Algorithms for Comparing and Analyzing 3D Geometry

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Niloy J. Mitra
September 2006

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Leonidas J. Guibas Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marc Levoy Associate Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mark Pauly

Approved for the University Committee on Graduate Studies.

Abstract

The world around us consists of objects of vastly varying shapes, sizes, and geometric complexity. A natural question is how to capture such geometric variations? We propose algorithms for capturing, comparing, and analyzing such 3D geometry.

In the first part of this thesis, we propose algorithms to automate the various stages of a standard *shape acquisition pipeline*. Typically, a 3D scanner captures object geometry from multiple directions. From each viewpoint we get a partial geometric model or a *scan*. Scan *registration* involves stitching these scans together to form one consolidated model. We present an algorithm for the automatic *global rigid alignment* of two 3D shapes, without any assumptions about their initial poses. We implicitly evaluate all the possible correspondence assignments between a set of selected feature points without having to explicitly enumerate each of the correspondences. The rough initial alignment is further refined using *local alignment* which is posed as a minimization of the *squared distance* between the underlying surfaces. We locally approximate the squared distance field using quadratic functions and then develop a linear system whose solution gives the local aligning rigid transform. After model registration, we often get an incomplete representation of the scanned object with areas of missing data. We present a novel approach for plausible *3D model completion* using geometric priors. Our method retrieves suitable context models from a model database, warps the retrieved models to conform with the input data, and consistently blends the warped models to obtain the final consolidated 3D shape.

In the second part, we introduce two shape analysis tools. We present a new algorithm that processes geometric models and efficiently discovers and extracts a compact representation of their Euclidean *symmetries*. These symmetries can be partial, approximate, or both. The extracted symmetry graph representation captures important high-level information

about the structure of a geometric model. We also propose a compact shape signature based on *probabilistic fingerprints*. Our method is robust to noise, invariant to rigid transforms, handles articulated deformations, and effectively detects partial matches. These compact fingerprints are used to efficiently estimate similarity across multiple 3D shapes where directly evaluating similarity is expensive and impractical.

We demonstrate the utility of all our algorithms for a wide variety of geometry processing applications on a range of scanned geometric models of varying sizes, complexity, and details.

Acknowledgements

This thesis is the result of my work at Stanford whereby I have been accompanied and supported by many people. Here is my opportunity to express my gratitude for all of them.

The first person I would like to thank is my advisor Leonidas Guibas. I consider myself extremely lucky for being one of Leo's students. He has always been a source of inspiration and support over these years, been extremely patient with me, taught me how to conduct research, present research ideas, and ask the right questions. I cannot possibly express my gratitude for him in words.

I would like to thank Mark Pauly with whom I have worked extensively. Working with Mark is a pleasure. The long hours we spend discussing algorithms, tuning minor details, fixing codes, and running experiments resulted in a significant part of the work presented in this thesis.

I would also like to thank the other faculty members with whom I had the privilege of working during the course of my PhD life: Gunnar Carlson, Pat Hanrahan, Jean-Claude Latombe, Marc Levoy, Rajeev Motwani from Stanford, Joachim Giesen from Max-Planck-Institut für Informatik, Markus Gross from ETH Zurich, and Helmut Pottmann from Vienna University of Technology. Also I acknowledge the help received from the following: Pierre Alliez, Mario Botsch, Michael Hoffer, Richard Keiser, Doo Young Kwon, Bob Sumner, and Martin Wicke. I also thank Manuela Cavegn, John Gerth, Ada Glucksman, Heather Gentner, and Hoa Nguyen for their help with administrative tasks. I acknowledge the support from the funding agencies: Darpa, ITR, NIH, and NSF. I express my sincere thanks to the Joseph W. and Hon Mai Goodman Stanford Graduate Fellowship that enabled me to work on topics I felt excited about.

I would like to thank Natasha Gelfand and An Nguyen for working with me on various

topics during my PhD years. I would like to specially thank Qing Fang, Daniel Russel, and Afra Zomorodian for the many hours we spend together discussing and arguing over things not related to my work. I express my many thanks to all the present and past members of the Leo Guibas group, and also my friends in the Stanford Graphics lab for the good and bad times we spend together leaving me with lots of fond memories to cherish.

I take this opportunity to thank a few of my friends: Anupam Datta, Amal Ekbal, Gaurav Garg, Mahesh Hardikar, Inam Rehman, Debasis Sahoo, and Padma Sundaram. Also thanks to the many others from the Stanford Cricket Club, the Stanford Alpine Club, the Stanford Climbing Gym, and the Stanford Outing Club for helping me have fun, enjoy life, and remain sane during my stay at Stanford.

I dedicate this thesis to my family: my parents, my brother, and my longtime girlfriend and now fiancé, Devasree. This is a good place to express my sincere thanks to my parents for always being by my side, for giving me a good education, and encouraging me to pursue my dreams. I owe it to them for any good values that I have in myself. My brother has always been my good friend who taught me to question all things around and think logically. Kudos to Devasree for her patience during my long PhD years, helping me keep faith in myself, and for all her love and tenderness.

Thank you all, without your help I would not have been here today.

Contents

List of Tables

List of Figures

1

Introduction

Inspiration is needed in geometry, just as much as in poetry.

— Alexander Pushkin

In the last decade, 3D geometry has emerged as an ubiquitous digital medium. It is now easy to generate 3D models — we can either acquire complex geometry of real world objects using 3D scanners, or use state-of-the-art modeling tools to create complex geometric models. Accessibility of such techniques has resulted in a fast growing collection of 3D shapes. Significant effort is being devoted towards organizing and categorizing such large volume of models. Efficient handling of such databases requires a notion of similarity score and effective methods for computing it. An useful notion of similarity should be invariant to initial model poses, robust to perturbations, capable of handling articulations, and most importantly, work when the match is only *partial*. In this thesis, we investigate shape similarity and use our understanding to solve various geometry processing problems including global and local alignment of shapes, shape completion, shape analysis using symmetry detection, and partial shape retrieval.

The concept of shape similarity is not well defined, because *similarity* means different things for different applications. Thus it is unlikely that a single approach can handle all shape matching problems. Even for other types of digital media like text, audio, images partial similarity computation is an active area of research and meaning of similarity changes significantly with applications. Our problem is significantly more challenging, as digital 3D

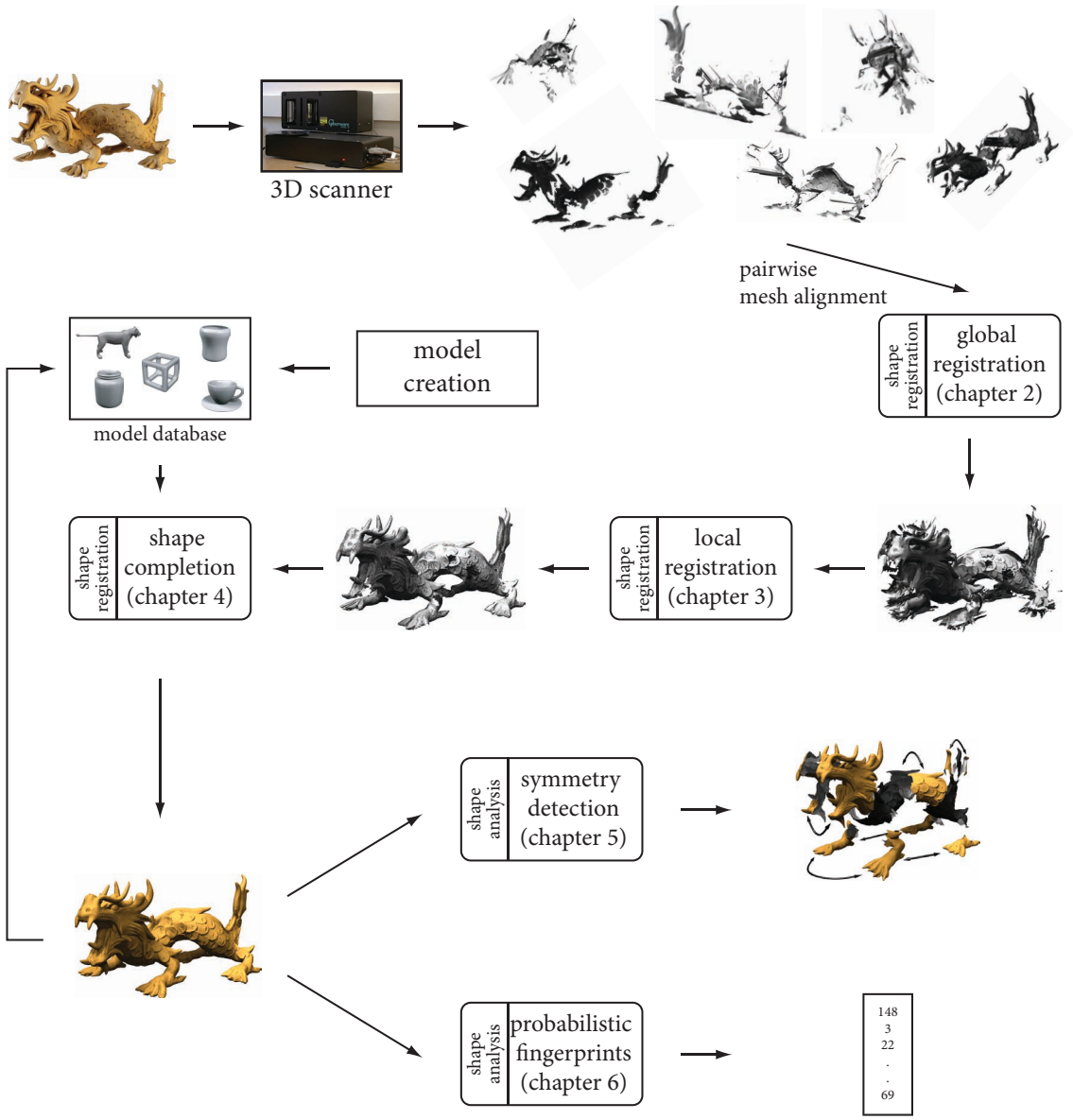


Figure 1.1: *Thesis Outline.* Multiple partial scans of an object is acquired using a laser scanner. The partial scans, in arbitrary starting poses, are pairwise globally aligned and further processed using local refinement. Scans with missing parts are consolidated and repaired using geometric priors in the form of a shape database. Shape analysis generate higher level understanding in the form of symmetry relations and compact probabilistic fingerprints.

models have no linear ordering, lack a canonical parameterization, and occur in arbitrary initial poses.

Different shape similarity measures have been proposed and investigated by researchers in computer vision, computer graphics, and computational geometry. An extensive survey of various similarity measures was presented by Veltkamp and Hagedoorn in [VH00]. Based on the intended application, one can choose a suitable candidate from a large set of similarity measures. However, there are some basic properties shared by all the popular measures. Given two shapes P and Q, let $D(P, Q)$ denote their *distance* with respect to the given poses. Similarity is inversely related to the distance between the models. Since it is easy to switch between distance and similarity, for the rest of the thesis we will take such liberty depending on the context. When D is a metric, it satisfies the following:

- *Identity:* $D(P, P) = 0$.
- *Symmetry:* $D(P, Q) = D(Q, P)$.
- *Triangle Inequality:* $D(P, Q) + D(Q, R) \geq D(P, R)$.

For partial similarity detection only the identity constraint is satisfied. In many cases, having a one-sided distance measure is more desirable. Usually such a asymmetric distance measure is affected by the extents of the parts of the object which do not match, and also by the regions which are common to both objects.

Figure 1.2: *Laser Scanners.* Model Shop Color 3D Scanner and high-resolution Desktop 3D Model 15 in action.

To define a notion of distance invariant to initial model pose, we use $\inf_\alpha D(P, \alpha(Q))$ where α denotes a rigid transform. Finally to handle partial matching, when Q is a part of the model P, the distance measure should satisfy $\inf_\alpha D(P, \alpha(Q)) = 0$.

Once we have a pair of shapes along with an associated shape similarity measure, it is natural to seek algorithms for the following tasks:

- Find the rigid transform that best aligns the two objects.
- Decide if it is possible to align the two shapes such that the residual error is less than a given threshold.
- Identify regions of the objects which are present in both the objects. Further, establish correspondences between the common parts.

Using such basic primitives we can solve a variety of problems which, at a cursory glance, look quite different. In this thesis, we develop algorithms for shape alignment and registration, object improvement and completion, shape analysis, and compact shape signature.

Figure 1.1 shows our contributions to the various stages of the geometry processing pipeline — for shape acquisition and for shape analysis. Following are brief descriptions of the problems addressed in this thesis.

3D scanners are popular devices for shape acquisition. Among the various acquisition techniques, laser scanners are the most widely used. Figure 1.2 shows a few laser scanners in action. In each phase of a multi-phase acquisition process, a scanner looks at an object from some viewpoint and captures the observed geometric information. Multiple such scans are then combined to create a complete model of the object – this is known as the *registration* or the *alignment* problem. Let us formalize this problem a little bit, starting with some notations.

Assume that we have a notion of distance between a pair of objects in some given relative orientation. Then the registration problem can be thought of searching over all possible relative orientations of the two shapes to find the orientation that minimizes the distance between them. Thus given two shapes, the model P and the data Q, our goal is to find the best rigid transform α that minimizes $D(P, \alpha(Q))$. One can imagine computing the distance $D(P, \alpha(Q))$ for all possible rigid transforms α – let us call this the *error landscape*. Even for

simple shapes, the error landscape often has multiple local minima (see Figure 1.3). Clearly any naive optimization on this landscape for computing the global minima can easily get stuck at local minima.

An important question at this stage is how to estimate an aligning transform close to the global minimum. In some cases, we can get such a transform using information about the intermediate motion of the scanner. However, this requires a reasonable calibration between the scanner components. In situations when we do not have such a reliable starting guess, we have to solve the registration problem globally without any assumption on the initial poses. The problem becomes easy when the data is a copy of the model. This special case is commonly referred to as the *whole-to-whole* matching problem. In such a scenario, we can solve for the aligning translation by simply aligning the centroids of the corresponding objects. Subsequently, we extract the rotational component as the transform that brings the respective principal component axes of the two objects into alignment (see Figure 1.4).

However, in most cases scans agree only in parts – commonly referred to as the *partial matching* problem. Simple principal component based alignment methods are not useful for partial matching (Figure 1.4). In fact any other global descriptor based approach fails. The reason is that global properties of two objects sharing a common part can be quite different.

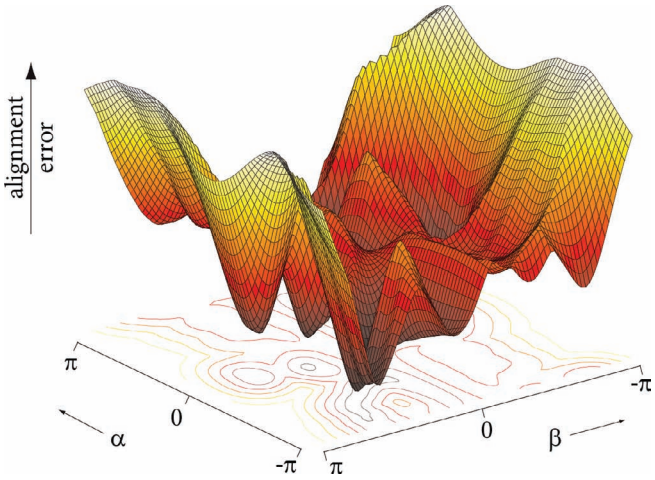
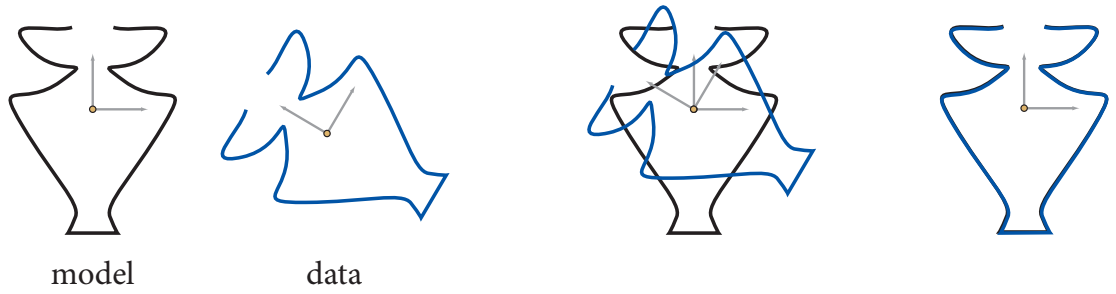
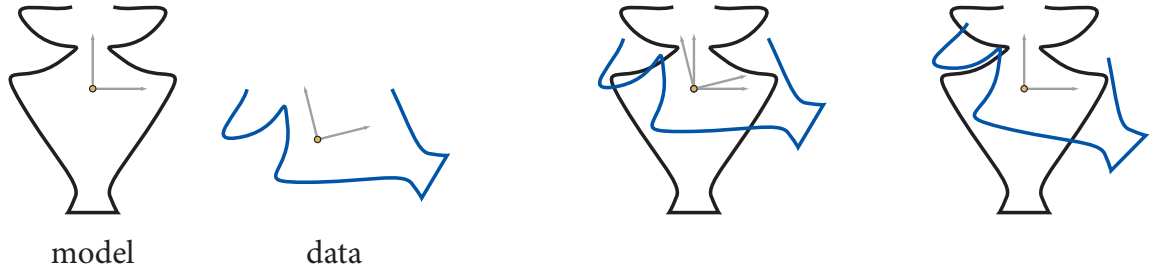


Figure 1.3: *Error landscape.* Plot of error landscape obtained by computing distance between a model and a data surfaces. The position of the model is kept fixed, while we rotate the data about x and the y-axes indicated by angles α and β, respectively. The global minima is at the origin. In this example, the model and data are both the Stanford Bunny.



(a) Whole-to-whole matching



(b) Partial matching

Figure 1.4: *Alignment using principal component axes.* Translation vector is computed by aligning the centroids of the objects. Rotational part is computed by aligning the respective principal component axes of the shapes. While this approach works for whole-to-whole matching, it fails in case of partial matching where the region of overlap is not known.

So we have to compute the aligning transform using a method which does not depend on global characteristics of the shape but rather on the geometry of the common part. The problem becomes challenging as the common parts are not known in advance.

Luckily the low dimensionality of the space of rigid transforms comes to our rescue. Theoretically, if we can correctly establish correspondence between just 3 points in a non-degenerate configuration across the model and data shapes, then we can uniquely solve for the aligning rigid transform. In practice due to robustness and stability issues, we establish correspondence between a few more than 3 points across the model and the data scans.

To this end, in chapter 2, our algorithm computes for each surface point a descriptor based on local geometry that is robust to noise. A small number of feature points are automatically selected from the data shape according to the uniqueness of the descriptor value at the point. For each feature point on the data, we use the descriptor values of the model

to find potential corresponding points in the data shape. We then develop a fast branch and bound algorithm based on distance matrix comparisons to select the optimal correspondence set. Once we have a set of corresponding point pairs we then use the information to bring the two shapes into a coarse alignment.

Now we have the model and the data in rough initial alignment. Since we are close to the global minima in the error landscape, a local optimization based approach refines the solution towards a better alignment. In chapter 3, we formulate the problem of aligning two scans as a minimization of the squared distance between the underlying surfaces. Local quadratic approximants of the squared distance function are used to develop a linear system whose solution gives the best aligning rigid transform for the given pair of point clouds. The rigid transform is applied and the linear system corresponding to the new orientation is build. This process is iterated until it converges. The point-to-point and the point-to-plane Iterated Closest Point (ICP) algorithms can be treated as special cases in this framework. We analyze the convergence behavior of our algorithm and of point-to-point and point-to-plane ICP under our proposed framework, and derive bounds on their rate of convergence.

Now given several scans, we apply our global and local alignment technique for scan pairs to solve the multiple scan alignment problem using a technique proposed by Pulli [Pul99]. After stitching together many scans, we get a representation of the whole object. Unfortunately in many cases we find regions with missing parts in the scanned result. At this stage, we can either acquire more scans, or just use prior knowledge to plausibly fill in regions of missing parts.

In chapter 4, we present a novel approach for obtaining a complete and consistent 3D model representation from incomplete surface scans, using a database of 3D shapes to provide geometric priors for regions of missing data. Our method retrieves suitable context models from the database, warps the retrieved models to conform with the input data, and consistently blends the warped models to obtain the final consolidated 3D shape. We define a shape matching penalty function and corresponding optimization scheme for computing the non-rigid alignment of the context models with the input data. This allows a quantitative evaluation and comparison of the quality of the shape extrapolation provided by each model. Our algorithm explicitly accommodates uncertain data and can thus be applied directly to raw scanner output. The information gained during the shape completion process is utilized

for future scans, thus continuously simplifying the creation of complex 3D models.

Although 3D geometry can be captured using acquisition techniques as described, there are applications where one wishes to create geometrical models for which physical replicas does not exist. In such a scenario, one can model geometry starting from scratch using a wide variety of commercially or publicly available tools. Another possibility is using such modeling tools to edit and deform existing models to create novel shapes. In the end we end up with shapes of varying size and details. Large collection of such acquired or created shapes are easily available in several commercial and public repositories. We are interested in gaining some high level understanding of these shapes. Towards this end we want to estimate intra and inter object similarity. Detecting shape similarities is a key component of many digital geometry processing algorithms such as shape clustering, automatic model alignment, and model retrieval. Similarity measures are useful in answering questions like: Are there repeated parts in the object? Is there partial symmetry in the object? Are multiple objects simply articulated poses of the same shape? How to create a rough embedding of models in a shape space?

In the second part of this thesis, we show how many such questions are efficiently answered using our understanding of shape similarity and techniques for shape alignment. For example, detecting symmetries of an object is essentially solving the partial shape alignment problem between two copies of the same object while ignoring the identity transform. Not surprisingly, many of our techniques carry over from the first part of the thesis.

In chapter 5, we present a new algorithm that processes geometric models and efficiently discovers and extracts a compact representation of their Euclidean symmetries. These symmetries can be partial, approximate, or both. Our method is based on matching simple local shape signatures in pairs and using these matches to accumulate evidence for symmetries in an appropriate transformation space. A clustering stage extracts potential significant symmetries of the object, followed by a verification step. Based on a statistical sampling analysis, we provide theoretical guarantees on the success rate of our algorithm. The extracted symmetry graph representation captures important high-level information about the structure of a geometric model which in turn enables a large set of further processing operations, including shape compression, segmentation, consistent editing, symmetrization, indexing for retrieval, etc.

As the final contribution of this thesis, in chapter 6, we probabilistically determine shape similarity in almost constant time, independent of the resolution of the data. While the driving motivation behind this work is efficient shape database search, our tool is useful for a variety of other geometry processing purposes. Our algorithm is simple, easily parallelizable, and works on streaming data, making it a suitable candidate for massive shape database processing.

Our framework is based on local shape signatures and is designed to allow for quick pruning of dissimilar shapes, while guaranteeing not to miss any shape with significant similarities to the query model in shape database retrieval applications. Since directly evaluating 3D similarity for large collections of signatures on shapes is expensive and impractical, we propose a suitable but compact approximation based on *probabilistic fingerprints* which are computed from the shape signatures using Rabin's hashing scheme and a small set of random permutations. Our approach does not explicitly align the models to compute their similarity. We provide a probabilistic analysis that shows that while the preprocessing time depends on the complexity of the model, the fingerprint size and hence the query time depends only on the desired confidence in our estimated similarity. Our method is robust to noise, invariant to rigid transforms, handles articulated deformations, and effectively detects partial matches. In addition, it provides important hints about correspondences across shapes which can then significantly benefit other algorithms that explicitly align the models.

1.1 Contributions

The goal of this thesis is to understand and evaluate shape similarity for a variety of applications. The proposed methods are not restricted to any particular shape representation and solve a wide range of geometry processing problems including registration of scans, shape completion, symmetry detection of shapes, shape database classification, and feature point identification.

The main contributions of this thesis are (see Figure 1.1):

- A framework for global and local (partial) alignment of multiple shapes specified in arbitrary initial poses.

- Scan completion of partial 3D scans using geometric priors from existing databases.
- Partial and approximate symmetry detection for 3D geometry.
- Probabilistic fingerprints for shapes as a reliable and compact shape descriptor with applications to a wide variety of geometry processing problems.

Part I

Shape Registration

2

Global Registration

A poor relation is the most irrelevant thing in nature.

— *Charles Lamb*

The problem of global registration is a fundamental problem in shape acquisition and shape modeling. In this chapter we present an algorithm for the automatic rigid alignment of two 3D shapes (data and model), without any assumptions about their initial positions. Registration plays an important role in 3D model acquisition, object recognition, and geometry processing. The algorithm first computes, for each shape, a set of robust feature points based on local geometry. For each feature point in the model, we find a small set of candidate points in the data using a range search in the feature space. Now to align the data and the model shapes, we need to solve a combinatorial problem. We use a branch and bound approach to search for the correspondences between the feature points across the data and model shapes. Since a rigid transform preserves inter-point distance, we can evaluate correspondence assignments by comparing distance matrices computed from the data and model, respectively. Further, using a branch and bound approach involving distance matrices we can evaluate all the possible correspondence assignment without having to explicitly enumerate each of them. Thus we avoid explicitly computing the aligning transforms while solving the correspondence problem. Once we have the correspondence, we use it to get a good initial rigid alignment for the models.

It is part of most 3D shape acquisition pipelines, where self-occlusions and scanner limitations usually require the acquisition of multiple partial scans. To build a complete model, the partial scans need to be brought into a common coordinate system (Figure 2.1), which is usually done by pairwise registration. This problem is particularly difficult when no information is available about the initial position of the model and data shapes, the inputs contain noise, and the shapes overlap only over parts of their extent. Also the regions of overlap are not known in advance. While registration is a common problem in 3D scanning, model alignment is also a popular pre-processing step for various geometry processing applications such as texture transfer, morphing, or watermarking [CWPG04]. Before we describe our approach for robust global registration [GMGP05], let us study some of the existing techniques.

2.1 Related Work

Solutions to the registration problem can be broadly divided into two general classes. One class, known as *voting* methods, makes use of the fact that the rigid transform is low dimensional and exhaustively searches for the small number of parameters needed to specify the optimal transform. Generalized Hough transform [HB94], geometric hashing [WR97], and pose clustering [Sto87] quantize the transformation space into a 6-dimensional table. For each triplet of points in the model shape and each triplet in the data shape, the aligning transformation between the triplets is computed and a vote is recorded in the corresponding cell of the table. The entry with the most votes gives the optimal aligning transform. Another variant of this scheme, the alignment method [HU90], tallies for each transform proposed by two triplets of points how many points of the data are brought by the transform close to a point in the model. The transform which brings the most data points within a threshold of a point in the model is chosen as the optimal aligning transform. Voting methods are guaranteed to find the optimal alignment between the data and model shapes and are independent of the initial pose of the input shapes. However, since these methods tend to be costly, they are usually not used for global registration of scan data.

The second class of approaches to the registration problem tries to solve the underlying

correspondence problem: for each point on the data shape, the goal is to find its corresponding point on the model. Given a set of corresponding point-pairs, a rigid transform can be computed that best positions the two shapes so that the distance between corresponding points is minimized [ELF97]. When the initial positions of the model and data are close, the

Figure 2.1: *Scanning the David.* Cyberware gantry being used to scan the statue of David. The image is courtesy of the Digital Michelangelo Project.

correspondences and the transform are usually found using a variant of the Iterated Closest Point algorithm (ICP) [BM92, RL01, MGPG04]. The algorithm assigns to each point in the data its closest point in the model as a correspondence, computes the aligning transform, and applies it to the data shape. This process is iterated until some convergence criterion is

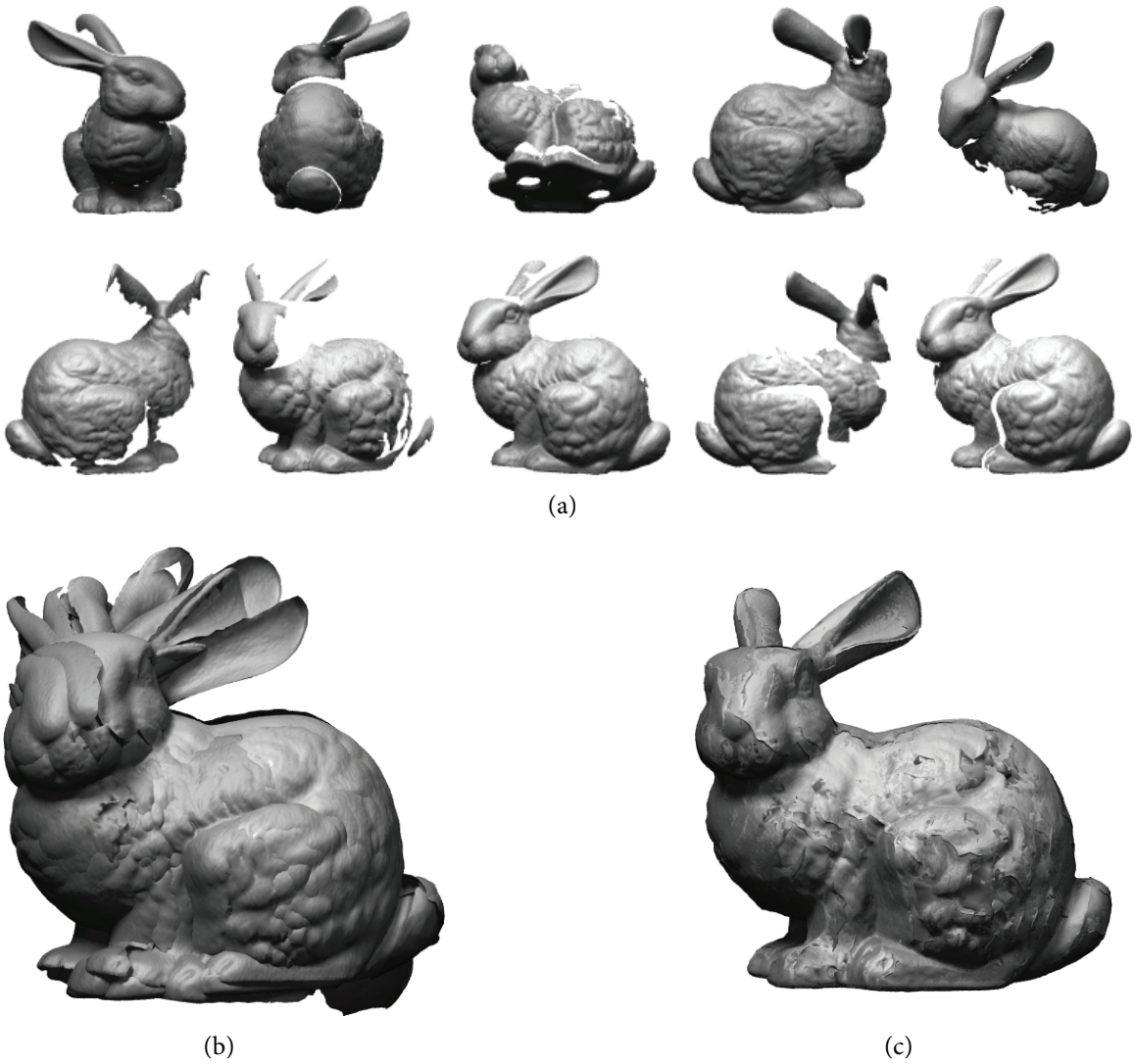


Figure 2.2: *Automatic registration of range data.* Top: 10 input scans (shown here in good position for visualization, the actual input positions are arbitrary). Bottom left: Registration after applying our algorithm to overlapping pairs of scans. Bottom right: Registration after applying ICP and error relaxation to the initial pose produced by our algorithm.

reached. The main limitation of ICP and its variants is that, as a local optimization method, it is not guaranteed to find the globally optimal alignment, and therefore is only effective when the initial position of the input shapes is close to the correct alignment [MGPG04]. In shape acquisition systems, the input scans are usually positioned manually, and then registered using ICP. This approach was used in the Digital Michelangelo Project [LPC+00].

Both the voting schemes and the correspondence search can be improved by using geometric descriptors. A geometric descriptor is a quantity computed for each point of the model and the data, based on the shape of the local neighborhood around the point. Points whose descriptors are similar potentially correspond. High-dimensional, or rich, descriptors such as spin images [JH99] and shape contexts [BMP02] provide a fairly detailed description of the shape around each point in transformation-independent manner. The advantage of rich descriptors is that given a point in the data shape, it is likely that only a few points in the model shape will have a similar descriptor, and the point with the best-matching descriptor is likely to be the correct corresponding point. Incorrect correspondences are few and can be removed using outlier detection methods [FB81], which means that rich descriptors can be used to directly solve the correspondence problem. Huber [HH03] uses spin images computed from subsampled input data for automatic global registration of range data. Rich descriptors are particularly popular for object recognition and shape retrieval, where the computation of descriptors can be amortized over large number of comparison queries [BMP02, FMK+03].

Low-dimensional descriptors, on the other hand, usually compute only a few values per point. Examples of such descriptors include curvature and various curvature-based quantities such as shape index [Koe90] and curvedness [KD92]. Low-dimensional descriptors are typically much easier to compute, store, and compare than high-dimensional rich descriptors. However, for a given point in the data shape, there may be many points in the model shape with the same descriptor value. Therefore, low-dimensional descriptors are usually used in conjunction with a voting scheme [BS97] to reduce the size of the search space or with an iterative alignment scheme to improve the funnel of convergence (set of starting positions which result in correct alignment) [SLW02, GLB99].

Since the inputs to the registration algorithm are usually large, a common speedup technique is to pick a set of feature points on the model and data based on the computed descriptor values [MKY01]. The registration is then performed only with respect to the feature points, which results in significant reduction of the size of the search space. Feature extraction methods, however, can suffer from the problem of picking inconsistent points on the model and data, since the two shapes are processed separately. The resulting set of feature points, therefore, may not have a good alignment. Because of possible errors in feature selection, correspondence assignment techniques based on geometric descriptors usually build large correspondence sets to increase robustness to incorrect features and pairings. Therefore, these methods, unlike the voting schemes, do not make use of the low-dimensionality of the aligning rigid transform.

Contributions

In this work, we develop a new global registration algorithm, based on robust feature identification and correspondence search using geometric descriptors. The main contributions of our method are as follows:

- Our algorithm makes use of the fact that the aligning transform is low-dimensional to robustly find a small set of matching point-pairs that specify the optimal alignment.

- We focus on identifying a small number of feature points on the data shape, and then searching the entire model shape for correspondences. This approach avoids the problem of selecting incompatible features that is common in other feature-based registration methods.

- We use a novel shape descriptor based on an integral operator on the underlying shape, for identifying features in the data and selecting potential correspondence points in the model. Unlike the more common curvature based descriptors that use differential operators, integral descriptor does not suffer from instability issues. Our feature selection algorithm picks points on the data shape which have uncommon descriptor values across a range of scales.

- For each feature point, the correspondence search algorithm examines the entire model shape to identify the optimal corresponding point. The search is made efficient by using a measure of quality of correspondences based on computing only intrinsic quantities of the model and data shapes. This allows us to avoid computing an aligning transformation, and results in an efficient branch-and-bound algorithm. Additionally, we use the rigid transform constraints for efficient pruning of the search space.

- Since our algorithm only uses descriptor values, which are invariant under rigid transforms, and intrinsic geometric properties of the input shapes, we are able to align the model and data shapes without any assumptions about their initial position. The pose produced by our algorithm is further refined by ICP, producing an automatic global registration pipeline.

2.2 Integral Volume Descriptor

Let $\mathbf{P}$ be the input shape, consisting of N points $\mathbf{p}_1 \dots \mathbf{p}_N$. The input can be specified as a mesh or as a point cloud. An m-dimensional geometric descriptor is a function that assigns to each point $\mathbf{p} \in \mathbf{P}$ a vector $f(\mathbf{p}) \in \mathbb{R}^m$. To be useful in registration algorithms, a descriptor should be invariant under rigid transformations, robust to noise, and based on local geometry around $\mathbf{p}$ (since the input shapes may be only partially overlapping). We will restrict our attention to low-dimensional descriptors, since they are cheaper to compute, store, and compare than rich descriptors.

Most of the common low-dimensional shape descriptors are based on differential quantities of the shape, since they are invariant under rigid transformations. The main limitation of differential descriptors, which has made them unpopular in registration algorithms, is that any noise present in the input gets amplified when derivatives are computed. As a result, algorithms that rely on differential descriptors need to perform careful smoothing of both data and model shapes.

An alternative approach, that has yielded promising results in object recognition and feature classification, is to use local shape invariants that are based on integration instead of differentiation [MHYS04, CRT04]. Integral descriptors retain the desirable properties of differential invariants such as locality and invariance under rigid transformations, but

are more robust to noise. Manay et al. [MHYS04] showed that integral invariants have descriptive power comparable to curvature-based descriptors, but are more effective in 2D object recognition in the presence of noise. In this section, we extend the integral invariants of [MHYS04] to 3D.

2.2.1 Definition of Descriptor

We develop a 3D integral invariant, called the *integral volume descriptor*. This invariant is defined at each vertex $\mathbf{p}$ of the input shape as follows,

$$V_r(\mathbf{p}) = \int_{B_r(\mathbf{p}) \cap \overline{S}} dx. \tag{2.1}$$

Here the integration kernel $B_r(\mathbf{p})$ is a ball of radius r centered at the point $\mathbf{p}$, and $\overline{S}$ is the interior of the surface represented by $\mathbf{P}$. The quantity $V_r(\mathbf{p})$ is the volume of the intersection of the ball $B_r(\mathbf{p})$ with the interior of the object defined by the input mesh. The invariant is illustrated in 2D in Figure 2.3(a).

