

FEATURE ANALYSIS AND REGISTRATION
OF SCANNED SURFACES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Natasha Gelfand
September 2006

© Copyright by Natasha Gelfand 2006
All Rights Reserved

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 Co-Advisor

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 Principal Co-Advisor

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.

Sebastian Thrun

Approved for the University Committee on Graduate Studies.

Abstract

In the last decade there have been significant technological advances in the design of tools for digitizing 3D shape of objects, leading to large repositories of 3D data, and the need to develop efficient algorithms to process and analyze scanned 3D shapes.

Most 3D scanners produce as their output a set of point samples corresponding to the shape of the digitized object as seen from a single viewpoint. To digitize the entire shape, several scanning passes are required, with either the scanner or the object being re-positioned between the scans. The problem of *shape registration* deals with computing the relative transformations between the scans to bring all scans into a common coordinate system. In this thesis we present several algorithms for registration of scanned surfaces. Our algorithms are based on analyzing local and global surface properties of the input shapes to improve the convergence of registration and the quality of the computed alignment.

First, we present an algorithm for automatic approximate alignment of two partially overlapping 3D shapes (data and model) without any assumption about their initial positions. The algorithm uses the distribution of values of a robust shape descriptor to select a set of feature points on the data shape and compute their potential corresponding points on the model shape. The resulting correspondence search space is explored using an efficient branch and bound algorithm based on distance matrix comparisons, and the best set of corresponding point pairs is used to compute the aligning transformation. The resulting alignment algorithm is used for registration of partially overlapping scanned surfaces, for simple symmetry detection, and for matching and segmentation of shapes undergoing articulated motion.

Next, we develop a surface descriptor based on surface self-similarity under continuous rigid motion, called *surface slippage*. We show that by analyzing a certain matrix computed from the point positions and surface normals of a pointset, one can differentiate among pointsets that correspond to planes, spheres, surfaces of revolution and surfaces of linear extrusion. We use the slippage descriptor to develop a segmentation algorithm for reverse engineering surfaces of mechanical parts, a problem that is frequently encountered in the field of Computer Aided Design.

Finally we apply surface slippage in the context of the Iterated Closest Point (ICP) algorithm, which is a widely used method for local registration of two 3D shapes. The quality of the alignment obtained by this algorithm depends heavily on choosing good corresponding points from the two datasets. If too many points are chosen from featureless regions of the data, ICP can converge slowly or find the wrong pose, especially in the presence of noise. Performing slippage analysis on the input shapes allows us to select a set of features on the input that minimizes the uncertainty in the final pose and improves the algorithm's convergence.

Acknowledgments

My sincere thanks to my advisors Leonidas Guibas and Marc Levoy for their support and inspiration during my stay at Stanford. I feel truly lucky to have been able to learn from them and work with them on a variety of exciting topics.

I would like to thank my orals committee, consisting of Sebastian Thrun, Helmut Pottmann and Bernd Girod. Helmut has also been a wonderful collaborator who always brought unbridled enthusiasm and some solid mathematics to all our joint projects.

I want to thank my co-author, collaborator, climbing partner, conference roommate, and good friend Niloy Mitra for his omnipresent good cheer.

My officemates over the years have always been a source of support, research ideas, debugging help, paper proofreading, funny quotes and time-wasting diversions. A special thanks goes to Daniel Russel, with whom I've shared the same office for all but our first year at Stanford. Thanks to my officemates James Davis, Erika Chuang, Brad Johanson, Jie Gao, Qing Fang, Mark Pauly, An Nguyen, and Maks Ovsjanikov. Thanks to the members of the Geometry group and the the Stanford Graphics Lab for making the lab a great place to do research.

Finally, thanks to my husband for being my biggest fan.

Contents

Abstract	iv
Acknowledgments	vi
1 Introduction	1
1.1 Shape Similarity in Geometry Processing	1
1.2 Range Image Registration	6
1.3 Shape Descriptors	9
1.4 Feature Point Selection	10
1.5 Self-Similarity and Segmentation	12
1.6 Overview of Thesis	13
1.6.1 Contributions	13
1.6.2 Outline	14
2 Prior Work	15
2.1 Overview	15
2.2 Pairwise Registration	17
2.3 Global Pairwise Registration	19
2.3.1 Transformation Search Methods	19
2.3.2 Correspondence Search Methods	20
2.3.3 Shape Descriptors	21
2.4 Local Pairwise Registration	23
2.4.1 Distance Metrics	24

2.4.2	Closest Point Search	25
2.5	Multi-View Registration	26
2.6	Summary	28
3	Global Registration Using Combinatorial Search	29
3.1	Integral Invariants	31
3.1.1	Integral Volume Invariant	32
3.1.2	Implementation	34
3.2	Feature Point Selection	35
3.2.1	Basic Algorithm	37
3.2.2	Scale-space Representation and Persistent Features	38
3.3	Distance Metric Based on Intrinsic Shape Properties	40
3.4	Matching Algorithm	42
3.4.1	Initial Correspondence Set	42
3.4.2	Initial Bound	44
3.4.3	Branch-and-bound Search	46
3.4.4	Partial Matching	47
3.5	Results	48
3.5.1	Object Registration	48
3.5.2	Symmetry Detection	50
3.5.3	Articulated Matching	51
3.6	Summary	54
4	Local Surface Slippage	56
4.1	Rigid Motions	57
4.2	Surface Slippage	57
4.3	Computing Slippable Motions	58
4.4	Application: Shape Segmentation Using Local Slippage Analysis	62
4.4.1	Segmentation	62
4.4.2	Point Classification	64
4.4.3	Similarity score	65
4.5	Robust Segmentation Algorithm	69

4.5.1	Multi-pass segmentation algorithm	69
4.5.2	Initial patch size selection	72
4.5.3	Post-processing steps	73
4.6	Segmentation Results	73
4.7	Summary	77
5	Stable Sampling for Local Registration	78
5.1	Geometric Constraints for Local Registration	79
5.1.1	Stability of the Solution	81
5.2	Improving ICP's Stability Through Sample Selection	82
5.2.1	A Measure of Stability	83
5.2.2	Optimizing the Measure	84
5.3	Accelerations and Enhancements	85
5.4	Results	87
5.4.1	Using Stable Sampling for Pairwise Registration	87
5.4.2	Using Stable Sampling in a Multi-view Registration System	90
5.5	Limitations	91
5.6	Summary	94
6	Conclusions and Future Work	95
6.1	Summary	95
6.2	Understanding Registration Algorithms	97
6.3	Shape Similarity Benchmarks	98
6.4	Benchmarking Registration Algorithms	100
6.5	Future Work	104
	Bibliography	105
A	Properties of the Integral Volume Invariant	115
A.1	Relation between curvature and the integral area invariant of a 2D curve	115
A.2	Mean curvature and the volume descriptor	117
A.3	Influence of a surface perturbation on the volume descriptor	118

B	Bounds on cRMS and dRMS Error Metrics	120
B.1	Upper bound	120
B.2	Lower bound	121

List of Tables

- 3.1 Global registration examples summary 49
- 4.1 Kinematic surfaces 61

List of Figures

1.1	Shape retrieval from a database.	2
1.2	Object recognition problem.	4
1.3	Range image registration.	6
1.4	Correspondence problem in 3D registration.	9
2.1	Pairwise registration problem	17
2.2	Influence of distance metric on the convergence funnel of ICP.	24
2.3	Closest point search using descriptors.	26
3.1	Illustration of the volume integral descriptor in 2D.	33
3.2	Computing the integral volume invariant in the presence of holes.	35
3.3	Volume descriptor.	36
3.4	Persistent feature selection.	40
3.5	Dragon registration results	50
3.6	David registration results	51
3.7	Bunny registration results	52
3.8	Symmetry detection using global registration	53
3.9	Articulated matching using global registration	53
4.1	Examples of mechanical parts composed of kinematic surfaces.	62
4.2	Coloring of point of a shape consisting of kinematic surfaces	66
4.3	Segmentation uncertainty at early iterations.	68
4.4	Segmentation of a simple model into slippable components.	74
4.5	Segmentation without sharp edges.	75

4.6	Segmentation of a mechanical parts consisting of cylinders and surfaces of revolution	75
4.7	Segmentation of a part consisting of planar and cylindrical components. . .	76
5.1	Incised plane registration example.	88
5.2	Points picked by the stable sampling algorithm	88
5.3	Incised spheres registration example.	89
5.4	Aligning two scans of Forma Urbis Romae fragment 033abc	90
5.5	Multi-view registration using stable sampling.	91
5.6	Dicing-based nonrigid registration using stability analysis	92
5.7	Effect of noise on covariance sampling.	93

Chapter 1

Introduction

"Before beginning, plan carefully."

–Marcus Tullius Cicero

1.1 Shape Similarity in Geometry Processing

In the last decade there have been significant technological advances in the design of tools for digitizing and modeling 3D shape of objects. Commercial laser range finders, available from companies such as Minolta, Cyberware, and others, are able to capture the geometric shape of existing objects at resolutions of up to 0.25mm. Alternatively, 3D shapes can be created virtually by using a 3D modeling or CAD tools such as zBrush or 3D Studio.

As a result, 3D content creation and use have become almost routine, leading to large repositories of 3D data. 3D geometry is now emerging as a new form of digital content and taking its place along side more traditional digital media such as sound, images, and video. The area of digital geometry processing aims to develop effective and efficient algorithms for the problems of reconstruction, representation, modification, retrieval and analysis of 3D models.

A fundamental problem that is commonly encountered in many digital geometry applications is shape matching, or determining which shapes or parts of shapes are similar. Examples of shape processing problems which make use of shape similarity analysis are:

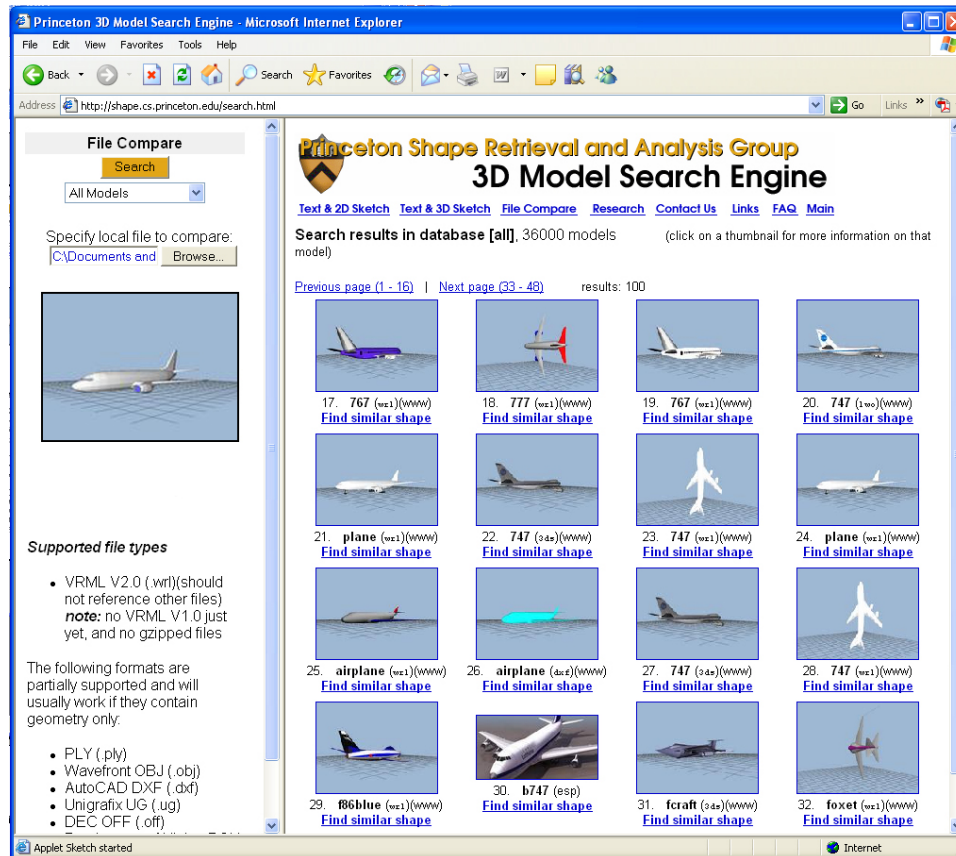


Figure 1.1: Shape retrieval from a database. The shapes in the database are compared to a query model, and the most similar shapes are displayed to the user. (Image courtesy <http://shape.cs.princeton.edu>)

- **Database shape retrieval.** 3D content generated by 3D scanning and modeling tools are often stored in basic online 3D shape repositories. Some, such as 3D Cafe [15], TurboSquid [100], and the like are collections of general 3D objects, usually created by a modeling tool. Others are specialized databases, such as the Protein Data Bank [6], the CAESAR dataset of human shapes, and repositories of mechanical parts [83, 81]. Others still are repositories of scanned data, such as Stanford's 3D model archive [82].

Most of these repositories are accessed just by browsing through images captured from the stored 3D geometry. Retrieving models from such databases in an intelligent way is the quintessential shape matching task. Given a query model, its shape is compared to the shape all models stored in the database, and the results are ranked and returned to the user in order of decreasing similarity. Of particular note is the development of a search engine for 3D models at Princeton [35], which aims at providing a tool for searching 3D content similar to text search engines for the WWW. An example shape retrieval application is shown in Figure 1.1.

- **Shape segmentation.** Shape segmentation deals with decomposing a single object into a set of meaningful components, usually represented by surface patches. What constitutes a component is often application specific, but frequently the goal is to create a set of patches where points within a patch are similar to each other. This is the case when we want to approximate the shape by a set of planes [21], developable surfaces [76], quadric surfaces [57], etc. Shape segmentation can be viewed as a problem in shape similarity, where the shape comparisons are performed among different parts of a single shape, instead of among several different shapes.
- **Model repair.** Often digitized 3D models contain artifacts such as noise and holes, which are the result of difficulties during the capture process, such as scanner limitations or unfavorable surface properties of the acquired objects. Post-processing such data to obtain pleasing 3D models requires de-noising [32, 51] to smooth out the acquisition noise and hole-filling [24] to interpolate across the missing surfaces. Most of the early methods for model repair and improvement aimed at creating a smooth manifold surface in the areas of noisy or missing data by using local information around the damaged area to produce the new geometry. Recently, several new approaches to model repair have been proposed that improve the quality of the model in a given area by using information from similar 3D shapes identified by a shape matching algorithm. The de-noising method of Yoshizawa et al. [109] performs surface smoothing by both classic point averaging over a local neighborhood, as well as averaging the noisy surface with other similar parts of the same model. This method successfully removed scanner noise while preserving more true model features than

earlier methods based on just local operations. Recent hole filling methods have also benefited from shape similarity analysis by copying geometry from similar looking areas of the same model [92], or of other similar looking shapes [73, 56] to create a hole filling surface.

- **Range image registration.** Most scanning devices produce as their output a raw set of point samples. Due to self-occlusions of the scanned object, and the field of view limitations of the scanning device, only a partial view of the object can be acquired at a time. To digitize the entire shape, several scanning passes are required with either the scanner or the object being moved in between the scans. *Scan alignment* or *registration* algorithms bring a set of scans into a common coordinate system so that a shape reconstruction algorithm can be applied to produce the complete digitized object as shown in Figure 1.3. The majority of registration algorithms operate by identifying similar areas in different scans and computing a transformation that positions the scans such that the matching areas overlap with each other.

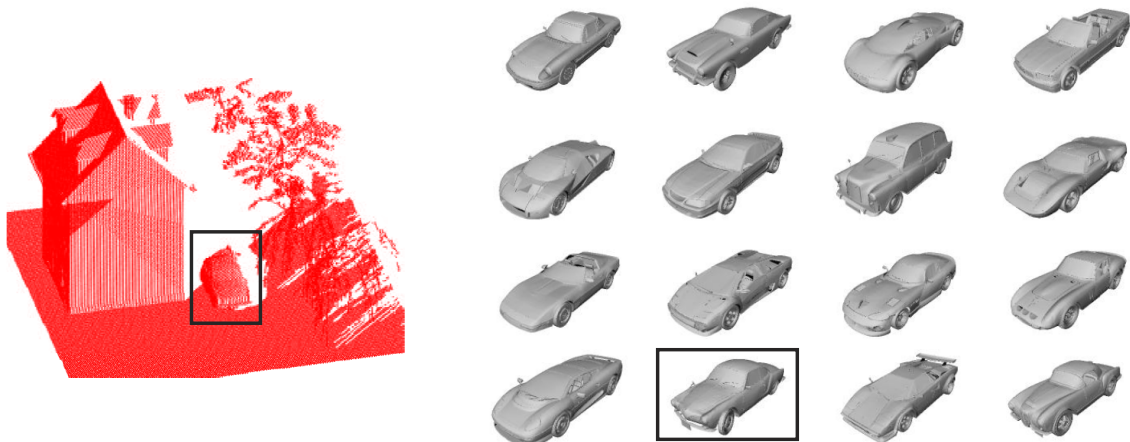


Figure 1.2: An example of an object recognition problem. Given a scan of a cluttered scene (left) and a database of car models (right), the goal is to identify which model of car is present in the scene and identify its location. (Images courtesy of [8])

- **Object recognition.** Recognizing specific objects in noisy and cluttered scenes is a

challenging problem encountered in computer vision. Given scan of a scene and a database of candidate 3D models (e.g. people, cars), the goal is to identify if any of the models in the database is present in the scene, and to identify their position and orientation (Figure 1.2).

The last two applications described above have a more restricted structure than a general shape matching problem. Instead of looking for approximate similarity of entire objects, as we would do in a shape retrieval application, we instead know that the shapes (or parts of shapes) are related by a transformation from a certain transformation class (e.g. rigid, affine, or articulated). The goal is to discover the parts of the input shapes that are the same, and to recover the transformation that positions the objects such that the distance between the overlapping parts is minimized.

Shape registration is both easier and harder than the general shape matching problem. On the one hand, we have extra information available to us in a registration problem. For example, we can use the knowledge that the overlapping scans have to create an approximately manifold surface, and that matching a scan into the space that was seen as unoccupied from some other view is very unlikely, to discard erroneous matches between pairs of scans [47, 104]. On a more local scale, we know that pairwise distances between points on one shape and their matching points on the other shape need to be preserved. Therefore, when looking for overlapping areas of two input shapes, we can discard from consideration areas which do not satisfy the rigidity preservation constraint.

On the other hand, shape registration is a problem in *partial matching*, since the inputs generally do not overlap over their entire extents. Partial matching is more difficult than the global *total matching* problem, since we cannot use the usual global shape properties such as center of mass, principal axes orientation, moments etc, which have been very successful in helping solve the general shape matching problem. Again, shape segmentation can be also thought of as a partial matching problem, where the matching parts both come from the same shape.

In this thesis, we focus on developing and analyzing local and global surface measures that are helpful in solving the problems of partial matching and range image registration under

rigid transformations. In the rest of this chapter, we examine in more detail the various sub-problems encountered in range image registration and related partial shape similarity problems, and give an overview of our contributions.

1.2 Range Image Registration



Figure 1.3: Range image registration. Left: a scan is acquired from a single viewpoint. Middle: To cover the entire object, multiple viewpoints are necessary. Right: views registered into final model. (Images courtesy Digital Michelangelo Project [58])

Most shape acquisition devices produce a set of discrete samples that give the depth information about the given object as seen from a single viewpoint. The resulting set of depth samples is called a *range image*, which is similar to a regular image, in that every sample represents depth (instead of color as in a regular image) at a given point on the object. Due to limited field of view of most scanners, and the self-occlusions that occur in most objects, several scans (also called views) are required to capture the entire object surface. Either the object or the scanner is moved between successive views. Since the points in each range image are usually given in the local coordinate system of the scanning device, a transformation of each view into the world coordinate system needs to be computed to build the complete model of the scanned object (Figure 1.3).

One way to recover the transformation for each range image is to track the motion of the object as it is being scanned. This can be done, for example, by placing it onto a calibrated turntable, or tracking the motion of the scanner's head as it scans different sides

of the object. However, this either restricts the scanning to small objects that can fit onto a turntable, or requires a complicated scanner design. Another approach, then, is to include a certain amount of overlapping area between successive views of the object. After the object has been completely scanned, these overlapping areas are used to recover the relative transformation between pairs of views. By applying this process for all pairs of overlapping scans, the entire object is built up. Range image registration algorithms find the overlapping areas between pairs of scans, compute a transformation that best aligns the overlapping areas, and bring all the scans into a common coordinate system.

A registration algorithm that computes a relative transformation between a pair of shapes, usually called the *model* and the *data*, should have the following properties:

- **Initialization invariance.** We would like to be able to find the optimal alignment between two shapes independently of their initial position. This will allow for automatic positioning of the scans as they are acquired.
- **Tolerance to partial matches.** Most of the time, only a subset of points in the model will match to a subset of points in the data. The overlap area is usually unknown, although we may have an estimate of what percentage of the surfaces overlap each other. Determining the actual areas of overlap is one of the goals of a pairwise registration algorithm.
- **No user input.** Many matching and algorithms rely on the presence of user-specified markers on the objects, or on an initial rough manual positioning of the surfaces. We would like our algorithms to be completely automatic.
- **Fast convergence.** Although not necessarily real-time, it is useful to be able to find the alignment between the objects quickly. This way, a feedback can be provided to the user about the progress of the scanning.

Existing range image registration algorithms focus most of their efforts in analyzing the input shapes to find the best set of corresponding points, which are then used to bring the two shapes into alignment (see Chapter 2 for an overview of registration techniques). Search for correspondences implicitly means that all of the heavy processing is done on pairs of

shapes. Although registration requires two inputs, we will show that it is often helpful to analyze each input shape separately first. The motivation of this approach is two-fold. First, in a complete range image registration system for a given set of scans, many pairs of scans are matched using a pairwise registration algorithm, in fact several early registration pipelines [47, 58] have been brute-force and performed a pairwise matching between all pairs of scans. Since each scan participates in several pairwise matches, performing expensive analysis of each scan is less expensive than performing analysis of pairs of scans, especially if it simplifies the pairwise matching problem. Second, we can make use of the simple observation that since the model and data shapes in each pairwise registration problem are similar over some part of their extent, we can gain insight on how the data shape will behave during registration with some model shape, without actually having to look at the model.

In the first part of the thesis, we develop an algorithm for registration of overlapping scans that is independent of the scans' initial pose. Under some reasonable assumptions about the shape of the input scans and the amount of overlap between them, our algorithm is guaranteed to find a close estimate of their correct relative pose, without any use of markers or user guidance. The shape properties used to make registration effective include the distribution of shape descriptors on the data shape, and the matrix of pairwise point distances of the data and model shapes.

In the second part of this thesis, we examine how a given shape behaves under a rigid motion, namely whether there exist certain rigid motions under which a given shape is invariant. This will allow us to develop an intrinsic surface property which is useful for detecting small features in otherwise featureless areas of the input scans. By adapting a well known local registration algorithm to sample points from constraining features, we obtain an algorithm which is able to match such difficult to register shapes effectively.

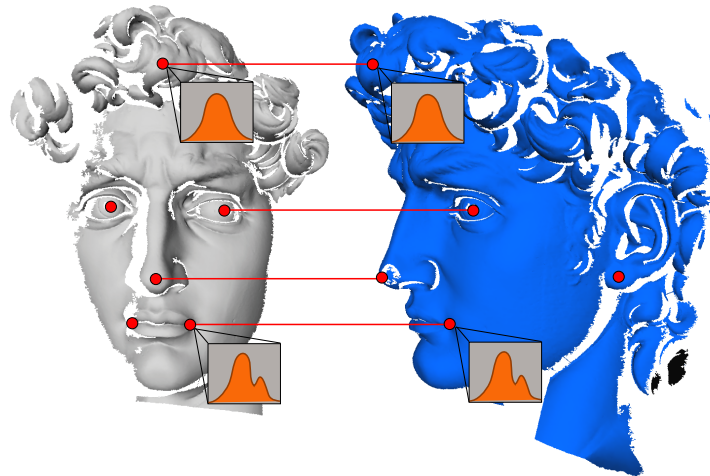


Figure 1.4: Finding corresponding points using shape descriptors. Since it's usually too expensive to compute correspondences for each point, a set of features (shown as red points) are selected from both shapes. A hypothetical shape descriptor (shown as a plot of the values) is computed for each point, and points whose shape descriptors are similar are said to be in correspondence (shown as red lines).

1.3 Shape Descriptors

Shape registration can be related to the classic problem in computer vision: a search for correspondences. Given two views of the same object, the goal is to find correspondences between the views and find a transformation that brings the two views into alignment (Figure 1.4).

In computer vision the correspondence problem is commonly solved using descriptors and feature tracking [105]. Similar approaches are applied in 3D shape processing. Given a point of the input shape, a descriptor is a quantity computed based on the surface around this point that captures the local geometric information in a way that allows for easy comparison. Usually, a geometric shape descriptor represents the local surface around each point by a fixed dimensional vector. Determining the similarity between two points on a shape amounts to computing the Euclidean distance between two descriptor vectors. Points in the two views with similar descriptors are potentially in correspondence (Figure 1.4). To be useful for computing correspondences in registration algorithms, geometric descriptors

need to have the following properties:

- **Transformation invariance.** The shape descriptor should not change when a transformation is applied to the input shape. There is a large number of shape descriptors that are invariant under rigid transformations (reviewed in more detail in Section 2.3.3), while invariance under affine and articulated deformations is harder to achieve.
- **Locality.** The shape descriptor should not rely on global properties of the shape, such as center of mass, principal axes orientation etc. In registration, two views usually overlap only over part of their extent, which precludes the use of global shape properties as it is often done in shape matching. Instead, shape descriptors should be computed based on local neighborhood information around each point.

Examples of shape descriptors include low-dimensional properties such as curvatures, and curvature-based quantities such as shape index [54] and curvedness [55], and high-dimensional descriptors such as spin images [50] and shape contexts [8]. Low dimensional descriptors are usually easier to compute and compare, while higher dimensional descriptors provide more discriminating power.

We develop a new shape descriptor, related to local mean curvature, based on integration of a spatial function over a kernel centered at each surface point. As opposed to the differential invariants such as curvature, our integration-based descriptor is more robust to noise. This descriptor is used in a registration algorithm to help pick potential feature points on the input surfaces and identify corresponding points for those features.

1.4 Feature Point Selection

In most real-world scanning applications the size of the input data is too large to search for a corresponding point on the model shape for each point on the data shape, especially if this search process is expensive. To simplify the problem, a subset of points on the input shapes are chosen to be used for correspondence computation, these points are called *features*

(Figure 1.4). Feature points are usually chosen in some way that will make correspondence computation easier, such as from some salient or unique areas of the input shapes.

