relative pose that improves the alignment between the two views according to a measure of registration error (e.g., the squared distance between closest surface points). Typically pair-wise registration is formulated as an optimization problem in which this registration error is minimized as a function of the relative pose. The algorithm usually finds a relative pose that is closer to its true value than the initial pose, though such a



(a)



(b)

Figure 2.7: a) The distribution of relative pose error (equation 6.5) for the 358 pair-wise matches found for the widget test set. The distribution consists of two distinct groups: one for correct matches, ranging from 0 to 5mm; and one for incorrect matches, ranging from about 50 to about 200mm. b) A close-up of the region containing correct matches shows that a few ambiguous matches fall in between these two groups.

result cannot be guaranteed. Figure 2.6 shows the effect of pair-wise registration on two views using the relative pose from pair-wise surface matching for initialization. In terms of the model graph, pair-wise registration updates the relative pose for a single edge.

2.5.3 Justification for labeling matches "correct" or "incorrect"

Up to now, we have been labeling pair-wise matches as "correct" or "incorrect" implicitly discounting the fact that the relative poses are continuous-valued parameters. There is, however, a reason why relative poses are predominantly either very close to their true value (i.e., a *correct match*) or very far off (i.e., an *incorrect match*). This binary quality is a consequence of the fact that we always follow pair-wise surface matching with pair-wise registration in order to improve the match. The *region of convergence* of a pair-wise registration algorithm is the set of initial relative poses from which the algorithm converges to the minimum of its objective function closest in pose space to the ground truth solution.[4]

[4]See section 6.1.1 for our definition of distance in pose space.

The size and shape of this region depends on the algorithm and the surfaces being registered, but in practice, it can be quite large. If surface matching outputs a relative pose anywhere within the region of convergence, the resulting match, after pair-wise registration, will be correct. Likewise, if the starting relative pose is outside the region, the final match will be incorrect. The large size of the region of convergence ensures that incorrect matches will be far away from correct matches in pose space.

In a small percentage of matches, the convergence properties near the correct solution are not so clear cut. There may be multiple local minima in the objective function near the correct minimum, or the objective function may be nearly flat or valley-shaped in the vicinity, a symptom that the match is differentially ambiguous.

2.5.4 Multi-view registration

Multi-view registration is a generalization of pair-wise registration. Such algorithms align two or more surfaces in a common coordinate system provided that good initial estimates of the poses are known. The input is a set of N views and (depending on the algorithm) either an absolute pose for each view or a set of relative poses. The output is an improved set of absolute poses according to some measure of global registration error.

Multi-view registration is motivated by the fact that relative pose estimates always contain a small amount of error. Naively composing a long sequence of relative poses leads to an accumulation of error (figure 2.8a and b). If the view at the end of this sequence overlaps the view at the beginning, the accumulated error will appear visually as an unexpected discontinuity between the overlapping surfaces. Given an independent estimate of the relative pose between the first and last view (e.g., from pair-wise surface matching), multi-view registration will eliminate the gap by distributing the accumulated error across the entire sequence of relative poses in a principled way (figure 2.8c and d).

In terms of the model graph, multi-view registration updates the relative poses of all edges and sets the absolute pose of each view. Multi-view registration imposes pose consistency on a model graph that contains cycles (since it computes absolute poses). Enforcing

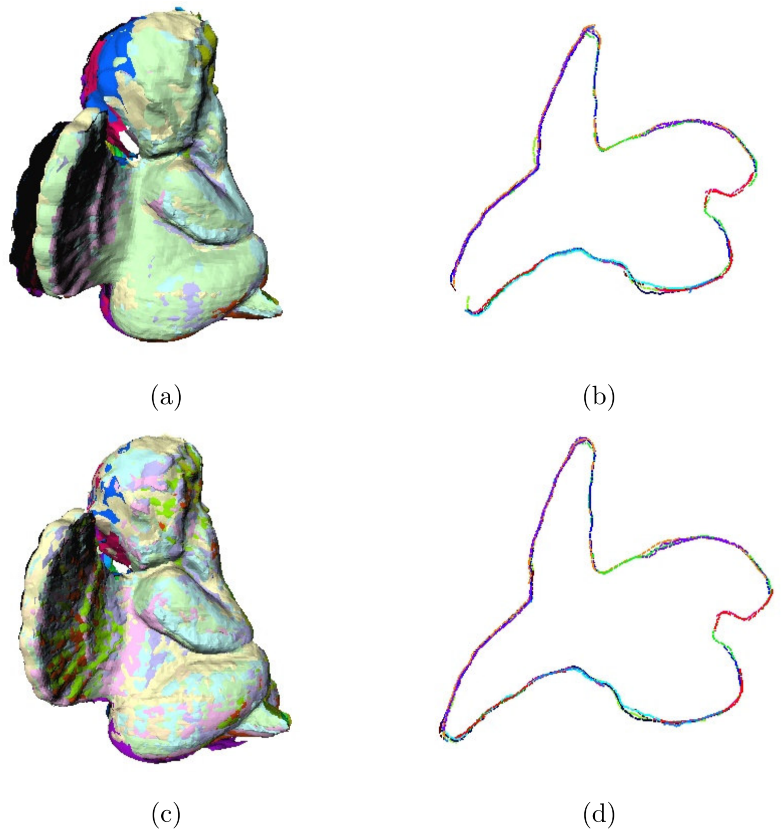


Figure 2.8: Accumulating error from a sequence of pair-wise matches leaves a visible gap in the surface, which can be eliminated through multi-view registration. a) A 3D view of the angel1 model before multi-view registration with a large gap on the angel's wing (left side of figure). b) A horizontal cross-section of the object. The gap is visible in the bottom-left corner. c) A 3D view of the model after multi-view registration. d) The corresponding cross-section shows the gap has been eliminated.

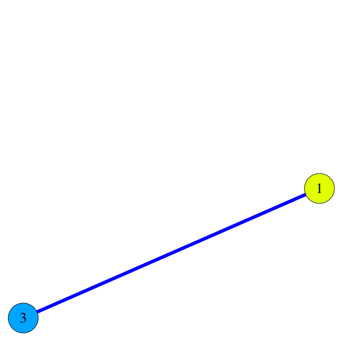
the pose consistency constraint implies that the pair-wise error for some views may actually increase. In a graph with no cycles, multi-view registration is equivalent to performing pair-wise registration independently on each pair of adjacent views.

As with pair-wise registration, multi-view registration performs local search in the vicinity of the initial pose estimates. These algorithms are typically formulated as an optimization problem similar to that of pair-wise registration, except that the cost function includes terms for all overlapping views and is minimized in terms of the absolute poses rather than relative poses. If the input model graph contains ambiguous matches, it is possible that other correct matches will draw an ambiguous match toward its correct value. However, because it performs local search, multi-view registration is unlikely to converge to the correct solution if the input graph contains incorrect matches.

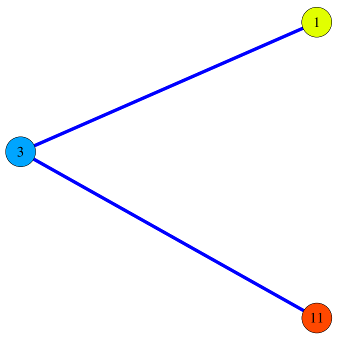
2.6 Topological inference

Topological inference is the process of inferring new constraints between non-adjacent views in a model graph. The idea is based on Sawhney's topology inference [69], which is an image-based geometric inference process used in 2D mosaicing. As an example of topological inference in 3D, consider the model graph in figure 2.3a. Pair-wise surface matching found a match between V_1 and V_3 and between V_3 and V_{11} but not between V_1 and V_{11} (figure 2.9. By composing transforms for the sequence, we discover that V_1 and V_{11} overlap, thereby establishing an additional edge in the model graph. In this case, the quality of the inferred match between V_1 and V_{11} is high, evidence that the matches along the path $\langle(1,3),(3,11)\rangle$ are correct. On the other hand, if the quality of the inferred match had been low, it would suggest that one of the matches along the path is incorrect, though we cannot tell which one.

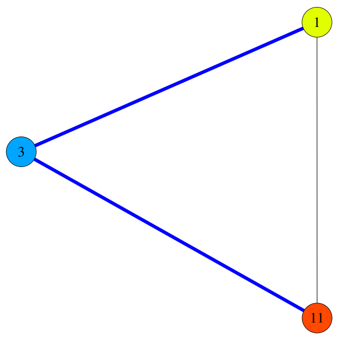
For multi-view surface matching, we use topological inference for two purposes. First, whenever we perform multi-view registration, we augment the model hypothesis, using topological inference to find all overlapping view pairs. In terms of the model graph, topological inference inserts new edges between view pairs, establishing the graph cycles that



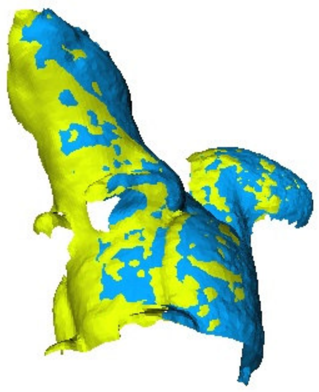
(a)



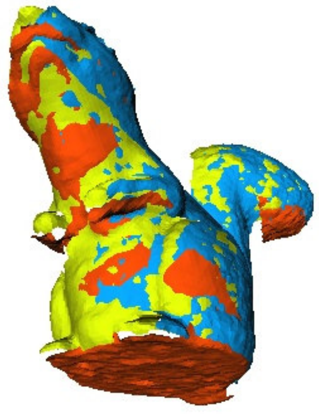
(b)



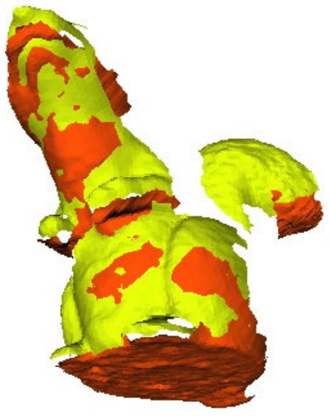
(c)



(d)



(e)



(f)

Figure 2.9: Topological inference example. Pair-wise surface matching found correct matches between views 1 and 3 (a and d) and between views 3 and 11 but not between views 1 and 11. By considering the topology of the combined sequence of views (b and e), we find that views 1 and 11 overlap, thereby establishing an additional model graph edge (c and f).

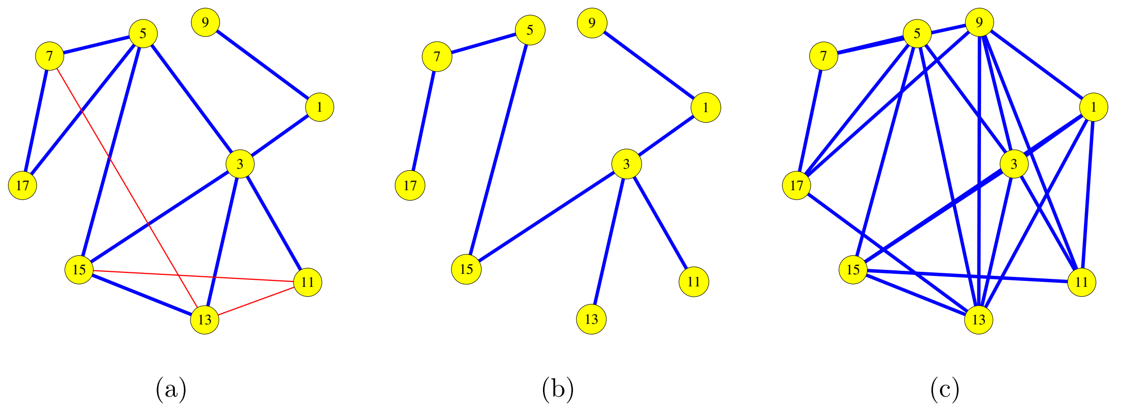


Figure 2.10: Edges inferred by topological inference. a) G_{LR} for the views in figure 2.1. b) An example model hypothesis from G_{LR}. c) The hypothesis in (b) augmented with the links found by topological inference. Topological inference creates the cycles necessary for multi-view registration to be effective and creates links equivalent to all the correct matches from (a) as well as many additional correct matches that were not found by pair-wise surface matching.

are necessary for multi-view registration to be effective (figure 2.10).

Second, topological inference is implicit in our global quality measure Q_{G} and our global consistency test C_{G}. When computing these quantities for a model hypothesis, we use all connected pairs of views, not just the adjacent ones. For the global consistency test, this allows us to eliminate hypotheses that contain locally consistent, but incorrect, matches. These processes will be described in more detail in the next two chapters.

In section 2.4, we claimed that the acyclic restriction on our hypotheses reduced the size of the hypothesis space without eliminating correct solutions. We can now justify this claim with the help of topological inference. Consider the model hypothesis in figure 2.10. Even though the hypothesis contains only a subset of the correct matches from G_{LR}, when we perform topological inference, *all* of the correct matches from G_{LR} are re-established in the augmented graph. This effect is one of the advantages of using topological inference before performing multi-view registration – it reinstates the edges from pair-wise registration that are compatible with a given model hypothesis without having to explicitly classify edges as compatible or incompatible. Of course, topological inference also finds other overlapping pairs that were not found with pair-wise surface matching.

Chapter 3

Evaluating model quality

This chapter addresses the question, “What is a good model hypothesis?” Informally, we would like a hypothesis to connect all the input views, but only if this can be accomplished using only correct matches. More formally, given a model hypothesis H, we want to define a function $Q_{\mathrm{G}}(H)$ such that the hypothesis that maximizes Q_{G} corresponds to the correct solution to the multi-view surface matching problem. Rather than attack this problem directly, we first look at the simpler problem of computing the local quality Q_{L} of a pair of registered views and then use this result to construct a global quality measure for model hypotheses.

We begin in section 3.1 by presenting a maximum likelihood framework for learning local quality measures from training data. We show that this approach has significant advantages over traditional measures of registration quality. We then use this framework to derive four local quality measures: the overlap quality measure (section 3.2) and three measures based on measurements along the sensor’s line of sight (section 3.3). Then, in section 3.4, we derive several global quality measures for evaluating model hypotheses. Finally, in section 3.5, we show how a thresholded version of local quality can be used as a local consistency test to eliminate the worst pair-wise matches and as part of a global consistency test to eliminate the worst model hypotheses.

3.1 Maximum likelihood local quality framework

The concept of local quality is related to the verification process present in most 3D object recognition systems. At some point, such a system must decide whether the hypothesized match is "good enough." Typically, this is done by thresholding some statistic of the data or of features derived from the data. The question is what statistic/features should be used and how to set the threshold. The answers depend on the sensor and the scene characteristics. For example, two surfaces that are ten centimeters apart might indicate a good registration for terrain observed from twenty meters away but not for a desktop object seen from one meter.

Rather than relying on fixed, user-defined thresholds, which tend to be brittle, we explicitly model the behavior of the system for a given sensor and scene type, using supervised learning to determine a statistical model of local quality from training data. The benefit of this approach is that we do not need detailed knowledge of the sensor or even of the surface matching and pair-wise registration algorithms. Although we present this method in the context of multi-view surface matching, the idea has broad applicability. For example, a 3D recognition algorithm could employ an analogous model of registration quality to reduce its false positive rate.

In the next two sections, we derive several local quality measures that differ in the details but share the same high-level framework based on statistical pattern recognition [6]. For now, we assume that the pair of views under consideration is the result of pair-wise surface matching. In section 3.4, we will explore the effects of relaxing this assumption. Our approach is to estimate the probability that a match is correct based on a vector of features derived from the registered surfaces. We denote the event that the match is correct by M^+, the event that it is incorrect by M^-, and the vector of derived features by $\boldsymbol{x}$. With this terminology, $P(M^+|\boldsymbol{x})$ is the posterior probability of a correct match given the observed features, and $P(M^-|\boldsymbol{x})$ is that of an incorrect match. We define the quality of a match between surfaces S_i and S_j with relative pose T_{ij} to be the log odds ratio of the posterior probabilities:

$$\begin{aligned} Q_{\mathrm{L}}(S_i, S_j, T_{ij}) &= \log\left(\frac{P(M^+|\boldsymbol{x})}{1 - P(M^+|\boldsymbol{x})}\right) = \log\left(\frac{P(M^+|\boldsymbol{x})}{P(M^-|\boldsymbol{x})}\right) \\ &= \log\left(\frac{P(\boldsymbol{x}|M^+)P(M^+)}{P(\boldsymbol{x}|M^-)P(M^-)}\right), \end{aligned} \tag{3.1}$$

where the last step follows from Bayes' rule. The form of $\boldsymbol{x}$, and the method by which it is computed from S_i, S_j, and T_{ij}, depend on the specific quality measure and will be detailed in the next few sections.

We estimate the distributions $P(\boldsymbol{x}|M^+)$ and $P(\boldsymbol{x}|M^-)$ and the prior probabilities $P(M^+)$ and $P(M^-)$ from labeled training data. The priors are estimated directly from the frequency of M^+ and M^- in the data. We model the conditional distributions using parametric density estimation, first choosing a parametric form for the distributions and then computing the maximum likelihood parameters. We model our features using the Gamma distribution ($\texttt{Gamma}(x|\alpha, \beta)$. The Gamma distribution has a flexibly-shaped probability density function (PDF) that encompasses the exponential distribution and can approximate a Gaussian. Our motivation for using parametric methods is primarily the compactness of the representation. If our distributions were not well-represented by a standard distribution function, non-parametric methods, such as kernel density estimation, could be used instead. One disadvantage to the Gamma distribution is that the PDF for $x = 0$ is 0, which can potentially cause problems when fitting distributions with significant mass near $x = 0$. A solution to this problem is to use a generalized form of the Gamma distribution, $\texttt{Gamma}(x|\alpha, \beta, \gamma)$. The extra parameter shifts the PDF, allowing it to take on non-zero probability for $x > \gamma$. In practice, we do not see a significant difference when using this generalized form.

Using this framework, we have derived several local quality measures. The process is demonstrated using a training set of 443 matches (153 correct and 290 incorrect) obtained by exhaustive pair-wise registration on three real data sets (angel4, y1, and eagle) using the Vivid sensor. The results are illustrated and evaluated using a separate test set of 436 matches (189 correct and 247 incorrect) from exhaustive pair-wise matching for three different real data sets (angel1, squirrel, and dwarf).

3.2 The overlap local quality measure

Our first measure is the overlap local quality measure, which is based on two features: overlap fraction and overlap distance. Intuitively, surfaces that overlap significantly and are close wherever they overlap should have high quality.

Overlap fraction, F_{OV}, is the maximum proportion of each surface that lies in the overlapping region. Let R_i be the overlapping region of surface S_i with $T_{ij}S_j$ and R_j be the overlapping region of $T_{ij}S_j$ with S_i. The overlap fraction is

$$F_{\text{OV}} = \max\left(\frac{A(R_i)}{A(S_i)}, \frac{A(R_j)}{A(S_j)}\right), \tag{3.2}$$

where $A(S)$ denotes the surface area of S. Using the maximum ensures that if one surface is a subset of the other, F_{OV} will be 1, which coincides with our intended meaning of overlap fraction. Overlap distance, D_{OV}, is the root mean squared (RMS) distance between a set of closest point pairs in the overlapping region. This is essentially the same as the error measure used in the ICP algorithm (equation 5.2); however, in the ICP algorithm, the error was only computed in one direction, while here, we compute distances in both directions.

To compute F_{OV} and D_{OV}, we first sample K points, $\boldsymbol{p}_{ik}$, from S_i and an additional K points, $\boldsymbol{p}_{jk}$, from S_j. We then determine whether each sample point overlaps the other surface using the overlap definition (section 2.3). Let u_{ik} be the overlap indicator variable for the points $\boldsymbol{p}_{ik}$ ($u_{ik} = 1$ if $\boldsymbol{p}_{ik}$ overlaps S_j and $u_{ik} = 0$ otherwise) and likewise u_{jk} for the points $\boldsymbol{p}_{jk}$. Let N_i and N_j be the number of overlapping points from $\boldsymbol{p}_{ik}$ and $\boldsymbol{p}_{jk}$ respectively (i.e., $N_i = \sum_{k=1}^{K} u_{ik}$ and $N_j = \sum_{k=1}^{K} u_{jk}$). Then equation 3.2 is approximated by

$$F_{\text{OV}} \approx \max\left(\frac{N_i}{K}, \frac{N_j}{K}\right) \tag{3.3}$$

If we let $d_{ik} = u_{ik}D_{\text{CP}}(T_{ji}\boldsymbol{p}_{ik}, S_j)$ and $d_{jk} = u_{jk}D_{\text{CP}}(T_{ij}\boldsymbol{p}_{jk}, S_i)$, then the overlap distance can be computed by

$$D_{\text{OV}} = \sqrt{\frac{\sum_{k=1}^{K}(d_{ik}^2 + d_{jk}^2)}{N_i + N_j}} \tag{3.4}$$

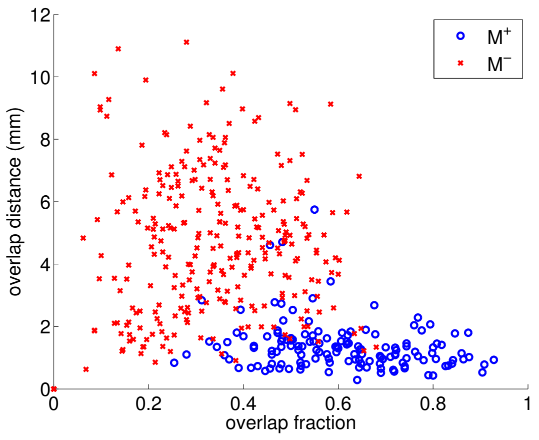


Figure 3.1: The joint distribution for overlap distance (D_{OV}) and overlap fraction (F_{OV}) for correct matches (M^{+}) and incorrect matches (M^{-}) in the training set.

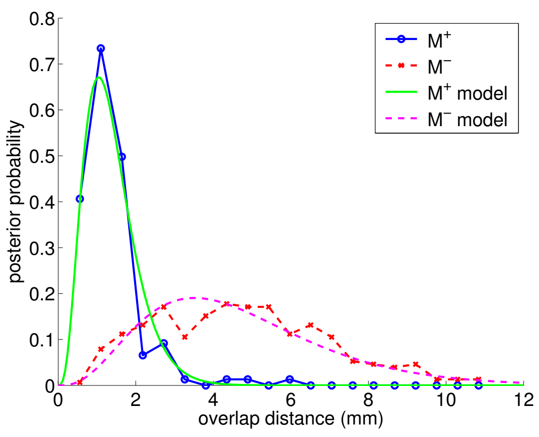
If F_{OV} is zero or if there are insufficient overlapping sample points to reliably compute D_{OV}, we leave Q_{L} undefined.

Finally, we estimate the joint distributions $P(D_{\mathrm{OV}}, F_{\mathrm{OV}}|M^{+})$ and $P(D_{\mathrm{OV}}, F_{\mathrm{OV}}|M^{-})$. Figure 3.1 shows these distributions for the training data. Empirically, correct matches tend to have a larger overlap fraction and a smaller overlap distance than incorrect matches, but within each class, F_{OV} and D_{OV} are nearly uncorrelated. Therefore, we make the simplifying assumption that F_{OV} and D_{OV} are conditionally independent given the class label M, which means the joint distributions can be factored:

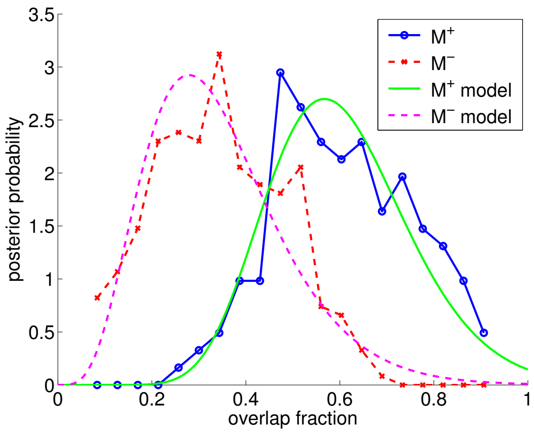
$$P(D_{\mathrm{OV}}, F_{\mathrm{OV}}|M^{+}) = P(D_{\mathrm{OV}}|M^{+})P(F_{\mathrm{OV}}|M^{+}) \tag{3.5a}$$

$$P(D_{\mathrm{OV}}, F_{\mathrm{OV}}|M^{-}) = P(D_{\mathrm{OV}}|M^{-})P(F_{\mathrm{OV}}|M^{-}) \tag{3.5b}$$

Histograms of the factored distributions are shown in figure 3.2. We model each factored distribution with a separate Gamma distribution, $\texttt{Gamma}(\alpha, \beta)$, computing the maximum likelihood parameters, $\hat{\alpha}$ and $\hat{\beta}$, from the training data. The resulting PDFs are shown overlaid in figure 3.2. We estimate the priors, $P(M^{+})$ and $P(M^{-})$ directly from the number of correct and incorrect matches in the training data. Inserting the relevant probabilities



(a)



(b)

Figure 3.2: Marginal distributions for overlap distance (D_{OV}) and overlap fraction (F_{OV}) for correct (M^+) and incorrect matches (M^-) in the training set. The maximum likelihood Gamma distributions, which serve as our parametric models for each class, are shown overlaid.

into equation 3.1, we obtain the overlap local quality measure:

$$Q_{\mathrm{L}} = \log \left(\frac{P(D_{\mathrm{OV}}|M^+)P(F_{\mathrm{OV}}|M^+)P(M^+)}{P(D_{\mathrm{OV}}|M^-)P(F_{\mathrm{OV}}|M^-)P(M^-)} \right) \tag{3.6}$$

Figure 3.3 shows the distribution of Q_{L} for a test data. The wider the separation between the two distributions, the more discriminating the quality measure.

Using our probabilistic framework for local quality, it is easy to combine several independent features into a single quality measure, since the probabilities are just multiplied. Moreover, features with different units can be integrated using this framework. Here, we combine overlap fraction, a unitless feature, with overlap distance, in millimeters. More disparate features, such as color or texture similarity, could also be employed using the same techniques.

3.3 Visibility-based quality measures

The main problem with the overlap local quality is that it only considers a thin envelope of space close to the two surfaces. In some cases, obviously incorrect matches will have a high

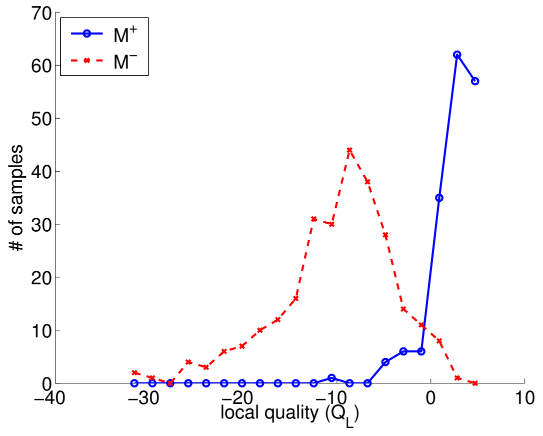


Figure 3.3: A two-class histogram showing the distribution of the overlap quality measure for correct (M^+) and incorrect matches (M^-) in the test set.

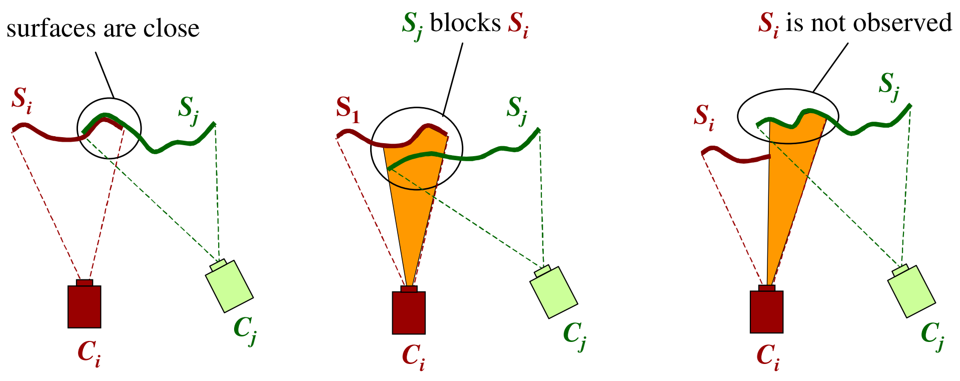


Figure 3.4: A 2D example of visibility consistency concept (from the perspective of C_i. a) If two surfaces are correctly registered, they should be close wherever they overlap (circled region). If the surfaces are incorrectly registered, two situations can occur: b) A free space violation (FSV) occurs in regions where S_j blocks the visibility of S_i from C_i (circled region). c) An occupied space violation (OSV) occurs in regions where S_j is not observed from C_i even though it is expected to be.

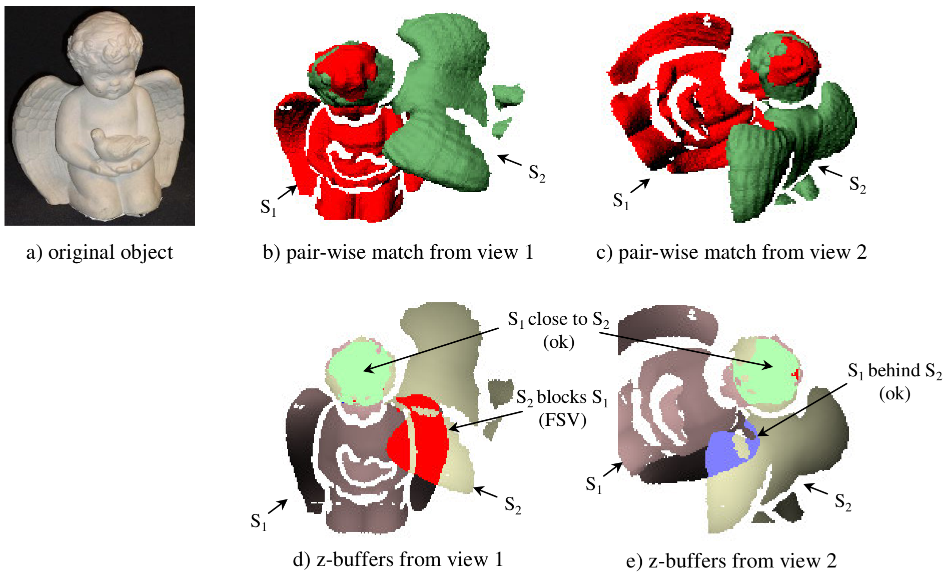


Figure 3.5: An example of visibility consistency for an incorrect match. The indicated FSV pixels in the z-buffer for view 1 suggest an incorrect match.

Q_L simply because the overlapping regions have similar shapes.

For surfaces derived from range sensors, we can develop more powerful quality measures that take advantage of the sensor's entire viewing volume, using measurements along the line of sight from each of the sensor viewpoints. We call this concept *visibility consistency.* For example, consider the registered surfaces in figure 3.4 viewed from the sensor position C_i. For a correct match, the two surfaces have similar range values wherever they overlap (figure 3.4a). For an incorrect match, two types of inconsistencies can arise. A *free space violation* (FSV) occurs when a region of S_j blocks the visibility of S_i from C_i (figure 3.4b/d). An *occupied space violation* (OSV) occurs when a region of S_j is not observed by C_i, even though it should be (figure 3.4c/e). Free space violations are so named because the blocking surface violates the assumption that the space is free along the line of sight from the sensor to the sensed surface. Similarly, OSVs violate the assumption that range sensor detects occupied space. The idea of visibility consistency is not new. It has been used previously in other 3D vision contexts, including hypothesis verification [7], surface registration [20], range shadow detection [56], and surface reconstruction [65][72].

When deriving our visibility-based quality measures from the perspective of sensor position C_i, we assume the surface S_j has been transformed into the coordinate frame of C_i using T_{ij}. An independent measure can be computed from the viewpoint of C_j, in which case the surface S_i is assumed to be transformed by T_{ji}. We denote these single viewpoint quality measures Q_{L1}.

We can detect FSVs with respect to sensor position C_i by projecting a ray from the center of projection of C_i through a point p on S_i. If the ray passes through S_j at a point $\boldsymbol{q}$ which is significantly closer to C_i than $\boldsymbol{p}$, then $\boldsymbol{q}$ is an inconsistent point. We must test whether $\boldsymbol{q}$ is *significantly* closer because even for correctly registered surfaces, $\boldsymbol{p}$ and $\boldsymbol{q}$ will not have precisely the same range. Similarly, we can detect OSVs by projecting a ray from the center of projection of C_i through a point $\boldsymbol{q}$ on S_j. If the ray does not pass through S_i, $\boldsymbol{q}$ is a potential OSV. Unfortunately, even for a correct match, there are many reasons why a portion of S_j may be undetected when seen from C_i. For example, the surface could be out of the sensor's range or field of view; the laser return could be too low due to dark, specular, or obliquely viewed surfaces; or, for triangulation-based scanners, the surface may lie inside a self-shadowed region. We do not attempt to explicitly model all of these effects, but instead capture the general behavior with a probability distribution as described in section 3.3.3.

For sensors with perspective geometry, we can efficiently implement FSV and OSV detection using z-buffers [26] to synthesize range images from each camera's viewpoint. The z-buffer algorithm computes the line of sight distance from the sensor to the nearest surface, measured along the sensor's optical axis (i.e., z-value rather than range value). The algorithm is commonly used in rendering 3D computer graphics is built into modern graphics hardware. A correction factor must be applied to convert the z-value into a range value, but since this factor is constant for a given pixel and focal length, it can be precomputed for efficiency.

For sensors with non-perspective geometry, the standard z-buffer algorithm may not work. For example, the Z+F and BF2 scanners have a cylindrical geometry. If we use the standard z-buffer algorithm for this geometry, the range values would be incorrect for triangles that span a substantial azimuth angle. It is possible to modify the z-buffer algorithm to handle cylindrical geometry, but, in practice, this error is small relative to the other types of errors corrupting the data.

Using the z-buffer algorithm, we generate separate synthetic range images (R_i and R_j) for S_i and S_j from the perspective of C_i. For each pixel, k, where both range images are defined, we compute the range difference:

$$D_{i,j}(k) = R_i(k) - R_j(k) \tag{3.7}$$

Regions where $D > 0$ are potential FSVs, since the range to S_j is less than to S_i (figure 3.5d). Regions where $D < 0$ do not give us any information about visibility consistency.

We have developed three local quality measures based on visibility consistency. The first one, which we call *FSV-raw*, uses the raw depth differences in overlapping regions as feature measurements (section 3.3.1. The second measure, which we call *FSV*, is analogous to the overlap local quality measure and uses a statistic of the depth differences as well as an estimate of overlap fraction (section 3.3.2. The third measure, which we call *FSV-OSV*, incorporates a statistic for the OSV region into the FSV quality measure (section 3.3.3.

3.3.1 The FSV-raw local quality measure

For the FSV-raw measure, the range difference measurements, $\boldsymbol{D} = \{D_{i,j}(1), \ldots, D_{i,j}(K)\}$, constitute our feature vector $\boldsymbol{x}$. Using our probabilistic framework, the quality measure from sensor viewpoint C_i is

$$Q_{\mathrm{L1}}(S_i, S_j, T_{ij}) = \log\left(\frac{P(\boldsymbol{D}|M^+)P(M^+)}{P(\boldsymbol{D}|M^-)P(M^-)}\right). \tag{3.8}$$

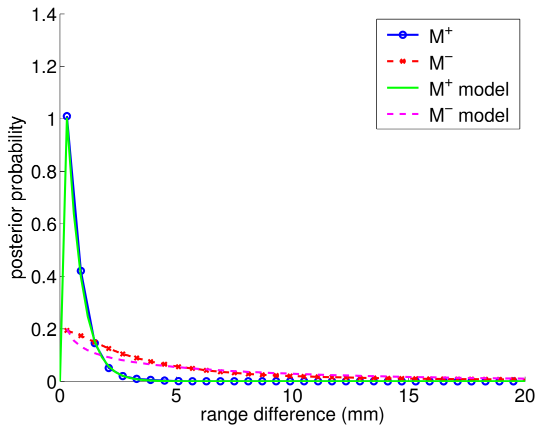


Figure 3.6: The distribution of range differences ($\boldsymbol{D}$) for the FSV-raw quality measure for correct (M^+) and incorrect matches (M^-) in the training set. The maximum likelihood Gamma distributions, which serve as our parametric models for each class, are shown overlaid.

Assuming the samples in $\boldsymbol{D}$ are independent[1], the joint probability can be factored to obtain

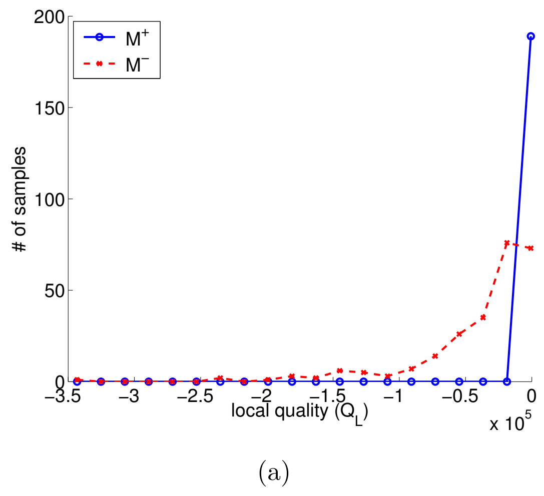
$$\begin{aligned} Q_{\mathrm{L1}} &= \log\left(\frac{\prod_{k=1}^{K} P(D_{i,j}(k)|M^+)P(M^+)}{\prod_{k=1}^{K} P(D_{i,j}(k)|M^-)P(M^-)}\right) \\ &= \sum_{k=1}^{K} \log P(D_{i,j}(k)|M^+) + \log P(M^+) - \\ &\quad \sum_{k=1}^{K} \log P(D_{i,j}(k)|M^-) - \log P(M^-) \qquad (3.9) \end{aligned}$$

We computed the range differences for the overlapping regions in the range images for the matches in the training data (figure 3.6). As expected, the range differences for correct matches tend to be smaller than for incorrect matches. The maximum likelihood Gamma distributions approximate the class posteriors well[2].

As previously mentioned, an independent quality measure, $Q_{\mathrm{L1}}(S_j, S_i, T_{ji})$, can be computed with respect to sensor viewpoint C_j. The two measures are combined by including

[1]This assumption is not strictly true, but it greatly simplifies the computations

[2]In [32], we implemented this measure using mixtures of Gaussians, but we found that the Gamma distribution provides a better fit for the data.



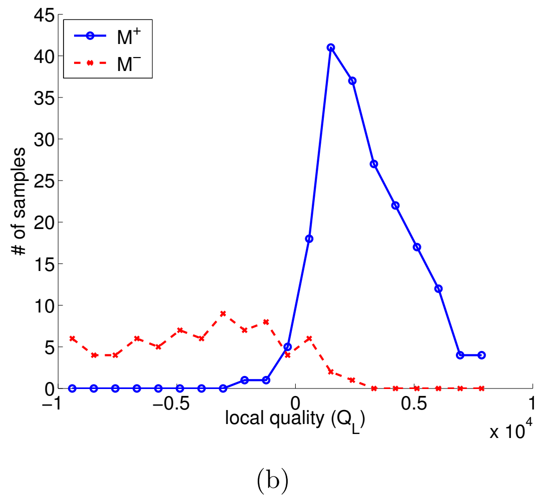


Figure 3.7: Two-class histograms showing the distribution of the FSV-raw quality measure for correct (M^+) and incorrect matches (M^-) in the test set. a) The distribution over the full range of Q_L. b) A close-up of the histogram in (a) that focuses on the region where correct and incorrect matches overlap.

these range differences in equation 3.9 to form the FSV-raw quality measure:

$$Q_\mathrm{L} = \log\left(\frac{\prod_{k=1}^{K} P(D_{i,j}(k)|M^+)\prod_{k=1}^{K} P(D_{j,i}(k)|M^+)P(M^+)}{\prod_{k=1}^{K} P(D_{i,j}(k)|M^-)\prod_{k=1}^{K} P(D_{j,i}(k)|M^-)P(M^-)}\right), \tag{3.10}$$

which can be written as

$$Q_\mathrm{L} = Q_\mathrm{L1}(S_i, S_j, T_{ij}) + Q_\mathrm{L1}(S_j, S_i, T_{ji}) - C, \tag{3.11}$$

where $C = \log(P(M^+) - \log(P(M^-)$ compensates for fact that the priors appear in both Q_L1 terms and are therefore double counted. Figure 3.7 shows the distribution of Q_L for the test data.

3.3.2 The FSV local quality measure

The FSV local quality measure is analogous to the overlap quality measure. Instead of computing the overlap fraction and distance in 3D, these features are computed along the sensor's line of sight. For this measure, the feature vector $\boldsymbol{x}$ consists of two features: range overlap fraction (F_ROV) and FSV distance (D_FSV).

We define F_{ROV} for surfaces S_i and S_j observed from sensor viewpoint C_i to be the ratio of the visible overlapping area (A_{OV}) of the two surfaces to the visible surface area of S_i ($A(S_i)$):

$$F_{\text{ROV}} = \frac{A_{\text{OV}}}{A(S_i)} \tag{3.12}$$

To compute A_{OV} and $A(S_i)$, we cannot simply count pixels in the range images. If we did, the surfaces in close regions would have more influence on the estimate, since the surface area corresponding to a range image pixel increases with the square of the range. To compensate for this bias, we weight each pixel by its estimated surface area. This estimate can be computed from the z-buffer depth values and the intrinsic parameters $\mathcal{I}$ for V_i:

$$A_z(S_i, k) = \frac{wh(z(k))^2}{f^2} \tag{3.13}$$

where w is the pixel width, h is the pixel height, $z(k)$ is the z-value of pixel k in the z-buffer for S_i, and f is the focal length. Using this result, we define the visible surface area ($A(S_i)$) as

$$A(S_i) \approx \sum_{k \in K_i} A_z(S_i, k), \tag{3.14}$$

where K_i is the set of defined pixels in R_i (i.e., those corresponding to S_i). Similarly, we define the visible overlap area (A_{OV}) as

$$A_{\text{OV}} = \sum_{k \in K_{ij}} A_z(S_i, k) \tag{3.15}$$

where K_{ij} is the set of pixels such that $D_{i,j}(k) > 0$.

We use a similar approach for computing the FSV distance (D_{FSV}), which is the area-weighted RMS range difference in the overlapping region.

$$D_{\text{FSV}} = \sqrt{\sum_{k \in K_{ij}} w_k (D_{i,j}(k))^2} \tag{3.16}$$

where the weight $w_k = A_z(k)/A_{\text{OV}}$.