The reader is referred to [Gelo6] for a detailed description of properties of integral descriptors and how to evaluate them.

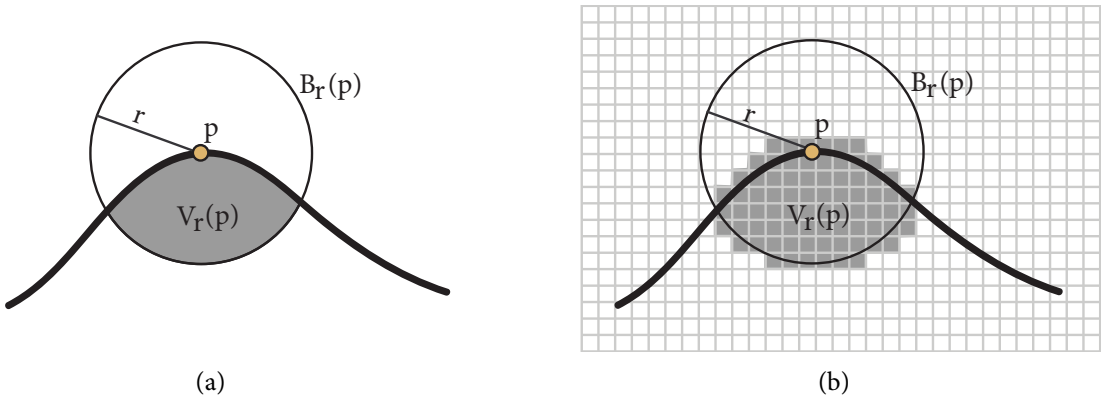


Figure 2.3: *Illustration of the volume integral descriptor in 2D.* (a) We take the intersection of a ball of radius r centered at point $\mathbf{p}$ with the interior of the surface. (b) Discretization of the volume descriptor as computed by our algorithm. The cell size of the grid is ρ.

2.3 Feature Point Selection

Our registration algorithm is based on finding correspondences in the model **Q** for a small number of feature points picked from the data shape **P**. The features are selected in a way that makes the search for correspondences particularly simple. The key property of our feature selection algorithm is that feature points should come from regions with relatively rare descriptor values. Since the data and model shapes are similar over the matching region, and we use descriptor values to select potential corresponding points in the model for each feature point in the data, points with relatively rare descriptor values are likely to have only a few potential correspondence points. Thus, our feature selection algorithm specifically picks points such that the resulting search for correspondences will be fast. Additionally, we do not need to select many points as features, since a rigid transform can be specified using only a small number of points. Selecting a small number of feature points, such that each will have only a small number of potential correspondences results in a tractable correspondence search problem.

We first briefly describe an algorithm for a general descriptor, and then use a scale-space representation of the volume descriptor together with persistence [CZCG04] across multiple scales to robustly select feature points.

2.3.1 Basic Algorithm

Let f be the geometric descriptor which associates with each point $\mathbf{p}_i$ a value $f(\mathbf{p}_i)$. The descriptor can be of any dimension, in this section we assume that the descriptors are one-dimensional $f(\mathbf{p}_i) \in \mathbb{R}$. A point $\mathbf{p}$ is defined to be a feature if its descriptor value is rare among all descriptors computed for the data shape **P**. The feature point selection proceeds as follows:

1. Compute a histogram of descriptor values, $f(\mathbf{p}_i)$ for all points in **P**. The number of bins b in the histogram is computed using Scott's rule, $b = 3.49\sigma_f N^{-\frac{1}{3}}$, where σ_f is the standard deviation of the N descriptor values [Sco79].

2. To select feature points, we identify the k least populated bins such that the total number of points in these bins is smaller than some maximum threshold s. The points that

belong to these bins are the potential features. Intuitively, features are those points which are dissimilar from the rest of the shape, which is captured by the low occurrence of their descriptor values. The parameter s controls the tradeoff between accuracy of the transform (more correspondences) and running time of the algorithm. In our implementation, we set $s = 0.01N$.

3. Since nearby points are likely to belong to the same feature, we want to prevent the algorithm from picking points that are too close to each other. We also want the points to cover the whole shape since in case of partial matching we do not know a priori which part of the data shape will overlap with the model. When a point $\mathbf{p}_i$ is picked, we mark all points that fall into a ball of radius R_e around $\mathbf{p}_i$ as unavailable for selection. Enforcing the minimal separation distance between the feature points also results is more stable configurations in the correspondence search stage of the algorithm (Section 2.5).

Notice that this process is not invariant to the order of points in $\mathbf{P}$. This means that it cannot be used to pick canonical points on the data *and* model shapes. As mentioned before, we do not rely on feature points being canonical, since we will search the entire model shape for correspondences, instead of trying to match up canonical points. This means, as long as a feature point lies in the overlap region between the model and data, it will have a correspondence assigned to it by the matching stage of our registration algorithm.

The above algorithm works with any kind of descriptor which can be represented as a vector in $\mathbb{R}^m$. Since we are picking as features those points of $\mathbf{P}$ that have uncommon descriptor values, we need a descriptor that is robust to noise, making integral descriptors particularly well suited for this kind of approach. Figure 2.4(b) shows the feature points selected on the dragon model corrupted with zero-mean Gaussian noise.

Although volume descriptors are robust to noise, they can still label spurious points as features due to presence of outliers. For this reason, to make our algorithm more robust, we compute descriptors at varying scales by changing the radius r of the neighborhood ball. For more discussion on scale-space selection please refer to [Gel06].

The result of the feature selection stage is a set of feature points $\mathbf{P}'$ selected from the data shape. For each feature, we are given the coordinates of the point, scale-space representation

of the volume descriptor, and the range of radii for which this point was classified as a feature. We now develop an algorithm that finds, for each feature point on the data, a corresponding point on the model.

2.4 Distance Metrics

Given a set of n feature points $\mathbf{P}'$ selected from data shape $\mathbf{P}$, the goal of the correspondence search algorithm is to find, for each $\mathbf{p}_i \in \mathbf{P}'$, a point $\mathbf{q}_i \in \mathbf{Q}$, that is the best match to $\mathbf{p}_i$. Let $\mathbf{P}'$ and $\mathbf{Q}'$ be two sets of points with correspondences given as $(\mathbf{p}_i, \mathbf{q}_i)$. We present two ways of evaluating the quality of the correspondence based on the coordinates of the points $(\mathbf{p}_i, \mathbf{q}_i)$.

The standard measure of distance between two point sets with known correspondences is the *coordinate root mean squared error*, or cRMS. This error measures how close each point $\mathbf{p}_i$ comes to each corresponding point $\mathbf{q}_i$ after an optimal rigid aligning transform is computed for the entire set of corresponding points.

$$\text{cRMS}^2(\mathbf{P}', \mathbf{Q}') = \min_{\mathbf{R},\mathbf{t}} \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i||^2, \tag{2.2}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector. The optimal aligning rigid transformation needs to be computed before the error can be evaluated, which can be done using one of the methods described in [ELF97]. The cRMS distance metric is the most frequently used measure of residual error in registration algorithms.

An alternative metric of distance between two point sets with known correspondences is the *distance root mean squared error*, or dRMS. This metric is commonly used in computational molecular biology for comparing the similarity of two protein shapes [Koe01]. The dRMS error is computed by comparing all internal pairwise distances of the two point sets, and is defined as

$$\text{dRMS}^2(\mathbf{P}', \mathbf{Q}') = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (||\mathbf{p}_i - \mathbf{p}_j|| - ||\mathbf{q}_i - \mathbf{q}_j||)^2. \tag{2.3}$$

The triangle inequality and the property that the optimal transform aligns the centroids

of $\mathbf{P}'$ and $\mathbf{Q}'$ allows us to upper bound dRMS using cRMS as follows,

$$\text{dRMS}(\mathbf{P}', \mathbf{Q}') \leq \sqrt{2}\ \ \text{cRMS}(\mathbf{P}', \mathbf{Q}'). \tag{2.4}$$

To compute a lower bound, we need to examine both $\mathbf{Q}'$ and its reflection around any arbitrary plane $\overline{\mathbf{Q}'}$ (since dRMS is invariant under reflection, but cRMS is not). The lower bound can be shown to be

$$\frac{1}{k\sqrt{n}} \min(\text{cRMS}(\mathbf{P}', \mathbf{Q}'), \text{cRMS}(\mathbf{P}', \overline{\mathbf{Q}'})) \leq \text{dRMS}(\mathbf{P}', \mathbf{Q}'). \tag{2.5}$$

Here n is the number of corresponding point pairs and k is a constant, depending on ratio of the diameter of the data shape to the feature exclusion radius used in Section 2.3. These bounds mean than when the dRMS of two point sets is small, their cRMS will also be small (when there is no reflection), indicating that the point sets are in good alignment. Therefore, we can use dRMS instead of cRMS to evaluate how well two point sets correspond.

The advantage of dRMS is it does not require computation of the aligning transform before the quality of the correspondence can be evaluated. It is, in fact, only comparing intrinsic properties of the two sets of corresponding points, namely the internal pairwise distances of each pointset, as opposed to comparing the distances between the two point sets. This means that, given the set of feature points $\mathbf{P}'$, its pairwise distance matrix needs to be computed only once, and then compared to pairwise distance matrices of the potential correspondence sets $\mathbf{Q}'$. Additionally, since only intrinsic properties of the point sets are examined in dRMS computation, we will be able to efficiently prune correspondence sets that contain wrong matches without having to compare the entire sets $\mathbf{P}'$ and $\mathbf{Q}'$. This will allow us to develop an efficient branch-and-bound algorithm, as described in the next section.

2.5 Correspondence Search

The algorithm presented on this section was joint designed with Natasha Gelfand. These ideas were born and shaped as a result of our many long discussions. Please refer to her thesis [Gelo6] for additional comments.

2.5.1 Computing Potential Correspondences

Let $\mathbf{P}'$ be the set of n points picked by the feature selection algorithm. For each feature point $\mathbf{p}_i$ we also have the scale-space representation of the volume descriptor $(V_{r_1}(\mathbf{p}_i), \ldots, V_{r_k}(\mathbf{p}_i))$, and the values r_a^i, r_b^i, which are the minimum and maximum radii of the kernel of the volume descriptor for which $\mathbf{p}_i$ is a persistent feature. We now use the descriptor values to select potential corresponding points in the model shape $\mathbf{Q}$ for each feature point.

For our descriptor-based matching algorithm, we first compute the same scale-space representation of the volume descriptor on the model shape. That is, we compute volume descriptors for radii $r_1, r_2, \ldots, r_k$ for each point on the model shape. Let $\mathbf{p}$ be a feature point selected from the data shape, and let r_b be the largest feature radius. We perform a range query [AMN+98] in the model, and select all points $\mathbf{q}$ such that $|V_{r_b}(\mathbf{p}) - V_{r_b}(\mathbf{q})| < \epsilon$. We can also perform the range query for any radius between r_a and r_b of $\mathbf{p}$, however we prefer the largest possible radius since it gives the most stable descriptor. The variation of the descriptor values ϵ can be related to the grid size ρ and the radius of the volume descriptor r as $\epsilon \approx 0.75\rho/r$. This accounts for the variation in the value of the volume descriptor due to discretization using the voxel grid. We pick ρ to be large enough to account for noise in the data.

The range query results in the set of points $C_{initial}(\mathbf{p})$ whose volume descriptor for the given radius is similar to the descriptor value at $\mathbf{p}$. Similar to the approach in the feature selection algorithm, we want to pick a set of points that represent distinct areas of the model. We cluster all points in $C_{initial}(\mathbf{p})$ into clusters of radius R_c and pick from each cluster the point $\mathbf{q}$ that minimizes $|V_{r_b}(\mathbf{p}) - V_{r_b}(\mathbf{q})|$. This gives the final set of correspondences for $\mathbf{p}$, $C(\mathbf{p})$. We repeat this procedure for each point in the feature set.

Using a range search instead of exact match of the descriptor values makes it more likely that the correct correspondence of $\mathbf{p}$ is included in the set $C_{initial}(\mathbf{p})$ (under a reasonable noise model). After clustering, we are guaranteed that the correct correspondence is within R_c of a point in $C(\mathbf{p})$. It follows that the correct set of corresponding points of $\mathbf{P}'$ has cRMS at most R_c, and dRMS is bounded by $\sqrt{2}R_c$. The value of R_c, therefore, is a knob that controls the quality of the resulting registration.

2.5.2 Matching Algorithm

Even though we have a comparatively small number of feature points, and each feature point has a small number of potential correspondences, exhaustive exploration of the space of all correspondences can still be prohibitively expensive. The key observation that will allow us to develop a fast algorithm is that we can use the rigidity constraints of the aligning transform to efficiently eliminate a large set of potential correspondences.

Given a set of feature points $\mathbf{P}' = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ selected from the data shape, and a set of potential correspondences for each point in the model shape $(C(\mathbf{p}_1), \ldots, C(\mathbf{p}_n))$, we want to select a set of points $\mathbf{Q}'$ such that $\mathbf{q}_i \in C(\mathbf{p}_i)$ and the error metric of Equation 2.3 is minimized over all sets of such correspondences. Since we will only be considering points in $\mathbf{Q}$ that belong to some potential correspondence set, we will change the notation slightly in this section to simplify the explanation of the algorithm. Given a feature point $\mathbf{p}_i$, we will designate the j-th member of the potential correspondence set $C(\mathbf{p}_i)$ as $\mathbf{q}_i^j$.

Consider a pair of feature points $(\mathbf{p}_i, \mathbf{p}_j)$. According to their descriptor values, any pair of points $(\mathbf{q}_i^k, \mathbf{q}_j^l)$ can be used as corresponding points. Rigid transform constraints tell us that the distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ needs to be the same as the distance between their correspondences in the model. Since we are using correspondences that are only approximate within the clustering radius R_c, the correspondence pairs need to satisfy the relationship

$$\left| ||\mathbf{p}_i - \mathbf{p}_j|| - ||\mathbf{q}_i^k - \mathbf{q}_j^l|| \right| < 2R_c. \tag{2.6}$$

We apply this thresholding rule in a branch-and-bound algorithm for finding the best set of correspondences. Let $\mathbf{Q}' = (\mathbf{q}_1^*, \ldots, \mathbf{q}_n^*)$ be the current best set of correspondences for the set of feature points $\mathbf{P}'$, and let $E_{min} = \text{dRMS}(\mathbf{P}', \mathbf{Q}')$ be the error of the current best correspondence set. We initialize the set of correspondences using a greedy algorithm described in Section 2.5.3. The branch-and-bound correspondence search proceeds as follows:

1. Assume corresponding points have been assigned for the first $k-1$ feature points, which gives us a partial correspondence set $(\mathbf{q}_1^{c_1}, \ldots, \mathbf{q}_{k-1}^{c_{k-1}})$. We are looking for the correspondence for the k-th feature point.

2. **Threshold:** For each potential correspondence of $\mathbf{p}_k$, apply the thresholding test of

Equation 2.6 with respect to all previously selected points. That is we verify that Equation 2.6 holds for all pairs $(\mathbf{p}_i, \mathbf{p}_k), (\mathbf{q}_i^{c_i}, \mathbf{q}_k^j)$ for $i = 1, \ldots, k-1$. If one of the tests fails, we can prune the branch that includes the correspondence pair $(\mathbf{p}_k, \mathbf{q}_k^j)$.

3. **Prune:** For each $\mathbf{q}_k^j$ that passes the thresholding test, form the partial correspondence $(\mathbf{q}_1^{c_1}, \ldots, \mathbf{q}_{k-1}^{c_{k-1}}, \mathbf{q}_k^j)$ and evaluate the dRMS error of this partial correspondence. If the partial error is greater than the error of the current best estimate E_{min}, discard $\mathbf{q}_k^j$ as a correspondence.

4. **Branch:** For each of the remaining $\mathbf{q}_k^j$ that pass both the thresholding and the pruning tests, assign $c_k = j$, and recursively invoke Step 1. Once all correspondences for $\mathbf{p}_k$ have been examined, we backtrack and assign the next correspondence to the previous point $\mathbf{p}_{k-1}$.

5. **Bound:** If all feature points have been assigned correspondences, compute the error of the match E. If the dRMS error is less than E_{min}, we potentially have a better correspondence set, and a new bound, unless the current assignment is actually a reflection. We can rule out reflection by making sure the cRMS error of the current correspondence set is also small. If the cRMS error check passes, we assign $E_{min} = E$ and $\mathbf{Q}' = (\mathbf{q}_1^{c_1}, \ldots, \mathbf{q}_n^{c_n})$.

The branch-and-bound algorithm is possible because we are using the dRMS error metric, which can be computed for partial correspondences without the need for the optimal aligning transform. The only time when the aligning transform in computed is in the last step, and only if we need to update the bound.

2.5.3 Greedy Bound

The initial correspondence and error bound is established using a hierarchical greedy algorithm. The algorithm first finds the best correspondences for each pair of feature points. Then it combines the pairs to form best corresponding sets of four points, then combines fours into eights and so on.

1. **Form pairs:** For each pair of feature points $(\mathbf{p}_i, \mathbf{p}_j) \in \mathbf{P}'$, choose the best pair of corresponding points $(\mathbf{q}_i^k, \mathbf{q}_j^l)$ in their associated potential correspondence sets. The best

matching pair of correspondences is one that minimizes the distance metric penalty $\left|||\mathbf{p}_i - \mathbf{p}_j|| - ||\mathbf{q}_i^k - \mathbf{q}_j^l||\right|$. This gives us the set E_2 of $O(n^2)$ two-point correspondences. We sort E_2 in order of increasing distance discrepancy.

2. **Combine pairs:** Combine two-point correspondences into four-point correspondences. Given a two-point correspondence $e \in E_2$, find the two-point correspondence in E_2 that does not contain any of the points of e, and that minimizes the dRMS error of the resulting four-point correspondence. Remove from E_2 all correspondences that have the same endpoints as the new four-point correspondence, and continue until the set E_2 becomes empty. Call this set E_4, and again sort it by increasing dRMS error.

3. **Build hierarchy:** We continue merging in this manner, merging pairs of elements of a set E_k to form the set E_{2k}. We typically stop at either E_8 or E_{16}.

4. **Assign the rest of the points:** We pick the correspondence from the resulting set E_k that has the smallest dRMS error. We use this partial (8 or 16 point) correspondence to compute the rigid transform $(\mathbf{R}, \mathbf{t})$ that minimizes the cRMS error (Equation 2.2) and apply it to the entire feature point set $\mathbf{P}'$. For all points in $\mathbf{p}_i \in \mathbf{P}'$ that do not yet have correspondences, we assign the point $\mathbf{q}_i^j \in C(\mathbf{p}_i)$ that is closest to $\mathbf{R}(\mathbf{p}_i) + \mathbf{t}$. We use this as the initial correspondence $(\mathbf{P}', \mathbf{Q}')$ and initialize E_{min} to $\text{dRMS}(\mathbf{P}', \mathbf{Q}')$ in the algorithm described in Section 2.5.2.

This approach is greedy because each step picks the best correspondences to merge together and never backtracks. Therefore is it possible that an incorrect correspondence is found for $\mathbf{P}'$. However, as long as some points are matched to their correct corresponding points in Step 1, the algorithm tends to produce a tight bound that greatly speeds up the basic branch-and-bound algorithm. In practice, this approach often results in a very good guess of the correct alignment, resulting in effective pruning in the branch-and-bound algorithm.

2.5.4 Partial Matching

When the model and data shapes overlap only over part of their extent, not all the feature points picked on the data will have corresponding points in the model. Therefore, we modify our matching algorithm to handle such partial matches.

In addition to performing the search over all correspondences, we also need to find the subset of the feature points that are the same in the model and data. We augment the set of potential correspondences for each point, $C(\mathbf{p}_i)$, with the not present value $\emptyset$. When a point is assigned $\emptyset$ as a correspondence, it does not contribute to the computed dRMS error. We want to maximize the number of feature points that get assigned a valid corresponding point in the model, while still keeping the dRMS error of the correspondence set low.

Suppose we know that k feature points are missing from the model, but do not know which k. We can run our correspondence search algorithm, but prune away any branch that has more than k points assigned the $\emptyset$ correspondence. This will select the best $n - k$ feature points that have the best correspondences. Since we do not know k, we can run the same algorithm for k ranging from 0 to $n - 3$ (since only three points are needed to specify a rigid transform). For robustness, we actually require at least 5 points to have a valid correspondence. We can detect the maximum k since the error will sharply decrease once $n - k$ reaches the correct number of common feature points. Figure 2.5(d) shows the dRMS error vs. the number of matched feature points for the David model.

2.6 Results

2.6.1 Object Registration

We applied our algorithm to a number of registration problems. Although in the examples the model and data shapes are shown in similar positions, the reader should keep in mind that our algorithm does not depend on any assumptions about the initial positions of the input shapes, and the input shapes were given to our algorithm in arbitrary positions. Timing results for the experiments are given in Figure 2.1.

In the first example, we use the algorithm for whole object alignment in the presence of significant noise. We align the dragon model to a copy of itself corrupted by zero-mean Gaussian noise. Figure 2.4 shows the results. Our alignment brings the data (noisy) shape close enough to the model (smooth) shape that applying one iteration of standard ICP with point-point error metric [RL01] brings the shapes into exact alignment.

Figure 2.5 shows the results of applying our algorithm to register partially overlapping range data. We take two raw scans of the David's face, subsample them, and convert to a mesh

representation. We do not perform any other smoothing or surface reconstruction. The scans are given in arbitrary initial positions (scanner coordinates) and brought into close alignment by our algorithm. The pose computed by our algorithm is refined by running three iterations of ICP. Fifteen feature points were picked on the data shape, eight of which were assigned correspondences and used to compute the alignment.

Finally, we use our algorithm to build a complete model out of constituent range scans. Given as input ten range scans of the Stanford bunny taken from different view points, we bring all scans to a common coordinate frame using our algorithm. The rough alignment accumulates errors since we align each scan only to one other, and do not perform any bundle adjustment. However, the scans are now close enough to refine the pairwise matches using ICP, and diffuse the accumulated error over all scans using an error relaxation algorithm proposed by Pulli [Pul99]. This gives us a completely automatic model construction

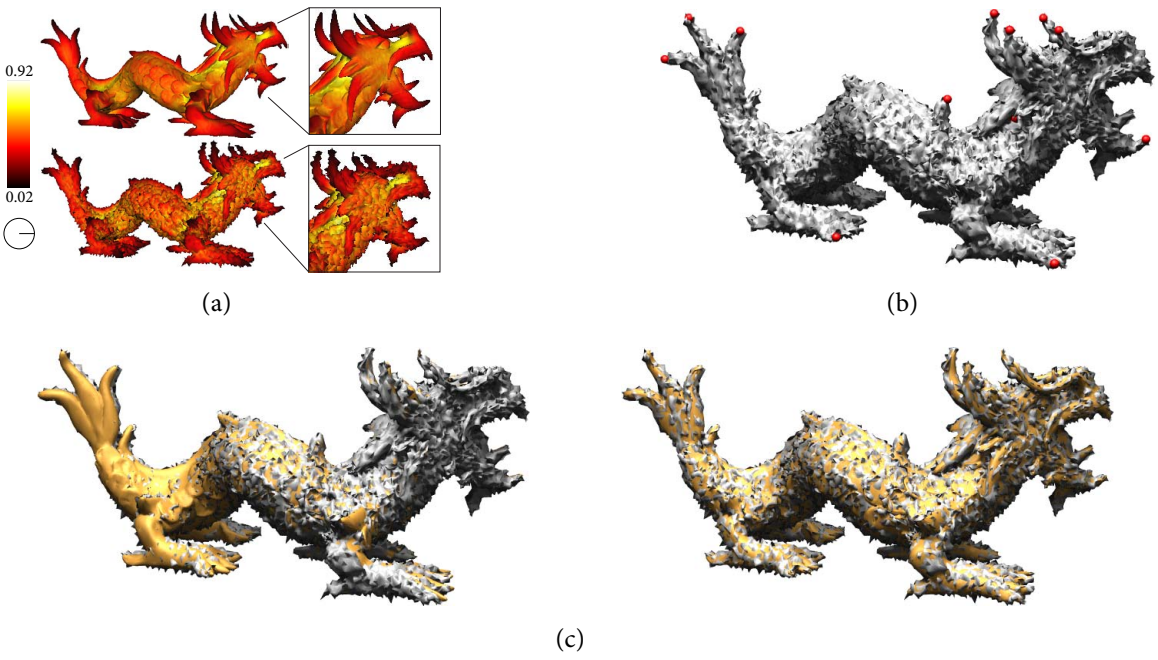


Figure 2.4: *Dragon example.* (a) Input to the matching algorithm: Smooth dragon (the model) and noisy dragon (the data) with descriptor values shown at each point. Even under noise the descriptor values at feature points look similar. (b) Feature points picked on the data shape. (c) Top: Registration after applying our algorithm. Bottom: Registration after refinement by ICP.

	model size	selection time	num features	corr time	num corr
Dragon	29,455	6.3	38	2.2	9
David	68,480	84.5	15	35.7	6
Bunny	35,000	21.8	11	13.9	4
Part	20,002	5.9	13	15.7	8
Hinge	45,311	19.0	30	1.2	12

Table 2.1: *Timing Chart.* Input size, running time (in sec), and number of feature points for the registration experiments. In all cases the model size and data size are similar, so we only give the size of the model. The feature selection time includes descriptor computation for both data and model. We also indicate the number of selected feature points and average number of potential correspondences ($|C(\mathbf{p})|$) for each point.

pipeline. The result is shown in Figure 2.2.

2.6.2 Symmetry Detection

Our registration algorithm can be trivially extended to detect symmetry in objects by matching an object to a copy of itself. Instead of returning the best matching orientation, we return all matches with small error. Since the feature points picked by our algorithm are spaced far apart, the difference between the symmetry configurations and other matches will be large. Figure 2.6 shows the results of detecting symmetries of a mechanical part. Notice that the graph of error in Figure 2.6 shows eight configurations with small error, which corresponds to the eight-way symmetry of the model. Later in Chapter 5, we describe an algorithm for detecting partial and approximate symmetries in 3D geometry.

2.6.3 Articulated Matching

Our global registration algorithm can be used to discover rigid parts in objects that undergo articulated deformation. In this case, $\mathbf{P}$ and $\mathbf{Q}$ are two positions of the object. We want to decompose the shape $\mathbf{P}$ into the minimum set of parts $\mathbf{P}_1 \ldots \mathbf{P}_k$, such that each $\mathbf{P}_i$ can be aligned to a part of $\mathbf{Q}$ using a rigid transform. Here, we present a simple proof of concept implementation.

We perform articulated decomposition by partial matching of $\mathbf{P}$ and $\mathbf{Q}$. This gives the

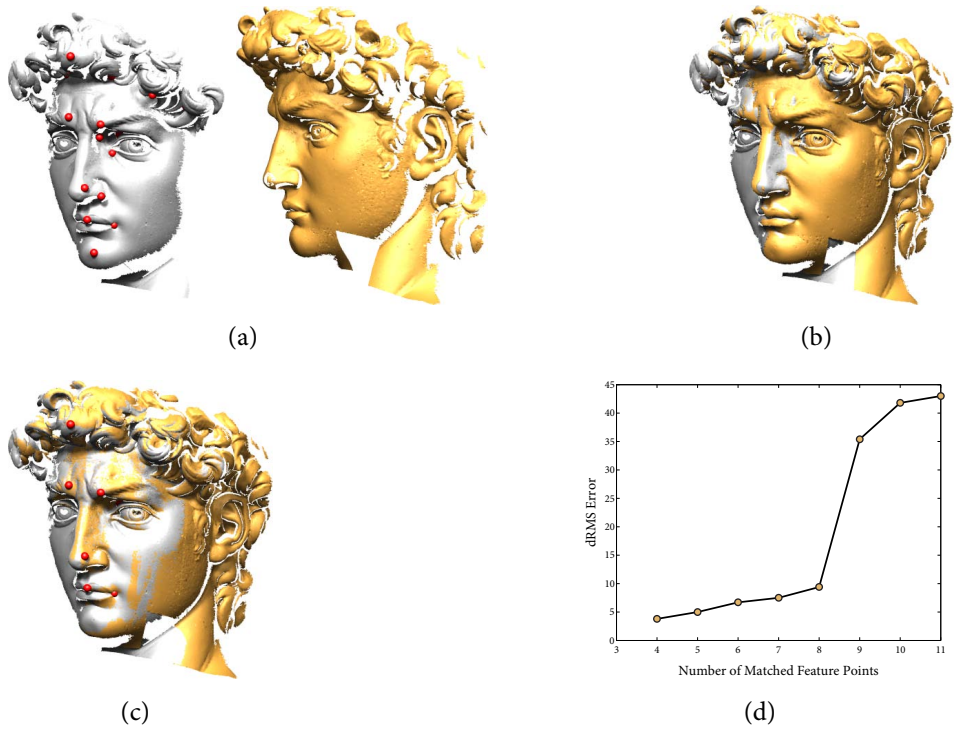
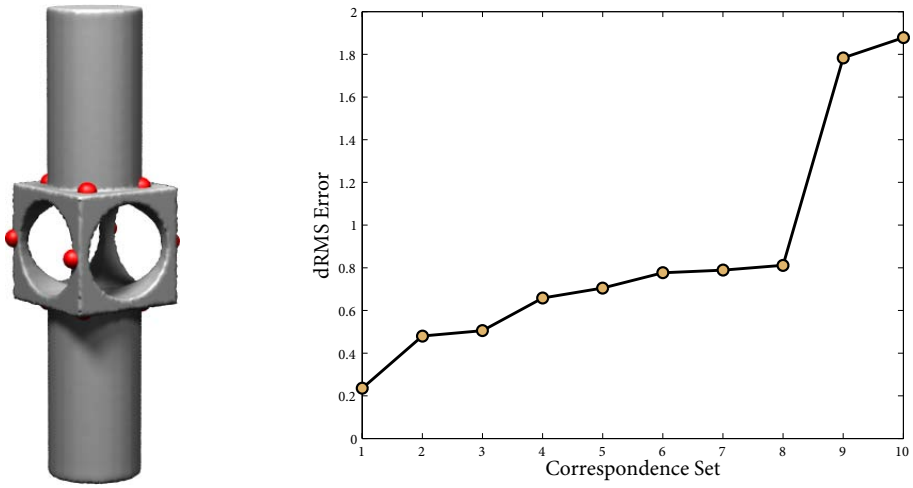


Figure 2.5: *David example.* (a) Two scans of the David's face. Feature points picked on the data shape are shown in red.(b) Registration after applying our algorithm. (c) Registration after refinement by ICP. Points actually used to compute alignment in (b) are shown in red. (d) Graph of dRMS error as the function of the number of matched features. Notice the significant increase in error for more than 8 points, which is the correct number of common features.

transform $\mathbf{R}_1, \mathbf{t}_1$. We apply the transform to the data shape, and classify all points of the data that fall within a threshold of the model as belonging to component $\mathbf{P}_1$. We then separate $\mathbf{P}_1$ and the corresponding $\mathbf{Q}_1$ from the input shapes and repeat the partial matching algorithm with $\mathbf{P} - \mathbf{P}_1$ and $\mathbf{Q} - \mathbf{Q}_1$. We repeat the process until the size of the residual set becomes too small. Figure 2.7 shows the result of segmenting a shape into rigid components using this algorithm.

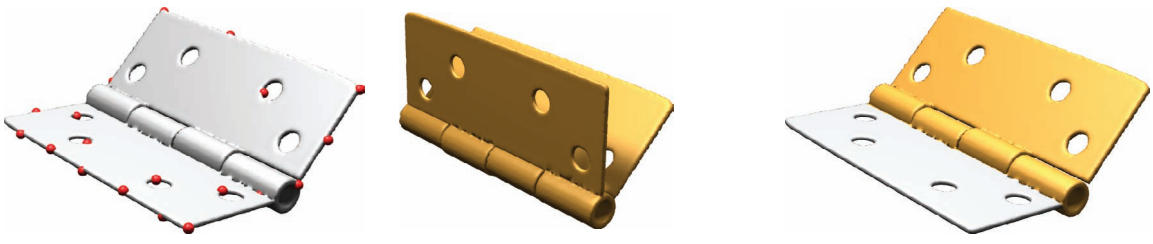
The features picked on the data shape in Figure 2.7 also point one of the advantages of the non-canonical nature of our feature selection and correspondence search. If a linear feature is present in the input, such as the long edge of the hinge model, our feature selection algorithm discretely samples the edge at intervals given by the exclusion radius R_e. If we were



(a) (b)

Figure 2.6: *Symmetry detection using registration.* (a) Feature points picked by our algorithm, when the shape is aligned to a copy of itself. (b) Graph of the error for different correspondence sets. The eight correspondences with small error indicate the eight-way symmetry of the shape.

picking and matching features on both data and model shapes, this discrete sampling could potentially result in two sets of points which do not match each other. However, since we only pick features on one shape, the data, and then search the entire model, we always find a compatible set of points (to within the error given by the clustering radius E_c) with which to align the features.



(a) (b)

Figure 2.7: *Simple articulated matching.* (a) Two input positions of the shape. Feature points picked by our algorithm are shown in red. (b) Using repeated partial matching, the algorithm discovers two rigid components.

2.7 Summary

In this chapter, we presented a global registration algorithm that aligns two 3D shapes without any assumptions about their initial positions. Our algorithm is able to align whole and partially overlapping shapes, and is robust to noisy data. The algorithm works well in the presence of strong point-like features in the input data.

However, if the input shapes do not have strong sharp features in regions of overlap, the correspondence search space examined by the algorithm becomes quite large. As a result of such a combinatorial explosion our algorithm becomes impractical. Later in chapter 6 we present a randomized scheme to efficiently handle such cases.

3

Local Registration

The shortest distance between two points is under construction.

— *Leo Aikman*

In this chapter, we describe the second part of our registration algorithm. Global registration gets the input shapes in rough initial alignment, and now our goal is to further refine their positions. The main intuition behind our approach is once we are near the solution in the error landscape (Figure 1.3), we can perform an optimization to descend to the global minimum, the rigid transform that aligns the shapes with the minimum residual error. Although in this chapter we use point clouds as our modeling primitive, our method extends easily to other representations. Before we discuss the details of our method [MGPG04], let us take a look at some of existing methods for local shape alignment.

3.1 Related Work

A popular method for aligning two point clouds is the Iterated Closest Point (ICP) algorithm [BM92, CM91]. This algorithm starts with two point clouds and an estimate of the aligning rigid body transform. It then iteratively refines the transform by alternating the steps of choosing corresponding points across the point clouds, and finding the best rotation and translation that minimizes an error metric based on the distance between the corresponding points.

Despite a large amount of work on registration, convergence behavior of many registration algorithms, under different starting conditions, and error metrics, limited work has been done towards theoretical analysis of the algorithm [ESE06]. Experimentally, it has been shown that the rate of convergence of ICP heavily depends on the choice of the corresponding point-pairs, and the distance function that is being minimized [RL01]. Many enhancements of ICP-style algorithms for registration propose different error metrics, and point selection strategies, to improve ICP's convergence behavior [GIRL03, Fit01, JH03, CLSB92, RL01].

Two distance metrics are commonly used in ICP and its variants. The *point-to-point* distance of Besl [BM92] uses the Eucledian distance between the corresponding points. This leads to an ICP algorithm that converges slowly for certain types of input data and initial positions. Another error metric is the *point-to-plane* distance of Chen and Medioni [CM91], which uses the distance between a point and a planar approximation of the surface at the corresponding point. When the initial position of the data is close to the model, and when the input has relatively low noise, ICP with point-to-plane error metric has faster convergence than the point-to-point version. However, when the shapes start far away from each other, or for noisy point clouds, point-to-plane ICP tends to oscillate and fails to converge [GIRL03].

Another reason behind the slow convergence of registration algorithms based on ICP, is the local nature of the minimization. The only information used by the algorithm is the point correspondences. As a result, the minimized error function only approximates the squared distance between the two point clouds up to first order.

In this chapter, we propose an optimization framework for studying registration algorithms. We pose registration between two point clouds as an optimization over the space of rigid transforms. We develop an objective function that is a second order approximant to the squared distance between the model and the data. Higher order information about the surfaces represented by the point clouds, such as local curvatures, are incorporated into this quadratic approximant. Using such approximant to the squared distance function, we develop a registration algorithm. When the model and the data PCDs are close, our algorithm has a rate of convergence similar to ICP with point-to-plane error metric. Moreover, our method has a stable behavior even when the initial displacement is large. We also explain the convergence properties of the point-to-point and point-to-plane ICP variants, in terms

of the accuracy of the distance function that they use during minimization.

3.2 Registration of Point Cloud Data

Let $P = \{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}\}$ and $Q = \{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_m}\}$ be two point clouds in $\mathbb{R}^d$. The goal of the registration algorithm is to find a rigid body transform α composed of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ that best aligns the *data* PCD Q to match the *model* PCD P.