The "feature analysis" part of this thesis will focus on examining various characteristics of 3D shapes when they are used as inputs for pairwise registration. In particular, we will try to identify the properties and points on each input shape that will make it easier (or harder) to process it in a registration algorithm. By performing such feature analysis the actual pairwise registration algorithms can remain fairly simple but still effective in achieving good alignment results.

Feature point selection usually requires analyzing both local surface properties around each point, such as shape descriptors, and also more global properties of the whole shape. Most of the time features are selected independently on the data and model shapes, since selecting them on both shapes together requires the solution to the underlying correspondence problem. Therefore, a major challenge for feature selection algorithms is to pick a consistent or canonical set of points on each shape. Since only the selected feature points are used for computing the alignment of the model and data, if the two sets are inconsistent it will be impossible to find correspondences between them. The consistency requirement often results in algorithms that either have to over-sample the two shapes to guarantee that some corresponding points will be included in both sets or perform complicated analysis to identify the salient points on the two shapes and rely on the assumption that the same point will be salient in all inputs [60, 67]. Other approaches such as [99] analyze the model and data together and pick consistent pairs of feature points, which is an expensive method since it implicitly has to solve the correspondence problem.

We present two feature selection methods in this thesis that are aimed at simplifying the underlying correspondence and registration problems. The first method analyzes the distribution of shape descriptors on the surface of a shape and selects points whose descriptor values are rare. This results in a set of salient feature points in the data which are likely to have only few correspondences in the model, making it easy to find the best set of correspondences. The second feature selecting method is based on analyzing invariance of the surface under rigid motion. We select feature points on the data shape which will best

constrain its alignment to the model shape, which results in improved convergence of registration algorithms.

1.5 Self-Similarity and Segmentation

Until now, the shape similarity problem under consideration always involved two shapes which were matched to each other. We can also think of studying how a shape is similar to itself, for example whether there exists more than one way to register a shape to a copy of itself, or whether a shape is composed of several similar components. Understanding how a shape matches to a copy of itself can help us understand how it matches to another different shape. We can draw an analogy here with well known string matching algorithms such as Knuth-Morris-Pratt or Boyer-Moore [22], where the pattern string is first matched against itself to understand its structure, and only then is matched against the text string.

Shape self-similarity can be used to determine whether a shape possesses any symmetries. If a shape is symmetric, there will be several transformations or groups of transformations under which the distance between a shape and its transformed copy is small. Therefore, a global registration algorithm can be used to determine whole-object symmetries. Recently, more sophisticated algorithms for determining partial symmetries (effectively self-similarity of parts of the same shape) have also been developed by Podolak et al. [74] and Mitra et al. [65].

Object segmentation can also be thought of as a self-similarity problem, where the goal is to decompose an object into a set of components such that points within each component are similar to each other with respect to some measure. Therefore studying feature detection for pairwise matching problems such as registration also results in a set of shape properties that can be directly applied in a useful segmentation algorithm. It is not surprising, therefore, that many segmentation algorithms are descriptor based, and in fact the same descriptors that are used for registration have been successfully applied to segmentation problems. For example curvature based measures have been used in [55] for segmentation of general shapes along sharp edges.

One of the surface properties which allows us to develop a better registration algorithm, namely the distribution of points and normals across the surface of an object, can also result in an effective segmentation algorithm for decomposing the input into a set of surfaces corresponding to planes, spheres, surfaces of revolution and surfaces of linear extrusion. This approach can be particularly useful for reverse-engineering of mechanical-type objects, a problem that is frequently encountered in the field of Computer Aided Design.

1.6 Overview of Thesis

1.6.1 Contributions

In this dissertation we explore the use of local shape descriptors and intrinsic shape properties to develop more efficient and robust algorithms for shape registration and segmentation. The contributions of this work are as follows:

- We describe a new surface descriptor called *integral volume invariant* which is related to mean curvature but computed by performing integral instead of differential operations on the surface.
- We develop a feature selection method based on persistence [27] and distribution of values of the integral volume invariant values across the surface of a 3D shape. The feature selection method effectively captures a small set of geometrically salient points on the input surface.
- The feature selection method is used in a global registration algorithm for aligning two shapes without a prior estimate of their relative transformation. The algorithm explores the space of potential alignments using an efficient branch and bound technique, and is guaranteed to find the best set of correspondences for the selected feature points.
- We develop a new shape descriptor based on surface invariance under a rigid motion, called *surface slippage*. Using this shape descriptor in a feature selection algorithm,

we improve the performance of a well known local registration method, and achieve good registration results in cases that have been problematic for previous methods, namely when two shapes are largely self-similar with only a few constraining features.

- The surface slippage descriptor leads to a way to describe local continuous self-similarity of a surface. Since shape segmentation can be thought of as grouping areas of a surface into similar patches, we can apply the slippage shape descriptor in a segmentation algorithm. This results in a segmentation algorithm for decomposing a shape into pointsets that correspond to kinematic surfaces (planes, cylinders, surfaces of revolution and surfaces of linear extrusion). The new segmentation algorithm is applied to reverse engineering of models of mechanical parts.

1.6.2 Outline

The rest of this thesis is organized as follows:

- In Chapter 2, we review the current literature on shape registration. We will pay particular attention to the various uses of shape descriptors for feature selection and correspondence computation.
- In Chapter 3, we describe our global registration algorithm based on fast combinatorial search of the space of all corresponding points [39].
- In Chapter 4, we study local self-similarity of the surface and develop the slippage shape descriptor. We take a short aside from the registration problem to describe the application of slippage to reverse engineering of models of mechanical parts [37].
- We return to registration in Chapter 5, and describe the use of the slippage property in a local registration algorithm to register inputs which have only a sparse set of feature points [38].
- Chapter 6 presents a summary and directions for future work.

Chapter 2

Prior Work

"If you want to make an apple pie from scratch, you must first create the universe."

–Carl Sagan

2.1 Overview

This chapter provides a survey of previous work on shape registration and its use in 3D scanning applications. A survey of more general shape matching techniques can be found in [103]. As described in the previous chapter, while shape matching addresses the problem of general shape similarity, shape registration is a partial matching problem and has an additional requirement of finding the best transformation that aligns (or superimposes) the shapes.

Broadly speaking, there are two classes of registration problems. *Pairwise registration* deals with finding a transformation that best aligns two scans, while *multiview registration* processes larger groups, in most cases all the scans for the object. The general pairwise registration problem is usually stated as follows. Given two input shapes **P** and **Q**, usually

called the data and the model, and a transformation class \mathcal{A} , we are looking for a transformation $\alpha \in \mathcal{A}$ that, when applied to \mathbf{P} , minimizes the distance between $\alpha(\mathbf{P})$ and \mathbf{Q} (see Figure 2.1):

$$\mathcal{E}(\mathbf{P}, \mathbf{Q}) = \min_{\alpha} d^2(\alpha(\mathbf{P}), \mathbf{Q}) \quad (2.1)$$

In range image registration \mathcal{A} is usually the group of rigid transformations and d^2 is a measure of distance between a pair of surfaces. Since the model and data scans overlap only partially, the distance should be computed only over this initially unknown overlap area. The quality of the registration is measured as the residual distance after applying the best transformation.

It is possible to assemble the entire object out of several scans by just applying a pairwise registration algorithm. The first scan is used as an anchor and each new scan can be registered to some existing scan that is already part of the assembled group. This process, however, tends to produce poor results since pairwise errors tend to accumulate. For example, given a "ring" of scans registered pairwise in sequence, the first and last scan usually do not match up well. To address this problem, *multiview registration* algorithms have been developed. Given K partial scans of an object $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_K$, the goal is to find K transformations $\alpha_1, \alpha_2, \dots, \alpha_K$ (where α_1 is usually the identity) such that the following residual is minimized:

$$\mathcal{E}(\mathbf{P}_1, \dots, \mathbf{P}_K) = \min_{\alpha_1, \dots, \alpha_K} \sum_{i=1}^{K-1} \sum_{j=i+1}^K d^2(\alpha_i(\mathbf{P}_i), \alpha_j(\mathbf{P}_j)). \quad (2.2)$$

That is, we want to minimize total distance between all overlapping scans after applying all the transformations (we assume that if a pair of scans do not overlap, their contribution to the total alignment error is 0). The additional challenge in multi-view registration is to decide which pairs of scans to include in the system in Equation 2.2. Most multi-view registration algorithms first pre-align the scans pairwise as described above and then try to adjust the resulting transformations to minimize the total registration error over all pairs

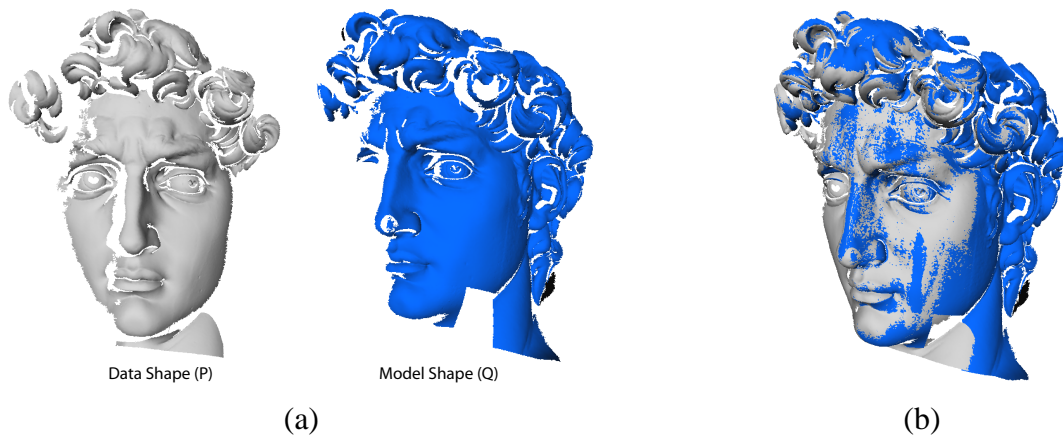


Figure 2.1: Pairwise registration problem. (a) Data shape (\mathbf{P}) and Model shape (\mathbf{Q}) in some arbitrary initial positions. (b) Result of registration: model and data are positioned so that matching parts overlap (as seen from the gray and blue surfaces inter-penetrating in the area of the overlap).

of scans. Pairwise registration has received more attention in the literature than multiview registration, since a good pairwise alignment results in a better-behaved multiview system. Below, we give a detailed review of existing techniques for pairwise registration, and a short overview of multiview registration algorithms.

2.2 Pairwise Registration

Pairwise shape registration can be decomposed of two subproblems: *correspondence* and *alignment*. The correspondence problem in shape registration is similar to correspondence problem in computer vision [105], we want to identify parts of \mathbf{P} and \mathbf{Q} that represent the same physical part of the scanned object. The alignment problem then is to minimize the distance between the corresponding parts.

In the rest of this section, we will assume that the input shapes are point-sampled, with \mathbf{P} consisting on N points $\{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ and \mathbf{Q} consisting of M points $\mathbf{q}_1, \dots, \mathbf{q}_M$. We indicate that a pair of points from \mathbf{P} and \mathbf{Q} are in correspondence by assigning to \mathbf{p}_i a corresponding

point \mathbf{q}_{c_i} in \mathbf{Q} . The goal of a pairwise registration algorithm is to find a set of correspondence variables $\{c_1, c_2, \dots, c_k\}$ and parameters of a rigid transformation \mathbf{R}, \mathbf{t} , that minimize:

$$\mathcal{E} = \min_{\mathbf{R}, \mathbf{t}, c_1, \dots, c_k} \sum_{i=1}^k d^2(\mathbf{R}\mathbf{p}_i + \mathbf{t}, \mathbf{q}_{c_i}). \quad (2.3)$$

Correspondence and alignment suffer from a "chicken and egg" relationship. If the correspondences $\{c_1, \dots, c_k\}$ are given, it is usually trivial to find the correct alignment. In particular, it is well known that given a set of corresponding point-pairs $(\mathbf{p}_i, \mathbf{q}_{c_i})$, we can find a rigid transformation (\mathbf{R}, \mathbf{t}) that minimizes the sum of squared distances between each \mathbf{q}_{c_i} and its corresponding transformed point $\mathbf{R}\mathbf{p}_i + \mathbf{t}$ [28]. In cases when other distance measures between pointsets are used, this computation is more difficult, but still possible (we will discuss the different distance measures in Section 2.4.1). On the other hand, if the two shapes are aligned, a correspondence for a given point in \mathbf{P} is usually just its closest point in \mathbf{Q} .

At its core, registration is an optimization problem, which can either be stated as optimization in the space of the parameters of the optimal transformation $\alpha \in \mathcal{A}$, or as an optimization in the space of all possible correspondence assignments between points in \mathbf{P} and \mathbf{Q} , the correspondence variables $\{c_i\}$. As a result, many of the proposed solutions for registration of shapes are based on different function optimization methods.

Solutions to the pairwise registration problem generally fall into two categories, based on the type of optimization that they perform. *Global registration* refers to algorithms that seek to find the best alignment between model and data shapes without any prior assumptions about their initial positions. In effect, they are looking for the global minimum of the distance function. These algorithms are usually based on a correspondence search using local shape descriptors, as reviewed in Section 2.3. *Local registration* algorithms assume that the relative transformation between the input shapes is approximately correct, i.e. the closest local minimum of the distance function is the correct alignment, and seek to locally improve the registration. These methods are almost exclusively based on function optimization using local search.

2.3 Global Pairwise Registration

Global registration algorithms are applied when the input scans start in arbitrary positions, and no estimate of the aligning transformation is available. Solutions to the global registration problem fall into two general classes, based on whether they search over the space of all point correspondences or over the space of all aligning transformations.

2.3.1 Transformation Search Methods

Transformation search methods make use of the fact that the rigid transform is low dimensional and exhaustively search for the small number of parameters needed to specify the optimal transform. One class of approaches, known as voting methods, quantize the transformation space into a six-dimensional table and accumulate votes in each cell of the table based on the geometry of the input shapes. Generalized Hough transform [44], geometric hashing [108], and pose clustering [102] compute a transformation that aligns each triplet of points in the model shape to each triplet of points in the data shape and record a vote 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 [48], 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, in general, explore the entire set of transformations or alignments (up to some quantization error), and therefore are likely to find the optimal alignment between the data and model shapes, independently of the initial position of the input shapes. However, these methods can be costly, having complexity of at least $O(n^3)$, and therefore they are not usually used directly for global registration of scanned data. Voting methods are often preceded by simplification of the input shapes or feature extractions steps. Additionally, their storage requirements for quantizing the transformation space make voting methods more useful when the pre-processing costs can be amortized over many shape matching queries, such as in object recognition and shape retrieval applications, as opposed to more

on-line nature of scan data registration problems.

A second class of transformation search methods are based on exploring the error landscape of the function in Equation 2.3. Since the minimization over all transformations is a complicated non-linear function, its error landscape contains many local minima. Therefore many of the methods for escaping from local minima have been applied to transformation search. These include evolutionary search [30], genetic algorithms [19, 98] and simulated annealing [69]. Although sometimes effective, these methods can be slow, are not guaranteed to converge to the correct alignment, and their performance is difficult to analyze and highly depends on the kind of input given to the algorithm.

2.3.2 Correspondence Search Methods

Correspondence search methods, as the name implies, search for the globally best set of corresponding points and compute the aligning transformation in the process of evaluating potential correspondences. The unknowns that are optimized over are all potential correspondences in the model for the points in the data, which results in a problem with a very large number of free variables. However, the geometric constraints given by the transformation class that is considered for registration (e.g. rigid, affine, etc.) results in a set of constraints that need to be satisfied for the corresponding points. The constrained optimization approach has been used by Anguelov et al. [3] develop a method for registration of deformable objects which optimizes the joint probabilistic model over all point-to-point correspondences under a set of constraints that preserve local and global geodesic distances between corresponding point pairs. The resulting problem has the structure of a Markov Random Field, and the best assignment of corresponding point-pairs is then found using Belief Propagation.

Of particular interest is the class of correspondence search methods known as interpretation trees, which were originally used for matching features extracted from 2D images. We will review this method here, since it can be directly applied for 3D registration, and it forms the basis of our global registration algorithm in Chapter 3. The algorithm, as it was

originally proposed, deals with finding legitimate and consistent pairings between two sets of features extracted from 2D images: points, line segments, and curved segments. This problem can be stated as one of constrained search over an interpretation tree. Each node of the tree describes a partial pairing of the model and data primitives, and the node with the best possible pairing, as measured by a matching function, is chosen as the result of the search. The advantage of these search methods is that they explore all possible assignments between model and data primitives, and are guaranteed to find the best possible alignment globally.

Basic interpretation tree search is an inherently exponential process, making it infeasible for all but the simplest shape matching problems. The same observation as for the constrained optimization case helps here: the transformation that aligns the model and data primitives provides geometric constraints among sets of primitives, which can significantly reduce the search space. For example, a rigid transformation has to preserve pairwise distances, therefore distances between pairs of data points have to be the same as distances between their corresponding model points.

The constrained interpretation tree search was used by Grimson and Lozano-Perez for matching polygonal [41] and curved [43] objects in 2D, and later extended to matching sparse range or tactile sensor data to an image [42]. The geometric constraints employed by the matching system are preserving the angle between pairs of segments, and preserving the range of distances between pairs of segments. These approaches have worked well for recognizing planar and laminar objects, but have not yet been extended to matching more general 3D shapes.

2.3.3 Shape Descriptors

Both the voting schemes for transformation search and the correspondence search algorithms 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 [50] and shape contexts [8] 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 [31], which means that rich descriptors can be used to directly solve the correspondence problem.

In global registration, rich shape descriptors are often used to solve the correspondence problem directly: i.e. choose for each point in the data the point in the model with the best matching shape descriptor. Huber [47] 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 [8, 35].

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 [54] and curvedness [55]. 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 [7] to reduce the size of the search space and the number of point-to-point comparisons. A more common use for low-dimensional descriptors is in local registration by iterative refinement, where they can improve the funnel of convergence (set of starting positions which result in correct alignment) of the algorithm [93, 40].

2.4 Local Pairwise Registration

Local registration methods start with an estimate of the best aligning transformation between the model and data shapes. This estimate can come from a global alignment algorithm, from tracking the scanner position, or from user input. The algorithm then refines this estimate to better register the two shapes.

Most local registration algorithms are based on the Iterated Closest Point algorithm (ICP) developed by Besl and McKay [11] and Chen and Medioni [18]. We give a basic overview of ICP here, since it is used in later parts of this thesis. A detailed survey of ICP is provided in [85]. ICP algorithms solve Equation 2.3 by alternating the following steps:

1. **Find correspondences.** For each point $\mathbf{p}_i \in \mathbf{P}$, assign the correspondence variable c_i to be the index of the point in \mathbf{Q} that is closest to \mathbf{p}_i . Usually, some thresholding on the distance between \mathbf{p}_i and \mathbf{q}_{c_i} is performed, and points which are too far away from their correspondences are discarded as being outside the overlap area [73].
2. **Compute transformation parameters.** Given the set of corresponding point-pairs $(\mathbf{p}_i, \mathbf{q}_{c_i})$ compute the transformation (\mathbf{R}, \mathbf{t}) that minimizes sum of squared distances between the corresponding points. If the distance measure is Euclidean distance between points, then the rigid transform can be found in closed form [28], otherwise an approximation to the transform [64] or local search [18] can be used.
3. **Move \mathbf{P} and iterate.** Apply the computed transformation to the points of \mathbf{P} and iterate the process. The algorithm terminates when the change in residual in Equation 2.3 is sufficiently small.

The ICP algorithm converges to a local minimum of Equation 2.3, hence an estimate of the aligning transform that is fairly close to the correct alignment is required. ICP and other local search algorithms are evaluated using two criteria. The *funnel of convergence* (Figure 2.4) of a given local registration algorithm is a set of points in transformation space which when used as an initial position for \mathbf{P} result in the algorithm correctly aligning it to \mathbf{Q} . If $(\mathbf{R}^*, \mathbf{t}^*)$ is the global minimum of Equation 2.3 in transformation space, the funnel

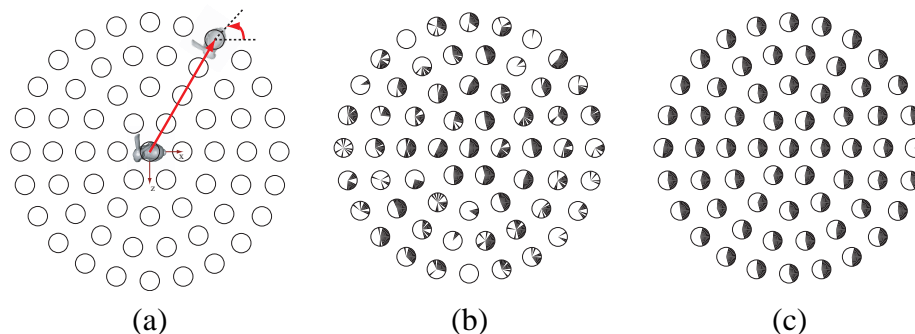


Figure 2.2: Influence of distance metric on the convergence funnel of ICP. (a) Visualization of initial positions. Each circle represents a translation in xy -plane, and arcs in the circle represent rotations around z -axis. (b) Positions from which an ICP with a point-plane error metric was able to converge to the correct alignment for a particular registration problem are shaded. Notice that as the transformations get larger, the funnel of convergence gets smaller. (c) Same visualization but for ICP using the second order approximation to the squared distance function. This is a better approximation to the true point-surface distance, and therefore results in a wider funnel of convergence.

of convergence is the area of transformation space around $(\mathbf{R}^*, \mathbf{t}^*)$ for which the global minimum is also a local minimum. Clearly, we would like local registration algorithms like ICP to have wide funnels of convergence. In addition to having a wide funnel of convergence we would also like the algorithm to converge to the correct alignment quickly, i.e. within a small number of iterations.

It has been shown both experimentally [85] and theoretically [64] that the speed and funnel of convergence of ICP heavily depends on the choice of corresponding points and the distance metric that is minimized when computing the aligning transformation.

2.4.1 Distance Metrics

Two distance metrics are commonly used in ICP and its variants. The point-to-point distance of Besl [11] uses the sum of Euclidean distances between the corresponding points. This allows the aligning transformation to be computed exactly in closed form [28], but also leads to an algorithm that converges only linearly for certain types of input, and initial positions when \mathbf{P} and \mathbf{Q} are already close to a correct alignment.

Another common distance metric is the point-to-plane distance of Chen and Medioni [18], which uses the distance between a point \mathbf{p}_i and the planar approximation to the surface defined by \mathbf{Q} and the corresponding point \mathbf{q}_{c_i} . In this case, it is possible to solve for the transformation in closed form using the small-angle approximation for rotations. When the initial positions of the data is close to the model, and when the input has relatively low noise, ICP with point-to-plane error metric can have quadratic rate of convergence. When the shapes start far away from each other, or for noisy point clouds, point-to-plane error metric can lead to an ICP algorithm that fails to converge [38].

Mitra et al. [64] show that the combination of point-to-point and point-to-plane error metric weighted by the local curvature and the distance between the points gives a second-order approximation to the squared distance function between the point \mathbf{p}_i and the surface defined by \mathbf{Q} . Using the approximation to the squared distance function results in an ICP algorithm with quadratic rate of convergence when the data and model shapes are close, and with linear rate of convergence, but a wide convergence funnel when the shapes are farther away (see Figure 2.4).

2.4.2 Closest Point Search

Although originally named Iterated Closest Point, it is now more appropriate to call ICP the Iterated *Corresponding* Point algorithm [85], since many different strategies for choosing corresponding points have been proposed.

We again refer the reader to the survey of ICP variants by Rusinkiewicz and Levoy for the different variants of closest point computations. In this section, we discuss how shape descriptors can be incorporated into the local search framework of ICP.

Assume we have a set of scalar-valued descriptor functions f_1, \dots, f_m which are computed for each point of \mathbf{P} and \mathbf{Q} . We can treat the points on the model and data as lying in a $(m+3)$ -dimensional Euclidean space: $\mathbf{p}_i = (p_{ix}, p_{iy}, p_{iz}, f_1(\mathbf{p}_i), \dots, f_m(\mathbf{p}_i))$ and similarly for points of \mathbf{Q} . Closest points are computed in this $(m+3)$ -dimensional space, which now includes not only distances between points but also their similarity [93]. Figure 2.3 shows

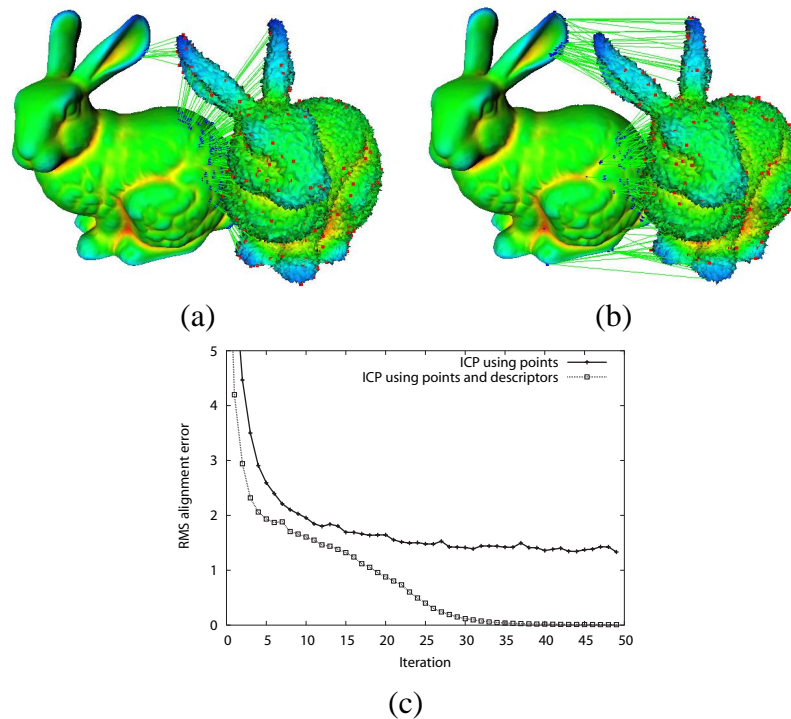


Figure 2.3: Closest point search in point and descriptor space. (a) Closest points using point positions only. (b) Closest point search in 4 dimensions, using point positions and a curvature-based shape descriptor. Notice that the corresponding points now come from similar-looking areas of the shapes. (c) Comparison of the rate of convergence of regular ICP and ICP with descriptor-based closest point search. Regular ICP does not converge from this starting position, while ICP with descriptor-based closest point search finds the correct alignment.

how the corresponding points change when local descriptors are included in the closest point computation, and the resulting improvement in the rate of convergence of ICP.