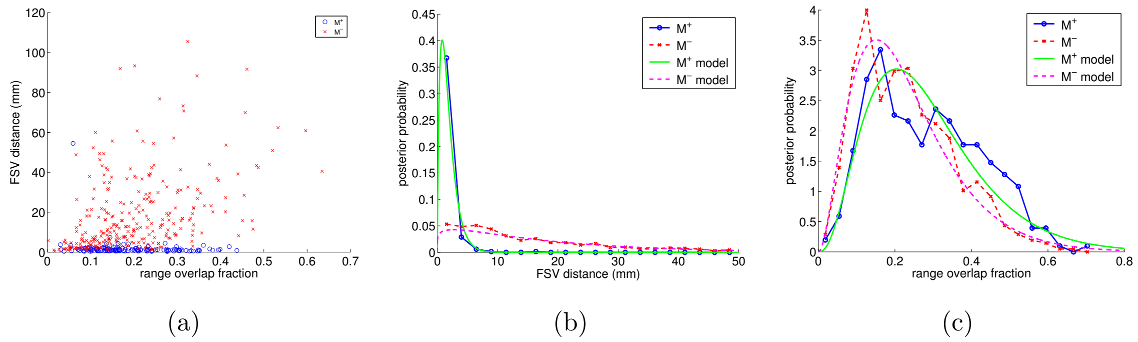


Figure 3.8: Distributions for the FSV local quality measure. a) The joint distribution for FSV distance (D_{FSV}) and range overlap fraction (F_{ROV}) for correct (M^+) and incorrect matches (M^-) in the training set. b/c) The marginal distributions for D_{FSV} and F_{ROV} with the maximum likelihood model shown overlaid. Correct matches tend to have much smaller D_{FSV} and slightly larger F_{ROV} than incorrect matches.

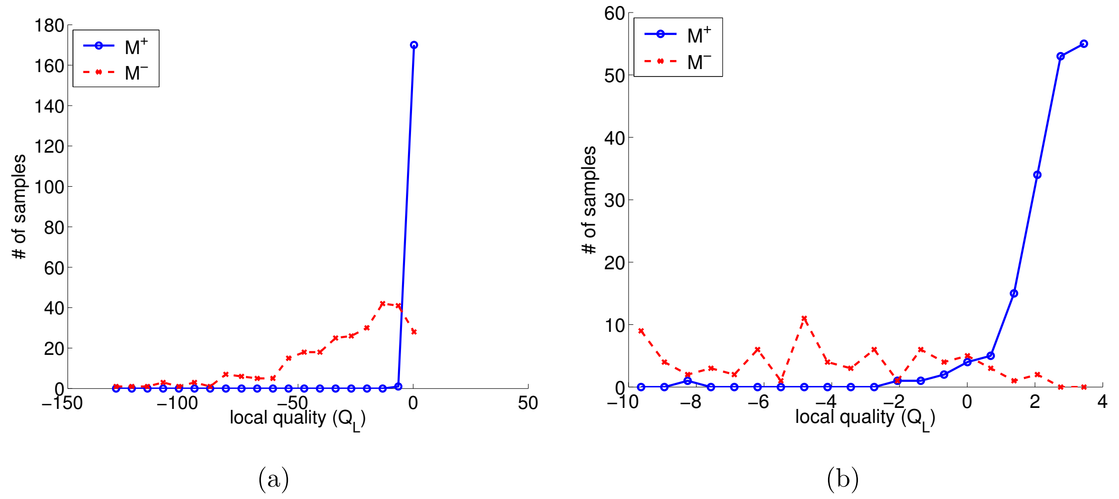


Figure 3.9: Two-class histograms showing the distribution of the FSV quality measure for correct (M^+) and incorrect matches (M^-) in the test set. a) The distribution over the full range of Q_{L}. b) A close-up of the histogram in (a) that focuses on the region where correct and incorrect matches overlap.

Figure 3.8a shows the joint distribution of these two features for the matches in our training set. If F_{ROV} is zero or if there are insufficient overlapping pixels to reliably compute D_{FSV}, we leave Q_{L} undefined.

The rest of the derivation of the FSV quality measure follows the same procedure used in the overlap quality measure derivation: assume independence of the features, factor the distributions, and model them using Gamma distributions. The factored distributions and maximum likelihood models are shown in figure 3.8b and c. The resulting quality measure is

$$Q_{\text{L1}}(S_i, S_j, T_{ij}) = \log\left(\frac{P(D_{\text{FSV}}|M^+)P(F_{\text{ROV}}|M^+)P(M^+)}{P(D_{\text{FSV}}|M^-)P(F_{\text{ROV}}|M^-)P(M^-)}\right) \tag{3.17}$$

As with the FSV-raw measure, an independent quality measure ($Q_{\text{L1}}(S_j, Si, T_{ji})$) can be computed from the sensor viewpoint C_j, and the results combined as in equation 3.11. Figure 3.9 shows the distribution of Q_{L} for the test data.

3.3.3 The FSV-OSV local quality measure

For our final local quality measure, FSV-OSV, we incorporate a feature based on OSVs into the FSV quality measure. We use a simple method to model OSVs from sensor viewpoint C_i. Specifically, any pixel that is defined in R_j and undefined in R_i is considered to be an OSV unless it falls outside the sensor's viewing volume. The boundaries of the range image R_i implicitly prevent any region of S_j outside of the sensor's field of view from being counted as OSVs. Since our sensors do not provide the near and far limits of their sensing range (and these can change from scan to scan), we estimate these limits from the observed minimum and maximum range in R_i.

As previously mentioned, there are a number of reasons that a portion of S_j might not be observed from viewpoint C_i, and, consequently, even correct matches are expected to have some OSVs. However, incorrect matches often generate significantly more OSVs. With this in mind, we create a feature called the *OSV fraction* (F_{OSV}), which we define in a manner similar to the range overlap fraction. First, we estimate the area of the OSV

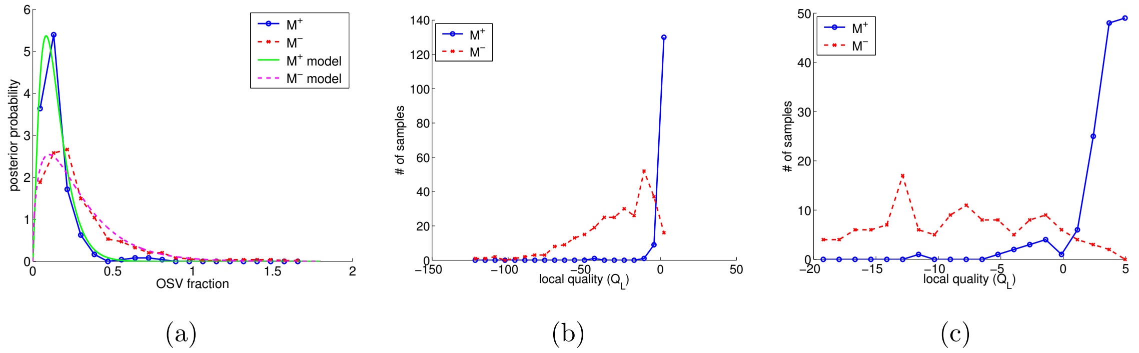


Figure 3.10: Distributions for the FSV-OSV quality measure. a) The marginal distribution for the OSV fraction (F_{OSV}) for correct (M^+) and incorrect matches (M^-) in the training set. The maximum likelihood model for each class is shown overlaid. Correct matches typically have smaller F_{OSV} than incorrect matches. b) Two-class histogram showing the distribution of the FSV-OSV quality measure for correct and incorrect matches in the test set. c) A close-up of the histogram in (b) that focuses on the region where correct and incorrect matches overlap.

pixels:

$$A_{\mathrm{OSV}} = \sum_{k \in K_{OSV}} A_z(S_j, k) \tag{3.18}$$

where K_{OSV} is the set of of OSV pixels from range image R_j. The OSV fraction is then defined as

$$F_{\mathrm{OSV}} = \frac{A_{\mathrm{OSV}}}{A_{\mathrm{OV}}} \tag{3.19}$$

Figure 3.10a shows the distribution of F_{OSV} for the training data. As expected, a higher proportion of OSVs is observed for incorrect matches. Combining F_{OSV} with the FSV quality measure gives us the FSV-OSV quality measure:

$$Q_{\mathrm{L1}} = \log\left(\frac{P(D_{\mathrm{FSV}}|M^+)P(F_{\mathrm{ROV}}|M^+)P(F_{\mathrm{OSV}}|M^+)P(M^+)}{P(D_{\mathrm{FSV}}|M^-)P(F_{\mathrm{ROV}}|M^-)P(F_{\mathrm{OSV}}|M^-)P(M^-)}\right) \tag{3.20}$$

As with the FSV-raw measure, the single viewpoint Q_{L1} measures are combined using equation 3.11. Figure 3.10b and c show the distribution of Q_{L} for the test data.

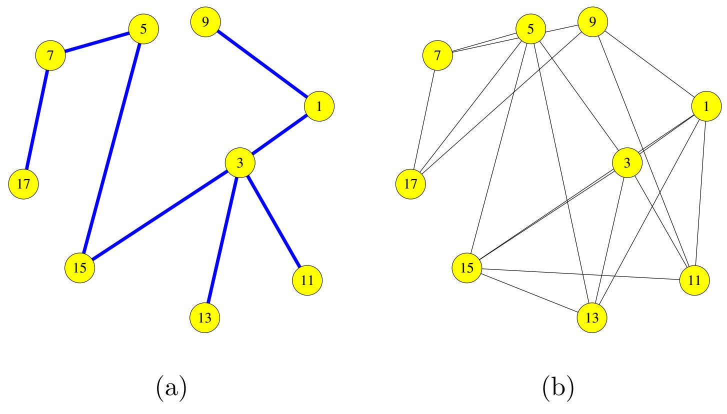


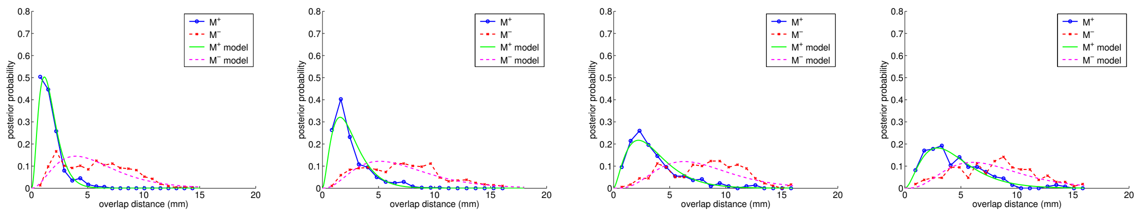
Figure 3.11: Non-adjacent view pairs used in the global quality computation. a) A model hypothesis for the views from figure 2.1. b) The edges in this model graph represent the view pairs for which Q_{L} contributes to the computation of Q_{G}.

3.4 From local quality to global quality

Now that we have a method for computing local quality for a pair of views, we can extend this method to handle an entire model graph G. Assuming the edges in G arise from surface matching and that the local quality measures of the matches are independent, we could define model quality as the sum of the local quality measures. However, this ignores the information encoded in non-adjacent views. Instead, we compute global quality by summing the local quality measured between all connected views:

$$Q_{\mathrm{G}}(G) = \sum_{(i,j)\in E_c} Q_{\mathrm{L}}(V_i, V_j, T_{ij}) \tag{3.21}$$

where E_c the set of connected (not necessarily adjacent) view pairs in G. We assume that G is pose consistent. This assumption ensures that the relative pose for each view pair is well-defined. When computing Q_{G} for a model hypothesis that contains hidden edges, the hidden edges are first deleted (i.e., quality is computed on a part by part basis). We call this global quality measure `sum_local` because it is the sum of local quality measures. Each of the local quality measures derived earlier in this chapter (overlap, FSV-raw, FSV, and FSV-OSV) has a corresponding `sum_local` global quality measure.



path length 2 path length 4 path length 6 path length 8

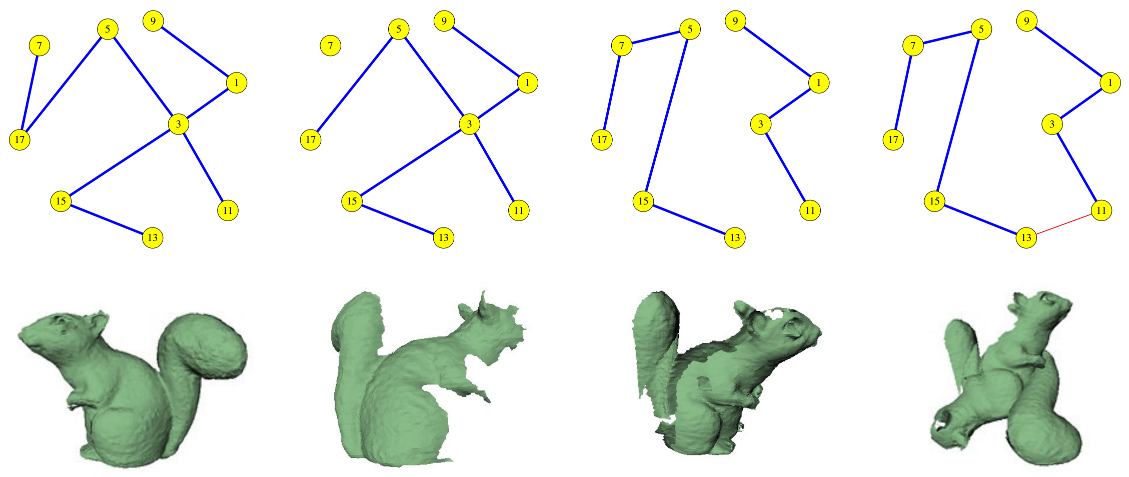
Figure 3.12: Variation of overlap distance (D_{OV}) as a function of path length.

The fact that Q_{G} is computed between non-adjacent views in G is important. These non-adjacent comparisons produce new, non-local measurements in a manner analogous to topological inference (figure 3.11). These measurements make the global quality measure much more powerful than it would be if only matches found by pair-wise surface matching were used.

Unfortunately, the statistical quality model that we learned for computing Q_{L} for pair-wise matches does not apply to the computation of Q_{G}, since we compute local quality between non-adjacent views, and the distributions of the feature values used in the computation change as a function of path length. For example, accumulating registration error will cause D_{OV} for correct matches to increase with longer path lengths (figure 3.12).

Our solution is to learn a separate statistical model for each path length ($l = 2 \ldots L$). To do this, we generate random paths of a given length from G_{LR} of the training data. The model for path length L is used to compute the quality for any path longer than L. The prior probabilities $P(M^+)$ and $P(M^-)$ can no longer be estimated from data, since the multi-view surface matching algorithm actively chooses which matches to include in the model hypothesis. We therefore assume uniform priors (i.e., we set $P(M^+) = P(M^-) = 0.5$). Figure 3.13 shows the value of the `sum_local` measure for several model hypotheses.

We must also consider the effects of topological inference and multi-view registration on our statistical quality model. Topological inference establishes new links in the model graph, thereby shortening the path lengths between non-adjacent views. Multi-view registration improves the alignment of all the views using these new links. If we were to use



(a) $Q_{\mathrm{G}} = 115$ (b) $Q_{\mathrm{G}} = 97.3$ (c) $Q_{\mathrm{G}} = 69.9$ (d) $Q_{\mathrm{G}} = -3183$

Figure 3.13: Global quality values for several versions of the squirrel model. The model hypothesis is shown in the top row with the corresponding 3D visualization in the bottom row. a) Correct model. b) Correct model with a single view detached; c) Correct model split into equally sized two parts (only one part shown in 3D). d) Model with one error.

the statistical model for the original path lengths, the quality measure would be overly pessimistic. Instead, we use the shortest path length between views in the augmented graph for choosing the appropriate statistical model. A second effect of topological inference is that the overlap features (F_{OV} and F_{ROV}) are no longer informative. For short sequences of pair-wise matches, correctly registered pairs are likely to have more overlap than incorrectly registered ones. When topological inference is applied, any amount of overlap may be detected, and a small overlap fraction does not indicate an incorrect registration. Therefore, we omit the overlap features from the computation of Q_{L} for graphs where topological inference has been applied.

This definition of global quality has an implicit bias toward solutions with more overlap. For example, given eight views of the corners of a cube, the hypothesis that the object is a 3-sided pyramid would be preferable to that of a cube (assuming that G_{LR} contains the appropriate pair-wise matches). For the pyramid hypothesis more view pairs would contribute to Q_{G} (every pair of views) than for the cube hypothesis (views on opposite sides would not contribute). Since all Q_{L} terms have approximately equal values, the hypothesis with more terms is preferred. This effect does not generally happen in real scenes unless they are perfectly symmetric.

We also developed an alternative definition of global quality based on the point-to-point multi-view registration cost function (equation 5.9). If we denote the distance between the k^{th} correspondence in the overlapping region of S_i and $T_{ij}S_j$ by d_{ijk} (assuming outliers have been removed), then we can define global quality as

$$Q_{\mathrm{G}}(G) = -\sum_{(i,j)\in V_c}\sum_{k}(d_{ijk})^2 \tag{3.22}$$

The problem with this definition of Q_{G} is that it is maximized by the empty model hypothesis (i.e. N model parts), which is a degenerate solution. Our solution is to include a penalty term proportional to the expected value of the overlap distance measurements

$$Q_{\mathrm{G}}(G) = -\sum_{(i,j)\in V_c}\sum_{k}((d_{ijk})^2 - \lambda \mathrm{E}[(d_{ijk})^2]) \tag{3.23}$$

where $\lambda \geq 1$ is a weighting parameter. Now, even though the empty model hypothesis still results in $Q_{\mathrm{G}} = 0$, the quality will be improved if a match with smaller than expected closest point distances is added. The weighting parameter, λ, controls the relative cost of adding an edge to the hypothesis. For example, if $\lambda = 2$, an edge will be preferred even if it results in overlap distances that are, on the average, nearly twice as large as expected. The larger the value of λ, the more biased the measure will be toward model graphs with more overlap. We call this global quality measure `dist_sqr`. Figure 3.13 shows the value of the `dist_sqr` measure for several model hypotheses.

The value of $\mathrm{E}[(d_{ijk})^2]$ depends on the sensor used to collect the data. It can be estimated from multi-view registration of a set of training views that are known to be correct. Using the same argument that we make for outlier elimination (section 5.3.2), $\mathrm{E}[(d_{ijk})^2]$ can be estimated as σ^2 from equation 5.19. The parameter, λ, is set by the user (we use $\lambda = 1$).

Depending on the application, other global quality functions could be devised. For example, if it is known *a priori* that the scene consists of a "closed" object (i.e., an object with an inside and an outside), then we can define a global quality measure based on the length of the "open boundary."

$$Q_{\mathrm{G}}(G) = -\sum_{i=1}^{N} \frac{l_i}{L_i} \tag{3.24}$$

where L_i is the total length of the boundary edges of S_i and l_i is the length of the boundary edges in S_i that do not overlap any other surface, S_j. With this definition of Q_{G}, the preferred model for the example of the eight views of a cube would be a cube.

3.5 Local and global consistency tests

Associated with each local quality measure Q_{L} is a Bayesian classifier C_{L}, which is a thresholded version of the corresponding Q_{L}:

$$C_{\mathrm{L}} = \begin{cases} \text{true} & \text{if } Q_{\mathrm{L}} > \log\lambda \\ \text{false} & \text{otherwise} \end{cases} \tag{3.25}$$

We call C_{L} the local consistency test, and a match that passes this test is said to be locally consistent. We use the local consistency test to remove the worst matches from G_{LR}, conservatively choosing the threshold λ based on training data to minimize the chances of removing any correct matches.

Just as the `sum_local` global quality measure extended local quality to an entire model graph, we can extend the local consistency test C_{L} to handle an entire model graph. A model graph G is globally consistent if every pair of connected views in G is locally consistent:

$$C_{\mathrm{G}}(G) = \begin{cases} true & \text{if } \forall_{(i,j)\in E_c} C_{\mathrm{L}}(V_i, V_j, T_{ij}) = true \\ false & \text{otherwise} \end{cases} \tag{3.26}$$

where E_C the set of connected (not necessarily adjacent) view pairs in G. We assume that G is pose consistent, and any hidden edges in a hypothesis H are removed before computing $C_{\mathrm{G}}(H)$. As with the definition of Q_{G}, computing consistency between non-adjacent views is important. The non-adjacent tests provide many independent consistency tests, which can mean the difference between success and failure during the search for a correct model hypothesis.

Chapter 4

Multi-view surface matching search algorithms

This chapter addresses the question, "How do you find the best model hypothesis in an exponentially large search space?" We have developed two classes of algorithms to perform this search: iterative addition (section 4.2) and iterative swapping (section 4.3). The iterative addition algorithms, based on minimum spanning trees, incrementally construct a model by repeatedly adding edges to an initially empty model hypothesis while ensuring that the hypothesis passes the global consistency test at each iteration. This type of algorithm can be very fast, but the output hypothesis, though globally consistent, is not necessarily the *best* model according to the global quality measure. The second class of algorithms uses an edge swapping strategy to address this issue. Starting with a globally consistent model hypothesis H, a new hypothesis H' is proposed by replacing one of the edges in H with another edge from G_{LR}. These algorithms maximize global quality while maintaining the global consistency property. Before going into the details of each class of algorithms, we briefly review our multi-view surface matching framework.

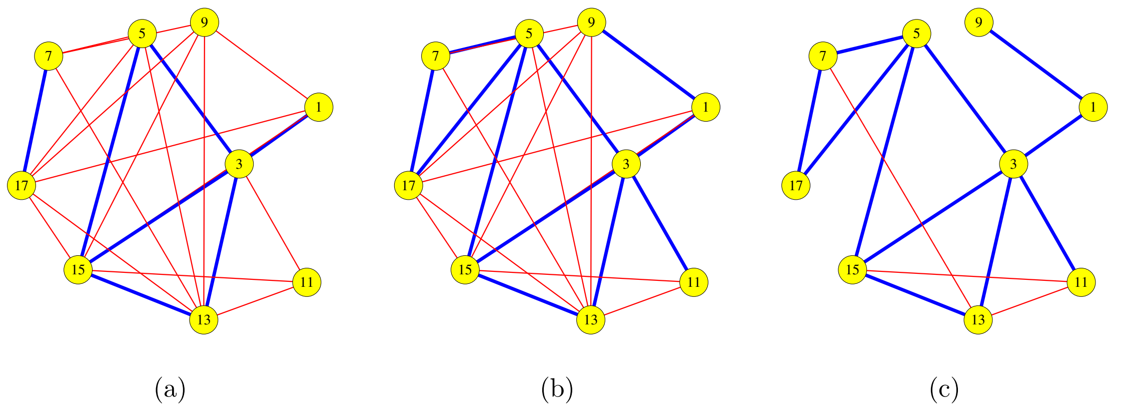


Figure 4.1: Model graphs for the local phase for the views in figure 2.1. a) The result of exhaustive pair-wise surface matching. b) The result of pair-wise registration. Notice that several incorrect matches converged to their correct values (e.g., $E_{1,9}$). c) The result of local consistency filtering. This last model graph is G_{LR}.

4.1 Multi-view surface matching review

Recall that our framework for multi-view surface matching, introduced in section 1.3, consists of two phases. In the local phase, we perform exhaustive pair-wise surface matching (figure 4.1a), followed by pair-wise registration (figure 4.1b) and local consistency filtering (figure 4.1c), producing the model graph G_{LR} that serves as input to the global phase. The goal of the global phase is to find within G_{LR} a model hypothesis containing only correct matches and having the minimal number of components.

It is important that we include only correct matches in a model hypothesis. Unlike some problems where a few mistakes have only a minor effect on the overall solution, a single incorrect match has a catastrophic effect for multi-view surface matching (figure 4.2). The reason for this follows from the fact that the relative pose for an incorrect match is far from its true value. An incorrect relative pose will therefore corrupt the absolute poses of one or more views in the graph, resulting in a visibly incorrect model.

We formulate the global phase search as a mixed continuous/discrete optimization problem. The discrete optimization is a combinatorial search over acyclic sub-graphs of G_{LR}, and the continuous optimization is a local refinement of the absolute pose parameters around

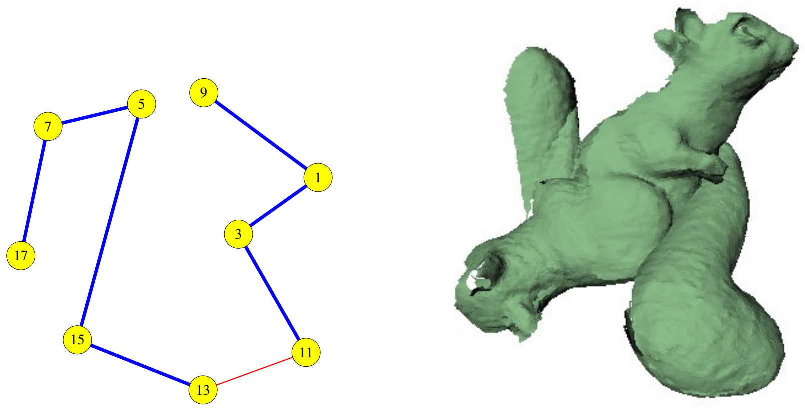


Figure 4.2: A single incorrect match results in catastrophic failure. a) The model graph contains an incorrect match between views 11 and 13. b) The resulting registered views in 3D.

the initial values dictated by the relative poses from pair-wise matching. Our approach is to decompose the optimization into two alternating sub-problems: a discrete optimization over model graph topology for fixed poses and a continuous optimization over absolute poses for a fixed model hypothesis. For a fixed hypothesis, the continuous optimization is simply the multi-view registration problem. As explained in section 2.6, whenever we perform multi-view registration, we first augment the hypothesis using topological inference to find all overlapping view pairs. In our algorithms, we denote this process by the function `find_all_overlaps`(H).

Even if we only consider the discrete optimization, the number of potential solutions is at least as large as the number of spanning trees in G_{LR}, which is $\mathrm{O}(N^N)$. In fact, the multi-view surface matching problem is NP-complete (appendix B).

4.2 Iterative addition algorithms

Iterative addition algorithms construct a solution to the multi-view surface matching problem by repeatedly adding edges to an initial model hypothesis (H_0) containing nodes but no edges. Given that the goal is to find a connected sub-graph of G_{LR} containing only correct matches, an obvious approach is to compute the minimum spanning tree of G_{LR}, using the local quality Q_{L} of the pair-wise matches for the edge weights.[1] The minimum spanning tree represents the best N-1 pair-wise matches that form single-part model. There are two well-known algorithms for computing the minimum spanning tree of a graph: Kruskal's algorithm and Prim's [13]. Kruskal's algorithm begins with N components and repeatedly merges components by inserting the best unused edge that connects two components. Prim's algorithm starts with a seed node and grows a solution by adding the best unused edge that connects the seed component with any of the remaining nodes. Each method leads to a different multi-view surface matching search algorithm. We begin by describing an iterative merging approach based on Kruskal's algorithm.

4.2.1 The iterative merging algorithm (`iterative_merge`)

With iterative merging, Kruskal's algorithm forms the discrete portion of the search. After each discrete step, we use topological inference to augment the current hypothesis with edges for all overlapping views. Multi-view registration on this augmented graph forms the continuous optimization portion of the search. Finally, the updated hypothesis is verified using the global consistency test C_{G} and rejected if it fails.

The pseudo-code for this algorithm is shown in figure 4.3. Initially, H represents an N-part model (line 1). The edges of G_{LR} are sorted by their local quality and tested one at a time. In each iteration through the loop, the best untested edge from G_{LR} is selected, and if it connects two components, a temporary model hypothesis H' is created with the new edge added, thereby joining two model parts (line 4). Topological inference is applied to H' to form the augmented graph H'', which includes edges for all overlapping views (line 5). The alignment of the views in H'' is improved using multi-view registration (line 6), and the resulting poses are copied back into H' (line 7). If the resulting model hypothesis is globally consistent (line 8), the new edge is accepted, and H' becomes the starting point for the next iteration (line 9). Eventually, the algorithm either finds a spanning tree of G_{LR}, resulting

[1] Actually, we use $-Q_{\mathrm{L}}$ since larger Q_{L} means a better match.

```
The iterative_merge algorithm
Input: $G_{\mathrm{LR}}$ for the input views
Output: Model hypothesis $H$ with absolute poses specified
1: $H \leftarrow H_0$
2: for all edges $E_{i,j} \in G_{\mathrm{LR}}$, sorted in decreasing order by $Q_{\mathrm{L}}$ do
3:   if $N_i$ and $N_j$ are not connected in $H$ then
4:     $H' \leftarrow H \cup E_{i,j}$
5:     $H'' =$ find_all_overlaps($H'$)
6:     $H'' =$ multiview_register($H''$)
7:     update $H'$ using the poses of $H''$
8:     if $C_{\mathrm{G}}(H') = true$ then
9:       $H \leftarrow H'$
```

Figure 4.3: Pseudo-code for the `iterative_merge` algorithm

in a single-part model, or the list of edges is exhausted, resulting in a multi-part model. Figure 4.4 shows a trace of the algorithm in operation. The final model, corresponding to the hypothesis in figure 4.4f, is shown in figure 4.5.

4.2.2 Variations on the `iterative_merge` algorithm

In order to better understand the behavior of the `iterative_merge` algorithm, we experimented with several modified versions of the basic algorithm (`full` hereafter). The first two variations, `no_mvreg` and `mst_only`, are simplifications of the `full` algorithm that analyze the effect of the global consistency test and the continuous optimization. The `no_mvreg` algorithm omits the continuous optimization step (lines 5 and 6), and the `mst_only` algorithm skips the global consistency test (line 8) as well.

The `mst_only` algorithm simply finds the minimum spanning tree of G_{LR}. It has the advantage that it is very fast, but the result may not be globally consistent. The global consistency test can be performed at the end as a verification, but it is not possible to correct the problem. Another disadvantage is that this algorithm will not output a multi-part hypothesis in the event that a correct single-part solution does not exist within G_{LR}.

The `no_mvreg` algorithm integrates the global consistency check into `mst_only`, effectively allowing a single step of backtracking. However, the accumulation of small pair-wise

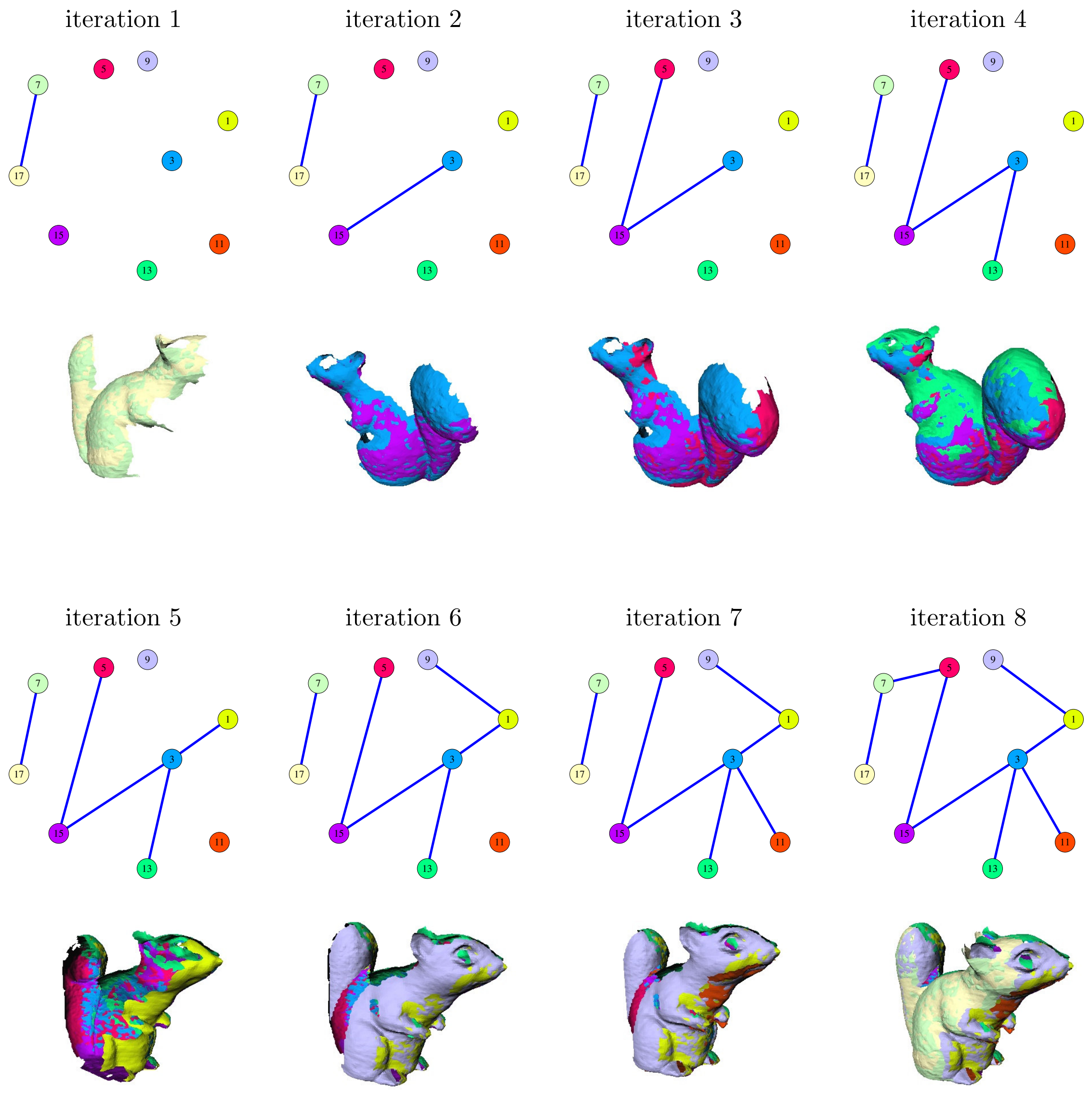


Figure 4.4: Trace of the `iterative_merge` algorithm. The model hypothesis for each iteration is shown along with the 3D visualization of the affected model part.

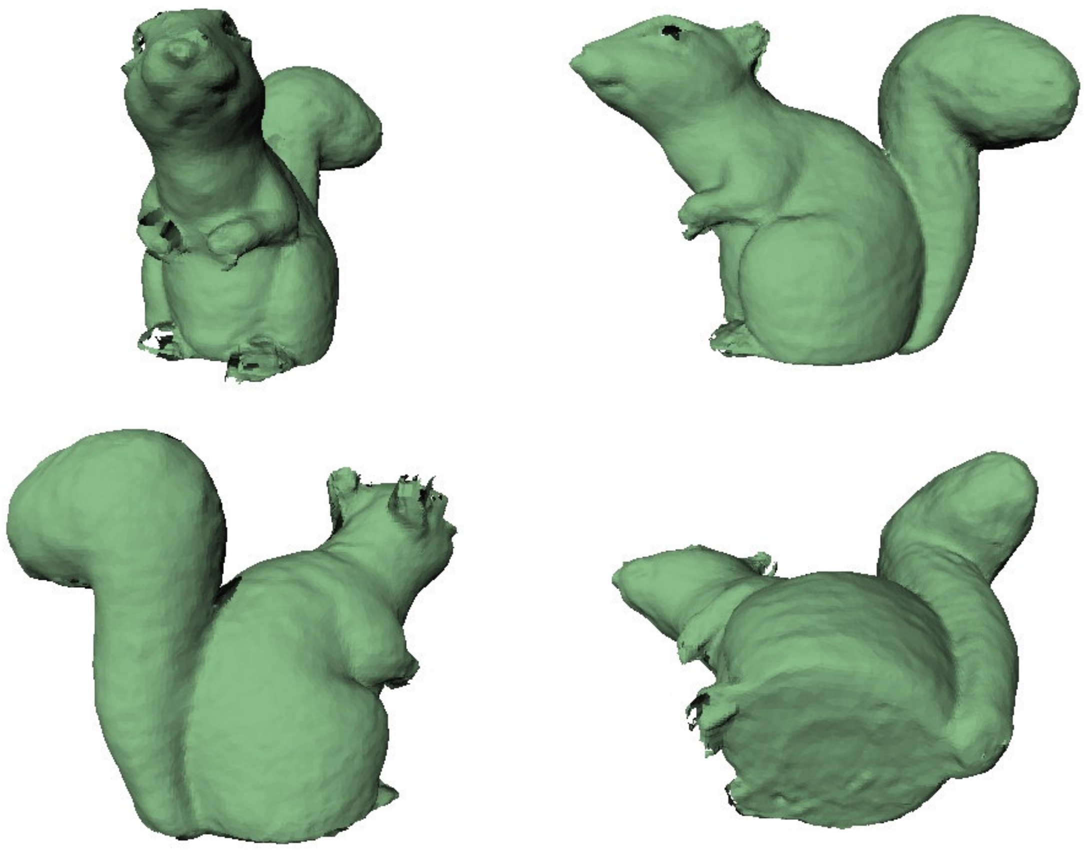


Figure 4.5: The registered views resulting from the execution of the `iterative_merge` algorithm shown in figure 4.4

registration errors can cause a hypothesis to be globally inconsistent even though it contains only correct matches. The multi-view registration in the `full` algorithm solves this problem by distributing the accumulated errors over the entire model.

Finally, since multi-view registration is a relatively computationally intensive operation, we can skip this step as long as the model hypothesis remains globally consistent. We call this algorithm `fast_mvreg`. With this algorithm, we add a global consistency check before step 5 and immediately accept H' if the test succeeds, only performing steps 5 through 7 if the test fails. Since the multi-path-length quality model developed in section 3.4 accounts for the accumulating errors in a sequence of matches, the algorithm can execute several discrete updates between continuous updates. This gives us nearly the efficiency of the `no_mvreg` algorithm without the pitfalls of accumulated error. The results of this experiment are given in section 7.5.1.

4.2.3 The iterative growing algorithm (`iterative_grow`)

The `iterative_grow` algorithm is structured identically to `iterative_merge` except that Prim's algorithm is used instead of Kruskal's. We must make two additional changes to the algorithm. First, we need to select a seed node, which we choose to be one of the end nodes of the highest quality edge in G_{LR}. Second, since the algorithm only joins views to the seed component that maintain global consistency, we may run out of viable edges before finding a spanning tree. In such a case, the algorithm is restarted using the remaining nodes and edges, and the final output is the union of individual model hypotheses, each of which represents a separate model part. Figure 4.6 shows a trace of the algorithm.

4.2.4 Discussion

The `iterative_grow` algorithm has the advantage that it quickly builds up a model part containing many views that will subsequently provide many constraints through topological inference. The `iterative_merge` algorithm, on the other hand, may form many parts with just a few views, any of which could contain an incorrect match. The disadvantage of

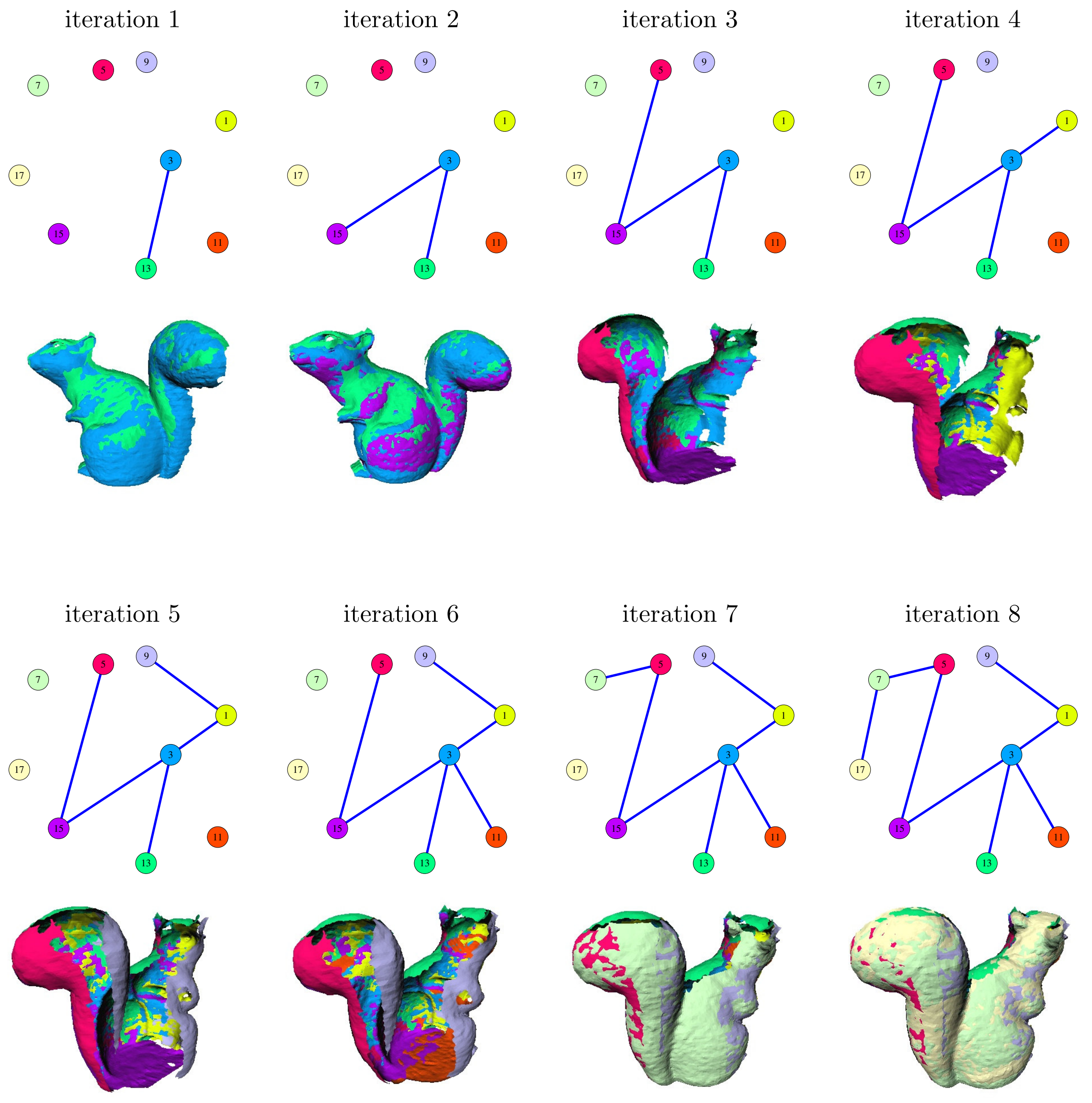


Figure 4.6: Trace of the `iterative_grow` algorithm. The model hypothesis for each iteration is shown along with the 3D visualization of the model.

`iterative_grow` is that once the initial seed is chosen, the set of viable edges is severely restricted – only edges in G_{LR} adjacent to the nodes in the seed component are allowable. As a result, `iterative_grow` may be forced to choose relatively low-quality edges in the first few iterations, increasing the chances of making an undetected mistake. The experiments in chapter 7 show that in practice, `iterative_merge` and `iterative_grow` fail on the same objects, which suggests that the generally only a few views need to be combined before the consistency constraints become fully effective.

The complexity of the iterative addition algorithms is dominated by the multi-view registration step. The inner operations in each algorithm will be executed $O(E)$ times, since in the worst case, every edge will be tried. Topological inference and the global consistency test are $O(N^2)$ operations. Multi-view registration requires solving a linear system of size $6(N-1)$, which is $O(N^3)$. The remaining operations are all $O(N)$ or $O(E)$. This gives the algorithm a total running time of $O(EN^3)$. However, the linear system in the multi-view registration step is sparse. The sparseness is directly related to the number of edges in the graph used for multi-view registration, so if each view only overlaps a few other views, the resulting matrix will be quite sparse.