Registration algorithms based on ICP work as follows. Given the initial position of the data with respect to the model, the algorithm chooses a set of k point pairs $(\mathbf{p}_i, \mathbf{q}_i)$ from the model and the data. The distance between the model and data PCDs is approximated by the sum of distances between the point pairs. The algorithm then searches for the rigid transform that minimizes the residual distance, ϵ, between the model and the transformed data:

$$\epsilon(\alpha) = \sum_{i=1}^{k} d^2(\alpha(\mathbf{q}_i), \mathbf{p}_i), \tag{3.1}$$

where d can be point-to-point distance of Besl and McKay [BM92] or point-to-plane distance of Chen and Medioni [CM91]. Notice, however, that the basic assumption is that the sum of squared distances between pairs of points is a good approximation for the distance between two PCDs.

In the point cloud setting, we actually know that the model and data PCDs are not arbitrary collections of points, but are sampled from some underlying surfaces Φ_P and Φ_Q. In this case, it is more appropriate to minimize the distance from the data PCD to the *surface* represented by the model PCD. Pottmann and Hofer showed that when the data and the model are close, the point-to-plane distance is a good approximation to the distance between a data point and the surface represented by the model PCD. On the other hand, when the model and the data are far apart, the point-to-point distance is a better choice [PLH02].

The goodness of a given error metric is determined by two properties. First, we would like the error metric to accurately reflect the distance between a data point $\mathbf{q}$ and the surface represented by the model PCD. Second, we would like the distance approximation to be valid not just at a point $\mathbf{q}$, but in a neighborhood around it. Both point-to-point and point-to-plane error metrics are based only on first order information about the underlying input

surfaces. As a result, they do not provide a good approximation to the distance when we move around in the neighborhood of a data point $\mathbf{q}$.

In this chapter, we are concerned with developing a good approximation to the distance function between two point clouds. In order to show theoretical bounds on the convergence behavior of registration algorithms based on our distance function, we pose the problem of registration of two point clouds as an optimization problem over the space of rigid transforms. This leads to the following optimization problem: we are searching for the best rigid transform $\alpha = (\mathbf{R}, \mathbf{t})$ that minimizes the error given by

$$\epsilon(\alpha) = \sum_{i=1}^{m} d^2(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_P), \tag{3.2}$$

where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector. The function $d^2(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_P)$ is the distance from the transformed data point $\mathbf{q}_i$ to the surface represented by the PCD P. Given the optimization problem of Equation 3.2, it is clear that the convergence behavior of any such optimization procedure depends on the accuracy of the function d^2. We call d^2 the *squared distance function* to the surface Φ_P. Since the surface Φ_P from which the point model was sampled is generally complicated and unknown, a good approximant to the squared distance function is required.

Contributions

We develop a quadratic approximant to the squared distance function to the surface represented by a point cloud, and use this approximant in a registration algorithm. Our approximant has the desired property of being valid not only at the query point where it is computed, but also in a neighborhood around the query point. This property allows us to pose the registration problem in an optimization framework, and use methods such as Newton iteration, that depend on computing accurate derivatives of the objective function. In our optimization framework using the squared distance function, point-to-point and point-to-plane ICP variants are reduced to two special cases of the general minimization problem.

Our distance function approximates the squared distance from a data query point to the surface represented by the model PCD up to second order. We develop two methods for computing such local quadratic approximants. The first, uses local curvature of the surface

to incorporate second order information into the squared distance function. The second method, approximates the global error landscape by locally fitting quadric patches to the squared distance function to the surface. The quadric patches are stored in a special octree like structure called the *d2tree* [LPZ03]. For any point $\mathbf{q}$, the registration algorithm queries this special structure for the corresponding approximant to the squared distance to the surface. Unlike common ICP variants, the d2tree data structure allows us to perform registration without explicitly using nearest neighbors for correspondence.

Both of the above techniques incorporate information about the shape of the neighborhoods of the input surface into the error function. As a result we get better convergence behavior than purely local methods of ICP. Using our distance function, we develop a registration algorithm for point clouds that has fast convergence and is more stable behavior than standard ICP variants. When two point clouds are close to each other, our algorithm has quadratic convergence, similar to point-to-plane ICP. However, unlike point-to-plane ICP, our algorithm has more stable convergence behavior and is less prone to oscillations when the initial distance between the model and the data is large.

3.3 Registration using the squared distance function

In this section, we assume the existence of a function d^2 that for any point $\mathbf{x} \in \mathbb{R}^d$, gives the squared distance to the model PCD surface Φ_P. Such a squared distance function defines the error landscape for our objective function as indicated by Equation 3.2. So this function d^2 is important for registration algorithms. Later in Section 3.4, we describe how to generate local quadratic approximants F^+ of this function $d^2(\mathbf{x}, \Phi_P)$. Assuming these approximants are available, we now show how the registration problem can be solved in a least squared sense by a gradient descent search. Simply put, we try to place one point cloud in the squared distance field of the other, in order to minimize the placement error. Given Φ_P, we expect the points near its *medial axis* $MA(\Phi_P)$ to have bad quadratic approximants, since locally, the squared distance function is not smooth. If we detect such points, we leave them out of our optimization procedure.

We employ an iterative scheme to solve the nonlinear optimization problem over the complex error landscape. At each stage, we solve for a rigid transform composed of a rotation

$\mathbf{R}$ followed by a translation $\mathbf{t}$. We use F^+ to solve for the rigid transform $\alpha = (\mathbf{R}, \mathbf{t})$ that brings the data PCD Q to the model PCD P. We apply this transform and repeat.

3.3.1 Registration in 2D

We first explain the process in 2D. At any point $\mathbf{x} = [x \;\; y] \in \mathbb{R}^2$, we assume the availability of an approximant F^+ such that $F^+(\mathbf{x}) \approx d^2(\mathbf{x}, \Phi_P)$. Let F^+ be specified in the form

$$F^+(\mathbf{x}) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F,$$

where A, B, C, D, E, F are the coefficients of the approximant. More compactly, we can write F^+ in a quadratic form as

$$F^+(\mathbf{x}) = \begin{bmatrix} x & y & 1 \end{bmatrix} \quad \mathcal{Q}_{\mathbf{x}} \quad \begin{bmatrix} x & y & 1 \end{bmatrix}^T , \tag{3.3}$$

where $\mathcal{Q}_{\mathbf{x}}$ is a symmetric matrix that depends on $\mathbf{x}$. Importantly, the approximant $F^+(\mathbf{x})$ is a valid locally around $\mathbf{x}$.

We denote any point $\mathbf{q}_i$ of the data PCD Q by $[x_i \quad y_i]$. Let a matrix $\mathbf{R}$ corresponds to a rotation by angle θ around the origin. Our goal is to solve for a rigid transform, which consists of $\mathbf{R}$ followed by a translation vector $\mathbf{t} = [t_x \quad t_y]$, that minimizes $\sum_{i=1}^{m} F^+(\mathbf{R}\mathbf{q}_i + \mathbf{t})$. For small θ, we can linearize the rotation by using $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. So after each iteration step, we get, $[x_i \quad y_i] \mapsto [x_i - \theta y_i + t_x \quad \theta x_i + y_i + t_y]$. If locally $\mathcal{Q}_{\mathbf{q}_i}$ stays approximately fixed, the residual error between the transformed data and the model PCDs is given by

$$\begin{aligned} \epsilon(\theta, t_x, t_y) = \sum_{i=1}^{m} \begin{bmatrix} x_i - \theta y_i + t_x & \theta x_i + y_i + t_y & 1 \end{bmatrix} \mathcal{Q}_{\mathbf{q}_i} \\ \begin{bmatrix} x_i - \theta y_i + t_x & \theta x_i + y_i + t_y & 1 \end{bmatrix}^T, \end{aligned} \tag{3.4}$$

where $\mathcal{Q}_{\mathbf{q}_i}$ denotes the matrix representing the approximant of the squared distance function to Φ_P around the point $\mathbf{q}_i$. Our goal is to find values of θ, t_x, and t_y that minimize this residual error. Setting the respective partial derivatives of the error ϵ to zero, we get the following

linear system

$$\left[\sum_{i=1}^{m}\begin{pmatrix} \mathcal{I}_i & \mathcal{J}_i & \mathcal{K}_i \\ \mathcal{J}_i & 2A_i & B_i \\ \mathcal{K}_i & B_i & 2C_i \end{pmatrix}\right]\begin{bmatrix} \theta \\ t_x \\ t_y \end{bmatrix} = -\left[\sum_{i=1}^{m}\begin{pmatrix} \mathcal{L}_i \\ 2A_ix_i + B_iy_i + D_i \\ B_ix_i + 2C_iy_i + E_i \end{pmatrix}\right], \tag{3.5}$$

where $\mathcal{I}_i = 2(C_ix_i^2 - B_ix_iy_i + A_iy_i^2)$, $\mathcal{J}_i = B_ix_i - 2A_iy_i$, $\mathcal{K}_i = 2C_ix_i - B_iy_i$, $\mathcal{L}_i = B_i(x_i^2 - y_i^2) + 2(C_i - A_i)x_iy_i + E_ix_i - D_iy_i$ and, $A_i, B_i, C_i, D_i, E_i, F_i$ denote the entries of the matrix $\mathcal{Q}_{\mathbf{q}_i}$. The transformation resulting from solving Equation 3.5 is applied to Q. This completes one iteration of our gradient descent process. Next we use the approximants corresponding to the new positions of $\mathbf{q}_i$ to get another linear system whose solution is again applied to the data PCD. This process is iterated until the residual error falls below a pre-defined error threshold or a maximum number of iteration steps is reached.

Since we do not make any assumption about the initial alignment of P and Q, the rigid transform computed at any step, can be large. In such cases, we can only take small steps in the direction of the transform because its computation is based on approximants that are valid only locally. This issue of applying a $1/\eta$-fraction of a rigid transform is an important problem and has been studied in depth in other places [Ale02].

We propose a simple way for computing fractional transforms. In our notation, the computed transform vector $[\theta \;\; t_x \;\; t_y]$ denotes a rigid transform composed of a rotation matrix $\mathbf{R}$ followed by a translation vector $\mathbf{t}$. This maps $\mathbf{q}$ to a point $\mathbf{Rq} + \mathbf{t}$. Let the fractional transform be composed of a rotation matrix $\mathbf{R}'$ and a translation vector $\mathbf{t}'$. We define ($\mathbf{R}'$,$\mathbf{t}'$) to be a $1/\eta$ fraction of ($\mathbf{R}$,$\mathbf{t}$) if the following relation holds,

$$(..(\mathbf{R}'(\mathbf{R}'\mathbf{q} + \mathbf{t}') + \mathbf{t}')...\eta \text{ times}) \equiv \mathbf{Rq} + \mathbf{t}. \tag{3.6}$$

From this relation, we can get $\mathbf{R}' = \mathbf{R}^{1/\eta}$, and $\mathbf{t}' = (\mathbf{R} - \mathbf{I})^{-1}(\mathbf{R}' - \mathbf{I})\mathbf{t}$ where, $\mathbf{I}$ is the identity

matrix. By $\mathbf{R}^{1/\eta}$, we mean a rotation around the origin by angle θ/η. We defer the important issue of choosing a suitable value for η to Section 3.5.

3.3.2 Registration in 3D

In this section, we extend the results from the previous section to point cloud surface data in $\mathbb{R}^3$. For any point $\mathbf{x} = [x \;\; y \;\; z] \in \mathbb{R}^3$, let the corresponding local quadratic approximant F^+ be specified in the form

$$F^+(\mathbf{x}) = Ax^2 + Bxy + Cy^2 + Dxz + Eyz + Fz^2 + Gx + Hy + Iz\,, \tag{3.7}$$

where A through I are the coefficients of the quadratic approximant. With slight abuse of notation, this equation can be written in a quadratic form as $F^+(\mathbf{x}) = \mathbf{x}\mathcal{Q}_{\mathbf{x}}\mathbf{x}^T$, where $\mathbf{x}$ now denotes $[x \;\; y \;\; z \;\; 1]$.

Once again, our goal is to find the rigid transform which brings Q in best alignment with P. Let the rigid transform be composed of a rotation matrix, $\mathbf{R}$, that is parameterized by three angles (α, β, γ) in the *X-Y-Z fixed angle orientation* convention, followed by a translation vector $\mathbf{t} = [t_x \;\; t_y \;\; t_z]$. Under small motion, the rotation matrix can be linearized as,

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \\ &\quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \\ &\approx \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix}. \end{aligned} \tag{3.8}$$

Hence $\mathbf{q} \mapsto \mathbf{R}\mathbf{q} + \mathbf{t}$.

Now the registration problem reduces to finding values of $\alpha, \beta, \gamma, t_x, t_y, t_z$ that minimize

the residual error

$$\epsilon(\alpha,\beta,\gamma,t_x,t_y,t_z)=\sum_{i=1}^{m}(\mathbf{R}\mathbf{q}_i+\mathbf{t})\,\mathcal{Q}_{\mathbf{q}_i}(\mathbf{R}\mathbf{q}_i+\mathbf{t})^T. \tag{3.9}$$

This least square problem can be solved by setting the respective partial derivatives to zero. The resulting linear system is given by

$$\begin{aligned}\left[\sum_{i=1}^{m}\begin{pmatrix}\mathcal{P}_i & \mathcal{S}_i\\ \mathcal{S}_i^T & \mathcal{R}_i\end{pmatrix}\right]\begin{bmatrix}\alpha & \beta & \gamma & t_x & t_y & t_z\end{bmatrix}^T=\\ -\begin{bmatrix}\sum_{i=1}^{m}\begin{pmatrix}\mathcal{U}_i\\ \mathcal{V}_i\\ \mathcal{W}_i\\ 2A_ix_i+B_iy_i+D_iz_i+G_i\\ B_ix_i+2C_iy_i+E_iz_i+H_i\\ D_ix_i+E_iy_i+2F_iz_i+I_i\end{pmatrix}\end{bmatrix},\end{aligned} \tag{3.10}$$

where,

$$\mathcal{P}_i=\begin{bmatrix}\mathcal{J}_i & \mathcal{M}_i & \mathcal{N}_i\\ \mathcal{M}_i & \mathcal{K}_i & \mathcal{T}_i\\ \mathcal{N}_i & \mathcal{T}_i & \mathcal{L}_i\end{bmatrix},$$

$$\mathcal{S}_i=\begin{bmatrix}B_ix_i-2A_iy_i & 2C_ix_i-B_iy_i & E_ix_i-D_iy_i\\ -D_ix_i+2A_iz_i & -E_ix_i+B_iz_i & -2F_ix_i+D_iz_i\\ D_iy_i-B_iz_i & E_iy_i-2C_iz_i & 2F_iy_i-E_iz_i\end{bmatrix},$$

$$\mathcal{R}_i=\begin{bmatrix}2A_i & B_i & D_i\\ B_i & 2C_i & E_i\\ D_i & E_i & 2F_i\end{bmatrix},$$

$$
\begin{aligned}
\mathcal{J}_i &= 2(C_i x_i^2 - B_i x_i y_i + A_i y_i^2), \\
\mathcal{K}_i &= 2(F_i x_i^2 - D_i x_i z_i + A_i z_i^2), \\
\mathcal{L}_i &= 2(F_i y_i^2 - E_i y_i z_i + C_i z_i^2), \\
\mathcal{M}_i &= -E_i x_i^2 + D_i x_i y_i + B_i x_i z_i - 2A_i y_i z_i, \\
\mathcal{N}_i &= -D_i y_i^2 + E_i x_i y_i - 2C_i x_i z_i + B_i y_i z_i, \\
\mathcal{T}_i &= -B_i z_i^2 - 2F_i x_i y_i + E_i x_i z_i + D_i y_i z_i, \\
\mathcal{U}_i &= B_i(x_i^2 - y_i^2) + 2(C_i - A_i) x_i y_i + E_i x_i z_i - D_i y_i z_i + H_i x_i - G_i y_i, \\
\mathcal{V}_i &= D_i(z_i^2 - x_i^2) - E_i x_i y_i + 2(A_i - F_i) x_i z_i + B_i y_i z_i - I_i x_i + G_i z_i, \\
\mathcal{W}_i &= E_i(y_i^2 - z_i^2) + D_i x_i y_i - B_i x_i z_i + 2(F_i - C_i) y_i z_i + I_i y_i - H_i z_i,
\end{aligned}
$$

and A_i through I_i correspond to the entries of the matrix $\mathcal{Q}_{\mathbf{q}_i}$ that represents the local quadratic approximant around the point $\mathbf{q}_i$.

As in the 2D case, whenever the computed transform ($\mathbf{R}$,$\mathbf{t}$) is large, we utilize a fractional transform given by $\mathbf{R}' = \mathbf{R}^{1/\eta}$ and $\mathbf{t}' = (\mathbf{R} - \mathbf{I})^{-1}(\mathbf{R}' - \mathbf{I})\mathbf{t}$ where, $\mathbf{I}$ denotes the identity matrix. A $1/\eta$-fraction of the rotation matrix $\mathbf{R}$ can be computed by the techniques proposed by Alexa [Ale02].

3.4 Squared Distance Function

Given a 3D point cloud P, we describe two methods for constructing a quadratic approximant F^+ to the squared distance function d^2 from any point $\mathbf{x} \in \mathbb{R}^3 \setminus MA(\Phi_P)$ to Φ_P. At any point $\mathbf{x}$, our goal is to construct an approximant F^+ such that $F^+(\mathbf{x}) \approx d^2(\mathbf{x}, \Phi_P)$ is second order accurate. Points on the medial axis $MA(\Phi_P)$ have non-differentiable squared distance function and hence, their second order accurate approximants do not exist. In the construction phase, we ensure that the approximants are non-negative over $\mathbb{R}^3$, since F^+ is used as an objective function in a minimization process as shown in Section 3.3. In 2D, similar approximants can be easily computed.

Before we describe how to compute F^+ for a given PCD, we summarize a few basic results on the squared distance function of a surface as observed by Pottmann and Hofer [PH03].

For each point on a given surface, we assume that the unit normal $\vec{n}$ along with the principal curvature directions $\vec{e}_1, \vec{e}_2$ are given. These three unit vectors combine to form a local coordinate system called the *principal frame*. At *umbilical points*, where the principal curvature directions are not well-defined, any two orthogonal unit vectors on the tangent plane may be used as $\vec{e}_1, \vec{e}_2$. Let ρ_i be the *principal radius of curvature* in the direction $\vec{e}_i$. The *normal footpoint* $\mathbf{y}$ denotes the closest point on the surface from $\mathbf{x}$. Let x_1, x_2, x_3 represent the coordinates of $\mathbf{x}$ in the principal frame at $\mathbf{y}$. The signed distance from $\mathbf{x}$ to its normal footprint is denoted by d. The sign of d is positive if $\mathbf{x}$ and the centers of the osculating circles at $\mathbf{y}$, lie on the same side of the surface around $\mathbf{y}$.

The second order Taylor approximant [PH03] of the squared distance function to the surface at a point $\mathbf{x}$ can be expressed in the principal frame at $\mathbf{y}$ as

$$F_d(\mathbf{x}) = F_d(x_1, x_2, x_3) = \frac{d}{d-\rho_1} x_1^2 + \frac{d}{d-\rho_2} x_2^2 + x_3^2. \tag{3.11}$$

We shall use δ_j, $j = 1, 2$, to denote $d/(d-\rho_j)$.

Let us look at two important special cases.

- For $d = 0$ we obtain $F_d(x_1, x_2, x_3) = x_3^2$. Thus, if we are close to the surface, i.e. in its *'near-field'*, the squared distance function to the tangent plane at the normal footpoint, is a quadratic approximant.

- For $d = \infty$ we obtain $F_d(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$, which is the squared distance from $\mathbf{x}$ to its footpoint $\mathbf{y}$. So the distances to normal footpoints are second order accurate if we are in the *'far-field'* of the surface.

In order to use F_d as an objective function for a minimization, we want the approximant to be non-negative over $\mathbb{R}^3$. To this end, we replace δ_j with

$$\hat{\delta}_j = \begin{cases} d/(d-\rho_j) & \text{if } d < 0, \\ 0 & \text{otherwise.} \end{cases}$$

The resulting approximant F^+ is positive definite and is given by

$$F^+(\mathbf{x}) = \hat{\delta}_1 x_1^2 + \hat{\delta}_2 x_2^2 + x_3^2. \tag{3.12}$$

This quadratic approximant F^+ of d^2 is simply a weighted sum of the squared distance functions x_1^2, x_2^2, x_3^2 to three planes: the two principal planes and the tangent plane at the normal footpoint. Based on this observation, we transform Equation 3.12 to the global coordinate system as,

$$\begin{aligned} F^+(\mathbf{x}) &= \hat{\delta}_1 \left(\vec{e}_1 \cdot (\mathbf{x} - \mathbf{y})\right)^2 + \hat{\delta}_2 \left(\vec{e}_2 \cdot (\mathbf{x} - \mathbf{y})\right)^2 + \\ &\quad \left(\vec{n} \cdot (\mathbf{x} - \mathbf{y})\right)^2. \end{aligned} \tag{3.13}$$

We can now express this equation in the form given by Equation 3.7 to get values for the coefficients A through I.

3.4.1 On-Demand Computation

Given a point $\mathbf{x}$, in our first method for computing a second order accurate squared distance field F^+ for a given PCD P, we perform an *on-demand* computation of Equation 3.13. For this method we first need to compute the normal footpoint of $\mathbf{x}$ to P. As an approximation, we treat $\mathbf{p}$, the closest point to $\mathbf{x}$ in P, as the normal footpoint. This point is found using an approximate nearest neighbor data structure [AMN+98]. Figure 3.1 shows the scenario in 2D. When the P is a sparse sampling of Φ_P, we can use the underlying moving least square (MLS) surface to get a better approximation for the normal footpoint [AK04]. We further need to evaluate local curvatures at points of P in order to use Equation 3.13. These quantities are computed in the preprocessing step of our algorithm.

At each point of a given PCD, we first determine the principal frame using a local covariance analysis as detailed in [CP03, MNG04]. If the the underlying surface Φ_P is *regular*, at each of point $\mathbf{p}$ of P, a local parametrization exists. In the principal frame at $\mathbf{p}$, we estimate the local surface by least square fitting a quadratic function of the form $ax^2 + bxy + cy^2 + dx + ey$ to the neighboring points in P. Once we estimate the coefficients a through e, we can use facts from differential geometry to get the Gaussian curvature K and the mean curvature

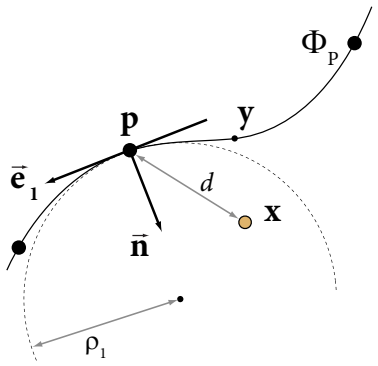


Figure 3.1: *Point Footprint.* A query point $\mathbf{x} \in \mathbb{R}^2$ has a footpoint $\mathbf{y} \in \mathbb{R}^2$ on the surface Φ_P represented by a PCD P. We approximate the footpoint by $\mathbf{p}$, the closest point in P from $\mathbf{x}$. The principal frame at the footpoint is spanned by $\vec{e}_1$ and $\vec{n}$. The osculating circle to Φ_P at $\mathbf{p}$ has a radius of curvature ρ_1. In the figure, the signed distance d from $\mathbf{x}$ to the footpoint $\mathbf{p}$ is positive.

H using

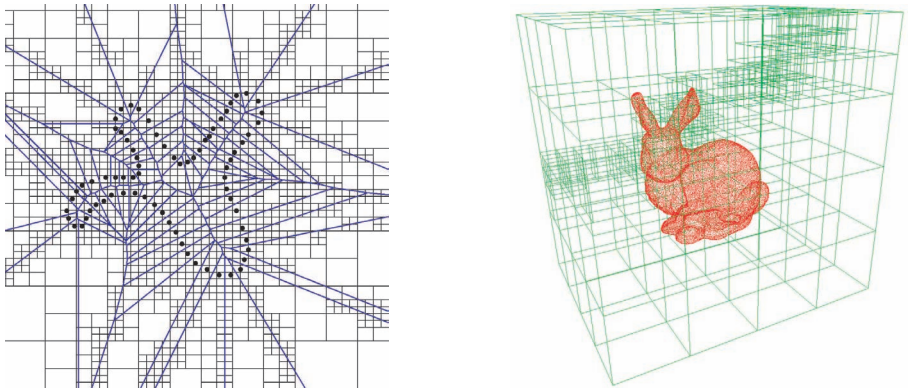
$$\begin{aligned} K &= \frac{4ac - b^2}{(1 + d^2 + e^2)^2} \\ H &= \frac{a(1 + e^2) - bde + c(1 + d^2)}{(1 + d^2 + e^2)^2}. \end{aligned} \tag{3.14}$$

Finally we evalute the principal radii of curvature ρ_i as $1/(H \pm \sqrt{H^2 - K})$.

The correctness of these estimates depends on the sampling density of the given PCD and on the measurement noise. Further, the neighborhood size used for the least square fits can be adapted to the local shape [MNG04]. In low noise scenarios, when the local estimates of the differential properties can be reliably computed, the approximants F^+ given by this method are good.

3.4.2 Quadratic Approximants using d2Tree

Our second method for computing approximate quadratic approximants involves least square fitting of quadratic patches to a sampled squared distance function. For a given PCD, these quadratic patches are pre-computed and stored in a special data structure called the *d2Tree* [LPZ03]. Given any point $\mathbf{x}$, in this method we do a point location in the cells of the d2Tree



(a) (b)

Figure 3.2: *d2Tree for curves and surfaces.* d2Tree can be used in 2D (left) and in 3D (right) to store quadratic approximants of the squared distance fields correct to some error threshold. The maximum number of levels and the error threshold, which are parameters used during the construction of this quad-tree like data-structure, determine the size of the cells. In the 2D case, we overlay the Voronoi diagram of the PCD on top of the d2Tree, to illustrate that small cells are created around the medial axis.

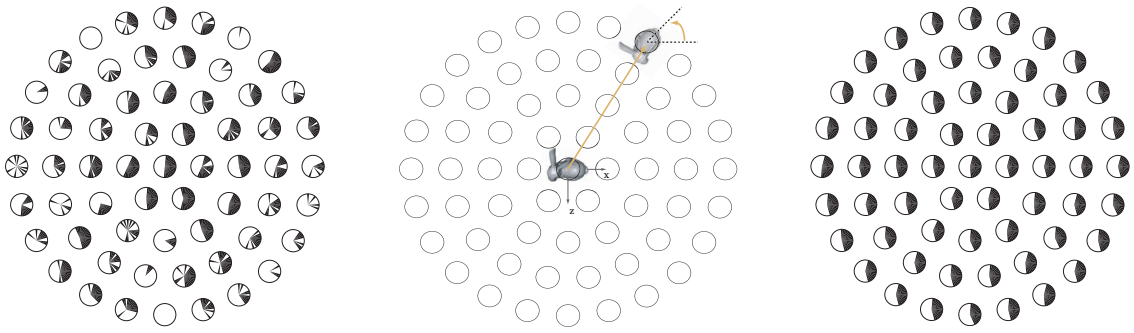
and return the quadratic approximant stored in the corresponding cell.

Simply put, the d2Tree is an octree-like (quad-tree in 2D) data structure, where each cell stores a quadratic fit to the squared distance function, correct to some maximum error threshold. The approximants are stored in the form as given in Equation 3.7 (Equation 3.3 in 2D). Details of a top-down construction of *d2Tree* can be found in [LPZ03]. Here we describe a bottom-up construction, which is computationally more efficient.

As a first step, a sampled squared distance field is build for an input PCD by sweeping the space starting from the PCD P and propagating the squared distance information [LPZ03]. Depending on the number of levels, which is an input to the algorithm, the space is divided into smallest allowable cells (see Figure 3.2). In each cell, a quadric patch, that best fits the sampled squared distance field, is computed. The fitting error and the matrices used to compute the coefficients of the fit are saved in each cell. At the next level, the neighboring cells (four in 2D and eight in 3D) are merged to form a larger quadric patch, only if the resulting fitting error is below the given error threshold. The larger quadric patches can be efficiently fitted by re-using the matrices stored in the smaller cells. The quadratic matrix stored in any cell is made positive semi-definite during construction. The maximum number

of levels of the tree and the error threshold are the required parameters for the construction of this data structure. Notice that there exists a tradeoff between the size of the cells and the accuracy of the quadratic approximants.

Unlike the on-demand method for computing quadratic approximants described before, the d2Tree approach does not need estimates of the local curvature or any nearest neighbor structure. Quadratic approximants computed by d2Tree, implicitly learn the local curvature information by fitting quadrics to the sampled squared distance field. We find this method to be robust to noisy or under-sampled PCD. Given a query point $\mathbf{x}$, computing $F^{+}(\mathbf{x})$ simply involves a point location in this d2Tree structure, and does not require any explicit correspondence between points of the input PCDs.



(a) (b) (c)

Figure 3.3: *Funnel of convergence for aligning the bunny model to itself.* The bunny is rotated (around the y-axis) and translated (along the x-z plane) to generate different initial positions for the *data PCD*. The figure in the middle denotes the sampling pattern used to get the initial positions. The rotation angle is sampled at 10° intervals, while the maximum radial translation of the bunny is around $5\times$ the height of the bunny. Regions in black denote convergence to the correct solution. The convergence funnel of the point-to-plane ICP (left) is found to be quite narrow and unstable. Under similar conditions, our on-demand algorithm (right) is found to have a significantly broader, and much more stable convergence funnel. The shape of the convergence funnel corresponding to our d2Tree based approach is similar.

3.4.3 Point-to-point and point-to-plane ICP error metrics as special cases of quadratic approximant

In our framework, the standard ICP algorithms can be reduced to special cases by selecting suitable approximants to the squared distance function. Basic point-to-point ICP uses squared distance to the closest point as its approximant, i.e. $F^+(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|^2$ while the Chen–Medioni point-to-plane ICP uses $F^+(\mathbf{x}) = (\vec{n} \cdot (\mathbf{x} - \mathbf{y}))^2$ as the quadratic approximant (Equation 3.7). In the form given by Equation 3.13, point-to-point ICP has $\hat{\delta}_i = 1$, and point-to-plane ICP has $\hat{\delta}_i = 0$. From an optimization perspective, there are only slight differences between this ICP-like algorithm and the standard ICP [BM92], which does not linearize the motion [Pot04]. However, as pointed out in Section 3.4, such approximants are second order accurate only in the 'far field' and 'near field' of the PCD, respectively, and hence neither of these algorithms is well-behaved for all initial placements of the model and data PCDs.

3.5 Convergence Issues

In this section, we discuss the convergence behavior of point-to-point and point-to-plane ICP algorithms, and then give bounds on the convergence rates for our algorithm. In contrast to ICP algorithms, our scheme uses second order accurate square distance approximants at all point in space, and hence, exhibits better convergence properties.

Experimentally, the point-to-point ICP algorithm converges linearly. In a recent result, Pottmann has provided theoretical justification for this behavior [Pot04]. We define a *low residual* problem as one where the data shape fits the model shape well, and a *zero residual* problem as one where the fit is exact. For a low residual problem, when the minimizer is approached tangentially, point-to-point ICP has a very slow convergence[Pot04, RL01].

We recall that the Chen-Medioni approach iteratively minimizes the sum of squared distances to the tangent planes at the normal footpoints of the current data point locations. This implies a gradient descent in the error landscape, where the squared distance to the tangent plane is used to define the objective function. From an optimization perspective, this process corresponds to Gauss–Newton iteration. For a zero residual problem, and a sufficiently good initial position, this algorithm converges quadratically [Pot04]. In practice,

Algorithm 1 Brings a data PCD Q in alignment to a model PCD P using *on-demand* computation or *d2Tree* based approach or point-to-point ICP or point-to-plane ICP.

if using *on-demand* method **then**
  Build an approximate nearest neighbor structure for P.
  Pre-compute principal frame and radii of curvature ρ_j, $j = 1, 2$, at each point $\mathbf{p}_i \in P$ as described in Section 3.4.1.
else if using *d2Tree* method **then**
  Build *d2Tree* with a suitable error threshold. (see Section 3.4.2)
end if
$count \leftarrow$ MaxCount
repeat
  for each point $\mathbf{q}_i \in Q$ **do**
    Compute $F^+(\mathbf{q}_i)$ using method described in Section 3.4.1 for *on-demand* approach. For *d2Tree* based approach refer to Section 3.4.2. For point-to-point or point-to-plane ICP refer to Section 3.4.3.
  end for
  Using $F^+(\mathbf{q}_i)$, build and solve the linear system given in Section 3.3.
  if Armijo condition not satisfied **then**
    Take $1/\eta$ fraction of the computed rigid transform (see Section 3.3). Value of η chosen via line-search to satisfy Armijo rule (see Section 3.5).
  end if
  if (residual error $<$ ErrorThreshold) **then**
    break
  end if
  $count \leftarrow count - 1$
until $count \neq 0$

the Chen-Medioni method also works well for low residual problems. Notice that in the 'near-field', the squared distance to the tangent plane is a second order accurate approximant to d^2. So point-to-plane ICP performs much better for fine registration than the point-to-point ICP algorithm. However, there is no reason to expect convergence when the two PCDs are sufficiently apart in the transform space, and in practice, this is found to be true.

For low residue problems, our algorithm also exhibits quadratic convergence, which means that the error reduction is of the form

$$\epsilon_+ \leq C\epsilon_c^2 \tag{3.15}$$

where, C denotes the convergence constant, ϵ_c denotes the error in the current step, and ϵ_+ denotes the error after application of the computed rigid transform. Here, the error measures the distance of a position of the PCD to the optimal final position, e.g. via the quadratic mean of the distances between current and final data point locations. For each point $\mathbf{q}_i$, our algorithm computes a second order approximant F^+ to the squared distance function of the surface Φ_P represented by PCD P. Using these approximants, we derive the best aligning transform for Q by following a gradient descent with Newton iteration steps [Kel99] in the rigid transform group (see Section 3.3). We continue until the residual error falls below a pre-defined threshold or a maximum number of iteration steps has been reached. Since the presented method is a Newton algorithm, it converges quadratically [Kel99, Pot04].

As mentioned in Section 3.3, if the residue ϵ is large, we apply only $1/\eta$ fraction of the computed transform to prevent oscillations, or even divergence. Various line search strategies exist for choosing good values for η [Kel99]. In our implementation, we used the *Armijo condition* [Kel99] to select η. This results in a *damped Gauss–Newton algorithm*. It is well known in optimization, that algorithms which uses the Armijo rule converge linearly. Hence, to ensure faster convergence for large residue problems, it may be better to select a quadratic approximant of the motion, instead of a linear one [Pot04].

Our gradient descent based optimization can get stuck at a local minimum. We bound the maximum residual error for a given PCD pair, and use it to detect a local minimum. A point cloud P, sampled from a surface Φ_P, is said to be *sampled r-dense*, if any sphere with radius r centered on Φ_P contains at least one sample point in P [HDD+92]. Suppose that the model PCD P is an r_P-dense sampling of Φ_P. Further, assume the measurement noise only perturbs any point by a maximum amount of σ_P and σ_Q respectively for the given PCDs P, Q. Under this restrictive sampling model, when Φ_Q represents a subset of Φ_P, a final residual matching error ϵ greater than $M(r_P + \sigma_P + \sigma_Q)^2$ indicates the algorithm has been stuck at a local minima during the search process. When such a situation happens, we may randomly perturb Q to a new orientation, and try to align the two PCDs starting from that position.

To further study this global convergence property, we define the *funnel of convergence* for a registration algorithm, as the set of all initial poses of a PCD Q, which can be successfully aligned with P, using the given algorithm. Notice that the funnel only measures

global convergence and not speed. A *broad funnel* indicates that the algorithm can successfully handle a wide range of initial positions. An algorithm is said to have a *stable funnel* if the convergence zones, in the transform space, are clustered and not arbitrarily distributed. A stable funnel is desirable, since this can enable a systematic way of generating positions for random re-starts, using some branch and bound approach. Experimentally, we observe that our algorithm has a broader and more stable funnel of convergence as compared to the point-to-plane ICP variant. This can be explained by the fact, that our algorithm makes use of higher order surface properties.

3.6 Results

We test our algorithm on a variety of data sets with different amounts of noise, and compare its performance against point-to-point and point-to-plane ICP algorithms.

A brief summary of our registration framework is given in Algorithm 1. We compare the performance of approaches based on the choice of the approximant F^+ of the squared distance function at any point $\mathbf{x}$:

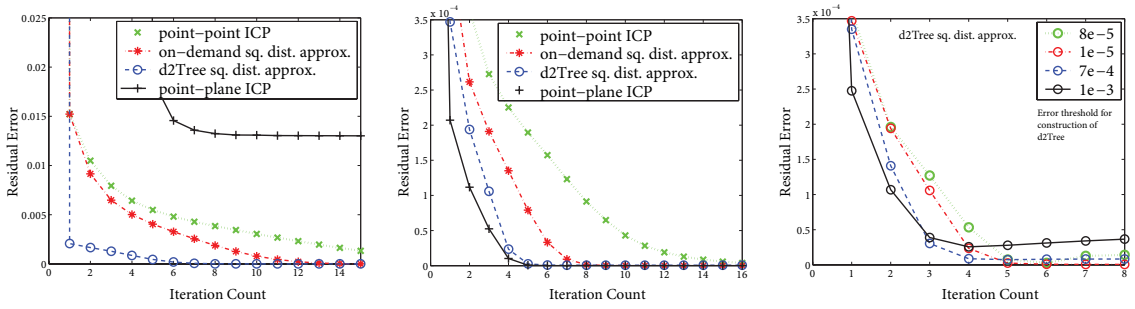
1. on-demand computation of quadratic approximant (Section 3.4.1)
2. quadratic approximant using d2Tree (Section 3.4.2)
3. squared point-to-point distance (point-to-point ICP)
4. squared distance to the tangent point at the footpoint of $\mathbf{x}$ (point-to-plane ICP)

In our implementation, we test for Armijo condition to ensure stability of the algorithms.

