FEATURE POINTS ON POINT-BASED SURFACE AND THEIR APPLICATIONS

by

Xinju Li

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2009

Doctoral Committee:

Assistant Professor Igor Guskov, Chair
Professor Satinder Singh Baveja
Professor Robert Krasny
Associate Professor Gregory H. Wakefield
Assistant Professor Clayton D. Scott

DEDICATION

To my parents.

ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge many individuals who supported my work. First and foremost, my deep gratitude and appreciation go to my thesis adviser Igor Guskov. Igor is smart, knowledgeable, and supportive. I enjoyed his guidance and inspiration for the past years. His supervision on my research and his funding support made this work possible.

I also want thank to the rest of my committee members for their time, effort, kindly help, and valuable comments.

To Dr. Jacob Barhak and Geoffrey W. Blake from Engineering Research Center in Reconfigurable Manufacturing Systems (ERC/RMS) at the University of Michigan. Their great work and discussion are invaluable to me. Thanks Reuven Katz for his advices on my project and the financial support from ERC/RMS Center.

To the friendly and helpful staffs in CSE Division of the EECS Department: Cynthia Watts, Rita Rendell, and many others.

To my friends at University of Michigan: Jing Zhang, Min Zhu and Yunyao Li and many others. Many thanks for their friendship, encouragement, and support throughout the past years.

Finally, I would like to thank my parents who are always there for me. To my sisters and two brother in laws for their encouragement and generous help.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Surface acquisition methods are becoming popular for many practical applications in manufacturing, art, and design. With the growing amount of geometric data, efficient tools for matching and recognition of complex surfaces become more important. In order to achieve such efficiency, many existing methods operate on a limited subset of feature points sampled from the surfaces, often randomly.

In this thesis, we introduce an alternative way to achieve the efficiency by detecting a set of salient feature points from complex 3D geometry data. The method builds a scale-space representation for the input surface and use local extrema of the difference along normal direction between neighbor scales as salient points (or features). For every feature detected, we define a point signature vector that reflects the variation of local surface normals. Salient points and their signatures are invariant to rigid transformation and are stable under surface variation. This provides a good basis for a single feature to find its correct match with good probability in a large database of features.

We show the effectiveness of selected features and their signatures by applying them to solve several 3D computer vision problems. We first use the features for pairwise surface registration that matches two partial surface scans or matches a partial scan to its CAD model. The result of pairwise surface matching is used to align multi-view scans of the same object to reconstruct the complete model. We also use the selected features and their signatures for 3D object recognition, and evaluate their performance on both synthetic and real world 3D data with clustering and occlusion. Experiments demonstrate that the proposed features and signatures are robust for the applications.

CHAPTER I

INTRODUCTION

The traditional way of acquiring 3D shapes in computer vision is by recovering depth information from 2D camera images. With the advance of affordable 3-D range sensor systems available to companies and researchers [10], the amount of available complex geometric data grows fast. It brings the need for efficient shape matching and retrieval tools in several application fields such as surface registration, shape alignment, and object recognition.

One way to achieve efficiency is to augment each shape with a set of its salient features. Such technologies have been used widely in computer vision field in 2D image domain [13, 9, 78, 80, 55, 51]. One example is scale-invariant feature transform (SIFT) [51] that detects scale-invariant features which are proved to be efficient and robust for 2D image alignment and recognition. However, related research in 3-D is a few years behind its 2D counterparts. The main contribution of this thesis is the development of a geometry processing method for efficient detection of robust local feature sets from dense geometry data as well as the exploration of their applications.

In real world, shapes are measured from one point of view at one time. For most devices, such as laser scanner, measured data is not complete due to limitation of viewpoint, background clutter, and self-occlusion. Figure 1.1 shows examples of datasets acquired using different devices and captured from different view points (Stanford Bunny [1] datasets acquired by laser scanners and a 3D printer). For such partial data, global surface description often used for surface matching and recognition cannot be used. A better solution,

proposed in this document, is to use a collection of local surface description instead. Since the features are defined only on the local properties of the 3D shape, they can deal with incomplete surfaces with artificial boundaries and holes (Figure 1.1) efficiently.

Surface measurement using devices like laser scanner is a discrete process. Due to the imperfection of measuring instrument, the datasets acquired with different devices have different surface density and noise level. As you can see in Figure 1.1, (C) is much coarser and noisier than (B). Our proposed feature detector in this thesis is based on a multi-scale smoothing technique. Therefore it can eliminate the influence of surface noise to some degree and is able to handle surface detail on different scales. In this Chapter, we will briefly introduce the feature points and their applications.