One problem with the iterative addition algorithms is that they do not explicitly represent the possibility that the best solution is a multi-part model. Although the algorithms can output multi-part models, this behavior is really just a by-product of the global consistency test. Another problem is that these algorithms can output a model hypothesis that is globally consistent but still not the correct solution. The reason lies in the fact that symmetries in the input views may allow for multiple globally consistent hypotheses, only one of which can be the correct one. To address these two issues, we turn to our second class of algorithms, the iterative edge swappers.

4.3 Iterative swapping algorithms

Iterative swapping algorithms search G_{LR} for the best model hypothesis according to the global quality measure Q_{G}. Additionally, they explicitly represent multi-part model hy-

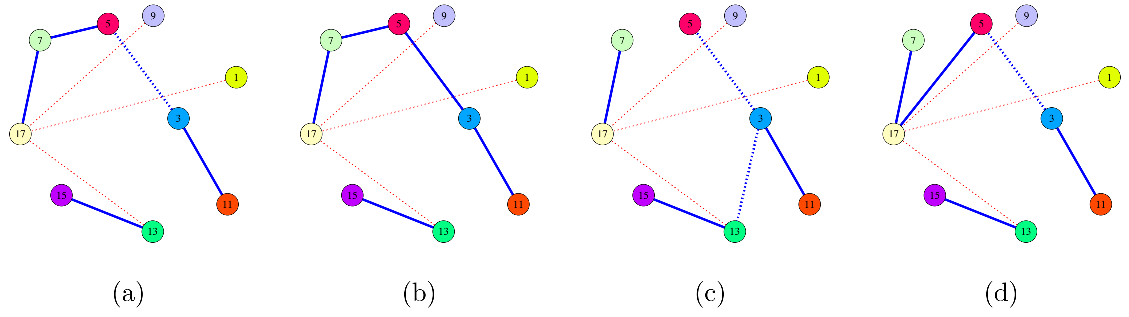


Figure 4.7: a/b) An example edge flip. The edge between V_3 and V_5 is flipped from hidden to visible. c/d) An example edge swap. The edge between V_3 and V_{13} is swapped for the one between V_5 and V_{17} taken from G_{LR} (figure 4.1c).

potheses, which lets us actively balance the competing goals of forming a single-part model and minimizing the error between overlapping views. Recall that we defined a model hypothesis for this class of algorithms to be a spanning tree of G_{LR} with an additional edge attribute called visibility (figure 2.3c). Hidden edges indicate divisions between parts, which are connected by visible edges. This hypothesis representation allows us to search the space of all acyclic sub-graphs of G_{LR} using an iterative edge swapping algorithm.

The iterative swapping algorithms begin with a globally consistent model hypothesis chosen from G_{LR}. Such a hypothesis is guaranteed to exist, since we can choose any spanning tree of G_{LR} and label all of the edges as hidden. In this case, the hypothesis will consist of N parts, which are trivially globally consistent. Alternately, we can use one of the iterative addition algorithms to generate an initial hypothesis – a good idea, since those algorithms often find the correct solution or get very close to it. At each iteration, a new model hypothesis H' is proposed based on the current hypothesis by applying one of two types of update operations: an edge flip or an edge swap. An edge flip involves choosing an edge from the current hypothesis and toggling its visibility state (figure 4.7a/b). If the edge is changed from visible to hidden, the model part is split into two parts at this edge. Conversely, toggling an edge from hidden to visible joins two model parts. For an edge swap operation (figure 4.7c/d), a new hypothesis is generated by choosing an edge from the current hypothesis and removing it. Then an edge from G_{LR} that re-establishes the spanning tree property is chosen and inserted into the hypothesis with its visibility set to true. The new hypothesis is then tested in the same manner as with the iterative addition algorithms, first using topological inference to find all overlaps, then performing multi-view registration on this augmented graph, and finally testing for global consistency. If the new hypothesis is globally consistent, the global quality Q_{G} is computed, and if the quality is better, the proposed update is accepted.

4.3.1 Greedy edge swapping (`greedy_swap`)

In our outline of the edge swapping algorithm, we did not specify how we decide which type of edge update operation to perform or how we choose which edge to update. If we try all possible next states in a fixed order, we obtain a greedy hill-climbing algorithm, which will converge to a local maximum of Q_{G}. We call this algorithm `greedy_swap`. In our implementation, we first try all edge flip operations and then try the edge swap operations, repeating until no improvement can be found.

For a large model graph, the number of possible update operations grows rapidly, making the problem intractable. The number of possible edge flips is $\mathrm{O}(N)$, but the number of swaps is $\mathrm{O}(N^2)$, since there are $N-1$ edges that can be disconnected and up to $N-2$ edges from G_{LR} that can re-establish the spanning tree. Combining this result with the complexity of multi-view registration ($\mathrm{O}(N^3)$) gives an $\mathrm{O}(N^5)$ algorithm, which can quickly get out of hand. One solution to this problem is to non-deterministically choose the update at each iteration, which leads to our second approach: stochastic edge swapping.

4.3.2 Stochastic edge swapping (`random_swap`)

The `random_swap` algorithm randomly selects an update operation (flip or swap) and the affected edges at each iteration. The edge selection can be performed with uniform randomness or using edge weights as a guide (i.e., swapping out the lower quality matches for higher-quality ones). Pseudo-code for this algorithm is shown in figure 4.8, and figures 4.9 and 4.10 show a trace of it in operation.

```
The random_swap algorithm
Input: $G_{\mathrm{LR}}$ for the input views,
$H_{\mathrm{init}}$, a globally consistent hypothesis from $G_{\mathrm{LR}}$
Output: Model hypothesis $H$ with absolute poses specified
1: $H \leftarrow H_{\mathrm{init}}$
2: repeat
3:    choose flip or swap with $p_{\mathrm{flip}}$
4:    if flip then
5:       $H' \leftarrow$ propose_random_flip($H$)
6:    else {swap}
7:       $H' \leftarrow$ propose_random_swap($H$)
8:    $H'' =$ find_all_overlaps($H'$)
9:    $H'' =$ multiview_register($H''$)
10:   update $H'$ using the poses of $H''$
11:   if $C_{\mathrm{G}}(H') = true$ and $Q_{\mathrm{G}}(H') > Q_{\mathrm{G}}(H)$ then
12:      $H \leftarrow H'$
13: until $H$ unchanged in $K$ iterations
```

Figure 4.8: Pseudo-code for the `random_swap` algorithm

Now we address the question of how to choose an edge on which to operate when weighted swapping is used. Edges are chosen by computing weights for each edge based on the local quality Q_{L} and then sampling from the resulting probability mass function (PMF). Let $\mathcal{E} = \{E_1, E_2, \ldots, E_K\}$ be the set of edges from which we wish to sample. Let q_k be the value of Q_{L} computed for E_k. Recall that Q_{L} is defined as a log odds ratio, which means Q_{L} can be any real number, positive or negative. The possibility of negative weights makes it impossible to sample using the q_k values directly. One approach is to convert q_k back into a probability:

$$q_k = \log\left(\frac{p_k}{1-p_k}\right) \Rightarrow p_k = \frac{e^{q_k}}{(1+e^{q_k})} \tag{4.1}$$

Unfortunately, this does not work very well in practice. The problem is that even for moderately small negative values of q_k, p_k will be very close to zero, which means that the associated edge will rarely, if ever, be selected during sampling. Our approach is to add an offset, Q_{off}, to each q_k (i.e., $q'_k = q_k + Q_{\mathrm{off}}$) such that the maximum q'_k is C times larger

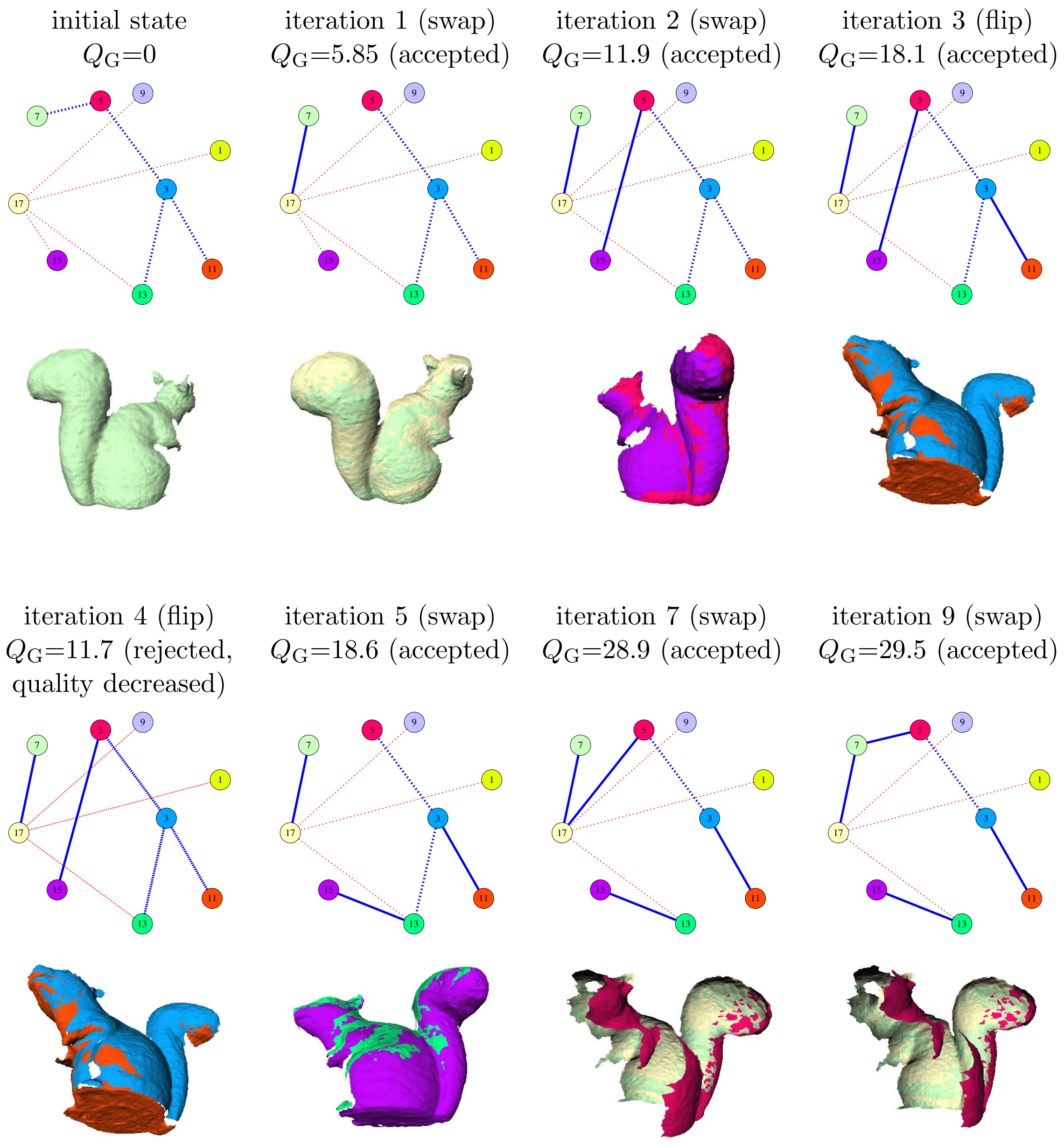

initial state
Q_{G}=0
iteration 1 (swap)
Q_{G}=5.85 (accepted)
iteration 2 (swap)
Q_{G}=11.9 (accepted)
iteration 3 (flip)
Q_{G}=18.1 (accepted)
iteration 4 (flip)
Q_{G}=11.7 (rejected, quality decreased)
iteration 5 (swap)
Q_{G}=18.6 (accepted)
iteration 7 (swap)
Q_{G}=28.9 (accepted)
iteration 9 (swap)
Q_{G}=29.5 (accepted)
7
5
9
1
17
3
15
11
13

Figure 4.9: Trace of the `random_swap` algorithm (`weighted_swap` variation). The initial state is selected randomly. Only some of the rejected updates are shown.

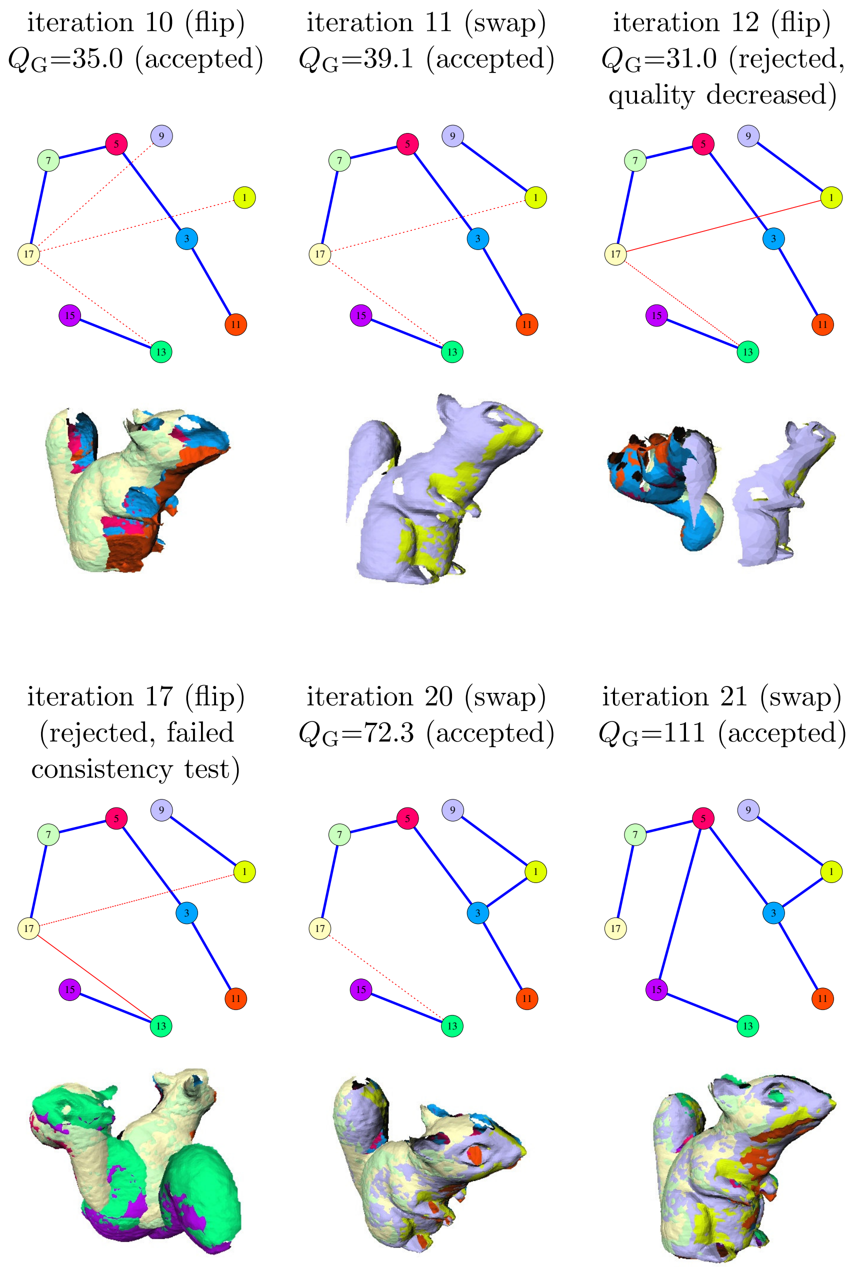

iteration 10 (flip) Q_{G}=35.0 (accepted)
iteration 11 (swap) Q_{G}=39.1 (accepted)
iteration 12 (flip) Q_{G}=31.0 (rejected, quality decreased)
iteration 17 (flip) (rejected, failed consistency test)
iteration 20 (swap) Q_{G}=72.3 (accepted)
iteration 21 (swap) Q_{G}=111 (accepted)
1
3
5
7
9
11
13
15
17

Figure 4.10: Continuation of the trace of the `random_swap` algorithm.

than the minimum q'_k. The weight for an edge is then

$$w_k = \frac{q'_k}{Q}, \tag{4.2}$$

where $Q = \sum_{k=1}^{K} q'_k$. The user-specified parameter, C ($C > 1$), controls the relative frequency with which the best edge will be selected with respect to the worst edge.[2] Q_{off} is computed from C and the q_k values:

$$Q_{\text{off}} = \frac{\max_k(q_k) - C \min_k(q_k)}{C - 1} \tag{4.3}$$

The resulting weights, w_k, are all positive and sum to one. Therefore, we can treat them as a PMF, from which we can directly sample. This gives us a method for choosing edges such that high-quality edges are more likely to be selected than low quality ones. We also need a method for choosing edges such that low quality edges are more likely to be selected. To accomplish this, we define a set of dual weights $\overline{w}_k$. The dual weights are computed by negating the edge quality values and computing a dual offset $\overline{Q}_{\text{off}}$ such that $\max_k(-q_k) = C \min_k(-q_k)$. In this case,

$$\overline{Q}_{\text{off}} = \frac{-\min_k(q_k) + C \max_k(q_k)}{C - 1} \tag{4.4}$$

Now that we have a method for sampling edges, we can detail the operations in steps 3 through 7 for the `weighted_swap` variation of the `random_swap` algorithm. First, the operation type (flip or swap) is selected, choosing swap with probability p_{flip}. If the operation is a swap, we want to replace a low-quality edge with a high-quality one. In this case, we select an edge to remove using the dual weights, computing $\overline{w}_k$ for the set of edges in the current hypothesis. Next, the set of viable edges $\mathcal{E}$ from G_{LR} that will re-establish the spanning tree property is computed, omitting the edge that was just removed. If $|\mathcal{E}| = 0$, then the removed edge is the only viable edge, and the proposed update is rejected. Otherwise, we select an edge using the weights w_k computed over the set of viable edges and add the selected edge to the hypothesis with its visibility set to true. If the operation is an edge flip, a slightly different method is used for choosing an edge to flip. First, we randomly choose,

[2]We use $C = 10$.

with probability p_{vis}, whether to flip an edge from visible to hidden or vice versa. When flipping from hidden to visible, we want to bias our selection toward high-quality edges, so we sample from the set of hidden edges using the weights w_k. Similarly, when flipping from visible to hidden, we want to select a low-quality edge, so we sample from the dual weights $\overline{w}_k$ of the visible edges in the current hypothesis.

Even with stochastic updates, the edge swapping algorithm can be susceptible to local minima. One strategy for avoiding local minima in high-dimensional optimization problems is to use simulated annealing [43]. With simulated annealing, an update is sometimes accepted even if the global quality decreases. The probability of accepting a lower-quality hypothesis is a function of the size of the reduction in quality as well as parameter T, which is called the temperature:

$$\Delta Q = Q_{\text{G}}(H') - Q_{\text{G}}(H) \tag{4.5}$$

$$\begin{aligned} &\text{if} \quad \Delta Q < 0, \quad \text{accept} \quad H' \\ &\quad \text{otherwise,} \quad \text{accept} \quad H' \quad \text{with probability} \quad p = e^{-(\Delta Q/T)} \end{aligned} \tag{4.6}$$

Another strategy for encouraging the algorithm to cover the state space is to perform multiple edge update operations on each iteration. Our experiment in section 7.5.4 shows that these advanced approaches are not necessary for the scenes that we have modeled.

4.3.3 Variations on the `random_swap` algorithm

In a manner analogous to the variations of the `iterative_merge` algorithm described in section 4.2.2, we tested several versions of the `random_swap` algorithm: `no_mvreg`, `qual_only`, and `fast_mvreg`. The `no_mvreg` algorithm omits the continuous optimization step (lines 8 through 10), stressing the capabilities of our multi-path length quality model. The `qual_only` algorithm also omits the global consistency test in step 11. This algorithm tests whether just optimizing Q_{G} is sufficient or if the global consistency test provides an additional benefit. The `fast_mvreg` algorithm adds a global consistency test and global quality calculation (line 11) before line 8. If the proposed hypothesis is globally consistent and the quality has improved, the hypothesis is immediately accepted without performing multi-view registration. The results of this experiment are given in section 7.5.2.

Chapter 5

Component algorithms

This chapter discusses the theory and implementation of the component algorithms used in multi-view surface matching. Specifically, we cover pair-wise surface matching (section 5.1), pair-wise registration (section 5.2), and multi-view registration (section 5.3). In chapter 2, we described these algorithms from a high level and illustrated their operation with examples. Up to this point, we have treated each algorithm essentially as a black box with well-defined inputs, outputs, and behavior. Now we present them in more detail. It should be clear that multi-view surface matching is not bound to a specific implementation of these component algorithms. If a new algorithm with similar high-level behavior but better performance is found or developed, the existing algorithm can easily be replaced with the improved one. With the exception of our point-to-point matching multi-view registration algorithm (section 5.3.2), our implementations are modified or specialized versions of existing, proven algorithms.

5.1 Pair-wise surface matching

Pair-wise surface matching aligns two surfaces with no initial relative pose estimate. Such algorithms are commonly used in 3D object recognition systems, where the goal is to determine the pose (if any) of one or more 3D model objects within a 3D scene. Typically, these algorithms work by finding correspondences between the model and scene based on an invariant surface property such as curvature. Given a sufficient number of correct correspondences, the relative pose between model and scene can be estimated. This putative match is then verified by checking the surface similarity over the entire overlapping region. If the surfaces are sufficiently similar, the object is declared to be recognized, and the algorithm outputs the relative pose.

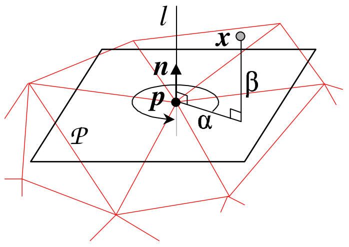
One approach to establishing model-scene correspondences is to use point signatures, which encode local surface properties in a data structure that facilitates efficient correspondence search and comparison. Proposed encodings include spin-images [39], splashes [73], point signatures [12], harmonic shape images [85], spherical attribute images [30], the tripod-operator [57], and point fingerprints [75]. An alternative approach is to explicitly detect extended features such as curves of maximal curvature [24], intersections of planar regions [1], or bitangent-curves [84] and match them between model and scene.

5.1.1 Spin-image surface matching

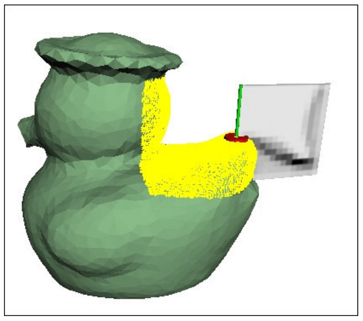
Our surface matching algorithm falls into the point signature category. It is based on Johnson's spin-image surface matching algorithm, which was used primarily for object recognition [39]. The algorithm is fast, matching two views in 1.5 seconds[1], does not require explicit feature detection, and operates on surfaces with arbitrary topology (e.g., meshes with holes and disconnected regions).

Spin-images represent local surface shape by projecting surface points into a two dimensional coordinate system relative to a given oriented point $\boldsymbol{b} = (\boldsymbol{p}, \boldsymbol{l})$ (figure 5.1). More precisely, a spin-image is a 2D histogram of points within a cylindrical region of support around $\boldsymbol{p}$. The axes (α, β) of a spin-image correspond to the r and z axes of a cylindrical coordinate system (r, θ, z) based at $\boldsymbol{p}$ with its z axis coincident with $\boldsymbol{l}$. Since spin-images are two-dimensional representations of 3D shape, they can be rapidly compared using image-based similarity measures, such as normalized correlation. Spin-images are invariant to

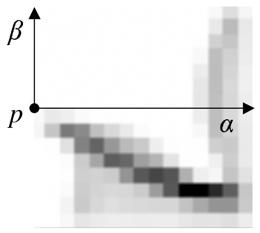
[1]Timing is for 1000-face meshes on a Linux PC with a 1.5GHz AMD Athlon processor.



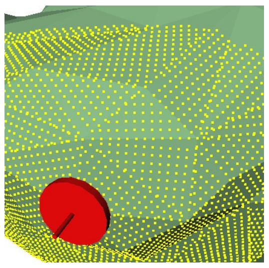
(a)



(b)



(c)



(d)

Figure 5.1: a) The spin-image at $\boldsymbol{p}$ is computed by histogramming all nearby points x in terms of their distance α to the the axis $\boldsymbol{l}$ and distance β to the tangent plane $\mathcal{P}$. b) An example with a real surface. The red marker indicates the basis point, and the yellow dots are points within the support region. The spin image can be visualized as a flag rotating around the axis $\boldsymbol{l}$, summing the surface points as they pass through the flag. c) The spin-image for the basis point in (b). d) The sampling pattern used in face-based spin-images.

rigid transformations and robust to missing data caused by occlusions.

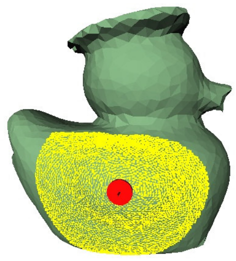
We now briefly describe the spin-image surface matching process; additional details can be found in [36]. The algorithm begins by computing point signatures for sample points on both surfaces. Each point signature from one surface is compared with those from the other surface, and the most similar pairs become candidate correspondences, some of which may be incorrect. The candidates serve as seeds in a clustering algorithm that finds groups of correspondences consistent with a single rigid-body transform. The relative pose computed from each group that passes this geometric consistency test is a candidate match. Finally, the candidate matches are refined by a version of the iterative closest point (ICP) algorithm, verified, and ranked. The top matches are provided as output.

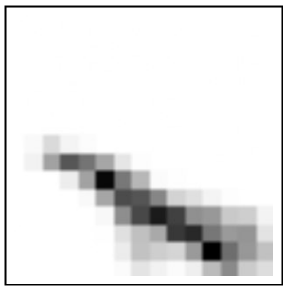
5.1.2 Face-based spin-images

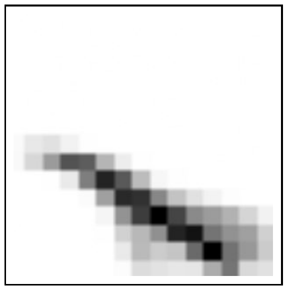
In Johnson's work, spin-images for a mesh representation were computed using the vertices of a mesh. This vertex-based approach leads to two problems. First, the mesh must have an approximately uniform distribution of vertices. Johnson used a mesh decimation algorithm to enforced this property. Second, vertex-based spin-images limit the choice of support size and number of bins in the spin-image histogram. If the support size is too small, or too many bins are used, then only a few points fall into each bin, and the resulting histogram becomes unstable. Small perturbations in vertex location cause the spin-images for identically shaped surfaces to be dissimilar.

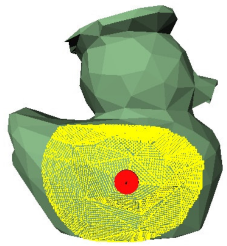
Our solution to these problems is to define spin-images in terms of mesh faces rather than vertices. Using face-based spin-images frees us from the dependency on the particular distribution of vertices on the surface, making spin-images independent of the underlying mesh resolution. The need for a resolution independent matching algorithm is particularly evident when modeling terrain using ground-based sensors, where the terrain is viewed from a shallow angle. In such cases, variation in resolution on the order of 100:1 is not uncommon. As a result, surfaces of widely different resolutions are frequently compared.

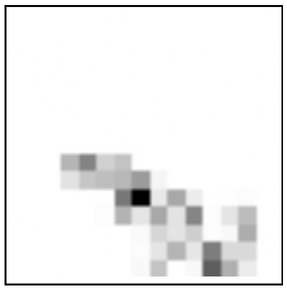
To compute face-based spin-images, we sub-sample each mesh face by linearly interpo-


(a) high resolution 3D


(b) vertex-based


(c) face-based


(d) low resolution 3D


(e) vertex-based

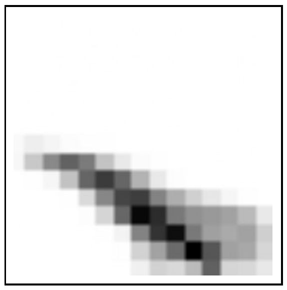

(f) face-based

Figure 5.2: Resolution independence of face-based spin-images. a) 3D view showing a sample basis point and its support region for a high-resolution mesh (4253 faces). b) The corresponding vertex-based spin-image. c) The face-based spin image for the same point. d-f) The corresponding figures for a low-resolution mesh (500 faces) show that the face-based spin image maintains its similarity to the high-resolution version while the vertex-based one does not.

lating in a raster-scan pattern[2] (figure 5.1d). The sub-sampled points are projected into spin-image coordinates, and a constant factor for each point is added to the corresponding spin-image bin. A quick geometric test eliminates faces that are entirely outside the support region. Experiments in section 7.2 show that face-based spin-images represent surface shape more accurately than vertex-based spin-images, resulting in improved similarity between corresponding points.

The resolution independence achieved by face-based spin-images has implications for all spin-image matching systems. First, it generalizes the spin-image representation by removing the uniform edge length requirement, which constrained the class of meshes for which

[2]Carmichael developed an alternate method in which each triangle is projected into spin-image coordinates and the resulting shape is then raster scanned [10].

spin-images could be employed. Johnson's edge length equalization step can now be eliminated or replaced with any shape-preserving mesh simplification algorithm (section 8.1.4). Second, any meshes of widely different resolutions – not just terrain – can now be compared with one another (figure 5.2). Third, surfaces can be stored and compared at the resolution that minimally represents them, reducing the required storage space and computation time for matching. Finally, the surface interpolation concept can be extended to any surface representation that can be sampled (e.g., parametric surfaces).

5.2 Pair-wise registration

Pair-wise registration algorithms improve an initial relative pose estimate by minimizing an objective measure of registration error. The dominant algorithm in this category is the iterative closest point (ICP) algorithm, which repeatedly updates the relative pose by minimizing the sum of squared distances between closest points on the two surfaces (*point-to-point matching*) [5]. Chen and Medioni proposed a similar method in which the distance between points and tangent planes is minimized instead (*point-to-plane matching*) [11]. Rusinkiewicz' survey of ICP algorithms provides an elegant taxonomy and unifying framework for comparing the numerous extensions to the basic algorithm [66].

We have implemented two pair-wise registration algorithms. The first is a point-to-point matching algorithm, a variant of the ICP algorithm extended to handle partially overlapping surfaces [86]. The second is a point-to-plane matching algorithm similar to Chen and Medioni's algorithm.

5.2.1 Point-to-point matching (the ICP algorithm)

Although the ICP algorithm is well-known, we present a brief description for completeness and as an introduction of our notation, which becomes somewhat more complicated in the case of multi-view registration.

As input, the algorithm is given two surfaces, S_i and S_j, and an initial relative pose T_{ji}. Recall that T_{ji} is an operator that transforms points from the coordinate system of S_i to that of S_j:

$$\boldsymbol{p}' = T_{ji}\boldsymbol{p} \equiv \boldsymbol{R}_{ji}\boldsymbol{p} + \boldsymbol{t}_{ji}, \tag{5.1}$$

where $\boldsymbol{R}_{ji}$ is a rotation matrix and $\boldsymbol{t}_{ji}$ is a translation vector. For notational convenience, we will omit the ji subscript on T_{ji} for the remainder of this section. We parameterize T with a vector $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z, t_x, t_y, t_z)^{\mathrm{T}}$, where θ_x, θ_y, and θ_z are rotation angles about the x, y, and z axes (the X-Y-Z Euler angles) and t_x, t_y, and t_z are the components of the translation vector..

First, a set of sample points $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_K\}$ is selected from S_i. The objective function to be minimized is the mean squared distance between the sample points, transformed by T, and the surface S_j:

$$E(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \sum_{k=1}^{K} \frac{u_k}{U} \left[D_{\mathrm{CP}}(T\boldsymbol{p}_k, S_j)\right]^2 \tag{5.2}$$

where u_k is an inlier indicator variable (i.e., $u_k = 1$ if $\boldsymbol{p}_k$ has a corresponding point on S_j and $u_k = 0$ otherwise), $U = \sum_{k=1}^{K} u_k$ (i.e., the number of inlier points), and D_{CP} is the closest point distance (equation 2.1).

The ICP algorithm capitalizes on the fact that, for relative poses near the solution, corresponding points on two surfaces are well-approximated by the closest points. Recall that the closest point to $T\boldsymbol{p}_k$ on S_j is the point $\boldsymbol{q} = \boldsymbol{q}_k$ that minimizes equation 2.1. The pairings $(\boldsymbol{p}_k, \boldsymbol{q}_k)$ are called correspondences. For fixed correspondences, equation 5.2 reduces to

$$E(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \sum_{k=1}^{K} \frac{u_k}{U} \|T\boldsymbol{p}_k - \boldsymbol{q}_k\|^2 \tag{5.3}$$

which has a closed form solution [23]. The algorithm repeats the following steps until convergence[3]:

1. For fixed $\boldsymbol{\theta}$, compute the closest point $\boldsymbol{q}_k$ and the value of u_k for each sample point $\boldsymbol{p}_k$.

[3]It should be noted that the ICP algorithm is only guaranteed to converge if S_i is a subset of S_j and outlier rejection is disabled (i.e., u_k all set to 1).

2. Using the correspondences $(\boldsymbol{p}_k, \boldsymbol{q}_k)$, compute the $\boldsymbol{\theta}_{\text{new}}$ that minimizes equation 5.3.

3. If $E(\boldsymbol{\theta}_{\text{new}}) < E(\boldsymbol{\theta})$, set $\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{new}}$, otherwise stop iterating.

In our implementation, we use the vertices of the mesh representation of S_i as sample points $\mathcal{P}$. Also, we use the overlap test (2.3) for outlier rejection ($u_k = 1$ iff $T\boldsymbol{p}_k$ overlaps S_j), with the distance threshold updated dynamically according to the method described by Zhang [86].

5.2.2 Point-to-plane matching

The main drawback to the ICP algorithm is slow convergence for certain surfaces. When the two surfaces need to slide relative to one another, point-to-point correspondences act to hold the surfaces in place. Establishing point-to-point correspondences is basically a zeroth-order approximation of the closest point distance D_{CP} (equation 2.1). This suggests an algorithm that uses tangent planes, a first order approximation of D_{CP}. Using point-to-plane correspondences should allow surfaces to slide more easily and lead to faster convergence. Experiments in section 7.3 confirm this prediction.

For point-to-plane matching, $\|T\boldsymbol{p}_k - \boldsymbol{q}_k\|$, the point-to-point distance for fixed correspondences, is replaced with $T\boldsymbol{n}_k \cdot (T\boldsymbol{p}_k - \boldsymbol{q}_k)$, where $\boldsymbol{n}_k$ is the surface normal at $\boldsymbol{p}_k$. The cost function (equation 5.3) becomes

$$E(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \sum_{k=1}^{K} \frac{u_k}{U} \left[T\boldsymbol{n}_k \cdot (T\boldsymbol{p}_k - \boldsymbol{q}_k)\right]^2 \tag{5.4}$$

Unlike the ICP algorithm, where the transform that minimizes E for fixed correspondences can be found in closed form, this algorithm requires non-linear optimization to find the transform at each iteration. Consequently, even though fewer iterations are necessary, each iteration requires more computation than with the ICP algorithm. For this algorithm, we actually use our multi-view point-to-plane registration algorithm (section 5.3.3), which reduces to optimizing 5.4 in the two-view case.

5.3 Multi-view registration

Multi-view registration generalizes pair-wise registration to an entire network of overlapping views, optimizing the absolute pose parameters of all views. Multi-view registration is commonly used in modeling from reality systems as a final step to improve the overall model quality by distributing pair-wise registration errors over the entire model.

A wide variety of multi-view registration algorithms have been proposed. One approach is to update the absolute poses one at a time by registering each view with the adjacent overlapping views in a pair-wise manner. In this vein, Bergevin et. al. register view pairs using a modified version of Chen and Medioni's algorithm [3]. Pulli uses a similar approach based on the ICP algorithm [62]. Alternatively, the absolute poses can be updated simultaneously. Benjemaa and Schmitt derived a multi-view extension of the ICP algorithm [2], and Neugebauer developed a multi-view version of Chen and Medioni's pair-wise algorithm [53]. Another idea is to view the relative poses as constraints on the absolute poses. Lu and Milios derive uncertainty measures (covariance matrices) for the relative poses and solve for the absolute poses that minimize the distance between the computed and measured relative poses according to the Mahalanobis distance metric [50]. Stoddart and Hilton use the analogy of a mechanical system, in which corresponding points are attached by springs, to derive a set of force-based incremental motion equations that are equivalent to gradient descent [74]. Eggert et. al. also use a mechanical system analogy, but they include an inertia term and also update the correspondences over time [20]. Goldberger uses a model-based approach based on the EM algorithm [29]. At each iteration, every view is registered to the current model and then a new maximum likelihood model is created from the updated views.

We have implemented two multi-view registration algorithms that simultaneously update all absolute poses. The first is a novel point-to-point matching algorithm that extends ICP to more than two views, and the second is a point-to-plane matching algorithm that is a modified version of Neugebauer's algorithm [53]. In our derivation, we have used Neugebauer's notation when possible.

5.3.1 Overview of the algorithms

We are given N input views expressed as surfaces $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$, each defined in its own local coordinate system, and a model graph G_{in}. The nodes of G_{in} correspond to the surfaces S, and the edges of G_{in} specify which views overlap. Each edge e_{ij} is annotated with the relative pose T_{ij} that registers the associated view pair. The relative poses are assumed to be approximately correct. G_{in} may contain cycles and is not required to be pose consistent.

Recall that an absolute pose T_i is an operator that transforms points from the local coordinate system of S_i to world coordinates:

$$\boldsymbol{p}' = T_i\boldsymbol{p} \equiv \boldsymbol{R}_i\boldsymbol{p} + \boldsymbol{t}_i, \tag{5.5}$$

where $\boldsymbol{R}_i$ is a rotation matrix and $\boldsymbol{t}_i$ is a translation vector. We parameterize T_i with a vector $\boldsymbol{\theta}_i = (\theta_{ix}, \theta_{iy}, \theta_{iz}, t_{ix}, t_{iy}, t_{iz})^{\mathrm{T}}$, where θ_{ix}, θ_{iy}, and θ_{iz} are rotation angles about the x, y, and z axes (the X-Y-Z Euler angles) and t_{ix}, t_{iy}, and t_{iz} are the components of $\boldsymbol{t}_i$. When necessary, we write T_i as $T_i^{(\theta)}$ to make the parameterization explicit.

Given a set of points $\mathcal{P}_i = \{\boldsymbol{p}_{i1}, \boldsymbol{p}_{i2}, \ldots, \boldsymbol{p}_{iK(i)}\}$ on each surface S_i, we pose the multi-view registration problem as a least squares minimization over the $6N$ absolute pose parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^{\mathrm{T}}\boldsymbol{\theta}_2^{\mathrm{T}}, \ldots, \boldsymbol{\theta}_N^{\mathrm{T}})^{\mathrm{T}}$. We seek to minimize the following cost function:

$$E(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{j \in \operatorname{adj}(i)} \sum_{k=1}^{K(i)} \frac{u_{ijk}}{U} \left[D(T_i\boldsymbol{p}_{ik}, T_jS_j)\right]^2 \tag{5.6}$$

where u_{ijk} is an inlier indicator variable (i.e., $u_{ijk} = 1$ if $\boldsymbol{p}_{ik}$ has a corresponding point on S_j and $u_{ijk} = 0$ otherwise), $U = \sum_{i,j,k} u_{ijk}$ (i.e., the total number of inlier points). D is the distance between point $\boldsymbol{p}_i$ and surface S_j in world coordinates. Its definition depends on whether we are minimizing point-to-point distances or point-to-plane distances and will be specified in the corresponding section below. In either case, the output of the algorithm is a model graph G_{out} with the absolute poses T_i given with respect to a base view $S_b \in \mathcal{S}$.

Both the point-to-point and point-to-plane matching algorithms follow the same basic framework of two nested loops: an outer correspondence update loop and an inner non-linear optimization loop. A set of sample points $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ are randomly chosen with uniform density over all the surfaces. The sampling density determines the number of points, $K(i)$, that will be sampled from each surface. The outer loop is analogous to step 1 of the ICP algorithm (section 5.2.1). At each iteration, correspondences are established between the sample points and each of the adjacent views in G_{in}. The inner loop is an optimization of the cost function $E(\boldsymbol{\theta})$ for fixed correspondences. This is analogous to step 2 of the ICP algorithm. We use the Levenberg-Marquardt algorithm for this optimization.

5.3.2 Point-to-point matching

For point-to-point matching, we define D in equation 5.6 as the closest point distance D_{CP} (equation 2.1), which is the same distance metric used in the ICP algorithm for the two-view case.

Finding correspondences

For each sample point $\boldsymbol{p}_{ik}$ on S_i, we find the closest point $\boldsymbol{q}_{ijk}$ on each surface S_j whose view is adjacent to view i in G_{in}. The first time correspondences are computed, we use the relative poses T_{ji} rather than the absolute poses. This can be accomplished by replacing $D(T_i\boldsymbol{p}_{ik}, T_jS_j)$ in equation 5.6 with $D(T_{ji}\boldsymbol{p}_{ik}, S_j)$. Using relative poses ensures that we use the best correspondences between each pair of views in the event that G_{in} is not pose consistent. Note that each corresponding point $\boldsymbol{q}_{ijk}$ is represented in the coordinate system of view j. If a correspondence passes the overlap test (section 2.3), the inlier indicator variable u_{ijk} is set to 1, otherwise, the point is considered to be an outlier, and u_{ijk} is set to 0. The distance threshold for the overlap test is dynamically updated as the algorithm progresses (see below). Correspondences can be computed efficiently using geometric data structures such as the KD-tree [59] and additional enhancements such as caching correspondences between iterations [71].