On the bunny model, which consists of $50,282$ points, we compare the convergence funnel of point-to-plane ICP and that of our algorithm based on on-demand computation. A copy of the same PCD is rotated around the y-axis and translated to different positions along the x-z plane. Figure 3.3 shows that the convergence funnel of point-to-plane ICP is quite narrow when the initial displacement is large. Under similar conditions, our algorithm is found to have a much broader convergence funnel. Our convergence funnel is also more stable. Experimentally, the initial translation is found to have little effect on the convergence of our algorithm.

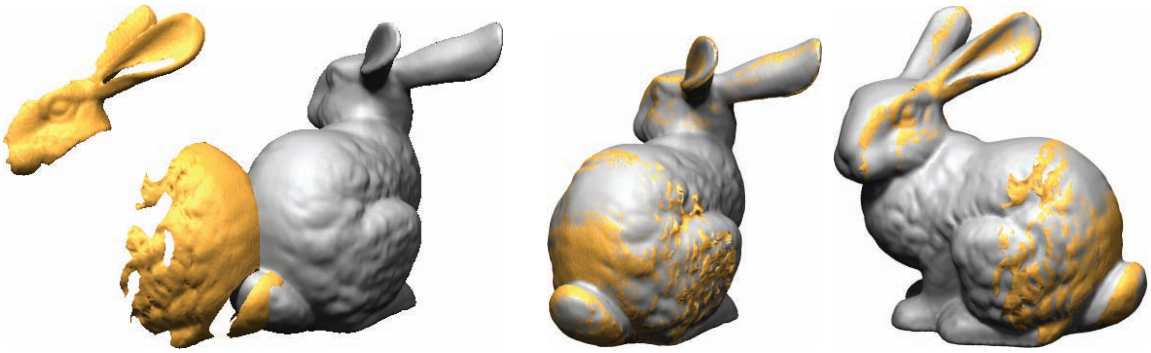
Next we compare the convergence rates for the four variants listed before. For both on-demand and d2Tree based approaches, the pre-computation time depends on the size n of the model PCD. For point-to-point, point-to-plane and on-demand computation, at each iteration, F^+ for a point $\mathbf{x}$ can be computed in O(1) time after the nearest neighbor query has been answered. For d2Tree, the nearest neighbor query is replaced by point location in the d2Tree cells. The solution of the linear system involves an inversion of a 6×6 matrix. Since the amount of work in each iteration step for any of the algorithms is roughly same, we simply count the number of iterations for comparing speed.

In Figure 3.4, we plot the residual error vs iteration count for four approaches. In the presence of noise and for large residues, point-to-plane ICP often fails to converge. In such noisy scenarios, since the estimates of local principal radii are bad, our on-demand algorithm is found to be marginally worse than point-to-point ICP. The d2Tree based method still converges fast, since the cell-sizes automatically get adjusted during their construction



(a) (b) (c)

Figure 3.4: *Residual error plots.* Plots of residual error vs iteration count for bunny PCD. When the *model* and *data* PCDs, both corrupted with noise, start far apart in the transform space, the point-to-plane ICP fails to converge to the correct solution (left). However, algorithms using any of the other square distance approximants do converge, with the *d2Tree* based approach converging fastest. Middle: For good initial position and small residual problem (the two PCDs align well), the point-to-point ICP algorithm has a slow convergence, while optimization based on any of the other squared distance approximants, converges quadratically. The figure to the right shows the effect of changing the error threshold value used for constructing the d2Tree. As the threshold is increased, a larger neighborhood of the squared distance function is captured by each of the cells of the tree and hence, the algorithm converges faster. However, for sufficiently high error threshold values, the distance approximants get too crude, and the method starts to deteriorate.



(a) (b)

Figure 3.5: *Subpart matching on the bunny model.* A partial scan of the bunny (shown in purple) is registered to the bunny, the *model* PCD. The initial arrangement of the PCDs is shown to the left. Our algorithm found the correct match (middle,right) in six iterations.

phase, to partially average out the effect of noise. However, in low residual cases, for reasons explained in Section 3.5, all algorithms except for point-to-point ICP converge quadratically. The threshold value used for constructing the d2Tree is also varied. As the threshold is increased, a larger neighborhood of the squared distance function is captured by each of the cells of the tree and hence, the algorithm converges faster. However, for sufficiently high error threshold values, the distance approximants get too crude, and the method starts to deteriorate.

Our algorithm is able to handle the case when the data PCD is a subset of the model PCD. We take a partial scan of the bunny consisting of $17,600$ points. This scan is from scanned data and is corrupted with measurement noise. We used the on-demand algorithm to match the partial scan to the complete bunny model PCD. The starting arrangement and the final match are shown in Figure 3.5.

Finally we test the robustness of our approach in presence of noise and varying sampling density. We try to align a part (consisting of $14,519$ points) of a ball-joint with the socket of a hip-bone represented by $132,538$ points. Note the sampling density and sampling pattern are vastly different across the two models. The ball-joint is much densely sampled compared to the hip-bone. Even in this case, for reasonable starting positions, we got a good final alignment (see Figure 3.6). The whole ball-joint is shown just to illustrate the goodness of the alignment. We manually selected a part of the ball-joint to satisfy our constraint that Φ_Q

(a) (b)

Figure 3.6: *Subpart matching on the hip-joint model.* The goal is to fit a part (shown in purple) of the ball-joint to the hip-bone (shown in green). The starting arrangement (left) and the final alignment (right) are shown. The whole ball-joint is shown to help the reader judge the correctness of the match. Our algorithm is robust enough to handle varying sampling density and noise in the given PCDs.

represents a subset of Φ_P.

3.7 Summary

In this chapter, we defined distance between two shapes. Given such a notion of distance and assuming the models are in good initial alignment, we showed how to bring the objects into a better alignment using Gauss-Newton optimization.

Using techniques presented so far, we can acquire geometry of an object by capturing its shape from multiple directions and then stitching the scans together using global, followed by local registration. In many cases, once we put the scans together we observe that we failed to capture the complete object. At this point we can acquire more scans corresponding to the missing regions. Is there an alternate solution? In the next chapter, we explore this question.

4

Shape Completion using Geometric Priors

Done is better than perfect.

— *Scott Allen*

In a scanning scenario, after having put together several scans, it is often the case, that we find regions missing in the combined object. In many applications accurately determining the missing regions, using further scans, is not always required. In fact any *plausible* and seamless substitution of the missing parts is acceptable. In this chapter, we describe how to perform *shape completion* using information from a database of existing shapes. Using our framework, we can obtain a complete and consistent 3D model representation using incomplete scans and geometric priors [PMG+05].

Obtaining a complete and consistent 3D model representation from acquired surface samples is often a tedious process, and can easily take multiple hours, even for an experienced user. Significant manual assistance is often required for tasks such as scan path planning, data cleaning, hole filling, alignment, and model extraction. Knowledge about the acquired shape gained in this process is typically not utilized in subsequent scans, where the same time consuming procedure has to be repeated all over again. Our goal is to simplify the model creation process by exploiting previous experience on shapes stored in a 3D model database. This allows the generation of clean and complete 3D shape models even

from highly incomplete scan data, reducing the complexity of the acquisition process significantly. The main idea is to mimic the experienced-based human approach to shape perception and understanding. This is possible because we make extensive use of prior knowledge about shapes, acquired over years of experience. When seeing an object, we immediately put it into context with other similar shapes that we have previously observed and transfer information from those shapes to fill missing parts in the perceived object.

In digital 3D shape acquisition, we are faced with a similar problem: Most optical acquisition devices will produce incomplete and noisy data due to occlusions and physical limitations of the scanner. How can we obtain a complete and consistent representation of the 3D shape from this acquired data? One approach is to apply low-level geometric operations, such as noise and outlier removal filters [Tau95, JDD03, FDCO03, WPH+04] and hole-filling techniques based on smooth extrapolations, e.g., [DMGL02, VCBS03, Lie03, CDD+04]. These methods are successful in repairing small deficiencies in the data, but have difficulties with complex holes or when large parts of the object are missing. In such cases, trying to infer the correct shape by only looking at the acquired sample points quickly becomes infeasible. A common way to address this ill-posed problem is to use an explicit prior in the form of a carefully designed template model. The prior is aligned with the acquired data and holes are filled by transferring geometric information from the warped template. The cost of designing the template model is quickly amortized when a whole set of similar objects is digitized, as has been demonstrated successfully with human heads [BV99, KHYS02], and bodies [ACP03]. We extend this idea to arbitrary shapes by replacing a single, tailor-made template model with an entire database of 3D objects. This allows shape completion by *combining* geometric information from different context models.

To successfully implement such a system, we need to address the following issues: How can we extract models from the database that provide a meaningful shape continuation in regions of missing data? How can we compute and consistently evaluate local shape deformations that align the context models with the acquired data? How can we select among multiple database models the ones that provide the most adequate shape completion in different regions of the scan? And finally, how can we blend geometric information from different models to obtain a complete and consistent representation of the acquired object?

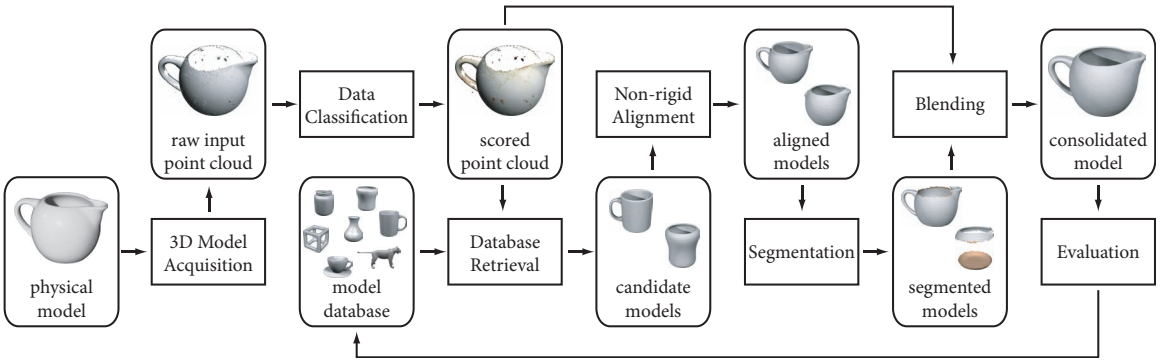


Figure 4.1: *Shape completion pipeline.* High-level overview of our context-based shape completion pipeline.

4.1 Related Work

Surface reconstruction from point samples has become an active area of research over the last few years. Early methods based on signed distance field interpolation have been presented by [HDD+92] and [CL96]. Voronoi-based approaches with provable guarantees where introduced by [ABK98] and [DG03], who extended their work in [DG04] to handle noisy input data. While these techniques can handle small holes and moderately under-sampled regions in the input data, they are less suited for data sets with large holes that often occur in 3D data acquisition. To address this problem, various methods for model repair based on smooth extrapolation have been proposed. [CBC+01] use radial basis functions to extract manifold surfaces from incomplete point clouds. [DMGL02] presented an algorithm that applies volumetric diffusion to a signed distance field representation to effectively handle topologically complex holes. An extension of this idea has been proposed by [VCBS03], who use partial differential equations to evolve the distance field, similar to inpainting techniques used for images. [SACO04] presented a system that preserves high-frequency detail by replicating local patches within the acquired 3D data set, as an extension of the 2D method proposed in [DCOY03].

Other approaches for shape completion are the triangulation-based method proposed by [Lie03] and the system based on finite elements presented in [CDD+04]. The underlying assumption in all of these methods is that an appropriate shape continuation can be inferred

from the acquired sample points only, using generic smoothness or self-similarity priors for missing parts of the model. Since the surface reconstruction problem is inherently ill-posed, the use of explicit template priors has been proposed by various authors. [RA99] presented a method to recognize and fit a parametric spline surface to acquired surface data. Template-based hole filling has also been used in [BV99, KHYS02, BMVS04, ACP03], where input data and morphable template model were represented as triangle meshes. These methods are well-suited for object classes with well-defined shape variability, where a single template model can be adjusted to fit the entire acquired data set. Our approach differs in that we are not assuming a priori knowledge of the underlying shape space, but try to infer automatically how to combine different context models that are retrieved from a database for the specific input data produced by the scan.

A central component of our system is a method for computing non-rigid alignments of database models with the acquired input data. [ACP03] and [SP04] presented alignment algorithms similar to ours that use an optimization framework to compute a smooth warping function. We extend this scheme to allow a quantitative comparison of the quality of the alignment across different models, which has not been a concern in previous methods. An interesting alternative has been proposed by [ASK+04], who introduced a probabilistic scheme for unsupervised registration of non-rigid shapes. Our system bears some resemblance to the shape modeling system proposed by Funkhouser et al. [FKS+04], where the user can create new models by cutting and pasting rigid parts of existing shapes retrieved from a shape database. The focus of their work is on creative design and interaction, while we concentrate on model repair and shape completion of surface scans.

Contributions

We define a shape similarity metric and corresponding non-rigid alignment method that allows to consistently evaluate the quality of fit of different context models. We present an automatic segmentation method that locally selects the best matching geometry, and a blending scheme that allows to combine contributions from different models, while preserving appropriate continuity constraints. Our algorithms are designed to work with uncertain data and can directly be applied to highly incomplete raw scanner output. We show how these methods can be integrated to yield a complete context-based shape completion pipeline.

4.1.1 Overview

Figure 4.1 gives an overview of our shape completion pipeline. 3D acquisition devices typically produce a set P of possibly noisy point samples $\mathbf{p}_i \in \mathbb{R}^3$ that describe (parts of) a 2D boundary surface of a 3D object. We assume that P is equipped with approximate normals, which are commonly provided by the scanner, or can be estimated directly from the point samples [HDD+92]. This input point cloud is pre-processed using multi-scale analysis to obtain a scalar confidence estimate that quantifies the consistency of each sample point with its neighboring samples. Subsequent stages of the pipeline will take these confidence weights into account to adapt the processing to the uncertainty in the acquired data. In the next stage, we retrieve a small set of candidate models from the database using a combination of multiple retrieval methods. The candidate models are then warped to match the shape of the input point cloud. To compute this non-rigid alignment we use an optimization process that balances geometric error, distortion of the deformation, and semantic consistency defined by a small set of feature correspondences. We segment the warped models into parts that best correspond to the input data based on a local shape similarity metric. Context information is propagated into regions of missing data, while continuously updating the alignment to ensure consistency between different context models. The segments are then combined using a geometric stitching technique that blends adjacent parts from different models to avoid

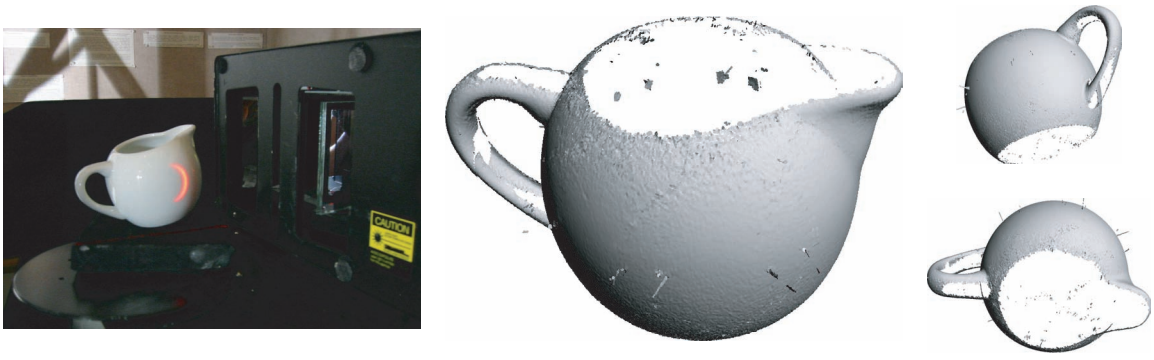


Figure 4.2: *Scanning a coffee creamer.* Left: Acquisition setup with Cyberware Desktop 3D Scanner 15 and physical model, right: Raw point cloud obtained from six range images in a single rotational scan. The spiky outliers are artifacts caused by specular reflections.

visual discontinuities along the seams. If a successful shape completion has been obtained, the final model is entered into the database for future use as a context model. The following sections discuss these individual stages in more detail, using the data set shown in Figure 4.2 to illustrate the complete shape completion pipeline. Note that in this model almost half of the surface geometry is missing due to occlusions, including the bottom and interior of the pot, as well as parts of the handle. Hole-filling techniques based on extrapolation would not be able to recover the correct shape from this data, but would require a significantly more complex scanning procedure with multiple scans of different poses of the object.

4.2 Data Classification

As illustrated in Figure 4.2, the acquired sample set P is inherently unreliable and cannot be treated as ground truth. Noise and outliers introduce uncertainty that needs to be considered when trying to reconstruct a consistent model. We compute per-point confidence estimates as a combination of two local geometry classifiers that analyze the distribution of samples within a small sphere centered at each sample point. The first classifier $c_i^\lambda \in [0,1]$ measures the quality of fit of a local tangent plane estimate at $\mathbf{p}_i \in P$, while the second classifier $c_i^\sigma \in [0,1]$ analyzes the uniformity of the sampling pattern to detect hole boundaries (see Appendix). The combination of both classifiers yields the confidence estimate $c_i = c_i^\lambda \cdot c_i^\sigma \in [0,1]$, which we evaluate at multiple scales by varying the size of the local neighborhood spheres. Similar to [PKG03], we look for distinct local maxima of c_i across the scale axis to automatically determine the appropriate scale at each sample point For all examples in this work we use ten discrete scale values, uniformly distributed between $2h$ and $20h$, where h is the minimum sample spacing of the scanning device. Figure 4.3 shows the results of this multi-scale classification. For more details, please refer to appendix a.

4.3 Database Retrieval

To transfer geometric information from the knowledge database to the acquired object, we need to identify a set of candidate models $M_1, \ldots, M_n$ that are suitable for completing the input data P. Database retrieval of 3D objects has gained increasing attention in recent years

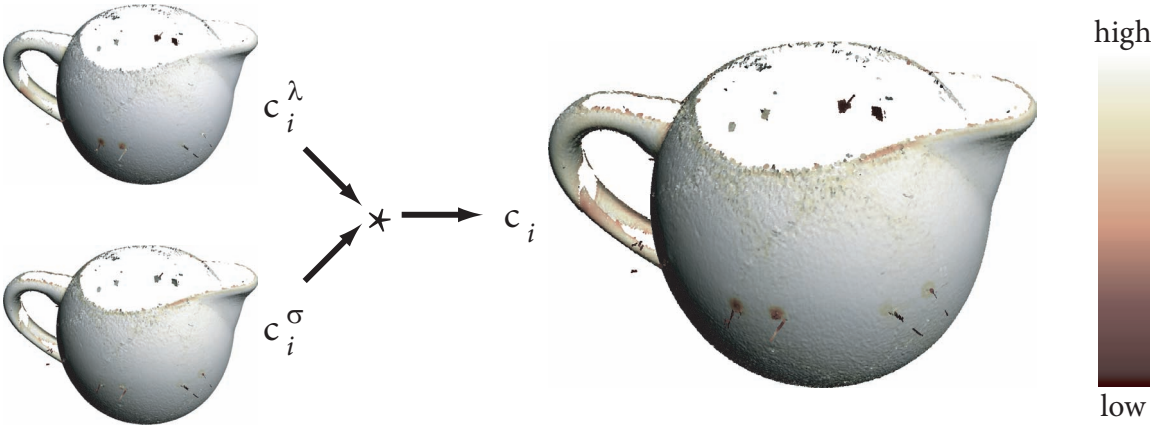


Figure 4.3: *Confidence measure for data samples.* Quality of fit estimate c_i^λ and local uniformity estimate c_i^σ are combined to yield the final confidence c_i.

and a variety of shape descriptors have been proposed to address this problem (see [TV04] for a recent survey). In our case the retrieval problem is particularly difficult, since we are dealing with noisy, potentially highly incomplete data. We thus rely on a combination of textual search and shape-based signatures, similar to [FKS+04]. We first confine the search space using a few descriptive keywords provided by the user. On this restricted set of models, we compute a similarity measure based on point-wise squared distances. We use PCA to factor out global scaling and estimate an initial pose. Then we apply the alignment method described in chapters 2 and 3 to optimize the rigid part α of the transform by minimizing the sum of squared distances $E(M, P)$ between acquired shape P and a database model M given as

$$E(M,P) = \sum_{i \in P} c_i \|\alpha(\mathbf{p}_i) - \mathbf{q}_i\|^2, \tag{4.1}$$

where $\mathbf{q}_i$ is the closest point on M from $\alpha(\mathbf{p}_i)$. Note that the squared distances are weighted by the confidence estimate c_i to avoid false alignments due to outliers or noisy samples. Equation 4.1 can by evaluated efficiently by pre-computing the squared distance field of each database model as described in detail in [MGPG04]. The residual of the optimization is used to rank the retrieved context models. Objects that align well with the acquired data (low residual) are likely candidates for a successful shape completion and will thus be given a high score. Figure 4.4 shows the results of the database retrieval for the coffee creamer

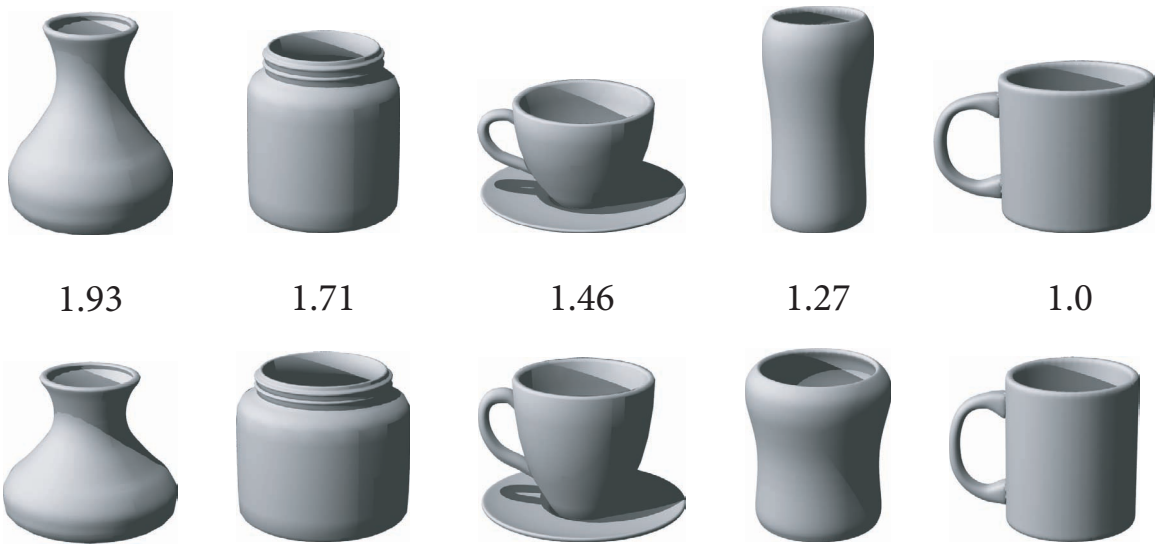


Figure 4.4: *Database search.* Models retrieved by a combination of geometric and textual query ordered from left to right according to decreasing relative alignment error. Top row: Original database models, bottom row: Aligned models after re-scaling.

example. Later in chapter 6, we present a randomized algorithm for efficient retrieval of partial matches from a database.

4.4 Non-rigid Alignment

The global similarity transform computed in the retrieval stage will in general not align the extracted context models exactly with the acquired data. We thus need to deform each model M before transferring shape information from M to P. The goal is to find a smooth warping function $T : M \rightarrow \mathbb{R}^3$ such that the deformed model $M' = T(M)$ matches P. At the same time we want the distortion of M induced by T to be as small as possible. The idea is that if only a small deformation is necessary to align a model with the acquired data set, then this model is more likely to provide a meaningful continuation of the shape in regions of missing data. We capture this intuition by defining a shape matching penalty function Ψ that combines the distortion of the transform and the geometric error between warped model and input data. The optimal warping function T can then be determined by minimizing this

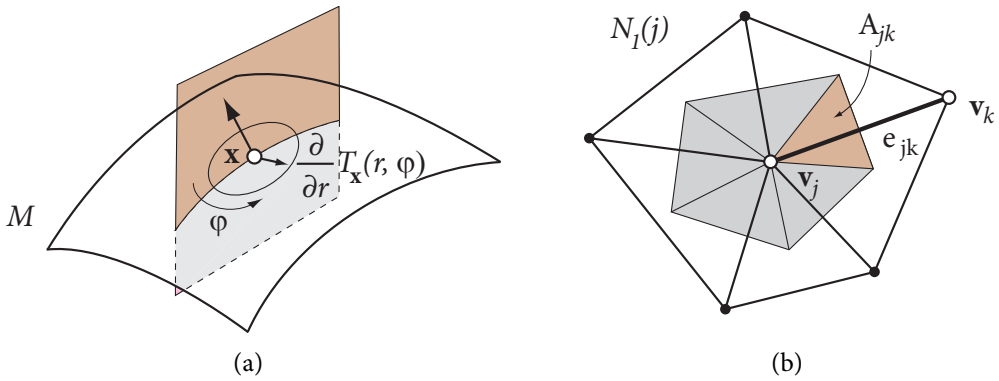


Figure 4.5: *Measuring Distortion.* Measuring distortion for a continuous surface (a) and in the discrete setting (b). The shaded region in (b) shows the area A_j of the restricted Voronoi cell of $\mathbf{v}_j$.

function [ACP03], [SP04]. Similar to the rigid alignment computed in Section 4.3, we use the residual of the optimization to evaluate the quality-of-fit of each database model. Hence the shape penalty function should be compatible across different context models to allow a quantitative comparison between the models. Additionally, we need to be able to do this comparison locally, so that we can determine which model best describes the acquired shape in a certain region of the object.

4.4.1 Distortion Measure

To meet the above requirements and make the penalty function independent of the specific discretization of a context model, we derive the distortion measure for discrete surfaces from the continuous setting (see also [Levo1]). Let S be a smooth 2-manifold surface. We can measure the distortion $\Phi(S, T)$ on S induced by the warping function T as

$$\Phi(S,T) = \int_S \int_\varphi \left(\frac{\partial}{\partial r} T_{\mathbf{x}}(r,\varphi) \right)^2 d\varphi d\mathbf{x}, \tag{4.2}$$

where $T_{\mathbf{x}}(r, \varphi)$ denotes a local parameterization of T at $\mathbf{x}$ using polar coordinates (r, φ). The inner integral measures the local distortion of the mapping T at $\mathbf{x}$ by integrating the squared first derivative of the warping function in each radial direction (see Figure 4.5 (a)). Since we represent database models as triangle meshes, we approximate T as a piecewise linear

function by specifying a displacement vector $\mathbf{t}_j$ for each vertex $\mathbf{v}_j \in M$. The angular integral in Equation 4.2 is discretized using a set of normal sections defined by the edges $\mathbf{e}_{jk} = \mathbf{v}_j - \mathbf{v}_k$, where $k \in N_1(j)$ with $N_1(j)$ the one-ring neighborhood of vertex $\mathbf{v}_j$. We approximate the first derivative of T using divided differences, which yields the discrete version of the distortion measure $\Phi(M, T)$ as

$$\Phi(M, T) = \sum_{j \in M} \sum_{k \in N_1(j)} A_{jk} \left(\frac{\mathbf{t}_j - \mathbf{t}_k}{|\mathbf{e}_{jk}|} \right)^2. \tag{4.3}$$

As shown in Figure 4.5 (b), A_{jk} is the area of the triangle defined by $\mathbf{v}_j$ and the Voronoi edge dual to $\mathbf{e}_{jk}$ in the Voronoi diagram of P restricted to $N_1(j)$. Note that $A_j = \sum_{k \in N_1(j)} A_{jk}$ is the area of the Voronoi cell of $\mathbf{v}_j$ restricted to M, hence the surface area of M is given as $A_M = \sum_j \sum_k A_{jk}$.

4.4.2 Geometric Error

Additionally, we define a geometry penalty function Ω that measures the deviation of the deformed model from the input sample. For two smooth surfaces S_1 and S_2, we can define the squared geometric distance of S_1 to S_2 as

$$\Omega(S_1, S_2) = \int_{S_1} d(\mathbf{x}, S_2)^2 d\mathbf{x}, \tag{4.4}$$

where $d(\mathbf{x}, S_2)$ is the shortest distance of a point $\mathbf{x} \in S_1$ to the surface S_2. To discretize this equation we represent the surface S_P defined by P as a collection of tangent disks attached to each sample $\mathbf{p}_i \in P$. The orientation of the disk is given by the normal at $\mathbf{p}_i$ and its radius is determined from the size of a local k-neighborhood, similar to [PKKG03]. We can then approximate the geometric error by summing up the area-weighted squared distance of each transformed vertex $\mathbf{v}_j + \mathbf{t}_j$ of M to the closest compatible point $\mathbf{q}_j$ on S_P, leading to

$$\Omega(P, M, T) = \sum_{j \in M} \omega_j A_j \|\mathbf{v}_j + \mathbf{t}_j - \mathbf{q}_j\|^2. \tag{4.5}$$

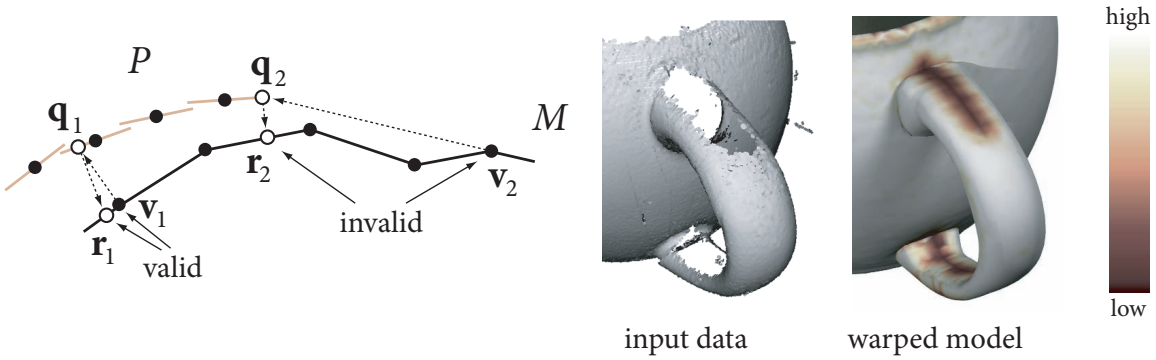


Figure 4.6: Correspondence weights v_j are determined using a bidirectional closest point search.

By compatible we mean that the normal of the tangent disk of $\mathbf{q}_j$ deviates less than 90^o from the normal of $\mathbf{v}_j$ to avoid matching front- and back-facing parts of both surfaces. The additional weight ω_j is defined as the product of two terms: The confidence estimate c_j of the sample point in P associated with $\mathbf{q}_j$, and a correspondence weight v_j that quantifies the validity of the correspondence between $\mathbf{v}_j$ and $\mathbf{q}_j$. Since P can be incomplete and M might contain parts that do not match with the acquired object, we need to discard vertices of M from contributing to the geometric error, if no valid correspondence with the samples of P can be established. We define the correspondence weight using a simple, but effective heuristic. Let $\mathbf{r}_j$ be the closest point on the surface of model M from $\mathbf{q}_j$. If $\mathbf{v}_j$ and $\mathbf{r}_j$ are close, then we have a strong indication that the correspondence is valid. We thus set $v_j = \exp(-\|\mathbf{v}_j - \mathbf{r}_j\|^2/h^2)$, where h is the average local sample spacing of P (see Figure 4.6).

4.4.3 Optimization

We combine the distortion metric Φ and the geometric error Ω to define the shape matching penalty function Ψ as

$$\Psi(P, M, T) = \alpha \cdot \Phi(M, T) + (1 - \alpha) \cdot \Omega(P, M, T), \tag{4.6}$$

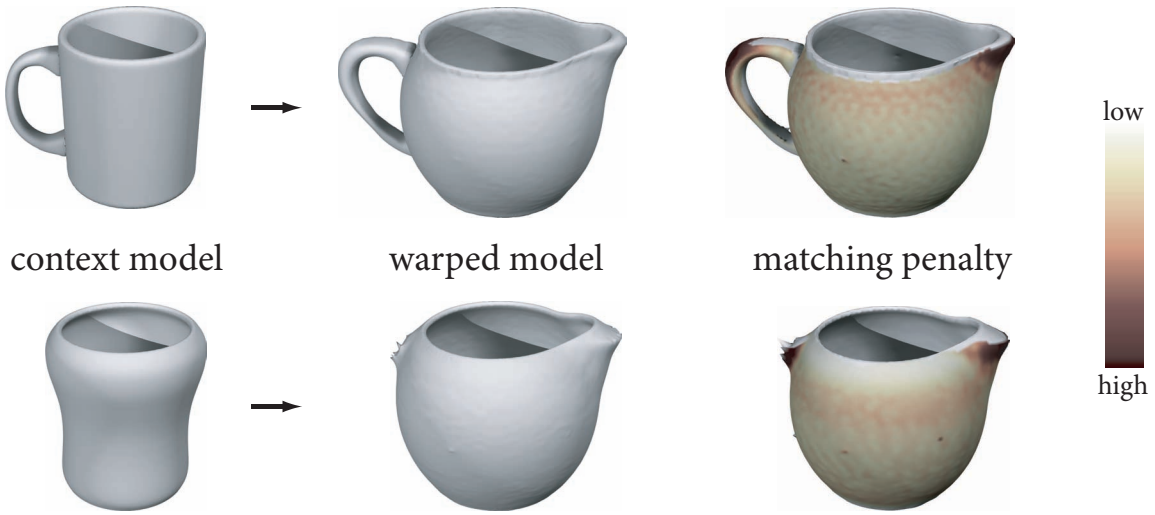


Figure 4.7: *Non-rigid alignment.* Points without valid correspondence are colored in gray in the images on the right.

where $\alpha \in [0,1]$ is a parameter that allows to balance distortion and geometric error. The warping function T is then computed iteratively by minimizing Ψ with respect to the unknown displacement vectors $\mathbf{t}_j$. This yields a sparse linear system of size $3n \times 3n$, where n is the number of vertices in the mesh, that we solve using a sparse matrix conjugate gradient solver. We use a multi-level optimization scheme similar to the method proposed by [ACP03] and later adopted by [SP04].

Figure 4.7 shows the two warped context models for the coffee creamer example.

Feature Correspondences. To avoid local minima in the optimization of Equation 4.6, we adapt the geometric penalty to include a small set $F \subset M$ of user-specified feature points. The user explicitly defines this set by selecting vertices of M together with corresponding points in P. The influence of each feature vertex $\mathbf{v}_j \in F$ can be controlled by scaling the weight ω_j in Equation 4.5. Explicit feature points are crucial for models for which the correct correspondence cannot be derived with the purely geometric approach of Section 4.4.2. An example is shown in Figure 4.14, where the semantics of each part of the models is clearly defined and needs to be observed by the warping function. Feature points also provide a mechanism for the user to control the non-rigid alignment for difficult partial matches as shown in Figure 4.15. Similar to [ACP03], we start the optimization with

high feature weights and strong emphasis on the smoothness term to obtain a valid initial alignment on a low resolution model. At higher resolutions we decrease the influence of the feature points and steadily increase α to about 0.9 so that the geometric error dominates in the final alignment.

4.5 Segmentation

After non-rigid alignment, we now need to determine how to compose the final model from different parts of the warped context models. In particular, we need to decide which model provides the most adequate shape continuation in regions of missing data. This decision is based on the matching penalty Ψ computed during the alignment stage, as it provides a measure of how well the deformed database models approximate the shape of the acquired object. We first compute a segmentation of the context models into patches that cover the input point cloud in regions of high data confidence. In the next section we will describe how to extrapolate geometric information from these patches to consistently fill in missing regions and obtain a complete model representation.