2.5 Multi-View Registration

There is relatively less work dealing with multi-view registration as compared to the pairwise registration literature.

Multi-view registration also has local and global sub-problems. Given K scans of an object

$\mathbf{P}_1, \dots, \mathbf{P}_K$, global multi-view registration deals with deciding which pairs of scans contain overlapping areas and can be aligned together using a pairwise registration algorithm. Most of the global multiview registration methods are graph based, with the scans representing the nodes of the graph, and an edge connecting nodes \mathbf{P}_i and \mathbf{P}_j if they can be aligned using a pairwise algorithm. The weight of the edge is related to the residual error after the alignment. The goal of multi-view global registration is to pick a set of edges from the graph which will result in the best possible reconstruction of the object. The first system for multi-view registration of range images was developed by Huber [47], which picks the best set of pairwise matches by computing a best weight spanning tree in the graph. Huber also proves that the general multi-view global registration problem is NP-hard.

Given a set of graph edges (corresponding to matches between pairs of scans) and a set of K initial transformations produced by a global multi-view registration algorithm, the goal of local multi-view registration is to refine the transformations to minimize Equation 2.2. Only the pairs of scans that correspond to the graph edges computed by global multi-view registration are included in the minimization. This problem is closely related to the well-known bundle adjustment problem in photogrammetry [62].

Krishnan et al. [86] propose a numerical optimization technique which directly optimizes Equation 2.2 over the K sets of transformation parameters. Other methods include doublet-based heuristics, which perform repeated pairwise registrations of groups of scans to spread out the accumulated matching error. Pulli [80] proposes a method in which the accumulated error is distributed among the K scans by repeatedly picking a single scan and aligning it to the group formed by the other $K - 1$ scans using ICP. This heuristic has good performance when the scans are already in good alignment, but tends to diverge when the initial alignments have errors or outliers. A more sophisticated heuristic has been proposed by Sharp et al. [94] where the accumulated error is spread out along loops of edges in the matching graph.

2.6 Summary

Despite the large amount of work on local and global registration described above, there are still several unsolved problems that remain. Pairwise methods based on local search often suffer from slow convergence and shallow local minima in the presence of "difficult" geometry, such as scans which contain relatively few features that can constrain the alignment.

Global pairwise registration methods that are based on matching corresponding points by using high dimensional shape descriptors, such as the spin images [50], although they often perform well in practice, are not guaranteed to find the correct alignment, since they do not necessarily explore the entire search space of correspondences.

Voting methods, such as the Hough transform, are guaranteed to find the correct alignment, but are expensive for anything other than small inputs. A common speedup technique is to pick a set of feature points on the model and data shapes, often based on computed descriptor values. The alignment is then performed only with respect to these features [60, 67]. 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, existing global registration techniques have to oversample their inputs, reducing their efficiency.

Chapter 3

Global Registration Using Combinatorial Search

"Never stop exploring."

–North Face logo

As described in Chapter 2, global registration is the problem of optimal alignment of two three-dimensional shapes in arbitrary initial positions. The problem is usually encountered in the first part of the registration pipeline, when range images coming from the scanner need to be roughly aligned into a common coordinate system. However, this problem is not exclusive to shape acquisition, optimally positioning two shapes is also encountered in shape retrieval [35], shape modeling [34], segmentation [4], object reassembly [45, 70] etc.

In this chapter, we present a new global registration algorithm that combines the aspects of both descriptor-based alignment to compute an initial correspondence set and of function optimization to compute the best alignment. We use the integral invariants framework [61] to robustly compute a multi-scale descriptor based on local geometry at each point. A small number of feature points are automatically picked from the data shape according to the uniqueness of the descriptor value at the point. We use the descriptor values of each feature point to find its potential correspondences in the model. The best set of corresponding

points is chosen from this initial set by a fast combinatorial method based on distance matrix comparisons. 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 performing integral operations on the underlying shape, for identifying features in the data and selecting potential correspondence points in the model. 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 results of our global alignment are sufficiently close to the correct registration pose to be used as a starting pose for automatic refinement with ICP. In addition to using the results for registration, we show several other shape processing applications that use global alignment based on our algorithm.

The integral volume invariant described in this chapter has been developed jointly with

Helmut Pottmann, who also describes a general framework for 3D integral invariants in [77]. The matching algorithm and its applications are the result of a close collaboration with Niloy Mitra, and are additionally described in his thesis [63].

3.1 Integral Invariants

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 [61, 20]. 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. [61] 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 [61] to 3D.

Integral invariants are defined by integrating a function over a moving domain (usually a

ball of certain radius) centered at each surface point.

$$f(\mathbf{p}) = \int_{B_r(\mathbf{p})} f(\mathbf{x}) d\mathbf{x}.$$

Here $B_r(\mathbf{p})$ is defined as the interior of the ball of radius r centered at \mathbf{p} . Different choices of $f(\mathbf{x})$ give rise to different descriptors, while changing the radius of B_r gives descriptors computed at different scales. Pottmann et.al present a comprehensive study of various integral invariants in [77]. Here, we derive a simple multi-scale invariant which is related to mean curvature at \mathbf{p} .

3.1.1 Integral Volume Invariant

Assuming the point set \mathbf{P} is sampled from some surface Φ , the simplest function $f(\mathbf{x})$ that gives rise to an integral invariant is the indicator function χ , which is one for points that lie on the interior of Φ and zero elsewhere. Using the indicator function we can define the *integral volume invariant* as

$$V^r(\mathbf{p}) = \int_{B_r(\mathbf{p})} \chi d\mathbf{x}. \quad (3.1)$$

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 point set. The invariant is illustrated in 2D in Figure 3.1(a). Assuming the intersection of the interior of Φ and $B_r(\mathbf{p})$ is simply-connected, the volume descriptor is related to mean curvature at \mathbf{p} as follows,

$$V^r(\mathbf{p}) = \frac{2\pi}{3}r^3 - \frac{\pi H}{4}r^4 + O(r^5). \quad (3.2)$$

The leading term is the volume of the half-ball of radius r , and the correction term involves the mean curvature H at the point \mathbf{p} . The full derivation of Equation 3.2 is given in Appendix A.

To show that this descriptor is robust to noise, let \mathcal{P} be the patch that bounds the surface

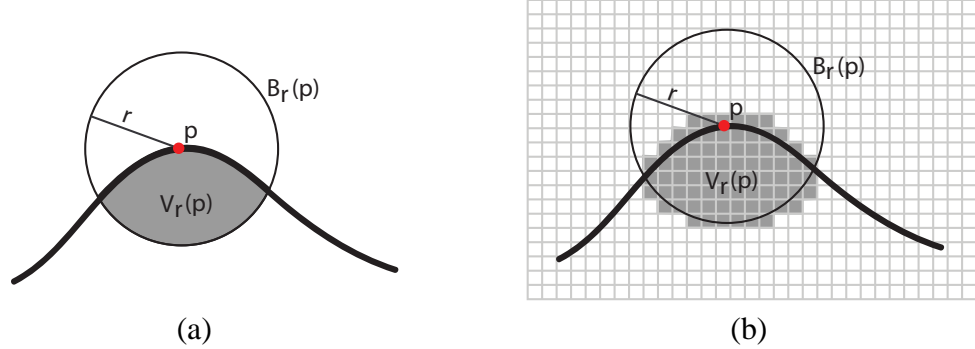


Figure 3.1: Illustration of the volume integral descriptor in 2D. (a) We take the intersection of a ball of radius r centered at point \mathbf{p} with the interior of the surface. (b) Discretization of the volume descriptor as computed by our algorithm. The cell size of the grid is ρ .

where it intersects the ball $B_r(\mathbf{p})$. Noise causes a perturbation that moves \mathbf{p} to \mathbf{p}' and thus the kernel ball $B_r(\mathbf{p})$ undergoes a translation to $B_r(\mathbf{p}')$. The latter intersects the perturbed surface in a patch \mathcal{P}' . Translating $B_r(\mathbf{p}')$ back to $B_r(\mathbf{p})$ moves \mathcal{P}' to a patch \mathcal{P}^* . Apart from a negligible part along the intersection with the ball $B_r(\mathbf{p})$, the change of the value of the volume descriptor is given by the oriented volume ΔV between \mathcal{P} and \mathcal{P}^* . Let us assume that \mathcal{P} can be expressed as a parametric surface $\mathbf{s}(u, v)$. We express the perturbation towards \mathcal{P}^* using a displacement field $\tau(u, v)$ in normal direction of each point of \mathcal{P} . Then, the change in volume descriptor at \mathbf{p} due to the perturbation can be shown to be

$$\Delta V = \int_{\mathcal{P}} \tau(u, v) dA - \int_{\mathcal{P}} \tau^2(u, v) H dA + \frac{1}{3} \int_{\mathcal{P}} \tau^3(u, v) K dA, \quad (3.3)$$

where K is the Gaussian curvature at \mathbf{p} (full derivation is given in Appendix A). We assume the perturbation noise is independently, identically distributed with mean zero and variance σ^2 . Let H_{max} be the maximum mean curvature over the patch \mathcal{P} , and \mathcal{A} be the area of the patch, then the expected change in the volume descriptor can be bounded by $|E[\Delta V]| \leq H_{max} \sigma^2 \mathcal{A}$. The change in descriptor value relative to the volume of the integration kernel is proportional to σ^2/r^2 , which shows the robustness of the descriptor to noise.

3.1.2 Implementation

We compute the integral volume invariant on a voxel grid, as shown in figure 3.1(b). For a given grid size ρ , we build the occupancy grid G_Φ , where $G_\Phi(c) = 1$ if the voxel c lies on the interior of Φ or intersects its boundary, and is zero otherwise. Additionally, we build the ball grid G_B which contains the scan-converted ball of radius r placed at the center of the grid. We compute the integral volume invariant by convolution:

$$V(c) = (G_B * G_O)(c).$$

The above expression can be evaluated efficiently by using the Discrete Fourier Transform of the ball and occupancy grids. The occupancy grid G_Φ can be computed using scan conversion algorithms for meshes [68] or ray shooting algorithms for point clouds [1], which work for generally well behaved surfaces with few holes. If the inputs are range images, which often contain many holes and disconnected patches, the above scan conversion algorithms may fail to distinguish between inside and outside voxels. However, in this case we can use the scanning direction usually available with a range image (i.e. most range images are oriented towards the scanner), to perform the occupancy grid computation. We set the grid size ρ to be large enough to account for the perturbation of the vertices due to noise. Once the convolution is computed, the value of the volume descriptor at each vertex of the input shape is given by the value of the descriptor at the voxel containing the vertex.

Boundaries and holes in the input surfaces affect the value of the volume descriptor of all cells that lie within distance r of the hole, since they result in gaps in the occupancy grid (Figure 3.2). We fill small holes in the input by performing a dilation of the occupancy grid by several (up to 5) voxels, followed by a contraction. This method is effective for filling holes that are the result of noise in the scanning process or very small occlusions, but it does not work for holes that are the result of major self-occlusions on the object. Although we can fill larger holes by a more expensive method such as [24, 68], we found that just discarding points which are within radius r of a boundary still leaves enough area for the algorithm to pick feature points, so in most cases more expensive processing is not

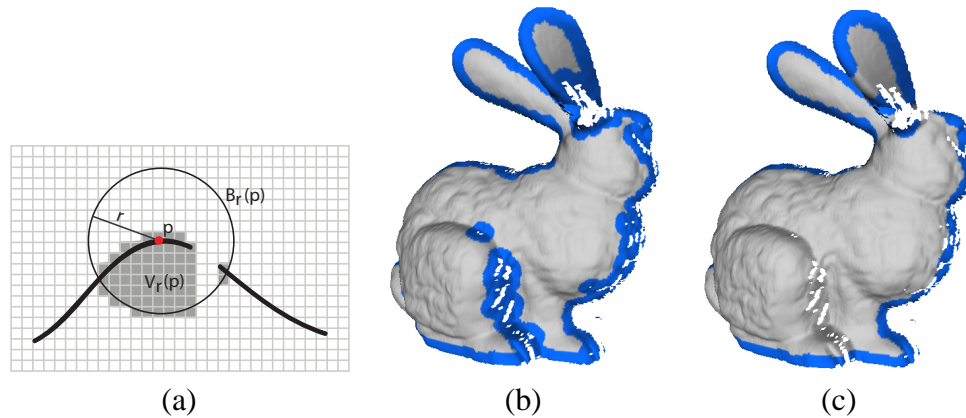


Figure 3.2: Computing the integral volume invariant in the presence of holes. (a) Holes in the input create errors in the occupancy grid (b) This causes all points within radius r of a hole or boundary to become invalid (in blue). (c) Small holes in the occupancy grid were filled by dilation and contraction, removing some of the invalid points. The inside/outside voxel classification was computed using the scanning direction.

necessary.

Integral descriptors are particularly suited for multiscale representation since the scale is given by the radius of the kernel B_r . Figures 3.3 (a) and (b) show the volume descriptor computed for the Stanford Bunny model for two different ball radii. In Section 3.2.2 we describe an algorithm that uses the multiscale representation of the volume descriptor to robustly identify persistent features.

3.2 Feature Point Selection

As is common for many global alignment algorithms (see Section 2.3), our approach is based on computing a set of corresponding points between the model and data shapes. There are two main components to the algorithm: we select features and find correspondences by using the integral volume invariant (although the process would work for any low-dimensional descriptor) and we use distances between pairs of points to verify partially built correspondence sets. In this section, we describe our feature selection process,

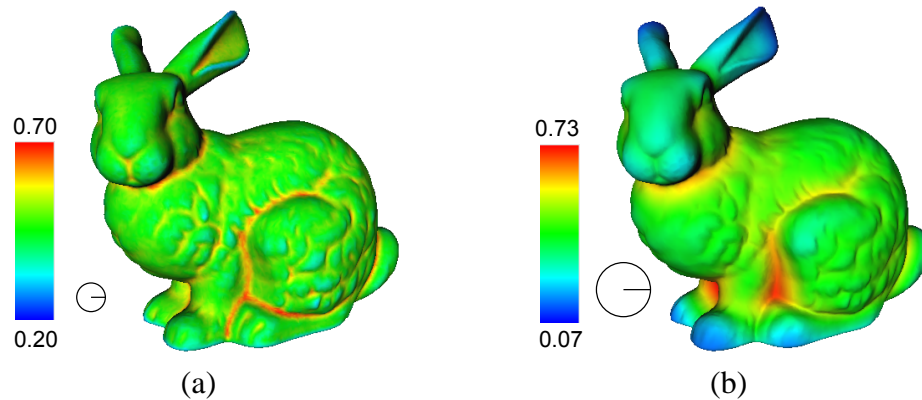


Figure 3.3: Values of the normalized volume descriptor computed for the bunny model (a) for small and (b) large integration kernels. We normalize the descriptor value with respect to the volume of the ball B_r and use the full blue-red scale for the range of values to ease visualization. The radius of the integration kernel, the color scale of the values, and the minimum (blue) and maximum (red) value of the descriptor are shown on the left of each figure.

which is based on analyzing the distribution of descriptor values on the data shape.

The overall structure of the feature selection and correspondence search steps is similar to other global aligners. We select a small number of points from the data shape as features, and we use the descriptor values at those points to find their potential correspondences in the model. The main novel property of our feature selection process is to select those points as features that will make the search for correspondences particularly simple. To this end, feature points are selected from the areas of the data shape which have uncommon descriptor values. Since the data and the model shapes are similar over the matching region (underlying assumption of registration), and we use descriptor values to select corresponding points in the model for each feature point in the data, points with rare descriptor values are likely to have only a few corresponding points. Thus, the 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 search problem.

3.2.1 Basic Algorithm

We will present the feature selection algorithm in terms of a general descriptor, since it does not rely on any particular properties of the integral volume invariant. In the next section, we will use the fact that it is easy to compute scale-space representation of the integral invariants to develop a scale-sensitive feature selection 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, although 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 \mathbf{P} . The feature point selection proceeds as follows:

1. Compute a histogram of descriptor values, $f(\mathbf{p}_i)$ for all points in \mathbf{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 [90].
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_{max} . 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_{max} controls the tradeoff between accuracy of the transform (more correspondences) and running time of the algorithm. In our implementation, we set $s_{max} = 0.01N$.
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 in more stable configurations in the correspondence search stage of the algorithm (Section 3.4).

Notice that this process is not invariant to the order of points in \mathbf{P} . This means that it cannot be used to pick a set of canonical points on the data and model shapes. This is in contrast to many other global aligners, which reduce the size of the correspondence space by first subsampling the model and data by picking some canonical features, and then search for optimal correspondence assignment among the points in the feature sets. In contrast, our correspondence search algorithm does not need canonical points, since it will search the entire model shape for correspondences. 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 3.5 shows the feature points selected on the dragon model corrupted with zero-mean Gaussian noise.

3.2.2 Scale-space Representation and Persistent Features

The concept of a point being a feature is usually scale-dependent. For example a small "bump" may be a feature if we are considering the object at a small scale, but should be ignored when we are looking at the object's overall shape. For this reason, many descriptors and feature selection algorithms employ the notion of a scale-space [67, 72], where each potential feature point is considered at several scales simultaneously.

Integral invariants are particularly well suited for scale-space representation, since the scale on which the invariant is computed is naturally controlled by the radius of the integration kernel B_r . The main problem that we need to solve is to determine what is the appropriate

descriptor scale that should be considered at each point \mathbf{p}_i . It is expected that this scale should be different for different features, and we would like to be able to determine the scale automatically.

We make use of the following observation: If a point \mathbf{p} is an actual feature point, it should be present over a set of consecutive scales of the descriptor. A point that is an outlier, on the other hand, is likely to disappear as a feature as the scale is varied. Therefore, we use the persistence [16] of a feature point in the scale-space representation to identify true features and discard outliers.

Most shapes contain features at different scales, so we do not expect a point to be a feature over the entire scale-space of the descriptor. Instead, we define as a feature a point whose descriptor value is rare over a set of consecutive kernel radii of the volume descriptor. Small scale features will be persistent for small radii of the descriptor and large-scale features over large radii, and outliers may look like features for some radii but are not persistent. In addition to identifying a point as a feature, our algorithm automatically identifies the scale of the feature.

To implement the persistence algorithm, we sample the scales of the volume descriptor at discrete intervals. We divide the range $r_{min} \leq r \leq r_{max}$ of possible ball radii into k intervals ($k = 5$ in our implementation) and convolve the occupancy grid with the different ball grids. To avoid discretization errors, r_{min} is set to 10ρ , where ρ is the resolution of the voxel grid. We also limit r_{max} to some fraction (usually set to 0.1) of the diameter of the input shape to preserve the locality property of the shape descriptor. We also normalize the magnitude of the volume descriptor for each radius r by the volume of the ball B_r . For a point \mathbf{p} to be a feature, it should be selected as a feature for a set of continuous scales. We use the basic algorithm described in Section 3.2.1 to identify feature points for each radius of the volume descriptor, and then select only those points that are a feature for at least two consecutive radii. Figure 3.4 shows feature points selected using the scale-space algorithm.

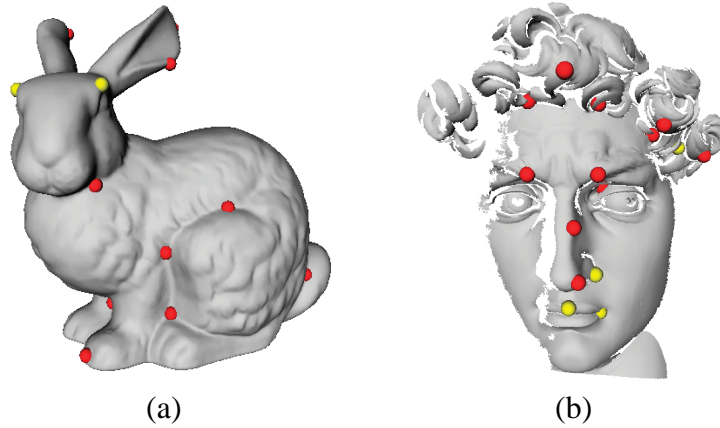


Figure 3.4: Persistent feature selection. Feature points selected using the scale-space algorithms on the (a) bunny and (b) David models. Features in red are persistent over larger radii of the descriptor, while features in yellow are persistent over smaller radii.

3.3 Distance Metric Based on Intrinsic Shape Properties

We now come to the second part of the global alignment algorithm: the search for corresponding points for the set of feature points. Given a set of n feature points selected from data shape \mathbf{P} , the goal of the correspondence search algorithm is to find, for each feature point \mathbf{p}_i , a point $\mathbf{q}_i \in \mathbf{Q}$, that is the best match to \mathbf{p}_i . In order to determine if a set of points $(\mathbf{p}_i, \mathbf{q}_i)$ form a good corresponding set, we need to develop a distance metric for correspondences. In Section 2.4.1, we introduced the point-to-point distance metric as a way of evaluating distance between two point sets in correspondence. It is also known as *coordinate root mean squared error*, or cRMS, in fields such as structural biology, since it measures how close each point \mathbf{p}_i comes to corresponding point \mathbf{q}_i after an optimal rigid aligning transform is computed for the entire set of corresponding points. Let \mathbf{P}' and \mathbf{Q}' be two point sets, with correspondences given as $(\mathbf{p}_i, \mathbf{q}_i)$. As defined in Section 2.4.1, cRMS is given by:

$$\text{cRMS}^2(\mathbf{P}', \mathbf{Q}') = \min_{\mathbf{R}, \mathbf{t}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2, \quad (3.4)$$

The drawback of this distance metric is that the aligning transformation cannot be computed

incrementally. Being able to update the distance measure without having to re-compute it is a desirably property, since our matching algorithm will build up the correspondence set $(\mathbf{p}_i, \mathbf{q}_i)$ incrementally.

The assumption that the data and model are related by a rigid transform allows us to develop a measure of quality of the proposed correspondences without having to compute the actual aligning transformation. This new distance metric is based on the observation that any rigid transform has to preserve inter-point distances. Namely, if the set of points in \mathbf{Q}' is a good match for the points in \mathbf{P}' , the distances between all pairs of points in \mathbf{P}' should be the same as between their corresponding points in \mathbf{Q}' .

The metric based on internal intra-point distances of \mathbf{P}' and \mathbf{Q}' is known as 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 [53]. The dRMS error is computed by comparing all internal pairwise distances of the two point sets:

$$\text{dRMS}^2(\mathbf{P}', \mathbf{Q}') = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n (\|\mathbf{p}_i - \mathbf{p}_j\| - \|\mathbf{q}_i - \mathbf{q}_j\|)^2. \quad (3.5)$$

Same as the cRMS error, this metric assumes that the correspondences between the two point sets are known, with the point \mathbf{p}_i having a corresponding point \mathbf{q}_i .

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,

$$\text{dRMS}(\mathbf{P}', \mathbf{Q}') \leq \sqrt{2} \text{cRMS}(\mathbf{P}', \mathbf{Q}'). \quad (3.6)$$

To compute the 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

$$\frac{1}{k\sqrt{n}} \min(\text{cRMS}(\mathbf{P}', \mathbf{Q}'), \text{cRMS}(\mathbf{P}', \overline{\mathbf{Q}'})) \leq \text{dRMS}(\mathbf{P}', \mathbf{Q}'). \quad (3.7)$$

Here n is the number of corresponding point pairs and k is a small constant, depending on ratio of the diameter of the data shape to the feature exclusion radius used in Section 3.2. The full derivation of the bounds is given in Appendix B. These bounds mean that 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.

dRMS has the advantage that 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 described in the next section.

3.4 Matching Algorithm

We now have all the ingredients to develop a matching algorithm. Given a set of n feature points \mathbf{P}' picked from the data, we want to select a set of best matching points \mathbf{Q}' from the model. We treat this problem as a search over all possible sets of n points in the model, and present an efficient way to explore this search space.

3.4.1 Initial Correspondence Set

We first narrow down the potential correspondences for each \mathbf{p}_i by performing a range query over the descriptor values in \mathbf{Q} . For each feature point \mathbf{p}_i we have the scale-space representation of the volume descriptor $(V_{r_1}(\mathbf{p}_i), \dots, V_{r_k}(\mathbf{p}_i))$, 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.

To perform the descriptor query, we compute the same scale-space representation of the volume descriptor on the model shape by computing volume descriptors for radii r_1, r_2, \dots, 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 in the model, and select all points \mathbf{q} such that $|V_{r_b}(\mathbf{p}) - V_{r_b}(\mathbf{q})| < \varepsilon$. 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 $\varepsilon \approx \frac{3\rho}{4r}$ (which is the difference in volume of two balls whose centers are one voxel apart). This accounts for the variation in the value of the volume descriptor due to discretization using the voxel grid. Since we pick ρ to be large enough to account for noise in the data, therefore ε also absorbs the noise term in Equation 3.3.

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 ensures 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.

After performing the range query and clustering for each feature point, we obtain the correspondence search space, where we need to select n matching points from the set $C(\mathbf{p}_1) \times C(\mathbf{p}_2) \times \dots \times C(\mathbf{p}_n)$. Although this is a smaller search space than searching over

all of \mathbf{Q} , since we chose the feature points such that each $C(\mathbf{p}_i)$ is small, it's still too large to explore exhaustively. We compute an initial set of correspondences using the hierarchical greedy algorithm of Mitra [63] described below. Other heuristics for selecting a set of corresponding point-pairs given a correspondence search space can be used here also. While it does not guarantee the optimal assignment, the greedy algorithm often produces the correct correspondence set in practice, and it is quite fast. Since we want to guarantee that we fully explore the correspondence search space and find the optimal assignment of correspondences, we also develop a combinatorial search algorithm based on branch and bound. The initial heuristic algorithm can then be used to produce a good initial estimate for the branch and bound algorithm, resulting in efficient pruning of incorrect correspondence sets.

3.4.2 Initial Bound

We initialize the correspondence set and matching error using a greedy algorithm developed by Mitra and described also in [63]. As with all branch and bound approaches, a tight initial bound helps make the algorithm more efficient, since more incorrect branches are pruned by the bound during the search. Any heuristic for selecting a set of point pairs from a correspondence search space can be used to obtain the initial bound. For example, recently Huang et al. [45] presented a matching algorithm based on forward search, which is more expensive than the method described here, but provides very tight bounds in practice. While several iterations of Huang's algorithm are required to explore the entire correspondence search space, a single iteration is often enough to find a correspondence set that is close to the correct matching configuration, which can then be verified and refined using the branch and bound method.

For completeness, we include Mitra's greedy algorithm here. The algorithm is based on selecting sets of matching point-pairs hierarchically. At each step, we select a set of matching points that best optimizes the current dRMS error. The algorithm is greedy, since it never backtracks even if a better choice for a given correspondence becomes available at later steps.

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 3.4) 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 3.4.