Fixed correspondence minimization

With fixed correspondences the distance function D becomes:

$$D(T_i\boldsymbol{p}_{ik}, T_jS_j) = D(T_i\boldsymbol{p}_{ik}, T_j\boldsymbol{q}_{ijk}) = \|T_i\boldsymbol{p}_{ik} - T_j\boldsymbol{q}_{ijk}\| \tag{5.7}$$

It is convenient to divide the transforms T into a known part $T^{(\tau)}$ and a correction part $T^{(\delta)}$ such that $T = T^{(\delta)}T^{(\tau)}$. At each iteration, we transform each sample by $T_i^{(\tau)}$ and each corresponding point by $T_j^{(\tau)}$:

$$\boldsymbol{p}_{ik}^{(\tau)} = T_i^{(\tau)}\boldsymbol{p}_{ik} \tag{5.8a}$$

$$\boldsymbol{q}_{ijk}^{(\tau)} = T_j^{(\tau)}\boldsymbol{q}_{ijk} \tag{5.8b}$$

Combining equations 5.5, 5.6, 5.7, 5.8a, and 5.8b, E becomes:

$$\begin{aligned} E(\boldsymbol{\delta}) &= \min_{\boldsymbol{\delta}} \sum_{i=1}^{N} \sum_{j\in\mathrm{adj}(i)} \sum_{k=1}^{K(i)} \frac{u_{ijk}}{U} \left\| T_i^{(\delta)}\boldsymbol{p}_{ik}^{(\tau)} - T_j^{(\delta)}\boldsymbol{q}_{ijk}^{(\tau)} \right\|^2 \\ &= \min_{\boldsymbol{\delta}} \sum_{i,j,k} \left\| \sqrt{\frac{u_{ijk}}{U}} \left(\boldsymbol{R}_i^{(\delta)}\boldsymbol{p}_{ik}^{(\tau)} + \boldsymbol{t}_i^{(\delta)} - \boldsymbol{R}_j^{(\delta)}\boldsymbol{q}_{ijk}^{(\tau)} - \boldsymbol{t}_j^{(\delta)} \right) \right\|^2 \\ &= \min_{\boldsymbol{\delta}} \sum_{i,j,k} \|F_{ijk}(\boldsymbol{\delta})\|^2 \end{aligned} \tag{5.9}$$

Let K be the total number of correspondences (i.e., $K = \sum_{i=1}^{N}\sum_{j\in\mathrm{adj}(i)} K(i)$). By stacking the 3x1 $\boldsymbol{F}_{ijk}$ vectors into a single $3K$x1 vector $\boldsymbol{F}$, equation 5.9 is placed in standard least squares form:

$$E(\boldsymbol{\delta}) = \min_{\boldsymbol{\delta}} \|F(\boldsymbol{\delta})\|^2 \tag{5.10}$$

We minimize 5.10 using the Levenberg-Marquardt algorithm [60]. Levenberg-Marquardt is a non-linear least squares optimization algorithm that can be viewed as a hybrid between Gauss-Newton and gradient descent. Each iteration of the algorithm computes an update to $\boldsymbol{\delta}$ ($\boldsymbol{\delta}_t$) by solving a system of equations:

$$(\boldsymbol{J}_F^{\mathrm{T}}\boldsymbol{J}_F + \lambda\boldsymbol{I})(\boldsymbol{\delta}_t - \boldsymbol{\delta}_{t-1}) = \boldsymbol{J}_F^{\mathrm{T}}F(\boldsymbol{\delta}_{t-1}) \tag{5.11}$$

where $\boldsymbol{J}_F$ is the Jacobian of F computed at $\boldsymbol{\delta} = \boldsymbol{\delta}_{t-1}$, I is the identity matrix, and λ is the step-size parameter. When λ is small, equation 5.11 reduces to the Gauss-Newton update, and when λ is large, it reduces to a gradient descent update. With transformed correspondences (equations 5.8a and 5.8b), $\boldsymbol{\delta}_{t-1} = 0$, and equation 5.11 reduces to

$$(\boldsymbol{J}_F^{\mathrm{T}}\boldsymbol{J}_F + \lambda\boldsymbol{I})\boldsymbol{\delta}_t = \boldsymbol{J}_F^{\mathrm{T}}F(0) \tag{5.12}$$

which greatly simplifies the form of $\boldsymbol{J}_F$. If we denote the 3×3 identity matrix by $\boldsymbol{I}_3$, and define $[\boldsymbol{p}]_\times$ to be the cross product operator for the point $\boldsymbol{p}$

$$[\boldsymbol{p}]_\times = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \tag{5.13}$$

then the Jacobian of $\boldsymbol{F}_{ijk}$ ($\boldsymbol{J}_{F_{ijk}}$) is a sparse $3\times 6N$ matrix with elements:

$$\boldsymbol{J}_{F_{ijk}}[1{:}3,\ 6i{+}1{:}6i{+}7] = \begin{bmatrix} -\left[\boldsymbol{p}_{ik}^{(\tau)}\right]_\times & \boldsymbol{I}_3 \end{bmatrix} \tag{5.14}$$

and

$$\boldsymbol{J}_{F_{ijk}}[1{:}3,\ 6j{+}1{:}6j{+}7] = \begin{bmatrix} \left[\boldsymbol{q}_{ijk}^{(\tau)}\right]_\times & -\boldsymbol{I}_3 \end{bmatrix} \tag{5.15}$$

and zeros elsewhere. Also, from equation 5.9, we see that

$$\boldsymbol{F}_{ijk}(0) = \boldsymbol{p}_{ik}^{(\tau)} - \boldsymbol{q}_{ijk}^{(\tau)} \tag{5.16}$$

$\boldsymbol{J}_F$ is formed by stacking the individual Jacobians $\boldsymbol{J}_{F_{ijk}}$ into a $3K\times 6N$ matrix. Similarly, the $\boldsymbol{F}_{ijk}(0)$ vectors are stacked to form $\boldsymbol{F}(0)$. In practice, it is not necessary to store these large matrices and vectors explicitly. Instead, the quantities $\boldsymbol{J}_F^{\mathrm{T}}\boldsymbol{J}_F$ and $\boldsymbol{J}_F^{\mathrm{T}}\boldsymbol{F}(0)$ can be computed directly from individual $\boldsymbol{J}_{F_{ijk}}$ and $\boldsymbol{F}_{ijk}$ values. The matrix $\boldsymbol{A} = (\boldsymbol{J}_F^{\mathrm{T}}\boldsymbol{J}_F + \lambda I)$ has a sparse block structure, which can be exploited to speed up the computation of $\boldsymbol{\delta}_t$. If we represent $\boldsymbol{A}$ using 6x6 blocks $\boldsymbol{B}$:

$$\boldsymbol{A} = \begin{bmatrix} B_{11} & \cdots & B_{N1} \\ \vdots & \ddots & \vdots \\ B_{1N} & \cdots & B_{NN} \end{bmatrix} \tag{5.17}$$

```
Point-to-point multi-view registration
Input: Surfaces $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$, model graph $G_{\text{in}}$ with relative poses
specified, sampling density $d$, and base view $V_b$
Output: model graph $G_{\text{out}}$ with absolute poses specified

  determine number of samples: $K(i) = d \cdot \texttt{area}(S_i)$
  sample surfaces: $P_i = \texttt{sample}(S_i, K(i))$
  initialize transforms using $G_{\text{in}}$: $T_i^{(\tau)} = T_{bi}$
  loop
    compute corresponding points $\boldsymbol{q}_{ijk}$ (equation 5.7)
    determine outliers $u_{ijk}$ (overlap test)
    loop
      transform correspondences: $\boldsymbol{p}_{ik}^{(\tau)}$ and $\boldsymbol{q}_{ijk}^{(\tau)}$ (equations 5.8a and 5.8b)
      compute $\boldsymbol{\delta}$ update (equation 5.12)
      update transforms: $T_i^{(\tau)} = T_i^{(\delta)} T_i^{(\tau)}$
  construct $G_{\text{out}}$ from $G_{\text{in}}$ and $T_i^{(\tau)}$
```

Figure 5.3: Pseudocode for the point-to-point multi-view registration algorithm. $\texttt{area}(S)$ is a function that computes the surface area of S, and $\texttt{sample}(S, N)$ samples N points from surface S.

then the only non-zero entries of $\boldsymbol{A}$ will be the diagonal blocks B_{ii} and non-diagonal blocks B_{ij} where $e_{ij} \in G_{\text{in}}$.

In its current form, this minimization problem is underconstrained. The cost function (equation 5.9) remains unchanged if all points and their correspondences are transformed by the same transform T. This effect is related to the gauge freedom problem in bundle adjustment [78] and structure from motion [52]. We use a simple method to supply the required additional constraint – we fix the absolute pose of one view, called the base view, to be the identity transform T_{I}. We apply this constraint by removing the columns for the base view from the Jacobian and explicitly setting the base view transform to T_{I}.

Our algorithm is summarized in figure 5.3.

Outlier elimination

As mentioned previously, outliers are eliminated when the correspondences are computed using the overlap test. Outliers are correspondences that are significantly further apart than the majority of correspondences. In practice, outliers arise from sensor anomalies (such as range errors that occur at the occluding boundaries in a range image), from errors in range shadow removal (see section 8.1.2), or from incorrect correspondences (i.e., point pairs that actually lie on two different physical surfaces).

We use a statistical method described by Neugebauer [53] to dynamically set the distance threshold in the overlap test at each iteration based on correspondences from the previous iteration. If we model the signed distance between corresponding points as an uncorrelated normally distributed random variable with zero mean and variance σ^2, then E should follow a chi-square distribution with $U - 6(N-1)$ degrees of freedom. The expected value of E is

$$\mathrm{E}[E] = \sigma^2(U - 6(N-1)). \tag{5.18}$$

Therefore, we can estimate σ for an individual point using the total residual from the previous iteration:

$$\sigma_t = \sqrt{\frac{E_{t-1}}{U - 6(N-1)}} \tag{5.19}$$

We use 3σ as the distance threshold in our overlap test. For the initial iteration, we use a large threshold $D_{\max}$, which is supplied by the user.

Termination criteria

We place an upper limit on the number of iterations that the algorithm will perform. Additionally, the algorithm terminates if 1) E stops decreasing and 2) the surfaces stop moving. The first criterion is expressed in relative terms – E has stopped decreasing if

$$\frac{E_{t-1} - E_t}{E_{t-1}} < T_{\text{cost}}. \tag{5.20}$$

For estimating surface motion, we compute the bounding box correspondence error (equation 6.6) between the surfaces at iteration $t-1$ and iteration t. Surface S_i has stopped moving if

$$\mathrm{BBCE}(T^{(\theta)}S_i, T^{(\theta_{t-1})}S_i) < T_{\text{motion}}. \tag{5.21}$$

Multi-resolution extension

Multi-view registration can be accelerated with a multi-resolution strategy. We execute the single-resolution algorithm described above multiple times, each time doubling the sampling density and using the final poses from the previous resolution as initial poses.

5.3.3 Point-to-plane matching

The derivation of the point-to-plane matching algorithm follows the same outline as that of the point-point algorithm. The main difference is that an alternate distance function is used, which changes the cost function and Jacobian. For point-to-plane matching, we define D as:

$$D(T_i\boldsymbol{p}_{ik}, T_jS_j) = T_i\boldsymbol{n}_{ik} \cdot (T_i\boldsymbol{p}_{ik} - T_j\boldsymbol{q}_{ijk}) \tag{5.22}$$

where $\boldsymbol{n}_{ik}$ is the surface normal of S_i at $\boldsymbol{p}_{ik}$ and $\boldsymbol{q}_{ijk} = \min_{\boldsymbol{q}\in S_j} \|T_i\boldsymbol{p}_{ik} - T_j\boldsymbol{q}\|$. In other words, we again use the closest point distance for finding correspondences, but instead of using this distance directly in our minimization, we minimize the distance between the normal plane of $\boldsymbol{p}_i$ and its corresponding point $\boldsymbol{q}_{ij}$. Thus, $\boldsymbol{F}_{ijk}$ in equation 5.9 is replaced by:

$$F_{ijk}(\boldsymbol{\delta}) = \frac{u_{ijk}}{\sqrt{U}}\left[\boldsymbol{R}_i^{(\delta)}\boldsymbol{n}_{ik}^{(\tau)} \cdot \left(\boldsymbol{R}_i^{(\delta)}\boldsymbol{p}_{ik}^{(\tau)} + \boldsymbol{t}_i^{(\delta)} - \boldsymbol{R}_j^{(\delta)}\boldsymbol{q}_{ijk}^{(\tau)} - \boldsymbol{t}_j^{(\delta)}\right)\right] \tag{5.23}$$

where $\boldsymbol{n}_{ik}^{(\tau)}$ is the rotated normal vector of $\boldsymbol{p}_{ik}$:

$$\boldsymbol{n}_{ik}^{(\tau)} = \boldsymbol{R}_i^{(\tau)}\boldsymbol{n}_{ik} \tag{5.24}$$

In this case, $\boldsymbol{F}_{ijk}$ is a scalar. The Jacobian of $\boldsymbol{F}_{ijk}$, $\boldsymbol{J}_{F_{ijk}}$, is a $1{\times}6N$ vector with elements:

$$\boldsymbol{J}_{F_{ijk}}[6i{+}1{:}6i{+}7] = -G_{ijk} \tag{5.25}$$

and

$$\boldsymbol{J}_{F_{ijk}}[6\text{j}{+}1{:}6\text{j}{+}7] = G_{ijk} \tag{5.26}$$

and zeros elsewhere, and

$$G_{ijk} = \begin{bmatrix} (\boldsymbol{n}_{ik}^{(\tau)} \times \boldsymbol{q}_{ijk}^{(\tau)}) \\ -\boldsymbol{n}_{ik}^{(\tau)} \end{bmatrix}^{\mathrm{T}} \tag{5.27}$$

5.3.4 Discussion

Multi-view registration requires $O(N)$ time to generate samples (a fixed number of samples per view) and $O(E)$ time to compute correspondences. Finding correspondence with the surface mesh representation is an $O(\log(V))$ operation using a KD-tree data structure, where V is the number of vertices in the mesh [59]. The optimization involves solving a linear system, which is approximately $O(N^3)$. However, the matrix is sparse, depending on the degree of each node. In practice, each view overlaps only a small number of other views, so the degree of the nodes tends to be low.

As previously mentioned, our point-to-plane algorithm is based on Neugebauer's algorithm [53]. Our version differs from his in the following ways:

- *sample points* - Neugebauer computes sample points on a regular grid in the range image for a view. This has the effect of varying sample density on the surface as a function of range and orientation. Surfaces facing the sensor or close to it will be sampled more densely than obliquely viewed or distant surfaces.

- *corresponding point computation* - Neugebauer computes corresponding points by projecting into range images. His method is more efficient than computing closest points in $\mathcal{R}^3$, but it introduces significant error for obliquely viewed surfaces

- *stopping criteria* - Neugebauer uses a statistical test for stopping based on the variance of each unknown. This is a reasonable approach, but it involves inverting a $6(N\text{-}1)\times 6(N\text{-}1)$ matrix, which eliminates the possibility of using sparse methods in the optimization.

Chapter 6

Model evaluation criteria

In this chapter, we discuss the methods for evaluating the success of multi-view surface matching and the component algorithms. When the ground truth is known, we can precisely quantify the errors produced by our algorithms, allowing us to compare competing algorithms objectively. When ground truth is unknown, qualitative evaluation methods are useful for subjectively evaluating the results.

6.1 Quantitative evaluation

We use two types of quantitative error measurements: pose error, which measures the accuracy of the relative or absolute poses output by registration algorithms; and reconstruction error, which measures the fidelity of the reconstructed shape to the original scene.

6.1.1 Correspondence error

Given a ground truth transform, T_{GT}, and a transform estimate from registration, T_{est}, we define the error transform as

$$T_{\mathrm{err}} = T_{\mathrm{GT}}^{-1} T_{\mathrm{est}}. \tag{6.1}$$

The question is then how to compute the "magnitude" of T_{err}. A common solution is to compute separate norms for the rotation and translation components. If we represent the

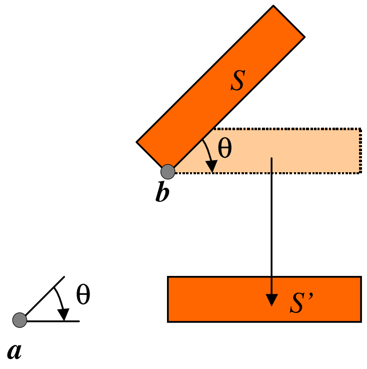


Figure 6.1: Origin dependency of pose error. Aligning S with S' using the coordinate system origin a results in zero translation error, while the same operation using the coordinate system origin b has a large translation error.

rotation component of T_{err} using Euler angles (i.e., $T_{\text{err}} = (\omega_x, \omega_y, \omega_z, t_x, t_y, t_z)$), then the translation and rotation error norms are defined as

$$E_t = \sqrt{t_x^2 + t_y^2 + t_z^2} \tag{6.2}$$

$$E_\omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{6.3}$$

Equation 6.3 is an approximation that is valid for small angles (i.e., those where $sin(\omega) \approx \omega$). For larger rotation errors, the rotation component of T_{err} can be represented as a rotation θ about an axis $\hat{K}$. This is called the equivalent angle-axis representation [14]. In this case, the rotation error is magnitude of the rotation angle ($E_\omega = |\theta|$).

There are several problems with this approach. First, since the units of the two norms are not the same, it is not obvious how to combine E_t and E_ω. If one algorithm outputs a transform with a small E_t and a large E_ω while a second algorithm gives a transform with large E_t and small E_ω, which solution is superior? A second problem is that the magnitude of E_t depends on the choice of coordinate system origin. For example, consider the transform associated with aligning S with S' in figure 6.1. If the coordinate system origin located at a is used, E_t is zero, but if the origin located at b is used, E_t is non-zero. In both cases E_ω remains the same.

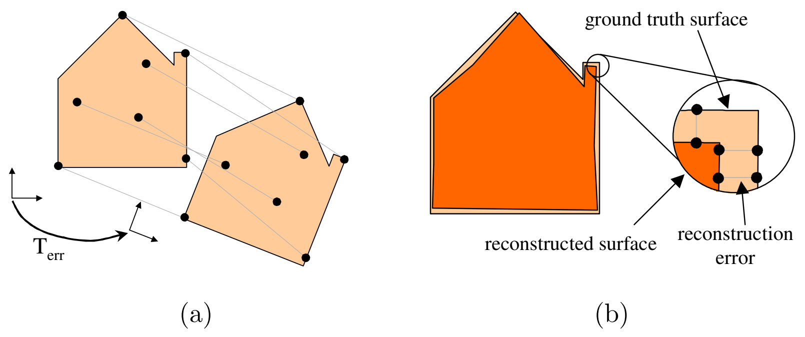


Figure 6.2: a) The maximum correspondence error (MCE) for a surface is the maximum displacement of any point on the surface from its ground truth position. b) The maximum reconstruction error (MRE) is the maximum distance of any point on the reconstructed surface to the ground truth surface.

Rather than separating T_{err} into translation and rotation components, we use a unified, data-oriented error measure proposed by Simon [70] called correspondence error. The correspondence error for a point $\boldsymbol{p} \in \mathbb{R}^3$ is the Euclidean distance between the position of $\boldsymbol{p}$ when transformed by T_{est} and the position of $\boldsymbol{p}$ when transformed by T_{GT}. In other words, correspondence error is the displacement of the point from its true position. Equivalently, this is the displacement D of $\boldsymbol{p}$ when transformed by T_{err} ($D(\boldsymbol{p}, T) = \|T\boldsymbol{p} - \boldsymbol{p}\|$).

We can extend the definition of correspondence error to a surface S by computing statistics of D over the surface (figure 6.2a). We define the maximum correspondence error (MCE) as

$$\text{MCE}(S, T) = \max_{\boldsymbol{p} \in S} D(\boldsymbol{p}, T) \tag{6.4}$$

Theorem 6.1. *The MCE for a triangle mesh $\mathcal{M}$ occurs at one of the vertices of $\mathcal{M}$.*

Proof: The displacement $D(\boldsymbol{p}, T)$, is a convex function of $\boldsymbol{p}$ (i.e., $D(a\boldsymbol{p} + (1-a)\boldsymbol{p}, T) \leq aD(\boldsymbol{p}, T) + (1-a)D(\boldsymbol{q}, T)$ for $a \in [0, 1]$). The surface of a 3D triangle is a convex domain (e.g., using Barycentric coordinates). Therefore, the maximum of D, a convex function, over a mesh face, a convex domain, must be attained at one of the face's corners. Since $\mathcal{M}$ is composed exclusively of faces, the maximum of D over $\mathcal{M}$ must occur at the corner of some face in $\mathcal{M}$, which, by definition, is a vertex of $\mathcal{M}$. $\square$

Maximum correspondence error is an intuitive measure of registration accuracy for meshes. For example, if the MCE for a mesh is 5mm, then no point on that mesh is displaced by more than 5mm from its true location. If the goal is to reconstruct the scene with 1cm accuracy, this error level is sufficiently small. In contrast, when using translation and rotation errors, it is not immediately clear what will be the effect of a 2° rotation error and a 3mm translation error. Depending on the size of the mesh, location of the coordinate system origin, and the axis of rotation, this level of error may or may not be significant.

Because maximum correspondence error is a worst case error and because we can compute it exactly, we use it extensively in the analysis of registration algorithm performance. Computation of the MCE is straightforward when analyzing the absolute pose error of a view. We simply evaluate equation 6.4 using the error transform T_{err} and the vertices of the view's surface mesh.

When analyzing relative pose errors, two views are involved. The error obtained using the vertices of the first view will be different from that obtained using those of the second view. We solve this problem by defining the relative pose maximum correspondence error (RMCE) to be the maximum of the two individual measures:

$$\text{RMCE}(S_i, S_j, T_{i,j,\text{err}}) = \max(\text{MCE}(S_i, T_{i,j,\text{err}}), \text{MCE}(S_j, T_{i,j,\text{err}}^{-1})) \tag{6.5}$$

Maximum correspondence error computed using all vertices of a mesh is an $O(N)$ operation. We can compute an upper bound on the correspondence error in constant time by computing the displacement of the vertices of the mesh's bounding box[1]. We call this the bounding box correspondence error (BBCE):

$$\text{BBCE}(S, T) = \max_{\boldsymbol{p} \in \mathcal{B}} D(\boldsymbol{p}, T) \tag{6.6}$$

where $\mathcal{B}$ is the set of vertices of the bounding box of surface S. The proof that the BBCE is an upper bound for MCE is essentially the same as the proof of theorem 6.1. We have

[1]Computation of the bounding box is itself $O(N)$, but it only needs to be computed once.

not proven the tightness of this bound, but in practice, the BBCE is about 40% larger than the associated MCE. We use the BBCE as a stopping criterion for pair-wise and multi-view registration (see section 5.3). To make MCE and BBCE independent of scale, we generally normalize these measures by the size of the surface used for the calculation.

6.1.2 Reconstruction error

While correspondence error is well-suited for measuring registration errors, our ultimate goal is to create geometrically accurate models. Reconstruction error captures this notion of geometric fidelity. We define the reconstruction error for a point $\boldsymbol{p}$ on the reconstructed surface S_{R} to be minimum distance from $\boldsymbol{p}$ to the ground truth surface S_{GT} (figure 6.2b)[2]. This is simply the closest point distance D_{CP} (equation 2.1). We then define the maximum reconstruction error (MRE) for the surface S_{R} as

$$\mathrm{MRE}(S_{\mathrm{R}}, S_{\mathrm{GT}}) = \max_{\boldsymbol{p} \in S_{\mathrm{R}}} D_{\mathrm{CP}}(\boldsymbol{p}, S_{\mathrm{GT}}) \tag{6.7}$$

Normalizing the MRE by the size of S_{GT} makes the measure scale-independent. Unlike the MCE, it is not possible to compute the exact MRE using only the vertices of a surface's mesh. In practice, we compute the MRE using a dense sampling of points from S_{R}. If desired, we can compute other statistics of the reconstruction error, such as the average or RMS error.

6.2 Qualitative evaluation

When ground truth is not available, we can subjectively judge the success of multi-view surface matching by visually inspecting the registered surfaces or the reconstructed model. Since we have *a priori* knowledge of the scene shape, visual inspection is generally adequate to determine whether the poses of a model hypothesis are correct. If necessary, we examine cross-sections of the registered views to verify the alignment of small features. For data sets with unknown ground truth, we use qualitative evaluation to determine the correct absolute

[2]We are assuming the two surfaces are registered with one another.

poses, which we then use as a proxy for the ground truth. Although we can never know with certainty whether our proxy ground truth is correct, we can at least determine whether different algorithms (or repeated trials of the same algorithm) find the same solution.

We can also subjectively judge the success of our overall modeling from reality process by qualitatively evaluating the reconstruction. Gross reconstruction errors (e.g., those that arise from incorporating an incorrect match into the model) are obvious. There are, however, methods to judge the fine geometric accuracy of models that contain only correct matches. First, we can compare small features on the reconstructed object. Reconstruction errors will eliminate or blur small features before affecting large ones. Second, we can view the original and reconstructed objects from the same point of view and compare the shape of the silhouettes. Finally, we can examine the result of texture mapping (section 8.2.2). Incorrect geometry often results in incorrect coloring of the surface. For example, if part of the reconstructed surface extends beyond the true boundary of the object, pixels from the background will be projected onto the object.

Chapter 7

Experiments

In this chapter, we present experimental results using real and synthetic data and demonstrate the versatility of our automatic modeling from reality framework. We begin with a description of our method for synthesizing test data (section 7.1). Next, we detail experimental results, beginning with the component algorithms in sections 7.2 and 7.3 and following with tests of our local and global quality measures (section 7.4) and our multi-view surface matching algorithms (section 7.5). Finally, in section 7.6, we demonstrate the generality and robustness of our framework by automatically modeling a variety of scene types, such as terrain and building interiors, covering a range scales from 0.2 meters to 200 meters, and using three different range sensors.

7.1 Generating real and synthetic test data

The experiments in sections 7.2 through 7.5 use range data from real and synthetic objects. The views of the real objects were collected using the hand-held modeling technique (section 1.4) with the Vivid scanner. Since the ground truth poses for these objects are not known, we use the poses and geometry of a qualitatively correct reconstruction of each object as a proxy for the ground truth. The real test objects are shown in figure 7.1.

For ground truth analysis, we use synthetic range images of digital objects. The objects

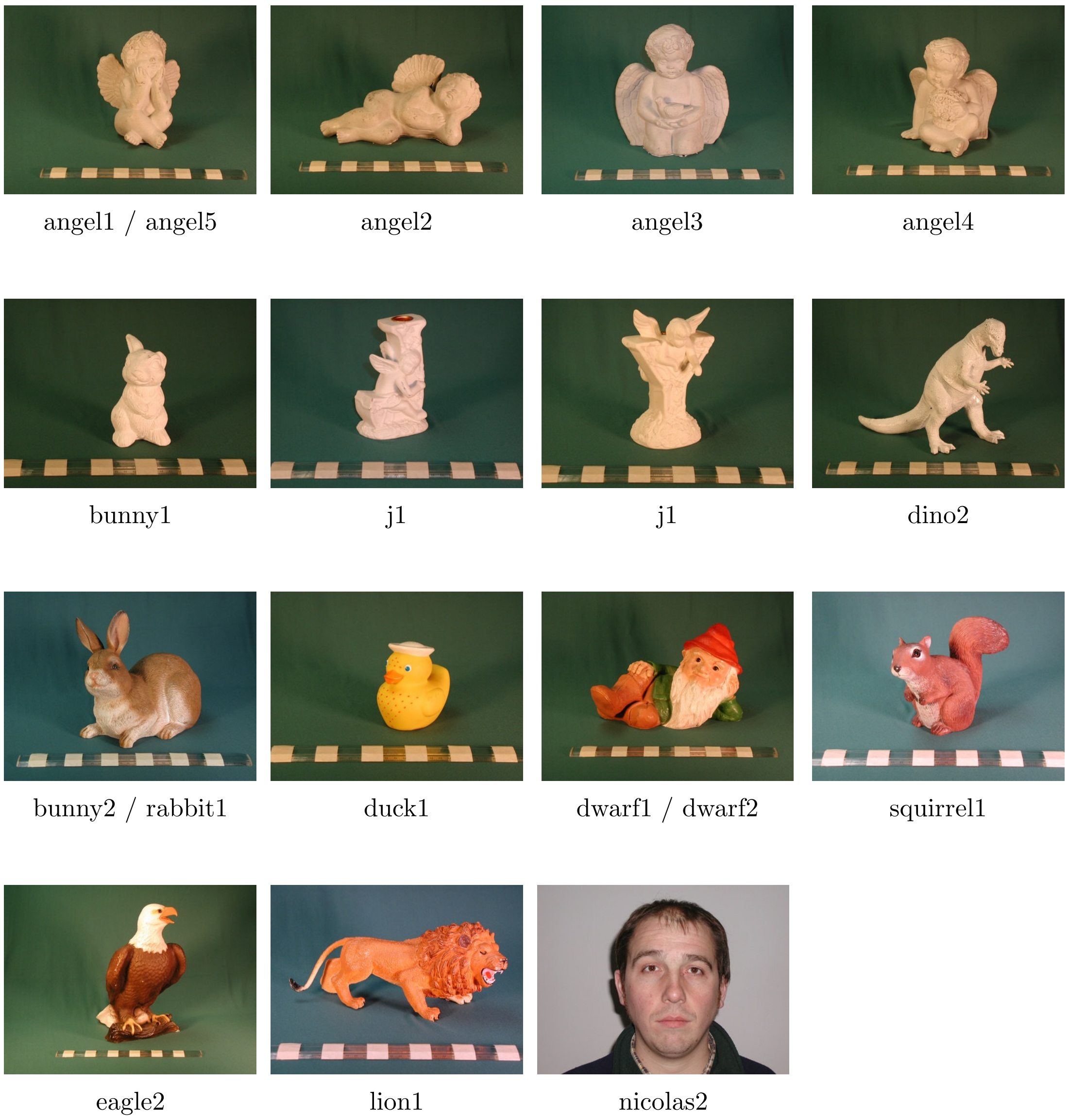


Figure 7.1: Real test objects used in the experiments along with the name of the test set. Objects with multiple names were used to create multiple test sets. A ruler with inch-long markings is shown for scale.

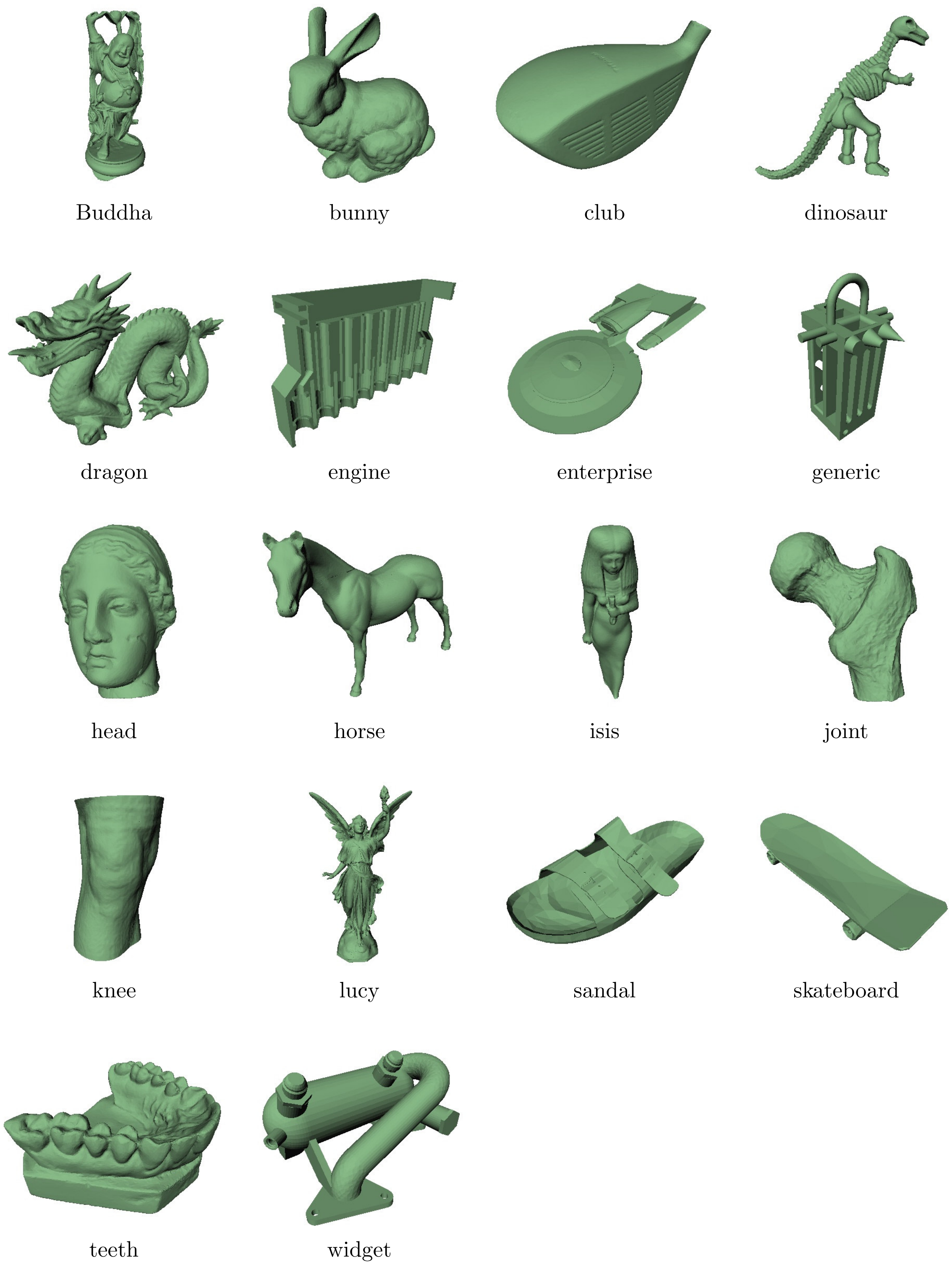


Figure 7.2: Synthetic test objects used in the experiments.

Model archive	URL
Stanford University	www-graphics.stanford.edu/data/3Dscanrep/
Georgia Tech	www.cc.gatech.edu/projects/large_models/
Cyberware	www.cyberware.com/samples/index.html
Northwestern University	www.cs.northwestern.edu/˜watsonb/school/ teaching/351/Models/

Table 7.1: Internet sources of ply format models

Iteration	Number of cameras	Mean angle (degrees)	Min. angle (degrees)	Max angle (degrees)
0	8	70.5	70.5	70.5
1	32	31.8	31.4	32.2
2	128	15.8	12.0	18.6
3	512	7.9	5.5	9.9

Table 7.2: Statistics of the angle between adjacent cameras on the viewing sphere for several iterations of the tesselation algorithm.

were obtained from 3D model archives on the Internet (table 7.1). Each object was scaled to be 200mm in size, making it roughly the same size as the objects scanned with the Vivid. For computational efficiency, the original meshes were simplified to 40,000 faces using Garland's algorithm [27], a process which does not significantly affect the shape or appearance of the objects. The digital test objects are shown in figure 7.2.

We generate synthetic range images from a set of viewpoints distributed approximately evenly over the surface of a sphere. A tessellation of a sphere is formed by repeatedly subdividing an octahedron. Specifically, we begin with an octahedron inscribed within the unit sphere, and subdivide each edge, forming four triangles from each original triangle. The newly created edges are then re-projected onto the unit sphere, and cameras are placed at the center of each face, pointing toward the origin. The distribution of cameras for the first few iterations is shown in table 7.2. After one iteration, we have 32 cameras spaced approximately 32 degrees apart, which is the level of tessellation used in our experiments.

Our synthetic range images simulate the Vivid scanner operating under worst case conditions. They are generated by rendering the object from a given camera position and

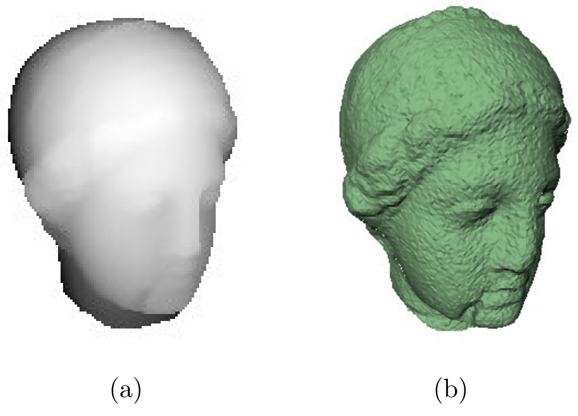


Figure 7.3: a) An example synthetic range image from the head test object. b) The resulting surface mesh with noise added to simulate the worst case performance of the Vivid.

converting the resulting z-buffer into a range image. To more accurately model range image formation, we add Gaussian noise to the range measurements. We have implemented two noise models: one in which the standard deviation (σ) is constant, and a more sophisticated model in which σ varies with range and focal length according to our noise analysis of the Vivid (see equation A.1 in appendix A.1.3). Since our synthetic test objects have a relatively shallow depth of field, the choice of varying or fixed noise model does not significantly affect our results. In our experiments, we use $\sigma = 1$mm, which simulates the noise level for viewing dark surfaces at a distance of one meter (see appendix A.1.3).

7.2 Pair-wise surface matching experiments

In this section, we describe experiments relevant to pair-wise surface matching:

- **face-based spin-images** – verify the claim that face-based spin-images are a more accurate spin-image representation and are resolution-independent
- **mesh resolution** – analyze the effect of mesh-resolution on surface matching performance
- **spin-image support size** – determine the size of spin-images for best performance
- **grid-size** – analyze the effect of face-based spin-image grid-size on surface matching performance

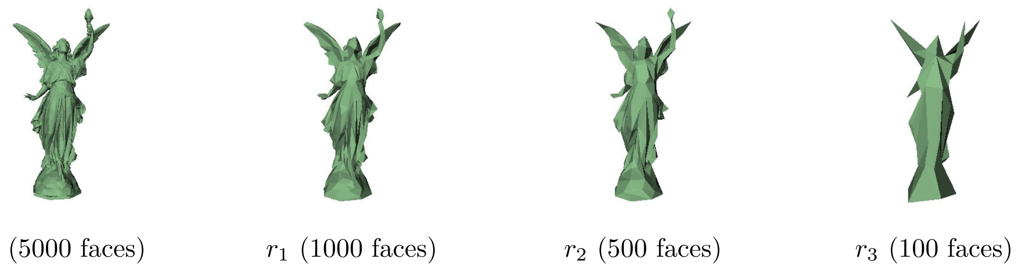


Figure 7.4: A resolution hierarchy of meshes used in the face-based spin-image experiment.

- **noise** – analyze surface matching performance under controlled noise circumstances
- **overlap** – determine how much overlap is needed to successfully register two surfaces

In these experiments, unless otherwise noted, we used square 15×15-bin spin-images with a support size of 50mm (i.e., $0 \leq \alpha < 50$, $-25 \leq \beta < 25$). We used face-based spin-images with an interpolation density (grid-size) of 1.67mm (half the spin-image bin size). Input mesh resolution was 1000 faces. Matches were evaluated using the relative pose maximum correspondence error (RMCE) (equation 6.5) and classified as correct if RMCE $<$ 10mm (5% of the model size), ambiguous if 10mm $\leq$ RMCE $<$ 20mm, and incorrect otherwise. In cases where ambiguous matches are not listed separately, they are classified as incorrect.

7.2.1 Effect of face-based spin-images

In this experiment, we show that when comparing surfaces of different resolutions, face-based spin-images for corresponding points are more similar than vertex-based spin-images for the same points.

We used Garland's mesh simplification algorithm [27] to express a test object (lucy) at several resolution levels (figure 7.4): r_0 (5000 faces, 2500 vertices), r_1 (1000 faces, 500 vertices), r_2 (500 faces, 250 vertices), and r_3 (100 faces, 50 vertices). Let $I_{\mathrm{v}}(\boldsymbol{b}, r)$ be the vertex-based spin-image for basis point $\boldsymbol{b}$ computed for resolution level r and $I_{\mathrm{f}}(\boldsymbol{b}, r)$ be the analogous face-based spin-image. We used the vertices of the highest resolution mesh (r_0)

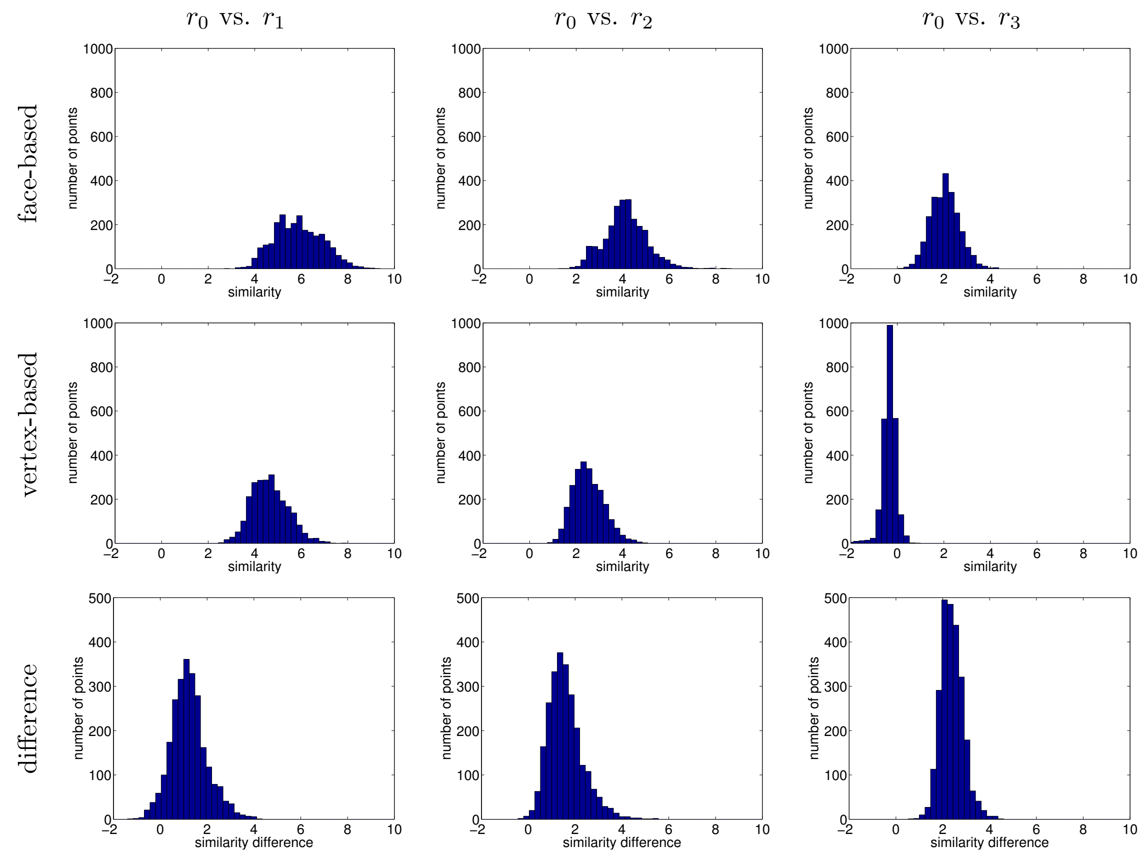


Figure 7.5: Spin-image similarity for corresponding points. The distribution of similarity using face-based spin-images (top row), vertex-based spin-images (middle row), and the difference in similarity (face-based - vertex-based) for each sample point (bottom-row). The experiment compares spin-images from a high-resolution mesh (r_0) to several lower resolution meshes – medium resolution (r_1) (left column), low resolution (r_2) (center column), and very low resolution (r_3) (right column).