The initial segmentation is computed using an incremental region growing process as shown in Figure 4.8. Starting from a seed point $\mathbf{p}_i \in P$, we determine which model best matches the acquired data in the vicinity of that point by evaluating the matching penalty on a small patch around $\mathbf{p}_i$. The model M_k with the smallest local penalty will be our candidate for this region. We then successively expand this patch by adding triangles adjacent to the patch boundary. The incremental growth is stopped wherever we encounter a different model M_l with a smaller matching penalty, indicating that this model provides a better representation of the acquired shape in that region. To evaluate this criterion we require a mapping between candidate models, which we establish using the correspondence computed in the alignment stage. For each new candidate vertex $\mathbf{v}_j \in M_k$ adjacent to the current patch boundary, we look at the corresponding point $\mathbf{q}_j \in S_P$ used to compute the geometric error in Equation 4.5. We then find all vertices in M_l that were mapped to points in the vicinity of $\mathbf{q}_j$ and compare the matching penalty of these vertices with the one at $\mathbf{v}_j$. If a vertex with a smaller value is found, the triangle of $\mathbf{v}_j$ will not be added to the patch. We also discard this triangle if the correspondence weight v_j (see Section 4.4.2) is low, indicating

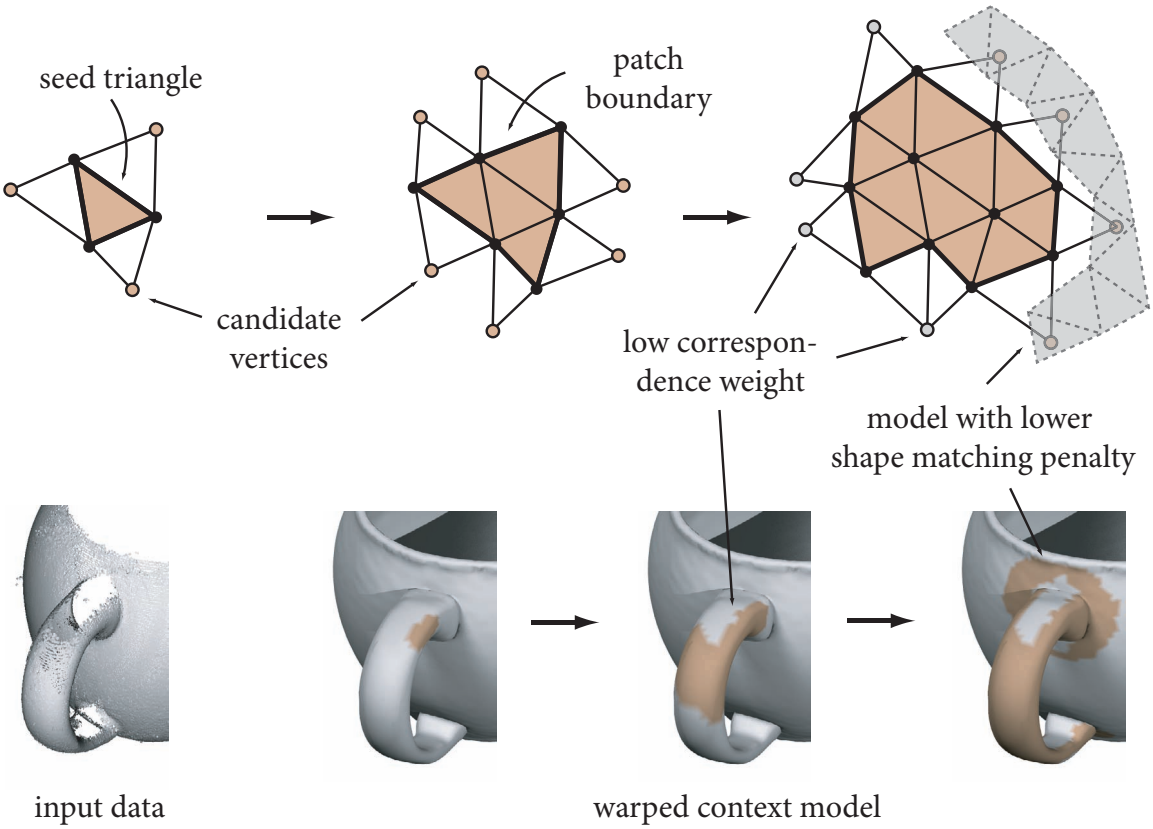


Figure 4.8: *Selecting from the warped models.* Incremental region growing for segmenting the context models.

that we have reached a hole boundary in the data.

The growth of a patch terminates as soon as no more candidate vertices can be added, as illustrated in Figure 4.8. We seed the patch creation by maintaining a priority queue of all samples $\mathbf{p}_i \in P$ with high confidence (we use the top 5% of samples in all our examples) that have not yet been visited. The queue is sorted according to decreasing confidence c_i such that samples with high confidence will be used first to initiate a new patch. The region growing is terminated once the queue is empty.

4.6 Blending

The segmentation of the warped context models provides a suitable representation of the scanned object in regions of high data confidence. To fill in parts where no reliable samples could be acquired, we need to extrapolate geometric information from the context patches. Filling holes is straightforward if only one candidate model covers the entire boundary of a hole. For example, the hole on the top of the creamer's handle is entirely enclosed by a single patch from the warped cup model (see Figure 4.8). We can thus simply copy and paste the corresponding surface part of that model. The situation is more complicated when two or more models meet at a hole boundary. Copying and pasting parts of each model will not yield a consistent surface, since the candidate models do not agree away from the input data. Even if a cut along an intersection curve can be found, an unnatural crease might be created that causes visual artifacts. To address this issue we propose an incremental blending method illustrated in Figure 4.9. Starting from the initial patch layout computed in the segmentation stage, we successively add samples to the input data by copying vertices from the patch boundaries of the segmented context models. These newly added sample points represent the continuation of the data surface at a hole boundary, as suggested by the best matching model in that region. We then re-compute the warping function for all retrieved database models to conform with this enhanced point set. Since the previous alignment provides a very good initial guess, only a few iterations of the optimization are required. After updating the alignment, we enlarge the context patches using the region growing algorithm described above. We repeat this procedure until the patch growing terminates, indicating that all the holes have been closed.

Stitching. The patch layout now provides the necessary pieces to compose the final model. We enlarge each patch by adding triangles along the patch boundary to create a smooth and seamless transition between adjacent patches. We achieve this blend by applying the same optimization as in the non-rigid alignment stage described in Section 4.4, except that we do not warp the models towards the input point cloud, but towards each other. Consider the example shown in Figure 4.10. As shown on the left, the two patches from the vase and the cup do not match exactly in the region of overlap. We therefore compute a warping function T_1 that aligns the vase with the cup and a warping function T_2 that aligns

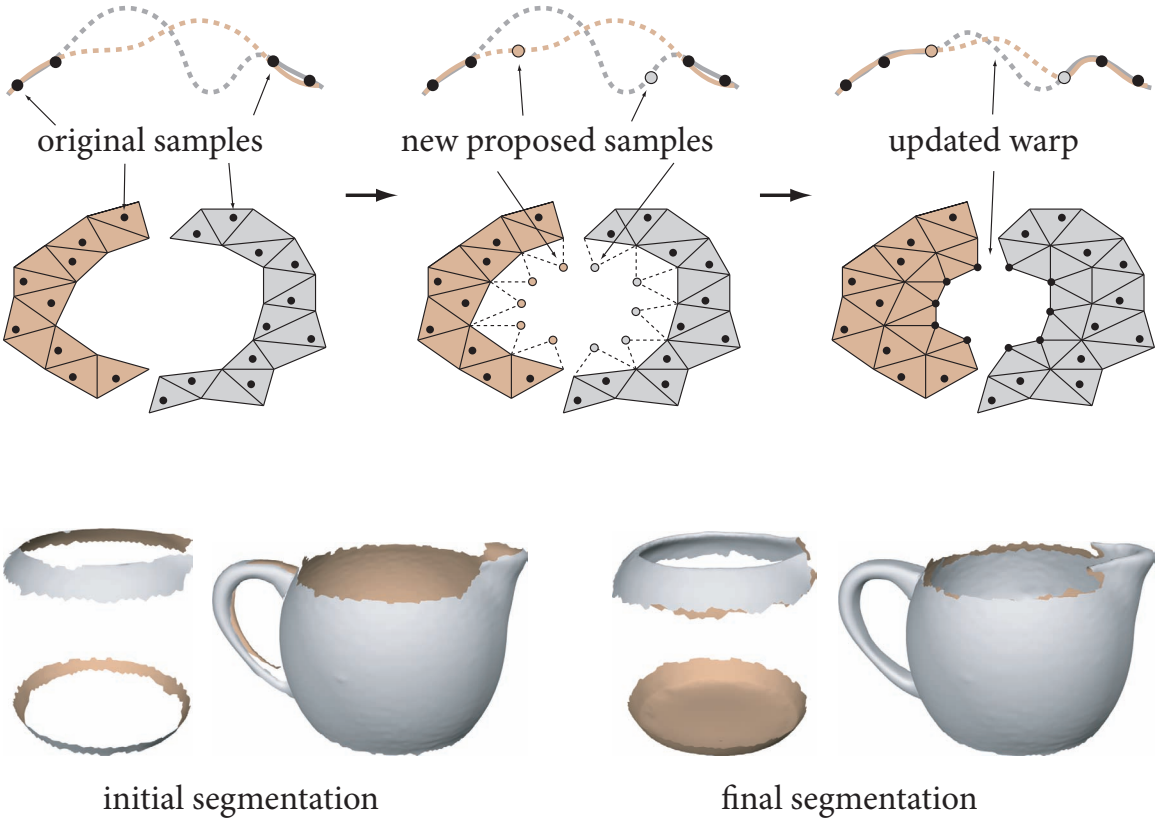


Figure 4.9: *Blending.* Top row from left to right: Two patches from different models meet at a hole boundary, new sample points are added from the model with lower shape matching penalty, both models are re-aligned with the enhanced point cloud and patches are enlarged. The top image shows a 2D illustration of the warped context models, the bottom image shows the current patches. Bottom row: Segmentation before and after blending. Back-facing triangles are colored.

the cup with the vase, and apply half of each transform to the corresponding model. A few iterations of this process create a conforming overlap region between the two patches. We then use the stitching method of [TL94] to obtain a single manifold surface.

4.7 Results and Discussion

We have tested our model completion pipeline on a number of acquired data sets with significantly different shape characteristics. All examples contain large, complex holes due to

Figure 4.10: *Stitching.* From left to right: Initial configuration, two intermediate steps of alignment, final stitched model.

occlusions, grazing angles, or specular reflections. Repairing these models without context information from the database would require substantial manual intervention using geometric modeling tools, since model completion techniques based on smooth extrapolation would not be able to create a consistent model. Figure 4.11 shows the final reconstruction of the coffee creamer example. Note how the characteristic features of the model are faithfully recovered and different parts of the two database models are blended in a natural way without any visual seams. The deformation of the context models even captures the spout, which is not present initially in any of the two models. However, in regions of insufficient input data, e.g., around the rim or at the top of the handle, the reconstructed model clearly exhibits characteristics of the context models. Apart from specifying optimization parameters and keywords for the textual search, this example requires no further user interaction. In particular, no feature points need to be specified to guide the alignment process. This leads to an overall processing time of less than two minutes.

Figure 4.11: *Reconstructed coffee creamer.* From left to right: Physical model, acquired data set, reconstructed model.

The acquired giraffe data set of Figure 4.14 has been completed with parts of the horse, camel, and lion. After computing the non-rigid alignment using 40 manually specified feature correspondences, the automatic segmentation and blending methods create a faithful reconstruction of the giraffe model. This example clearly demonstrates the advantages of *combining* context information from different models, since a satisfactory shape completion could not be obtained from any of the deformed context models alone. Figure 4.12 illustrates how shape completion is continuously simplified by enriching the database with already acquired and consolidated models. The two giraffes are completed using the model of Figure 4.14 as a context model. Even though the input data is noisy and consists of multiple, imperfectly aligned scans, a high-quality reconstruction is obtained.

A more complex example is shown in Figure 4.15. The input data is a single range image that contains large, complex holes due to occlusion. The two pillars shown in 3 and 4 are used as context models to repair the highly incomplete lower sections of the wall. The panels on the ceiling are completed successively using multiple iterations of our pipeline. The first panel on the lower left is repaired using a simple plane as a geometric prior. The consolidated panel is then used to fix the other panels in this arch. Once the whole arch is completed, it can be extracted to be used as a context model for the right arch. Note that the panels are not exact copies of each other, so simple copy and paste operations will not yield adequate results. User assistance is required to select appropriate parts in the data that can be used as context models for other regions, and to provide an initial alignment for those parts using four feature correspondences per piece. Interaction time for a trained user is less then half an hour, compared to multiple hours that would be required with standard modeling tools.

Additional Constraints. The shape matching penalty defined in Section 4.4 only considers low-level geometric properties to determine the warping function for non-rigid alignment. However, many models have specific high-level semantics that are not considered in this measure. For example, certain models exhibit symmetries that should be preserved by the warping function. As shown in Figure 4.16, we can adapt the alignment by adding appropriate constraints in the optimization. Another typical example is articulated models, where deformations that describe rotations around joints should be penalized significantly less than ones that result in a bending of rigid parts of the skeletal structure. This can be achieved by using a full kinematic description of the context models to adjust the matching

Figure 4.12: *Shape reconstruction from low-quality data.* From left to right: Physical model, acquired data set, reconstructed model.

penalty function accordingly.

Evaluation. A distinct advantage of our method is that it not only provides a final consolidated surface mesh, but also allows a local evaluation of the quality of the reconstructed model. We can easily identify regions where no adequate shape completion can be obtained, either because no valid correspondence between input data and context models can be established, or because the distortion of the warping function is too high to provide a meaningful shape prior for the acquired data. The zoom of the giraffe's head shown in Figure 4.13 depicts a case where our method does not recover a semantically correct shape, since the horns of the giraffe are not present in any of the context models and the data set is incomplete in this region. In such cases, the user either needs to acquire more data, enrich the database by providing more suitable context models, or manually edit the final model.

Limitations. Our context model retrieval relies on textual queries, which requires a well annotated shape database. This is particularly important for models that only provide partial completions in a certain region of the input data, but disagree greatly in other parts. Pre-segmentation of database models can simplify the retrieval of partially matching shapes, but requires a substantially more involved database search.

Similar to [ACP03] and [SP04] we control the distortion of the warping function when

Figure 4.13: *Evaluating the final completed shape.* The color-coding in the right image shows the shape matching penalty, where red color indicates insufficient surface completion due to invalid correspondence between input data and context models.

computing the non-rigid alignment, not the shape of the deformed model. Thus we can make no guarantees that the warped model is free of self-intersections. We discard a context model when we detect such a case, yet constraining the deformation to prevent self-intersections might be a more adequate solution. The distortion measure that controls the smoothness of the warping function is isotropic, i.e., penalizes distortion equally in all radial directions. If the acquired model has a high-frequency detail, e.g., a sharp crease, that is not present in the context model, the weight on the distortion measure needs to be low

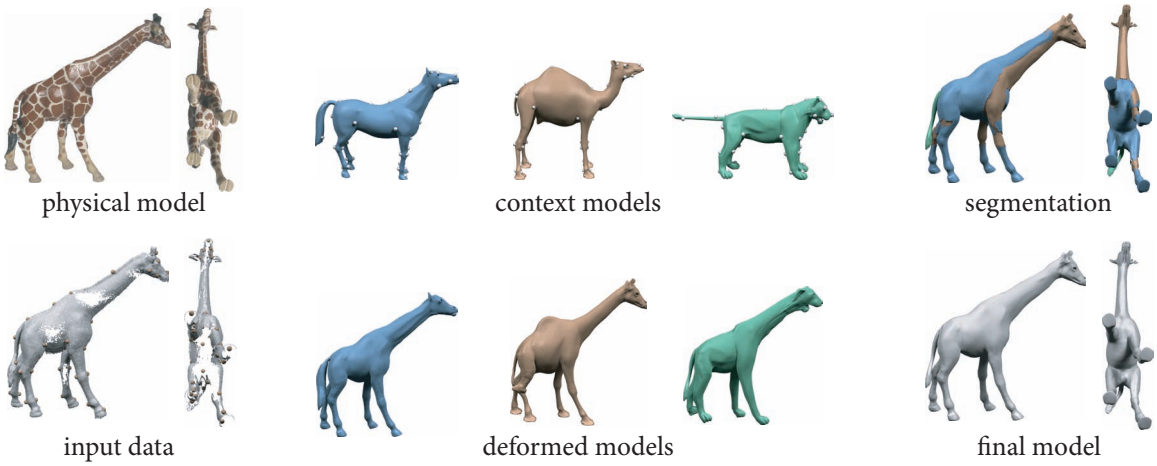


Figure 4.14: *Shape completion zoo.* Horse, camel, and lion are deformed, segmented and blended to yield the final shape of the giraffe.

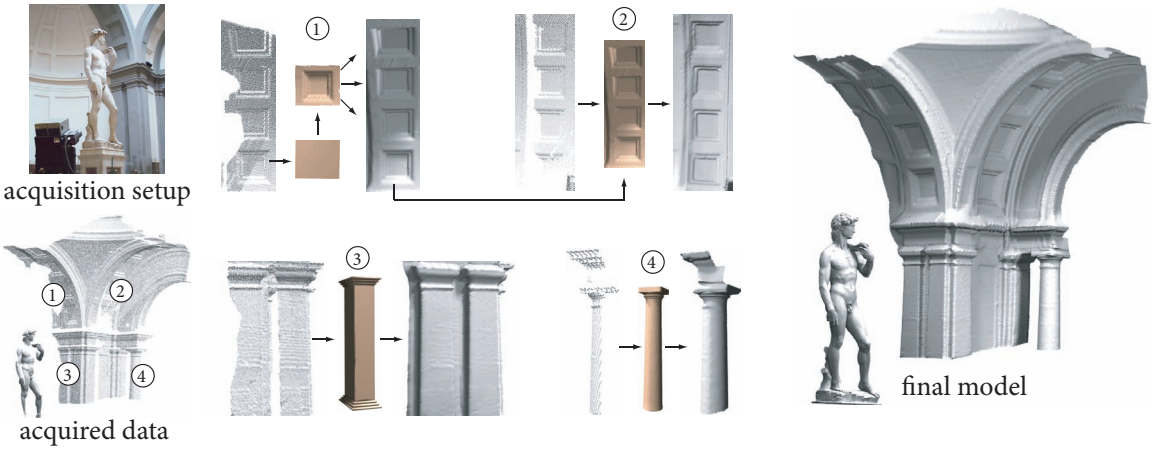


Figure 4.15: *Completion of a single range image acquired in the Galleria dell'Accademia in Florence.* Context models, shown in brown, are either retrieved from the database or extracted by the user from already completed parts of the model. The David model has been added for completeness.

(i.e., α has to be close to one in Equation 4.6), so that the warped context model can be aligned to this geometric feature. This, however, will also pick up noise present in the input data, as can be observed in Figure 4.15. A solution could be to design an anisotropic shape matching penalty that locally respects that characteristics of the input geometry, similar to anisotropic low-pass filters used in data smoothing. The blending method of Section 4.6 requires consistent topology of the context models in regions where two or more models

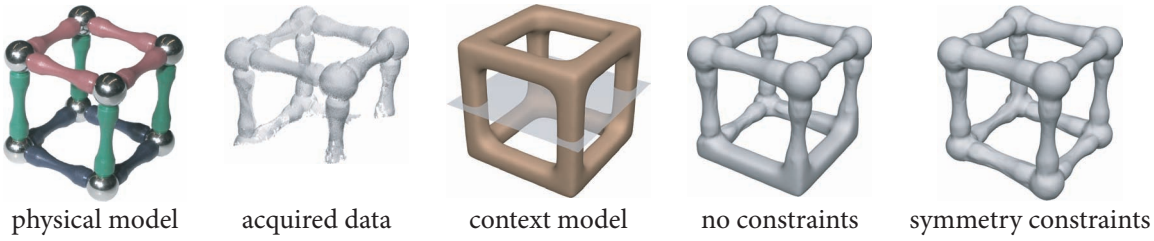


Figure 4.16: *Symmetry constraints for shape completion.* Symmetry constraints yield a semantically more adequate shape completion. The warping function for the model on the right has been constrained to be symmetric with respect to the semi-transparent plane shown in the center.

are blended. We detect topological mismatches from inconsistencies in the correspondence between different models, and exclude the model with higher shape matching penalty from the blending stage in this region. We can give no guarantees, however, that this heuristic always produces the correct shape topology. We thus also allow the user to manually discard individual models, if the topology is inconsistent, which provides more explicit control of the semantics of the consolidated shape.

4.8 Summary

In this chapter, we presented an example-based shape completion framework for acquired 3D surface data. Shapes from the database are deformed to fit the data. We employ an adaptive segmentation of the warped context models, which can then be blended consistently using incremental patch growing and continuous re-alignment, to yield the final consolidated shape representation. Our method is robust against noise and outliers and provides a quantitative evaluation of the quality of the produced output model.

So far in this thesis, we demonstrated how to quickly and efficiently acquire geometry of an object by scanning it from multiple directions and stitching the scans togethers. Regions of missing data are filled using geometric priors from database shapes. We also introduced several algorithms that are also applicable to a large variety of other geometry processing applications. Some of our techniques have already been used by others for solving related problems [Kol05, HFG+06, PHYH06, PGSQ06, BGK06, LBG06, EDK06, HSW06, FS06].

In the next part of the thesis, we investigate algorithms for analyzing global properties of 3D geometry.

Part II

Shape Analysis

In the first part of the thesis, we proposed several improvements for the shape acquisition pipeline — a branch and bound approach for determining a rough initial alignment for a pair of shapes, Gauss-Newton optimization for finer alignment of the shapes, and finally using geometric priors in the form of a model database to fill in regions of missing data. In short, we can quickly and efficiently scan an object in multiple sections, and stitch the scans together to get a consistent geometric representation of the object. Subsequently we can use state-of-the-art modeling tools to edit and deform existing models to generate novel 3D geometry.

At this stage we have little high level understanding of the underlying shape. In most cases, lack of high level shape understanding, forces us to store lot of redundant data and we end up processing the data inefficiently. Gaining a better understanding of the shapes, not only gives us a more compact shape representation, but more importantly, enables us to perform geometry processing in time proportion to the complexity of the model and independent of the resolution of the model. As we will see in subsequent chapters, the understanding we gained in the previous chapters helps us in our subsequent goal of shape analysis.

5

Partial and Approximate Symmetry Detection

Beauty depends on size as well as symmetry.

— *Aristotle*

In this chapter, we present a new algorithm [MGP06] that processes geometric models and efficiently discovers and extracts a compact representation of their Euclidean symmetries. Not surprisingly many of the techniques from previous chapters are used since symmetry detection is essentially partial shape registration of a shape with itself modulo the identity transform.

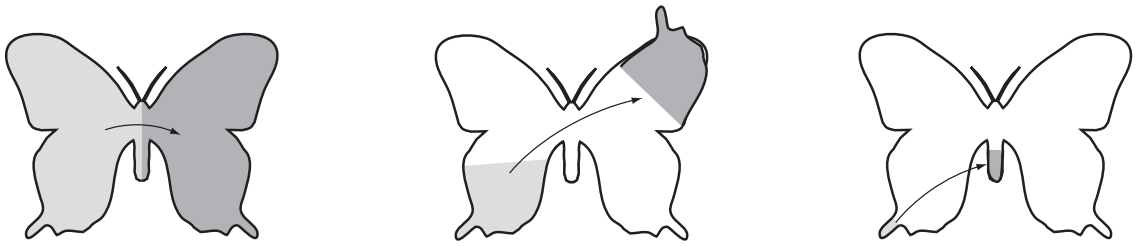
5.1 Introduction

Symmetry is an essential and ubiquitous concept in nature, science, and art. For example, in geometry, the Erlanger program of Felix Klein [Kle93] has fueled for over a century mathematicians' interest in invariance under certain group actions as a key principle for understanding geometric spaces. Numerous biological, physical, or man-made structures exhibit symmetries as a fundamental design principle or as an essential aspect of their function. Whether by evolution or design, symmetry implies certain economies and efficiencies of structure that make it universally appealing. Symmetry also plays an important role in

Figure 5.1: *Symmetry detection on a sculpted model.* From left to right: Original model, detected partial and approximate symmetries, color-coded deviations from perfect symmetry as a fraction of the bounding box diagonal.

human visual perception and aesthetics. Arguably much of the understanding of the world around us is based on the perception and recognition of shared or repeated structures, and so is our sense of beauty [Tho61].

In this chapter we present a novel method for detecting meaningful symmetries in digital 3D shapes. We understand symmetry as invariance under a set of transformations — in our case translation, rotation, reflection, and uniform scaling, the common generators of the Euclidean group. The figure below shows a 2D illustration. As can be seen in this example, symmetries or congruences that are quite apparent to us can be approximate and occur at different scales. Our goal is to define an algorithm that extracts (partial) symmetries at all scales, including approximate or imperfect symmetries of varying degree. This allows the user to select the subset of symmetries that are most meaningful for a specific application. Examples include scan registration and alignment, shape matching, segmentation and skeleton extraction, compression, advanced modeling and editing, and shape database retrieval.



reflection | reflection + rotation + translation | scale + rotation + translation

To achieve this goal, we separate the symmetry computation into two phases: In the first

step, we compute simple local shape descriptors at a selected set of points on the shape. These descriptors are chosen so that they are invariant under the group actions of interest. We use these local descriptors to pair up points that could be mapped to each other under a candidate symmetry action. We think of each such pair as depositing mass, or voting, for a specific symmetry in the transformation space of interest. In this space, pairs with similar transformations form clusters that provide evidence for the corresponding symmetry relation.

In the second step we use a stochastic clustering algorithm to extract the significant modes of this mass distribution. Since the mapping to transformation space does not preserve the spatial coherence or structure of samples on the input shape, we verify whether a meaningful symmetry has been found by checking the spatial consistency of the extracted subparts of the surface. Our clustering method provides the necessary surface correspondences, since every point mass in transformation space corresponds to a candidate pair of points in the spatial domain. Thus only a small set of candidates samples needs to be considered when detecting and extracting symmetric surface patches, avoiding a costly quadratic spatial search over the whole input data set.

This separation into two stages is crucial for the effectiveness of our algorithm. The underlying observation is the following: given a proposed symmetry relation, it is simple and efficient to verify whether this specific symmetry is present in the model; we just need to apply the symmetry transform and check whether the model is mapped onto itself, or a sub-part of the model is mapped to a corresponding sub-part. However, the number of all potential mappings is by far too large to do an exhaustive search. Therefore, we first accumulate statistical evidence for which symmetries are present via our clustering in transformation space. Only if this evidence is sufficient do we perform spatial verification to check whether a specific symmetry is actually valid. Thus the complexity of symmetry extraction depends primarily on the number and size of relevant symmetries present in the model and not on the complexity of the model itself or that of the underlying symmetry group. As part of our approach, we can provide a quantitative measure on the "exactness", or saliency, of a symmetry relation, which allows the user to control the degree of perfection in the extracted symmetries. In addition, by specifying the size of the set of local shape descriptors, the user can trade accuracy for computational efficiency. While fewer samples are sufficient

for detecting large global symmetries, small partial symmetries require a significantly denser sampling.

The final output of our algorithm is a "symmetry graph" of the object, which encodes the significant symmetries of the object, each described by a patch pair and the corresponding transformation between them. For objects that contain regular repeated structures, like windows or doors in architectural models, we can recover the symmetries of the repetition pattern through a basis reduction algorithm. This in effect leads to a sparser and more informative symmetry graph that contains only fundamental symmetry generators and avoids encoding separately symmetries that are just products of already recorded symmetries. This kind of repeated pattern discovery can be useful in consistent mesh editing applications.

5.2 Related Work

The problem of symmetry detection has been extensively studied in numerous fields including visual perception, computer vision, robotics, and computational geometry. Early methods concentrated on finding perfect symmetries in 2D or 3D planar point sets [Ata85], [WWV85]. Since the restriction to exact symmetries limits the use of these methods for real-world objects, Alt et al.[AMWW88] introduced a method for computing approximate global symmetries in 3D point sets, but the complexity of the algorithm makes it impractical for large data sets. Zabrodsky et al. [ZPA95] formalized the notion of approximate symmetry by expressing symmetry as a continuous feature. Sun et al. [SS97] proposed to examine the correlation of the Gaussian image to recover global reflective and rotational symmetries. Kazhdan and co-workers [KCD+02] introduced a shape descriptor that concisely encodes global reflective symmetries. Later they extended this work to rotational symmetries and used it for shape retrieval for database matching in [KFR04b].

Our method bears some similarity to the Hough transform, a popular feature extraction method mainly used in image processing [Hou59]. Starting from a set of sample points obtained using edge detection, the method repeatedly selects small subsets of these samples to estimate the parameters of the feature curve. Analogous to our approach, votes cast by all of these estimates are accumulated and the final feature curve is extracted based on the majority of votes. Recently ideas based on the Hough transform have been used by [LE06]

to detect reflective and rotational symmetries in images.

The RANdom SAmple Consensus (RANSAC) method proposed by Fischler and Bolles [FB81] is an algorithm for robust model fitting for data containing many outliers. In the context of shape matching the basic idea is to choose a random set of corresponding samples on the query and target shapes, apply the global transformation induced by these samples, and evaluate the matching error between the two shapes. If sufficiently many transformations are explored in this way, the relevant symmetries can eventually be determined. Since the evaluation of the matching error requires costly spatial proximity tests, geometric hashing [LW88] pre-computes all possible alignments by densely sampling the space of transformations and storing the resulting shape distribution in a hash grid. Gal and Cohen-Or [GCO06] recently presented an effective method for shape matching based on this idea. Their algorithm computes local shape descriptors that are grouped to form salient shape features. Using an empirical saliency measure, shape features are then used to pre-compute a geometric hash table that allows efficient partial matching.

While sharing some similarities, our method is fundamentally different from both geometric hashing and RANSAC. We avoid the costly exhaustive search of the former by computing the matching error of a transformation only *after* we accumulate sufficient evidence for a symmetry. At the same time our method requires minimal storage, in contrast to geometric hashing, where hash tables of up to 3.5 GBytes have been reported for complex geometric shapes [GCO06].

Contributions

We propose a new algorithm for pairing sample points on 3D shapes with compatible local descriptors to generate a distribution in transformation space whose peaks capture relevant symmetries of the object. We show how a stochastic clustering algorithm over this distribution detects potential symmetry candidates, and provide a surface patching method that extracts a reduced symmetry graph from the extracted clusters. Our algorithms can be applied to 3D models of different shape characteristics and representations. Memory requirements are minimal and the computation is output-sensitive in the sense that its complexity depends mainly on the number and extent of symmetries actually present in the object. In addition, we provide theoretical bounds on the success rate of our algorithm as a function

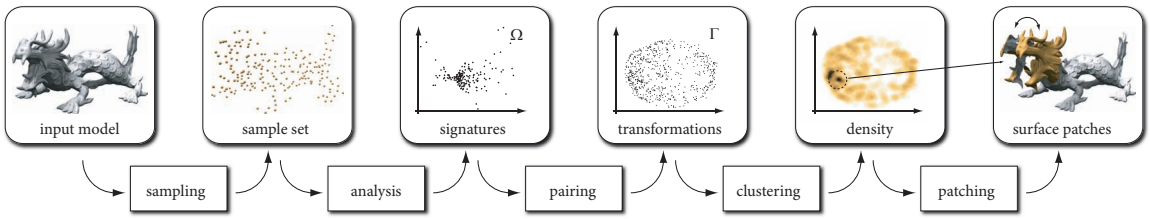


Figure 5.2: *The symmetry extraction pipeline.* Sampling yields a set P of surface points. For each $\mathbf{p}_i \in P$ a local signature is computed. Points $\mathbf{p}_i, \mathbf{p}_j$ with similar signatures are paired and a point in transformation space Γ is computed mapping the local frame of $\mathbf{p}_i$ to the one at $\mathbf{p}_j$. Clustering in Γ yields subsets of P that remain invariant under a certain transformation, which can be extracted using spatial region growing.

of the number of initial samples selected. These results indicate that the algorithm can be effective even for very large models that cannot fit in main memory.

5.3 Overview

We first give some intuition for our method by looking at the 2D example shown in Figure 5.3, where the goal is to detect reflective symmetries of the butterfly. Any pair of points $(\mathbf{p}, \mathbf{q})$ on the boundary of the model defines a unique reflection with respect to the bisector line through $(\mathbf{p}+\mathbf{q})/2$ with normal direction $\mathbf{p}-\mathbf{q}$. Hence such a pair can be understood as evidence for the existence of this specific reflective symmetry. By looking at all such pairs we can accumulate this evidence and extract the relevant symmetry relation(s). Only if many point pairs agree on (roughly) the same reflection line, do we have reason to believe that the corresponding symmetry is truly present in the model. Thus we can detect potential symmetries by looking at clusters of points in the space of transformations Γ, where each point corresponds to a specific reflection line. However, as shown in the illustration, the evidence of a single point pair is only reliable if the local geometry around the points is faithfully mirrored by the reflective transformation. This observation will allow us to significantly prune the set of all point pairs and avoid an exhaustive computation on a quadratic number of point pairs.

Since the mapping to Γ does not incorporate the spatial position of surface samples,

pairs from unrelated parts of the object can be mapped to the same point in transformation space. Thus in a second phase we extract spatially coherent components of the model that are invariant under the extracted symmetry transformations. Using the point pair correspondences present in the cluster, we perform an incremental region growing algorithm to verify a specific symmetry. Figure 5.2 gives a high-level overview of our symmetry extraction pipeline. The following sections will elaborate on the individual stages and provide details of our approach.

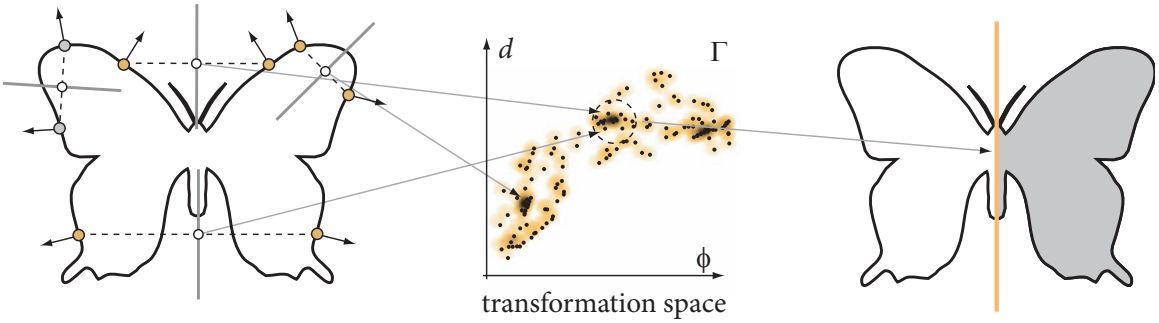


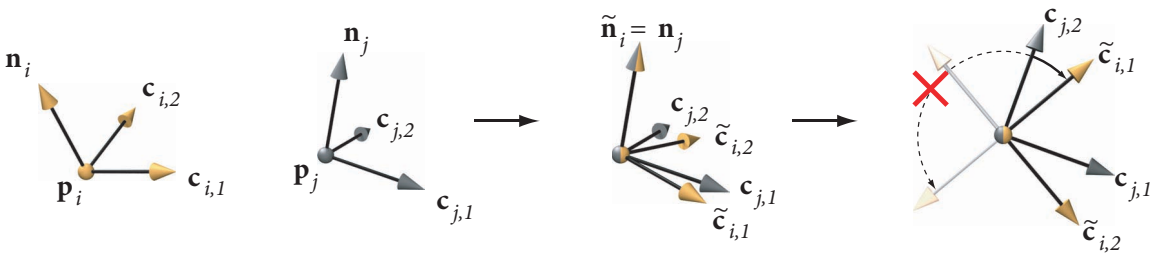
Figure 5.3: *Illustration of symmetry detection for reflections.* Every pair of points defines a symmetry line l that can be described by a distance d and an angle ϕ. Multiple points clustered in a small region in transformation space provide evidence of a symmetry. The pair on the top left is discarded due to normal inconsistency.

5.4 Signatures and Transformations

We consider the Euclidean transformation group generated by translations, rotations, reflections, and uniform scalings. Our goal is to find parts of a given 3D shape that are invariant under transformations in this symmetry group or some lower-dimensional subgroup.

In order to apply the ideas sketched above, we need to compute the transformation $\mathbf{T}_{ij}$ that maps a point $\mathbf{p}_i$ on the surface of the model to another point $\mathbf{p}_j$. While point positions are sufficient for defining a unique plane of reflection as in the example above, we cannot determine all degrees of freedom of a general Euclidean transform from the spatial positions alone. We therefore compute geometry signatures at each sample point $\mathbf{p}_i$ based on the concept of normal cycles [CSM03]. We apply the algorithm proposed in [ACSD+03] to

approximate the curvature tensor at $\mathbf{p}_i$ within a sphere of radius r and compute integrated principal curvatures $\kappa_{i,1} \leq \kappa_{i,2}$ and principal directions $\mathbf{c}_{i,1}$ and $\mathbf{c}_{i,2}$. The radius r should be on the order of the local sample spacing to achieve sufficient averaging when computing the curvature tensor and avoid a strong dependence on the specific location of the sample points.



The principal directions define a local frame $(\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \mathbf{n}_i)$, with normal vector $\mathbf{n}_i = \mathbf{c}_{i,1} \times \mathbf{c}_{i,2}$. We orient this frame as a right-handed coordinate frame that aligns with the outward pointing surface normal by flipping signs of the appropriate vectors if necessary. In order to obtain a canonical rotational component $\mathbf{R}_{ij}$ of the transformation $\mathbf{T}_{ij}$ we first align the two normals along their common plane and then pick the smaller of the two rotations around the normal that aligns to one of the two possible choices of orientation in tangent space. The uniform scale component of $\mathbf{T}_{ij}$ is estimated from the ratio of principal curvatures as $s_{ij} = (\kappa_{i,1}/\kappa_{j,1} + \kappa_{i,2}/\kappa_{j,2})/2$, the translation is computed as $\mathbf{t}_{ij} = \mathbf{p}_j - s_{ij}\mathbf{R}_{ij}\mathbf{p}_i$. For a given pair $(\mathbf{p}_i, \mathbf{p}_j)$ we thus obtain a point in 7-dimensional transformation space Γ as $\mathbf{T}_{ij} = (s_{ij}, R^x_{ij}, R^y_{ij}, R^z_{ij}, t^x_{ij}, t^y_{ij}, t^z_{ij})$, where $R^x_{ij}, R^y_{ij}, R^z_{ij}$ are the Euler angles derived from $\mathbf{R}_{ij}$ and $\mathbf{t}_{ij} = [t^x_{ij} \quad t^y_{ij} \quad t^z_{ij}]^T$. In order to handle reflections, we also compute the transformation obtained when reflecting the model about an arbitrary but fixed plane.