1.1 Point-based surfaces

In computer graphics and computer vision community, there are several ways to represent surfaces. One example is tensor product spline surfaces, which use control points and interpolation or approximation techniques to add more points on the surface. Spline surfaces have their applications in animation and industrial design. Polygon meshes became dominant for surface representation because they are more flexible to handle both simple and complex geometries[64]. Polygon mesh gives the combined information of position of surface vertices as well as the connection between vertices. It includes a list of vertices that specifies the sampled surface points and a list of indices describing which vertices are linked into polygons. Meshes usually consist of convex polygons such as triangles and quadrilaterals.

Point cloud, also know as point-based surface (PBS), is a set of points in 3D space that describe the geometry of a given shape. It has become popular as the surface representation in computer graphics for many applications, such as rendering [30], shape modeling[63], and geometry processing [61, 60]. Figure 1.2 shows the example of a polygon surface and a point cloud. Points are the most basic geometry primitives that can be used to represent

shape. Compared with mesh-based surface, point cloud is simpler because it does not require global connectivity graph. All the geometry processing algorithms that work on point cloud can be adapted for mesh-based models. Thus, we choose point cloud as the surface presentation over mesh-based surface.

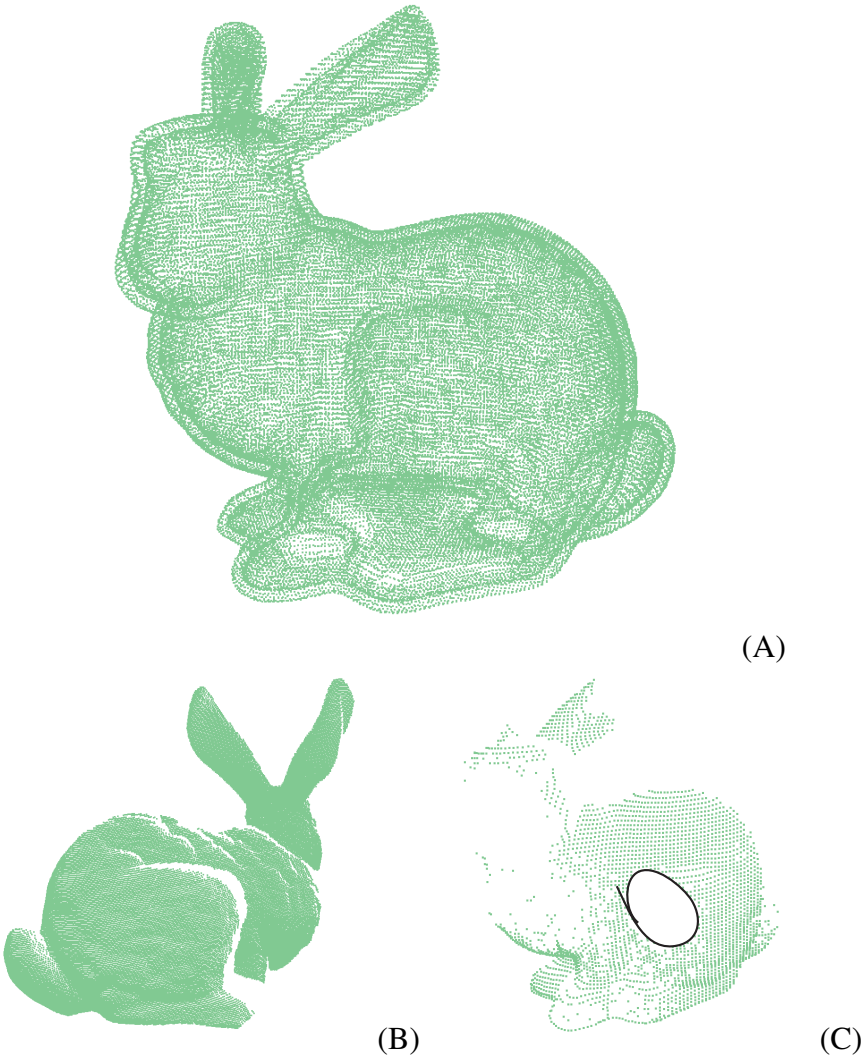


Figure 1.1: Surfaces of Standford Bunny model acquired with different devices. (A) the model is manufactured using rapid prototyping techniques with a ZPrinter 310[42]. As you can see, it prints all the layers of the model. (B) One piece of surface measured by laser scanner from one view point, covering only part of the model. Self-occlusion shows on the surface. (C) An even coarser surface measured, in middle of which a hole (marked with black) shows up due to the limited volume of the laser scanner.

In this thesis, we define point cloud, PBS $\mathcal{S} = \{(\mathbf{p}_i, \mathbf{n}_i)_{i \in I}\}$, as a set of surfels. A surfel $\mathbf{s} = (\mathbf{p}, \mathbf{n})$ on a surface is an oriented point defined by its position $\mathbf{p}$ and normal direction $\mathbf{n}$. For most data sets used in this document, we assume that the normal directions are

provided by the input data. Otherwise, it can be computed by fitting a tangent plane near the point or using the eigenvector of the smallest eigenvalue of an inertia matrix of the vertex neighborhood. For more information of surface normal estimation, please refer to [35, 56].