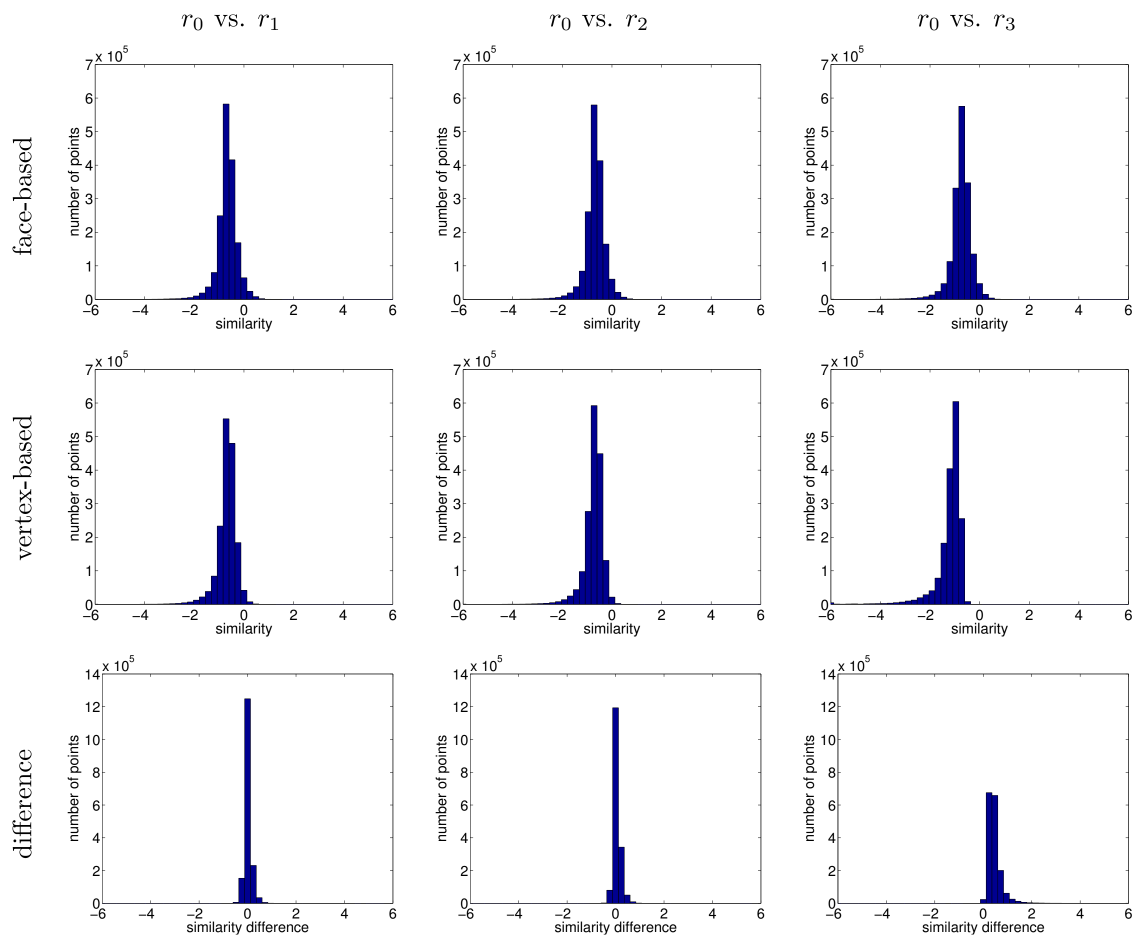


Figure 7.6: Spin-image similarity for non-corresponding points. The distribution of similarity using face-based spin-images (top row), vertex-based spin-images (middle row), and the difference in similarity (face-based - vertex-based) for each sample point (bottom-row). The experiment compares spin-images from a high-resolution mesh (r_0) to several lower resolution meshes – medium resolution (r_1) (left column), low resolution (r_2) (center column), and very low resolution (r_3) (right column).

as basis points for computing I_{f} and I_{v} for each resolution level. Since the meshes are (by default) registered with one another, the spin-images for the same basis point for different resolution levels are spin-images for corresponding points. Let $C(I, J)$ be the similarity measure for spin-images I and J as defined in [36].

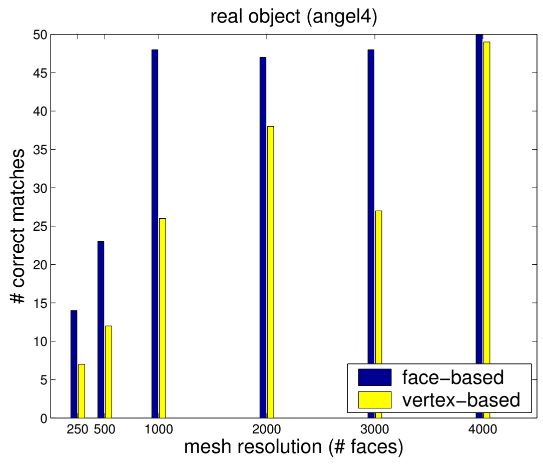
We computed the similarity between spin-images for corresponding points in r_0 and each of the lower-resolution meshes (r_1, r_2, and r_3) first using vertex-based spin-images and then using face-based spin-images (figure 7.5). For example, the top left plot shows the distribution of $C(I_{\mathrm{f}}(\boldsymbol{b}, r_0), I_{\mathrm{f}}(\boldsymbol{b}, r_1))$. The results show that face-based spin-images improve the similarity of corresponding points and that the improvement increases with decreasing mesh resolution. Specifically, the median increase in similarity when using face-based spin-images was 1.18 for r_1 (figure 7.5, bottom left), 1.49 for r_2 (figure 7.5, bottom center), and 2.34 for r_3 (figure 7.5, bottom right).

An increase in similarity for corresponding points is only useful if the similarity of non-corresponding points does not increase equally. To verify this, we computed the same quantities for non-corresponding points[1] (figure 7.6). For non-corresponding points, the median increase in similarity when using face-based spin-images was 0.0036 for r_1 (figure 7.6, bottom left), 0.0334 for r_2 (figure 7.6, bottom center), and 0.397 for r_3 (figure 7.6, bottom right). The results show that the change in similarity of non-corresponding points is minimal in comparison to the change that occurs with with corresponding points.

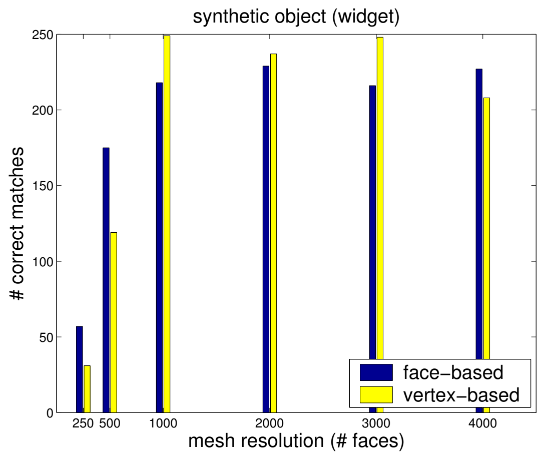
7.2.2 Effect of resolution on matching

The next question we address is the choice of mesh resolution. Since brute force matching is $\mathrm{O}(N^2)$ in the number of faces, it is in our interest to perform this operation at the lowest resolution possible without reducing the number of correct matches. Alternatively, we could use more advanced high-dimensional nearest neighbor algorithms, such as locality sensitive hashing [28], to reduce the computational complexity . This is an area of ongoing research.

[1]We define non-corresponding points as pairs of points separated by a distance of at least 25% of the object's size.



(a)



(b)

Figure 7.7: Effect of mesh resolution on surface matching performance for a real object (a) and a synthetic object (b). Performance falls off sharply for resolutions below 1000 faces.

In this experiment, we performed exhaustive pair-wise registration for the views of one synthetic object (widget) and one real object (angel4) while varying the mesh resolution from 250 to 4000 faces. The results, shown in figure 7.7, reveal a sharp drop-off in the number of correct matches for resolutions below 1000 faces. For these objects, this is the level of simplification at which the geometric fidelity of the mesh can no longer be maintained.

We performed the experiment twice (once with vertex-based spin-images and once with face-based ones) to determine whether the improvement in similarity from face-based spin-images translates into improved matching performance. Surprisingly, face-based spin images do not always perform better. A potential reason for this lies in the surface matching process, which uses a dynamically set threshold to eliminate low-quality correspondences. The similarity improvement of face-based spin images causes the algorithm to choose a more conservative threshold, and some additional correct correspondences are removed. For some difficult view pairs, these correspondences mean the difference between success and failure.

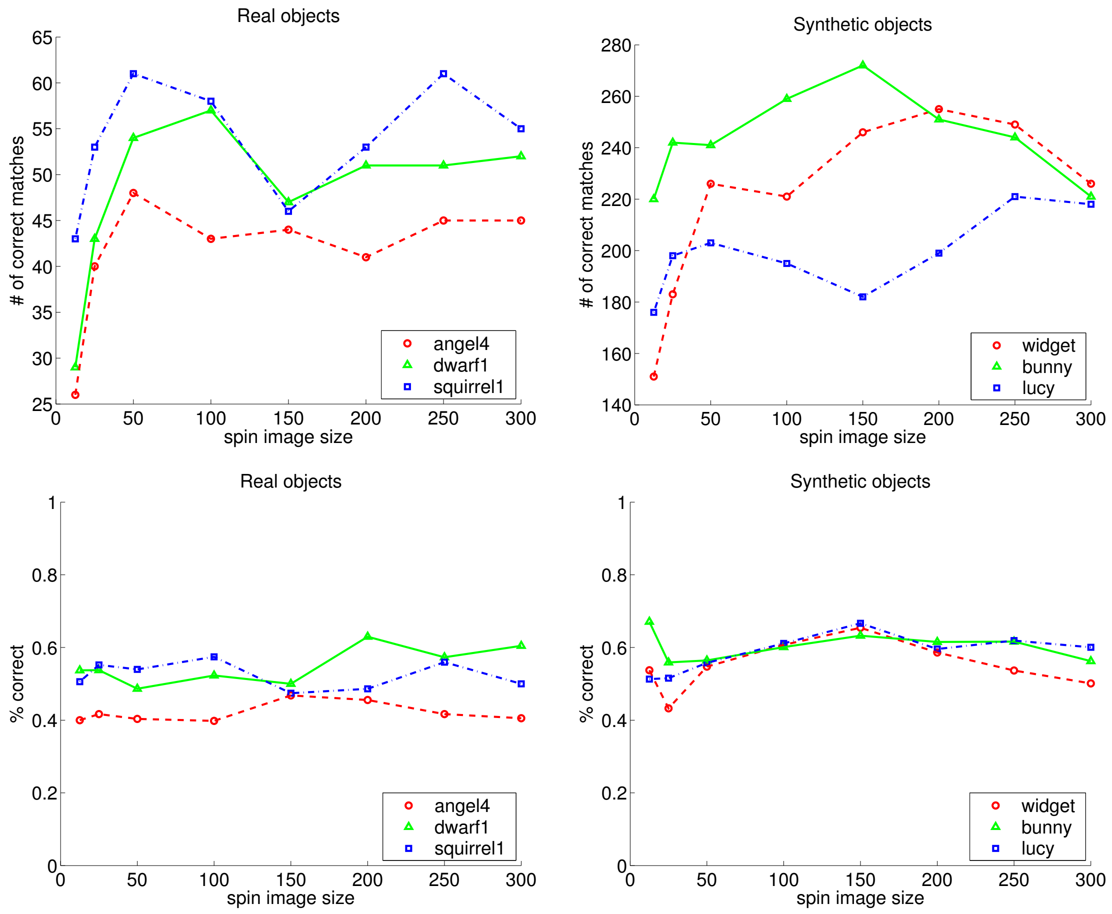


Figure 7.8: Effect of spin-image support size on surface matching performance for real objects (left column) and synthetic objects (right column). The absolute number of correct matches (top row) falls off for support sizes below 25% of the model size but the percentage of correct matches (bottom row) is independent of support size.

7.2.3 Choice of spin-image support size

This experiment shows that pair-wise surface matching performance is largely unaffected by the support size. We performed exhaustive pair-wise registration for the views of three synthetic objects (widget, bunny, and lucy) and three real ones (angel4, gnome, and squirrel1) while varying the spin image support size (limits of α and β parameters) from 12.5mm (1/16th of the model size) to 300mm (1.5 times the model size).

The results, shown in figure 7.8, demonstrate that matching performance is insensitive to the spin-image support size over a wide range. Only for the smallest settings does the number of correct matches begin to drop. The number of correct matches depends more on the shape of the the object than on the support size. The percentage of correct matches is typically between 40 and 60% regardless of support size, which means that the number of incorrect matches is also stable.

7.2.4 Choice of grid-size

Face-based spin-images have one parameter that must be set – the grid-size, which is the density of interpolated points on the mesh faces. A smaller grid-size should lead to an improved success rate but at the cost of speed of spin-image generation. Therefore, we want to choose the largest grid-size possible without reducing the number of correct matches.

For this experiment, we used the same three real and synthetic objects as in the last experiment. We performed exhaustive pair-wise surface matching while varying the grid-size from 0.42mm (25% of the spin-image bin size) to 13.4mm (8 times the bin size). The results (figure 7.9) show that, with the exception of dwarf1, all of the objects have a downward trend in the number of correct matches with increasing grid-size. As with the support size experiment, the percentage of correct matches is essentially constant over the range.

7.2.5 Effect of noise

For this experiment, we performed pair-wise surface matching while varying the the amount of additive Gaussian noise injected into our synthetic range images. As expected, the per-

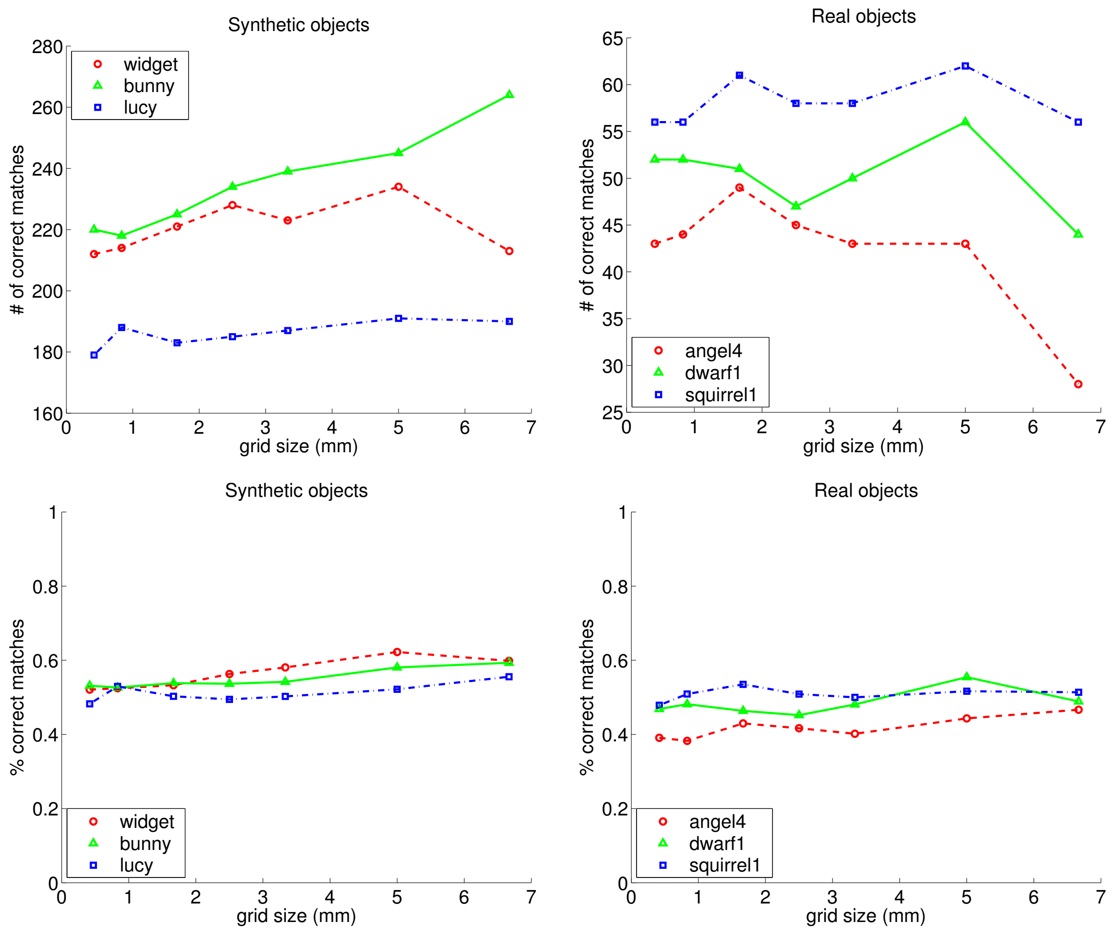


Figure 7.9: Effect of grid-size on surface matching performance for real objects (left column) and synthetic objects (right column). The absolute number of correct matches (top row) generally decreases with increasing grid-size, while the percentage of correct matches (bottom row) is relatively independent of grid-size.

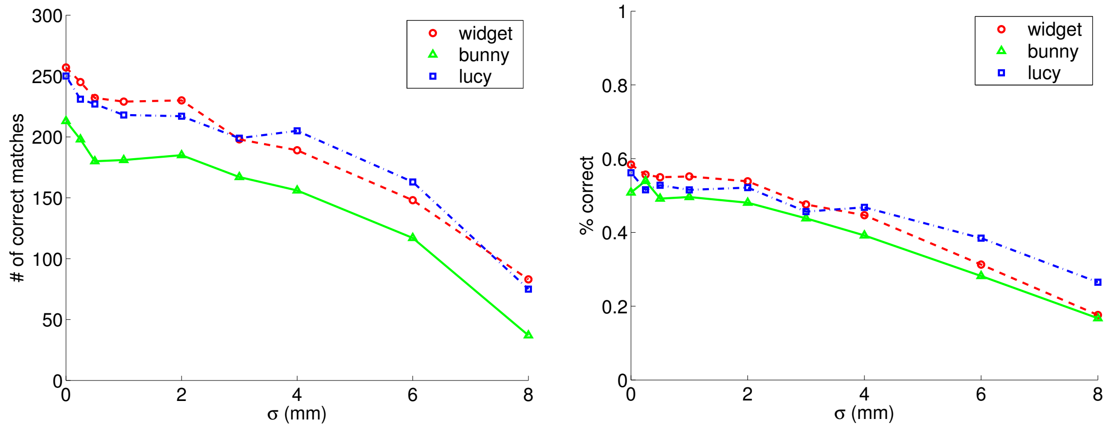


Figure 7.10: Effect of additive Gaussian noise on surface matching performance for three synthetic objects. The number of correct matches (left) and the percentage of correct matches (right) both drop off gradually with increasing noise levels. Typically σ <1mm for real objects scanned with the Vivid.

formance is reduced with increased noise (figure 7.10); however, the algorithm still performs well at noise levels well above those that occur with the real scanner. Unlike the previous experiments, the percentage of correct matches also declines with increasing noise.

7.2.6 Amount of overlap needed for success

The final experiment in this section examines the relationship between amount of overlap and matching success. This information is useful for users, who need to know how much they can move the object (or scanner) between scans.

For each pair of views in each test object, we computed the amount of overlap (section 2.3) using the ground truth relative poses. This is the ground truth overlap. We then performed exhaustive pair-wise registration and classified the results as correct, incorrect, or ambiguous. We eliminated view pairs with no ground-truth overlap, and generated a histogram of the remaining matches as a function of percentage of overlap. The results (figure 7.11) indicate that performance is data-dependent (as expected), but we can draw some general conclusions. A ground-truth overlap of over 50% virtually ensures a correct match will be found, while an overlap of less than 30% is unlikely to correctly match. Between 30

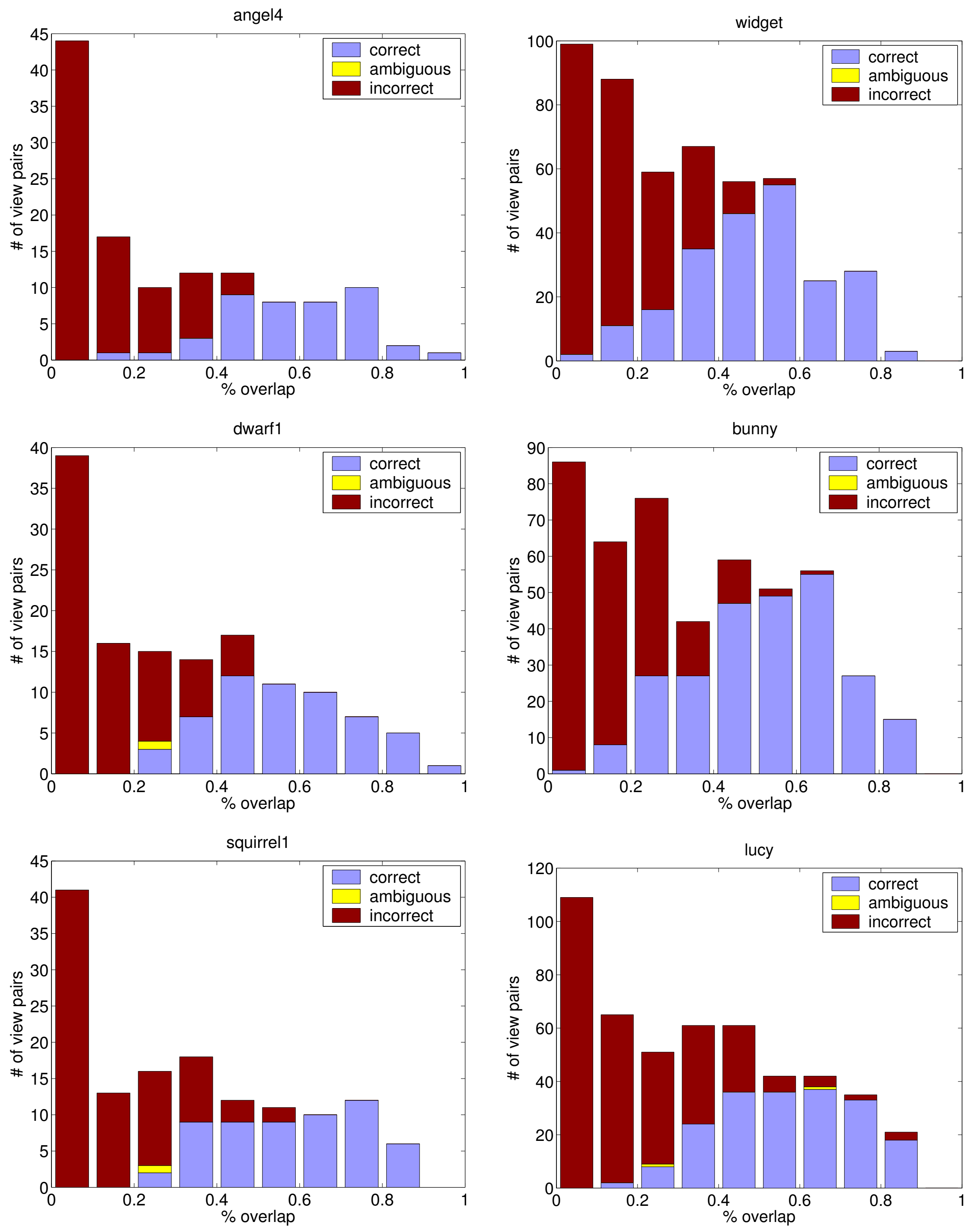


Figure 7.11: Surface matching performance as a function of the amount of overlap for three real objects (left column) and three synthetic ones (right column). The results indicate that users should try to maintain at least 50% overlap between view pairs to ensure successful matching.

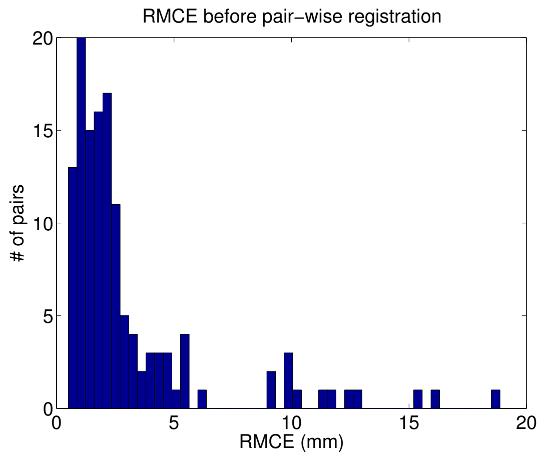
and 50%, the probability of a correct match changes rapidly. Based on these figures, users should try to maintain about 50% overlap between pairs of views.

7.3 Pair-wise registration experiments

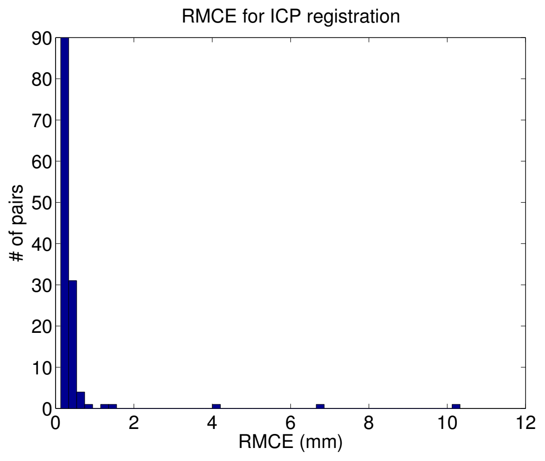
In this section, we briefly investigate the performance of our pair-wise registration algorithms. Since these algorithms have been well-studied by other researchers, we limit our experiments to a brief comparison of point-to-point and point-to-plane algorithms. For the point-to-point algorithm, we used a standard implementation of the iterative closest point (ICP) algorithm as described in [36]. For the point-to-plane algorithm, we use the two-view version of the multi-view point-to-plane registration algorithm described in section 5.2.2. We also tested using the two-view version of our multi-view point-to-point algorithm, but it did not perform as well as ICP because it does not take advantage of the fact that the optimal alignment for fixed correspondences can be solved in closed form for the two view case.

We tested the two algorithms by performing pair-wise registration on all pair-wise matches from three synthetic objects (widget, bunny, and lucy). The accuracy was evaluated using the relative pose maximum correspondence error (RMCE) (equation 6.5) for each object. The results for the widget object are shown in figure 7.12. The point-to-plane algorithm performed better in general (median improvement 41%), although for a few matches, the ICP algorithm fared better.

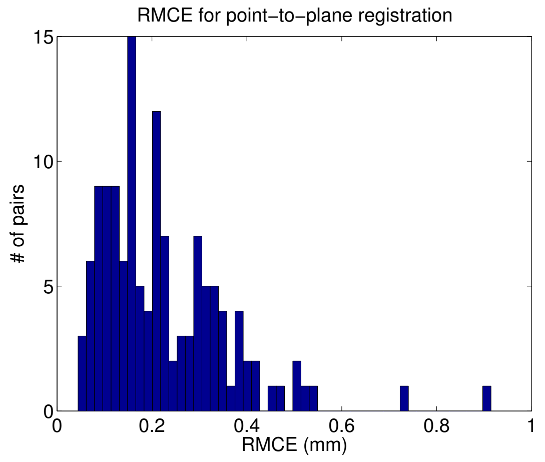
We also compared the speed of the two algorithms. Figure 7.13 shows a sample trace of each algorithm, both in terms of number of iterations and time of execution. The point-to-plane method is several orders of magnitude faster. This illustrates the slow convergence problem that we described in section 5.2.2.



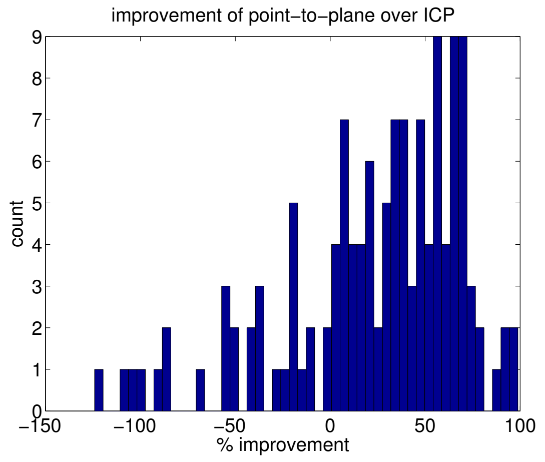
(a)



(b)

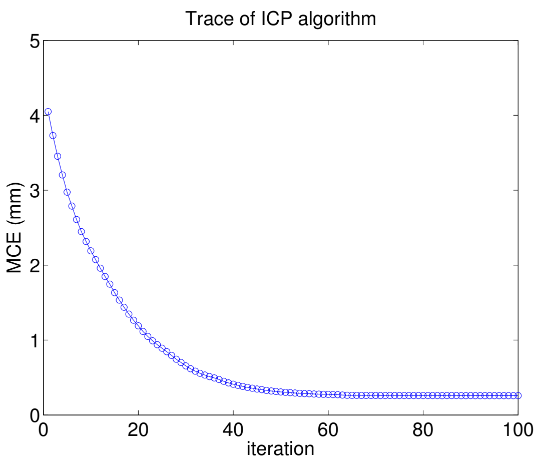


(c)

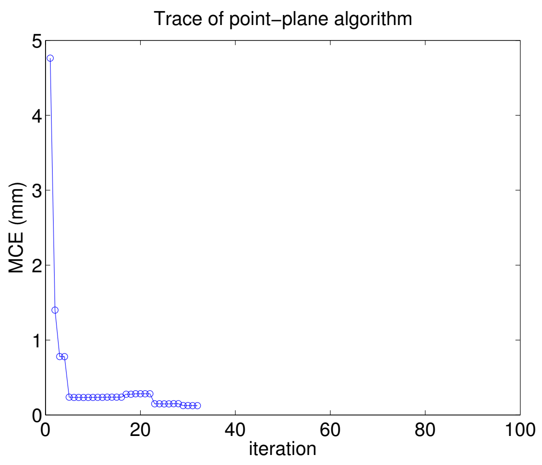


(d)

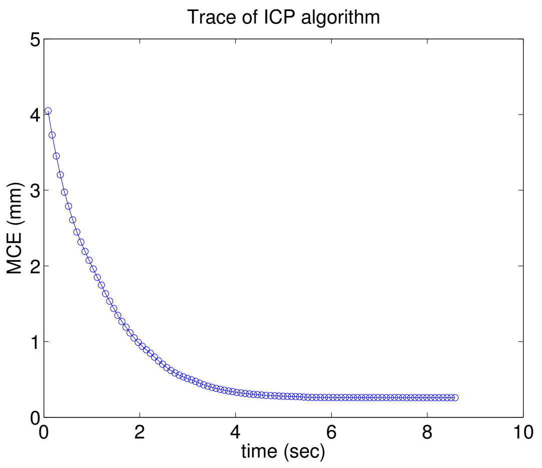
Figure 7.12: Accuracy of point-to-point pair-wise registration vs. point-to-plane registration for the widget test set. a) Initial relative pose error for correct matches (the output of exhaustive pair-wise surface matching). b) Relative pose error after running the ICP algorithm (point-to-point matching). c) Relative pose error after running the point-to-plane matching algorithm. d) Percentage improvement of the point-to-plane algorithm over the point-to-point algorithm.



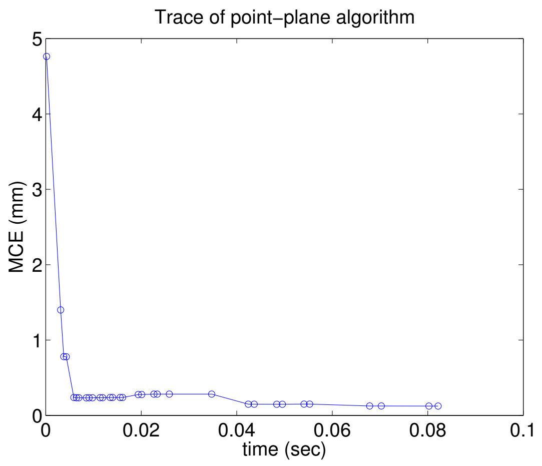
(a)



(b)



(c)



(d)

Figure 7.13: Trace of pair-wise registration for a typical execution of the ICP algorithm (left column) and the point-to-plane algorithm (right column). The top row shows the relative pose error as a function of iteration number, and the bottom row shows it as a function of time. The trace is for views 1 and 15 of the bunny data set.

7.4 Local and global quality measure experiments

In this section, we describe experiments relevant to our local and global surface consistency measures.

- Local quality measure comparison
- Verify global quality is maximized at the ground truth solution
- Verify ground truth solution is globally consistent

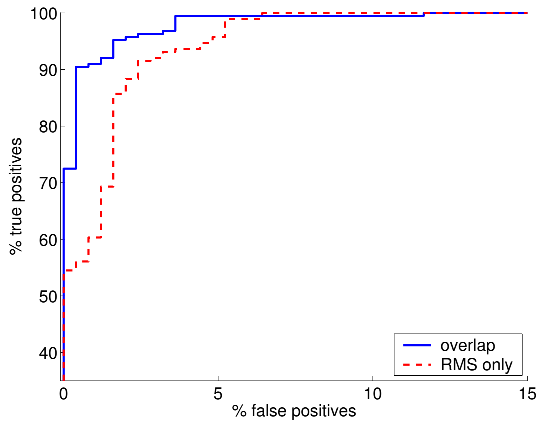
7.4.1 Local quality measure comparison

One method for comparing our local quality measures is by comparing the performance of the corresponding local consistency classifier (C_{L}) using ROC curves. An ROC curve shows the trade-off between the false positive rate (i.e., percent of incorrect matches classified as correct) and the true positive rate (i.e., percent of correct matches classified as correct) as the parameter λ is varied. A perfect classifier achieves 100% true positives with 0% false positives – the upper left corner in an ROC plot. Given two ROC curves, the curve that is higher and to the left corresponds to the better classifier.

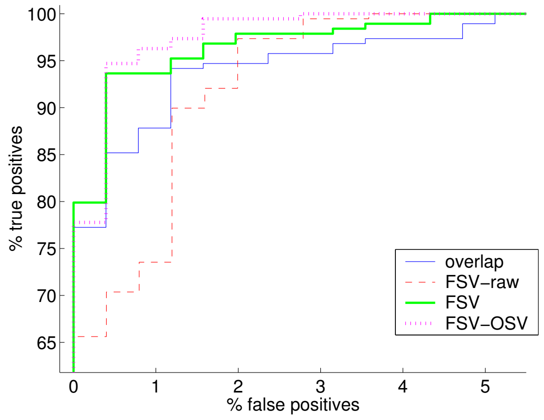
For this experiment, we used the same training and test data as was used in section 3.1. The local quality models were learned using the training data and evaluated using the independent test data set. Figure 7.14a compares the ROC curve for the overlap local quality measure with the curve for a classifier that uses only the RMS distance. The RMS distance is the traditional measure used in evaluating registration quality. This comparison shows that our overlap quality measure, which includes overlap fraction as well as RMS distance, provides a tangible improvement in performance. The ROC plot in figure 7.14b compares the performance of our four local quality measures.

7.4.2 Global quality is maximized at the correct solution

Since the iterative swapping algorithms seek to maximize the global quality measure, it is important to check that this function is actually maximized by the correct solution. To verify

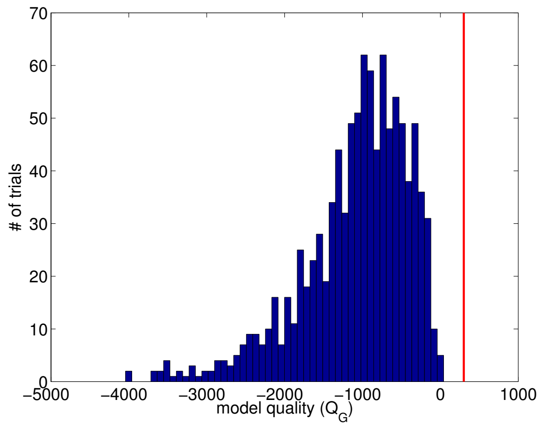


(a)

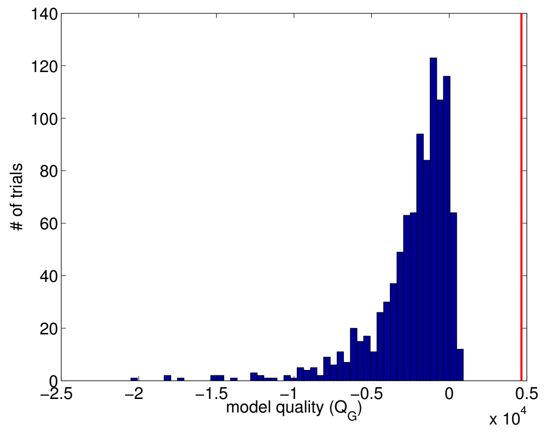


(b)

Figure 7.14: a) ROC curve comparison of the overlap local quality measure versus RMS distance. b) ROC curves comparing all four of our local quality measures.

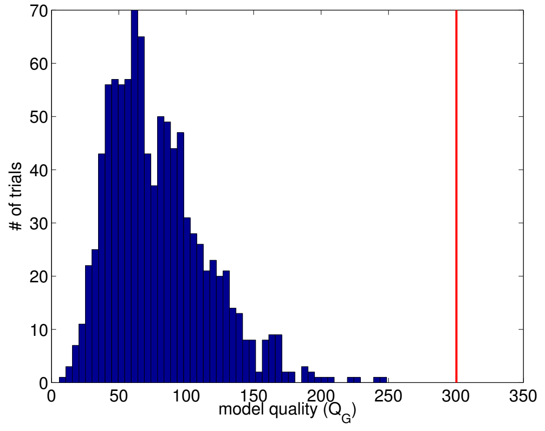


(a) `sum_local` quality measure

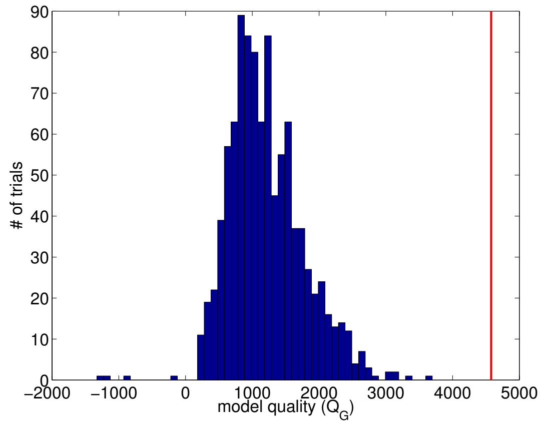


(b) `dist_sqr` quality measure

Figure 7.15: Global quality for random model hypotheses. The distribution of Q_{G} for 1000 random model hypotheses for the angel1 test object. Q_{G} for the ground truth hypothesis is shown by a thick (red) vertical line.



(a) `sum_local` quality measure



(b) `dist_sqr` quality measure

Figure 7.16: Global quality for random correct model hypotheses. The distribution of Q_{G} for 1000 random model hypotheses for the angel1 test object with the hypotheses restricted to contain only correct matches. Q_{G} for the ground truth hypothesis is shown by a thick (red) vertical line.

this, we performed the following experiment on three real test sets (angel4, y1, and eagle2). For each test set, we computed Q_{G} for the ground truth solution. Then, we repeatedly selected 1000 random model hypotheses from G_{LR} and recorded the associated Q_{G} values. The hypotheses were of the form used in the iterative swapping algorithms (i.e., they were spanning trees with edges marked as hidden or visible). We performed this experiment for each model using the `sum_local` measure (equation 3.21) and the `dist_sqr` measure (equation 3.23). None of the trials produced a larger Q_{G} than the ground truth single part solution. The distribution of Q_{G} for one model is shown in figure 7.15. Potentially, a hypothesis containing several correct partial models could result in a lower Q_{G} than the ground truth complete model. To test against this, we repeated the above experiment after removing the incorrect matches from G_{LR}. This ensures that the hypotheses will only contain correct matches, and they will differ only in how many components they contain. The results in figure 7.15 indicate that multi-part models do not have have better quality than the correct model. Notice that the `dist_sqr` measure occasionally generates large negative values for models containing only correct matches. This is caused by accumulation of error, since some random hypotheses contain long paths. The `sum_local` measure is not affected by this because it explicitly models the accumulation of error for long paths. The distributions for the other test models are not included in the figures, since they are qualitatively the same as the ones shown.

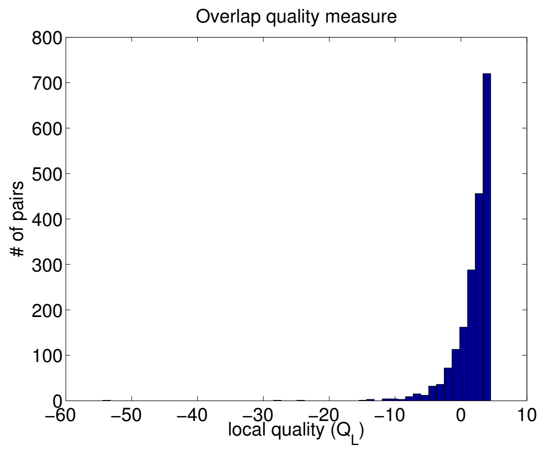
7.4.3 Correct model hypotheses are globally consistent

The purpose of this experiment is to verify that the ground truth solution for a model satisfies the global consistency test. If this is not the case, then those versions of our algorithm that check global consistency will certainly fail. To test this, we computed the local quality (Q_{L}) for all pairs of views for fifteen real test objects. The distribution of these values tells us how we should set the threshold on the global consistency test. Any pair of views that produces a Q_{L} smaller than this threshold will prevent our algorithms from finding the correct solution. Figure 7.17 shows the distribution of the samples for the four local quality measures. We will focus our attention on the overlap and FSV-OSV measures. The distributions decay exponentially with decreasing Q_{L}. Essentially all of the values fall above 20 for these three measures, so we use this value for our local and global consistency tests in all of our multi-view surface matching experiments.