This approach is greedy because each step picks the best correspondences to merge together and never backtracks. Therefore it is 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.

3.4.3 Branch-and-bound Search

The greedy algorithm above does not guarantee that we find the optimal correspondence. Therefore, we need an algorithm that will efficiently explore the space of all possible correspondence assignments and either verify the result of the greedy search, or find a better alignment. 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 $\mathbf{P}' = (\mathbf{p}_1, \dots, \mathbf{p}_n)$ selected from the data shape, and a set of potential correspondences for each point in the model shape $(C(\mathbf{p}_1), \dots, C(\mathbf{p}_n))$, 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 3.5 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

$$\left| \|\mathbf{p}_i - \mathbf{p}_j\| - \|\mathbf{q}_i^k - \mathbf{q}_j^l\| \right| < 2R_c. \quad (3.8)$$

We apply this thresholding rule in a branch-and-bound algorithm for finding the best set of correspondences. Let $\mathbf{Q}' = (\mathbf{q}_1^*, \dots, \mathbf{q}_n^*)$ 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 3.4.2. 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 $(\mathbf{q}_1^{c_1}, \dots, \mathbf{q}_{k-1}^{c_{k-1}})$. 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 3.8 with respect to all previously selected points. That is we verify that Equation 3.8 holds for all pairs $(\mathbf{p}_i, \mathbf{p}_k), (\mathbf{q}_i^{c_i}, \mathbf{q}_k^j)$ for $i = 1, \dots, k-1$. 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 $(\mathbf{q}_1^{c_1}, \dots, \mathbf{q}_{k-1}^{c_{k-1}}, \mathbf{q}_k^j)$ 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 $\mathbf{Q}' = (\mathbf{q}_1^{c_1}, \dots, \mathbf{q}_n^{c_n})$.

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 is computed is in the last step, and only if we need to update the bound.

3.4.4 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 3.6(d) shows the dRMS error vs. the number of matched feature points for the David model.

3.5 Results

3.5.1 Object Registration

We show the results of applying our algorithm to a number of pairwise registration problems. Although the data and model shapes are shown to be in similar positions in the figures, the input was actually given to the algorithm in arbitrary orientation. Table 3.1 gives the data size and timing results for the experiments.

The first example demonstrates that the integral invariants can be used for alignment in the presence of noise. We align the dragon model to a copy of itself corrupted by zero-mean gaussian noise. No explicit smoothing was performed for descriptor computation

	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 3.1: 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.

and matching. The results are shown in Figure 3.5. Our alignment brings the data (noisy) shape close enough to the model (smooth) shape that applying one iteration of standard ICP with point-to-point error metric brings the shapes into exact alignment.

Figure 3.6 shows the results of applying our algorithm to register partially overlapping range data. We take two raw scans of the David’s face, uniformly 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 a global adjustment algorithm [80]. This gives us a completely automatic model construction pipeline. The result is shown in Figure 3.7.

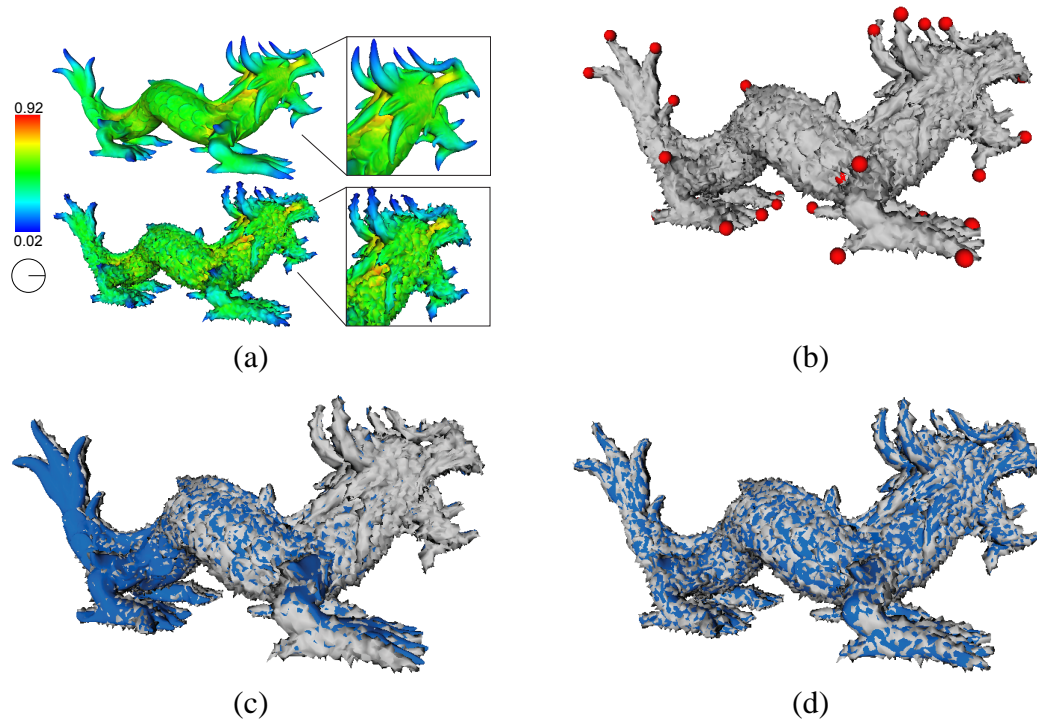


Figure 3.5: Dragon example. (a) Input to the matching algorithm: Smooth dragon (the model) and noisy dragon (the data) with descriptor values shown at each point. Even under noise the descriptor values at feature points look similar. (b) Feature points picked on the data shape. (c) Registration after applying our algorithm. (d) Registration after refinement the resulting pose by ICP.

3.5.2 Symmetry Detection

The global registration algorithm presented above can be used for several geometry processing applications. In [63] Mitra presents an algorithm for detecting simple symmetries in an object by matching it to a copy of itself. Instead of returning the best matching orientation, all matches with small error are returned. 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 3.8 shows the results of detecting symmetries of a mechanical part. Notice that the graph of error in Figure 3.8 shows eight configurations with small error, which corresponds to the eight-way symmetry of the model.

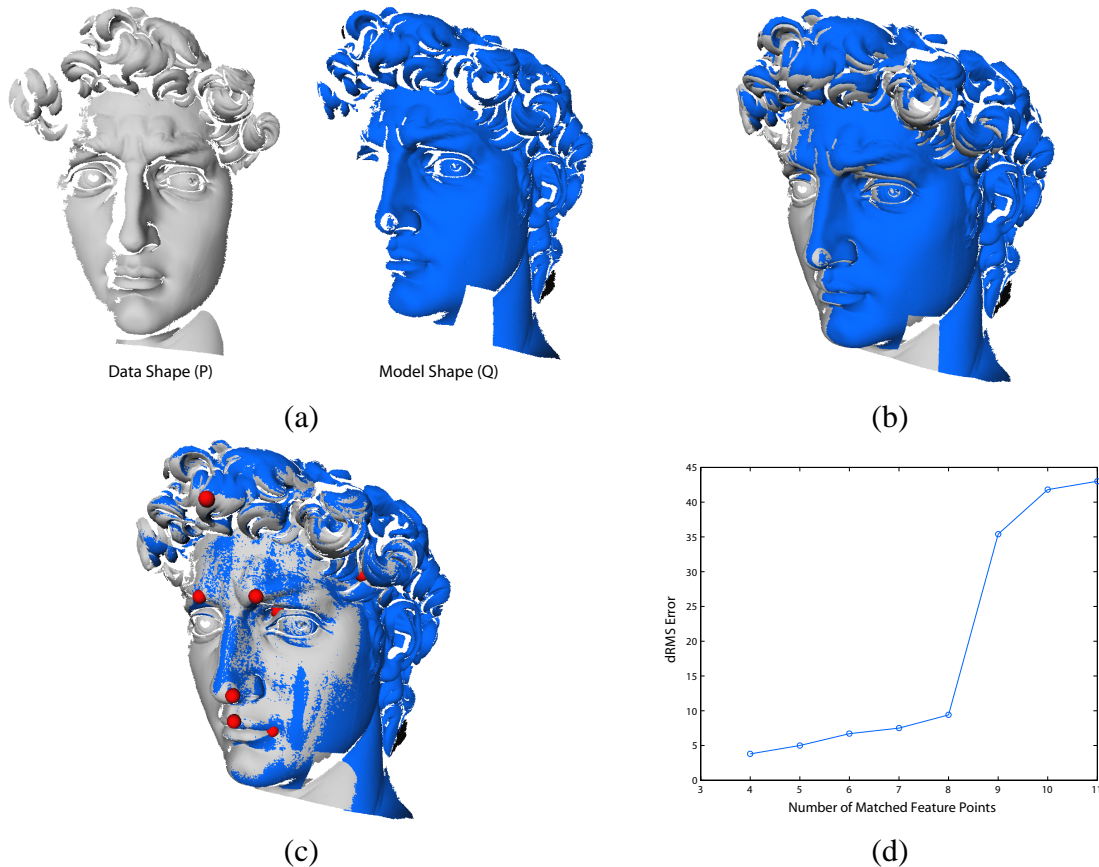


Figure 3.6: (a) Two scans of the David's face. Feature points picked on the data shape are shown in red. (b) Registration after applying our algorithm. (c) Registration after refinement by ICP. Points actually used to compute alignment in (b) are shown in red. (d) Graph of dRMS error as the function 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.

Clearly, this method works only for whole-object translational and rotational symmetries. Recently, more sophisticated methods based on local point matching [65, 74] have been developed for detecting richer classes of total and partial symmetries.

3.5.3 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

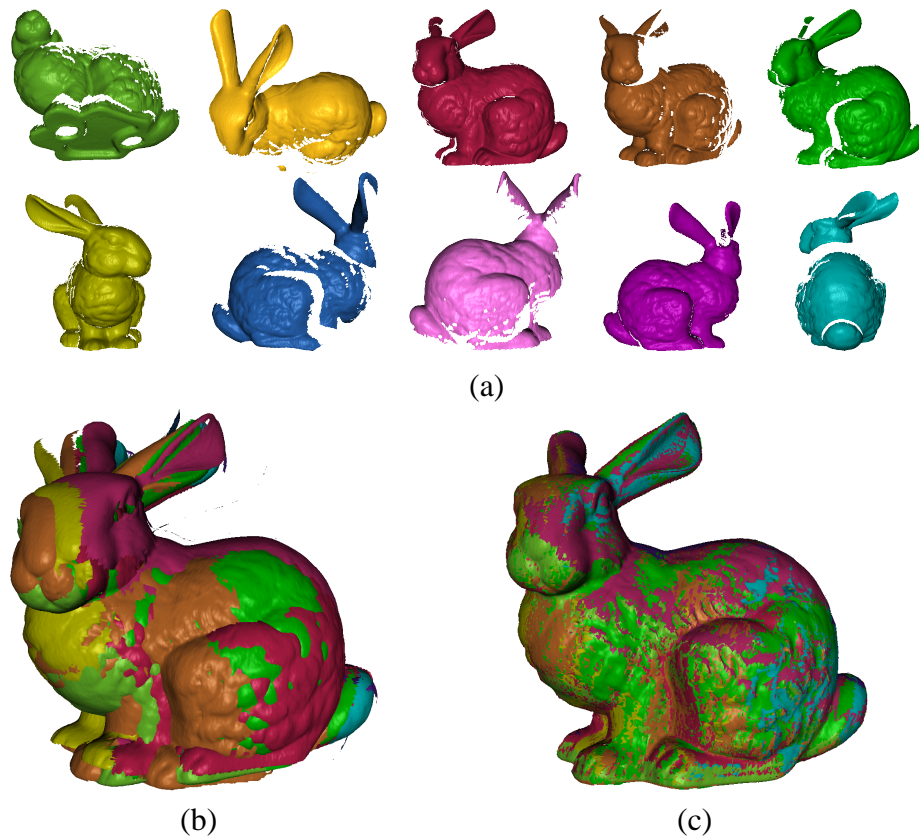


Figure 3.7: (a) 10 input scans (shown here in good position for visualization, the actual input positions are arbitrary). (b) Registration after applying our algorithm to overlapping pairs of scans. (c) Registration after applying ICP and error relaxation to the initial pose produced by our algorithm.

decompose the shape \mathbf{P} into the minimum set of parts $\mathbf{P}_1 \dots \mathbf{P}_k$, 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 developed in collaboration with Niloy Mitra [63].

We perform articulated decomposition by partial matching of \mathbf{P} and \mathbf{Q} . This gives the 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 3.9 shows the result of segmenting a shape into rigid components using

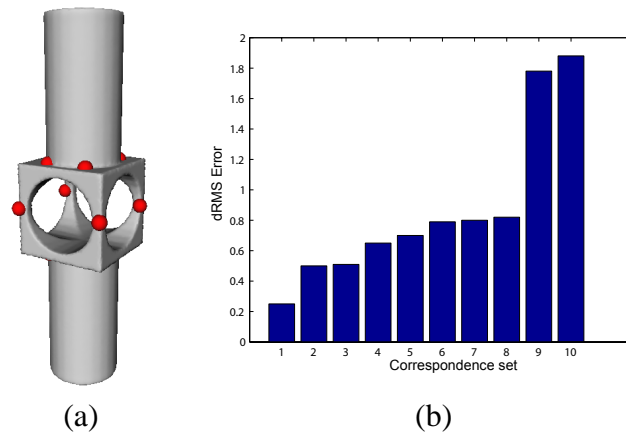


Figure 3.8: Symmetry detection using registration. (a) Feature points picked by our algorithm, when the shape is aligned to a copy of itself. (b) Graph of the error for different correspondence sets. The eight correspondences with small error indicate the eight-way symmetry of the shape.

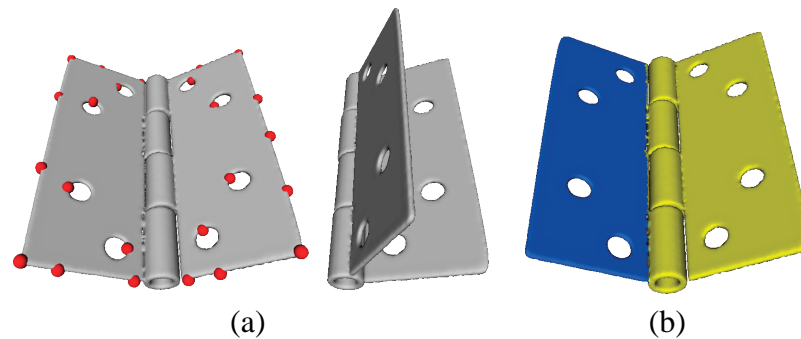


Figure 3.9: Simple articulated matching. (a) Two input positions of the shape. Feature points picked by our algorithm are shown in red. (b) Using repeated partial matching, the algorithm discovers two rigid components.

this algorithm.

The features picked on the data shape in Figure 3.9 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 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.

The shapes in Figures 3.8 and 3.9 also show that the integral volume invariant is only an approximation to the mean curvature at a point given by Equation 3.2, when the radius of the integration kernel is smaller than the local feature size at that point. For example, for thin structures such as the hinge, the integration kernel can be too large, which can result in integrating across to the other side of the hinge. In general, this is not a problem, since although we will not get quite a correct value of the invariant (which, for a planar object like the hinge in Figure 3.9 should be $1/2$), it will be consistent for the two hinge positions since the integration kernels are the same in the data and the model shapes. This can also be solved using more sophisticated methods for computing integral invariants such as explicit intersection of the voxels with the integration kernels, which can incorporate normal information into the computation [46].

3.6 Summary

We presented a global registration algorithm that aligns two three-dimensional shapes without any assumptions about their initial positions. Our algorithm is able to align whole and partially overlapping shapes, and is robust to moderately noisy data. The main insights of our approach were the method for selecting as features points with uncommon shape descriptors, evaluating a potential alignment between two sets of points without actually computing an aligning transformation by comparing distance matrixes of the two point sets, and an efficient branch-and-bound algorithm for exploring the entire correspondence search space. Since our algorithm explores all possible correspondences for the selected features, it is guaranteed to find the best assignment of correspondences to the feature points. Due to the nature of the feature selection process, our algorithm works particularly well in the presence of strong point-like features in the input data.

The limitations of our approach are two fold. First, when the input shapes do not have

strong point-like features, the correspondence search space examined by the algorithms can become quite large and the algorithm will be slow. Second, since we represent the data shape with a small set of feature points, we are guaranteed to only find the best set of correspondences for the selected features, not for all points on the data shape. In general, we assume that the selected feature points represent the underlying shape well, so the correct alignment of the feature points in the data to the model will lead to the correct alignment of the whole data and model shapes. However, in case of particularly specialized input, it is possible that the features do not faithfully represent the entire model, in which case our algorithm may find the wrong alignment. It is important, therefore, to check the correctness of the alignment after executing the global registration algorithm, which can be done by checking the sum of squared distances of all points in the data shape and their closest points in the model shape.

A significant improvement on our method has been developed recently by Huang et al. [46, 45]. The algorithm is based on feature selection and correspondence search using the integral invariants presented in this chapter, however instead of performing a clustering of descriptor values to select feature points, as described above, entire patches of similar descriptor values are selected as features. This addresses the problem of feature detection in the input data which lacks strong point-like features. In the correspondence search stage, corresponding patches in the model are selected for feature patches in the data based on both the value of the descriptor, and the shape of the patch. Similar to our approach, the algorithm also builds the entire correspondence search space, and then explores it using forward search, which is a particularly effective heuristic method for outlier detection.

Chapter 4

Local Surface Slippage

*"If he had been as you and you as he,
You would have slipped like him."*

–William Shakespeare

In the previous chapter we used an intrinsic property of a point set, namely the matrix of pairwise distances among its points, to develop a global registration algorithm. Under the assumption that two point sets are related by a rigid transformation, we were able to estimate their similarity without computing the actual alignment, just by comparing those intrinsic properties. We continue the theme of analyzing the intrinsic properties of shapes in this chapter, and develop the notion of local surface self similarity based on how the surface behaves under uniform rigid motions. We call this property local surface slippage.

We take a short digression from the topic of registration, and show how the local slippage property can be used to segment a surface of a mechanical part into meaningful components. Since slippage deals with the behavior of a surface under a rigid motion, it is clearly relevant to the topic of rigid registration. We return to registration in the next chapter, where we use the slippage of a surface to improve point sampling for a local registration algorithm.

4.1 Rigid Motions

A rigid motion $M(t)$ consists of two time-varying components: $R(t) \in SO(3)$, which determines the rotational part of the motion and $\mathbf{a}(t)$ which is the translational component. At time t , position of a point \mathbf{x} moving according to M is given by

$$\mathbf{x}(t) = R(t) \cdot \mathbf{x} + \mathbf{a}(t). \quad (4.1)$$

The first derivative of Equation 4.1 is the velocity vector of the point \mathbf{x} at time t , which we designate by $\mathbf{v}(\mathbf{x})$. It is well known that the velocity vector of each point \mathbf{x} at a given time t can be written as [13]:

$$\mathbf{v}(\mathbf{x}) = \mathbf{c} \times \mathbf{x} + \bar{\mathbf{c}}, \quad (4.2)$$

where \mathbf{c} and $\bar{\mathbf{c}}$ are 3-vectors. At a given time t all velocity vectors are zero only if $\mathbf{c} = \bar{\mathbf{c}} = 0$. This case is called the stationary instant of the motion. It has been shown that at every non-stationary instant t , the velocities of the points agree with the velocities of a uniform helical motion, a uniform rotation, or a uniform translation [79]. We differentiate among the three cases as follows:

- If $\mathbf{c} = 0$, the motion M is a translation with constant velocity along the direction $\bar{\mathbf{c}}$.
- If $\mathbf{c} \cdot \bar{\mathbf{c}} = 0$, M is a rotation with constant angular velocity.
- If $\mathbf{c} \cdot \bar{\mathbf{c}} \neq 0$, M is a uniform helical motion.

4.2 Surface Slippage

Given a surface S we call a rigid motion M a *slippable motion of S* if the velocity vector of each point $\mathbf{x} \in S$ is tangent to S at \mathbf{x} . If the instantaneous motion of each point is tangential, it means that the distance between the transformed surface and the original does not change, to first order. If we imagine two copies of the surface, one that is moving according to M and one that is stationary, then the surface can be thought of as sliding against itself,

without forming any gaps between the moving surface and the stationary copy. That is, the surface S is invariant under its slippable motions. Surfaces which are invariant under one of the types of rigid motions described above are known as kinematic surfaces [79]. One can differentiate among kinematic surfaces that are generated by rotational, translational or helical motions [78, 101].

A kinematic surface can be slippable in more than one way. The simplest example is a plane. Any translational motion along the plane is slippable, as is rotational motion around the plane's normal. A more interesting kinematic shape is a cylinder, for which slippable motions include rotations around the cylinder's axis and translations along the axis. Other kinematic shapes include spheres, helical surfaces, surfaces of revolution and surfaces of linear extrusion (translationally slippable surfaces).

Most surfaces are not slippable in their entirety. However, they often contain slippable components, or are composed of several different slippable surfaces. In registration, we often encounter surfaces which are almost slippable, i.e. they come from objects which are largely symmetric, but have small features. Special care is often needed to register scans that come from such objects, a problem that we will address in the next chapter. In the rest of this chapter, we will focus on how to compute local surface slippage for a pointset.

4.3 Computing Slippable Motions

Let \mathbf{x} be a point belonging to the surface S , and let \mathbf{n} be the vector normal to S at \mathbf{x} . We will examine how \mathbf{x} is affected by an instantaneous motion whose parameters are given by the 6-vector $[\mathbf{c} \ \bar{\mathbf{c}}]$. The amount of non-tangential motion is given by the dot product of the velocity vector with the normal at the point \mathbf{x} :

$$\mathbf{v}_{perp} = (\mathbf{c} \times \mathbf{x} + \bar{\mathbf{c}}) \cdot \mathbf{n}. \quad (4.3)$$

Kinematic surfaces are those for which there exists a rigid motion whose velocity vector

field is tangential to the surface at each point. We can write this condition as

$$\int_S ((\mathbf{c} \times \mathbf{x} + \bar{\mathbf{c}}) \cdot \mathbf{n})^2 dS = 0. \quad (4.4)$$

Now, we assume that the input data is given by a pointset \mathbf{P} of n points that have been sampled from some underlying surface S . Our goal is to determine if S is a kinematic surface and find its slippable motions. Each point $\mathbf{p}_i \in \mathbf{P}$ is given by a 3-vector of its coordinates $\mathbf{p}_i = [p_{ix}, p_{iy}, p_{iz}]$. We also assume that at each point \mathbf{p}_i we have a corresponding normal $\mathbf{n}_i = [n_{ix}, n_{iy}, n_{iz}]$ that approximates the normal vector to the surface S . If the input pointset is given as a mesh, we can use the triangles around \mathbf{p}_i to estimate the normal. If no connectivity information is given with the input, the normals can be estimated by plane fitting using the technique described in [66].

We can find the slippable motions of \mathbf{P} (and correspondingly S) by posing Equation 4.4 as a minimization problem. We want to find the parameters $[\mathbf{c} \ \bar{\mathbf{c}}]$ of a velocity vector field that, when applied to \mathbf{P} minimizes the motion along the normal direction at each point.

$$\min_{[\mathbf{c} \ \bar{\mathbf{c}}]} \sum_{i=1}^n ((\mathbf{c} \times \mathbf{p}_i + \bar{\mathbf{c}}) \cdot \mathbf{n}_i)^2. \quad (4.5)$$

Not surprisingly, the same minimization problem arises in the context of pointset registration [18, 85]. If we think of the pointset \mathbf{P} as having two copies, a moving version P_T and a stationary version P_O , Equation 4.5 minimizes the point-to-plane error metric of Chen and Medioni [18] between the transformed and stationary pointset. A slippable motion is the one where the point-to-plane distance between P_T and P_O is zero [38]. We analyze this case more in the next chapter.

Equation 4.5 is a least-squares problem whose minimum is the solution of a linear system $\mathbf{C}\mathbf{x} = 0$, where \mathbf{C} is a (covariance) matrix of partial derivatives of the objective function

with respect to the motion parameters.

$$\begin{aligned} \mathbf{C} = \mathbf{F}\mathbf{F}^T &= \begin{bmatrix} \mathbf{p}_1 \times \mathbf{n}_1 & \dots & \mathbf{p}_k \times \mathbf{n}_k \\ \mathbf{n}_1 & \dots & \mathbf{n}_k \end{bmatrix} \begin{bmatrix} (\mathbf{p}_1 \times \mathbf{n}_1)^T & \mathbf{n}_1^T \\ \dots & \dots \\ (\mathbf{p}_k \times \mathbf{n}_k)^T & \mathbf{n}_k^T \end{bmatrix} \\ &= \sum_{i=1}^n \begin{bmatrix} c_{ix}c_{ix} & c_{ix}c_{iy} & c_{ix}c_{iz} & c_{ix}n_{ix} & c_{ix}n_{iy} & c_{ix}n_{iz} \\ c_{iy}c_{ix} & c_{iy}c_{iy} & c_{iy}c_{iz} & c_{iy}n_{ix} & c_{iy}n_{iy} & c_{iy}n_{iz} \\ c_{iz}c_{ix} & c_{iz}c_{iy} & c_{iz}c_{iz} & c_{iz}n_{ix} & c_{iz}n_{iy} & c_{iz}n_{iz} \\ n_{ix}c_{ix} & n_{ix}c_{iy} & n_{ix}c_{iz} & n_{ix}n_{ix} & n_{ix}n_{iy} & n_{ix}n_{iz} \\ n_{iy}c_{ix} & n_{iy}c_{iy} & n_{iy}c_{iz} & n_{iy}n_{ix} & n_{iy}n_{iy} & n_{iy}n_{iz} \\ n_{iz}c_{ix} & n_{iz}c_{iy} & n_{iz}c_{iz} & n_{iz}n_{ix} & n_{iz}n_{iy} & n_{iz}n_{iz} \end{bmatrix} \end{aligned} \quad (4.6)$$

where $c_{ik} = (p_i \times n_i)_k$. Therefore, the slippable motions of P are those that belong to the null space of C . To compute the actual motion vectors, we compute the eigenvalue decomposition $C = X\Lambda X^T$. Eigenvectors of C whose corresponding eigenvalues are 0 correspond to the slippable motions of the pointset P . In practice, due to noise C is likely to be full rank. In this case, the slippable motions are those eigenvectors of C whose eigenvalues are sufficiently small. We can determine the type of slippable motion that each eigenvector corresponds according to the classification in Section 4.1. Table 4.1 shows examples of slippable shapes and their corresponding slippable motions.