5.4.1 Point Pruning

A differential surface patch at umbilic points, i.e., those for which $\kappa_{i,1} = \kappa_{i,2}$, is invariant under rotations around the surface normal. Pairs involving such points and their signatures do not define a unique transformation, but trace out curves in transformation space, which may quickly camouflage meaningful symmetry clusters. To avoid clutter in transformation space, we discard these points from the sample set, i.e., we only consider points on the surface

with distinct principal curvatures (and hence stable principal directions), which give rise to a unique transformation when paired with another compatible point. Apart from making the symmetry clustering more robust, point pruning has the additional advantage of reducing computation time. We obtain the adaptive sample set by applying a threshold $\gamma < 1$ on the ratio of curvatures: $\mathbf{p}_i \in P$, if $|\kappa_{i,1}/\kappa_{i,2}| < \gamma$. We use $\gamma = 0.75$ for all examples in this work.

5.4.2 Pairing

Given the reduced set of surface samples P and their signatures, we can now compute transformations for pairs of points in P. We select a random subset $P' \subset P$ and find all pairs $(\mathbf{p}', \mathbf{p})$ with $\mathbf{p}' \in P'$ and $\mathbf{p} \in P$ that provide evidence for a symmetry relation. In the Appendix we give theoretical bounds on the size of P and P' required to successfully find symmetries of a certain size.

As indicated above, the evidence of a selected point pair for a specific symmetry relation is only reliable, if a local surface patch around each point is invariant under a transformation from the considered symmetry group G. In the 2D illustration of Figure 5.3, for example, we can reject a pair, if the curvature estimates at both points differ too much, since curvature is invariant under reflection. To obtain an efficient pairing algorithm we map all samples to a signature space Ω and use the metric of that space to estimate the deviation from perfect invariance. Only point pairs that are close in Ω are considered as suitable candidates for a local symmetry relation, which avoids an exhaustive computation of a quadratic number of point pairs.

For the full 7-dimensional Euclidean group in 3D, the mapping from P to $\Omega_7 = \mathbb{R}$ is given as $\sigma_7(\mathbf{p}_i) = \kappa_{i,1}/\kappa_{i,2}$, since uniform scaling, rotation, and translation leave the ratio of principal curvatures unchanged. The sub-index 7 indicates the dimension of the symmetry group. For purely rigid transforms, we define $\sigma_6(\mathbf{p}_i) = (\kappa_{i,1}, \kappa_{i,2})$ with $\Omega_6 = \mathbb{R}^2$. We can now for a given sample $\mathbf{p}_i \in P'$ determine all suitable partners in P by performing a range query in Ω. Using standard spatial proximity data structures, e.g., a *kd*-tree, we can perform pairing in $O(n' \log n)$ time, where $n = |P|$ and $n' = |P'|$. If only reflections and/or translations are considered, we can additionally reject pairs based on the orientation of the local frames, as illustrated in Figure 5.3.

Figure 5.4 shows that pruning not only reduces the complexity of the clustering algorithm, but, even more importantly, avoids clutter in transformation space. By focusing only on locally consistent symmetry pairs, meaningful clusters are stably detected in Γ.

5.5 Clustering

The pairing computed in the previous stage provides us with a set of transformations that map local surface patches onto each other. Each pair thus provides evidence for a symmetry relation at the level of the local sample spacing. To extract meaningful symmetries at larger scales we need to accumulate this local evidence, i.e., find groups of pairs with a similar transformation that correspond to symmetric subsets of the model surface. This requires the definition of a distance metric in Γ, which is non-trivial, since scaling, rotation, and translation need to be combined in a single metric. We follow the approach of [ABD+00] and define the norm of a transformation $\mathbf{T} = (s, R_x, R_y, R_z, t_x, t_y, t_z) \in \Gamma$ as the weighted sum $\|\mathbf{T}\|^2 = \beta_1 s^2 + \beta_2(R_x^2 + R_y^2 + R_z^2) + \beta_3(t_x^2 + t_y^2 + t_z^2)$. The weights β_i allow to adjust the relative influence of the individual components of the transformation. In all our examples we set these weights so that a rotation by 180 degrees corresponds to a displacement of half the bounding box diagonal and a scaling factor of 10. A metric for Γ can then be derived as $d(\mathbf{T}, \mathbf{T}') = \|\mathbf{T} - \mathbf{T}'\|$, where the subtraction is component-wise, see also [HPR04] for a detailed discussion.

5.5.1 Mean-Shift Clustering

If the symmetries in the model are perfect (and the sampling includes point pairs that are perfectly symmetric), then all pairs of the same (discrete) symmetry relation map to a single point in Γ. Many real-world objects exhibit approximate symmetries, however, and the sampling will not be precisely symmetric in general. We thus need a method to find clusters in transformation space. When looking at the distribution of points in Γ, we immediately see that standard parametric clustering methods, such as k-means clustering, are not suitable for our purposes. In general we have no a priori knowledge on the number of (partial) symmetries of the input model, i.e., selecting k would be difficult. Furthermore, clusters are not necessarily isotropic, especially for approximate symmetries like the ones shown in

Figure 5.1. A more suitable clustering method is *mean shift clustering*, a non-parametric method based on gradient ascent on a density function ρ [CM02]. Mean shift clustering has also been used in [JT05] for skinning mesh animations and in [TSM05] for 3D motion estimation. This density function is defined as a sum of kernel functions K centered at each

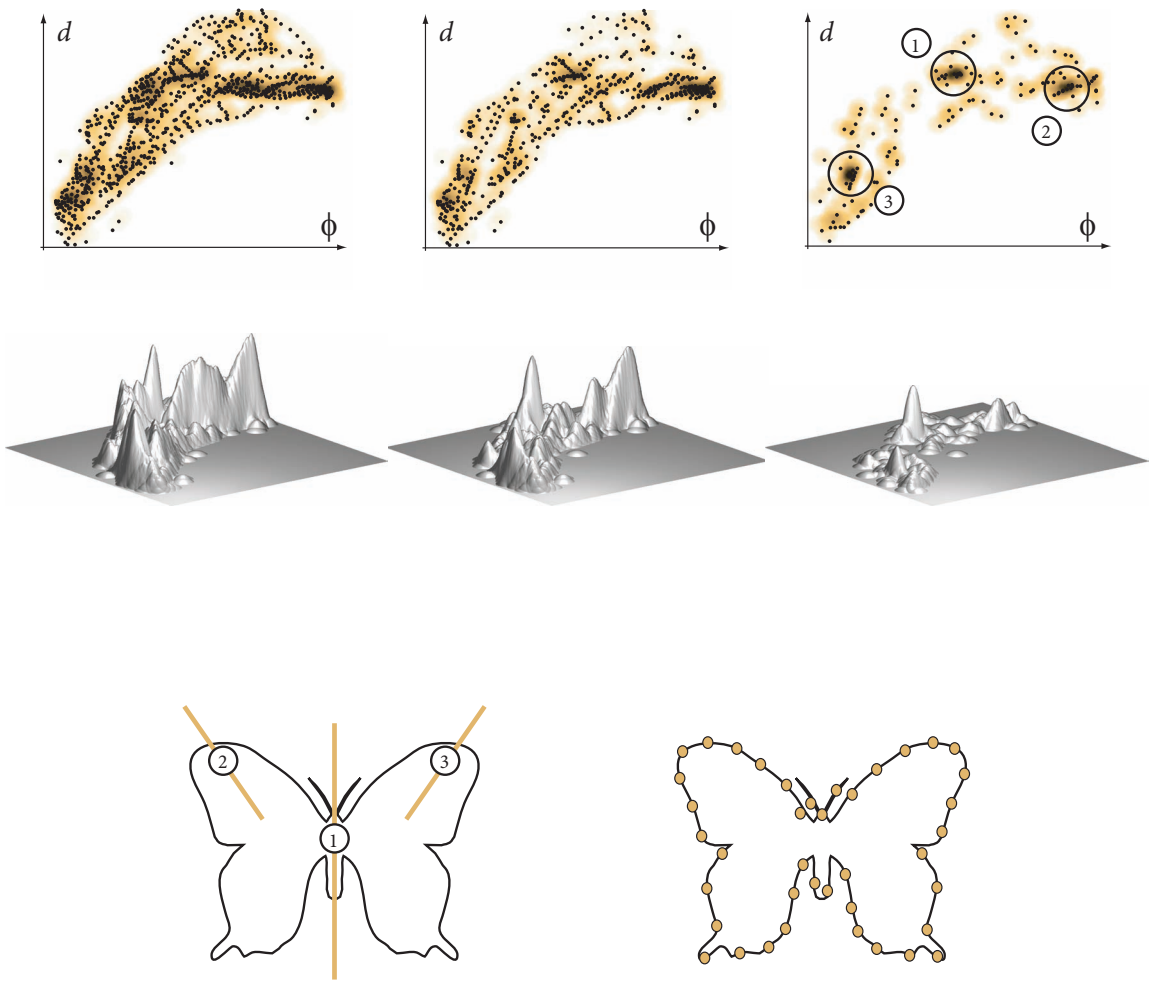


Figure 5.4: *Point pair pruning.* 40 samples on the butterfly lead to $\binom{40}{2} = 780$ points in transformation space. Pruning based on curvature reduces the set to 503 points, while additionally normal-based pruning yields 138 points. The density plots show how the meaningful symmetry clusters become significantly more pronounced.

point $\mathbf{T}_i$ in Γ as

$$\rho(\mathbf{T}) = \sum_i K(\|\mathbf{T} - \mathbf{T}_i\|/h).$$

We use the radially symmetric Epanechnikov kernel with bandwidth h as suggested in [CM02]. The significant modes of ρ are determined using gradient ascent. All points that flow into a local maximum of sufficient height are considered samples of a significant cluster C_k. The corresponding symmetry transformation $\mathbf{T}_k$ is then defined by the cluster's maximum. Essentially, the algorithm can be understood as a voting scheme: Every point pair votes for the symmetry relation that has been extracted from its local frames. If many votes are cast for the same symmetry, a local peak is created in the accumulated density function. For more details on mean-shift clustering we refer to [CM02].

5.6 Verification

A significant mode detected by the mean-shift clustering algorithm does not necessarily correspond to a meaningful symmetry. Since the spatial relation of sample points is lost during the mapping to transformation space, sample pairs from uncorrelated parts of the object can accumulate to form discernible clusters. The effectiveness of our method is based on the observation that statistically such spurious modes are rare (see also the analysis in the Appendix): It is highly unlikely that many uncorrelated point pairs agree on the same transformation, i.e., are mapped to the same point in 7D transformation space. We can thus afford to perform a spatial verification for each cluster C_k by extracting the connected components of the model that are invariant under the corresponding transformation $\mathbf{T}_k$. We compute these surface patches using an incremental patch growing process, starting with a random point of C_k, which corresponds to a pair $(\mathbf{p}_i, \mathbf{p}_j)$ of points on the model surface. Now we look at the one-ring neighbors of $\mathbf{p}_i$, apply $\mathbf{T}_k$, and check whether the distance of the transformed points to the surface around $\mathbf{p}_j$ is below a given error threshold. If so, we add them to the current patch. We keep extending this patch along its boundary until no more points can be added. During the growth process, we mark all visited samples on the surface and remove points in C_k that correspond to these samples. This process is then repeated using the next point in C_k until all points have been considered.

Since the transformation $\mathbf{T}_k$ at the cluster's maximum does not necessarily provide the best possible transformation for matching the surface patches, we incrementally refine $\mathbf{T}_k$ during the patch growing using the iterated closest points (ICP) algorithm [RL01]. The normalized residual of the ICP matching then provides a quantitative measure for the exactness of the symmetry [MGPG04]. Other measures, such as the Hausdorff distance can also be used. We end up with a collection of pairs of patches on the model surface that are mapped onto each other by the cluster's transformation $\mathbf{T}_k$. This information can be encoded in a weighted graph, where each node corresponds to a patch and each edge denotes the transformation that maps two patches onto each other, weighted by the matching error.

5.6.1 Compound Transforms

Many geometric objects exhibit symmetries in a structured or repetitive fashion resulting in a large number of clusters in transformation space [LCT04]. Encoding all pair-wise symmetry relations for such models leads to a complex and highly redundant symmetry graph and thus a costly verification stage. In this section we describe a simple basis reduction algorithm that computes a compact set of generators for all detected symmetries in transformation space [MKS04]. This significantly reduces the number of spatial consistency checks required for verification and yields a more informative symmetry graph that supports advanced editing operations and high level shape comparisons.

The algorithm shown below takes as input all extracted symmetry transformations T sorted in descending order of cluster height and iteratively processes each transformation $\mathbf{T}_i \in T$. During execution we maintain an alphabet A of generators and the language L that encodes T in terms of the alphabet A. A user parameter η controls the complexity of the algorithm by limiting the search to loops of length $\eta + 1$. The threshold δ measures the allowed deviation from the exact transformation.

Figure 5.5 shows an example of a reduced symmetry basis. Verification can now be applied more efficiently on the set L of compound transformations. For more details on basis reduction we refer to [MKS04].

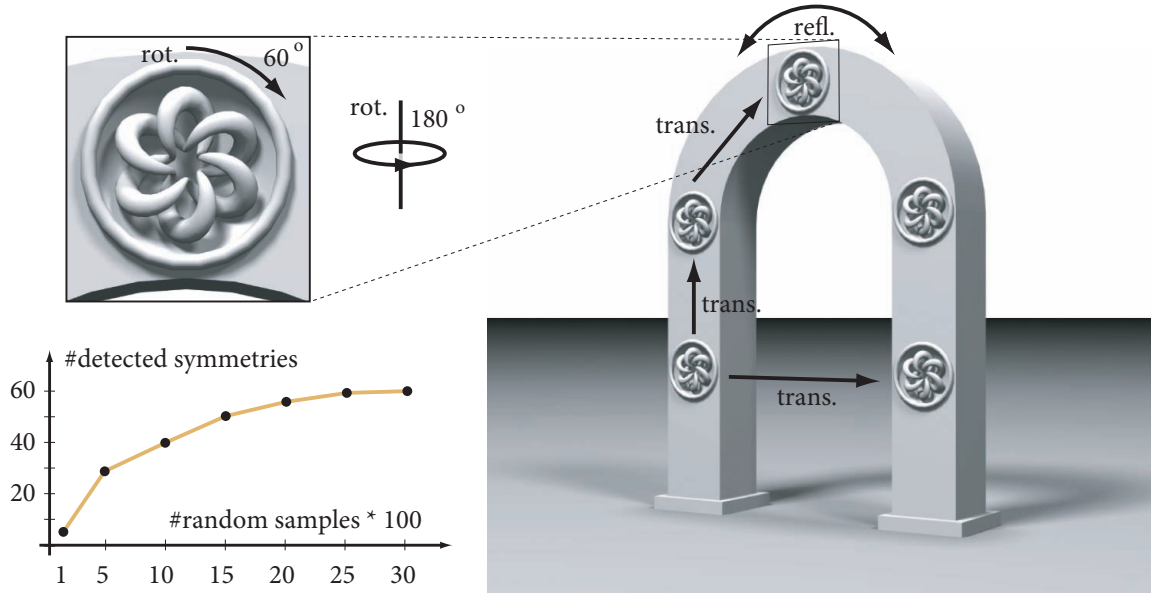


Figure 5.5: *Symmetry graph reduction.* Symmetry graph reduction for a model with structured symmetries at two different scales. 60 significant modes have been extracted in the clustering stage. The reduced basis contains 6 transformations, as indicated by the arrows. The graph on the left shows the number of detected symmetries as a function of random samples in the subset $P' \subset P$.

5.7 Results and Applications

We have implemented the pipeline sketched in Figure 5.2. An initial sample set is created by uniformly sampling the input model. After computing signatures using the method of [ACSD+03], point pruning yields the reduced sample set P. We then select a random subset $P' \subset P$, find all suitable pairs $(\mathbf{p}' \in P', \mathbf{p} \in P)$ based on the proximity in signature space, and compute the corresponding transformations. We perform mean-shift clustering using the method proposed in [AMN+98] to efficiently compute neighborhoods in 7D transformation space. Basis reduction and verification finally yield the symmetric patches. We show in the Appendix that our method is guaranteed to find existing symmetry relations provided the sampling is dense enough with respect to the size of the symmetric patches. The following examples verify this claim and demonstrate that practical results can be obtained even if the theoretical sampling requirements are not met.

Algorithm 2 Symmetry basis reduction.

Input: $T = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$
$A \leftarrow \{\mathbf{I}\}$
$L \leftarrow \emptyset$
for $i = 1$ to n **do**
  if $\exists\ (\mathbf{A}_1, \ldots, \mathbf{A}_\eta)$ with $\mathbf{A}_j \in A$ s.t. $|\mathbf{T}_i - \prod_{j=1}^{\eta} \mathbf{A}_j| \leq \delta$ **then**
    $L \leftarrow L \cup \{(\mathbf{A}_1, \ldots, \mathbf{A}_\eta)\}$
  else
    $A \leftarrow A \cup \{\mathbf{T}_i, \mathbf{T}_i^{-1}\}$
    $L \leftarrow L \cup \{(\mathbf{T}_i)\}$
  end if
end for

Figure 5.1 shows partial and approximate symmetry detection on a laser scan of a hand-sculpted model. The dragon has been sampled with 2470 points out of which 800 have been randomly selected to extract the five most significant modes. The deviations from perfect symmetry are visualized as the signed distances to the closest point on the perfectly symmetric patch. Since these displacements can be compactly encoded, a compressed representation of the surface can be computed based on the extracted symmetry graph.

Figure 5.6 shows an example using the full 7-dimensional symmetry group composed of uniform scaling, rotation, reflection, and translation. All major symmetries are faithfully recovered from only 500 random samples, drawn from an initial sample set of 2000 points.

Figure 5.7 shows a complex architectural model with symmetries at many different scales. The model has been sampled with 2254 points out of which 100 points (black spheres) and 500 points (yellow spheres) where randomly chosen leading to 280 and 1262 points in Γ, respectively. For visualization purposes we project the samples in transformation space to 2D using metric multi-dimensional scaling [CC94] as shown in (b). Note that the elliptical structures are due to errors caused by this projection. The two biggest modes map to the symmetries shown on the right, where the perfect global symmetry is faithfully recovered from only 100 random samples. Automatic model reduction and instantiation is shown in (c). Using the first eight significant modes, a reduction to only 14% of the original model size is achieved by taking out the corresponding symmetric patches. The resulting bounding

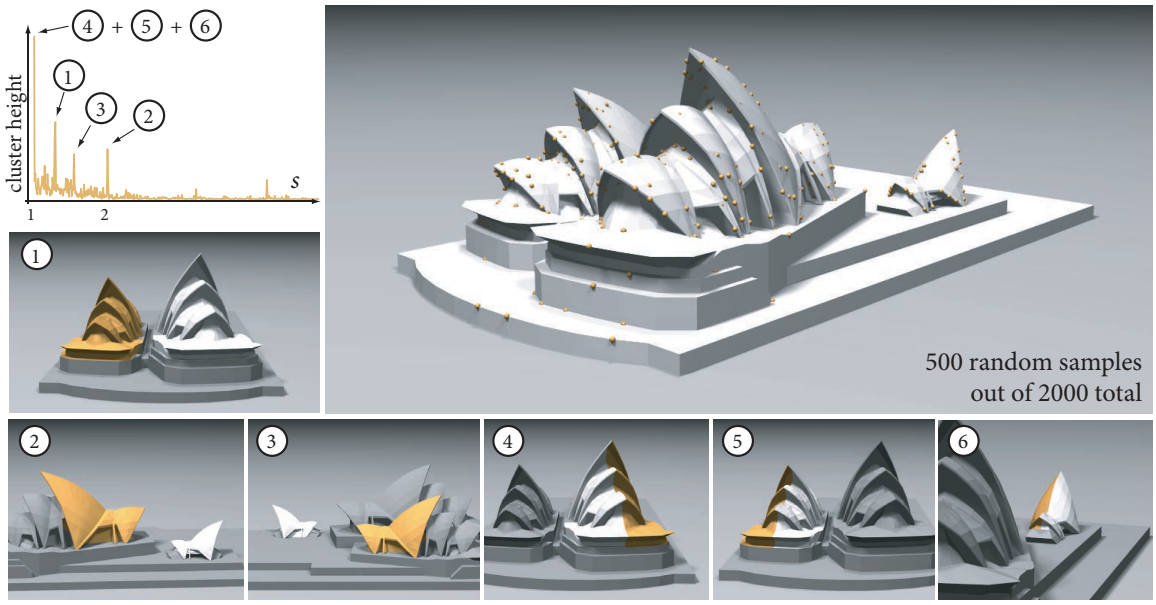
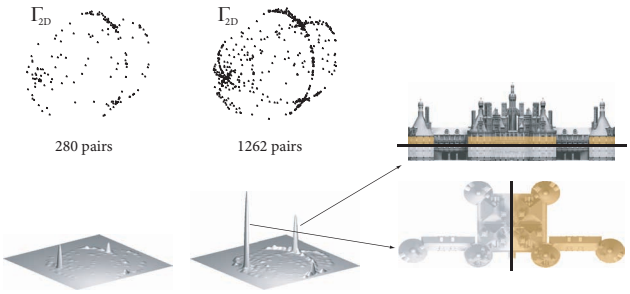


Figure 5.6: *Sydney opera house.* The six most significant modes of the Sydney Opera with the full 7-dimensional symmetries (1, 2, 3) and pure reflections (4, 5, 6). The graph shows the distribution of scaling factors.

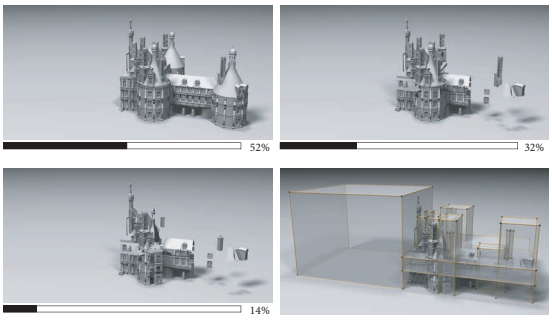
box hierarchy shown in the lower right corner supports efficient spatial queries for applications such as ray-tracing or collision detection directly from the reduced geometry and the corresponding symmetry relations. In (d) we utilized the extracted symmetries to perform advanced editing operations. The user can select a specific symmetry relation and modify certain parts of the model. The system will then automatically apply these modifications to all corresponding patches to maintain the original symmetry.

Figure 5.8 illustrates an application of our method for segmentation. Two poses of the horse have been sampled with 1000 points. We then selected 500 random points on pose A and paired these with samples from pose B, as shown in (d). The mapping to transformation space is thus restricted to only include pairs that contain one sample from either pose. We can then extract the rigid segments of the model as those parts that are invariant under a rigid transformation between the two poses (e). The projected density function is shown in (b). The biggest modes on the left correspond to the torso and head of the horse. The plot on the right is obtained after removing these parts from the model and adding 192

(a)



(b)



(c)

(d)

Figure 5.7: *Chambord castle.* (a) input model with random surface samples drawn from a total of 2254 samples, (b) points in transformation space projected to 2D and associated density plots; the symmetries corresponding to the biggest two modes are shown on the right, (c) successive reduction by taking out symmetric patches and resulting bounding box hierarchy, (d) advanced editing using the extracted symmetry relations.

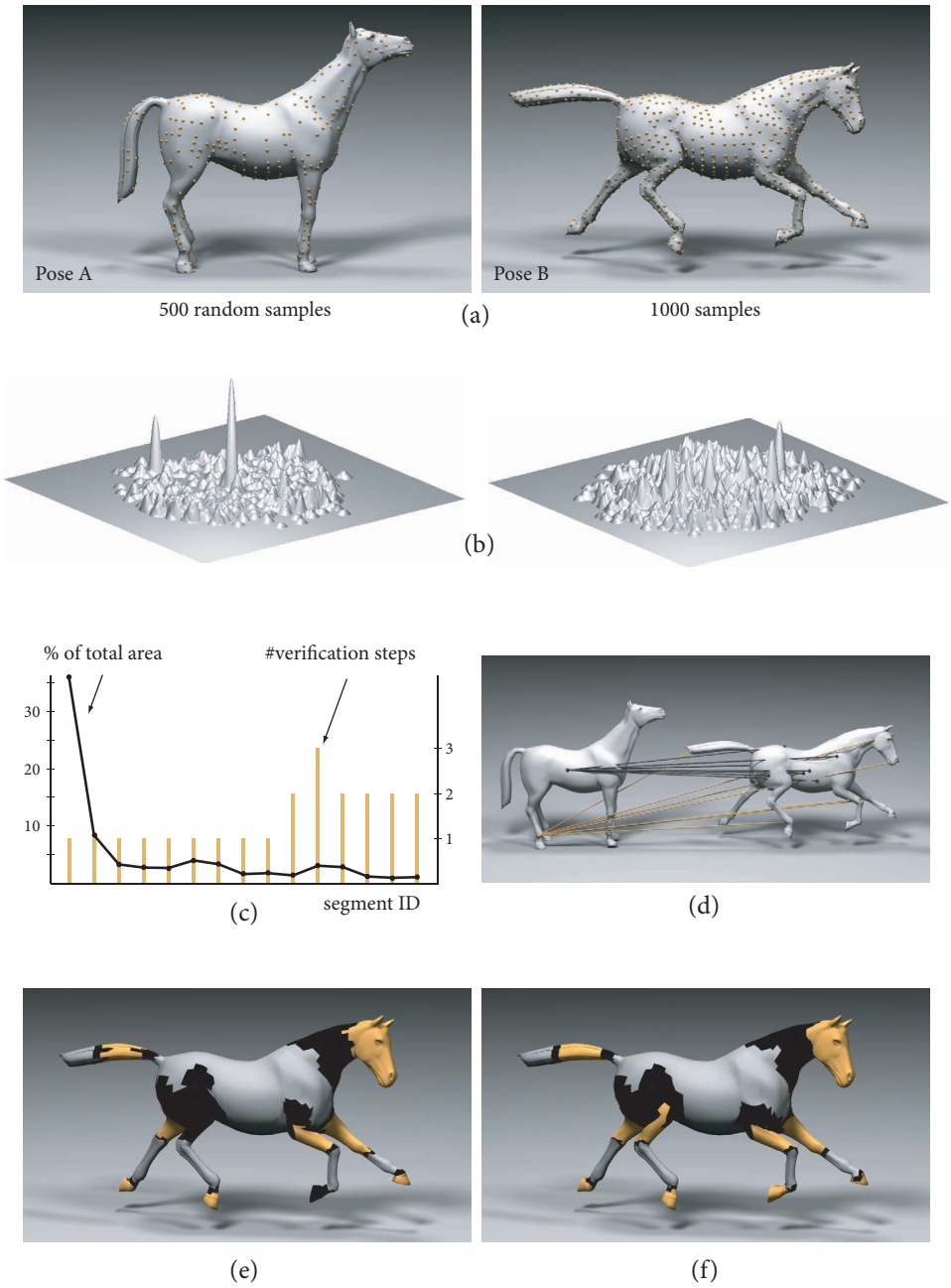


Figure 5.8: *Segmentation and correspondence for the horse model in two different poses.* (a) sampling distribution on both poses, (b) projected density plots, (c) extracted segments and verification effort, (d) sample pairing, (e) segmentation using our method, (f) segmentation obtained with explicit, exact point correspondences.

additional samples to the set of random points. Note that in the 6D transformation space clusters are significantly better separated than in the projected 2D space. As shown in (c), only few clusters have to be discarded in the verification stage. The image in (f) shows the result obtained when using the explicit and perfect one-to-one correspondence available for this specific example, i.e., each vertex is paired with exactly one vertex on the other pose (see also [JT05]). This example illustrates that our method can be used to compute a global correspondence map for articulated models *without* requiring any user-specified marker points. Potential applications include partial scan registration and skeleton extraction.

The performance data of Table 5.1 indicates how the computation depends on the symmetries of the model. The castle has a significantly more complex symmetry structure as compared to the dragon, hence computation times are substantially higher even though the models have roughly the same size. Note that unlike geometric hashing our method requires minimal additional storage, in all our example less than 500 KBytes of additional main memory were needed.

The set of symmetries extracted by our algorithm is limited to *discrete* symmetries, i.e., ones that can be described by a discrete set of points in transformation space. Continuous symmetries, such as those found in rotational or helical surfaces, lead to smooth curves in transformation space and are currently not detected by our method. This is primarily a limitation of our clustering algorithm, which is not well suited for extracting these types of continuous structures. Figure 5.9 illustrates this limitation in a slightly different context. The mean-shift algorithm finds two dominant modes that correspond to the reflections of the feet

Model	# Vertices	Sign.	Pairing	Cluster.	Verif.
Dragon	160,947	3.44	49.24	13.63	7.45
Opera	9,376	0.96	0.02	0.03	0.86
Castle	172,606	5.61	117.81	159.73	5.63
Horse	8,431	0.92	0.01	0.01	1.63
Arch	16,921	0.08	5.86	26.89	2.42

Table 5.1: *Timing Chart.* Timings in seconds for the different stages of the pipeline on a 2.8 GHz Pentium IV with 2GBytes main memory.

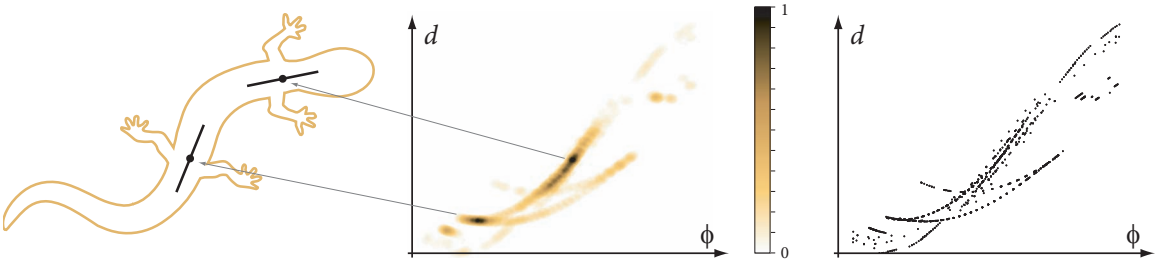


Figure 5.9: *Reflective symmetries of a 2D lizard.* Approximate reflective symmetry across the spline gets mapped to a curve in the 2D transformation space.

as shown in the illustration. The global reflective symmetry across the spine of the lizard, however, is obscured by the deformation of the specific pose of the model and hence not extracted. As the density plot reveals, this deformation is sufficiently small for the reflection planes to vary smoothly from the head to the tail. We can extract the corresponding curve in 2D transformation space using the method proposed in [ACDH06], but have not yet extended the implementation to higher dimensions. Another limitation of our method is inherent in the sample-based approach that we take. Although our algorithm successfully detects the major symmetries of an object, it may fail to identify small partial symmetries during the clustering stage due to the presence of sampling noise. In such a situation, pre-smoothing of the model is required, or the user has to manually tune the patch radii used for curvature estimation (see Section 5.4), until noise is sufficiently blurred out. Naturally, this leads to less distinct curvature estimates, which might cause the subsequent clustering algorithm to miss less pronounced symmetries.

5.8 Summary

In this chapter, we demonstrated how matching local shape signatures followed by clustering in transformation space leads to a provably efficient method for discovering and extracting partial and approximate symmetries of 3D geometric models. While the algorithm is simple and inherently distributed, the randomized construction makes it suited for streaming applications. Our approach accumulates local evidence for global reasoning which gives us a better understanding of the object.

Removing redundant information due to symmetries, and in the process achieving compression, is different from other mesh compression techniques. Our compressed mesh representation can still be used for certain mesh deformations without ever explicitly decompressing the data. We can also deform the model while respecting the original symmetries present in the object. Intuitively we represent the object in a new basis, where the basis shapes are chosen from the model, and are thus adaptive. While exhaustively registering all possible orientations of an object to itself is not feasible, we show how working in the transform domain lets us solve the problem effectively.

In the example of segmentation of horses in different poses, we saw how to use our algorithm for registration of model pairs. Subsequently we can use the obtained correspondence to evaluate similarity between the models. However, this approach requires processing the models in pairs to evaluate their (partial) similarity. Such an approach does not scale well to a database of shapes where pairwise processing of 3D geometry is not a feasible option. In the next chapter, we present a randomized scheme for generating compact shape representations to address this problem.

6

Probabilistic Fingerprints

Signature always reveals a man's character, and sometimes even his name.

— *Evan Esar*

In this chapter, we take a non-traditional approach to define and evaluate similarity between multiple 3D shapes without explicitly bringing them in alignment. Techniques which require extensive processing of shape pairs for evaluating their similarity are not useful when a large number of objects are involved. Further we will like a scheme where the cost of the algorithm depends on the similarity between the shapes and also on the confidence we desire in our estimates.

With this motivation, we present a probabilistic framework [MGGP06] for the efficient estimation of similarity between 3D shapes. Our goal is to establish correspondence across multiple models. For this purpose we need to consistently identify some points across multiple models. However, we do not require the points to be special or feature points – these points need not have any visual significance. We demonstrate how a carefully designed randomized selection gives us a small set of points with good correspondence. The ideas presented in this chapter not only are useful for shape alignment, but gives us a versatile technique for solving a variety of geometry processing problems.

We propose a suitable but compact approximation based on *probabilistic fingerprints* which are computed from the shape signatures using Rabin's hashing scheme and a small

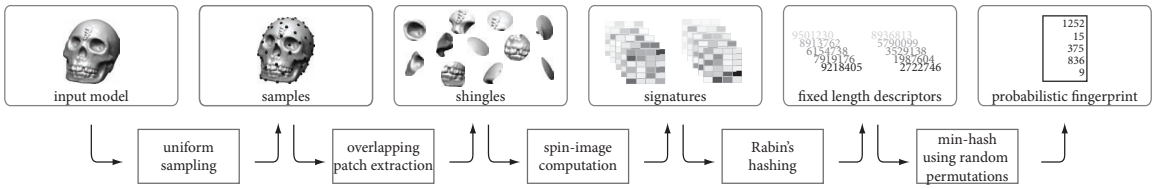


Figure 6.1: *Fingerprint generation.* We first cover an object with ρ-radius balls spaced δ apart with $\rho \gg \delta$. The intersection of each ball with the surface defines a *shingle*. For each shingle, we compute a descriptor, spin-image in this case, which is hashed using Rabin's scheme. Then, in the min-hashing phase, according to m random permutations, we select a small subset of descriptors and store them as the *probabilistic fingerprint*.

set of random permutations. We provide a probabilistic analysis that shows that while the preprocessing time depends on the complexity of the model, the fingerprint size and hence the query time depends only on the desired confidence in our estimated similarity. Our method is robust to noise, invariant to rigid transforms, handles articulated deformations, and effectively detects partial matches.

6.1 Introduction

We are primarily interested in the following question: Given two shapes in arbitrary poses, how can we meaningfully define their similarity and evaluate it efficiently? For database applications, off-line pre-processing of each shape is typically acceptable, if it results in fast query handling. In the same context, it is important to determine, quickly and reliably, when two shapes are *dissimilar*.

In this chapter, we propose an efficient method to define probabilistic fingerprints for 3D shapes and use it to estimate partial similarity. Our approach is complementary to existing work on shape descriptors and signatures, as we can make use of available shape descriptors to define partial similarity across multiple shapes in arbitrary poses. We compress these descriptors using a probabilistic hashing scheme motivated by ideas from the database community. Our fingerprints are such that if they largely disagree, then we can claim with high certainty that the corresponding shapes are dissimilar. This yields an efficient way to quickly filter large shape collections when searching for objects matching a particular model.

In our approach to partial shape similarity, we first cover a given 3D shape with a large collection of overlapping patches. Each patch is mapped to a point in a high dimensional space using a compact, local descriptor that is invariant to rigid transformations. We do not preserve any information about the relative spatial ordering of the patches. The shape is thus mapped to an unordered point set in a high dimensional signature space. We select descriptors that are robust to perturbations, so that patches which are very similar are likely to be mapped to the same locale of this signature space. This is important, as similar regions may not be covered by patches in exactly the same way across two shapes. Clearly, if two shapes are similar, then the corresponding point sets will have proximal regions in proportion to the partial similarity between the original objects. However, since we lose relative patch ordering, it is possible that two largely different shapes have a significant overlap in signature space. Statistically this is a rare event and leads to only a few false positives. It is made even more unlikely by ensuring large overlap between neighboring patches. Motivated by this intuition, we define similarity between two shapes in terms of the similarity between the signature sets. Our definition is invariant to rigid transforms, handles partial matching, and is robust to local deformations and articulated motion.

However, these large high-dimensional point sets have high storage requirements and are difficult to compare efficiently. We therefore compress signature information using a technique called min-hashing [Bro97] to generate a short *probabilistic fingerprint* for each signature set. Subsequently, fingerprints of multiple shapes are compared to estimate similarity between the signature sets, and hence between the original objects. We first map the signatures to a finite universe of numbers using Rabin's hashing scheme [Rab81]. Then, during min-hashing, we use a random permutation to assign a complete ordering to all elements of this finite universe of numbers. We can think of this ordering as the ranking of an 'expert', asked to evaluate the patches according to her criteria. According to the expert ranking, we then select the winner among the set of hashed signatures corresponding to an object. For each object, we collect the winners of m randomly chosen permutations and save them as the *probabilistic fingerprint* of the shape. The same random permutations are used for all shapes. This ensures that patches from different shapes are consistently ordered, according to each of the m chosen 'experts'.