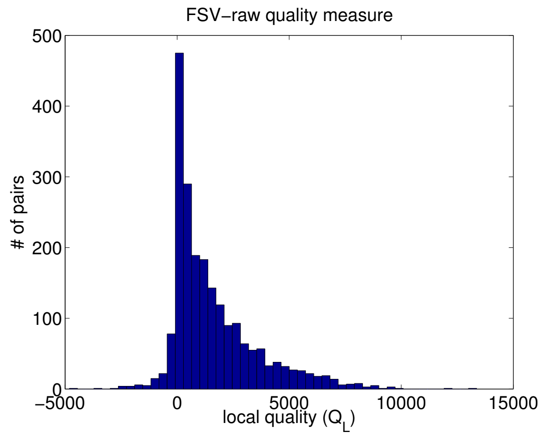
7.5 Multi-view surface matching experiments

In this section, we describe experiments relevant to our multi-view surface matching algorithms:

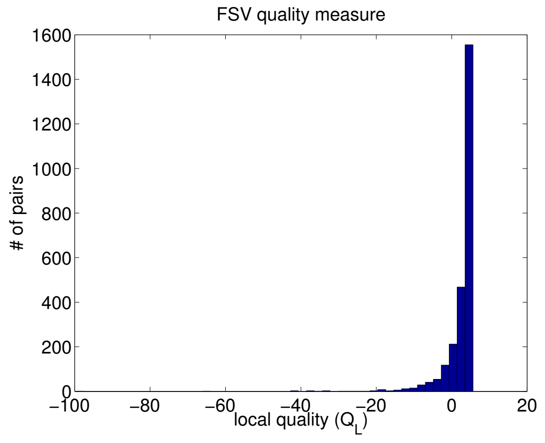
- Variations of the `iterative_merge` algorithm
- Variations of the `random_swap` algorithm
- Comparison of random vs. weighted swapping
- Sensitivity of the `random_swap` algorithm to initial state
- Choice of global quality measure
- Number of input views needed



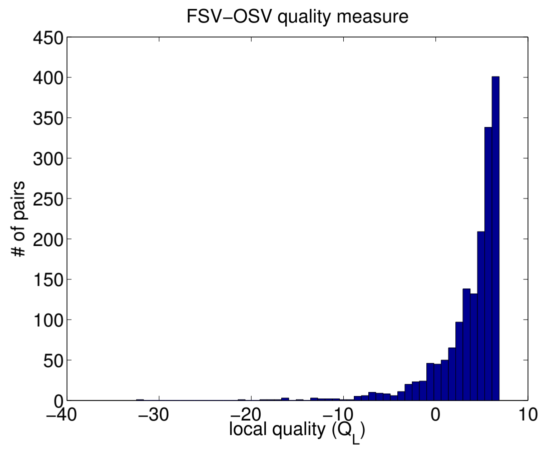
a) overlap



b) FSV-raw



c) FSV



d) FSV-OSV

Figure 7.17: Distribution of the local quality measure (Q_{L}) computed for all pairs of views in 15 correct models.

- Performance on a large database of objects
- Extreme close-up input views
- High-resolution models

7.5.1 Variations on the `iterative_merge` algorithm

This experiment compares the performance of three simplifications of the full `iterative_merge` algorithm (`full`) (see section 4.2.2 for details):

- `fast_mvreg` – only performs multi-view registration if the update fails the global consistency test.
- `no_mvreg` – does not perform the multi-view registration step.
- `mst_only` – does not perform multi-view registration or the global consistency test.

This is an updated version of an experiment originally reported in [34]. Unfortunately, the improved performance of our new registration quality measures does not allow us to differentiate the algorithms using our test objects. To create sufficiently challenging data sets, we disabled the pair-wise registration step in our process, using, instead, the direct output of pair-wise surface matching. We also learned new local quality models for this scenario using the standard procedure (section 3.1). The lower quality input results in a less discriminating registration quality measure, which places an increased burden on the multi-view surface matching algorithms.

With this setup, we ran each variation of the `iterative_merge` algorithm on eight test objects: four real and four synthetic (table 7.3). The results show that for some objects, where the ground truth solution contains multiple parts, all versions of the algorithm fail by merging the parts into a single model that is globally consistent but incorrect. For one object (angel3), the consistency test makes the difference between success and failure (the 16th best match is incorrect and is rejected by the first three variations). For the engine object, multi-view registration at every iteration aligns the views sufficiently to prevent an incorrect match in iteration 31 from passing the consistency test. The execution time

	`full`	`fast_mvreg`	`no_mvreg`	`mst_only`
angel1 (2 parts)	x (1 error)	x (1 error)	x (1 error)	x (1 error)
angel3	+	+	+	x (1 error)
dwarf2	+	+	+	+
squirrel1	+	+	+	+
buddha (2 parts)	x (2 errors)	x (2 errors)	x (2 errors)	x (2 errors)
teeth	+	+	+	+
club	+	+	+	+
engine	+	x (1 error)	x (1 error)	x (1 error)
correct	75.0% (6/8)	62.5% (5/8)	62.5% (5/8)	50.0% (4/8)
partial	0.0% (0/8)	0.0% (0/8)	0.0% (0/8)	0.0% (0/8)
wrong	25.0% (2/8)	37.5% (3/8)	37.5% (3/8)	50.0% (4/8)
median time (sec)	136	22.5	22.1	1.98

Table 7.3: Comparison of several versions of the `iterative_merge` algorithm on a set of real (top section) and synthetic (second section) test objects along with summary statistics (bottom section).

line shows that the improved success rate of the `full` algorithm comes at the expense of execution time.

7.5.2 Variations on `random_swap`

This experiment is analogous to the previous experiment, except that it is applied to the `random_swap` algorithm. We used the same setup to test three simplifications of the full `random_swap` algorithm (`full`) (see section 4.3.3 for details):

- `fast_mvreg` – only performs multi-view registration if the proposed update fails the consistency test or if the global quality goes down.
- `no_mvreg` – does not perform the multi-view registration step.
- `qual_only` – does not perform multi-view registration or the global consistency test.

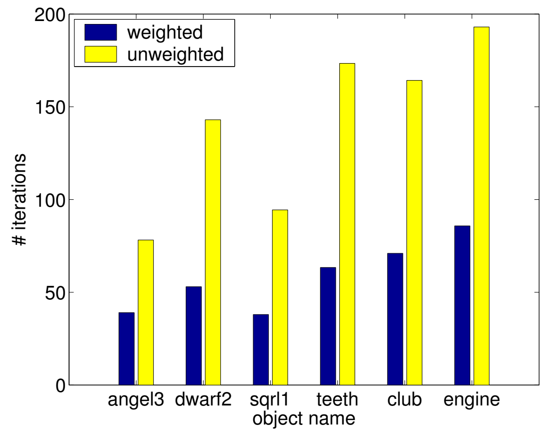
The results (table 7.4) show that the `full` and `fast_mvreg` algorithms overcome the failures that occurred with the `iterative_merge` algorithm. The only mistake happened with the engine object. In this and other experiments, we observed that the algorithms sometimes could not join all of the views of the engine object due to an interaction of our mesh preprocessing and the global consistency test. During preprocessing, we remove boundary

	`full`	`fast_mvreg`	`no_mvreg`	`qual_only`
angel1 (2 parts)	+	+	+	+
angel3	+	+	x (1 error)	x (1 error)
dwarf2	+	+	+	+
squirrel1	+	+	+	+
buddha (2 parts)	+	+	x (1 error)	x (1 error)
teeth	+	+	+	+
club	+	+	x (3 errors)	x (5 errors)
engine	3 parts	+	+	+
correct	87.5% (7/8)	100.0% (8/8)	62.5% (5/8)	62.5% (5/8)
partial	12.5% (1/8)	0.0% (0/8)	0.0% (0/8)	0.0% (0/8)
wrong	0.0% (0/8)	0.0% (0/8)	37.5% (3/8)	37.5% (3/8)
median time (sec)	557	857	658	804

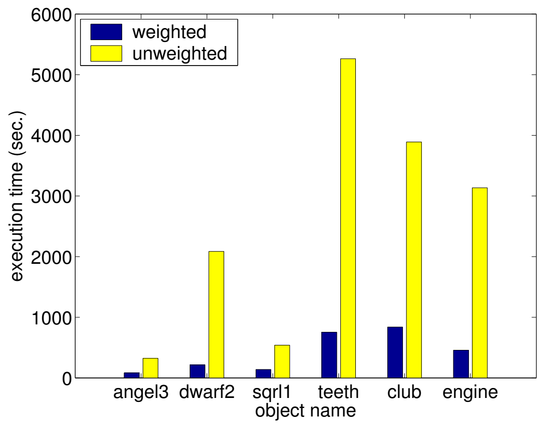
Table 7.4: Comparison of several versions of the `random_swap` algorithm on a set of real (top section) and synthetic (second section) test objects along with summary statistics (bottom section).

faces to eliminate geometric errors that occur at occlusion boundaries (section 8.1.3). For this object, boundary removal causes 60% of the visible surface to be removed in certain views. Consequently, the global consistency test fails due to an unexpectedly large number of OSVs. Possible solutions to this problem include: improving the preprocessing to only remove boundary faces if they are corrupted, including more objects in the local quality measure training set, or improving the occupied space violation test to ignore missing surfaces near boundaries.

The `no_mvreg` and `qual_only` algorithms tend to fail on the same objects in the same manner. This suggests that model hypotheses with high quality are also globally consistent. In other words, ensuring global consistency does not provide an additional benefit over just searching for high-quality hypotheses. This is in contrast to the `iterative_merge` algorithm, where the consistency check sometimes avoids incorrect solutions. Another difference between this experiment and the previous one is that the `fast_mvreg` version actually takes longer than the `full` algorithm. The reason is that the `fast_mvreg` frequently has to perform multi-view registration because the model quality usually goes down an incorrect update is proposed. For these cases, the model quality must be computed again after



(a)



(b)

Figure 7.18: a) The average number of iterations required for `weighted_swap` and `unweighted_swap` to attain the correct solution. b) The corresponding processing time.

multi-view registration, so these iterations take longer than they would have taken for the `full` algorithm.

7.5.3 Effect of weighted swapping

This experiment tests whether the use of weights for selecting update operations improves performance of the `random_swap` algorithm. We ran `random_swap` using weighted updates (`weighted_swap`) and unweighted updates (`unweighted_swap`) five times for each of six test objects. The results in figure 7.18 show that weighted updates are beneficial, reducing the number of iterations required to find the correct solution by a factor of two or more and reducing execution time significantly as well.

7.5.4 Sensitivity of `random_swap` to initial state

The next question we address is whether the `random_swap` algorithm is sensitive to its initial state in the hypothesis space. Ideally, the algorithm would find the same solution regardless of the starting state.

For this experiment, we ran the `weighted_swap` algorithm 30 times for each of six test

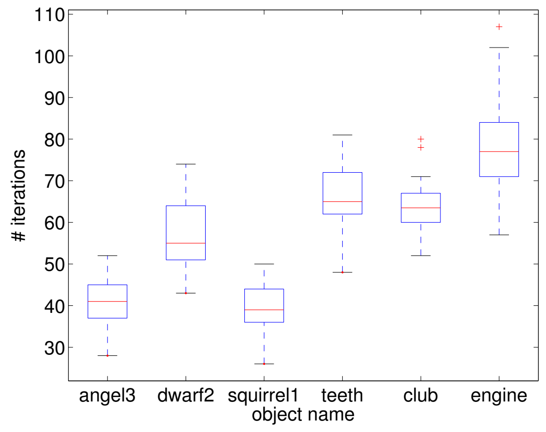


Figure 7.19: Sensitivity of `random_swap` to its initial state for 30 trials using each object in terms of the number of iterations required to attain the correct solution. The box plot shows the 25th and 75th percentiles (main box), the median (red line inside main box), the range of all inliers (whiskers on top and bottom), and any outliers (points beyond the whiskers).

objects (three real and three synthetic), starting from a different randomly chosen initial state each time. The results (figure 7.19) indicate that, in practice, the algorithm does not become trapped in local minima and that the number of iterations needed depends more on the shape and complexity of the object than on the initial state of the algorithm.

7.5.5 Choice of global quality measure

In this experiment, we test whether the choice of global quality measure (`sum_local` or `dist_sqr`) has an effect on multi-view surface matching performance. We ran the `weighted_swap` algorithm on eight test objects (four real and four synthetic), once for each quality measure. The results in table 7.5 show that the `sum_local` measure performs slightly better in terms of success rate as well as number of iterations. However, this difference is probably not significant.

7.5.6 Number of input views needed

One question that would interest end users is “how many input views are necessary for successful modeling?” To test this, we ran the `weighted_swap` algorithm for a set of six

	`sum_local`	`dist_sqr`
angel1	+	+
angel3	+	+
dwarf2	+	+
squirrel1	+	+
buddha (2 parts)	+	+
teeth	+	+
club	+	+
engine	+	x (1 error)
correct	100.0% (8/8)	87.5% (7/8)
partial	0.0% (0/8)	0.0% (0/8)
wrong	0.0% (0/8)	12.5% (1/8)
median iters (sec)	59	62

Table 7.5: Comparison of the `sum_local` and `dist_sqr` global quality measures for real objects (top section) and synthetic objects (middle section) along with summary statistics (bottom section).

	number of input views						
	32	27	22	17	12	7	2
joint	+	+	+	+	+	+	+
head	+	+	+	+	+	+	+
engine	+	+	+	5 parts	+	3 parts	+
teeth	+	+	+	+	+	+	+
club	+	+	+	+	+	+	+
knee	+	+	+	+	+	+	+
correct	100%	100%	100%	83.3%	100%	83.3%	100%
partial	0%	0%	0%	16.7%	0%	16.7%	0%
wrong	0%	0%	0%	0%	0%	0%	0%

Table 7.6: Performance as a function of the number of input views for six synthetic test objects.

synthetic objects (which all have the same number and distribution of viewpoints). We executed a sequence of trials for each object, randomly removing five input views in the first trial, ten in the second trial, and so forth. The results in table 7.6 show that the success rate is unaffected by the number of input views. For some of the reduced input trials, the ground truth solution consisted of multiple parts, but in all except two cases, the algorithm found the best possible solution. For the engine-17 trial, the ground truth solution contained three parts, and for the engine-7 trial, the ground truth solution contained two parts. The fact that our algorithm found more parts for these trials can be attributed to the problem with the engine data set explained in section 7.5.2. We can conclude that the limiting factor for the number of input views lies in the ability of the pair-wise surface matching algorithm rather than in the multi-view surface matching algorithm.

7.5.7 Performance on a large set of objects

Our final experiment tests the best version of each type of algorithm on the entire set of real and synthetic objects. We compare `iterative_merge` (section 4.2.1), `iterative_grow` (section 4.2.3), `weighted_swap` (section 4.3.2), and `greedy_swap` (section 4.3.1). The results are summarized in table 7.7, and examples of the reconstructed objects are shown in figures 7.20, 7.21 and 7.22.

Two of the objects (enterprise and skateboard) failed for all of the algorithms due to symmetry. The ground truth solution for these objects consisted of multiple parts, but the symmetry of the objects (e.g., the top and bottom of the enterprise's saucer section) led to an incorrect hypothesis with higher quality than the correct multi-part hypothesis. The incorrect solutions were globally consistent due to the unique shape of the objects and the limitations of the OSV quality measure. One possible solution is to develop a better OSV measure that explicitly includes the depth-of-field limits of the sensed data. Currently this information is inferred from the data, and is overly conservative. A second solution is to allow the multi-view surface matching algorithm to output multiple hypotheses and let the user choose the correct one.

	`iterative_merge`	`iterative_grow`	`weighted_swap`	`greedy_swap`
angel1	+	+	+	+
angel2	+	+	+	+
angel3	+	+	+	+
angel4	+	+	+	+
angel5	+	+	+	+
bunny1	+	+	+	+
bunny2 (3 parts)	+	+	+	+
duck1b	+	+	x (1 error)	+
dwarf1 (2 parts)	x (1 error)	x (1 error)	+	+
dwarf2	+	+	+	+
eagle2	+	+	+	+
j1	+	+	+	+
lion1	+	+	5 parts	3 parts
nicolas2	+	+	+	+
rabbit1	+	+	+	+
squirrel1	+	+	+	+
y1	+	+	+	+
dino2 (3 parts)	+	+	4 parts	4 parts
buddha (2 parts)	+	+	+	+
bunny	+	+	+	+
club	+	+	+	+
dinosaur	+	+	+	+
dragon	+	+	+	+
engine	+	+	+	+
enterprise (3 parts)	x (1 error)	x (1 error)	x (1 error)	x (1 error)
generic	+	+	+	+
head	+	+	+	+
horse	+	+	+	+
isis	+	+	+	+
joint	+	+	+	+
knee	+	+	+	+
sandal (3 parts)	+	+	+	+
skateboard (3 parts)	x (4 errors)	x (4 errors)	x (4 errors)	x (2 errors)
teeth	+	+	+	+
widget	+	+	+	+
correct	91.4% (32/35)	91.4% (32/35)	85.7% (30/35)	88.6% (31/35)
partial	0.0% (0/35)	0.0% (0/35)	5.7% (2/35)	5.7% (2/35)
wrong	8.6% (3/35)	8.6% (3/35)	8.6% (3/35)	5.7% (2/35)

Table 7.7: Performance on the entire set of real and synthetic test objects.

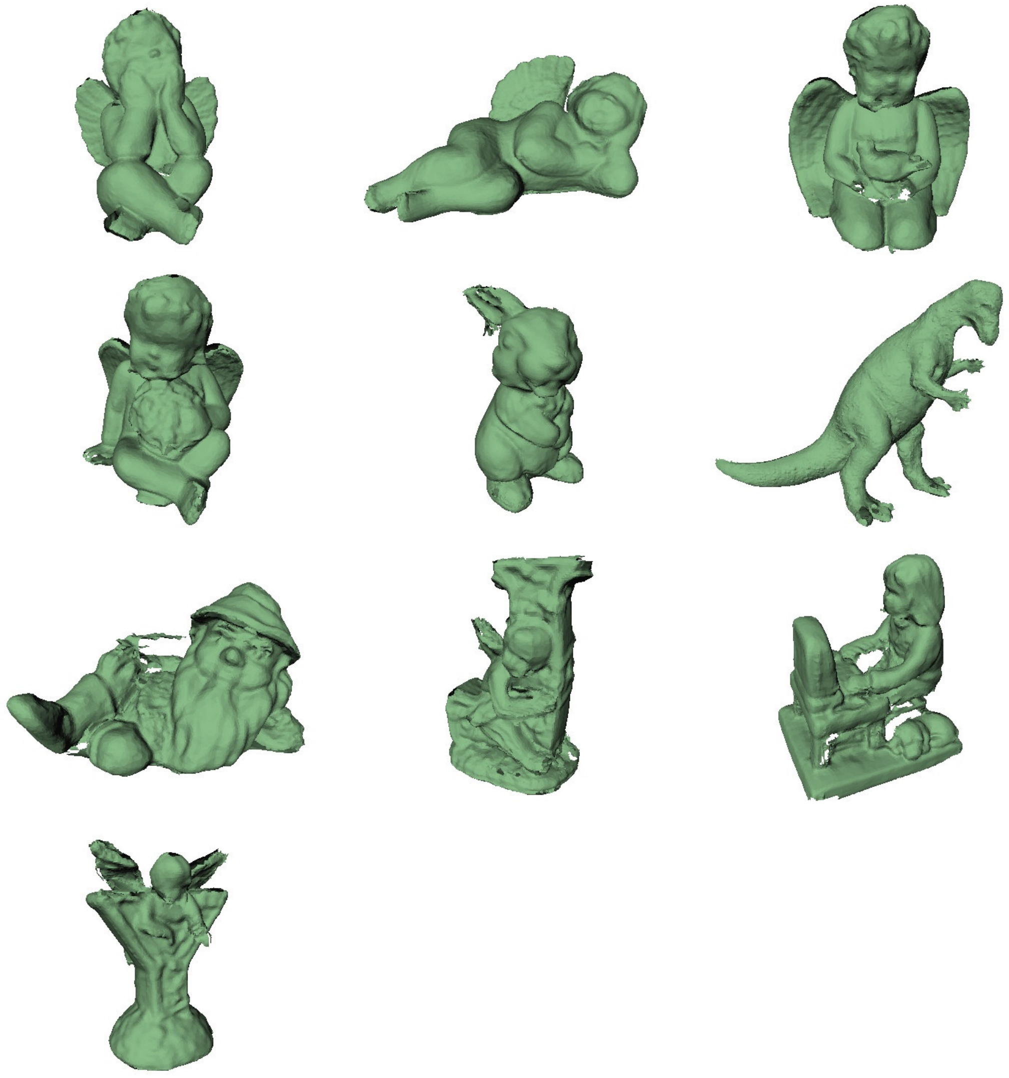


Figure 7.20: Examples of automatically modeled real objects created using the hand-held modeling method and the Vivid scanner. Compare to the original objects shown in figure 7.1.

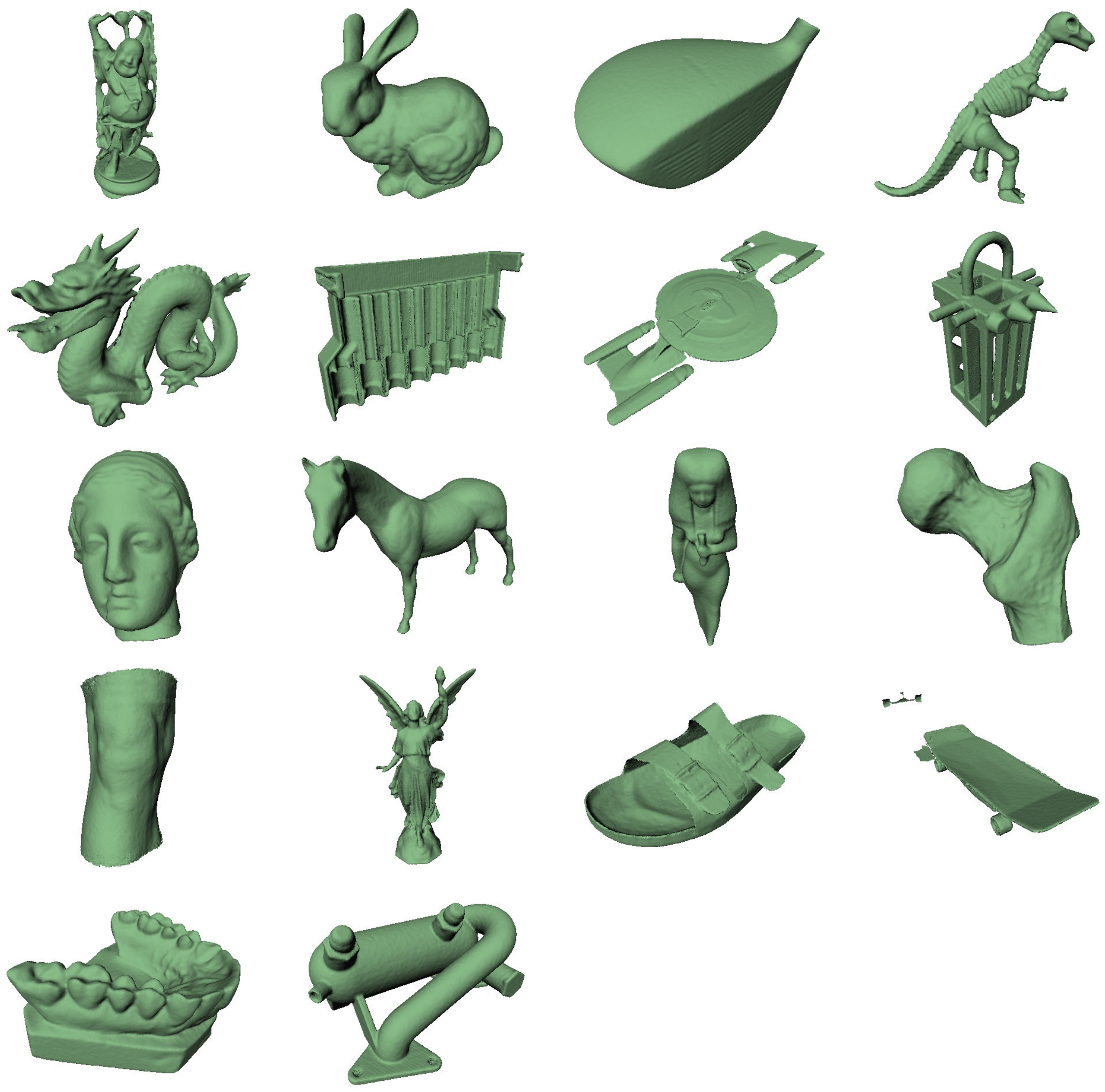


Figure 7.21: Reconstructed versions of the synthetic objects. Compare to the original objects shown in figure 7.2. All of the reconstructions are correct except for enterprise and skateboard.

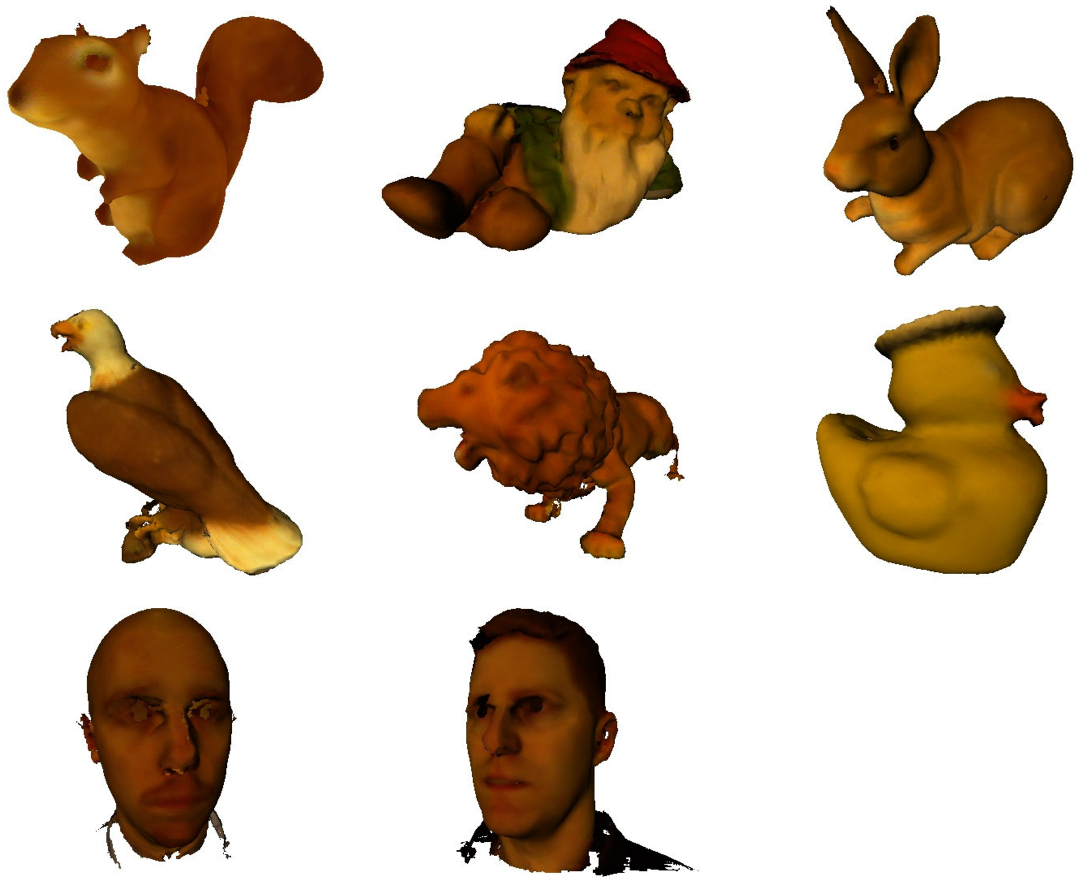


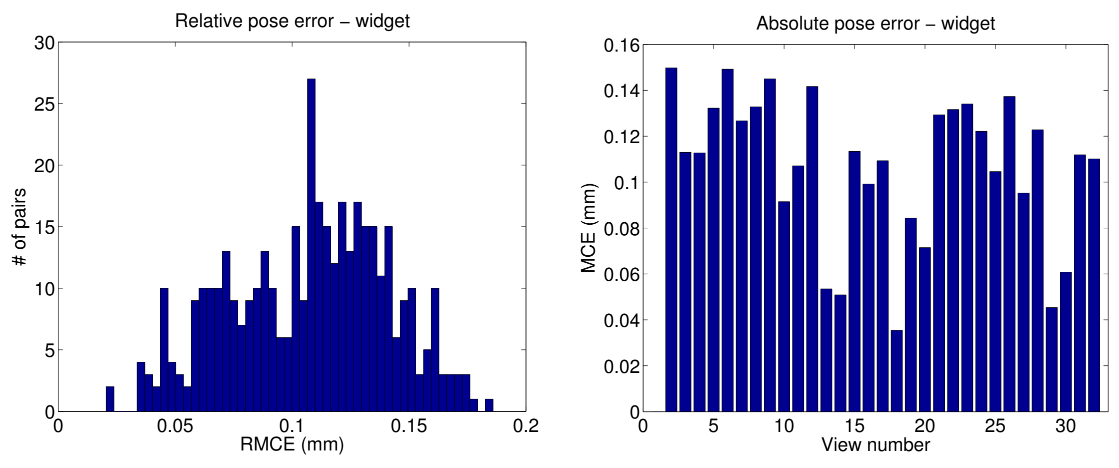
Figure 7.22: Examples of automatically modeled scenes with texture mapping applied as described in section 8.2.2.

One object (dwarf1) failed for the grow/merge algorithms but succeeded for the swapping algorithms. This is a similar situation to the enterprise data set, except that the swapping algorithms were able to find a globally superior solution using slightly lower quality pair-wise matches. For the duck1b object, only `weighted_swap` failed. In this case, the algorithm became trapped in a local minimum due to the symmetry of the object and geometric errors in the input data.

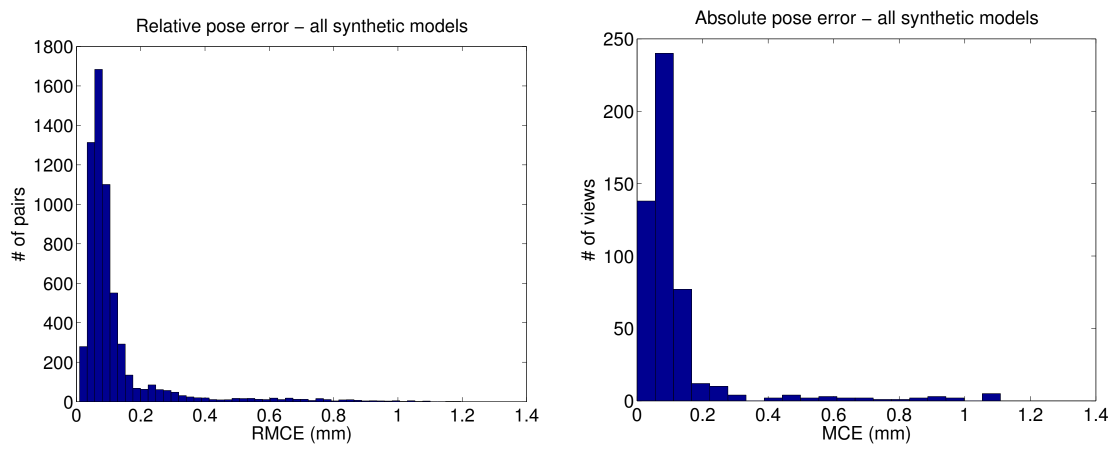
Finally, for two objects (lion1 and dino2) the swapping algorithms could not join the last few parts. For these objects, small errors in registration prevented the correct solution from being globally consistent. This is a disadvantage of the FSV measure – for some surface geometries, slight errors in registration can cause large free space violations. One solution would be to explicitly detect when this situation occurs.

Overall, the iterative addition algorithms actually perform slightly better than the swapping algorithms, a surprising result given the experiments of sections 7.5.1 and 7.5.2. However, the differences can be explained by the improved local quality measure used in this experiment. The best result can be obtained by first running either `iterative_merge` or `iterative_grow` and then using this result as the initial state for `weighted_swap`. Using this approach would result in success for all except two of the test objects.

For the synthetic objects, we also quantitatively evaluated the pose error and reconstruction error of the final model hypotheses. We omit the skateboard and enterprise objects from these calculations, since they were qualitative failures. Figure 7.23a shows the distribution of relative pose errors for all overlapping views in the final hypothesis for the widget data set, and figure 7.23b shows the corresponding absolute pose error for all the views. All of the errors fall below 0.1% of the model size. The distributions of relative and absolute pose errors for all synthetic objects (figure7.23c and d) indicate that the performance on the widget data is typical. Most of the outlier points in these two plots are due to the engine data set, which proved to be more difficult than the others as discussed previously.



(a) (b)



(c) (d)

Figure 7.23: Quantitative evaluation of synthetic models. a) Relative pose error for all overlapping view pairs in the output model hypothesis for the widget object. b) Absolute pose error for the views in the widget model. c) Relative pose error for all qualitatively correct synthetic models. d) The distribution of absolute pose errors for all synthetic models.

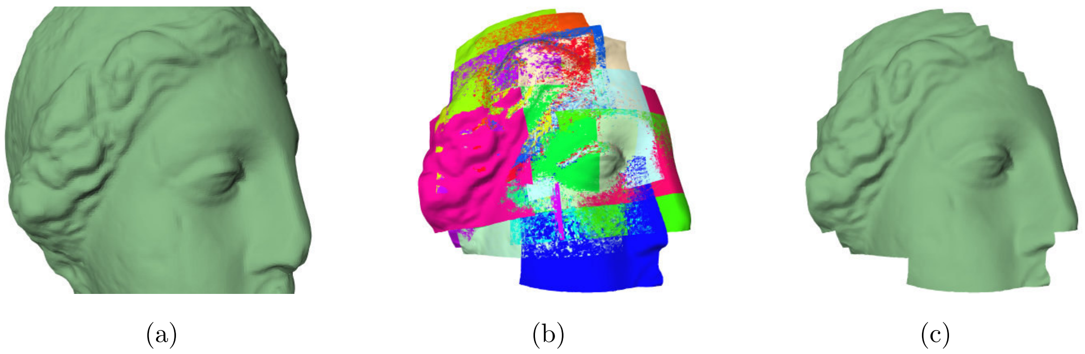


Figure 7.24: Modeling using extreme close-up views. a) The input scene – a high-resolution version of the head object. b) The automatically registered views with each view displayed in a different color. c) The reconstructed model seen from the same viewpoint as in (a).

7.5.8 Extreme close-up input views

This experiment shows that our automatic modeling algorithms can be applied in situations where the scene is scanned at an extremely close range. This type of data collection was used in the Digital Michelangelo project [47]. In this data collection scenario, each scan only covers a small part of the overall object being scanned, and several hundred scans may be necessary to cover the entire object. At such close range, hand registration becomes more difficult due to lack of distinguishing features, and accurate records of the location of each scan are critical. Automatic modeling relieves the scanning team from having to record the precise scan locations. Instead, the views covering each scene part (e.g., the statue's arm) could be input into our system and automatically registered.

As a demonstration of this technique, we created a set of views of the head test object[2] (figure 7.24a). The views simulate the capabilities of a high-resolution scanner scanning the scene from extreme close range. We generated 19 views covering the right eye and cheek of the scene and an independent set of 10 views of a different region, which were used for learning the local quality model for this scenario. Figure 7.24b and c show the successful result of automatic modeling for this test set. Our algorithm found the correct single-part solution for these views..

[2]We used a high-resolution version of the head model (269K faces) for this test.

7.5.9 High-resolution models

In the examples we have shown so far, both real and synthetic, some of the fine detail of the original scenes is lost in the reconstructed model. The detail loss in the real models is due to the noise characteristics of the Vivid scanner, which are relatively poor compared to other commercially available scanners. The Cyberware scanner used in the Digital Michelangelo project [47] has a one-sigma noise level of 50μm as compared to 1mm for the Vivid. The synthetic test views are injected with noise to simulate the worst case performance of the Vivid sensor, so it is unsurprising that the resulting models are also fuzzy. Our success in modeling using this data shows that our algorithms do not depend on extremely high quality data to succeed.

This experiment illustrates the level of detail that can be achieved if the input views are higher resolution and have no noise. For this experiment, we re-generated the input views of three synthetic objects, using a 400×400-pixel range image (rather than 200×200-pixels as per the Vivid) and disabling the synthetic noise injection. The reconstructed models were then generated using Curless' surface reconstruction algorithm and the original model hypothesis produced by our algorithm during the experiment in section 7.5.7. Figure 7.25 shows a side-by-side comparison of the original models and the reconstructed ones. The examples exhibit a barely discernible loss of detail, which can be attributed to the pixel quantization introduced by range image formation and the voxel quantization introduced by surface reconstruction.

7.6 Modeling using alternate sensors and scene types

The experiments in the previous sections focus on modeling small objects using the Vivid scanner or synthetic data that mimics that sensor's performance. In this section, we show demonstrate that the core concepts of this thesis are applicable across many sensors, scene types, and scene scales. In particular, we show results for two additional sensors (the BF2 and the Z+F), for two scene types (building interiors and terrain), and for scenes up to 200

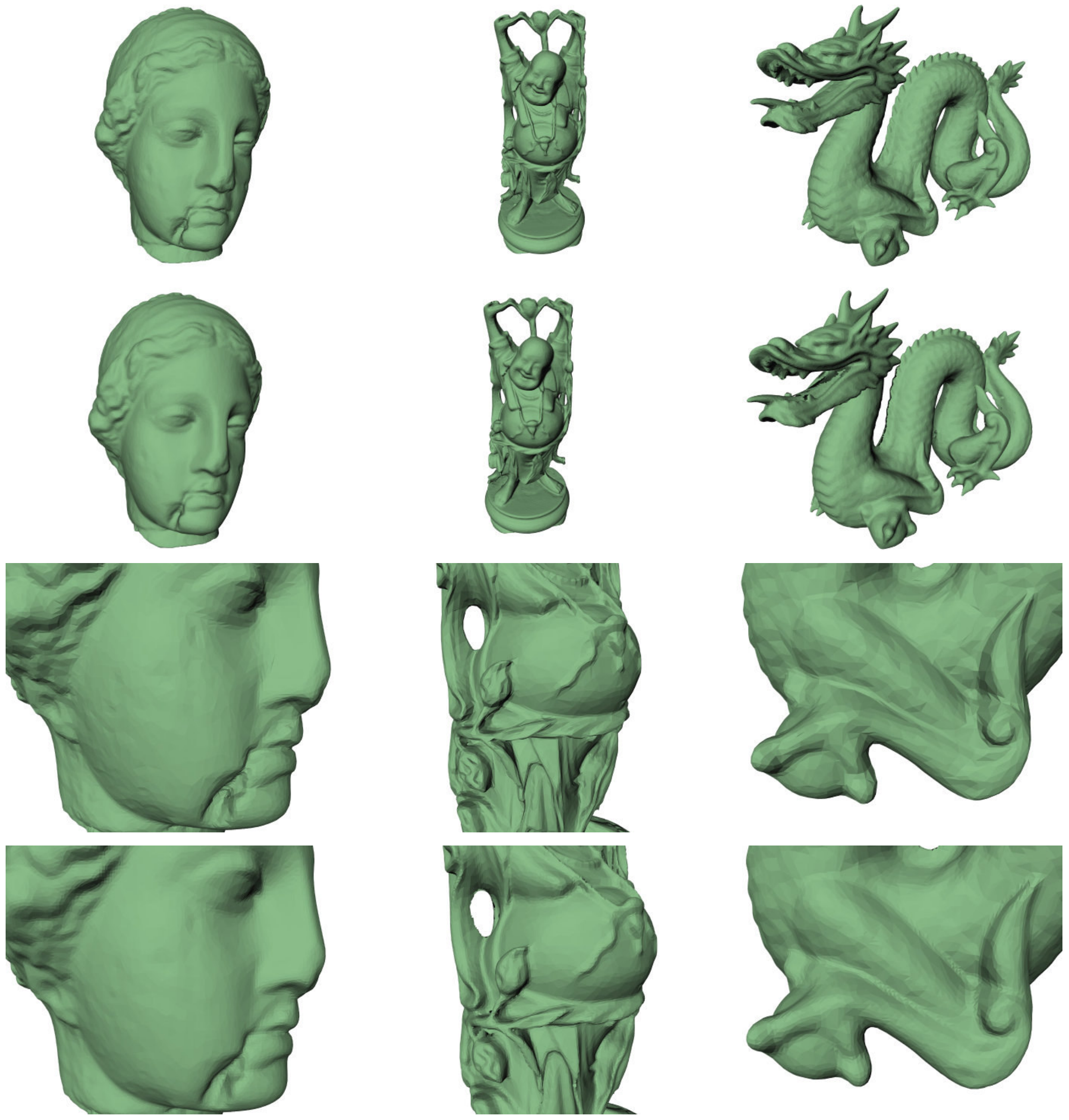


Figure 7.25: Side-by-side comparison of original scenes (first and third rows) and the reconstructed models (second and fourth rows). Even in the close-up views (third and fourth rows), it is difficult to detect any differences.

meters in size.

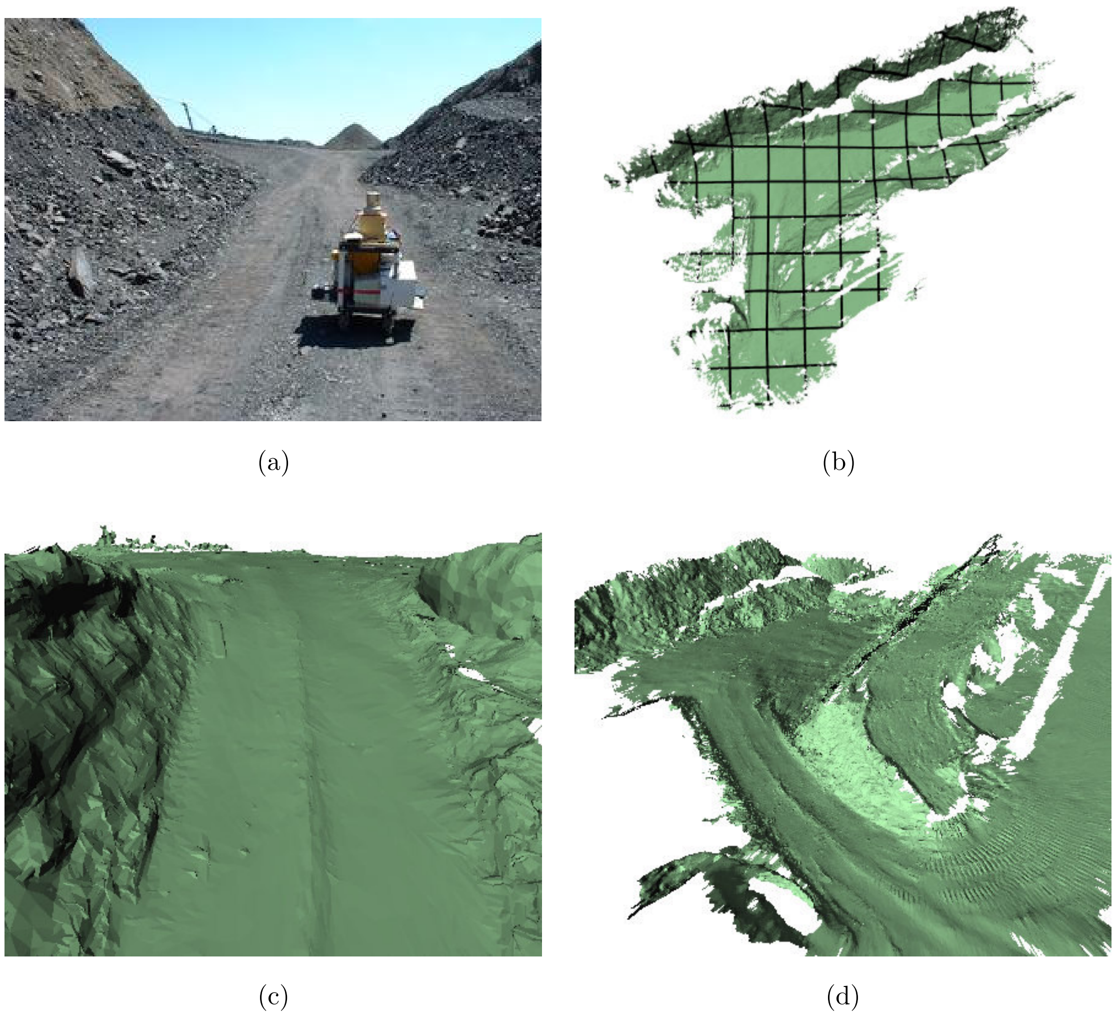
We did not have to modify many system parameters to work with these different scenarios. To accommodate the larger scene scale, we increased the maximum overlap distance threshold ($D_{\max}$) to one meter, increased the spin-image support size to ten meters. To accommodate the increased complexity of the scenes, we increased the resolution of the meshes used in pair-wise surface matching to 2000 faces. Finally, we learned two new local quality models – one for terrain modeling and one for interior modeling – using the same procedure described in section 3.1. Since we had much less data for these two scenarios, we learned each local quality model from a single data set (coal1 for the terrain models and carbarn2 for the interior models). Surprisingly, we found that using the terrain registration quality model for the interior modeling data sets worked fairly well; however, using a separate quality model tailored to the interior modeling scenario improved the performance.