While we used the equation of rigid motion to classify point sets into kinematic surfaces, the same classification has also been obtained by Pottmann et al. by considering the line complex formed by the points and normals of \mathbf{P} as shown in [79]. This results in the same covariance matrix as Equation 4.7. Looking at invariance of surface under rigid motion, which resulted in the slippage property, was motivated by studying the behavior of local registration algorithms, which will be described in the next chapter.

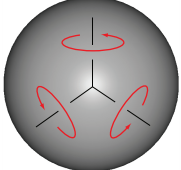
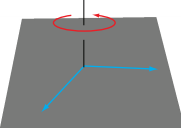

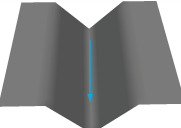
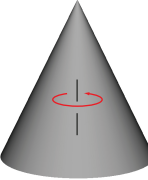
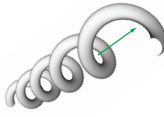
Num. small Eigenvectors	Type of eigenvectors	Type of Surface	
3	3 rotations	sphere	
3	2 translation, 1 rotation	plane	
2	1 translation, 1 rotation	cylinder	
1	translation	linear extrusion	
1	rotation	surface of revolution	
1	helical motion	helix	

Table 4.1: Kinematic surfaces. For each surface, we indicate the number of small eigenvalues of the covariance matrix in Equation 4.7 and the type of the corresponding eigenvectors. Notice, that the eigenvectors given are only one possible set of slippable motions for that shape. Any motion that is a combination of the slippable eigenvectors is also slippable (see Section 4.2). Translational motions are indicated by blue arrows, rotational motions by red arrows, and helical motions by green arrows.

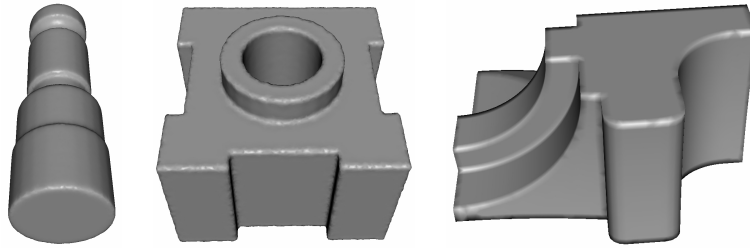


Figure 4.1: Examples of mechanical parts composed of kinematic surfaces.

4.4 Application: Shape Segmentation Using Local Slippage Analysis

The above computation for determining slippable motions can be performed for any set of points. In particular, we may be interested in computing the slippage of an entire point set or of some of its components. A case when it is interesting to decompose a point set into slippable components is in reverse engineering of scans of mechanical parts, which often consist of surfaces shown in Table 4.1. The problem, therefore, is as follows: given a set of points \mathbf{P} , we want to decompose it into a number of subsets $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_k$ such that each \mathbf{P}_i is large, connected, and slippable. We first introduce some background work in segmentation and reverse engineering, and then present a simple algorithm for finding slippable subsets in an input pointset. The approach will be based on developing a descriptor for a set of points based on the number and type of slippable motions that it possesses.

4.4.1 Segmentation

Reverse engineering applications deal with reconstructing a CAD model from an unstructured input dataset such as one that may come from a laser scanner. A significant problem in reverse engineering is the segmentation of the input dataset into a set of regions, such that each region can be approximated by a single simple surface. Segmentation is usually followed by surface fitting, where each component is approximated by best fitting parametric surface. The problems of segmentation and surface fitting are closely related: if we know

the surfaces, we can segment the input pointset by grouping together points that lie within a threshold of the same surface. On the other hand, given a segmentation of the pointset into components, we can find the approximating surfaces by finding the best fitting surface for each component. For general objects, segmentation and surface fitting problems often require user input. However, in many CAD applications, the underlying model is composed of kinematic surfaces, see examples in Figure 4.1. In such cases, automatic segmentation and surface fitting are often possible.

There is a large body of research dealing with shape segmentation. A general survey of segmentation and surface fitting in reverse engineering of CAD objects can be found in [106]. Most automatic segmentation methods fall into two general categories. Bottom-up, or region growing techniques, start with a set of seed points for which some local surface characteristics are computed. New points are then added to the seed regions as long as the computed surface characteristic does not change. The other approach to automatic segmentation is to proceed top-down. The original pointset is recursively subdivided until each subset belongs to a single component. This approach is common in image segmentation [95], however in model segmentation most techniques tend to use the bottom-up method, e.g. [88, 9]. For surface fitting of general shapes, the segmentation problem is generally difficult, and the user is often asked to indicate rough component boundaries [57], which are then refined and approximated with parametric patches. In Computer Graphics, automatic methods for segmentation of arbitrary shapes are often used for generating base mesh domains for multiresolution analysis and texture mapping. Such methods are generally based on generating regions that satisfy specific distance and planarity constraints [52, 36, 12, 87, 59].

The use of kinematic surfaces in geometry processing is not new. Pottmann et al. [78, 76] determine if a given pointset is sampled from a kinematic surface by analyzing the normals of \mathbf{P} in line-space. The input points are then re-projected onto the underlying kinematic surface for noise reduction or for inspection and quality assurance. Our segmentation algorithm based purely on region growing of slippable components is, to the best of our knowledge, new and is described in [37].

4.4.2 Point Classification

We cannot apply the method of Section 4.3 to the input pointset P directly, since P as a whole may not be slippable. Our goal is to discover a decomposition of P into P_1, P_2, \dots, P_k such that each P_i is large, connected, and slippable.

Our approach falls into the class of bottom-up segmentation algorithms. We start by computing for each point in the input a guess at what kind of kinematic surface it was sampled from. For each point $\mathbf{p}_i \in P$ we form a neighborhood P_i of m points around \mathbf{p}_i . This forms our original set of components. If the input data is given with the connectivity information, e.g. as a mesh, we can build each P_i by crawling the mesh structure outward from \mathbf{p}_i until m points are encountered. If the input is given as a point cloud, we just take the m nearest neighbors of \mathbf{p}_i .

Next, we compute the covariance matrix C_i of points in P_i according to Equation 4.7. We make two modifications to the basic equation to make the computation more numerically stable. First, we shift all points in P_i so that the center of mass lies at the origin of the coordinate system. Second, we uniformly scale the points so that the average radius of the patch is 1. These steps do not change the slippable motions of the pointset, but ensure that the magnitude of the $\mathbf{p}_i \times \mathbf{n}_i$ term is comparable with the \mathbf{n}_i term in the computation, making the computation more numerically stable.

The next step is to decide how many slippable motions the neighborhood around \mathbf{p}_i has. Let $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_6$ be the eigenvalues of C_i and $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_6$ be the corresponding eigenvectors. We call the eigenvalue λ_j “small” if the ratio $\frac{\lambda_6}{\lambda_j}$ is greater than a given threshold g (we use a value between 100 and 300 in our implementation). If k is the number of small eigenvalues of P_i , we call the eigenvectors $\mathbf{x}_1, \dots, \mathbf{x}_k$ the *slippage signature* of \mathbf{p}_i . We write the slippage signature in matrix form as $X_{1\dots k}$, with the eigenvectors corresponding to slippable motions arranged in columns.

The actual segmentation proceeds by aggregating neighboring points into slippable components. Originally, the neighborhood around each point \mathbf{p}_i is treated as a separate patch P_i . Each patch has a covariance matrix C_i and a slippage signature $X_{1\dots k}^i$. Notice that a patch

may not have any slippable motions (if all eigenvalues of C_i are large), in which case its slippage signature is empty. The algorithm proceeds as follows:

1. **Initialization:** Compute a similarity score between each pair of adjacent patches. In the case of mesh input, adjacency is easy to determine. In the case of point cloud input, two patches are adjacent if they share any vertices. The similarity score is based on both the number and the compatibility of the slippable motions of the two patches. We use a priority queue to store the patch pairs, with the pair that has the best similarity score at the top of the queue.
2. **Patch growing:** At each step, we select a pair of adjacent patches that are the most similar and collapse them into a single patch. The new covariance matrix is computed from the covariance matrixes of the two patches to obtain the slippage signature for the merged patch. We then update the similarity score between the new patch and its neighbors.
3. **Termination:** Stop aggregating when the similarity score of the pair of patches at the top of the queue drops below a threshold. We apply a cleaning step to remove any small patches that may have remained. The resulting set of patches is the segmentation of the pointset. Each slippable patch can be approximated by a single kinematic surface.

We now examine the steps of the above algorithm in more detail.

4.4.3 Similarity score

Given two patches P_i and P_j (these can be either single points, whose slippage signature is computed from an initial neighborhood, or merged patches during the running of the segmentation algorithm), we say that they belong to the same component if:

- Their corresponding covariance matrixes C_i and C_j have the same number of small eigenvalues.

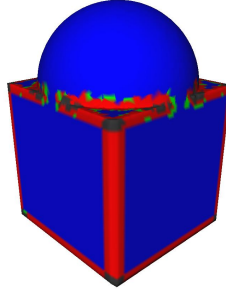


Figure 4.2: Coloring points of a shape consisting of kinematic surfaces based on the number of small eigenvalues of the region around the point. Points whose neighborhoods are one-slippable are colored red, and whose neighborhoods are three-slippable are colored blue. Gray regions correspond to points in neighborhoods with no slippable motions, while green points are incorrectly classified as being two-slippable (See Section 4.5). The width of the one-slippable regions depends on the size of the initial neighborhood around each point (set to 10 points in this example).

- The corresponding slippage signatures are the same, that is we can express the slippable motions of P_i as a combination of slippable motions of P_j and vice versa.

As described in Section 4.4.2, we call λ_k a small eigenvalue if $\frac{\lambda_6}{\lambda_k} > g$ for some minimum “condition number” g . The number of slippable motions for a patch P_i is given by the largest k for which the above condition holds:

$$s(P_i) = \operatorname{argmax}_k \left\{ \frac{\lambda_6}{\lambda_k} > g \right\} \quad (4.7)$$

This means that the distance between the moving and the stationary copy of the patch P_i changes as least g times slower in the direction of slippable motions than any other motions. For a non-degenerate patch (i.e. not a curve), the maximum number of slippable motions is 3. We call a pointset with k small eigenvalues k -slippable. Figure 4.2 shows a simple object whose points are colored according to the number of small eigenvalues in a region around each point. Notice that the planar and the spherical regions are colored the same. This means that we cannot use just the number of small eigenvalues as the surface descriptor for segmentation, we need to look at the corresponding slippable motions as well.

The first test for similarity between patches P_i and P_j rejects the patch pairs for which the number of slippable motions computed according to Equation 4.7 is different.

For the second similarity test, let $X_{1\dots k}$ and $Y_{1\dots k}$ be the slippage signatures of patches P_i and P_j respectively and let k be the number of slippable components of each patch. Since the first test was successful, k is the same for both patches. Each component of the slippage signature is a 6×1 vector corresponding to a rigid motion. Two slippage signatures are compatible if the rigid motions of one can be expressed as combination of rigid motions of the other.

In general, deciding if a given rigid motion M can be expressed as a combination of other rigid motions $M_1 \dots M_k$ is a difficult problem. The space of all rigid motions of \mathfrak{R}^3 , $SE(3)$, is a curved manifold, which means simple interpolation techniques cannot be applied to rigid motions [2, 71]. In our case, however, we are dealing with instantaneous rigid motions, since the eigenvectors of the matrix C correspond to velocities. This means that the eigenvectors of C lie in the tangent space of $SE(3)$, which is flat. As a result we can treat the components of the slippage signatures as vectors in \mathfrak{R}^6 . The columns of each slippage signature form an orthogonal basis for the space of all instantaneous slippable motions of the corresponding pointset. To answer if two slippage signatures $X_{1\dots k}$ and $Y_{1\dots k}$ are compatible, we just need to test if each column $X_{1\dots k}$ can be expressed as a linear combination of columns of $Y_{1\dots k}$.

Because of noise in the data we will never be able to perfectly express the slippable motions of P_i in terms of the slippable motions of P_j . Therefore, we need to look at the residual after the approximation. The amount by which two slippage signatures are dissimilar is given by the $(k + 1)$ st singular value of the combined matrix $[X_{1\dots k} Y_{1\dots k}]$. In the actual implementation we need to transform the rigid motions of one patch into the coordinate system of the other since we applied a shift and scaling in the computation of the covariance matrix.

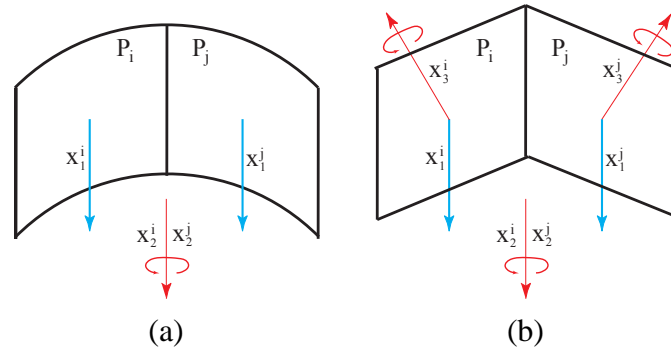


Figure 4.3: Potential segmentation algorithm problem. (a) Two patches that are part of a shallow cylinder look very similar to (b) two planar patches, especially at early iterations of the algorithm. The first two eigenvectors, which correspond to translation along the cylinder (blue arrow) and rotation around the cylinder's axis (red arrow) match for both patches. Since the cylinder is shallow, rotation around the patches's average normal (red arrow) also has a relatively small eigenvalue, making the patches look like they are two planes (b). However, we can distinguish this case since the corresponding eigenvectors will not match. Since classification errors are the motivation for our multi-pass algorithm.

We combine the two similarity tests into one similarity score as follows:

$$Sim(P_i, P_j) = \begin{cases} 0 & \text{if } s(P_i) \neq s(P_j) \\ F(\sigma_{k+1}) & \text{where } k = s(P_i) \text{ otherwise.} \end{cases} \quad (4.8)$$

where s is computed according to Equation 4.7, σ_{k+1} is the $(k+1)$ st singular value of the combined matrix $[X_{1\dots k} Y_{1\dots k}]$, and F is described below.

We would like the similarity score to increase as the patches become more similar, so the function F maps small singular values into high similarity scores. We also use F to map the similarity scores into the range of values between 0 and 1. F is a Gaussian centered around 0, whose width determines how different slippage signatures of two patches that are considered similar can be.

4.5 Robust Segmentation Algorithm

Using Equation 4.4.3, we can now assign a similarity score to each pair of patches and run our clustering algorithm. Since the width of the Gaussian in F controls how fast the similarity score drops when the slippage signatures become different, we can terminate the algorithm when the similarity score of the next candidate pair of patches drops too low. However, using Equation 4.4.3 in its present form will result in poor segmentation due to a number of robustness issues, largely due to incorrectly determining the number of slippable motions of a patch. We now present an algorithm that is tolerant to such mistakes.

4.5.1 Multi-pass segmentation algorithm

Incorrectly determining the number of small eigenvalues of a patch can affect the similarity score in two ways. First, if we set g in Equation 4.7 too high, we can miss some slippable motions for a patch and decide that two patches are incompatible because they have a different number of small eigenvalues. If we set g too low, we can pick eigenvectors that are not slippable as part of the slippage signature. This can make patches incompatible because the non-slippable motions of the patches are not likely to match. Therefore, the performance of our algorithm depends on how well we identify the number of slippable components of each patch.

In practice, it turns out that it is difficult to correctly determine the number of slippable motions of a patch, especially in the early iterations of the algorithm when the patches are still small, and in the presence of noise in the data. The reason for this is that a small neighborhood around a point generally looks planar, e.g. looking around a point we cannot tell if the point should belong to a plane or a cylinder of large radius. In general we cannot reliably classify which shape the neighborhood belongs to until it grows sufficiently large. But growing the patch depends on merging it with its neighbors, which is in turn based on comparing the slippage signatures. For example, if we try to treat two patches that are part of the cylinder as planar, we are likely to get a bad similarity score from their slippage signatures, since only two of the three eigenvectors in each slippage signature are

Algorithm 1 Multi-pass segmentation of a point set P into slippable components.

```

1: for each point  $\mathbf{p}_i \in P$  do
2:   Form patch  $P_i$  of  $m$  vertices (using mesh crawling or nearest neighbors).
3:   Form the covariance matrix  $C$  according to Equation 4.7 and compute its eigenvector
   decomposition  $C = X\Lambda X^T$ .
4: end for
5:  $k \leftarrow 3$ 
6: while  $k \neq 0$  do
7:   for each pair of adjacent patches  $(P_i, P_j)$  do
8:     Form combined matrix of slippage signatures,  $[X_{1\dots k} Y_{1\dots k}]$ , as described in Sec-
     tion 4.4.3.
9:     Compute  $(k + 1)$ st singular value  $\sigma_{k+1}$  of  $[X_{1\dots k} Y_{1\dots k}]$ .
10:    Compute  $Sim(P_i, P_j)$  according to Equation 4.9.
11:   end for
12:   Initialize priority queue to empty.
13:   Insert all pairs of adjacent patches into the priority queue in order of decreasing
     similarity score.
14:    $(P_i, P_j) = \text{EXTRACTMAX}(pqueue)$ 
15:   while  $Sim(P_i, P_j) > \text{MINSIMILARITY}$  do
16:      $P_{ij} = \text{MERGE}(P_i, P_j)$ 
17:     for  $P_k \in \text{neighbors}(P_i) \cup \text{neighbors}(P_j)$  do
18:       Remove pair  $(P_k, P_i)$  (correspondingly  $P_j$ ) from the priority queue.
19:       Compute  $Sim(P_k, P_{ij})$  according to Equation 4.9 and insert pair  $(P_k, P_{ij})$  into the
         priority queue.
20:     end for
21:      $(P_i, P_j) = \text{EXTRACTMAX}(pqueue)$ 
22:   end while
23:    $k \leftarrow k - 1$ 
24: end while

```

going to match well (See Figure 4.3). This is the general "chicken and egg problem" of segmentation [106]: we cannot make a decision about a pointset until we know what shape it belongs to, that is until we have segmented the input. Therefore, instead of trying to pick a correct value of g that will result in fewest misclassifications, we will instead make our algorithm robust against misclassifying shapes at the early stages of segmentation.

Our solution is based on the observation that any k -slippable shape is also $(k - 1)$ -slippable. Therefore, instead of comparing just the k -column slippage signatures of the two patches,

we should also compare the slippage signatures made from the first $(k - 1)$ eigenvectors etc. The best of the (at most three) similarity scores is assigned as the merge score for the two patches. In the case of misclassifying the cylindrical patches as planar, the first two eigenvectors of each patch will form the basis for the slippable motions of the cylinder, while the third will be the one that belongs to the plane (rotation around the normal at the center of the patch). In this case, comparing the slippage signatures with $k = 2$ will give a high similarity score.

However, picking too few components as slippable motions of a patch can result in undesirable segmentation. To illustrate, suppose we have a shape consisting of a cone and a cylinder, which share an axis of revolution. There are two valid segmentations into slippable components: both the cone and the cylinder are in one component, which is one-slippable, with the rotational motion; or there are two components, a two-slippable cylinder and a one-slippable cone. The input clearly consists of two different geometric shapes, so we would like the segmentation determined by our algorithm to reflect that.

We will do the segmentation in several passes. First, we try to merge together all three-slippable components. There may be many patches that are classified as three-slippable because the patch size is not large enough to correctly determine the number of slippable motions. However, the only patches that will be merged are those whose slippable motions are compatible, which are the patches that belong to planar and spherical components of the input. In the second pass, we merge all patches that are classified by the size of their eigenvalues as *at least two-slippable*, i.e. we allow a patch to "drop" a slippable motion. In comparing the slippage signatures of such patches, we only use the first two eigenvectors of each covariance matrix as the slippage signature (i.e. the largest slippable eigenvector is dropped). This handles the case of two-slippable patches being classified as planar as in Figure 4.3. In the final pass, we merge all patches that are at least one-slippable, using the first eigenvector as the slippage signature for each patch. In practice, we repeat this process several times, every time making the width of the Gaussian in F larger to accommodate more noise as patches become larger. Finally, since we tend to misclassify patches more often at the early stage of the segmentation, we do not allow patches that consist of a large number of points to drop slippable motions (this prevents the case of merging the cone with

the cylinder as described above). Our new similarity score is as follows:

$$\text{Sim}(P_i, P_j) = F(\sigma_{k+1}) \cdot G(P_i, k) \cdot G(P_j, k). \quad (4.9)$$

Here, the function G acts as a confidence multiplier for the similarity score. The simplest form of G is just a cutoff. At each iteration of the algorithm, let k be the minimum number of slippable components that we are considering. Then

$$G(P_i, k) = \begin{cases} 1 & \text{if } \frac{\lambda_6}{\lambda_k} \geq g \text{ and } \frac{\lambda_6}{\lambda_{k+1}} < g \\ 1 & \text{if } \frac{\lambda_6}{\lambda_{k+1}} \geq g \text{ and } |P_j| < N \\ 0 & \text{otherwise} \end{cases} \quad (4.10)$$

Here N is the desired size of individual segmented components. Notice that G also prevents the algorithms from merging patches which have different number of small eigenvalues, so we do not need to explicitly test for the number of small eigenvalues of a patch. Algorithm 1 contains the pseudocode for the multi-pass segmentation algorithm.

4.5.2 Initial patch size selection

The last important parameter that affects the robustness of our algorithm is the size of the initial neighborhood formed around each patch. As described above, we do not need to reliably determine the number of small eigenvalues of the patch, but if two patches belong to the same component, we would like the corresponding eigenvectors to match within the threshold set by F .

In practice, we noticed that the algorithm is most sensitive to the value of initial patch size m , as opposed to the value of the condition number g . Therefore, selecting the right patch size is important for good segmentation.

Unfortunately, this parameter is difficult to determine analytically. Therefore, in our implementation, we determine the correct initial patch size by the quality of the final segmentation. If the initial neighborhood size is too small, the eigenvectors in the slippage signature of each point will not match the point's neighbors, and the algorithm will not be able to aggregate large patches. The final output will be over segmented: there will be a large number of components, with only a few points in each. Therefore, if the final segmentation has too many components, we increase the value of m and try again.

4.5.3 Post-processing steps

After the completion of the algorithm, it is likely that a number of regions remains that are not part of any slippable component. One reason of this is that those parts corresponds to areas of the shape that are not slippable. However, often there are regions in the input that are actually part of a slippable shape, but due to noise did not get clustered correctly. We therefore allow large slippable regions to absorb their small neighbors, as long as the overall region remains slippable.

We also apply some simplification to the border between regions. The exact border between two components depends on the neighborhoods size, m , that was used in the initial point classification. For example, in Figure 4.2, a larger neighborhood size would increase the width of the one-slippable regions. While we may need larger size of m for the initial clustering, we often prefer the points to belong to a more slippable component at the end of the segmentation. Therefore, when a point can be classified as belonging to two different patches, as happens near the border between two regions, we assign the point to the more slippable region. In a way, we allow the more slippable regions to eat into the less slippable regions, but only as long as no regions become disconnected.

4.6 Segmentation Results

In this section, we show the results obtained by our segmentation algorithm.

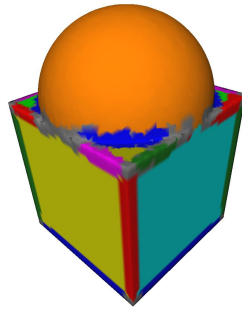


Figure 4.4: Segmentation of a simple model into slippable components. Grey areas correspond to stable regions. Notice that even though the spherical part and planar parts were classified as having the same number of small eigenvalues in Figure 4.2, since their slippable motions are incompatible with each other, they are segmented into separate regions.

Figure 4.4 shows the final segmentation of the shape in Figure 4.2. The shape consists of a cube with a hemisphere attached to one of the faces. The final segmentation consists of three-slippable and one-slippable regions, as well as a number of stable (not slippable) corners. The planar faces and the sphere are segmented into separate components, since even though they have the same number of slippable motions, the motions themselves are different. The segmentation also includes one-slippable components that correspond to the edges between the three-slippable regions. The width of these components depends on the size of the initial neighborhood that is built around each point. Our second cleaning step thins such edge pieces as long as they do not fall apart into disconnected regions. In our application, the input to the algorithm was given as a mesh. Therefore, we used the connectivity of the input to prevent the segmented regions from forming disconnected components during thinning. In the case when input data is given without any underlying connectivity, more sophisticated methods for preserving topology of the regions are required [27].

Many segmentation algorithms perform edge detection as the first step of segmentation, using the assumption that different regions in the input are separated by sharp edges [9]. Our algorithm does not need the edge detection step, and in fact can find boundaries between slippable regions even if no sharp edge is present in the data. Figure 4.5 shows a shape containing a cylindrical part that smoothly joins to a plane, which are correctly identified by our algorithm.

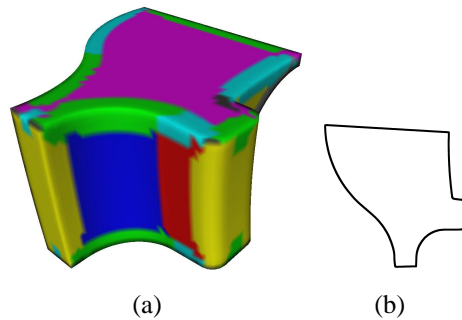


Figure 4.5: Segmentation without sharp edges. (a) The back of the fan disk model contains a cylindrical region (blue) and a planar region (red). Even though there is no sharp edge between them, our segmentation algorithm is able to recognize the two regions. (b) Outline of the bottom of the model, showing the shape of the cylindrical and planar region.

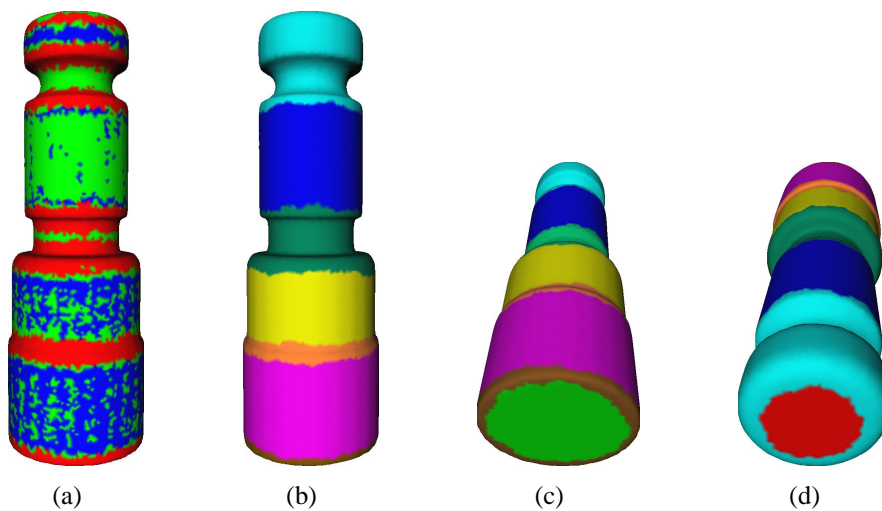


Figure 4.6: Segmentation of a mechanical part consisting mostly of cylinders and surfaces of revolution. (a) Initial classification of neighborhoods into one-slippable (red), two-slippable (green), and three-slippable (blue). Notice that some cylindrical neighborhoods are incorrectly classified as planar. (b) Segmentation obtained by our algorithm. (c) View of the bottom of the part. Notice that the join between the planar bottom and the cylindrical side is captured as a separate rotationally symmetrical component. (d) View of the top of the part.