We can efficiently detect if two shapes are similar using our shape fingerprints. However,

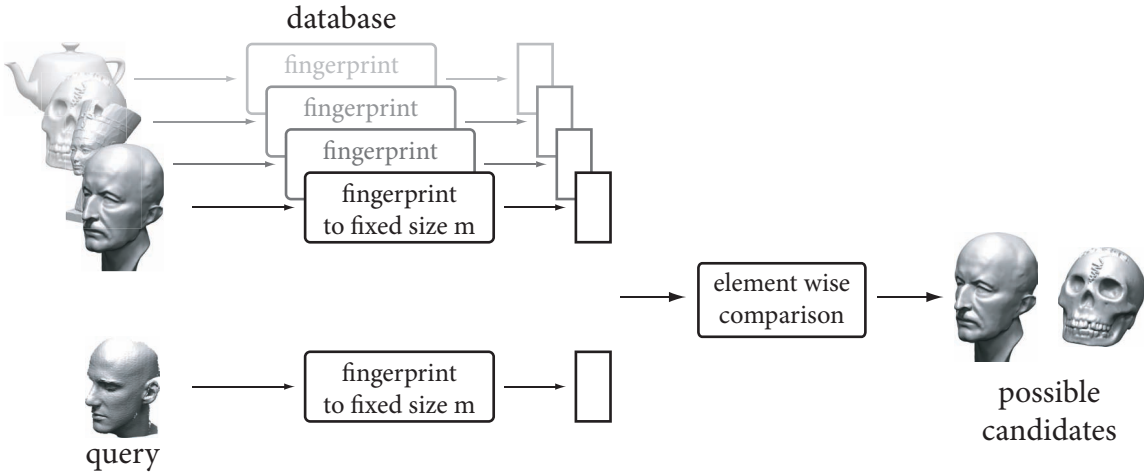


Figure 6.2: *Query Processing.* A query object is first processed to generate its fingerprint using the same parameters used to pre-process the database shapes. Objects with fingerprints similar to fingerprint of the query are returned as possible candidates.

as mentioned before, we may get a few false positive matches. In practice, the number of such false hits is very small and can be handled by match verification using more expensive partial similarity methods. Further, we can show that if two fingerprints are different, then with high probability the shapes are also different. Thus both false positives and false negatives are bounded.

6.2 Related Work

The problem of shape similarity and retrieval has been extensively studied in computer vision and graphics. It has been addressed in great detail by the extensive work done by the Princeton Shape Retrieval and Analysis Group [FKMS05]. Meaningful similarity between two 3D shapes, partial or whole, has to be invariant to rigid transforms. Global shape descriptors, invariant to rigid transformations, include spherical harmonics [KFR03], shape distributions [OFCD02], reflective symmetry descriptors [KCD+04], and Laplace-Beltrami Spectra [RWP05]. Similarity estimation between two models is then reduced to a comparison of the corresponding global shape descriptors. Alternatively, an object can be canonically oriented using principal component analysis (PCA) and descriptors computed on the

rotation normalized shape – examples include extended Gaussian images [Hor84] and shape histograms [AKKS99].

However, all of these global methods are less suitable for detecting partial matches. This problem can be addressed by establishing an explicit correspondence across feature points of the models to compute a good alignment [GMGP05]. Such a solution involves exhaustively considering the various correspondence assignments and is thus computationally expensive. Gal and Cohen-Or [GCO06] proposed a different method for determining partial similarity using geometric hashing techniques. Briefly, in a pre-processing stage, geometric hashing encodes all the possible candidate transforms in a large hash table. While this approach is more efficient, it trades computation time for memory, leading to space requirements of multiple gigabytes even for moderately complex models.

In a different setting, the problem of identifying text or web documents with partial similarity has been extensively studied. Effective solutions to this problem involve clever combinations of hashing and random sampling techniques [Blo70, SGM98, Broo0]. In these schemes, a text document is first converted to a set of overlapping text segments. Similarity between two documents is assigned based on the size of the intersection of the segment-sets which is efficiently estimated using random sampling techniques. Some of these concepts motivated our approach. Our problem, however, is significantly more challenging, as digital 3D shapes have neither the linear ordering nor the canonical decomposition into discrete tokens that is exploited in the text document case.

Contributions

We propose a new statistical approach to efficiently estimate partial or total shape similarity. We introduce the concept of probabilistic fingerprints for 3D shapes, provide a statistical analysis on its effectiveness for partial matching, and show its practical use on a number of different applications. The key insight is that the similarity of two shapes can be estimated by comparing signatures derived from very sparse sets of local patches generated on each model. Based on probabilistic arguments, we show how to *preselect* such patches, *without first needing to establish explicit correspondence relations between the models*. This approach produces a fixed-length fingerprint and avoids costly explicit alignment of the models. Our

similarity measure is invariant to rigid transforms, robust to perturbations, handles articulations, and most importantly, detects partial matches.

6.3 Shape Fingerprinting

Our goal is to reliably and efficiently estimate (partial) similarity between two shapes. The similarity measure should be invariant to rigid transforms and robust to small perturbations. Here we define such a similarity measure for a restricted class of shapes, namely surfaces in $\mathbb{R}^3$ whose normal is defined almost everywhere, e.g., a smooth surface (implicit or explicitly parameterized) or a triangle mesh. The measure is based on surface signatures that allow for effective compression using hashing. We call a hashed signature a fingerprint and start by listing the properties we expect in general from a shape fingerprint.

Fingerprint properties. A *probabilistic fingerprint* is a function f that assigns to each admissible shape a fixed size bit string, i.e. a string in $\{0,1\}^m$. The main purpose of fingerprints is to allow for efficient comparison of shapes. Given a definition of shape similarity (or dissimilarity), any meaningful fingerprint function should have the following properties.

1. Given two shapes S_1 and S_2, we want the following relations to hold with high probability:

   (a) If $f(S_1) \neq f(S_2)$, then S_1 and S_2 are dissimilar.

   (b) If $f(S_1) = f(S_2)$, then S_1 and S_2 are similar.

2. The number of bits m is small compared to the number of bits needed to encode the actual shapes.

3. The function f is efficiently computable. Also, whether $f(S_1) = f(S_2)$ can be quickly checked.

In the following we describe in detail a fingerprint and the corresponding notion of similarity/dissimilarity it is based on. The pipeline for computing the fingerprint is depicted in Figure 6.1.

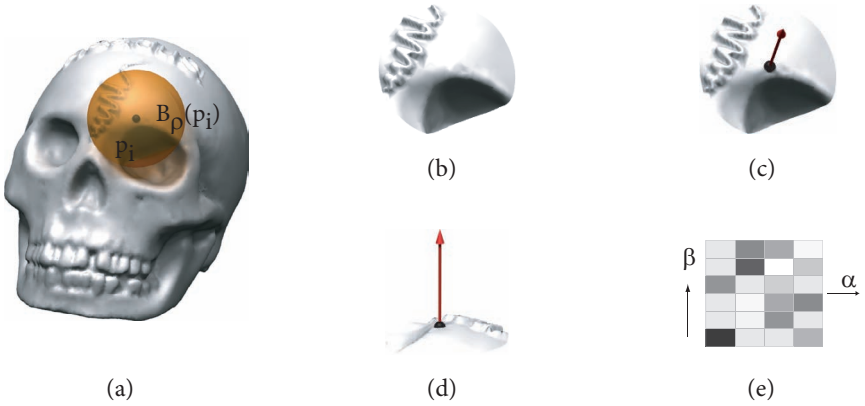


Figure 6.3: *Shingle generation.* (a) $B_\rho(p_i)$ is the neighborhood ball for a point p_i. (b) The selected surface patch (shingle) T_i around p_i. (c) The patch along with the surface normal at p_i. (d) The normal oriented along the z-axis. (e) Computed spin image for patch T_i. The signature is invariant to rigid transforms, and robust to sampling and small surface perturbations.

Our fingerprint relies on the concepts of *sample*, *shingle*, *signature*, *resemblance*, and *hashing*. Next we describe these concepts and how we use them for defining and computing a probabilistic fingerprint.

Figure 6.4: *Overlapping Shingles.* Shingles for two shapes S_1 and S_2 are computed using ρ-radius balls spaced roughly δ apart. If $\tilde{p}_i$ lies in a matching region between S_1 and S_2, then with high probability the shingle at $\tilde{p}_i$ will have a corresponding shingle from S_1 with a significant overlap as $\rho \gg \delta$.

Sample. In the first step, we generate a set of (approximately) uniformly sampled points on the input shape S. Let $P = \{p_1, \ldots, p_n\}$ be the set of sample points with sampling spacing δ. For our fingerprint to work well we assume that for any $p \in S$, $\exists\, p_i \in P$ such that $\|p_i - p\| \leq \delta$. Further, the number of such neighboring points is bounded by a small constant, preventing the sampling from being arbitrarily dense. There is a simple and efficient process for generating such a sample set: Let A be the surface area of the shape. On the surface of the object, we randomly place $n = \lceil A/\pi\delta^2 \rceil$ samples and uniformly spread them out using particle repulsion [Tur92].

Shingles. For each sample point $p_i \in P$, we define a neighborhood of radius ρ where $\rho \gg \delta$. A surface patch $T_i \subset S$ corresponding to point p_i is obtained as $T_i = S \cap B_\rho(p_i)$ where $B_\rho(p_i)$ denotes the ball of radius ρ around p_i (Figure 6.3). If multiple components are present in T_i, we retain only the component containing p_i (the surface is assumed to be a manifold). We refer to these patches as *shingles* and denote the multi-set of all shingles by $\mathcal{P}$. Keep in mind that $\mathcal{P}$ depends on the sample P. Given two shapes S_1 and S_2, with high probability, any shingle from the matching region has a corresponding shingle on the other shape with significant overlap (Figure 6.4).

Signatures. We compute a signature σ_i for each shingle $T_i \in \mathcal{P}$ and denote the multi-set of all signatures by $\mathcal{S}$. A signature σ_i is essentially a string that represents a shingle T_i. Any signature that is invariant to rigid transforms and robust to sampling and local perturbations can be used to this end. Here we use *spin images* [Joh97] which are defined as follows: Let the surface normal at any sample point $p_i \in P$ be n_i. For any point x in the corresponding shingle T_i, its spin-map is defined as:

$$(\alpha, \beta) = \left(\sqrt{\|\bar{y}\|^2 - \langle n_i, \bar{y} \rangle^2}, \langle n_i, \bar{y}/\|\bar{y}\| \rangle \right)$$

where $\bar{y} = x - p_i$. The spin-image s_i of T_i is simply the quantized version of the (α, β) space recording the spin-map of the points of T_i falling into a set of discrete bins (Figure 6.3). Since

spin images are robust to perturbations, if two shingles have significant overlap then they are likely to have the same signatures.

Resemblance. Now we introduce our similarity/dissimilarity measure. Given two surfaces S_1 and S_2 we define their resemblance r with respect to their corresponding signatures $\mathcal{S}_1$ and $\mathcal{S}_2$. Remember that $\mathcal{S}_1$ and $\mathcal{S}_2$ are multi-sets. For each $\sigma \in \mathcal{S}_i$ let $m_i(\sigma)$ denote its multiplicity in $\mathcal{S}_i$. The resemblance of S_1 and S_2 is defined as:

$$r(S_1, S_2) = \frac{|\mathcal{S}_1 \cap \mathcal{S}_2|}{|\mathcal{S}_1 \cup \mathcal{S}_2|},$$

where $|\mathcal{S}_1 \cap \mathcal{S}_2|$ denotes

$$\sum_{\sigma \in \mathcal{S}_1 \cap \mathcal{S}_2} \min(m_1(\sigma), m_2(\sigma))$$

and $|\mathcal{S}_1 \cup \mathcal{S}_2|$ denotes

$$\sum_{\sigma \in \mathcal{S}_1 \cup \mathcal{S}_2} \max(m_1(\sigma), m_2(\sigma)).$$

Since resemblance $0 \leq r(S_1, S_2) \leq 1$ is higher when two shapes are similar, we define distance between shapes as $1 - r(S_1, S_2)$. Observe that this definition is based on the signature sets, and hence depends on the scale parameter ρ used to define the shingles.

Finally, we want to estimate the resemblance using a much sparser representation for the shapes than their signatures, namely their fingerprints. To this end each signature is first hashed into a finite set $\mathcal{U}$ using Rabin's hashing scheme.

Hashing. Rabin's hashing scheme [Rab81] gives a low collision probability for a fixed bit budget by working with irreducible degree k polynomials over $\mathbb{Z}_2$. Let the number of bits required to represent a signature be t. For instance, if a spin image is computed using b bins and each bin is of length l bits, then t is upper bounded by the maximum length of the spin images which is $\log(b2^l)$. If Rabin's hashing scheme h maps each of the n signatures σ_i

corresponding to a shingle T_i down to k bits, then the probability of collision is bounded by

$$\Pr[h(\sigma_i) = h(\sigma_j) | \sigma_i \neq \sigma_j] \leq n^2 t / 2^k. \tag{6.1}$$

For example, if $n = 10^8$ and $t = 128$ then for $k = 80$ the probability of collision is less than 10^{-6}. Thus even with 10 bytes for each signature, we get low collision probability. Further, Rabin's hashing scheme can be very efficiently computed using simple bit arithmetic [Bro93, CL01]. For each signature we store only k bits corresponding to the coefficients of the degree k polynomial in $\mathbb{Z}_2$. We denote the universe of all k-bit numbers by $\mathcal{U}$ and we denote the multi-set of all hashed signatures as $\mathcal{I}$.

We can define the analog of our resemblance function r for multi-sets of hash values as

$$r'(S_1, S_2) = \frac{|\mathcal{I}_1 \cap \mathcal{I}_2|}{|\mathcal{I}_1 \cup \mathcal{I}_2|},$$

where $\mathcal{I}_i$ is the multi-set of hash values corresponding to surface S_i. Evaluating this function instead of $r(S_1, S_2)$ remains impractical, as the involved multi-sets are still too large even though we need less bits to store their elements than we need to store the original signatures. Moreover, set operations between these large unordered multi-sets require $O(n_i \log n_i)$ time where n_i is the number of set elements. As a solution, we further compress each of the multi-sets $\mathcal{I}$ of hash values to generate a small fingerprint. This is done by *min-hashing* using random permutations on the universe $\mathcal{U}$.

Probabilistic Fingerprint. Let $\pi_1, \ldots, \pi_m$ be m random permutations on $\mathcal{U}$, the universe of k-bit numbers. Intuitively, each permutation is like an 'expert' assigning an ordering to $\mathcal{U}$ according to her criteria. Given a multi-set $\mathcal{I}$ of hash values we use the random permutations π_i to compress the set as follows: We replace the multi set $\mathcal{I}$ by the length m sequence of k-bit strings obtained as

$$f(S) = \left(\min\{\pi_1(\mathcal{I})\}, \ldots, \min\{\pi_m(\mathcal{I})\}\right),$$

where the corresponding multiplicities are propagated in the obvious way. This sequence is our definition of a *fingerprint* for a surface S. To generate the permutations π_i, we apply 2-universal hashing [MR95] as an approximation for random permutations, using a random

pair of numbers as parameters.

Based on the fingerprints we estimate the resemblance $r(S_1, S_2)$ by

$$\hat{r}(S_1, S_2) = \frac{\sum_{j=1}^{m} \min(m_{1j}, m_{2j}) \chi\big(f(S_1)_j = f(S_2)_j\big)}{\sum_{j=1}^{m} D_j},$$

with

$$D_j = max(m_{1j}, m_{2j}) \chi\big(f(S_1)_j = f(S_2)_j\big) + m_{1j} \chi\big(f(S_1)_j < f(S_2)_j\big) + m_{2j} \chi\big(f(S_1)_j > f(S_2)_j\big)$$

where $\chi(\cdot)$ is the indicator function taking value 1 if the condition of its argument evaluates to *true*, and 0 otherwise. For the surface S_i, the j-th component of its fingerprint is $f(S_i)_j$ with multiplicity m_{ij}. When the fingerprint consists of strings with all multiplicities equal to one, the resemblance estimate reduces to $\hat{r}(S_1, S_2) = \sum_j \chi\big(f(S_1)_j = f(S_2)_j\big)/m$. Notice that to compare two fingerprints, we simply need to compare them element-wise without any need to solve for correspondences.

In the next section we show that choosing a large enough m gives, with high probability, a good estimate of resemblance. In practice $m \approx 1000$ is sufficient and hence the probabilistic fingerprints, in the order of 10KBytes, are very compact.

6.4 Analysis

In this section we analyze the performance of our fingerprints — as mentioned before our goal is to approximate the resemblance of two surfaces effectively and efficiently.

Rabin's hashing scheme maps any signature σ to a number with bit length k. This mapping obviously results in some collisions that can be quantified as:

$$\begin{aligned} \sigma_i = \sigma_j \quad &\Rightarrow \quad h(\sigma_i) = h(\sigma_i) \\ \sigma_i \neq \sigma_j \quad &\Rightarrow \quad \Pr[h(\sigma_i) = h(\sigma_j)] \leq p, \end{aligned} \tag{6.2}$$

where p is $n^2 t/2^k$, see Equation 6.1. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be multi-sets of signatures for the surfaces

S_1 and S_2, respectively. Let

$$\mathcal{A} = \mathcal{S}_1 \cap \mathcal{S}_2,\ \mathcal{B} = \mathcal{S}_1 \setminus \mathcal{S}_2, \text{ and } \mathcal{C} = \mathcal{S}_2 \setminus \mathcal{S}_1,$$

where the set operations again are defined in the multi-set setting, i.e. the $\setminus$ operation respects the multiplicities. We can relate $a = |\mathcal{A}|$ (size of a multi-set is the sum of the multiplicities of its elements), $b = |\mathcal{B}|$, and $c = |\mathcal{C}|$ to the resemblance of S_1 and S_2 as follows:

$$r(S_1, S_2) = a/(a+b+c). \tag{6.3}$$

Let $\mathcal{I}_1$ and $\mathcal{I}_2$ denote the multi-sets of hashed signatures for S_1 and S_2, respectively. Having set the terminology we now quantify the errors incurred by Rabin's hashing scheme.

Using Equation 6.2 and the definitions of a, b, and c in expectation we get:

$$\begin{array}{ccccc} a & \leq & |\mathcal{I}_1 \cap \mathcal{I}_2| & \leq & a+d \\ a+b+c-d & \leq & |\mathcal{I}_1 \cup \mathcal{I}_2| & \leq & a+b+c, \end{array}$$

where $d = (2bc + ac + ab)p$ is obtained by a simple union bound argument: For any element in $\mathcal{B}$ the probability to participate in a collision that affects the set operations is upper bounded by $(a+c)p$. Thus by linearity of expectation the expected number of such collisions contributed by $\mathcal{B}$ is upper bounded by $(a+c)bp$. Similarly, the expected number of collisions contributed by $\mathcal{C}$ is upper bounded by $(a+b)cp$. Adding these two bounds gives the bound d.

For the approximation quality of the resemblance by $r'(S_1, S_2) = |\mathcal{I}_1 \cap \mathcal{I}_2| / |\mathcal{I}_1 \cup \mathcal{I}_2|$ we get in expectation the following bounds:

$$r(S_1, S_2) = \frac{a}{a+b+c} \leq r'(S_1, S_2) \leq \frac{a+d}{a+b+c-d}.$$

Crucial is only the upper bound which we can also write as

$$\frac{a+d}{a+b+c-d} = \frac{r(S_1, S_2) + \epsilon}{1-\epsilon},$$

if we set $\epsilon = d/(a+b+c)$. By increasing k, i.e. the number of bits each signature gets

mapped to, we can make ϵ arbitrarily small. For small enough ϵ and assuming $r(S_1, S_2) > 0$ we get in expectation

$$\begin{aligned} r'(S_1, S_2) &\leq \frac{r(S_1, S_2) + \epsilon}{1 - \epsilon} \leq (r(S_1, S_2) + \epsilon)(1 + \epsilon) \\ &\leq r(S_1, S_2)(1 + \sqrt{\epsilon}). \end{aligned}$$

Using Markov's inequality [Fel57] this yields

$$\begin{aligned} &\Pr[r'(S_1, S_2) \geq \lambda(1 + \sqrt{\epsilon})r(S_1, S_2)] \\ &\leq \Pr[r'(S_1, S_2) \geq \lambda \mathrm{E}[r'(S_1, S_2)]] \\ &\leq 1/\lambda. \end{aligned} \tag{6.4}$$

Finally, we have to check how our estimate behaves under the random permutations that we used to compute the fingerprints. When all the strings have multiplicities one, we use the following fact, see [Bro97],

$$\Pr[f(S_1)_j = f(S_2)_j] = r'(S_1, S_2),$$

for all $j = 1, \ldots, m$. Hence estimating $r'(S_1, S_2)$ by using m random permutations is equivalent to performing m coin tosses to evaluate the bias of the coin. Using strong Chernoff bounds we can bound the estimated resemblance $\hat{r}(S_1, S_2)$ as

$$\Pr[(1 - \delta)r'(S_1, S_2) \leq \hat{r}(S_1, S_2) \leq (1 + \delta)r'(S_1, S_2)] \geq 1 - \eta, \tag{6.5}$$

if $m \geq 4 \ln(2/\eta)/(\delta^2 r'(S_1, S_2))$.

Combining our probabilistic bounds by taking a union bound for the event we dealt with in Equation 6.5 and the complement of the event we dealt with in Equation 6.4 we conclude that

$$(1 - \delta)r(S_1, S_2) \leq \hat{r}(S_1, S_2) \leq \lambda(1 + \delta)(1 + \sqrt{\epsilon})r(S_1, S_2)$$

with probability at least $1 - (\eta + 1/\lambda)$. That is, with high probability $\hat{r}(S_1, S_2)$ is very close to $r(S_1, S_2)$. The analysis can be easily extended for multi-sets giving us a similar result.

model	# vts.	uniform samp.	spin image	Rabin hash	min hash
skull	54k	0.8	7.5	0.05	4.5
Caesar	65k	1.4	7.3	0.08	10.3
bunny	121k	1.8	13.8	0.04	2.9
horse	8k	0.7	5.7	0.05	7.3

Table 6.1: *Performance.* Timing in seconds for the different stages of the fingerprint computation ($m = 1000$) on a 3 GHz Pentium 4 with 2GB RAM. Caesar and bunny refer to the complete models. Average query time is roughly 15msec.

Observe that the size m of the fingerprint depends on the desired confidence in our estimated resemblance.

6.5 Results and Applications

We have implemented the framework shown in Figure 6.1. Along with each fingerprint, we store some additional header information: a seed for the random number generator, sample spacing δ, shingle radius ρ, parameters for computing spin-images, and k, the degree of the polynomial used in Rabin's hashing scheme. The choice of these parameters is not critical for the success of our scheme provided the conditions given in Section 6.4 are satisfied. However, we can only compare fingerprints computed using consistent sets of parameters. Typical time requirements for the various stages are shown in Table 6.1.

Partial Matching. Our scheme is tailored to detect partial matches efficiently. In an experiment we take a bust of Caesar along with its three partial scans (Figure 6.5). The triangulations of the models are very different and thus test the robustness of our scheme. For each model, we independently compute their probabilistic fingerprint. Then, for each model pair, we compute its resemblance using the corresponding fingerprints (shown in black). For comparison, we compute the ground truth resemblance (shown in yellow) via spin image signatures. Since our resemblance measure is symmetric, the difference between diagonally opposite elements in the resemblance matrix quantifies our estimation error.

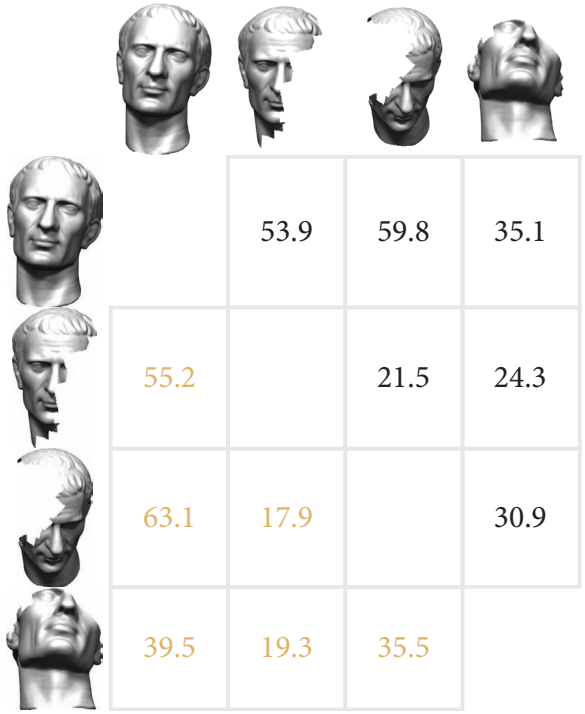


Figure 6.5: *Resemblance between Partial Scans.* In black, resemblance (in %) computed using fingerprints. In yellow, approximate ground truth computed from spin-images. Our resemblance definition being symmetric, difference between diagonally opposite elements quantifies the corresponding estimation error.

Articulated Motion. Resemblance, as measured by our scheme, is robust to articulated deformations. If large chunks of a model are rigidly deformed across two poses, then the corresponding shingles and their hashed descriptors are also preserved. Results on two articulated poses of a horse model are shown in Figure 6.6. The size of the shingle, determined by ρ, affects the resemblance score: smaller ρ gives higher resemblance and vice versa. The ground truth (yellow curve), determined using the spin-image signatures, is within $\pm 5\%$ of our estimated values.

Automatic Scan Alignment. The problem of automatic scan alignment has been previously addressed by Huber and Hebert in [HH03]. Their system can be made significantly faster using our scheme. We explain our method with reference to scans (in arbitrary initial orientations) of the Stanford bunny (Figure 6.7). In the pre-processing stage, for each of

the ten scans, we independently compute its fingerprint. Now for each pair of models, we estimate their resemblance using the respective fingerprints and store the edge joining that pair in a heap with the largest element on top. We then extract the top edge, try to explicitly align the corresponding patches using a global aligner [GMGP05], and if the alignment is not valid we just pick the next largest edge. If an alignment is valid, we merge the respective patches, and need to compute the fingerprint for the merged patch. However, given two patch fingerprints $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$, we can very efficiently estimate the fingerprint for the merged scan simply as $(\min(a_1, b_1), \ldots, \min(a_m, b_m))$ without explicitly computing the fingerprint for the merged scans. Using this estimated fingerprint, the heap can be efficiently updated by re-evaluating only the affected edges. In the figure, the number of required global registration or verification steps are shown in parenthesis. Since global registration is much more costly compared to fingerprint matching, our method, by quickly pruning away non-matching scans, greatly speeds up the whole process.

Adaptive Feature Selection. As shown previously, we can even identify shapes that match only partially. However, with a bit more effort we also get very good hints about regions of overlap. While comparing two fingerprints, we identify the min-hashes that agree and map them back to the original surface shingles. The union of these shingles give us a very good

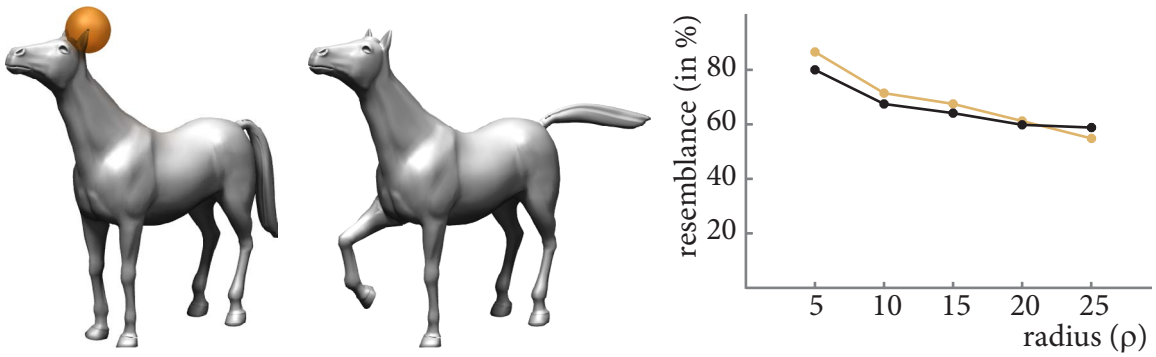


Figure 6.6: *Resemblance between Articulated Shapes.* The yellow ball shows a neighborhood with $\rho = 10$ used for defining shingles. At low values of ρ, we get a high resemblance, since the effect of articulation is felt only by few of the shingles. As ρ increases, resemblance goes down. In yellow, we show the ground truth.

estimate of the region of overlap (Figure 6.8). Subsequent global registration algorithms benefit significantly from this stage, since the adaptive feature points, given by matching shingle patches, very likely lie in areas of overlap and have correct correspondences. In cases when fingerprints are computed independently, we can similarly identify potential

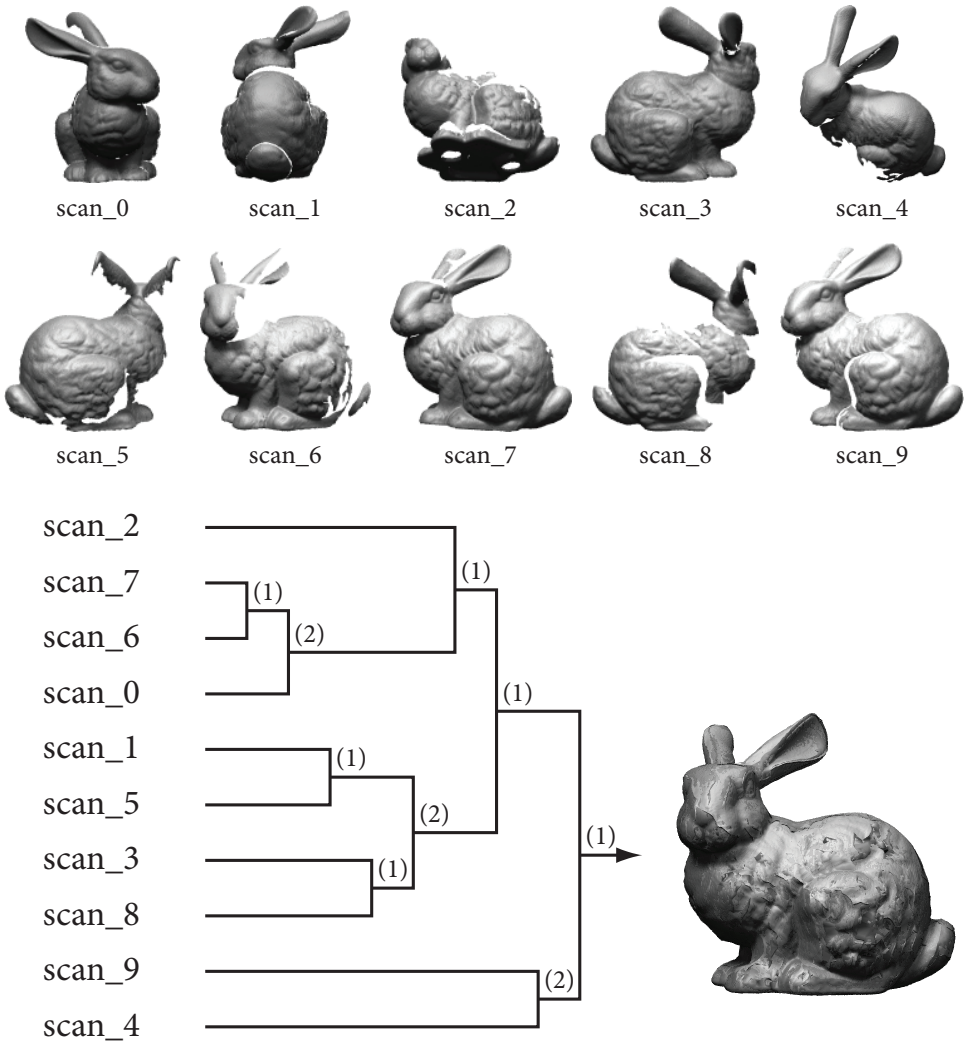
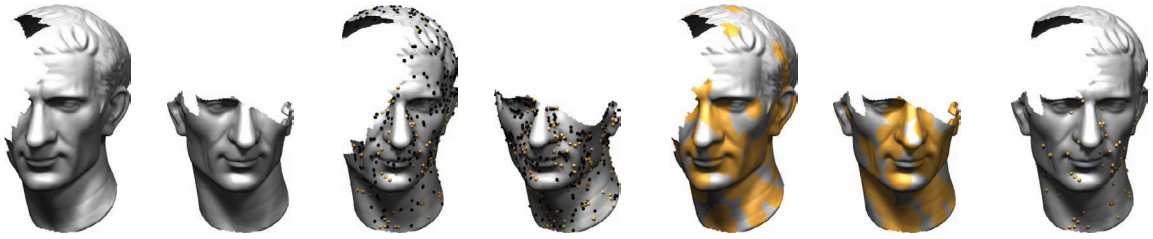


Figure 6.7: *Automatic Scan Alignment.* Given ten initial scans of the Stanford bunny in *arbitrary* poses, for each scan we compute its probabilistic fingerprint with $m = 1000$. Scan pairs with highest resemblance are picked, and then using [GMGP05] their alignment verified. If the scan-pair align, the fingerprint for the merged scan is estimated. The process continues until no more scans can be combined. The number of required global alignment steps is shown in parenthesis.

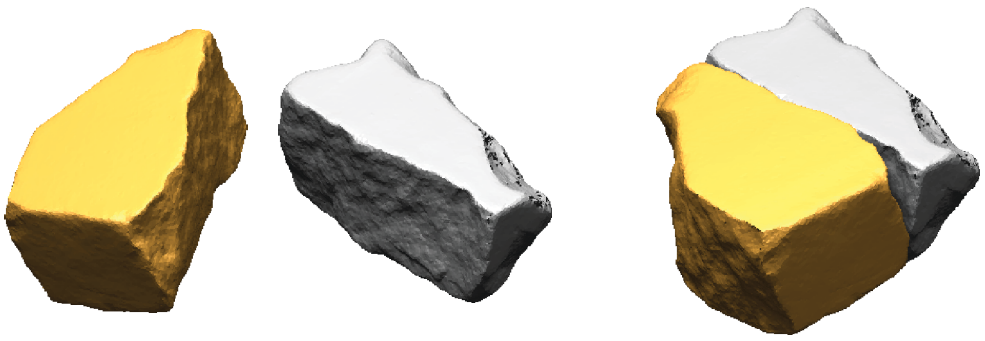


(a) (b) (c) (d)

Figure 6.8: *Adaptive Feature Point Selection.* (a) Two shapes in arbitrary poses. (b) For each scan, black and yellow balls denote shingle centers chosen by min-hash. In yellow, shingle centers whose min-hashes agree across the two models. (c) Hints about possible overlap regions obtained by mapping matching min-hashes back onto the objects. These are used for adaptive feature point selection. (d) Final alignment using chosen feature points. The set of features with correct correspondence is shown in yellow.

overlap regions across multiple shapes, if we additionally store shingle locations for the min-hashed patches along with the fingerprints. Timing complexity and storage requirements still remain O(m).

In order to identify partial complementary matches between two shapes S_i and S_j, we can use a similar method. The fingerprint for S_i is computed as usual. For S_j, when computing spin-images, we flip the point normals to take care of complementary shapes. More generally, for each shape we can compute its fingerprints and its complement fingerprint. A



Scan A Scan B Final Alignment

Figure 6.9: *Complementary Shapes.* Given two complementary scans in arbitrary poses, we find their alignment using our adaptive feature selection. To detect complementary shapes, flipped normals are used for computing fingerprint of scan B.

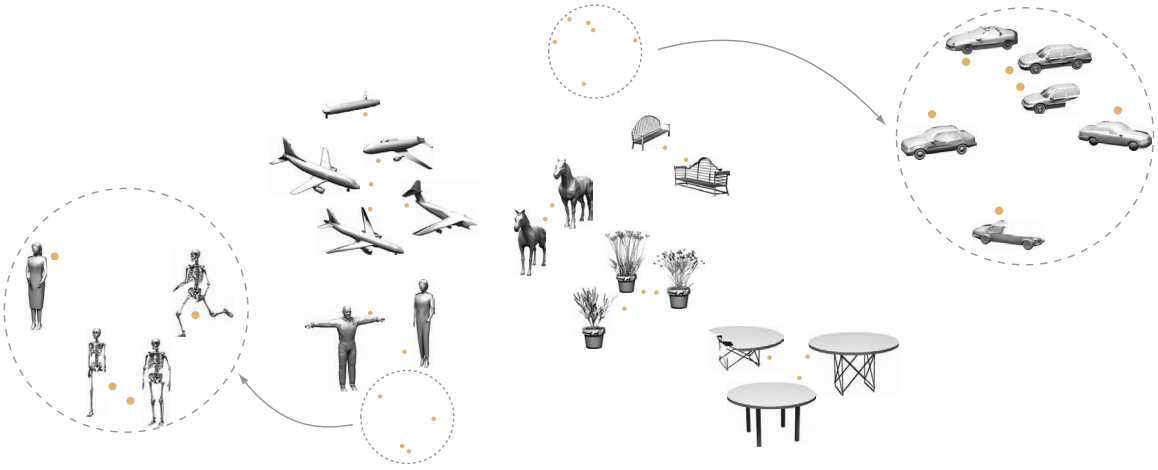


Figure 6.10: *Database Classification.* Shape classification result according to our probabilistic fingerprints. Given any two shapes S_i and S_j, distance between them is defined as $1 - r(S_i, S_j)$. Using this notion of distance, we compute the full distance matrix for a database of shapes. The 2D projection of the fingerprint shape space is computed using classical multi-dimensional scaling (MDS).

possible application is automatic alignment of broken fragments as shown in Figure 6.9. In this special scenario, using prior information, the flat surface of the scans can be automatically removed as proposed by Huang et al. [HFG+06] as these regions are known not to be in overlap areas.