7.6.1 Terrain modeling with the Z+F

The Z+F scanner is appropriate for modeling medium-scale outdoor environments. Performance characteristics are given in appendix A.2. By mounting the scanner on a cart, we can quickly obtain scans from various locations around the site to be modeled. Using this technique, we have collected terrain data sets at several different locations. Here, we highlight the results from an open pit coal mine, which we visited on three separate occasions. The three data sets are: coal1 (30 views, size = 202 meters), coal2 (20 views, size = 137 meters), and coal3 (34 views, size = 225 meters). Figures 7.26 and 7.27 show the resulting terrain models and some representative views of the models.

7.6.2 Terrain modeling with the BF2

The BF2 is the predecessor to the Z+F LARA scanner, and is therefore similar in design. Details are given in appendix A.3. Using the BF2, we collected data with ground truth at a slag heap near Carnegie Mellon (figure 7.28a). For this experiment, the sensor was mounted on a cart at a 15° angle, and scans were obtained at predetermined locations throughout



(a) (b) (c) (d)

Figure 7.26: The coal1 (30 views, size = 202m) and coal2 terrain models (20 views, size = 137m). a) Photograph of the modeled terrain. b) Top view of the reconstructed model with 10-meter grid lines. c) A view of the coal1 model showing the same region seen in (a). d) A perspective view of the coal2 model.

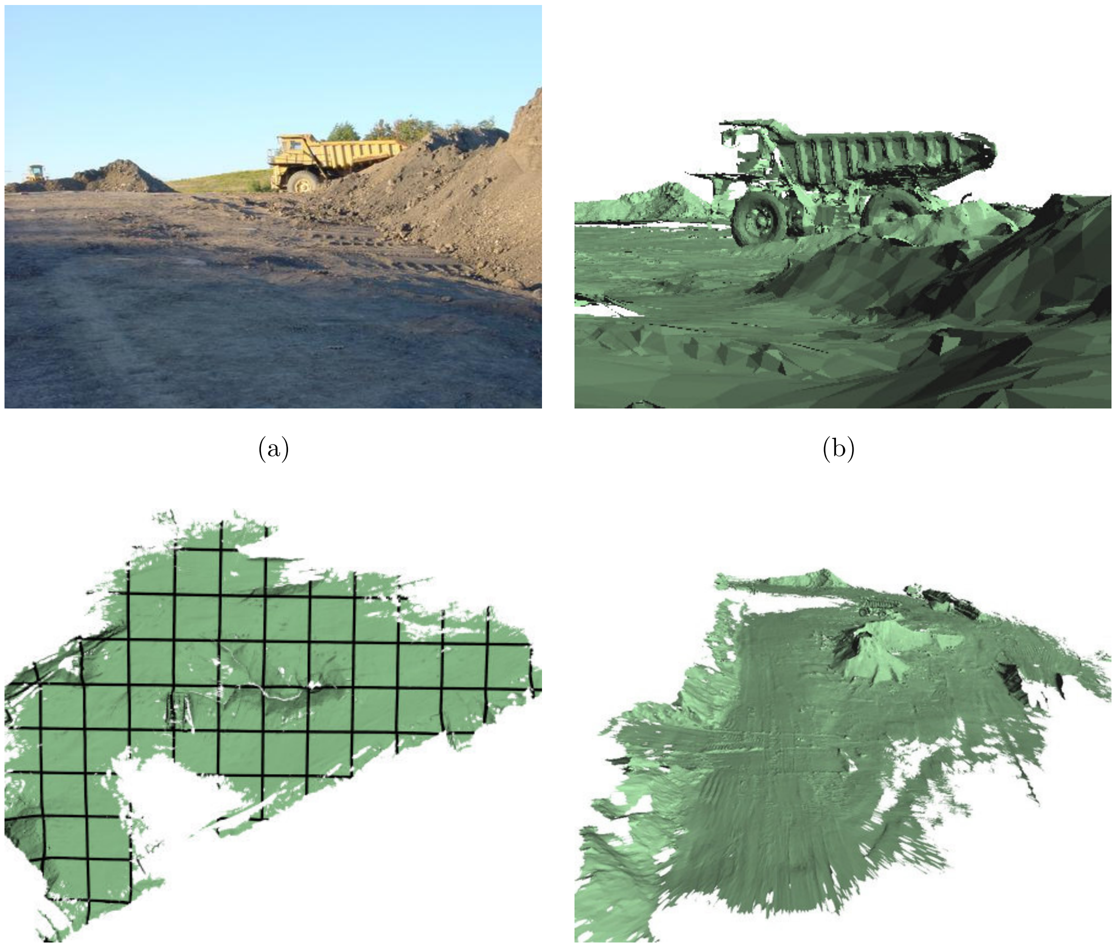
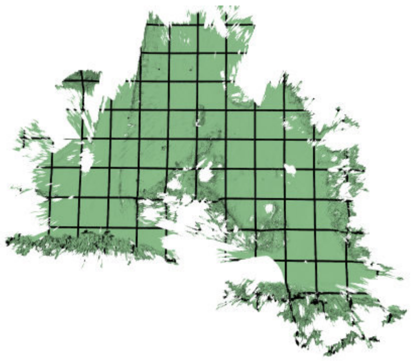
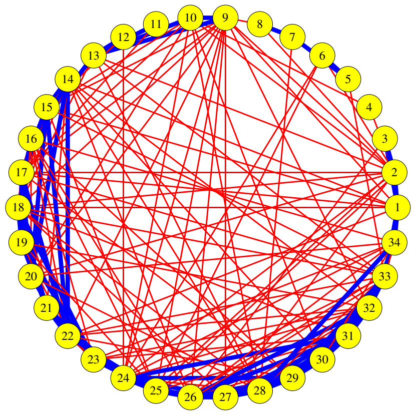


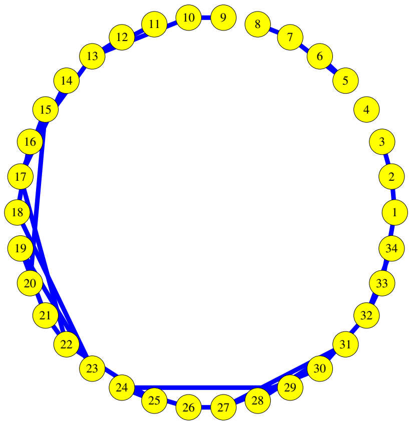
Figure 7.27: The coal3 terrain model (34 views, size = 225m). a) Photograph of the modeled terrain. b) A view of the output model showing the same region seen in a. c) Top view of the reconstructed model with ten-meter grid lines. d) A perspective view.

(a)

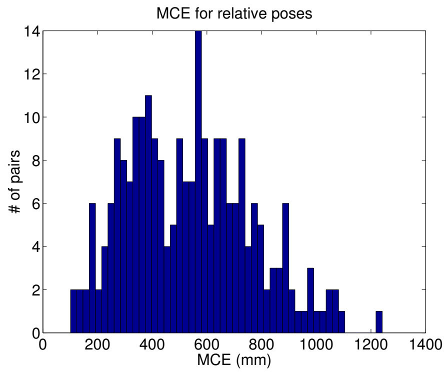

(b)



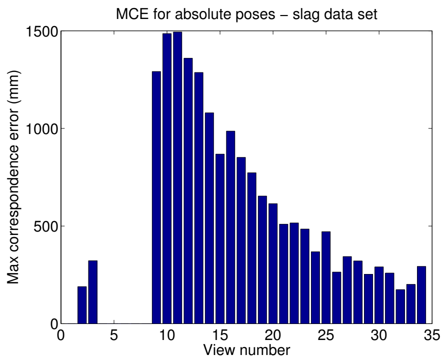
(c)



(d)



(e)



(f)

Figure 7.28: The slag heap terrain model (34 views, size = 186m). a) A photograph of the terrain. b) Top view of the reconstructed model with 10-meter grid lines. c) G_{LR} for this data set. d) The output model graph correctly finds the three components from G_{LR} containing only correct matches. e) The relative pose error for the views in the largest component. f) The absolute pose error for the same views.

the scene.

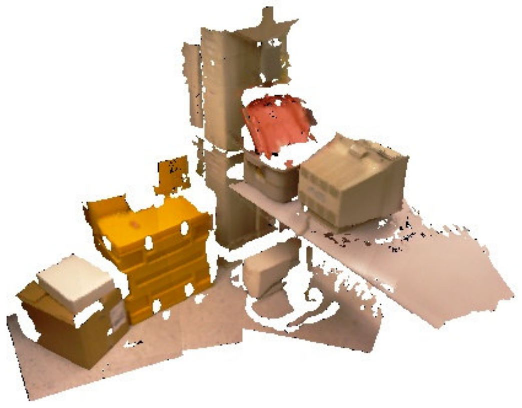
Ground truth registration was obtained using fiducials placed in the environment and surveyed using differential GPS. Additionally, we recorded the position of the sensor at each viewpoint using the same technique. The position of each fiducial in each range image was manually identified and converted to 3D sensor coordinates. The ground truth position of a view was then obtained by minimizing the sum of squared distances between the image-based measurements and their corresponding GPS measurements (using equation 5.3). The process was repeated for each view.

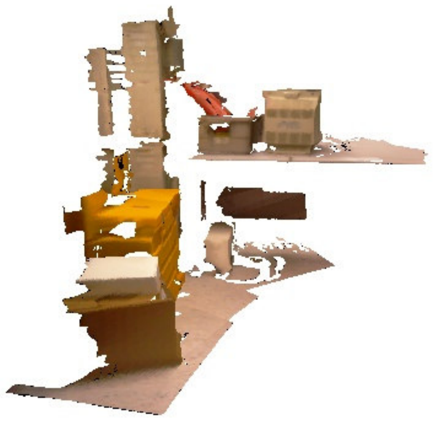
We collected 34 scans at intervals of approximately five meters in a large loop around a partially excavated hill. Unfortunately, due to the dense vegetation and limited field of view of the sensor, the pair-wise surface matching algorithm was unable to find correct matches between several sets of views, resulting in a G_{LR} consisting of three parts (figure 7.28c). Our algorithm successfully found the correct three-part model (figure 7.28d). Figure 7.28e shows the relative pose error for view pairs in the largest component. The median relative error is 0.52 meters, which is approximately 0.05% of the view size. Similarly, the absolute pose error plot (figure 7.28f) illustrates that the maximum absolute pose error for any view is 1.5 meters (0.8% of the model size). The steadily accumulating error from view 34 down to view 10 arises because the configuration of G_{LR} did not allow the algorithm to close the large loop in the data set.

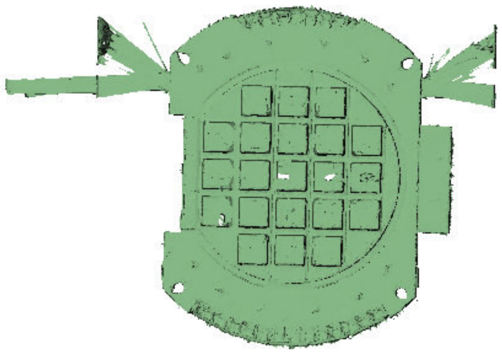
7.6.3 Interior modeling

We have successfully modeled the interior of several rooms and buildings using all three of the sensors previously described. Specifically, we used the Vivid scanner to model a corner of our lab (figure 7.29a and b), the BF2 and the Z+F to model the basement of our building (commonly known as the car barn) on three separate occasions (figure 7.30), and the Z+F to model an auditorium at CMU (figure 7.29c and d). All of these data sets were obtained using the same "sensor on a cart" method that was used for terrain modeling.

For one of the data sets (car_barn3), we also measured ground truth using fiducials


(a)


(b)


(c)

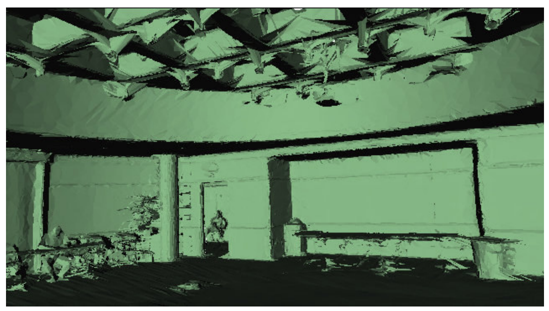

(d)

Figure 7.29: Example interior models. a) and b) Two views of a model of part of our lab scanned with the Vivid scanner (15 views, size = 3.3m). c) Top view of the Singleton auditorium model (6 views, size = 29m). d) A close-up of the inside of the room.

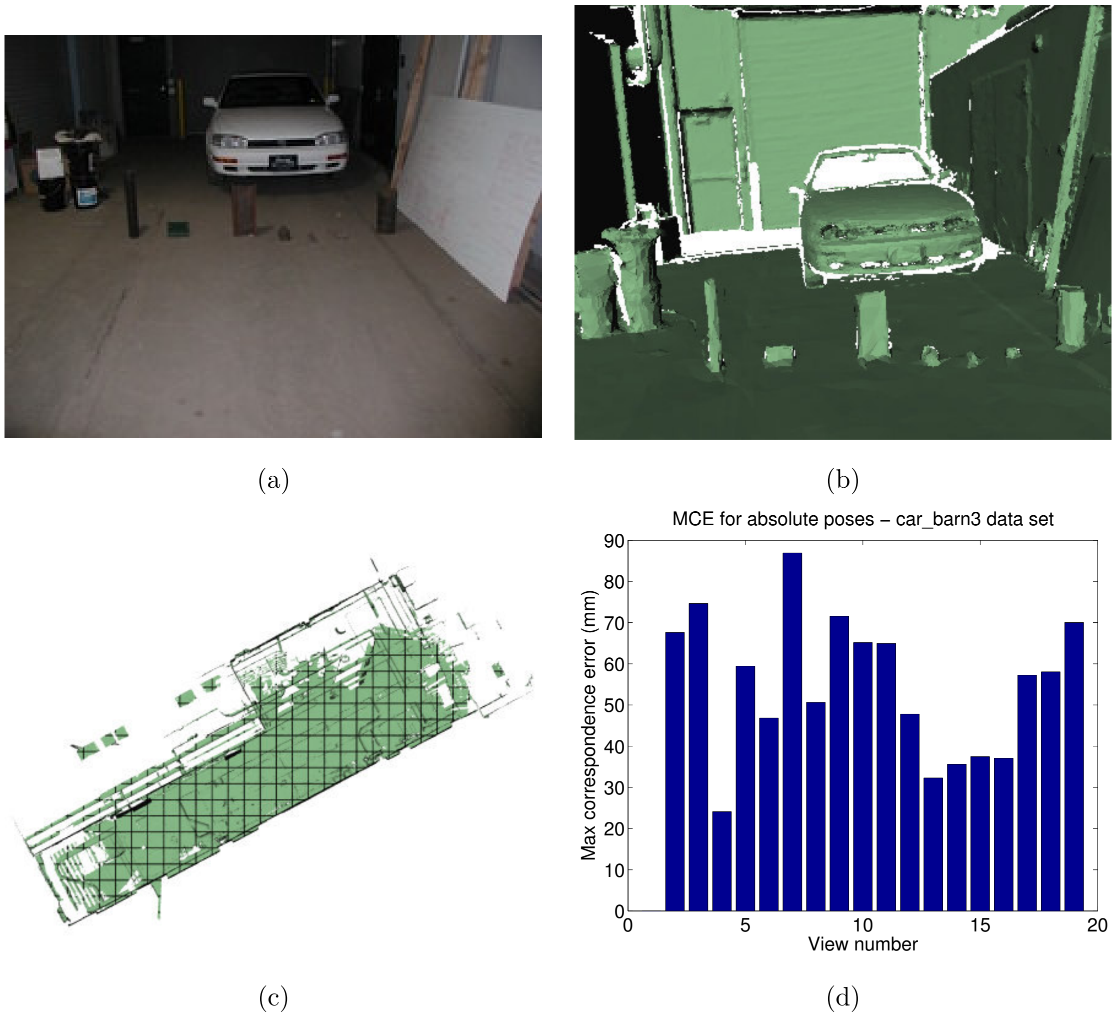


Figure 7.30: The car_barn3 interior model (19 views, size = 40m). a) A photograph of the scene. b) A view of the output model showing the same region seen in (a). c) Top view of the reconstructed model with one-meter grid lines. d) The absolute pose error for the views.

placed in the environment. Figure 7.30d shows the maximum correspondence error for the absolute poses, none of which exceeds 0.25% of the model size.

7.6.4 Example of a symmetric scene

Symmetric scenes can cause problems with our multi-view surface matching algorithm. In the introduction, we described an example of a sphere, where it is not possible to determine the original object based on shape alone. Fortunately, such a degree of symmetry does not frequently happen in real scenes. However, it is instructive to see what our algorithm will do when faced with a highly symmetric scene.

The warehouse1 data set consists of 28 views of a 3-room warehouse (figure 7.31) obtained with the Z+F scanner. The building is rectangular with a 4-way reflection symmetry. The rooms are arranged in series, with two identically sized small rooms at the ends and a longer room in between. The long room was too large for the scanner to see both ends simultaneously. The warehouse was empty except for a backhoe in one of the small rooms and a "cherry picker" and a small office in the other small room.

Although pair-wise surface matching found correct matches for view pairs within each room, it did not match any views between adjacent rooms (because there were no views bridging two rooms). Therefore, the correct solution for this data set would be three partial models – one for each room. Our algorithm found an incorrect but globally consistent solution that essentially folds the building in half along its center. The primary reason for this error is the large size of the center room. The open end of the building (figure 7.31b) in our algorithm's solution is consistent with the data that was observed. A sensor with a longer maximum range or a less symmetric scene would not have this problem. Detection and exploitation of scene symmetry is a subject of ongoing research (section 9.2.1).

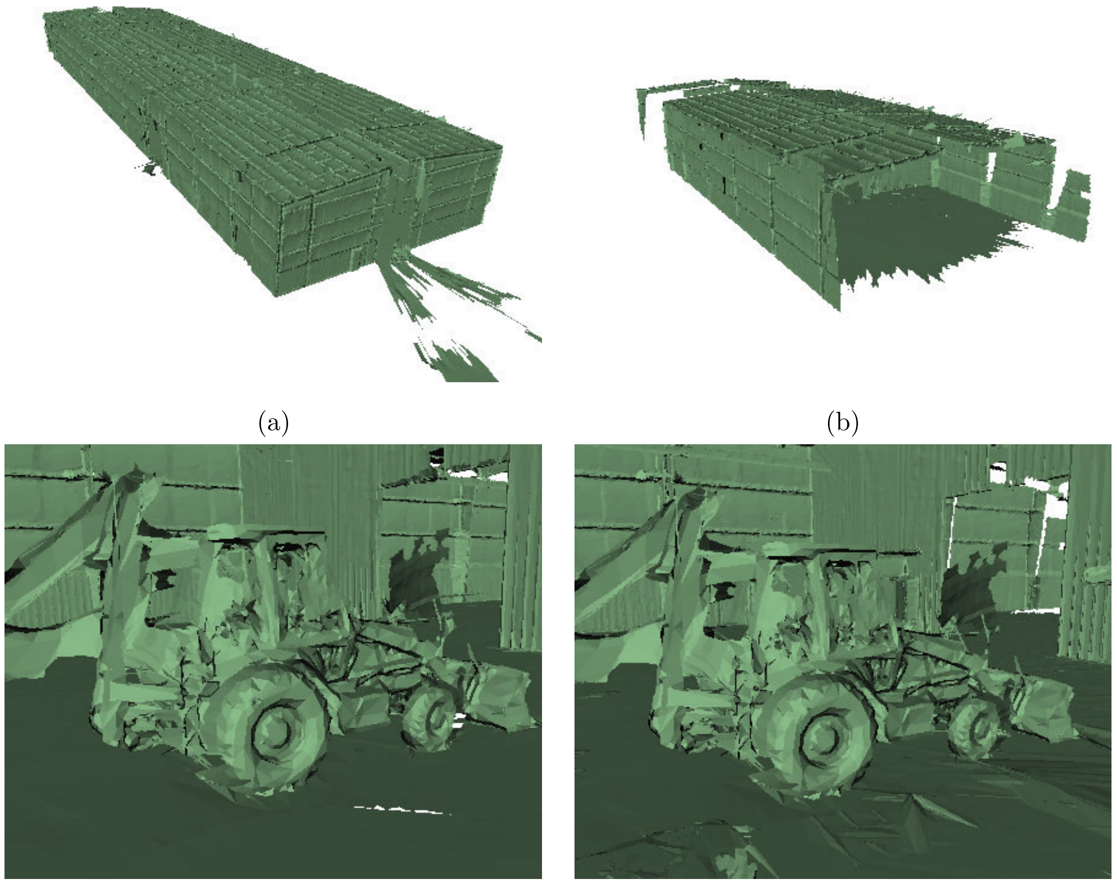

(a) (b) (c) (d)

Figure 7.31: The warehouse model, an example of failure due to symmetry (28 views, size = 125m). a) Perspective view of the manually determined correct model. b) The model produced by our algorithm folds the building in half. c) An example view from inside the correct model. d) The corresponding view in the incorrect model is nearly identical.

Chapter 8

A system for Fully Automated Modeling of Unstructured Scenes (FAMUS)

The bulk of our system has been covered in previous chapters. In this chapter, we give the details of the additional operations that, together with our multi-view surface matching algorithms, constitute our modeling from reality system. These operations can be divided into preprocessing and postprocessing steps. In the former category, we preprocess the input data to convert it into the surfaces that our algorithms operate upon. In the latter category, we use a surface reconstruction system developed by Curless [15] to integrate the registered surfaces. Additionally, we implemented a texture mapping algorithm to generate a texture map derived from co-registered color images of the scene.

8.1 Preprocessing operations

The basic goal of the preprocessing step is to convert the input range images into surface meshes. Additionally, we perform a number of other operations to condition the data. These operations include range shadow removal, noise filtering, mesh simplification, and smoothing.

8.1.1 Conversion to surface meshes

The basic conversion of a range image into a mesh is straightforward: pixels in the image are transformed to 3D coordinates using the range values and the sensor's intrinsic parameters $\mathcal{I}$. Triangles are formed by introducing edges between each pixel and its 4-connected neighbors and connecting the shortest diagonal of the resulting quadrilaterals. The main challenges are detection and removal of triangles that span range shadows and the handling of massive data sets produced by the Z+F and BF2 scanners.

8.1.2 Range shadow removal

Range shadows are discontinuities that occur at occluding boundaries in the scene. The basic mesh creation algorithm described above will generate triangles straddling the discontinuity, creating phantom surfaces. If these spurious triangles are not removed, they can cause problems in a number of places in our system. Specifically, they can decrease the success rate of pair-wise surface matching, reduce the accuracy of pair-wise and multi-view registration, cause correct matches to fail the local consistency test and correct model hypotheses to fail the global consistency test, and introduce errors in the reconstructed surface. Despite all these potential pitfalls, our system is robust to unremoved range shadow triangles. Nevertheless, it is in our best interest to detect and remove them when possible.

Range shadows can be detected based on triangle edge lengths or on the angle between the viewing direction and the surface normal. These methods capitalize on the fact triangles spanning a range shadow will be abnormally long and oriented nearly perpendicular to the viewing direction. We use the second method, angle thresholding, for scenes with a small depth of field (e.g., hand-held object modeling). A threshold of 75 to 80 degrees works well in practice.

These simple thresholding methods do not work as well for scenes with a large depth of field and obliquely viewed planar surfaces. This situation arises when modeling building interiors or terrain because the sensor is situated within the scene, rather than viewing it from a distance. Horizontal surfaces will be observed obliquely, especially at long ranges. With the angle or edge length thresholding methods, no fixed threshold correctly detects range shadows at long distances. Setting the threshold aggressively will eliminate all horizontal surfaces beyond a certain range, and setting it conservatively will fail to detect the range shadows entirely.

To solve this problem, we use an edge-detector to remove range shadows. Range shadows appear as step edges in the range image, but conventional gradient-based edge detection is not applicable due to the varying scene surface resolution over the image. Our approach is to use a one-dimensional model-based edge detector, which is applied independently to each adjacent pair of points in each row and column of the range image. Details are given in [33].

8.1.3 Other filtering operations

In addition to range shadow removal, a number of other operations are performed on the range data to remove invalid data. The operations used are sensor-specific and are implemented as a set of filters. The following is a list of filters used for the sensors used in this research:

Vivid

- **boundary filter** – Eliminates one layer of boundary points on the mesh. This filter is necessary because points near horizontal occluding boundaries have smaller range than expected. The cause is not certain, but one possible explanation is the sensor's range bias at low laser power levels (see figure A.3). As the laser stripe sweeps across an occluding boundary, only a portion of the stripe will strike the front surface. This thin stripe is equivalent to a lower laser power setting. Figure A.3 shows that measured range decreases by several millimeters at low laser power levels, which would lead to a curling effect at the horizontal occluding boundaries. Removing the boundary points is usually sufficient to eliminate the invalid data.

Z+F and BF2

- **range filter** – Removes points with range less than a given threshold - This filter eliminates points on the scanning platform (e.g., when the scanner is mounted on a vehicle or cart).
- **reflectance filter** – Removes points with low reflectance. This filter is necessary due to the scanner's ambiguity interval (section A.2). Points beyond the first ambiguity interval can generally be identified by their low reflectance.
- **median filter** – Removes points in noisy regions. When the sky is visible in a scan, the points are removed by a filter that eliminates points with a range value significantly different from the median range of the surrounding pixels.
- **blob filter** – Eliminates isolated points and fills in isolated holes.

8.1.4 Handling large data sets

The range images from high resolution 3D sensors give rise to large surface meshes that exceed the capabilities of our current computing hardware. For example, a 8000×1400-pixel range image from the Z+F equates to a mesh with over twenty million faces. Such a mesh would be very redundant because the sampled scene points are a function of the sensor geometry rather than scene geometry. We address this problem with a combination of range image down-sampling and mesh simplification. We use Garland's quadric algorithm [27] for mesh simplification for its speed and because it inherently simplifies low-complexity regions while maintaining the detail of high-complexity regions with more faces.

8.1.5 Additional mesh processing

At this point, we could use the meshes created as described above as input to our automatic modeling algorithms. However, we can obtain better surface-matching performance by applying mesh smoothing and additional mesh simplification.

- **mesh smoothing** - We use Taubin's algorithm [76] to smooth the meshes. Smoothing averages out uncorrelated noise and stabilizes the surface normals at the expense of some fine surface detail.

- **additional mesh simplification** - Since our surface matching engine is quadratic in mesh resolution, we create separate, low-resolution meshes for use by this algorithm. Experiments in section 7.2 showed that surface matching performance is nearly independent of resolution above 1000 faces. Clearly, the lower limit of resolution depends on scene geometry. In the future, we intend to replace this fixed resolution threshold with one based on the maximum displacement error between the original and simplified meshes.

8.2 Postprocessing operations

The absolute poses output by multi-view surface matching may be sufficient for some modeling applications, but in most cases, it is necessary to follow up with additional modeling operations. For example, in the reverse engineering application, shape primitives such as planes and cylinders would be fitted to the registered data. We have integrated two post-processing operations into our system: surface reconstruction and texture mapping.

8.2.1 Surface reconstruction

The purpose of surface reconstruction is to combine the data from multiple views into a single, compact description of the observed surfaces, eliminating the redundancy of the overlapping regions. Although there exist surface reconstruction algorithms that operate on unorganized sets of points [31][18], we focus on algorithms that take advantage of the structural information encoded in range images and surface meshes, since that is the form of our input data.

There are two main approaches to solving this problem: surface-based methods (also called parametric methods), which reconstruct directly from the input data [79][72], and volumetric methods, which reconstruct the surfaces by going through an intermediate, volume-based representation [15].

With surface-based methods, the task is to combine the data in overlapping regions. Turk and Levoy's zipper algorithm [79] operates on pairs of meshes. It first finds the topology of the surface by removing overlapping triangles and connecting the resulting boundaries together and then adjusts the vertices of the final mesh using a weighted average of nearby points from each input mesh. Soucy and Laurendeau [72] construct a Venn diagram of overlapping regions. For each region, a reference frame is defined, oriented in the average viewing direction of the participating views. The reconstructed surface is formed by a weighted average of the z-values of the views, and the surfaces from multiple regions are joined using Delauney triangulation.

Volumetric methods can be further broken down into two sub-categories: space-carving and implicit surfaces. In the space-carving category, Reed and Allen [64] use a volume intersection method. The mesh from each viewpoint is extruded along the viewing direction, and the final representation is the intersection of all extruded meshes. Pulli [61] developed a method that uses an oct-tree to represent the boundary between inside and outside of the model. Voxels are projected onto each range image and classified as inside, outside, or on the boundary, depending on depth relationship of the voxel and the range image data within the bounding box of the projection. Voxels on the boundary are repeatedly sub-divided until a maximum resolution is reached.

The methods in the implicit surface category create a vector field that accumulates some property (such as signed surface distance) for the region near each input surface. The zero crossings of this field represent the surface implicitly, and the surface can be extracted using methods such as the marching cubes algorithm [49]. Wheeler's consensus surface method [82] uses the signed distance to the closest consensus surface. A consensus surface for a point is a weighted average of nearby points that are considered to be on the same

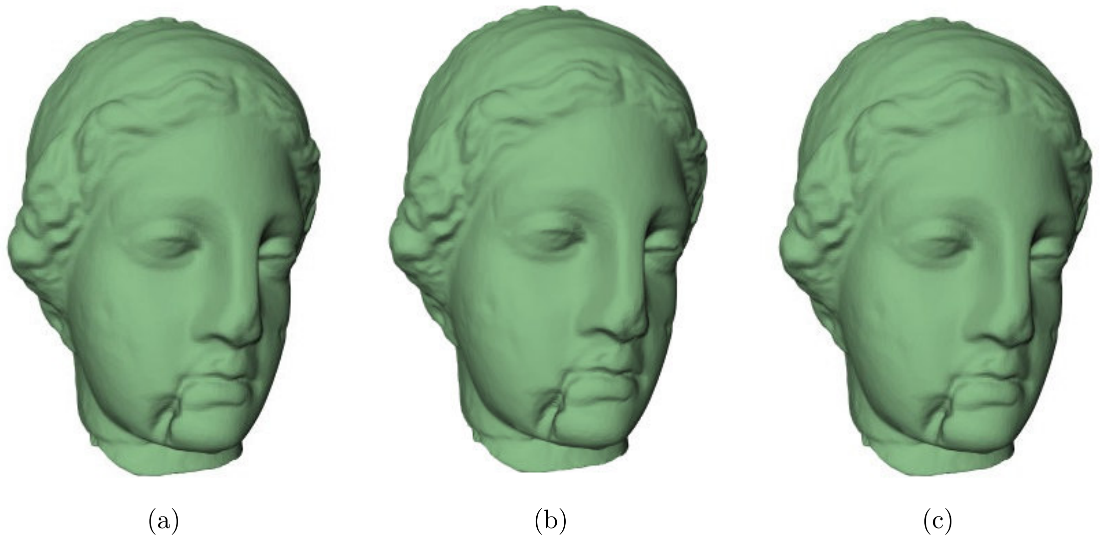


Figure 8.1: Surface reconstruction example. a) Original synthetic object. b) Registered views of the object. c) Result of surface reconstruction.

surface. Johnson [40] proposed a probabilistic occupancy grid method in which point data is convolved with two Gaussian filters that represent sensor uncertainty along the line of sight and surface continuity along the tangent plane. Curless' VRIP (Volumetric Range Image Processing) system [15] uses the signed distance to the nearest surface along the line of sight.

In our system, we use Curless' VRIP software, which is freely available on the Internet. The software is fast and produces geometrically accurate meshes for the noise level in our meshes (figure 8.1). The main disadvantage is that the implementation does not support the sensor models of our range scanners (perspective for the Vivid and cylindrical perspective for the Z+F and BF2). For the Vivid, we approximate with an orthographic sensor, and for the Z+F and BF2, we divide each image into several segments, each of which is approximated by an orthographic sensor.

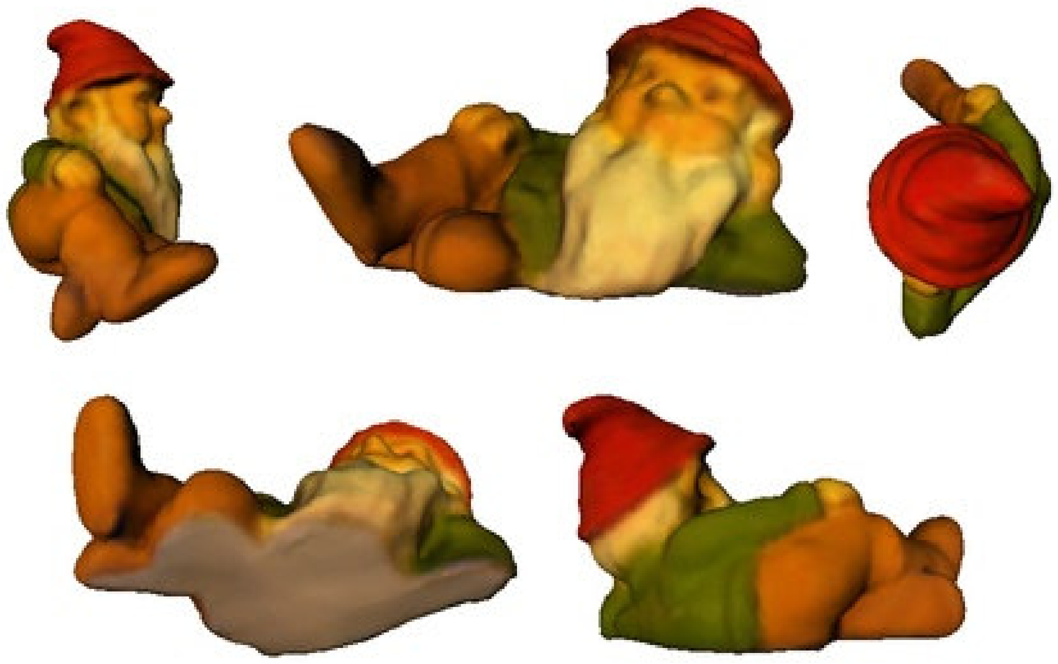
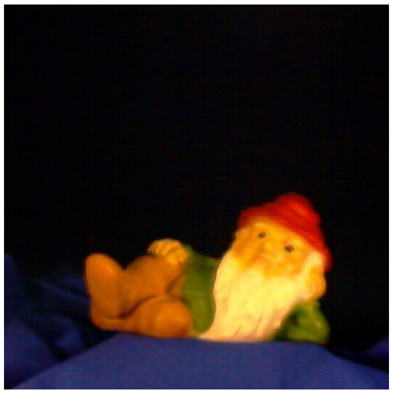


Figure 8.2: Texture mapping result. Several views of the texture mapped gnome.

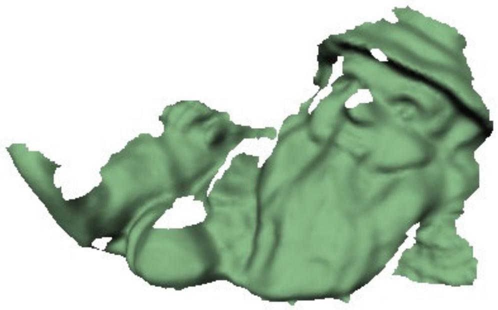
8.2.2 Texture mapping

We have applied texture mapping to some of our automatically modeled scenes. Although texture mapping is not the focus of this dissertation, the texture maps significantly improve the realism of the reconstructed scenes (figure 8.2). We generate texture maps for scenes scanned with the Vivid because it records registered color images of the scene during data collection. Unfortunately, the input images are slightly blurry due to the design of the sensor (the optics are optimized for the infrared laser), resulting in somewhat blurry texture maps.

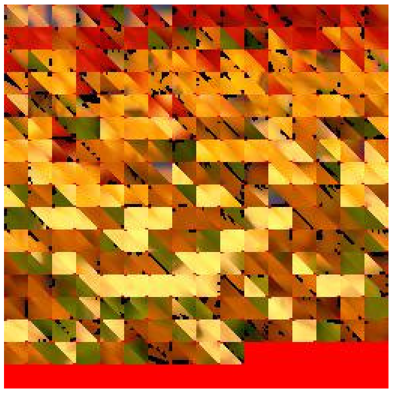
Our algorithm begins with a blank texture map that is divided into triangles – one for each triangle in the reconstructed mesh. The center of a pixel in the texture map corresponds to a 3D point on the mesh surface, and the question is "what color should this point be?" To answer this question, we project the 3D point into the range image of each view, using the range value for the projected point to determine whether the mesh point is



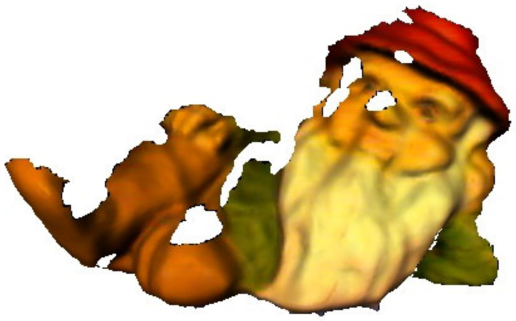
(a)



(b)



(c)



(d)

Figure 8.3: A texture mapping example using a single view. a) The color image for a view of the dwarf. b) The surface mesh for the view. c) An example texture map. An unusually low resolution surface mesh (250 faces) was used to artificially enlarge the texture map triangles for visualization only. d) The mesh from (b) with the texture map applied.

visible from a given view. The corresponding projected points in the color images for all visible viewpoints are then combined using a weighted average to determine the color of the mesh point, and the result is stored in the texture map. The weight for a component pixel is the cosine of the angle between the line of sight and the surface normal. This process is repeated for each pixel of the texture map (figure 8.3).

With hand-held modeling, it is important to have uniform lighting on the scene if texture mapping is to be used. Otherwise, the changing lighting conditions when the object is moved will cause dissimilarity among the pixels that correspond to the same object point, which reduces the realism of the textured object. We approximate uniform lighting with an array of three lights with diffusers.

Chapter 9

Conclusion

We conclude with a discussion of the significant contributions of this dissertation and the directions for future work.

9.1 Contributions

This dissertation contains four primary contributions:

- **Automatic modeling from reality and the multi-view surface matching problem** - Multi-view surface matching is a new class of registration algorithms that we have defined and placed in the context of existing registration algorithms. The theoretical problem of multi-view surface matching was motivated by the practical problem of automating modeling from reality. We have developed the first system that fully automates the modeling from reality process for 3D views obtained from unknown viewpoints.

- **Data driven probabilistic registration quality framework** - We have derived a novel, data-oriented method for implicitly modeling the effects of sensor noise, scene type, and the complex interaction of the component algorithms. Using this technique, we have developed statistical quality measures and methods to automatically determine the appropriate model parameters, eliminating the difficult problem of fine-tuning numerous “magic numbers.” We derived new quality measures based on the concept of visibility consistency and demonstrated that our framework provides a principled way to combine multiple, disparate measures of registration quality.

- **Multi-view surface matching search algorithms** - We developed deterministic and stochastic algorithms for efficiently searching for a correct model hypothesis in an exponentially large hypothesis space. The algorithms were shown to solve the multi-view surface matching problem for a variety of scenarios, sensors, and scene scales.

- **Model graph representation** - The model graph is a useful tool for representing and understanding the multi-view surface matching process, but it can also be applied to other 3D modeling problems, such as multi-view registration. Existing multi-view registration algorithms sometimes obscure or omit the method used for deciding which views may contain corresponding points. The model graph explicitly illustrates these topological relationships and supports geometric reasoning algorithms such as topological inference.

9.2 Future work

We have several ideas that we intend to pursue in the future. These ideas range from small updates to our existing framework to take advantage of additional model graph properties to a completely new framework that processes views sequentially.

9.2.1 Symmetry and differential ambiguity

Our experiments showed that highly symmetric surfaces are one of the main causes of failure for our multi-view surface matching algorithms. One possible solution is to explicitly detect symmetries within a view or a model hypothesis and incorporate knowledge of symmetries and their constraints into the search process. Kazhdan [42] has developed a 3D shape symmetry measure for fast object recognition. We could develop a localized version of such a descriptor applicable to individual views or model parts. Another idea is to modify our algorithms to output a distribution of model hypotheses rather than just the best one. The `random_swap` algorithm is essentially a Markov process, and as such, we could use Markov Chain Monte Carlo (MCMC) methods to generate a distribution of solutions.

Differentially ambiguous matches are another source of difficulty for our algorithms. The first step in handling underconstrained matches is to detect them. Simon [70] derived differential pose constraint measures for the purpose of selecting optimal control points in pair-wise registration. These measures could be used for detecting and classifying types of differential ambiguity. If we know which matches are differentially ambiguous, we can delay their inclusion in a model hypothesis until other, less ambiguous matches, have sufficiently constrained the model shape. Alternatively, we could consider explicitly including the differential constraints in the optimization process. Liu [48] showed that these constraints can be posed as a constraint satisfaction problem for the purpose of assembly planning.

9.2.2 Exploiting cycles

The cycles in a model graph contain information which we do not explicitly use. In our algorithms, topological inference re-establishes any correct matches that form a pose consistent cycle in G_{LR}. The fact that the pair-wise surface matching finds these inferred matches independently is independent evidence that the matches in a cycle are correct. Therefore, we hypothesize that the correctness of a cycle in G_{LR} could be efficiently determined by considering pose consistency of a cycle. When we defined pose consistency in chapter 2, it was a binary property – a model graph was either pose consistent or it was not. However, it is possible to relax that definition and, instead, define a measure of the *degree* of pose consistency for a cycle. One possible measure of pose consistency is the maximum displacement of a view's surface points when transformed by the relative pose defined by the cycle. This is the same as the relative pose maximum correspondence error (RMCE) that we used for evaluating the accuracy of our model hypotheses (section 6.1.1).