Figure 4.6 shows the segmentation of a mechanical part consisting mostly of cylinders and surfaces of revolution. The input pointset contains 20,000 vertices. The initial patch classification of a neighborhood of 30 vertices is shown in Figure 4.6(a). The size of the

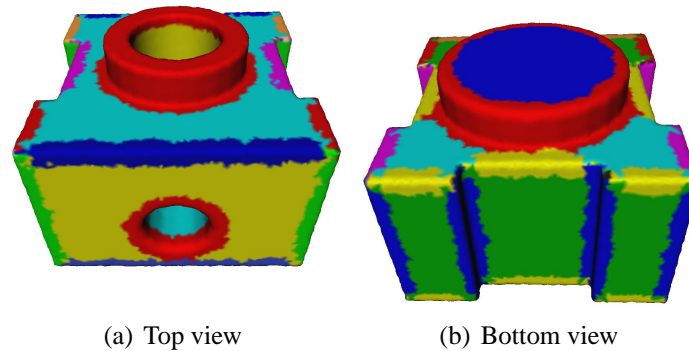


Figure 4.7: Segmentation of a part consisting of planar and cylindrical components. Edges between the planar components and between the planar and cylindrical components are captured as separate regions.

neighborhood was not large enough to correctly classify all vertices that belong to the cylindrical regions, as indicated by the blue coloring (corresponding to the three-slippable regions) at the lower part of the shape. The misclassified vertices were not aggregated in the first pass of the algorithm, since treated as planes, their similarity score was too low. However, in the second pass, they were correctly treated as two-slippable regions and clustered. The results are shown in Figure 4.6(b)-(d). Notice that the segmentation captures the edge between the planar bottom part and the cylindrical side part as a separate one-slippable component. The fact that the algorithm classifies sharp edges as separate components can be advantageous in reverse engineering. Since sharp edges are generally hard to capture accurately in laser scanning, this component is useful as an indicator of the area that is likely to contain a sharp edge. In the construction of a CAD model from this input pointset, this component can be replaced by a surface blend [9].

Finally, Figure 4.7 shows segmentation of another mechanical part, consisting of a larger number of slippable components. The model consists of 40,000 points, and the input neighborhood size was set to 30 vertices. After the initial segmentation, the post-processing pass thinned the one-slippable components to the maximum width of 5 vertices.

4.7 Summary

In this chapter, we continued the theme of analyzing intrinsic surface properties by developing a surface descriptor based on surface invariance under rigid motion, called surface slippage. Although we took a digression from the topic of registration, the slippage surface descriptor was in fact inspired by the behavior of surfaces during rigid registration, a topic we will present in the next chapter.

Since slippage can be computed for small patches as well as for entire surfaces, and is able to detect and classify common surfaces found in scans of mechanical parts, we developed a segmentation algorithm based on the slippage surface descriptor. Even our fairly simple greedy implementation was able to successfully segment mechanical parts into kinematic components without any user input. We expect that using a more sophisticated clustering technique such as [52] with our surface descriptor can lead to even better results.

Several directions are possible for future work. In particular, we observed that surface slippage is dependent on the size of the pointset or neighborhood for which it is computed. On the theoretical front, it should be possible to integrate the slippage descriptor with the persistence framework in [27], to achieve better understanding of the underlying shape. On the practical side, there is a significant amount of work in shape retrieval from large databases of mechanical parts [83, 81]. We expect that surface slippage can be a useful tool for such applications.

Chapter 5

Stable Sampling for Local Registration

"If you wish to drown, do not torture yourself with shallow water."

–Bulgarian proverb

The slippable shapes introduced in the previous chapter often require special care during registration. If the two scans being registered come from some slippable shape, there will not be a unique transformation that aligns them. In this chapter, we show that rigid registration of slippable scans suffers from the problem of shallow local minima in the error landscape. In particular, local optimization algorithms such as ICP encounter this problem and as a result converge slowly, find the wrong pose, or even diverge, especially in the presence of noise or miscalibrations in the input data. Using the slippage analysis introduced in the previous chapter, we develop a method for selecting the input points for ICP that minimizes the uncertainty in the resulting alignment by choosing samples that best constrain potential slippable motions. It is straightforward to apply slippage analysis to the case of pairwise registration. In this case, instead of considering how the shape slides against itself, we examine how the data shape \mathbf{P} slides against the model shape \mathbf{Q} .

5.1 Geometric Constraints for Local Registration

Local registration, as the name implies, assumes that the model and data surfaces are approximately correctly aligned and performs a local refinement of the alignment to obtain the final registered pose. We reviewed the basics of local registration in Chapter 2, where we noted that local registration is often performed by algorithms based on performing local optimization in the space of aligning transformations. The quality of the final pose between the two surfaces, therefore, depends on the number of local minima in the error landscape of the optimization problem. In this chapter we show that registration of slippable or nearly-slippable surfaces has error landscapes with shallow global minima and many local minima, making such surfaces difficult to register using standard local optimization methods.

Most of the optimization-based local registration algorithms are based on the popular ICP algorithms and its variants. In Chapter 2 we observed that the choice of distance metric that is used in the minimization highly affects the number of local minima in the error landscape, and that certain error metrics have larger funnels of convergence. In particular, it has been shown that when two surfaces are close to each other, the point-to-plane error metric of Chen and Medioni is the best approximation of the true distance between the surfaces. We will focus on this case here, since we assume that a good estimate of pose is available from a global alignment algorithm, such as one described in Chapter 3.

We again start with a set of n pairs of points in correspondence $(\mathbf{p}_i, \mathbf{q}_i)$, obtained for example by the algorithm in the previous chapters, or by the closest point search during an iteration of ICP. The point-to-plane error metric and the corresponding minimization problem are defined as follows:

$$\mathcal{E}_{plane} = \min_{\mathbf{R}, \mathbf{t}} \sum_{i=1}^n ((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i) \cdot \mathbf{n}_i)^2, \quad (5.1)$$

where \mathbf{n}_i are the normals to the surface Q at \mathbf{p}_i . The tangent plane at each \mathbf{q}_i gives the first order approximation to the actual surface at that point.

If the rotation that minimizes \mathcal{E}_{plane} is small (which is the case when the surfaces are already

close to the correct alignment), Equation 5.1 can be linearized by linearizing the rotation matrix \mathbf{R} . Using the small-angle approximation for rotations, we have:

$$\mathbf{R} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \quad (5.2)$$

which results in the following linear least squares problem:

$$\mathcal{E}_{plane} = \min_{\mathbf{c}, \bar{\mathbf{c}}} \sum_{i=1}^n ((\mathbf{p}_i - \mathbf{q}_i) \cdot \mathbf{n}_i + \mathbf{c} \cdot (\mathbf{p}_i \times \mathbf{n}_i) + \bar{\mathbf{c}} \cdot \mathbf{n}_i)^2. \quad (5.3)$$

The minimum of Equation 5.3 is found as the solution of a linear system $\mathbf{C}\mathbf{x} = \mathbf{b}$. This equation is virtually identical to the slippage constraint Equation 4.5 in Section 4.2 except for the presence of the residual vector:

$$\mathbf{b} = - \begin{bmatrix} (\mathbf{p}_1 - \mathbf{q}_1) \cdot \mathbf{n}_1 \\ \dots \\ (\mathbf{p}_n - \mathbf{q}_n) \cdot \mathbf{n}_n \end{bmatrix} \quad (5.4)$$

The matrix \mathbf{C} is the Hessian of the minimization problem, and it encodes how much the alignment error changes when the pointset \mathbf{P} is moved from its optimum alignment to \mathbf{Q} (where, ideally, the error is zero) by a transformation $[\Delta\mathbf{c}^T \ \Delta\bar{\mathbf{c}}^T]$:

$$\Delta\mathcal{E} = [\Delta\mathbf{c}^T \ \Delta\bar{\mathbf{c}}^T] \mathbf{C} \begin{bmatrix} \Delta\mathbf{c} \\ \Delta\bar{\mathbf{c}} \end{bmatrix} \quad (5.5)$$

As we saw in Chapter 3, certain types of geometry can lead to a Hessian that is not full rank. While this was a desirable property in the previous chapter, since it allowed us to identify slippable surfaces in the input point cloud, such surfaces can cause problems during registration.

5.1.1 Stability of the Solution

Let's examine what happens when the Hessian matrix \mathbf{C} is not full rank. There is no unique minimizing transformation to align \mathbf{P} to \mathbf{Q} since they are slippable surfaces, so any transformation in the null space of \mathbf{C} can be applied to the solution without changing the point-to-plane distance between the surfaces. In effect, the area in transformation space has a plateau around the optimum solution.

The problem arises when the input surfaces are mostly slippable, but actually contain enough features to constrain the slippable transformations. An example is two planar regions with indentations or incisions. Examples of such input are shown in Figure 5.1. If the size of these "lock and key" features is small, and only a subset of all the points are used in each iteration of the alignment algorithm, most of the points that enter the minimization will come from areas that are planar. If the data has no noise, the small number of points from the "lock and key" areas should be sufficient to resolve the ambiguity in the transform, and bring the surfaces into alignment. In reality, noise in the point positions and normals in the flat areas will overwhelm the contribution of the points sampled from the features, and the algorithm will fail to converge. In transformation space, there is a very shallow local minimum around the optimum transformation, which causes the algorithm to converge slowly. Additionally, the noise in the input causes this local minimum to have a smaller funnel of convergence.

There are several ways to approach the problem of shallow error landscape and false local minima in the presence of slippable surfaces. We can try to reduce the noise by smoothing the meshes. This can have an undesirable side effect of smoothing away the features that provide the constraints. We can try to use other constraints, such as color [10, 107]. We can also add more points to be used for minimization of Equation 5.3. Just adding more points will not improve convergence, since they are as likely to come from the flat areas as from the parts of the meshes that provide the constraints. We would like, therefore, to be able to detect whether the input data has any rotational or translational slippage, identify if there are any features that can better constrain the slippable transformations, and sample those features more densely.

5.2 Improving ICP's Stability Through Sample Selection

In this section we describe a greedy algorithm for selecting samples from the input meshes or point clouds in a way that will constrain the transformations that are close to slippable under the uniform sampling model. Our approach is based again on the idea of examining each surface separately, before it enters the registration. We will identify the potentially slippable transformations using the method in Section 4.2. Then, we will identify those points in each input mesh which best constrain each slippable transformation. These are the "lock and key" features that will help bring the mesh into alignment when it enters the registration pipeline. Our sampling algorithm will then try to distribute the points uniformly among the features, instead of just uniformly sampling the surfaces.

The two techniques that are the most similar to our approach are those of Simon [99] and Rusinkiewicz [85]. Simon developed several hill climbing algorithms for selecting a set of points on one of the input meshes that has the best potential for constraining all transformations when another mesh is aligned with it. These algorithms are especially well-adapted for dealing with noisy data, but do not address the problem when matching areas are only a subset of the input meshes. They are also designed for cases when only a very small number of points is required for alignment, and consider the input meshes in pairs, instead of pre-processing the meshes separately. As a result, they are too expensive to be used when large number of points are to be selected for minimization.

Rusinkiewicz [85] proposed a technique called normal-space sampling that is aimed at constraining translational sliding of the input meshes. When drawing samples from a mesh, the algorithm tries to ensure that the normals of the selected points uniformly populate the sphere of directions. The algorithm can be viewed as trying to equalize the eigenvalues of eigenvectors of \mathbf{C} that correspond to translations. We will use a similar approach to create a basis all six eigenvectors of \mathbf{C} .

5.2.1 A Measure of Stability

Since slippage is the result of shallow error landscape around the global minimum which is the solution of Equation 5.3, we can use the condition number of the Hessian \mathbf{C} to detect potential slippable transformations. Notice that we do not actually need to find the optimum alignment to detect slippable transformations, since the matrix \mathbf{C} is composed of just points of \mathbf{P} and normals of \mathbf{Q} . Let $\mathbf{x}_1 \dots \mathbf{x}_6$ be the eigenvectors of \mathbf{C} and $\lambda_1 \geq \dots \geq \lambda_6$ be the corresponding eigenvalues. As discussed in Section 4.2, if any of the λ_k are small compared of λ_1 , the corresponding eigenvector indicates a slippable transformation. Our measure of stability of ICP's solution will be the condition number of \mathbf{C} : $c = \frac{\lambda_1}{\lambda_6}$. We would like the Hessian to be well-conditioned, so the goal of our sampling strategy is to keep c as close to 1 as possible.

Although the Hessian of Equation 5.3 includes points from both \mathbf{P} and \mathbf{Q} (through their normals), we would like to be able to perform the slippage analysis and the sampling using each mesh separately. In the context of an actual registration problem each mesh \mathbf{P} is usually aligned against several \mathbf{Q} 's, so performing the sampling for each pair can become expensive. It turns out that looking at just \mathbf{P} alone is the correct approach, since we only want to add those points which will pull the meshes into alignment that is the global optimum. That is, we want to make the error landscape around the global minimum steep, while keeping the landscape shallow around the local minima to allow the algorithm to escape. The global minimum is achieved when the points in \mathbf{P} align exactly with their correct mates in \mathbf{Q} . In this case the normals in Equation 5.3 are the same for points \mathbf{p}_i and \mathbf{q}_i (since we assume that the two meshes are very similar in their area of overlap). Therefore, to constrain the correct transformations, we should analyze and constrain the covariance matrix that is computed using both points and normals from the mesh \mathbf{P} . This is exactly the same as the matrix in Equation 4.7, we will refer to it as \mathbf{C}_P to emphasize that although the registration consists of both \mathbf{P} and \mathbf{Q} , the sampling is done with respect to just \mathbf{P} . Similar to Section 4.2, we will pre-normalize all the points in \mathbf{P} so that their center of mass is at 0 and the its average radius is 1. This has the effect of equalizing the magnitude of the rotational and translational contributions of each point in the computation of \mathbf{C}_P .

5.2.2 Optimizing the Measure

We compute the covariance matrix \mathbf{C}_P according to Equation 4.7 and a uniform subset of points from \mathbf{P} . We usually select 10% of the points for the estimation step. We perform the eigenvector decomposition of \mathbf{C}_P and obtain the eigenvectors $\mathbf{x}_1 \dots \mathbf{x}_6$. In the previous chapter, we were interested in pointsets for which this matrix had a large condition number, since those corresponded to slippable surfaces. Here, since slippable surfaces result in slow convergence of the registration algorithm we instead want to create a pointset whose condition number is close to 1, i.e. as un-slippable as possible. We use the eigenvectors to compute a set of points that will result in a stable solution for the registration of \mathbf{P} and \mathbf{Q} as follows:

1. Let S be the initial set of candidate points. Ideally, S will contain all points on P that belong to the overlap area. We will discuss how to obtain the set S later. Form a 6-vector $\mathbf{v}_i = [\mathbf{p}_i \times \mathbf{n}_i^P, \mathbf{n}_i^P]$ for each point in S . Notice that here \mathbf{n}_i^P is the normal of the point \mathbf{p}_i as opposed to the normal of its closest point mate.
2. Form six sorted lists $L_1 \dots L_6$. Each list L_k contains the vectors \mathbf{v}_i sorted in decreasing order based on the magnitude of the dot product $\mathbf{v}_i \cdot \mathbf{x}_k$. The magnitude of this dot product determines how much a given point constrains each eigenvector \mathbf{x}_k . Hence, points in each list are sorted in order of decreasing contribution to geometric stability.
3. We now try to equally constrain all eigenvectors of \mathbf{C}_P . We will maintain an estimate of how each eigenvector is constrained by the already chosen points. Let $t_1 \dots t_6$ be the sums of $(\mathbf{v}_i \cdot \mathbf{x}_k)^2$ over the already chosen points. $(\mathbf{v}_i \cdot \mathbf{x}_k)^2$ is the amount of error incurred if the point \mathbf{p}_i is moved from its optimum position by the transformation \mathbf{x}_k . Therefore, we can think of these totals as our current estimate of the eigenvalues. We choose the next point from the sorted list that has the smallest total. This corresponds to the most unconstrained eigenvector.
4. Let \mathbf{p} be the chosen point. We compute $(\mathbf{v}_i \cdot \mathbf{x}_k)^2$ for each eigenvector \mathbf{x}_k and update the running totals.

Notice that this sampling strategy does not take into account the mesh \mathbf{Q} . We can think of this strategy as constraining the all transformations when \mathbf{P} is aligned to a copy of itself in the overlap region. Assuming that we are aligning two ideal, overlapping scans of the same object, this exactly corresponds to constraining the covariance matrix when we reach the global minimum. Once the points are sampled from \mathbf{P} , we can compute their closest points in \mathbf{Q} . We then proceed with the rest of the ICP algorithm as usual, now using the normals from \mathbf{Q} for the minimization of Equation 5.3.

The selection process above has to be performed for each input pointset or mesh, and uses all of the available points. Since nearest-neighbor search step in ICP can be expensive, to save time we often want to select only a subset of points in \mathbf{P} to register with \mathbf{Q} . In this case, processing all points of the input only to select a small subset of them leads to wasteful work. In the next section, we present several acceleration techniques that make the selection process take time proportional to the number of points that are selected from \mathbf{P} .

5.3 Accelerations and Enhancements

The most expensive step of the sampling algorithm is the Step 2, which sorts all points with respect to their product with each of the eigenvectors of \mathbf{C}_P computed in Step 1. To reduce the cost of these sorts, we instead sort the points into a specified number of bins. The points are left unsorted within each bin. Although not optimal, this still produces a good sampling, and the approximation error can be bounded by the size of the bins. Thus, the second step can be done in time proportional to $|\mathbf{P}|$.

The sampling strategy only uses the mesh \mathbf{P} to form the covariance matrix. Although this has the effect of assuming that the entire mesh \mathbf{P} will lie in the overlap area, it is also a valid sampling strategy if the "lock and key" features are spread relatively evenly over the surface of \mathbf{P} . By making this assumption, we are able to process each mesh in the registration pipeline separately, instead of performing the covariance computation and sampling for all pairs of meshes under registration.

If the feature distribution is highly uneven, or if extra processing is allowed to compute to covariance matrix for each pair of meshes, we can modify the steps of the above algorithm to compute the slippable motions of the overlap area between \mathbf{P} and \mathbf{Q} . We need to perform the overlap computation at two points in the algorithm: first to compute \mathbf{C}_P we should use only points from the part of \mathbf{P} that overlaps with \mathbf{Q} . Second, we should run the sampling algorithm only on points that belong to the overlap area.

The overlap can be computed only approximately, since accurately determining the part of \mathbf{P} that overlaps with \mathbf{Q} is exactly the goal of the registration algorithm. A common heuristic used in the local registration algorithms such as ICP, where \mathbf{P} and \mathbf{Q} are already close to their final registered pose, is to compute for each point \mathbf{p}_i the distance to its closest point in \mathbf{Q} . Points for which this distance falls below a certain threshold are assumed to lie in the approximate overlap area. Even using efficient nearest-neighbor data structure such as a k-d tree this can be expensive for large meshes. Additionally, it can be wasteful to test all point in \mathbf{P} for overlap if we only intend to use a small set of points for computing the aligning transform in each iteration of ICP.

To avoid testing all points in \mathbf{P} for overlap with \mathbf{Q} during the sampling algorithm, we implemented the following simple improvement in our system. Let S_P be a set of points randomly selected from \mathbf{P} . The size of S_P needs to be large enough to reliably determine the covariance matrix for the overlap region. The number of points depends on the size of the overlap between the two meshes, the resolution of the mesh, and the magnitude of the noise in the input data. In our experience, for meshes that overlap by 25% the number of points necessary to reliably compute the eigenvectors is on the order of several hundred. After performing the closest point test for this small set of initial points to discard points that fall outside the approximate overlap area, we use S_P to compute the covariance matrix \mathbf{C}_P and its eigenvectors.

The real source of inefficiency for computing the sampling for pairs of meshes comes in having to test all points of \mathbf{P} for overlap with \mathbf{Q} during the sorting step of the sampling algorithm. Instead, we process all points of \mathbf{P} regardless of whether they are in the overlap area. This allows us to delay the overlap test until Step 3. At that time, we can perform the

overlap test using the closest point search. If \mathbf{p}_i does not belong to the overlap area, we do not update the totals and choose the next point. This method is more efficient than using the brute-force approach of testing all points for overlap (if we use fast sorting), since we only perform as many nearest-neighbor tests as dictated by the sampling rate of our algorithm. Additionally, we can cache the computed closest points, and use them for minimization of Equation 5.3. In practice, this makes the amount of wasted work inversely proportional to the overlap area. With this implementation, ICP using our sampling strategy takes about 5 times longer per iteration than ICP using uniform sampling when the input meshes overlap by half their area.

5.4 Results

5.4.1 Using Stable Sampling for Pairwise Registration

We have applied our sampling algorithm to several types of synthetic and real data.

The first test case is two planar patches with two grooves forming an X (Figure 5.1(a)). Each patch has independently added Gaussian noise. This test case is similar to the one used by Rusinkiewicz [85] for normal-space sampling. Figure 5.1(b) shows the convergence rates for aligning these patches using uniform sampling, normal-space sampling, and our covariance-based sampling. Both normal-space and covariance sampling are able to find the correct alignment, while uniform sampling does not align the grooves correctly. Normal-space sampling takes more iterations to converge since distributing the points equally throughout the sphere of normals puts an equal number of points in the flat areas of the patches as it does in the grooves. Covariance sampling instead picks only those points that form a good basis for the normals.

Figure 5.2 shows the points picked by the sampling algorithm to constrain the eigenvectors of the covariance matrix. To simplify the visualization, we use a smaller version of the incised plane model and assume that the entire mesh is within the area of overlap. The initial covariance analysis reveals three unstable eigenvectors with approximately equal

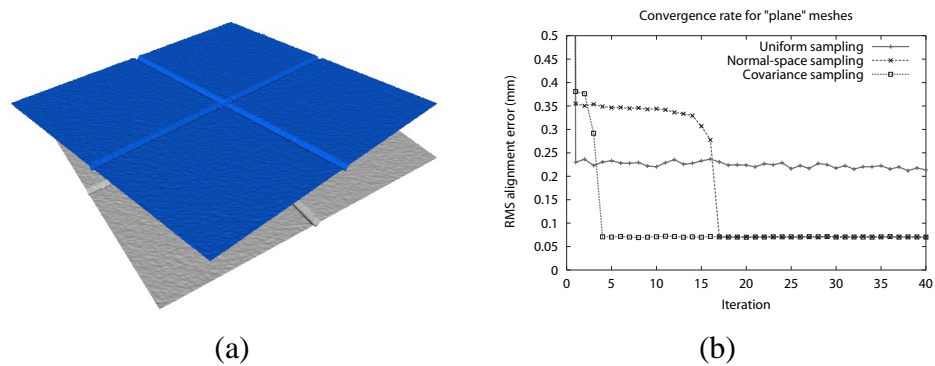


Figure 5.1: (a) Two planar patches with 1 mm deep grooves. Each patch has independently added zero-mean Gaussian noise with variance 0.05 mm. Initial condition number is 66.1. Condition number after selecting 30% of the points with our algorithm is 3.7. (b) Convergence rates using ICP with uniform, normal-space, and covariance sampling.

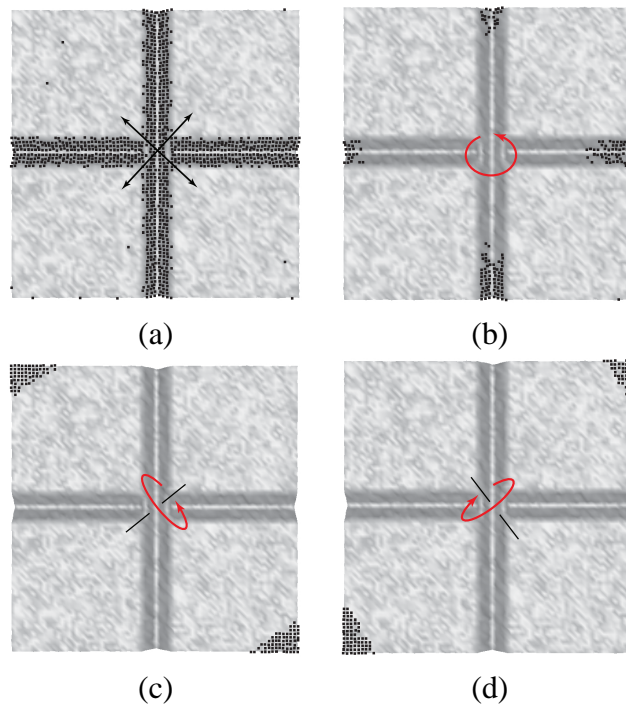


Figure 5.2: Points picked by our sampling algorithm for a patch with two grooves. (a) Points constraining two unstable translational eigenvectors. (b) Points constraining the unstable rotation. (c)-(d) Two remaining rotations are stable so they only require a few points. The eigenvector corresponding to translation in z is well constrained by the already picked points and does not contribute to the sampling.

eigenvalues: two translations in the xy plane and rotation around z . Notice that most of the points are picked from the areas in the grooves, since they are the ones that constrain the unstable eigenvectors. A few points from the corners are picked to additionally stabilize the rotations around the diagonals.

Figure 5.3 shows two spherical patches with grooves and noise. Here, covariance sampling is the only method that finds the pose that correctly aligns the grooves (Figure 5.3).