Database Classification and Retrieval. We use resemblance between pairs of shapes to efficiently classify a shape database and retrieve models from it (Figure 6.2). Our database comprises of models, in arbitrary initial positions, from the Princeton shape benchmark [FKMS05]. For each shape, we first compute its fingerprint. A shape distance matrix is then build using $1 - r(S_i, S_j)$ as the distance between any pair of shapes S_i and S_j. We extract a 2D embedding of the fingerprint shape space using multi-dimensional scaling [CC94] on the computed distance matrix. Figure 6.10 shows a selection of models in the embedded shape space. We get meaningful clustering of shapes even in the presence of articulations and partial matches. A typical query result from the processed database is shown in Figure 6.11. The resemblance scores for this query with any of the tables, planted pots, furniture, or car

models is less than 2%. Most of the models in our database being degenerate meshes, we expect a volumetric representation coupled with a suitable signature like spherical harmonics [KFR03] will further improve our performance. Our method is in complement with existing algorithms for shape matching and hence we can use many of the popular shape descriptors in our framework. However, a careful study has to be done to fully evaluate these benefits of our algorithm.

Mesh Authentication. Our scheme can be modified for authenticating geometric models. In the signature computation phase we increase the number of bins making the spin-images sensitive to minor perturbations. Then given a mesh and a partially modified copy, we can use fingerprints to probabilistically identify regions that remain unchanged. For example, if we compute such fragile fingerprints for the original skull model and one corrupted with 1% (of the bounding box) noise, their resemblance is 1.8%. However, if we reorder the vertices, or apply any rigid transform to the original mesh, the resemblance is almost 100%. Local deformations or partial matches can be detected as before. Moreover, for authentication, only a small fingerprint needs to be transmitted and compared with the fingerprint computed from a copy of the mesh. Further investigation needs to be done for quantifying immunity against other types of attacks [WC05].

6.5.1 Improvements and Limitations

Rabin's method can hash similar signatures to very different values. Such error can be reduced by using locality sensitive hashing (LSH) [IMRV97] where probability of collision between any two signatures is inversely related to their distance. In practice, this may improve the resemblance estimates.

As seen in Figure 6.6, in some cases the scale ρ has a significant effect on resemblance. To deal with this, we can compute a multi-scale fingerprint over ν different choices of ρ. Storage requirement increases ν fold.

In the current form, we cannot handle scaling. Though pre-scaling of objects may be done using anisotropic scaling [KFR04a], such a method fails for partial similarities. We are currently researching other possibilities to handle this scenario.

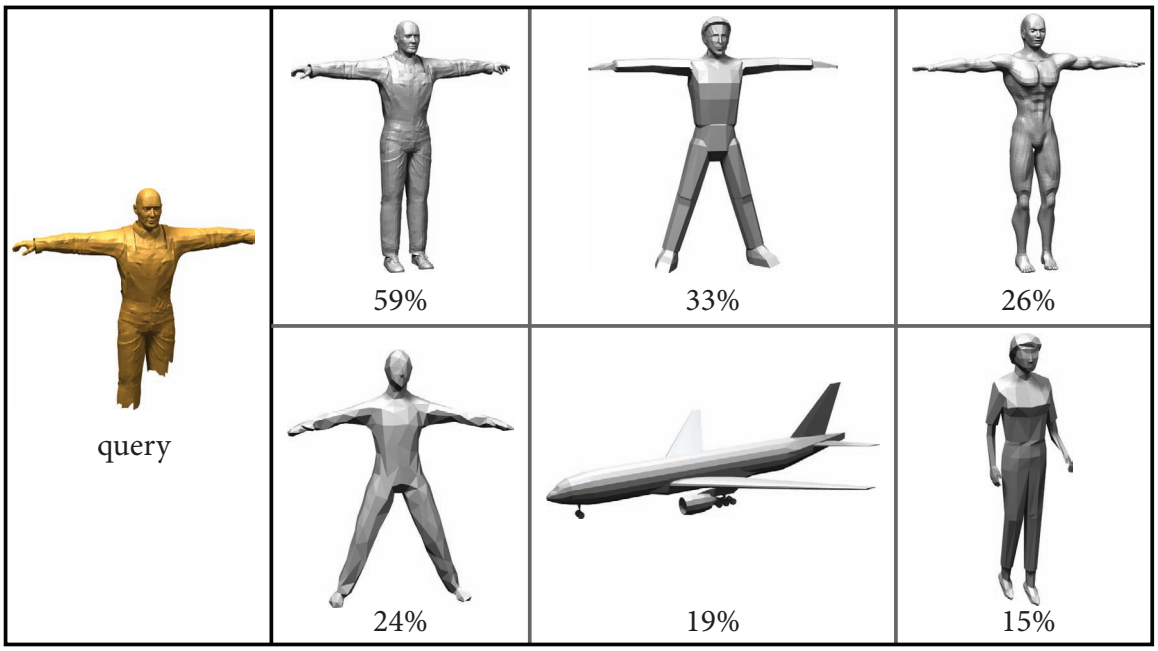


Figure 6.11: *Database Retrieval.* Given a query shape, we show the models retrieved by our algorithm from a database of shapes in arbitrary poses. Our scheme handles partial matches, and is robust to articulations. Corresponding resemblance scores are shown.

6.6 Summary

In this chapter, we showed how to build compact probabilistic fingerprints for digital geometry models that allow efficient model comparison for partial or total similarity. It is interesting that our scheme relies on randomness for selecting the 'shape features' through the presence or absence of which similarity is estimated. We give provable bounds on the quality of our shape comparisons demonstrating the power of randomization in a geometric context.

7

Conclusions

It is good to have an end to journey toward; but it is the journey that matters.

— *Ursula K. LeGuin*

This chapter concludes this thesis with a summary of principal contributions and some thoughts about extensions of this work and future research.

The underlying theme of the thesis has been to estimate inter and intra model similarity with applications to a variety of geometry processing problems. The techniques mainly rely on gathering evidence from local geometry to infer about global properties of objects. The use of randomization gave us output sensitive algorithms with running time related to the confidence we want in our final results.

7.1 Principal Contributions

In the context of geometry processing of 3D geometry, the main contributions of this thesis are:

- A distance metric based algorithm for partial alignment of 3D models in arbitrary initial poses.
- A squared distance field based optimization procedure for refining partial alignment of model pairs.

- A framework for shape completion using geometric priors.
- An algorithm for partial and approximate symmetry detection for 3D geometry.
- Probabilistic fingerprints as a compact representation of 3D models and useful for partial similarity estimation.

7.2 Future Work

An obvious extension of this work is to deploy our shape registration and completion tools for acquiring 3D geometry in massive scale. Example of such large scale scanning include scanning of city blocks [FZ04], historic artifacts [LPC+00]. The acquired models can then be used towards planning and restoration projects. Related efforts are also being made in the Google City Block project and Microsoft Virtual Earth.

Such large scale data acquisition will lead to a demand for better classification or grouping of captured shapes. In this thesis we explored some related questions about model reduction, compact shape representation using randomization techniques. The compactness of the proposed shape fingerprints may also enable distributed geometry processing tasks in sensor network settings, where geometry acquisition, storage, and retrieval may be required over geographically dispersed deployments.

The tools described in this thesis can be combined in several ways to form various integrated systems. One possible application is in reverse engineering and inventory control. At present, almost all major automobile manufacturers maintain a digital collection of model parts along with physical parts. While reverse engineering some model parts it is important to reuse as much prefabricated or predesigned parts as possible. To this end one can do the following: Get several partial scans of the object. Using global and local alignment, stitch the scans together to get one consistent model. Minor holes or regions of missing data can then be consolidated using our method of geometric prior based shape completion. Finally we can compute probabilistic fingerprints for the consolidated models and look for objects with partial matches in the model repository. Once we get a set of candidate models, we can verify the matches using our global registration algorithm. From a consumer point of view, a similar system can be used for searching hardware catalog for replacement parts for

a broken part poorly scanned using a cheap scanner.

Similar methods can be employed for face recognition for security reason, or for automated surveillance and detection of suspicious objects entering a region of interest. All these applications can be made robust by incorporating higher level information about the scanned objects – symmetries and structure of detected symmetries are examples of such attributes. Further research needs to be conducted in this area with rigorous testing on range data of various quality and resolution to evaluate the benefits of such methods.

Better understanding of shape similarity and formalizing the concept of a distance metric for shapes will help us to better understand the shape space. Such a space can prove to be very useful for animating, deforming or even editing 3D shapes.

We are also interested in applying these algorithms for data analysis on other forms of information like motion sequences [ZMCF05], protein folding sequences [Ste96, Doo96], time series data [SS00], etc. It may be interesting to extend the techniques to high dimensional data where pattern identification is not trivial. A possible such data source is the 9-dimensional Mumford data set [MLP03] generated from a natural image database.

A

Quality of Fit

In Chapter 4, in the data classification stage, we estimate quality of scanned data. In this section we briefly describe our approach. More discussion on this topic can be found in [PMG04].

The quality-of-fit estimate c_i^λ is derived from weighted covariance matrix

$$\mathbf{C}_i = \sum_j (\mathbf{p}_j - \mathbf{p}_i)(\mathbf{p}_j - \mathbf{p}_i)^T \phi_i(\|\mathbf{p}_j - \mathbf{p}_i\|),$$

where the weight function ϕ_i is the compactly supported fourth order polynomial

$$\phi_i(r) = \begin{cases} 1 - 6r^2 + 8r^3 - 3r^4 & r \leq 1 \\ 0 & r > 1 \end{cases}$$

with $r = \|\mathbf{p}_j - \mathbf{p}_i\|/h_i$. The support radius h_i defines the geometric scale at which the data is analyzed. Comparable results are obtained using truncated Gaussians, or similar positive, monotonously decreasing weight functions. Let $\lambda_i^1 \leq \lambda_i^2 \leq \lambda_i^3$ be the eigenvalues of $\mathbf{C}_i$. The normalized weighted least squares error of the best tangent plane estimate can be derived as $\lambda_i = \lambda_i^1/(\lambda_i^1 + \lambda_i^2 + \lambda_i^3)$ [PMG04]. Since $\lambda_i = 0$ indicates a perfect fit and $\lambda_i = 1/3$ denotes the worst possible distribution, we define $c_i^\lambda = 1 - 3\lambda_i$. We measure local sampling uniformity as the ratio $c_i^\sigma = \lambda_i^2/\lambda_i^3$. $c_i^\sigma = 0$ means that all samples in the support of ϕ_i lie on a line (see outliers in Figure 4.3), whereas $c_i^\sigma = 1$ indicates a uniform distribution of samples

around $\mathbf{p}_i$. Points on the boundary will be assigned intermediate values.

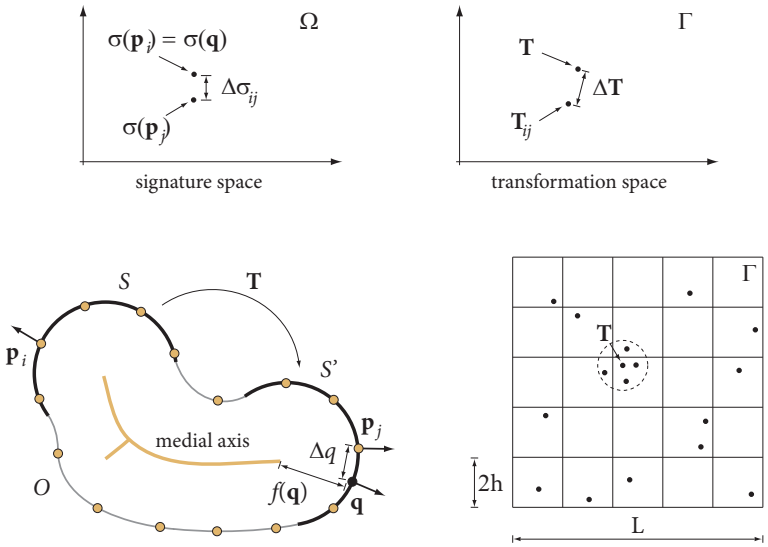
B

Theoretical Analysis

In this section, we provide probabilistic bounds on the sampling requirements of our symmetry detection algorithm presented in Chapter 5. More precisely, we define conditions on the sample set P and the number $n' = |P'|$ with $P' \subset P$ of random samples required to find a symmetry of a certain size with high probability.

Suppose we are given a smooth manifold surface O with a symmetric patch $S \subseteq O$ and a partial symmetry transformation $\mathbf{T} \in \Gamma$, such that $S' = \mathbf{T}(S) \subseteq O$. For conciseness of the exposition, we restrict the derivations to the group of rigid transformation, i.e., ignore uniform scaling. The analysis extends in a natural way, however. Assume $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is an ϵ-sampling of the surface O, i.e., for every point $\mathbf{x} \in O$ there exists a sample $\mathbf{p} \in P$ such that $|\mathbf{p}_i - \mathbf{x}| < \epsilon f(\mathbf{x})$, where $f(\mathbf{x})$ denotes the local feature size at $\mathbf{x}$, i.e., the smallest distance of $\mathbf{x}$ to the medial axis of O (see [AB98]).

For a given sample $\mathbf{p}_i \in P \cap S$, let $\mathbf{q} \in S'$ be the symmetric point of $\mathbf{p}_i$ on the surface, i.e., $\mathbf{q} = \mathbf{T}(\mathbf{p}_i)$. In general, this point will not be part of the sample set, i.e., $\mathbf{q} \notin P$. However, we can show that there exists a point $\mathbf{p}_j \in P$ such that $\mathbf{p}_j = \mathbf{T}_{ij}(\mathbf{p}_i)$ and $||\Delta\mathbf{T}|| = ||\mathbf{T} - \mathbf{T}_{ij}||$ is small.



Let $F_{\mathbf{q}} = [\mathbf{n_q}\ \mathbf{c_{q,1}}\ \mathbf{c_{q,2}}]$ denote the local frame at $\mathbf{q}$ spanned by the normal and the principal curvature directions (see Section 5.4). Then the transform mapping $\mathbf{p}_i \mapsto \mathbf{q}$ can then be expressed as a rotation $\mathbf{R} = F_{\mathbf{q}} F_{\mathbf{p}_i}^{-1}$ followed by a translation $\mathbf{t} = \mathbf{q} - \mathbf{R}\mathbf{p}_i$. Since P is an ϵ-sampling of O, $\exists\ \mathbf{p}_j \in P$ such that $|\mathbf{p}_j - \mathbf{q}| < \epsilon f(\mathbf{q})$. Using results from [AB98, CSM03] it follows that if $\epsilon < 0.08$, $||F_{\mathbf{q}} - F_{\mathbf{p}_j}|| \leq c_1\epsilon$ where c_1 is a constant depending on the radius of the ball used for estimating the curvature tensor. Let $\mathbf{T}_{ij} = (\mathbf{R}', \mathbf{t}')$ denote the transform mapping $\mathbf{p}_i \mapsto \mathbf{p}_j$. Using the previous relations and the triangle inequality one can show that

$$||\Delta\mathbf{T}||^2 = ||\mathbf{R} - \mathbf{R}'||^2 + \beta||\mathbf{t} - \mathbf{t}'||^2 \leq c_2^2\epsilon^2,$$

where c_2 is a constant depending on c_1, β and the diameter of O.

Due to the stability of local signatures on a smooth surface [MHYS04], we can choose a small $\Delta\sigma$ so that $\Delta\sigma_{ij} = \|\sigma_{\mathbf{p}_i} - \sigma_{\mathbf{p}_j}\| < \Delta\sigma$ when $|\mathbf{p}_j - \mathbf{q}| < \epsilon f(\mathbf{q})$. In other words, the signatures of $\mathbf{p}_i$ and $\mathbf{p}_j$ are sufficiently similar for our pairing algorithm of Section 5.4.2 to group these points and compute their transformation $\mathbf{T}_{ij}$ as a sample point in transformation space Γ. At the same time this transformation is close to the unknown transformation $\mathbf{T}$, hence $\mathbf{T}_{ij}$ provides reliable evidence for the symmetry relation that we want to find. Thus if we choose a clustering radius h larger than $c_2\epsilon$ it follows that for any point $\mathbf{p}_i \in S$, our algorithm will deposit at least one point in Γ within distance h of $\mathbf{T}$. If m is the number

of points in $P \cap S$, then for any random sample from P we get a vote within h of $\mathbf{T}$ with probability $p = m/n$.

Using the Chernoff bound [MR95] we can show that if n' points are independently and randomly chosen from P, then with probability greater than $1 - \alpha$ there will be at least k points within h of $\mathbf{T}$ in Γ, where

$$k = \left(1 - \sqrt{-2 \log \alpha / n' p}\right) n' p$$

and $\alpha \in (0, 1)$. Until now we have shown that with high probability a cluster of height at least k that includes the transformation $\mathbf{T}$ will appear in Γ. To complete the analysis, we now need to ensure that this cluster is in fact a pronounced local maximum of the transformation density function and will thus be successfully retrieved by the mean-shift clustering algorithm. We prove this claim using a counting argument on Γ partitioned into a set of bins.

Suppose the average number of neighbors in Ω for a query radius of $\Delta\sigma$ is μ. Then n' random samples results in roughly $M = \mu n'$ points in Γ. Let the maximum extent along any dimension in Γ be L. So partitioning Γ using a grid of size $2h$ results in $N = (L/2h)^d$ bins, where d represents the dimension of Γ. It is easy to see that if there are more than k points within h of $\mathbf{T}$, at least one bin of Γ will contain at least $k/2^d$ samples. Assuming that point pairs that are not related by any meaningful symmetry relation map to a random bin in Γ, we observe that our scenario is identical to M balls being independently and uniformly thrown into N urns. It is known that the maximum number of balls in any urn with high probability is given by [Gon81, MS98]

$$E(n', \mu, \Delta\sigma) = \log N / \log(N \log N / M).$$

In order for the bin corresponding to $\mathbf{T}$ to stand above the noise level, we select n' such that the following inequality holds:

$$E(n', \mu, \Delta\sigma) < \left(1 - \sqrt{-2 \log \alpha / n' p}\right) n' p / 2^d. \tag{B.1}$$

Thus if we mark a bin in Γ as interesting only if its height is more than $k/2^d$ then: (1) with probability $1 - \alpha$ a bin corresponding to the desired transformation $\mathbf{T}$ is correctly marked,

and (2) only a few bins corresponding to spurious transforms are falsely marked. These outlier bins are easily pruned away in the verification step.

Requiring an ϵ-sampling with $\epsilon < 0.08$ for P is a fairly severe restriction. While algorithms exists for computing such a point set for a given smooth surface [BO03], the sampling density would be prohibitively high. As commented in [AB98], empirical evidence suggest that these bounds are quite conservative. In practice, we successfully found the existing symmetries using a much less restricted sampling. Consider the example of Figure 5.7, where $d = 6$, $L/h \approx 20$, and $\mu = 10$. If we hypothetically assume our initial point set P satisfies the sampling requirements stated above (which is clearly not the case since the surface is not even smooth), then Equation B.1 prescribes $n' \approx 300$ to detect a global symmetries where $p = 0.5$ with probability more than 95%. We found that even with only 100 samples we can reliably detect the global reflective symmetry of the castle (see Figure 5.7). Observe that as the size of the symmetric patch becomes smaller, i.e., p decreases, Equation B.1 suggests higher values for n'. Also note that while our analysis has been restricted to perfect symmetries, it can easily be extended to approximate symmetries by increasing h and thereby decreasing the number of bins N.

Bibliography

[AB98] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. In *Symposium on Computational Geometry*, pages 39–48, 1998.

[ABD+00] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *IEEE Trans. on Robotics and Automation*, pages 442–447, 2000.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of SIGGRAPH '98*, pages 415–421, 1998.

[ACDH06] E. Arias-Castro, D. L. Donoho, and X. Huo. Adaptive multiscale detection of filamentary structures in a background of uniform random points. *Annals of Statistics*, 34(1), 2006.

[ACP03] B. Allen, B. Curless, and Z. Popovic. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, 2003.

[ACSD+03] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph. (TOG)*, 22(3):485–493, 2003.

[AK04] N. Amenta and Y. Kil. Defining point-set surfaces. In *ACM Trans. Graph.*, volume 23(3), pages 264–270, 2004.

[AKKS99] M. Ankerst, G. Kastenmüller, H. P. Kriegel, and T. Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *SSD*, pages 207–228, 1999.

[Ale02] M. Alexa. Linear combination of transformations. In *Computer graphics and interactive techniques*, pages 380–387, 2002.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Journal of the ACM*, volume 45, pages 891–923, 1998.

[AMWW88] H. Alt, K. Mehlhorn, H. Wagener, and Emo Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, 3:237–256, 1988.

[ASK+04] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, H. Pang, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *NIPS*, 2004.

[Ata85] M. J. Atallah. On symmetry detection. *IEEE Trans. on Computers*, 34(7):663–666, 1985.

[BGK06] G. H. Bendels, M. Guthe, and R. Klein. Free-form modelling for surface inpainting. In *Afrigaph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 49–58, 2006.

[Blo70] B. Bloom. Space/time trade-offs in hash coding with allowable errors. In *Communications of the ACM*, volume 13(7), pages 422–426, 1970.

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3d shapes. In *IEEE Transactions on PAMI*, volume 14, pages 239–256, 1992.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[BMVS04] V. Blanz, A. Mehl, T. Vetter, and H.P. Seidel. A statistical method for robust 3d surface reconstruction from sparse data. In *Int. Symp. on 3D Data Processing, Visualization and Transmission*, 2004.

[BO03] J. D. Boissonnat and S. Oudot. Provably good surface sampling and approximation. In *Sympos. on Geometry Processing*, pages 9–18, 2003.

[Bro93] A. Z. Broder. Some applications of rabinffks fingerprinting method. In *Sequences II: Methods in Communications*, 1993.

[Bro97] A. Z. Broder. On the resemblance and containment of documents. In *Sequences*, 1997.

[Bro00] A. Z. Broder. Identifying and filtering near-duplicate documents. In *CPM*, pages 1–10, 2000.

[BS97] G. Barequet and M. Sharir. Partial surface and volume matching in three dimensions. *PAMI*, pages 929–948, 1997.

[BV99] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of SIGGRAPH '99*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of SIGGRAPH '01*, pages 67–76, 2001.

[CC94] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, London, 1994.

[CDD+04] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu. A finite element method for surface restoration with smooth boundary conditions. *Comput. Aided Geom. Des.*, 21(5):427–445, 2004.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH '96*, pages 303–312, 1996.

[CL01] C. Chan and H. Lu. Fingerprinting using polynomial (rabin's method). Technical report, University of Alberta, 2001.

[CLSB92] G. Champleboux, S. Lavallee, R. Szeliski, and L. Brunie. From accurate range imaging sensor calibration to accurate model-based 3d object localization. In *IEEE Conference of CVPR*, pages 83–89, 1992.

[CM91] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *IEEE Conference on Robotics and Automation*, 1991.

[CM02] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–609, 2002.

[CP03] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 177–187, 2003.

[CRT04] U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. In *IEEE Trans. Vis. Comp. Graphics*, 2004.

[CSM03] D. Cohen-Steiner and J. Morvan. Restricted delaunay triangulations and normal cycle. In *Symposium on Computational Geometry*, pages 312–321, 2003.

[CWPG04] D. Cotting, T. Weyrich, M. Pauly, and M. Gross. Robust watermarking of point-sampled geometry. In *Shape Modeling International*, 2004.

[CZCG04] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. In *Geometry Processing*, pages 127–138, 2004.

[DCOY03] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM Trans. Graph.*, 22(3):303–312, 2003.

[DG03] T. K. Dey and S. Goswami. Tight cocone: a water-tight surface reconstructor. In *Proceedings of Solid Modeling '03*, pages 127–134, 2003.

[DG04] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *Proc. 20th Symp. Comp. Geom.*, pages 330–339, 2004.

[DMGL02] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *3D Data Processing, Visualization, and Transmission*, 2002.

[Doo96] R.F. Doolittle. *Computer methods for macromolecular sequence analysis*. Academic Press, 1996.

[EDK06] I. Eckstein, M. Desbrun, and C.-C. Jay Kuo. Compression of time-varying isosurfaces. In *Graphics Interface*, page to appear, 2006.

[ELF97] D.W. Eggert, A. Lorusso, and R.B. Fisher. Estimating 3D rigid body transformations: a comparison of four major algorithms. In *Mach. Vision and Appl.*, pages 272–290, 1997.

[ESE06] E. Ezra, M. Sharir, and A. Efrat. On the icp algorithm. In *Symposium on Computational Geometry*, page to appear, 2006.

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with appl. to image analysis and automated cartography. *Comm. of the ACM*, pages 381–395, 1981.

[FDCO03] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graph.*, 22(3):950–953, 2003.

[Fel57] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley and Sons, 1957.

[Fit01] A. W. Fitzgibbon. Robust registration of 2d and 3d point sets. In *British Machine Vision Conference*, 2001.

[FKMS05] T. Funkhouser, M. Kazhdan, P. Min, and P. Shilane. Shape-based retrieval and analysis of 3d models. *Communications of the ACM*, pages 58–64, 2005.

[FKS+04] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Trans. Graph.*, 2004.

[FMK+03] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3d models. *TOG*, pages 83–105, 2003.

[FS06] T. Funkhouser and P. Shilane. Partial matching of 3d shapes with priority-driven search. In *Symposium on Geometry Processing*, pages 131–142, 2006.

[FZ04] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. In *Int. J. Comput. Vision*, volume 60(1), pages 5–24, 2004.

[GCO06] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. In *ACM Transactions on Graphics*, volume 25(1), pages 130–150, 2006.

[Gel06] N. Gelfand. *Feature Analysis and Registration of Scanned Surfaces*. PhD thesis, Stanford University, September 2006.

[GIRL03] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *3D Digital Imaging and Modeling*, 2003.

[GLB99] G. Godin, D. Laurendeau, and R. Bergevin. A method for regist. of attributed range images. In *3DIM*, 1999.

[GMGP05] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Symposium on Geometry Processing*, pages 197–206, 2005.

[Gon81] G. Gonnet. Expected length of the longest probe sequence in hash code searching. In *Journal of the Association for Computing Machinery*, pages 289–304, 1981.

[HB94] Y. Hecker and R. Bolle. On geometric hashing and the generalized hough transform. In *IEEE SMC*, volume 24, 1994.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *ACM SIGGRAPH*, pages 71–78, 1992.

[HFG+06] Q. Huang, S. Flory, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. In *Siggraph*, page to appear, 2006.

[HH03] D. Huber and M. Hebert. Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 21(7), 2003.

[Hor84] B. Horn. Extended gaussian images. In *Proc. of IEEE*, volume 72(12), pages 1671–1686, 1984.

[Hou59] P.V.C. Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, 1959.

[HPR04] M. Hofer, H. Pottmann, and B. Ravani. From curve design algorithms to the design of rigid body motions. In *The Visual Computer*, pages 279–297, 2004.

[HSW06] M. Hofer, G. Sapiro, and J. Wallner. Fair polyline networks for constrained smoothing of digital terrain elevation data. In *IEEE Trans. Geoscience and Remote Sensing*, page to appear, 2006.

[HU90] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. In *IJCV*, 1990.

[IMRV97] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *Symposium on Theory of Computing*, pages 618–625, 1997.

[JDD03] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph.*, 22(3):943–949, 2003.

[JH99] A. Johnson and M. Hebert. Using spin-images for efficient multiple model recognition in cluttered 3-d scenes. In *PAMI*, pages 433–449, 1999.

[JH03] T. Jost and H. Hugli. A multi-resolution ICP with heuristic closest point search for fast and robust 3d registration of range images. In *3D Digital Imaging and Modeling*, 2003.

[Joh97] A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Inst., Carnegie Mellon Univ., August 1997.

[JT05] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Trans. Graph. (TOG)*, 24(3):399–407, 2005.

[KCD+02] M. Kazhdan, B. Chazelle, D. P. Dobkin, A. Finkelstein, and T. Funkhouser. A reflective symmetry descriptor. In *ECCV*, pages 642–656, 2002.

[KCD+04] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser, and S. Rusinkiewicz. A reflective symmetry descriptor for 3d models. In *Algorithmica*, pages 201–225, 2004.

[KD92] J. Koenderink and A. Doorn. Surface shape and curvature scales. In *Image and Vision Computing*, 1992.

[Kel99] C. T. Kelley. *Iterative Methods for Optimization*. SIAM, Philadelphia, 1999.

[KFR03] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Symp. on Geometry Proc.*, pages 167–175, 2003.

[KFR04a] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Shape matching and anisotropy. *ACM TOG*, 23(3):623–629, 2004.

[KFR04b] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Symmetry descriptors and 3d shape matching. In *Sympos. on Geometry Processing*, pages 116–125, 2004.

[KHYS02] K. Kähler, J. Haber, H. Yamauchi, and H. Seidel. Head shop: generating animated head models with anatomical structure. In *Eurographics Symp. on Comp. Animation*, pages 55–63, 2002.

[Kle93] F. Klein. Vergleichende betrachtungen über neuere geometrische forschungen. *Mathematische Annalen*, 43, 1893.

[Koe90] J. Koenderink. *Solid shape*. MIT Press, 1990.

[Koe01] P. Koehl. Protein structure similarity. In *Current Opinion in Structural Biology*, pages 348–353, 2001.

[Kol05] R. Kolluri. Provably good moving least squares. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1008–1017, 2005.

[LBG06] X. Li, J. Barhak, and I. Guskov. Robust alignment of multi-view range data to cad model. In *SMI*, page to appear, 2006.

[LCT04] Y. Liu, R. Collins, and Y. Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. In *IEEE PAMI*, pages 354–371, 2004.

[LE06] G. Loy and J-O. Eklundh. Detecting symmetry and symmetric constellations of features. In *European Conference on Computer Vision*, page to appear, 2006.

[Lev01] B. Levy. Constrained texture mapping for polygonal meshes. In *Proceedings of SIGGRAPH '01*, pages 417–424, 2001.

[Lie03] P. Liepa. Filling holes in meshes. In *ACM SIGGRAPH/Eurographics Symp. on Geometry Processing*, pages 200–205. Eurographics Association, 2003.

[LPC+00] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, 2000.

[LPZ03] S. Leopoldseder, H. Pottmann, and H. Zhao. The d^2-tree: A hierarchical representation of the squared distance function. In *Tech.Rep. 101, Institut of Geometry, Vienna University of Technology*, 2003.

[LW88] Y. Lamdan and H. J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *ICCV*, pages 238–249, 1988.

[MGGP06] N. J. Mitra, L. Guibas, J. Giesen, and M. Pauly. Probabilistic fingerprints for shapes. In *Symposium on Geometry Processing*, pages 121–130, 2006.

[MGP06] N. J. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics*, volume 25(3), pages 560–568, 2006.

[MGPG04] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. Registration of point cloud data from a geometric optimization perspective. In *Symposium on Geometry Processing*, pages 23–31, 2004.

[MHYS04] S. Manay, B. Hong, A. J. Yezzi, and S. Soatto. Integral invariant signatures. In *ECCV*, pages 87–99, 2004.

[MKS04] W. Magnus, A. Karrass, and D. Solitar. *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations*. Dover, 2004.

[MKY01] F. Mokhtarian, N. Khalili, and P. Yuen. Multi-scale free-form 3-d object recognition using 3-d models. *Image and Vision Computing*, 19(5):271–281, 2001.

[MLP03] D. Mumford, A. Lee, and K. Pedersen. The nonlinear statistics of high-contrast patches in natural images. In *International Journal of Computer Vision*, volume 54, pages 83–103, 2003.

[MNG04] N. J. Mitra, A. Nguyen, and L. Guibas. Estimating surface normals in noisy point cloud data. In *special issue of International Journal of Computational Geometry and Applications*, volume 14, pages 261–276, 2004.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MS98] M. and A. Steger. "balls into bins" — A simple and tight analysis. *Lecture Notes in Computer Science*, 1998.

[OFCD02] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. In *ACM Transactions on Graphics*, volume 21(4), 2002.

[PGSQ06] S. Park, X. Guo, H. Shin, and H. Qin. Surface completion for shape and appearance. In *Visual Computing*, volume 22(3), 2006.

[PH03] H. Pottmann and M. Hofer. Geometry of the squared distance function to curves and surfaces. In *Visualization and Mathematics III*, pages 221–242, 2003.

[PHYH06] H. Pottmann, Q. X. Huang, Y. L. Yang, and S. M. Hu. Geometry and convergence analysis of algorithms for registration of 3d shapes. In *International Journal Computer Vision*, volume 67(3), pages 277–296, 2006.

[PKG03] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled models. In *Proceedings of Eurographics*, 2003.

[PKKG03] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3), 2003.

[PLH02] H. Pottmann, S. Leopoldseder, and M. Hofer. Registration without ICP. In *Technical Report 91, Institut für Geometrie, TU Wien*, 2002.

[PMG04] M. Pauly, N.J. Mitra, and L. Guibas. Uncertainty and variability in point cloud surface data. In *Symposium on Point-Based Graphics*, pages 77–84, 2004.

[PMG+05] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, pages 23–32, 2005.

[Pot04] H. Pottmann. Geometry and convergence analysis of registration algorithms. In *Tech.Rep. 117, Vienna University of Technology*, 2004.

[Pul99] K. Pulli. Multiview registration for large datasets. In *Proc. 3DIM*, 1999.

[RA99] R. Ramamoorthi and J. Arvo. Creating generative models from range images. In *Proceedings of SIGGRAPH '99*, pages 195–204. ACM Press/Addison-Wesley Publishing Co., 1999.

[Rab81] M. O. Rabin. Fingerprinting by random polynomials. In *Center for Research in Computing Tech., Harvard Univ., Report TR-15-81*, 1981.

[RL01] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3D Digital Imaging and Modeling*, 2001.

[RWP05] M. Reuter, F. Wolter, and N. Peinecke. Laplace-spectra as fingerprints for shape matching. In *SPM*, pages 101–106, 2005.

[SACO04] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Trans. Graph.*, 23(3), 2004.

[Sco79] D.W. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979.

[SGM98] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the web. In *Proceedings of Workshop on Web Databases*, 1998.

[SLW02] G.C. Sharp, S.W. Lee, and D.K. Wehe. Icp registration using invariant features. *PAMI*, 24(1):90–102, 2002.

[SP04] R. W. Sumner and J. Popovic. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.

[SS97] C. Sun and J. Sherrah. 3d symmetry detection using the extended gaussian image. *IEEE PAMI*, 19, 1997.

[SS00] R. H. Shumway and D. S. Stoffer, editors. *Time Series Analysis and Its Applications*. Springer Texts in Statistics, 2000.

[Ste96] M.J.E Sternberg. *Protein structure prediction*. Oxford Press, 1996.

[Sto87] G. Stockman. Object recognition and localization via pose clustering. *CVGI*, pages 361–387, 1987.

[Tau95] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH '95*, pages 351–358, 1995.

[Tho61] D. W. Thompson. *On Growth and Form*. Chambridge, 1961.

[TL94] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH '94*, pages 311–318, 1994.

[TSM05] O. Tuzel, R. Subbarao, and P. Meer. Simultaneous multiple 3d motion estimation via mode finding on lie groups. In *ICCV*, volume 1, pages 18–25, 2005.

[Tur92] G. Turk. Re-tiling polygonal surfaces. In *SIGGRAPH*, pages 55–64, 1992.

[TV04] J. Tangelder and R. Veltkamp. A survey of content based 3d shape retrieval methods. In *Shape Modeling International*, 2004.

[VCBS03] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro. Inpainting surface holes. In *Int. Conference on Image Processing*, 2003.

[VH00] R. C. Veltkamp and M. Hagedoorn. Shape similarity measures, properties and constructions. In *VISUAL '00: Proceedings of the 4th International Conference on Advances in Visual Information Systems*, pages 467–476, 2000.

[WC05] H. Wu and Y. Cheung. A fragile watermarking scheme for 3d meshes. In *Multimedia and Security*, pages 117–124, 2005.

[WPH+04] T. Weyrich, M. Pauly, S. Heinzle, R. Keiser, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Symposium on Point-Based Graphics*, 2004.

[WR97] H. Wolfson and I. Rigoutsos. Geometric hashing: an overview. In *IEEE Comp. Sci. and Eng.*, pages 10–21, 1997.

[WWV85] J. D. Wolter, T. C. Woo, and R. A. Volz. Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer*, 1985.

[ZMCF05] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast. Dynamic response for motion capture animation. In *ACM Trans. Graph.*, volume 24(3), pages 697–701, 2005.

[ZPA95] H. Zabrodsky, S. Peleg, and D. Avnir. Symmetry as a continuous feature. *IEEE PAMI*, 17, 1995.