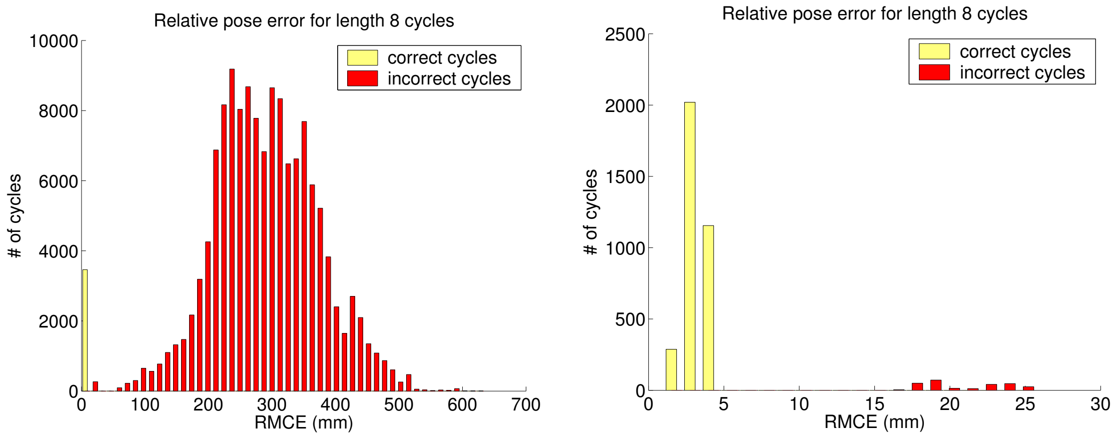


Figure 9.1: a) The distribution of relative pose errors for cycles of length 8 chosen randomly from the G_{LR} of the angel2 test set. b) A close up of the distribution in (a), showing the region close to zero.

We conducted a preliminary experiment to test this idea. In the experiment, we computed the pose consistency (i.e., RMCE) for random cycles of various lengths ($L = 3 \ldots N - 1$) chosen from a test model's G_{LR}. The cycles were classified as correct (those containing only correct matches) or incorrect (those with one or more incorrect match). For a given length, the distribution of pose consistency for each class shows what we expect – correct cycles have much lower RMCE than incorrect ones (figure 9.1). Surprisingly, the classes are easily separated for cycles even of length eight. This result suggests that the pose consistency of cycles could be beneficial in a multi-view surface matching algorithm.

There are, however, a couple of problems with using cycle pose consistency. First, there is no guarantee that every view will be part of a correct cycle in G_{LR}. For example, surface matching might correctly match a view with only one other view. In this case, no cycle consistency test could verify this view's correctness in a model hypothesis. This means that cycle consistency cannot solve the multi-view surface matching problem on its own. The second problem is potentially more damaging. A cycle can be almost perfectly pose consistent and yet still be incorrect. To see why this is so, consider a pair of views, V_i and V_j that results in an incorrect match. If the input data contains third view V_k obtained from the same (or nearly the same) position as V_i, then V_k and V_i are likely to be correctly matched (since they overlap almost completely). Furthermore V_k and V_j will probably be incorrectly matched in the same way as V_i and V_j (since V_i and V_k are almost the same). In such a situation, the cycle $V_i \rightarrow V_j \rightarrow V_k \rightarrow V_i$ will be incorrect but pose consistent. Consequently, we cannot rely entirely on pose consistency of individual cycles. Potentially, the pose consistency of multiple cycles in G_{LR} could be used to overcome this problem in the same way that our global consistency test overcomes the problem of incorrect matches that are undetectable with the local surface consistency test.

9.2.3 Scaling to large numbers of views (view selection)

In this work, our emphasis has been to show that the multi-view surface matching problem is solvable and that the solution can be robust across different sensors and problem domains. For this purpose, experiments on problems with a relatively small number of views are sufficient. However, if the algorithms are to be scaled to large numbers of views, we must face the problem of the $O(N^2)$ pair-wise matching.

Our proposed solution is to use intelligent view selection. Rather than exhaustively registering all pairs of views, heuristics can be employed to decide which views to register first. Incorporating view selection into our framework is equivalent to performing multi-view surface matching on a subset of the edges in G_{LR}. If we are unable to solve the problem using exhaustive matching, then we will also be unable to solve it using view selection. As such, view selection does not invalidate any of the results we have obtained with exhaustive registration.

We can imagine any number of view selection heuristics. Perhaps the most obvious heuristic is the temporal sequencing of the views. Views that are close together in the order that they were obtained are more likely to overlap than those that are widely separated in the sequence. Surface shape offers another possible heuristic. Views with similarly shaped regions are more likely to overlap, and therefore these view pairs should be matched first. This shape-based clustering of views is analogous to the technique of solving jigsaw puzzles by separating pieces according to their shape. The reconstructed models from each cluster can then be fed back as input views to our system and the entire process repeated. This interleaving of the local and global phases, which we call hierarchical modeling, can substantially reduce the total number of pair-wise registrations needed to construct a model. A third heuristic we could use is surface complexity. Low complexity surfaces, such as planes, are less likely to match correctly, since they can lead to differentially ambiguous matches. If we can develop a measure of surface complexity that predicts whether a match would be differentially ambiguous without actually performing the registration, then we could de-emphasize these views, matching other views first. Ideally, a more complete model will enable a successful match for the low complexity views later on. One possibility for implementing this would be to measure the differential ambiguity of the self-registration of subsets of a view's surface. These subsets would approximate the overlapping region that would been seen from other camera viewpoints.

9.2.4 Incorporating *a priori* knowledge

In some applications, we may have *a priori* knowledge of some of the unknowns in the multi-view surface matching problem. For example, there may be constraints on the absolute poses, such as an estimate of which direction is up (e.g., for sensors mounted on a vehicle or a tripod). Carmichael showed how these "common-sense constraints" can be incorporated into a pair-wise surface matching algorithm [9], and Johnson [37] developed similar method. Alternatively, an approximate relative pose may be available from odometry (e.g., if the sensor is mounted on a robot) . This type of constraint was used for pair-wise surface matching by Lu and Milios [50]. Overlap estimates provide a third type of *a priori* knowledge. For example, the user could specify some view pairs that are expected to contain overlapping regions (which is much easier than performing the manual registration). View pairs with overlap estimates would serve as a heuristic for view selection, these views would be registered first.

These three types of constraints can be incorporated into our framework as attributes of the model graph. In addition to improving the performance of pair-wise surface matching, the constraints could be integrated into the global consistency test. For example, the test would be able to immediately reject any model hypothesis that caused a view to violate the common-sense constraints for the sensor.

9.2.5 Sequential algorithms

The multi-view surface matching framework we have developed is batch-oriented – all the views are given at the start of the algorithm. In some applications, it is more appropriate to assume the views arrive one at a time. This would occur, for example, in a real-time modeling system, where the user expects immediate feedback on the model reconstruction as he collects the data. This feedback lets the user quickly identify holes in the model and obtain new scans to fill in the missing data. While the prospect of real-time automatic modeling seemed far-fetched at the onset of this line of research, processing speeds and our implementation of the component algorithms have progressed to the point where such a goal is now within reach.

Of the algorithms that we have developed, the `iterative_grow` algorithm is the most amenable to sequential arrival of views. The first view becomes the seed view to which the subsequent views are then joined. However, just because the views arrive sequentially does not mean that the full multi-view surface matching problem can be abandoned. If, for example, the first two views were to match incorrectly, the `iterative_grow` algorithm would not be able to recover. Therefore, we believe that such a system will still require the full multi-view surface matching algorithm to be used as a backup in the event that the sequentially constructed model fails the global consistency test.

Appendix A

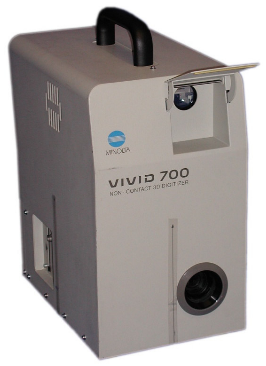
Range sensor characterization

The experiments and examples in this dissertation were conducted with three range sensors: the Minolta Vivid 700 (Vivid), the Zoller and Fröhlich LARA 25200 (Z+F), and the K2T Ben Franklin 2 (BF2). In this appendix, we review the capabilities and sensor geometry of each device and describe our noise analysis the Vivid.

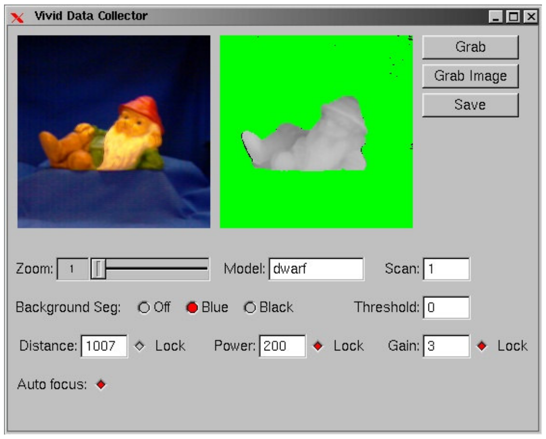
A.1 Minolta Vivid 700 (Vivid)

A.1.1 Sensor overview

The Vivid 700 (figure A.1a) operates by sweeping a laser line across the scene while viewing it from a CCD camera. It computes the range to a scene point by triangulation, using the known angles of the projected and received laser light and the relative position of the emitter and sensor. A scan takes 0.6 seconds and produces a 200×200-pixel range image with 8 bits of resolution and, since the sensor is a color CCD camera, a 400×400-pixel co-registered color image. The sensor geometry is that of a conventional camera – perspective projection. The operating range is 0.6 to 2.5 meters, suitable for scanning desktop objects.



(a)



Zoom: 1 Model: dwarf Scan: 1

Background Seg: ○ Off [✓ Blue] ○ Black Threshold: 0

Distance: 1007 ◇ Lock Power: 200 [✓ Lock] Gain: 3 [✓ Lock]

Auto focus: [✓]

(b)

Figure A.1: a) The Minolta Vivid 700 scanner. b) Our data collection user interface for the Vivid

A.1.2 Data collection user interface

We have implemented a user interface that simplifies data collection for the handheld modeling experiments (figure A.1b). The interface provides one-click scanning and saving of range images, allowing a typical object (≈ 20 views) to be scanned in about 3 minutes. The program also enables manual control of a number of scanner settings, including laser power, CCD gain, camera zoom, and approximate scene distance. We use a threshold on color image intensity or hue to filter background regions, which are masked with black or blue cloth. Background filtering is neccessary with handheld modeling to prevent extraneous surfaces from being included in the reconstructed model.

A.1.3 Noise analysis

We conducted several experiments in order to model the sensor noise and to determine the effect of various data collection parameters on the noise magnitude. Specifically, we investigated the effects of lighting, laser power, CCD gain, surface color, focal length, and range. We assume the noise is normally distributed with zero mean and variance σ^2.

In all experiments, unless otherwise noted, we scanned a planar, fronto-parallel, matte

parameter	default value
lights	off
laser power	200
CCD gain	3
object range	$\approx$ 2 meters
zoom	1
auto-focus	disabled

Table A.1: Default parameter values for the noise analysis experiments

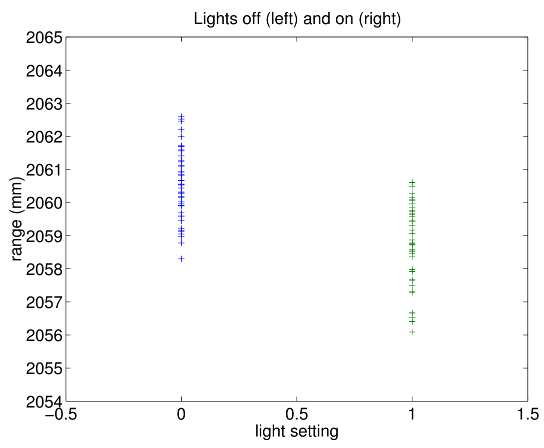
white target with scanning parameters shown in table A.1. For each experiment, we varied one parameter (e.g., CCD gain), and for each parameter setting, we acquired a set of 5 identical scans. To obtain a sufficiently large sample size, we use a 3x3-pixel patch from the center of each range image for a total of 45 samples per setting. We then compute σ for this set of samples and estimate 95% confidence intervals using bootstrap [19]. To determine whether varying a parameter has a significant effect on σ, we perform a Wald test on the difference between σ for pairs of settings. The null hypothesis is that the difference is zero (i.e., the change of setting has no effect on the noise). For a size α Wald test with $\alpha = 0.05$, $z > 1.96$ indicates that the change of setting has a significant effect on σ.

Lighting

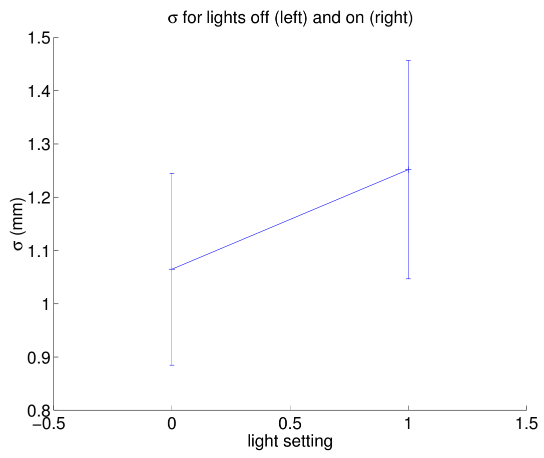
Subjectively, the noise level appears to be reduced when the room lighting is turned off. This experiment tests the significance of this effect using the two lighting settings: on and off. Although the noise does increase when the lights are on (figure A.2), the effect is not significant (z = 1.33).

Laser power

Increasing the laser power allows surfaces with lower reflectance to be scanned, but the question is whether this benefit comes at the expense of increased noise. For this experiment, we varied the laser power in 50-unit increments from 50 to 250. The results (figure A.3) show that the effect of laser power is not significant in general, but the Wald test does



(a)



(b)

Figure A.2: The effect of lighting on noise level. a) Individual range measurements for the lighting experiment. b) Noise (σ) and 95% confidence intervals for each lighting setting.

indicate that the power=150 setting is significantly different from the 50 and 200 settings (z = 2.49 and 2.30 respectively).

CCD gain

Higher CCD gain also allows low-reflectance surfaces to be scanned. The CCD gain has five settings of increasing gain (0 to 4). The results (figure A.4) indicate that the effect of CCD gain is not significant except at the highest setting (z = 3.21 between settings 3 and 4).

Surface color

In this experiment, we varied the surface material, expecting that lower-reflectance materials would have higher noise levels, since the laser stripe would be harder to detect. Using the materials shown in table A.2, we found that the noise can vary by a factor of 6 between the best case (matte white posterboard) and the worst case (black fabric) (figure A.5). We also noticed a large variation in absolute range (nearly 4cm difference between black and green).

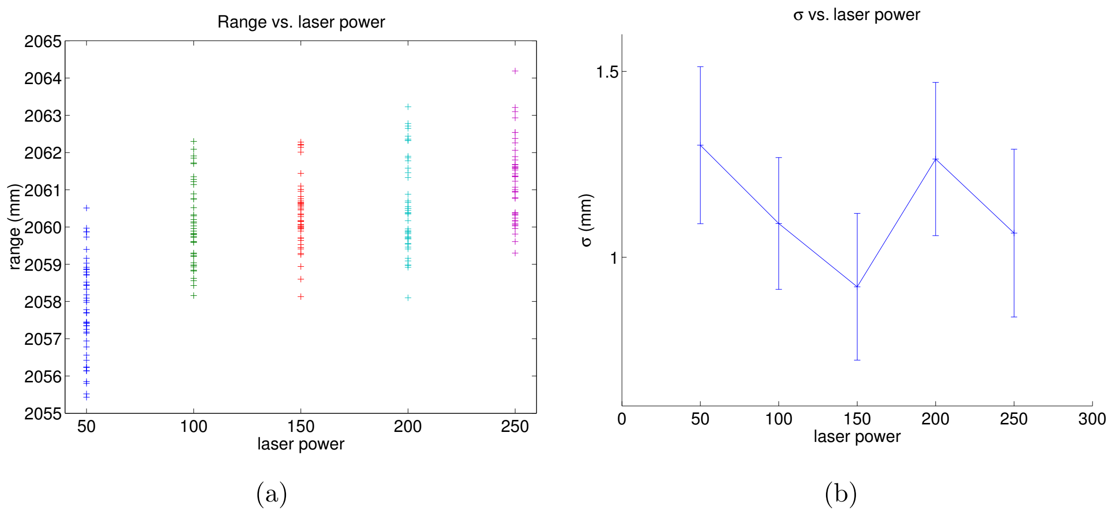


Figure A.3: The effect of laser power. a) Individual range measurements for the laser power experiment. b) Noise (σ) and 95% confidence intervals for each laser power setting.

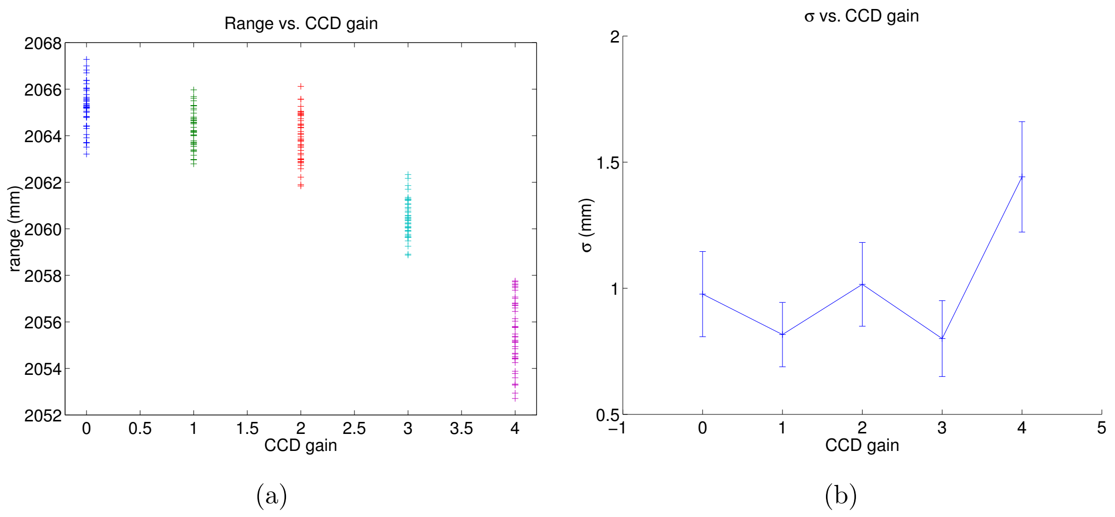
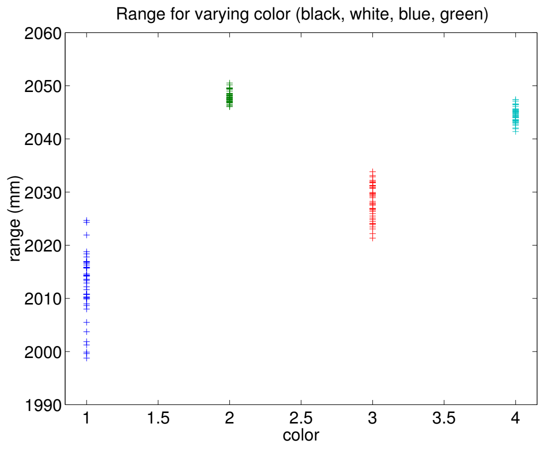
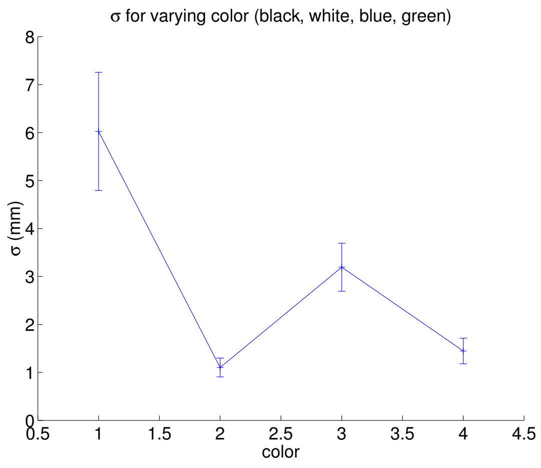


Figure A.4: The effect of CCD gain. a) Individual range measurements for the CCD gain experiment. b) Noise (σ) and 95% confidence intervals for each CCD gain setting.



(a)

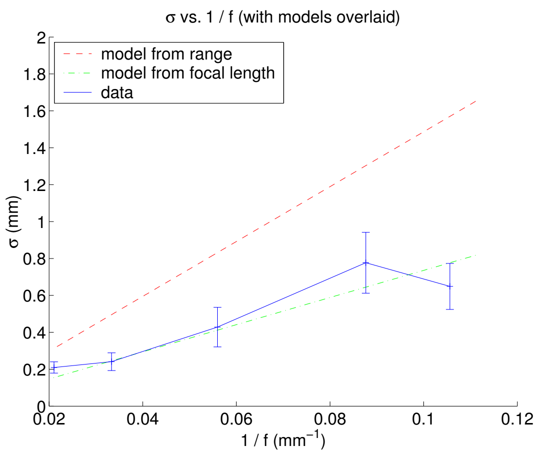


(b)

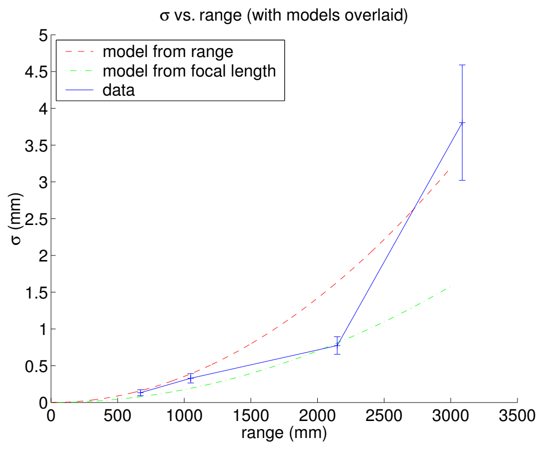
Figure A.5: The effect of material color. a) Individual range measurements for the material color experiment. b) Noise (σ) and 95% confidence intervals for each material color setting.

data set	color	σ (mm)
1	black fabric	6.03
2	white posterboard	1.10
3	blue fabric	3.19
4	green fabric	1.45

Table A.2: Materials used in the surface color experiment and the resulting noise levels.



(a)



(b)

Figure A.6: The effect of range and focal length on noise. a) Noise (σ) as a function of inverse focal length with the best fitting noise models. b) σ as a function of range with the best fitting noise models overlaid. The measured settings are shown with 95% confidence intervals.

Focal length and range

Based on the geometry of the sensor, which is the same as that of a stereo vision system, the theoretical relationship between focal length (f), range (r), and noise (σ) is

$$\sigma = k\frac{r^2}{f}, \tag{A.1}$$

where k is a constant. For fixed range, σ should be linear in $1/f$. Using the settings in table A.3, we obtained the results in figure A.6a. This is a reasonable, but not great, approximation to a linear relationship. Linear regression gives the best fitting model for the data

$$\sigma = k_{\mathrm{z}}\frac{r_0^2}{f}, \tag{A.2}$$

where $k_{\mathrm{z}} = 1.67E - 6$ and $r_0 = 2100\text{mm}$. Similarly, for constant focal length, σ should be quadratic in range. Using the settings in table A.4, we obtained the results in figure A.6b. The least squares approximation of the form

$$\sigma = k_{\mathrm{r}}\frac{r^2}{f_0}, \tag{A.3}$$

zoom	focal length (mm)	$1/f(mm^{-1})$	σ (mm)
1	9.47	0.1056	0.649
2	11.41	0.0876	0.777
4	17.87	0.0560	0.428
6	29.99	0.0333	0.241
8	47.51	0.0210	0.210

Table A.3: Zoom values used in the focal length experiment and the associated values of σ.

range (mm)	σ (mm)
671.5	0.133
1049	0.330
2148	0.775
3089	3.81

Table A.4: Range values used in the range experiment and the associated values of σ.

where $k_{\mathrm{r}} = 3.37e - 6$ and $f_0 = 9.50$mm is shown overlaid on the plot. Again, the model is not an excellent fit, but it will suffice for a rough approximation.

Ideally, k_{z} and k_{r} should be equal. The fact that they are not indicates that either equation A.1 does not fully capture the noise dependencies or some aspect of the data collection is inaccurate. To visualize the effect of this discrepancy on our noise model, we plot the focal length model (equation A.2) using k_{r} in figure A.6a and the range model (equation A.3) using k_{z} in figure A.6b. We observe that k_{z} fits the range model well, but k_{r} overestimates the noise in the focal length model.

Conclusions

Our experiments show that the noise is primarily a function of surface reflectance, focal length, and range. We can model the effect of focal length and range using equation A.1 with $k = k_{\mathrm{z}} = 1.67E - 6$. This model applies to matte white objects. For low surface reflectance scenes, k should be scaled by a factor of three to six.

From a noise reduction standpoint, the best strategy for scanning is to place the sensor as close as possible to the scene. In our experiments, we scan at a range of about 0.5 to 1.0

meter. For this range, a constant sigma of 0.3 mm for white objects and 1mm for darker objects is a reasonable worst case approximation.

A.2 Zoller and Fröhlich LARA 25200 (Z+F)

The Z+F LARA 25200 is an amplitude modulated continuous wave (AMCW) laser scanner. The scanner's laser emits light with sine-wave variation in power and detects the reflected signal. The phase shift between the detected waveform and the waveform being emitted is proportional to the distance to the scene. This phase difference measurement implies that a $360°$ phase shift will be indistinguishable from a $0°$ phase shift. The range where this aliasing effect occurs is called the ambiguity interval. Points at distances beyond this range "wrap around" and will be reported to have a range equal to the true range modulo the ambiguity interval. We eliminate points beyond the first ambiguity interval by filtering low-reflectance points in the preprocessing stage (section 8.1).

The scanner has a $360 \times 70°$ field of view and a 25.2 meter ambiguity interval. It has a data rate of 120,000 samples/second, acquiring an 8000×1400-pixel scan in 93 seconds. Each point in the scan contains a 15 bit range measurement and 16 bit reflectance value. The noise on the range measurements is zero mean Gaussian with a standard deviation of 2.67mm for a 20% reflective surface at 13 meters.

A.3 K2T Ben Franklin2 (BF2)

The Ben Franklin 2 scanner (BF2) is the predecessor of the Z+F. It has the same basic design, but with different performance characteristics. The BF2 has a smaller vertical field of view ($30°$), a larger ambiguity interval (52 meters), and produces smaller range images (6000×300 pixels).

Appendix B

Complexity of multi-view surface matching

We will show that a simplified version of multi-view surface matching is NP-complete. First, let us assume that our pair-wise surface matching algorithm makes no type 1 errors. That is, it always finds a correct match between pairs of views that actually do overlap. Second, let us assume that we have a polynomial time function, $C_{\mathrm{G}}(G)$ that detects inconsistent solutions once the absolute poses of all views have been specified. We can now write this simplified multi-view surface matching problem as a decision problem:

> **The simplified multi-view surface matching problem (SURF-MATCH)**
> Instance: A model graph G with relative poses specified, and a global consistency test C_{G}.
> Question: Does G contain a spanning tree G' such that $C_{\mathrm{G}}(G')$ is true?

Theorem B.1. *SURF-MATCH is NP-complete.*

Proof: Clearly the problem is in NP. Given a potential solution G', we simply output the value of $C_{\mathrm{G}}(G')$, which can be computed in polynomial time by assumption.

We transform an instance of the partition problem into SURF-MATCH. The partition problem is defined as follows:

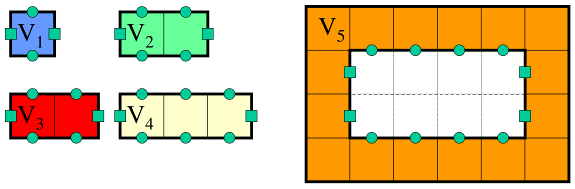


Figure B.1: Example of the input views for the SURF-MATCH instance corresponding to the partition problem $A = \{1, 2, 2, 4\}$. The four pieces are on the left, and the boundary is on the right.

The partition problem (PARTITION)

Instance: Given a finite set $\mathcal{A}$ and a "size" $s(a) \in Z^+$ for each $a \in A$.

Question: Is there a subset $\mathcal{A}' \subseteq \mathcal{A}$ such that $\sum_{a \in \mathcal{A}'} s(a) = \sum_{a \in \mathcal{A} - \mathcal{A}'} s(a)$?

Our reduction relies on the jigsaw puzzle analogy from section 1.1. We will construct a $2\frac{1}{2}$D object that, if assembled correctly, would also solve the corresponding partition problem.

Let $K = \frac{1}{2} \sum_{a \in A} s(a)$ and $N = |A|$. For each element $a \in A$, create a view V_i consisting of a rectangular planar surface 1 unit tall and $s(a)$ units wide (figure B.1, left). We call these views *pieces* since they act as the pieces of our puzzle. Additionally, create one view V_{n+1}, which we will call the boundary. The boundary is 4 units tall and $K + 2$ units wide with a 2 unit tall by K unit wide hole removed from its center (figure B.1, right).

The pieces are constructed in a manner that limits the orientation and alignment of pair-wise matches to 0 or 180 degree rotation and integer values of horizontal and vertical translation. Specifically, there are two types of features built into the edges of the pieces and the interior edges of the boundary: square features and round ones. These features are designed such that the overlapping regions of two views will be consistent if and only if the features overlap and have the same type. Square features are built into the vertical edges of each piece and the interior vertical edges of the boundary as shown figure B.1. Similarly, round features are built into the horizontal edges of each piece and the interior horizontal edges of the boundary. Additionally, the surfaces of the pieces each contain a unique pattern such that a pair will be inconsistent if they overlap at any point other than

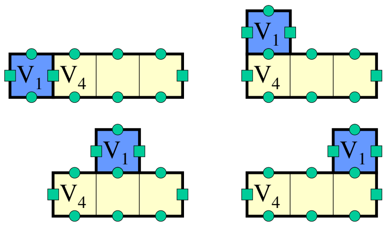


Figure B.2: Four of the allowable alignments between V_1 and V_4.

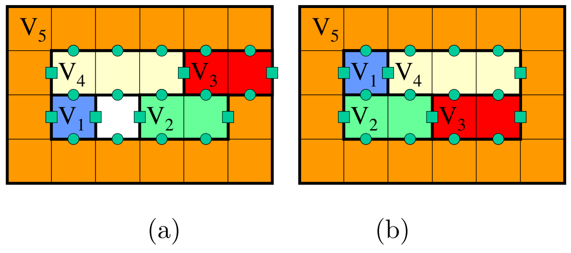


Figure B.3: Two possible solutions to the SURF-MATCH instance. a) An incorrect solution – V_3 overlaps the boundary and the bottom row is not tightly packed. b) A correct solution.

the aforementioned features.

We now construct the model graph G using the $N + 1$ views and adding edges for all possible pair-wise matches between each pair of pieces (figure B.2). If the size of two pieces is s_a and s_b, there are $4(s_a + s_b)$ matches. Similarly, we add edges for all matches between each piece and the boundary. For a piece of size s_a, there are $4(K - s_a + 1)$ such matches. It is not hard to see that this instance of SURF-MATCH can be constructed in polynomial time from the partition instance.

All that remains is to show that we have a solution to PARTITION if and only if we have a solution to SURF-MATCH. Given a solution to PARTITION, a SURF-MATCH solution can be constructed as follows: Let $\mathcal{V}_a$ be the set of views corresponding to A' and $\mathcal{V}_b$ be the views corresponding to $A - A'$. We select edges from G that tightly pack the views from $\mathcal{V}_a$ in the top row of the boundary piece. (Tightly packed means that each piece is adjacent to another piece on the left or right side.) Similarly, we select edges from G that tightly pack the views in $\mathcal{V}_b$ in the bottom row of the boundary piece (figure B.3).

Since $\sum_{a \in \mathcal{A}'} s(a) = \sum_{a \in \mathcal{A} - \mathcal{A}'} s(a) = K$, the pieces in each row will exactly fit inside the boundary. In the reverse direction, suppose we have a solution to SURF-MATCH. The pieces in the top row are converted into their corresponding members of the set $\mathcal{A}$ and are placed in $\mathcal{A}'$. Since the total width of the top row pieces is K, $\sum_{a \in \mathcal{A}'} s(a) = K$. A similar process for the bottom row pieces shows that $\sum_{a \in \mathcal{A} - \mathcal{A}'} s(a) = K$ as well. $\square$

Bibliography

[1] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. AVENUE: Automated site modeling in urban environments. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 357–364, May 2001.

[2] Raouf Benjemaa and Francis Schmitt. A solution for the registration of multiple 3D point sets using unit quaternions. In *Proceedings of the 5th European Conference on Computer Vision (ECCV '98)*, pages 34–50, June 1998.

[3] Robert Bergevin, Marc Soucy, Herve Gagnon, and Denis Laurendeau. Towards a general multi-view registration technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):540–7, May 1996.

[4] F. Bernardini and H. Rushmeier. Strategies for registering range images from unknown camera positions. In *Proceedings of Three-Dimensional Image Capture and Applications III*, pages 200–206, January 2000.

[5] Paul Besl and Neil McKay. A method of registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[6] Christopher Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[7] Robert Bolles and Patrice Horaud. 3DPO: A three-dimensional part orientation system. *International Journal of Robotics Research*, 5(3):3–26, Fall 1986.

[8] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for the Space Carving algorithm. In *Proceedings of the Eigth IEEE International Conference on Computer Vision (ICCV '01)*, pages 388–393, July 2001.

[9] Owen Carmichael and Martial Hebert. Unconstrained registration of large 3D point sets for complex model building. In *Proceedings 1998 IEEE/RSJ International Conference On Intelligent Robotic Systems*, 1998.

[10] Owen Carmichael, Daniel F. Huber, and Martial Hebert. Large data sets and confusing scenes in 3D surface matching and recognition. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 358–367, October 1999.

[11] Yang Chen and Gerard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–55, April 1992.

[12] Chin Seng Chua and Ray Jarvis. Point signatures: a new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85, October 1997.

[13] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[14] John Craig. *Introduction to Robotics – Mechanics and Control*. Addison-Wesley, 1989.

[15] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of ACM SIGGRAPH*, pages 303–312, 1996.

[16] Paul Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH*, pages 11–20, August 1996.

[17] Frank Dellaert, Steven Seitz, Charles Thorpe, and Sebastian Thrun. Structure from motion without correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2000.

[18] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[19] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, 1994.

[20] David Eggert, Fitzgibbon, and Robert Fisher. Simultaneous registration of multiple range views for use in reverse engineering of CAD models. *Computer Vision and Image Understanding*, 69(3):253–72, March 1998.

[21] S. F. El-Hakim, P. Boulanger, F. Blais, and J.-A. Beraldin. A system for indoor 3D mapping and virtual environments. In *Proceedings of Videometrics V (SPIE vol. 3174)*, pages 21–35, July 1997.

[22] W. Matusik et al. Image-based visual hulls. In *Proceedings of ACM SIGGRAPH*, pages 369–74, July 2000.

[23] O. Faugeras and M. Hebert. The representation, recognition, and positioning of 3d shapes from range data. *International Journal of Robotics Research*, 5(3):27–52, Fall 1986.

[24] F.P. Ferrie and M.D. Levine. Integrating information form multiple views. In *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, pages 117–22, November 1987.

[25] A. W. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-D reconstruction of independently moving objects. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 891–906, June 2000.

[26] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Grpahics Principles and Practice*. Addison-Wesley, 1987.

[27] Michael Garland and Paul Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH*, pages 209–16, August 1997.

[28] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the Twenty-Fifth International Conference on Very Large Data Bases*, pages 518–29, September 1999.

[29] Jacob Goldberger. Registration of multiple point sets using the EM algorithm. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 730–6, September 1999.

[30] K. Higuchi, Martial Hebert, and Katsushi Ikeuchi. Building 3D models from unregistered range images. *Graphical Models and Image Processing*, 57(4):315–333, July 1995.

[31] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of ACM SIGGRAPH*, pages 71–78, July 1992.

[32] Daniel F. Huber. Automatic 3D modeling using range images obtained from unknown viewpoints. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 153–160. IEEE Computer Society, May 2001.

[33] Daniel F. Huber and Martial Hebert. A new approach to 3D terrain mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, pages 1121–1127, October 1999.

[34] Daniel F. Huber and Martial Hebert. Fully automatic registration of multiple 3D data sets. In *IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum (CVBVS 2001)*, December 2001.

[35] Katsushi Ikeuchi and Yoichi Sato, editors. *Modeling from Reality*. Kluwer Academic Publishers, 2001.

[36] Andrew Johnson. *Spin-Images: A Representation for 3D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[37] Andrew Johnson. Surface landmark selection and matching in natural terrain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 413–20, June 2000.

[38] Andrew Johnson, Owen Carmichael, Daniel F. Huber, and Martial Hebert. Toward a general 3D matching engine: Multiple models, complex scenes, and efficient data filtering. In *Proceedings of the 1998 Image Understanding Workshop (IUW)*, pages 1097–1107, November 1998.

[39] Andrew Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–49, May 1999.

[40] Andrew Johnson and Sing Bing Kang. Registration and integration of textured 3D data. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 234–41, May 1997.

[41] Takeo Kanade, H. Kato, S. Kimura, A. Yoshida, and K. Oda. Development of a video-rate stereo machine. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, volume 3, pages 95 – 100, August 1995.

[42] Michael Kazhdan, Bernard Chazelle, David Dobkin, Thomas Funkhouser, and Szymon Rusinkiewicz. A reflective symmetry descriptor for 3d models. In *Algorithmica*, 2003. (to appear).

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–80, May 1983.

[44] Kiriakos Kutulakos and Steven Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–219, 2000.

[45] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–62, February 1994.

[46] Marc Levoy. The digital michelangelo project. In *Second International Conference on 3D Digital Imaging and Modeling*, pages 2–11, October 1999.

[47] Marc Levoy, Kari Pulli, Brian Curless, and Szymon Rusinkiewicz et al. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of ACM SIGGRAPH*, pages 131–144, 2000.

[48] Y. Liu and R.J. Popplestone. Planning for assembly from solid models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 222–7, May 1989.

[49] William Lorensen and Harvey Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–9, July 1987.

[50] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–49, October 1997.

[51] Daisuke Miyazaki, Takeshi Ooishi, Taku Nishikawa, Ryusuke Sagawa, Ko Nishino, Takashi Tomomatsu, Yutaka Takase, and Katsushi Ikeuchi. The great buddha project: modelling cultural heritage through observation. In *Proceedings of the 6th International Conference on Virtual Systems and MultiMedia*, pages 138–45, Oct 2000.

[52] Daniel D. Morris, Kenichi Kanatani, and Takeo Kanade. Gauge fixing for accurate 3D estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–50, December 2001.

[53] Peter Neugebauer. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *International Journal of Shape Modeling*, 3(1 & 2):71–90, 1997.

[54] K. Nishino and K. Ikeuchi. Robust simultaneous registration of multiple range images. In *Proceedings of the Fifth Asian Conference on Computer Vision*, pages 454–461, January 2002.

[55] K. Nishino, Y. Sato, and K. Ikeuchi. Eigen-texture method: appearance compression and synthesis based on a 3d model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1257–65, November 2001.

[56] Lars Nyland, David K. McAllister, Voicu Popescu, Chris McCue, and Anselmo Lastra. Interactive exploration of acquired 3D data. In *Proceedings of the 28th AIPR Workshop: 3D Visualization for Data Exploration and Decision Making*, pages 46–57, October 1999.

[57] F. Pipitone and W. Adams. Tripod operators for recognizing objects in range images: rapid rejection of library objects. In *Proceedings of the IEEE Conference on Robotics and Automation*, volume 2, pages 1596–1601, May 1992.

[58] Marc Pollefeys, Reinhard Koch, Maarten Vergauwen, and Luc van Gool. Metric 3D surface reconstruction from uncalibrated image sequences. In Reinhard Koch and Luc van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments. European Workshop, SMILE'98*, pages 139–54. Springer-Verlag, June 1998.

[59] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry – An Introduction*. Springer-Verlag, 1985.

[60] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 1992.

[61] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 205–11, May 1997.

[62] Kari Pulli. Multiview registration for large data sets. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 160–8, October 1999.

[63] Peter Rander. *A Multi-Camera Method for 3D Digitization of Dynamic, Real-World Events*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1998.

[64] M. K. Reed and P. K. Allen. 3D modeling from range imagery: an incremental method with a planning component. *Image and Vision Computing*, 17(2):99–111, February 1999.

[65] Philippe Robert and Damien Minaud. Integration of multiple range maps through consistency processing. In Reinhard Koch and Luc van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments. European Workshop, SMILE'98*, pages 253–65. Springer-Verlag, June 1998.

[66] Szimon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 145–52, May 2001.

[67] Szymon Rusinkiewicz. *Real-time Acquisition and Rendering of Large 3D Models*. PhD thesis, Stanford University, Stanford, California, August 2001.

[68] Hideo Saito, Shigeyuki Baba, Makoto Kimura, Sundar Vedula, and Takeo Kanade. Appearance-based virtual view generation of temporally-varying events from multi-camera images in the 3D room. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 516 – 525, October 1999.

[69] Harpreet Sawhney, Steve Hsu, and R. Kumar. Robust video mosaicing through topology inference and local-to-global alignment. In *Proceedings of the 5th European Conference on Computer Vision (ECCV '98)*, pages 103–19, June 1998.

[70] David Simon. *Fast and Accurate Shape-Based Registration*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1996.

[71] David Simon, Martial Hebert, and Takeo Kanade. Real-time 3d pose estimation using a high-speed range sensor. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '94)*, volume 3, pages 2235–2241, May 1994.

[72] Marc Soucy and Denis Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):344–58, April 1995.

[73] Fridtjof Stein and Gerard Medioni. Structural indexing: efficient 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–45, February 1992.

[74] A. J. Stoddart and A. Hilton. Registration of multiple point sets. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 40–4, August 1996.

[75] Y. Sun and M.A. Abidi. Surface matching by 3d point's fingerprint. In *Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV '01)*, pages 263–9, July 2001.

[76] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 351–8, August 1995.

[77] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[78] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – A modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, September 1999.

[79] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of ACM SIGGRAPH*, pages 311–18, July 1994.

[80] Sundar Vedula, Simon Baker, and Takeo Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June 2002.

[81] T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 541–55, May 2002.

[82] M. Wheeler, Yoichi Sato, and Katsushi Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 917–24, January 1998.

[83] Mark Wheeler. *Automatic modeling and localization for object recognition*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, October 1996.

[84] J.V. Wyngaerd, L. Van Gool, R. Kock, and M. Proesmans. Invariant-based registration of surface patches. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 301–6, September 1999.

[85] Dongmei Zhang and Martial Hebert. Harmonic maps and their applications in surface matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 524–30, 1999.

[86] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, October 1994.

[87] Conversation with ted blackman.

[88] Cyra web site – www.cyra.com.

[89] Quantapoint web site – www.quantapoint.com.