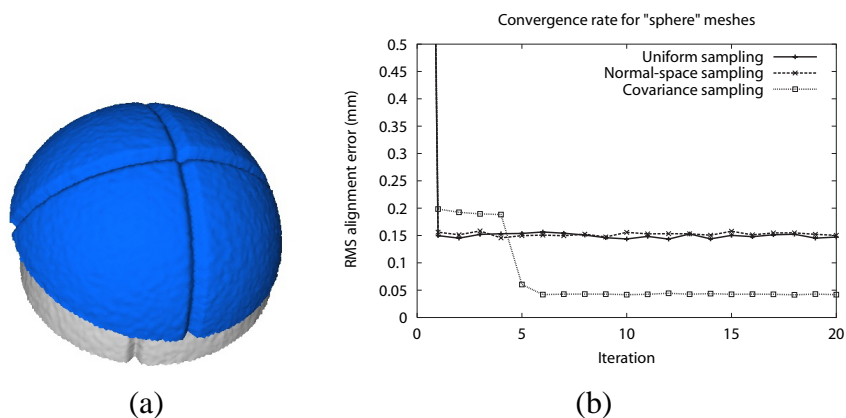


Figure 5.3: (a) Two spherical patches with 1 mm deep grooves. Each patch has independently added zero-mean Gaussian noise with 0.05 mm variance. This dataset has three slippable rotations. Initial condition number is 26.9. Condition number after selecting 30% of the points with our algorithm is 4.1. (b) Convergence rates for “incised sphere” meshes for uniform, normal space and covariance sampling

We have also applied our algorithm to real scan data. Figure 5.4(a) shows the sampling of two scans from the Forma Urbis Romae dataset [58]. Similar to the “incised plane” example, these meshes exhibit translational sliding in the plane and rotational sliding around the vector perpendicular to the plane of the meshes. Most of the samples are placed into the incisions on the scans to constrain the scans from sliding and rotating in their common plane. It took ICP 25 iterations to converge to the correct alignment (Figure 5.4(c)) from a rough manual positioning of the scans using our sampling strategy. Each input mesh contains about 300,000 points, and the algorithm was subsampling 10% of the points from each mesh to be used in alignment. With these settings, each iteration of ICP using our stable sampling took 5 seconds on a 400MHz Pentium II. One iteration of ICP using uniform sampling took 1.5 seconds, however when started from the same position, uniform

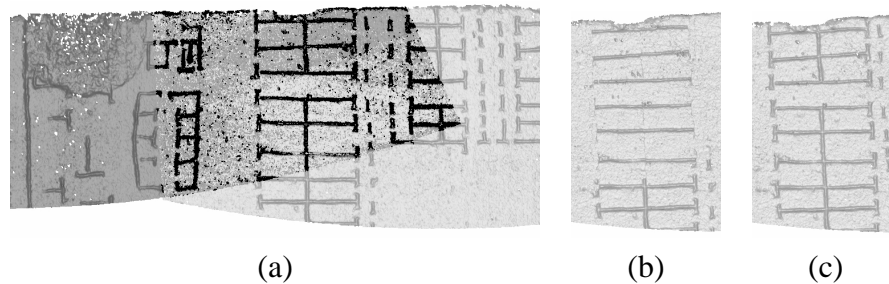


Figure 5.4: Aligning two scans of Forma Urbis Romae fragment 033abc. (a) Points selected by our sampling strategy are in black. Notice that in the outlined region there are relatively fewer constraints to prevent horizontal sliding than vertical sliding. (b) Therefore uniform sampling cannot align the vertical grooves in the outlined region as evidenced in this Z-buffer rendering of the two meshes by the fact that the vertical grooves are obscured. (c) Stable sampling produces the correct alignment making all the grooves visible.

sampling is unable to correctly align the vertical grooves (Figure 5.4(b)).

5.4.2 Using Stable Sampling in a Multi-view Registration System

In most 3D scanning systems, pairwise registration is usually followed by a multi-view global relaxation algorithm [80, 86], which spreads the accumulated alignment error over a set of views. Since a single mesh usually has several partners in this set, poor pose for one mesh can easily be propagated to its partners. Even if two views are aligned correctly, a shallow error landscape around the minimum can cause them to be pulled apart during global relaxation. Finally, if the output surface model is to be reconstructed from the input views by some sort of averaging [23], misaligned features can become blurred.

The stable sampling algorithm presented above has been integrated into the multiview registration system of Pulli [80]. The algorithm is based on using the point-pairs from the pairwise registration step to distribute the accumulated alignment error among a set of scans. In this case, using the point-pairs obtained by stable sampling instead of regular uniform sampling results in a lower residual error over the entire system. In particular, meshes tend to "hold together" better, especially in areas which are rich in features. This allowed the scans to slide in areas where there were few features, while preserving the alignment

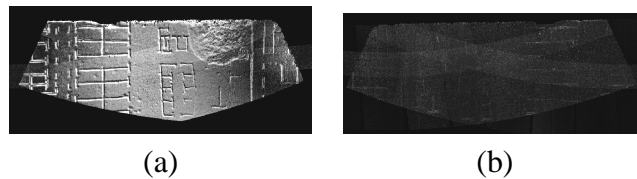


Figure 5.5: A visualization of residual error in the overlap portion of the pair of scans in Figure 5.4 after they and their partners have been processed by Pulli’s global registration [80]. Meshes in (a) were aligned using uniform sampling. Meshes in (b) using our geometrically stable algorithm. Error is in mm, black corresponds to 0, white to 1. The maximum error in (a) is over 1 mm, while the maximum error in (b) is 0.3 mm.

in feature rich areas. Figure 5.5 shows the residual error between the pair of scans from the Forma Urbis dataset examined above after the entire set of views has been processed by the global relaxation algorithm. Scans aligned with uniform sampling (Figure 5.5 (a)) have been pulled apart by as much as a millimeter, while those aligned by our algorithm (Figure 5.5 (b)) stayed together.

Finally, slippage analysis has been used in a multi-view non-rigid alignment systems by [49, 14] to determine featureless areas of the input scans. Both of the methods are based on subdividing the input meshes into small patches, which are assumed to be rigid. The patches are registered using rigid registration and merged together using a surface reconstruction algorithm in [49] or by warping the original input mesh to conform to the subdivided and registered patches using thin-plate splines in [14]. Slippage analysis is used to decide when to no longer subdivide a piece because it will not have enough constraining features to result in a correct rigid registration. Figure 5.6 shows the results of the dicing-based non-rigid alignment using the method in [49].

5.5 Limitations

The main limitation of the above algorithm is that the sampling cannot differentiate between points that come from features and points that look like features because they are the result of noise in the data. Since the algorithm prioritizes the points based on their influence of

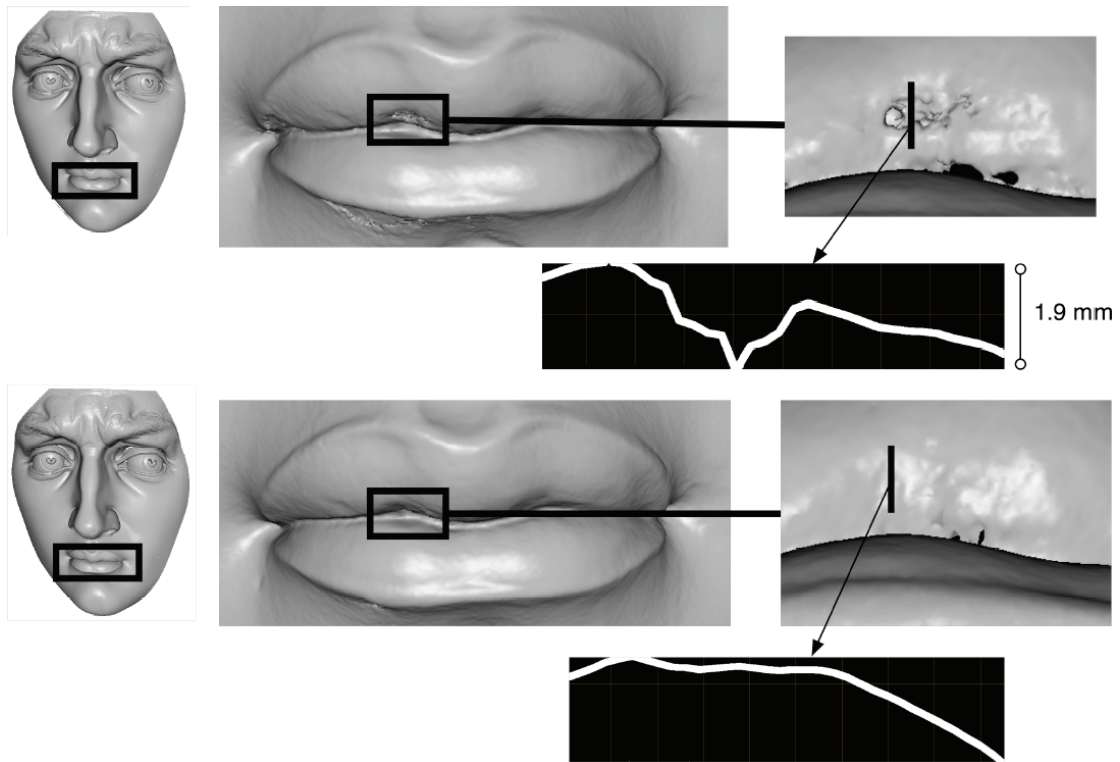


Figure 5.6: Results of dicing-based nonrigid registration algorithm using stability analysis [49]. Top: the lips exhibit a gross misalignment of 1.9 mm due to warp from incorrect scanner calibration, as shown by a depth plot across the highlighted segment. The misalignment results in surface irregularity after the final mesh is obtained using [23]. Bottom: using dicing and stability analysis the artifacts are removed.

the covariance matrix, it is possible that it can favor areas with significant noise, since the points there can look like good features for the algorithm to sample.

If the size of the features is large relative to the magnitude of the noise, applying simple smoothing the input data before applying the sampling can eliminate the selection of noisy points. However, if the size of the features is too small, or if the meshes are smoothed too much, the algorithm can still fail. Figure 5.7 shows success and failure cases of our sampling algorithm in the presence of noise.

Several more sophisticated approaches to the noise problem are also possible. Recently, feature-preserving smoothing methods have been developed [32, 51, 26], which produce

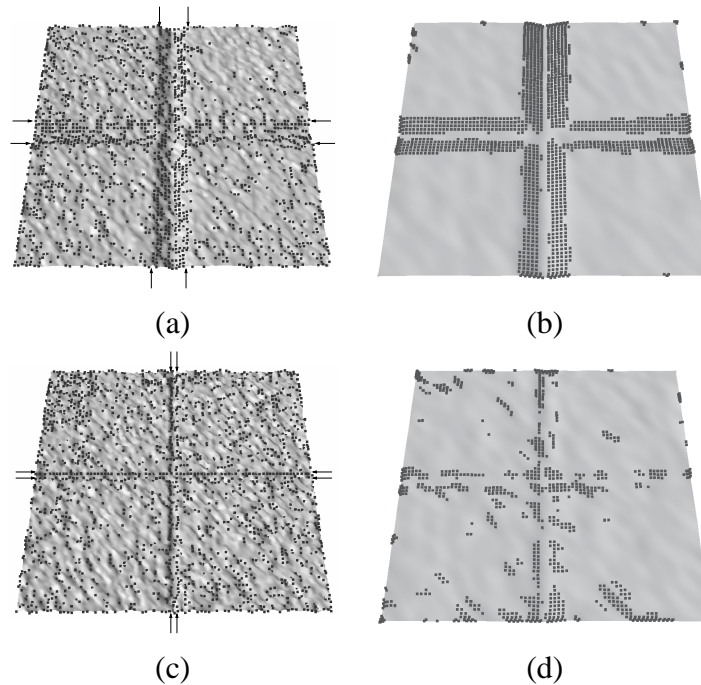


Figure 5.7: Effect of noise on covariance sampling. (a) A noisy patch with a cross in the center. The width of the grooves, indicated by black arrows, is 10 mm, the depth is 1 mm, the mean height of the noise is 0.2. Since the groove is shallow, the normals of points in the groove are comparable to normals of the noisy flat areas and the algorithm cannot distinguish between features and noise. (b) Performing 6 iterations of simple smoothing by averaging neighbors removes most of the noise but keeps the feature. (c) A similar patch, but the width of the groove is only 1 mm. (d) Since the size of the feature is comparable to the size of the noise, smoothing removes the noise and most of the feature, which means all areas of the patch now look identical, and covariance-based sampling fails.

significantly better results than simple Gaussian smoothing above. These methods can also be combined with a persistence framework in [27], to select points that are consistently picked as features through several levels of smoothing.

The implementation of the algorithm as described above has been largely aimed at aligning the scans from the Forma Urbis Romae dataset [58], which are mostly flat with groove-like features. As a result, the algorithm suffers from a limitation that by always picking the best-constraining points some areas of the input can be completely devoid of samples. Since the features are quite evenly distributed in our dataset, this was not a problem in our case. In

more difficult cases, an implementation based on importance sampling should be used. For example, one can sample directly from the space of all normalized 6-dimensional vectors, which would still pick out the features but also distribute some samples in other areas of the input. This approach is similar to the implementation of normal-space sampling in [85].

5.6 Summary

In this chapter we used the slippage analysis to develop a point selection strategy that improves geometric stability of the well-known Iterated Closest Point algorithm. We showed that slippable surfaces which lack any "lock and key" features have a plateau in the error landscape of the distance function, which means they cannot be aligned uniquely. When the input data has some features which should provide constraints for the alignment, but is close to slippable, the noise in the normals and point positions can overwhelm the constraints coming from the available features, which can lead to slow convergence of the ICP.

By performing slippage analysis of the data mesh, we were able to detect the cases when the input surfaces have only a few constraining "lock and key" features. We then developed a sampling strategy that selected from the data shape only those points which would provide aligning constraints, while avoiding noisy featureless areas.

While we only address the stability of pairwise alignment of meshes or point clouds in this chapter, a similar stability analysis can be applied to a larger collection of range images, e.g. during multi-view registration. The slippage analysis will need to be performed on the entire system, with points from all pairs of scans used in the computation of the covariance matrix to identify the potentially weakly-constrained inputs. Point selection for maximizing stability of a large set of scans is substantially more difficult than the pairwise step, since we have to consider how sliding of a single scan pair will affect the entire system.

Chapter 6

Conclusions and Future Work

"Would you tell me which way I ought to go from here?" asked Alice.

"That depends a good deal on where you want to get," said the Cat.

"I really don't care where" replied Alice.

"Then it doesn't much matter which way you go," said the Cat.

–Lewis Carroll

6.1 Summary

In this thesis, we examined several problems in geometry processing that deal with shape similarity, with the main focus on pairwise range image registration. We developed an algorithm for global alignment of two surfaces without relying on an initial estimate of their relative position and an improved local registration algorithm that can efficiently register a surfaces without many constraining features.

The overall theme of our approach to pairwise registration has been to first analyze each of the two input surfaces separately, as if it was being matched to a copy of itself. Since both surfaces come from scanning of the same physical object, they have to be similar across

their overlap area and we could gain insight on how a given surface will perform in a registration problem without having to know what it is being registered to. To characterize local surface geometry in a transformation-independent manner, we developed several new surface descriptors. The integral volume invariant captures local mean curvature information around each point in a multiscale and noise-robust way. The local slippage property characterizes local invariance of a surface patch under a rigid transformation. We used these surface descriptors to select feature points which were then used in pairwise registration.

The "feature analysis" theme of this work has focused on selecting points on the input surfaces in such a way as to make the overall registration problem easier. An integral volume invariant combined with persistence was used to select a small set of unique feature points and their potential correspondences. Since the feature points were selected from areas which were somehow salient with respect to the surface descriptor, which ensured that the resulting correspondence search space is not very large and can be explored efficiently. The local slippage property gives insight into the local error landscape of the registration problem around the globally best alignment. Slippage was used to select feature points in the local registration algorithm in a way that improved its convergence rate, especially in cases when the input data contains only a few areas with distinctive geometric information.

A perhaps somewhat unexpected use of the local slippage property was its application to reverse engineering. Since many machined parts consist of surfaces that are invariant under some rigid motion, known as kinematic surfaces, the local slippage descriptor was immediately applicable in a segmentation algorithm to decompose a point cloud into a set of kinematic components. This is not completely surprising however, since surface segmentation can be thought of as another example of the similarity problem, in this case within just a single object. The goal of many segmentation algorithms is to decompose an object into parts where all points within a single patch are somehow similar (e.g. approximated by a single kinematic surface). By posing segmentation as a similarity problem, we were also able to use our global registration algorithm for segmentation of an articulated object into rigid components. Here, the inputs were two positions of an articulated object, and the similarity problem was to find components that are the same after a rigid transformation.

Other shape similarity problems briefly touched on in this work included symmetry detection, non-rigid and multi-view registration.

6.2 Understanding Registration Algorithms

Shape similarity, matching, and registration are, at this point, fairly mature but still actively researched areas of geometry processing. As more and more new approaches to these problems are developed, there is a growing need in finding ways to perform comparative evaluation of new and existing methods.

While many practical methods have been developed for local and global registration, there has been relatively little work on deriving bounds on the performance of these optimization algorithms. Recently, bounds on the number of iterations of the Iterated Closest Point algorithm have been derived in [29, 5], which show that the number of iterations before converging to a local minimum is polynomial in the number of points, and independent of the dimensionality of the data. While of high theoretical significance, these bounds do not correlate well with the actual running times of ICP encountered in practice, where the number of iterations before convergence to a local minimum is often sub-linear.

Speed of convergence has also been analyzed from an optimization-based perspective. In [75] Pottmann and Hofer analyze how the rate of convergence of local registration changes when different metrics are used to measure squared distance between the input surfaces. An empirical study of ICP by Rusinkiewicz and Levoy [85] perform a similar analysis experimentally for the well known Iterated Closest Point algorithms, and show how the choice of error metric, correspondence computation and outlier rejection influences ICP's convergence.

The above studies deal with only limited cases of local registration, and are generally aimed at quantifying the rate of convergence of the optimization algorithm. In optimization-based registration algorithms, the focus is not only the rate of convergence but also the frequency of local minima in the error landscape, since those can prevent the algorithm from achieving

the correct result altogether. There is currently very little formal understanding of how the choice of the distance function, feature and correspondence selection methods and outlier rejection influence the frequency and size of local minima in the global registration error landscape. In particular the funnel of convergence analysis performed by Mitra et al. [64] for choosing the distance function can be applied to studying how the choice of different shape descriptors for computing correspondences can improve the convergence of different optimization-based registration methods.

Finally, as more and more matching algorithms are developed in the future, comparing their performance to existing work will become increasingly important. Currently, there are no established benchmarks for registration algorithms and some preliminary work in benchmarking general shape retrieval. In the next several sections, we will examine the current shape retrieval benchmarks, discuss some of their potential limitations, and propose some ideas for developing a new set of registration and shape matching tests.

6.3 Shape Similarity Benchmarks

The first question we have to ask ourselves when developing a shape retrieval benchmark is: how can we evaluate the performance of a shape retrieval algorithm. Given a query object, the goal of shape retrieval is to return a set of objects from the database that *best match the query*. But using what metric?

The first solution to benchmarking the growing number of shape retrieval algorithms was proposed by Shilane et al. The Princeton Shape Benchmark [97] is a set of 2000 3D models hand-classified into semantic groups. The classification groups the objects into functional classes (e.g. people, cars, tables etc.) of different levels of granularity (e.g. passenger cars, trucks etc). Two shapes are said to match if they belong to the same group, and not match otherwise. Shape retrieval algorithms can then be tested against this dataset by evaluating measures such as precision vs. recall plots, discounted cumulative gain etc. The Princeton Shape Benchmark is gaining wide use in the shape retrieval community. For example it was used as the test dataset in the first AIM@SHAPE contest for shape-based retrieval which

was held at the 2006 Shape Modeling International conference.

The drawback of a category-based benchmark such as this, is that the similarity decision is binary: either two objects are in the same class, in which case they are similar, or not. Most geometric matching algorithms assign a similarity *score* to a pair of shapes. The Princeton Shape Benchmark, however, does not include a quantification of geometric similarity against which an algorithm's results can be compared. In addition, when geometrically similar objects are placed into different categories, the benchmark can be misleading.

We propose that a shape matching benchmark should include some measure of geometric similarity in addition to a semantic grouping of shapes into functional classes. The question here is, of course, what to take as ground truth. For each pair of objects in the benchmark dataset set we need to define a notion of distance, which means we have to solve the shape matching problem again.

For total matching, we can imagine computing the ground truth by brute force, i.e. sampling the space of possible alignments between each pair of shapes and computing the distance between the surfaces for each transformation. Since it has to be done only once, and can be done offline, efficiency of comparison is not a big concern when creating the benchmark. The problem is harder for partial matching, since now we have to somehow identify the matching regions in each alignment configuration. As discussed in Chapter 2, given a proposed alignment a common heuristic is to define as matching regions the points on the two input shapes that lie within a threshold of each other.

Therefore, the shape benchmark that is based purely on geometry will consist of a set of shapes and a distance metric encoding the difference between all shape pairs. To evaluate the performance of shape retrieval algorithms, which often want to reflect not only geometric but also functional similarity among objects, the geometric benchmark can be combined with the labeling based dataset such as the Princeton Shape Benchmark. In fact, combining geometric and labeling-based methods have already proven to give good matching results, as shown in [96], where a labeling-based dataset was used to learn geometrically significant areas of input shapes, which were then used for shape matching.

6.4 Benchmarking Registration Algorithms

It should be easier to objectively compare performance of registration algorithms, since the ground truth can, with some work, be found exactly. The main component of a benchmark for registration should be a set of range images taken from known calibrated positions. This setting is similar to the test datasets used for evaluating stereo reconstruction algorithms, such as the frameworks proposed by Seitz et al. [91] for multi-view and Scharstein et al. [89] for binocular stereo.

The dataset developed by Seitz et al. for multi-view stereo consists of a number of images of an object taken from known calibrated camera positions. To compare the quality of the reconstruction, a 3D model of the shape obtained by laser scanning the object is also provided. Although this somewhat biases the tests since the 3D shape obtained from range scanning can potentially include errors, in practice this bias proved to be quite minor.

Obtaining 2D images of an object from known positions is relatively easy using a device such as the Stanford Spherical Gantry. It is currently not feasible to mount a high quality 3D scanner on such a device that would exactly track its position due to the size of most triangulation-based scanners. Instead, position of the scanner for each view can be computed by including a calibration target of known shape at a known location in the working volume of the scanner. The orientation of the target in each view will give the position and orientation of the scanner.

We can evaluate registration algorithms based on two metrics. First, given a calibrated dataset, we can immediately reconstruct a high quality 3D model of the scanned object, and use it as ground truth. Multi-view matching and reconstruction algorithms can then be tested based on how closely their reconstruction matches the target. Alternatively we can compare the transformations returned by a registration algorithm to the actual transformation between a pair (or among a set) of views which can be computed exactly from the calibrated dataset.

Once a faithful 3D model of the object is available we can imagine following two directions in generating test datasets:

- **Synthetic scans.** It is easy to generate a synthetic range image given a complete object model. All that is necessary is to determine which points on the model are seen by the virtual scanner, compute the distance from the scanner to each point, and return the resulting depth image. By modeling the scanner's parameters such as noise, grazing angle visibility and calibration distortion we can obtain synthetic datasets where all variables of the scanning process are known exactly.

Although such synthetic datasets will not reflect the kind of noise that is encountered in the actual scanning, a wide variety of datasets can be generated under easily controlled conditions. Therefore, they can be used to study how different registration algorithms adapt to various scanning conditions such as amount of noise, distortion, the size of overlapping regions and un-even sampling of the surfaces under precisely controlled conditions. In addition, it is likely that producing large datasets where position of each scan is known exactly by using a calibration target is a laborious process. Synthetic scans have the advantage that the position of each scan is known exactly at all times.

- **Real scans.** The disadvantage of a synthetic dataset is that we do not currently know how to accurately model every kind of noise that is present in the scanning process. Therefore, a calibrated dataset of real range scans is also necessary. Removing the calibration information from the set of scans used to build the ground truth 3D shape can provide one such dataset. However, we are likely to want to generate other range image datasets where the object is scanned under various challenging conditions, for example low laser intensity, poorly calibrated scanner, over and under sampling of various areas and the varying amounts of overlap.

Generating large sets of scans where the position of each range image is known exactly, while possible, is likely to be very laborious given currently available technology. However, given a good reconstruction of an object from a calibrated dataset, and a set of un-calibrated range images, the unknown position of each scan can be recovered by simple pairwise registration with the full object.

The next question to ponder is what kind of datasets should be generated. Registration algorithms should be tested against common difficulties that are encountered in 3D scanning. We describe some ideas on potential datasets below.

- **Ideal scanning conditions.** Current automatic algorithms work the best of datasets which contains few tens of scans, with pairs of scans overlapping by 30 – 70% of their extent, and with scans being evenly sampled. These conditions can be easily replicated in a synthetic dataset, and may be more difficult to replicate with real data: requiring some over-scanning and clean up.
- **Uneven sampling.** When scanning a physical object some areas often receive many more samples than others. Many registration algorithms, in particular those dealing with multi-view registration fail in the presence of heavy under or over sampling. We can create a dataset that simulates this by choosing several directions from which the object should be scanned multiple times.
- **Noise and miscalibration.** Amount of noise varies among different scanners, so ideally the registration benchmark should include datasets obtained from scanning the same object by several different 3D shape acquisition devices. In addition, subtle warps can be introduced into the data to see how well an algorithm adapts to the failure of the assumption that the aligning transformation is rigid. Such warp is often the result of incorrect calibration of scanner parameters, which often happens in large field scanning projects.
- **Varying overlap size.** Similar to over and under scanning, varying the size of the overlap between pairs of scans can generate a challenging dataset, in particular for methods based on global surface analysis. This can be easily achieved in either a real dataset by varying the amount by which objects are re-positioned after each scan or in a synthetic dataset by varying the distance between pairs of virtual scanner positions.
- **Grazing angles scans and holes.** In addition to noisy scans, scanning surfaces at a grazing angle, varying the laser intensity and scanning particularly convoluted self-occluding surfaces can result in a dataset where each range image contains significant

holes. Such datasets are useful to test how well shape matching adapts to even locally incomplete information.

- **Object size.** The final parameter that can be varied is the number of range images taken of each object, which usually depends on object size. This is particularly useful in evaluating multi-view registration algorithms which often scale poorly when dealing with a large number of range images.

Above we listed the kinds of datasets that can be generated for each object in the registration benchmark to simulate different scanning conditions. The next question to address is what kinds of objects should be included in the benchmark. These should reflect the kinds of surfaces often encountered in 3D shape acquisition. Therefore the dataset should include:

- **Smoothly varying surfaces.** The majority of objects that are scanned consist of relatively smooth surfaces with features of varying size that are uniformly distributed over the entire object. These should form the basic test for registration algorithms, while some of the more difficult but still common cases are discussed below.
- **Textured surfaces.** Surfaces with lots of surface texture, such as broken stone, chiseled marble, or toys with lots of surface detail can test the feature selection algorithms.
- **Smooth surfaces with few features.** Objects such as mechanical parts often consist of large planar or spherical shapes which are hard to register since there are few constraining features.
- **Symmetric objects.** These can be particularly problematic for multi-view registration algorithms, since many pairwise matches will have to be sorted through in a globally consistent manner.

Developing a benchmark such as the one described above is a fairly large undertaking, especially due to the need to provide good quality ground truth data and the large number of parameters that registration algorithms depend on which the data needs to address in a consistent and calibrated manner. However, once developed, it should provide a good tool

for comparing the performance of pairwise and multi-view registration algorithms, feature selection methods, and the effectiveness of various shape descriptors in a purely geometric fashion.

6.5 Future Work

With the development of fast 3D shape acquisition devices such as those based on structured light [84, 25, 33] or multi-view stereo [17], it is now possible to capture 3D geometry at the rate of up to 60 frames as second. This allows us to capture objects that are moving or deforming, since the scanners are fast enough to assume that the object is rigid for the duration of each frame. Developing methods specifically aimed at alignment, reconstruction and processing of moving and deforming shapes is a challenging research area currently lacking in effective algorithms. Datasets coming from real-time shape acquisition devices are usually noisy, fragmented and necessarily very large, presenting many interesting problems for researchers in geometry processing.

Bibliography

- [1] A. Adamson and M. Alexa. Ray tracing point set surfaces. In *Proc. Shape Modeling International*, pages 272–279, 2003.
- [2] M. Alexa. Linear combination of transformations. In *Proc. SIGGRAPH*, pages 380–387, 2002.
- [3] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, H. Pang, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Proc. NIPS*, 2004.
- [4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):408–416, 2005.
- [5] D. Arthur and S. Vassilvitskii. Worst-case and smoothed analyses of the ICP algorithm, with an application to the k-means method. In *Proc. FOCS*, 2006.
- [6] Protein Data Bank. <http://www.rcsb.org>.
- [7] G. Barequet and M. Sharir. Partial surface and volume matching in three dimensions. *PAMI*, pages 929–948, 1997.
- [8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, pages 509–522, 2002.

- [9] P. Benko, R. R. Martin, and T. Varady. Algorithms for reverse engineering boundary representation models. *Computer Aided Design*, 33(11):839–851, 2001.
- [10] F. Bernadini, I. Martin, and H. Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE Trans. on Visualization and Computer Graphics*, pages 318–332, 1991.
- [11] P. J. Besl and N. D. McKay. A method for registration of 3D shapes. *PAMI*, 14:239–256, 1992.
- [12] I. Boier-Martin. Domain decomposition for multiresolution analysis. In *Proc. Symposium on Geometry Processing*, pages 31–40, 2003.
- [13] O. Bottema and B. Roth. *Theoretical Kinematics*. Dover, 1979.
- [14] B. Brown and S. Rusinkiewicz. Non-rigid range-scan alignment using thin-plate splines. In *Proc. 3DPVT*, pages 759–765, 2004.
- [15] 3D Cafe. <http://www.3dcafe.com>.
- [16] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas. Persistence barcodes for shapes. In *Proc. Symposium on Geometry Processing*, pages 127–138, 2004.
- [17] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 22(3):569–577, 2003.
- [18] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proc. IEEE Conference on Robotics and Automation*, pages 2724 – 2729, 1991.
- [19] C. K. Chow, H. T. Tsui, and T. Lee. Surface registration using a dynamic genetic algorithm. *Pattern Recognition*, 37(1):105–117, 2003.
- [20] U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):516–524, 2004.

- [21] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions Graphics (Proc. SIGGRAPH)*, 23(3):905–914, 2004.
- [22] T. H. Cormen, C. H. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition)*. The MIT Press, 2001.
- [23] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH*, pages 303–312, 1996.
- [24] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *3D Data Processing, Visualization, and Transmission*, pages 428–861, 2002.
- [25] J. Davis, D. Nehab, R. Ramamoorthi, and S. Rusinkiewicz. Spacetime stereo: A unifying framework for depth from triangulation. (*PAMI*), 27(2):296–302, February 2005.
- [26] J. Diebel, S. Thrun, and M. Breunig. A bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics*, to appear.
- [27] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Computational Geometry*, 28:511–533, 2002.
- [28] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3D rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, pages 272–290, 1997.
- [29] E. Ezra, M. Sharir, and A. Efrat. On the ICP algorithm. In *Proc. ACM Symp. on Computational Geometry*, 2006.
- [30] Daniel Fischer, Peter Kohlhepp, and Frank Bulling. An evolutionary algorithm for the registration of 3-d surface representations. *Pattern Recognition*, 32(1):53–69, 1998.
- [31] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, pages 381–395, 1981.

- [32] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):950–953, 2003.
- [33] P. Fong and F. Buron. High-resolution three-dimensional sensing of fast deforming objects. In *Proc. IROS*, 2005.
- [34] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23(3):652–663, 2004.
- [35] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105, 2003.
- [36] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. ACM Symposium on Interactive 3D Graphics*, pages 49–58, 2001.
- [37] N. Gelfand and L. J. Guibas. Surface segmentation using local slippage analysis. In *Proc. Symposium on Geometry Processing*, 2005.
- [38] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *Proc. 3DIM*, pages 260–267, 2003.
- [39] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proc. Symposium on Geometry Processing*, 2006.
- [40] G. Godin, D. Laurendeau, and R. Bergevin. A method for registration of attributed range images. In *Proc. 3DIM*, pages 179–186, 1999.
- [41] W. E. L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal on Robotics Research*, 3(3):382–414, 1984.
- [42] W. E. L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *PAMI*, 9(4):469–482, 1987.

- [43] W. E. L. Grimson and T. Lozano-Perez. On the recognition of curved objects. *MIT AI Lab Memo*, (983), 1987.
- [44] Y. Hecker and R. Bolle. On geometric hashing and the generalized hough transform. In *IEEE SMC*, volume 24, pages 1328–1338, 1994.
- [45] Q.-X. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), 2006.
- [46] Q.-X. Huang and H. Pottmann. Automatic and robust multi-view registration. *Technical Report 152, Geometry Preprint Series, Vienna Univ. of Technology*, 2005.
- [47] D. Huber and M. Hebert. Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 21(7):637–650, 2003.
- [48] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. In *Proc. IJCV*, pages 195–212, 1990.
- [49] L. Ikemoto, N. Gelfand, and M. Levoy. A hierarchical method for aligning warped meshes. In *Proc. 3DIM*, pages 434–441, 2003.
- [50] A. Johnson and M. Hebert. Using spin-images for efficient multiple model recognition in cluttered 3-D scenes. *PAMI*, pages 433–449, 1999.
- [51] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22(3):943–949, 2003.
- [52] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *Proc. SIGGRAPH*, pages 954–961, 2003.
- [53] P. Koehl. Protein structure similarity. In *Current Opinion in Structural Biology*, pages 348–353, 2001.
- [54] J. Koenderink. *Solid shape*. MIT Press, 1990.

- [55] J. Koenderink and A. Doorn. Surface shape and curvature scales. In *Image and Vision Computing*, pages 557–565, 1992.
- [56] V. Kraevoy and A. Sheffer. Template based mesh completion. In *Proc. Symposium on Geometry Processing*, 2005.
- [57] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. SIGGRAPH*, pages 313–324, 1996.
- [58] 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: 3-D scanning of large statues. In *Proc. SIGGRAPH*, pages 131–144, 2000.
- [59] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proc. SIGGRAPH*, pages 362–371, 2002.
- [60] X Li and I. Guskov. Multiscale features for approximate alignment of point-based surfaces. In *Proc. Symposium Geometry Processing*, 2005.
- [61] S. Manay, B. Hong, A. Yezzi, and S. Soatto. Integral invariant signatures. In *Proc. ECCV*, pages 87–99, 2004.
- [62] E. M. Mikhail, J. S. Bethel, and J. C. McGlone. *Introduction to Modern Photogrammetry*. Wiley, 2001.
- [63] N. J. Mitra. *Thesis*. Ph.D. Dissertation, Stanford University, 2006.
- [64] N. J. Mitra, N. Gelfand, H. Pottmann, and L. J. Guibas. Registration of point cloud data from a geometric optimization perspective. In *Proc. Symposium on Geometry Processing*, pages 23–32, 2004.
- [65] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):560–568, 2006.

- [66] N. J. Mitra, A. Nguyen, and L. J. Guibas. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications*, pages 77–84, 2004.
- [67] 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.
- [68] F. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. on Vis. and Computer Graphics*, pages 191–205, 2003.
- [69] G. Papaioannou, E.-A. Karabassi, and T. Theoharis. Virtual archaeologist: Assembling the past. *IEEE Computer Graphics and Applications*, 21(2):53–59, 2001.
- [70] G. Papaioannou, E.-A. Karabassi, and T. Theoharis. Reconstruction of three-dimensional objects through matching of their parts. *PAMI*, 24(1):114–124, 2002.
- [71] F.C. Park and B. Ravani. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics*, 16(3):277–295, 1997.
- [72] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–289, 2003.
- [73] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. J. Guibas. Example-based 3D scan completion. In *Proc. Symposium on Geometry Processing*, pages 23–32, 2005.
- [74] S. Podolak, P. Shilane, A. Golovinsky, S. Rusinkiewicz, and T. Funkhouser. A planar reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):549–559, 2006.
- [75] H. Pottmann and M. Hofer. Geometry of the squared distance function to curves and surfaces. In H. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 223–244. Springer, 2003.
- [76] H. Pottmann, M. Hofer, B. Odehnal, and J. Wallner. Line geometry for 3D shape understanding and reconstruction. In *Proc. ECCV*, pages 297–309, 2004.

- [77] H. Pottmann, Q.-X. Huang, Y.L. Yang, and S. Kolpl. Integral invariants for robust geometry processing. Technical Report 146, Geometry Prepring Series, Vienna Univ. of Technology, November 2005.
- [78] H. Pottmann and T. Randrup. Rotational and helical surface approximation for reverse engineering. *Computing*, 60, 1998.
- [79] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer, 2001.
- [80] K. Pulli. Multiview registration for large datasets. In *Proc. 3DIM*, pages 160–168, 1999.
- [81] Part Solutions CAD Models Repository. <http://www.part-solutions.com>.
- [82] Stanford 3D Scanning Repository. <http://graphics.stanford.edu/data/3Dscanrep/>.
- [83] VirtualCAD Parts Repository. <http://www.virtualcad.com>.
- [84] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. In *Proc. SIGGRAPH*, 2002.
- [85] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3DIM*, pages 145–152, 2001.
- [86] Krishnan S., P. Lee, J. Moore, and S. Venkatasubramanian. Global registration of multiple 3D point sets via optimization-on-a-manifold. In *Proc. Symposium on Geometry Processing*, pages 187–196, 2005.
- [87] P.V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proc. SIGGRAPH*, pages 409–416, 2002.
- [88] N. Sapidis and P. Besl. Direct construction of polynomial surfaces from dense range images through region growing. *ACM Transactions on Graphics*, 14(2):171–200, 1995.

- [89] D. Scharsten and R. Szelinski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 41(1):7–42, 2002.
- [90] D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979.
- [91] S. Seitz, B. Curless, J. Diebel, D. Scharsten, and R. Szelinski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR*, 2006.
- [92] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23(3):878–887, 2004.
- [93] G. C. Sharp, S. W. Lee, and D. K. Wehe. Icp registration using invariant features. *PAMI*, 24(1):90–102, 2002.
- [94] G. C. Sharp, S. W. Lee, and D. K. Wehe. Multiview registration of 3D scenes by minimizing error between coordinate frames. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*, pages 587–597, 2002.
- [95] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, pages 888–905, 2000.
- [96] P. Shilane and T. Funkhouser. Selecting distinctive 3d shape descriptors for similarity retrieval. *Proc. Shape Modeling International*, 2006.
- [97] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The princeton shape benchmark. In *Proc. Shape Modeling International*, 2004.
- [98] L. Silva, O. R. P. Bellon, and K. L. Boyer. Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. *PAMI*, 27(5):762–776, 2005.
- [99] D. Simon. *Fast and Accurate Shape-Based Registration*. Ph.D Dissertation. Carnegie Mellon University, 1996.
- [100] Turbo Squid. <http://www.turbosquid.com>.
- [101] V. Srinivasan. *Theory of Dimensioning*. Marcel Dekker, 2003.

- [102] G. Stockman. Object recognition and localization via pose clustering. *CVGI*, pages 361–387, 1987.
- [103] J. W. H. Tangelder and R. C. Veltkamp. A survey and content based 3D shape retrieval methods. In *Proc. Shape Modeling International*, pages 145–156, 2004.
- [104] S. Thrun and B. Wegbreit. Shape from symmetry. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005.
- [105] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- [106] T. Varady, R. R. Martin, and J. Cox. Reverse engineering of geometric models - an introduction. *Computer-Aided Design*, 29(4):255–269, 1997.
- [107] S. Weik. Registration of 3-D partial surface models using luminance and depth information. In *Proc. 3DIM*, pages 93–100, 1997.
- [108] H. Wolfson and I. Rigoutsos. Geometric hashing: an overview. *IEEE Computer Science and Engineering*, pages 10–21, 1997.
- [109] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Smoothing by example: Mesh denoising by averaging with similarity-based weights smoothing by example: Mesh denoising by averaging with similarity-based weights. In *Proc. Shape Modeling International*, 2006.

Appendix A

Properties of the Integral Volume Invariant

In this section we derive the properties of the integral volume invariant used in Chapter 3. The derivations are due to Pottmann and are covered in more detail in [77].

A.1 Relation between curvature and the integral area invariant of a 2D curve

Given a closed planar curve c , and a point $\mathbf{p} \in c$, one takes the circular disk $B_r(\mathbf{p})$ with center \mathbf{p} and radius r and uses it as the region of influence for the computation of an integral invariant. In [61], the authors define the *integral area invariant* at point \mathbf{p} as

$$I^r(\mathbf{p}) := \int_{\bar{c} \cap B_r(\mathbf{p})} d\mathbf{x}. \quad (\text{A.1})$$

Here \bar{c} denotes the interior of c . I^r is the area of the intersection $\bar{c} \cap B_r(\mathbf{p})$ of the curve's interior and the influence disk $B_r(\mathbf{p})$ centered at the point \mathbf{p} . This is the two-dimensional analogue of the integral volume invariant defined in Chapter 3.

In the following, curvatures of the boundary curve c of a 2D domain D are always computed with the sign that arises from an orientation of the boundary with help of normals pointing into the interior of D . This implies that the boundary of a convex domain D gets nonnegative curvature. In 3D, we do the same and thus get nonnegative principal curvature for the boundary surface of a convex domain.

We will first show the relationship between the 2D integral area invariant at \mathbf{p} and local curvature κ at point \mathbf{p} .

$$I^r(\mathbf{p}) = \frac{\pi}{2}r^2 - \frac{\kappa}{3}r^3 + O(r^4). \quad (\text{A.2})$$

As expected, the term with r^2 describes the area of the semicircle (it belongs to an approximation of the curve by its tangent); the first correction term (with r^3) involves curvature κ . For a proof, consider a second order Taylor approximant of the curve at \mathbf{p} , expressed in the Frenet frame (x, y) . It reads

$$y = \frac{\kappa}{2}x^2.$$

It is sufficient to work with this parabola p instead of c . The (real) intersection points of the circle $x^2 + y^2 - r^2 = 0$ (boundary of B_r) with p have coordinates $(\pm t, \kappa t^2/2)$. It is easy to show that $t = r + O(r^3)$. To obtain the correction term, we have to compute

$$T = 2 \int_0^t \frac{\kappa}{2}x^2 dx + 2 \int_t^r \sqrt{r^2 - x^2} dx = \frac{\kappa t^3}{3} + 2 \int_t^r \sqrt{r^2 - x^2} dx.$$

Since for $x > t$ and $\kappa > 0$, the parabola is above the circle of radius r , we have

$$\int_t^r \sqrt{r^2 - x^2} dx < \int_t^r \frac{|\kappa|}{2}x^2 dx = \frac{|\kappa|}{6}(r^3 - t^3) = O(r^5).$$

Because of $t^3 = r^3 + O(r^5)$, we finally get $T = \frac{\kappa}{3}r^3 + O(r^5)$, which proves equation (A.2). (This does not prove $I^r = \frac{\pi}{2}r^2 - \frac{\kappa}{3}r^3 + O(r^5)$, since we have neglected third order terms of the curve, which might come in). We also see that the correction term may as well be derived from

$$\int_{-r}^r \frac{\kappa}{2}x^2 dx.$$

A.2 Mean curvature and the volume descriptor

Fortunately, the approach outlined in the previous section carries nicely over to the 3D case. With H as mean curvature at the point \mathbf{p} of consideration, the volume descriptor to kernel radius r satisfies

$$V^r = \frac{2\pi}{3}r^3 - \frac{\pi H}{4}r^4 + O(r^5). \quad (\text{A.3})$$

Obviously, the leading term is the volume of a half ball of radius r . The correction term involves the mean curvature H at the considered surface point (center \mathbf{p} of the ball). The proof resembles the 2D case. We use the principal frame at \mathbf{p} as coordinate frame and approximate the given surface up to second order by the paraboloid P ,

$$z = \frac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2).$$

Here, κ_1, κ_2 denote the principal curvatures at \mathbf{p} . In a very similar way as in the 2D case we show that it is actually not necessary to compute the intersection curve of the paraboloid P and the sphere $x^2 + y^2 + z^2 - r^2 = 0$ of radius r , since the part of the volume beyond the intersection is of order $O(r^5)$ and thus not relevant for our estimate. Hence, the computation of the desired correction term can be performed with the integral,

$$T = \frac{1}{2} \int_D (\kappa_1 x^2 + \kappa_2 y^2) dx dy,$$

taken over the circular disk $D : x^2 + y^2 \leq r^2$. Using polar coordinates (R, ϕ) , we get

$$\begin{aligned} T &= \frac{1}{2} \int_0^{2\pi} \int_0^r (\kappa_1 R^2 \cos^2 \phi + \kappa_2 R^2 \sin^2 \phi) R dR d\phi = \\ &= \frac{r^4}{8} \int_0^{2\pi} (\kappa_1 \cos^2 \phi + \kappa_2 \sin^2 \phi) d\phi = \frac{\pi(\kappa_1 + \kappa_2)r^4}{8} = \frac{\pi H r^4}{4}. \end{aligned}$$

A.3 Influence of a surface perturbation on the volume descriptor

The volume descriptor computes the volume of an object bounded by part of a sphere and a part P of a given surface. Let us discuss how the descriptor changes if P undergoes some perturbation. The perturbation moves \mathbf{p} to \mathbf{p}' and thus the kernel ball $B_r(\mathbf{p})$ undergoes a translation to $B_r(\mathbf{p}')$. The latter intersects the perturbed surface in a patch P' . Translating $B_r(\mathbf{p}')$ back to $B_r(\mathbf{p})$ moves P' to a patch P^* . Apart from a negligible part along the intersection with the ball $B_r(\mathbf{p})$, the change of the volume is given by the oriented volume V^* between P and the surface patch P^* . Let us assume that P is given as a parametric surface $\mathbf{s}(u, v)$, parameterized over a domain D . We express the perturbation towards P^* with a height field $\tau(u, v)$ in normal direction of P . Thus, P^* is given by

$$\mathbf{x} = \mathbf{s}(u, v) + \tau(u, v)\mathbf{n}(u, v),$$

where \mathbf{n} denote unit surface normals. Using a (u, v, w) parameter space, which is related to Cartesian coordinates via $\mathbf{x} = \mathbf{s}(u, v) + w\mathbf{n}(u, v)$, we can compute the volume as

$$\begin{aligned} V^* &= \int_D \int_0^{\tau(u, v)} (\mathbf{s}_u + w\mathbf{n}_u, \mathbf{s}_v + w\mathbf{n}_v, \mathbf{n}) dw du dv = \\ &\int_D \left[\int_0^{\tau(u, v)} [(\mathbf{s}_u, \mathbf{s}_v, \mathbf{n}) + w(\mathbf{n}_u, \mathbf{s}_v, \mathbf{n}) + w(\mathbf{s}_u, \mathbf{n}_v, \mathbf{n}) + w^2(\mathbf{n}_u, \mathbf{n}_v, \mathbf{n})] dw \right] dudv. \end{aligned}$$

Here, $\mathbf{s}_u = \partial\mathbf{s}/\partial u$, etc., and $(\mathbf{a}, \mathbf{b}, \mathbf{c}) := \det(\mathbf{a}, \mathbf{b}, \mathbf{c})$. It is now convenient to use a principal curvature parametrization $\mathbf{s}(u, v)$, which implies $\mathbf{n}_u = -\kappa_1\mathbf{s}_u$, $\mathbf{n}_v = -\kappa_2\mathbf{s}_v$. Moreover, we note that $(\mathbf{s}_u, \mathbf{s}_v, \mathbf{n})$ equals the surface area element dA of the surface patch P . Thus, we obtain (with Gaussian curvature $K = \kappa_1\kappa_2$),

$$V^* = \int_P \left(\int_0^{\tau(u, v)} [1 - 2wH + w^2K] dw \right) dA,$$

and finally,

$$V^* = \int_P \tau(u, v)dA - \int_P \tau^2(u, v)H dA + \frac{1}{3} \int_P \tau^3(u, v)K dA. \quad (\text{A.4})$$

The perturbation changes the volume descriptor from V to $V - V^*$. For a perturbation with zero mean, we have $\int_P \tau(u, v) dA = 0$, and thus the perturbation $\tau(u, v)$ enters only in powers ≥ 2 , which shows the robustness of the volume descriptor.

Appendix B

Bounds on cRMS and dRMS Error Metrics

In this chapter we prove the upper and lower bounds on the distance root mean squared error used in Chapter 3. Assume we have two sets of points $\mathbf{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ and $\mathbf{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_n\}$ in correspondence.

B.1 Upper bound

Let \mathbf{h}_i be a displacement vector that encodes the misalignment between \mathbf{p}_i and \mathbf{q}_i .

$$\mathbf{q}_i = \mathbf{p}_i + \mathbf{h}_i. \tag{B.1}$$

If we assume that \mathbf{P} and \mathbf{Q} are in optimal alignment, their centroids align, which means:

$$\sum_{i=1}^n \mathbf{h}_i = 0, \text{ and } \sum_{i=1}^n \sum_{j=1}^n \mathbf{h}_i^T \mathbf{h}_j = 0. \tag{B.2}$$

Then,

$$\begin{aligned}
n^2 \cdot \text{dRMS}^2 &= \sum_{i=1}^n \sum_{j=1}^n (\|\mathbf{q}_i - \mathbf{q}_j\| - \|\mathbf{p}_i - \mathbf{p}_j\|)^2 = \\
&\sum_{i=1}^n \sum_{j=1}^n (\|(\mathbf{p}_i - \mathbf{p}_i) + (\mathbf{h}_i - \mathbf{h}_j)\| - \|\mathbf{p}_i - \mathbf{p}_j\|)^2 \leq \\
&\sum_{i=1}^n \sum_{j=1}^n (\|\mathbf{h}_i - \mathbf{h}_j\|)^2 = \\
&\sum_{i=1}^n \sum_{j=1}^n (\|\mathbf{h}_i\|^2 + \|\mathbf{h}_j\|^2) - 2 \sum_{i=1}^n \sum_{j=1}^n \mathbf{h}_i^T \mathbf{h}_j = \\
&2n \sum_{i=1}^n \|\mathbf{h}_i\|^2 = 2n^2 \cdot \text{cRMS}^2.
\end{aligned}$$

Therefore, the upper bound on dRMS is given by:

$$\text{dRMS} \leq \sqrt{2} \cdot \text{cRMS}. \quad (\text{B.3})$$

B.2 Lower bound

Let

$$h_{ij} = \left| \|\mathbf{q}_i - \mathbf{q}_j\| - \|\mathbf{p}_i - \mathbf{p}_j\| \right|. \quad (\text{B.4})$$

Then

$$n^2 \cdot \text{dRMS}^2 = \sum_{i=1}^n \sum_{j=1}^n h_{ij}^2. \quad (\text{B.5})$$

We prove the lower bound on dRMS by construction. Pick two correspondences $(\mathbf{p}_1, \mathbf{q}_1)$ and $(\mathbf{p}_2, \mathbf{q}_2)$, and position the points such that points \mathbf{p}_1 and \mathbf{q}_1 coincide and the lines defined by $(\mathbf{p}_1, \mathbf{p}_2)$ and $(\mathbf{q}_1, \mathbf{q}_2)$ align with each other. Since this cannot be better than the best alignment, the maximum distance between \mathbf{p}_2 and \mathbf{q}_2 in this case is $2h_{12}$ by definition.

Now we add a third point-pair $(\mathbf{p}_3, \mathbf{q}_3)$. Since the distance between any pair of points in \mathbf{P} is greater than the exclusion radius R_E and less than the extent of the shape L , we have

$$\|\mathbf{p}_3 - \mathbf{q}_3\| \leq \frac{L}{R_E} (h_{12} + h_{23} + h_{13}). \quad (\text{B.6})$$

Adding the next point-pair $(\mathbf{p}_4, \mathbf{q}_4)$ completes the frame, and analogously we have

$$\|\mathbf{p}_4 - \mathbf{q}_4\| \leq \frac{L}{R_E} \sum_{i=1}^4 \sum_{j=1}^4 h_{ij} = H. \quad (\text{B.7})$$

We choose $\mathbf{p}_1, \dots, \mathbf{p}_4$ such that H is minimized over all such choices. We can now use this this frame to align the two point sets and compute the distance between \mathbf{P} and \mathbf{Q} as the difference of offsets to the basis points. Since cRMS is computed after an optimal alignment of \mathbf{P} and \mathbf{Q} it is necessarily smaller than the distance computed using our frame. With some abuse of notation, we will now assume that \mathbf{P} has been transformed as described above.

$$\begin{aligned} n \cdot \text{cRMS}^2 &\leq \sum_{i=1}^n \|\mathbf{p}_i - \mathbf{q}_i\|^2 \leq \\ &\frac{L^2}{R_E^2} \sum_{i=1}^n (H + h_{i1} + h_{i2} + h_{i3} + h_{i4})^2 \leq \\ &\frac{L^2}{R_E^2} \cdot k_1 \cdot \sum_{i=1}^n (H^2 + h_{i1}^2 + h_{i2}^2 + h_{i3}^2 + h_{i4}^2) \leq \\ &\frac{L^2}{R_E^2} \cdot k_2 \cdot \sum_{i=1}^n \sum_{j=1}^4 h_{ij}^2 = \frac{L^2}{R_E^2} \cdot k_2 n^2 \cdot \text{dRMS}^2. \end{aligned}$$

Which gives the lower bound on dRMS with cRMS:

$$\frac{1}{\frac{L}{R_E} \cdot k \cdot \sqrt{n}} \text{cRMS} \leq \text{dRMS}. \quad (\text{B.8})$$