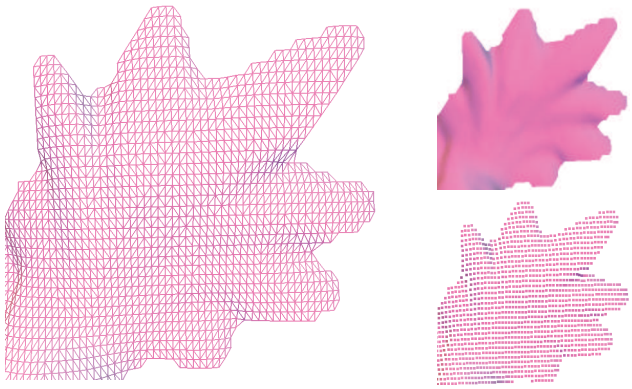
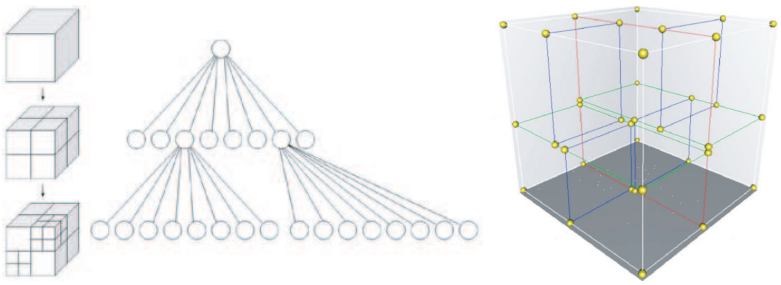


Figure 1.2: Example of mesh-based surface and point-based surface. Left shows the connectivity between polygons on mesh-based surface; Bottom right shows the point-based surface, where every point shows the position of one vertex.

On mesh-based surface, geodesic distance is often used to measure the distance between two vertices. Geodesic distance between two surface points p and q is defined as the length of shortest path between p and q on the surface. Since connection information between vertices is not available on point-based surface, we use the Euclidean distance to compute vertex-to-vertex distance and employ hierarchical data structures to enable fast spatial querying of point cloud. The algorithms presented in this thesis are implemented using KD-tree(k-dimensional tree) and Octree. An Octree is often used to recursively partition 3D space, which internal nodes split into eight child nodes. We implemented the Octree defined in [72] for spatial decomposition. KD-tree is a binary search tree that enables fast querying in high dimensional space. We use the definition of KD-tree library of [6] for both 3D spatial decomposition and nearest neighbor searching in higher dimensional spaces (the signature space is discussed in Chapter III). Figure 1.3 illustrates the structure of Octree and kd-tree that partition 3D spaces.



Octree KD-tree

Figure 1.3: Spatial data structure. Octree and 3-dimensional kd-tree. Pictures from www.wiki.com

1.2 Feature points and their signatures

With the increase of accuracy of surface acquisition techniques, point clouds contain considerable redundancy. For most applications addressed in this document, it is time-consuming and not necessary to process all points on the surfaces. To improve the efficiency, some methods [39, 23] randomly select a set of points from a given shape for further processing. In this thesis, we use the strategy that selects a set of representative points from each given shape to reduce the amount of information for processing. We call the set of selected points as features (also called feature points or salient points).

Features or feature points, are points on the surface with salient local geometry properties. Figure 1.4 shows an example of feature detection on a Chef model [54]. As you can see, prominent points such as eyes, ears, nose, and creases on the clothes are detected and marked with colored circles. The yellow circles indicate features detected from a concave area while red circles show the features from a convex region. By definition, these features reflect shape variation that is invariant to rigid transformation, since the proposed method detects features using relative position of local geometry near every vertex. Invariance to rigid transformation is an essential property of features for comparing shapes from different coordinate systems. We will discuss the robustness of these features relative to surface sampling and noise in Chapter II.

The detail about feature detection is discussed in Chapter II. We give a brief description here. To detect features on a given surface, we first compute the smooth version of its

point cloud with a Gaussian based kernel (or other kernel functions) across different kernel scales. Then the difference of successive smoothed surfaces is computed to obtain the difference-of-Gaussian (DoG) kernel filters. We detect the local extrema over DoG as features. For example, given two filter size s_1 and s_2, two smoothened surface versions and their difference-of-Gaussian (DoG) are calculated for all the points. Then, a vertex p is compared with its neighbors (within the Euclidean distance of s_1) using the DoG, and is marked as a feature point if p is a local extremum. By changing the kernel size, this method detects multi-scale features. Figure 1.4 shows the features of one scale, shown as the radius of the circles.

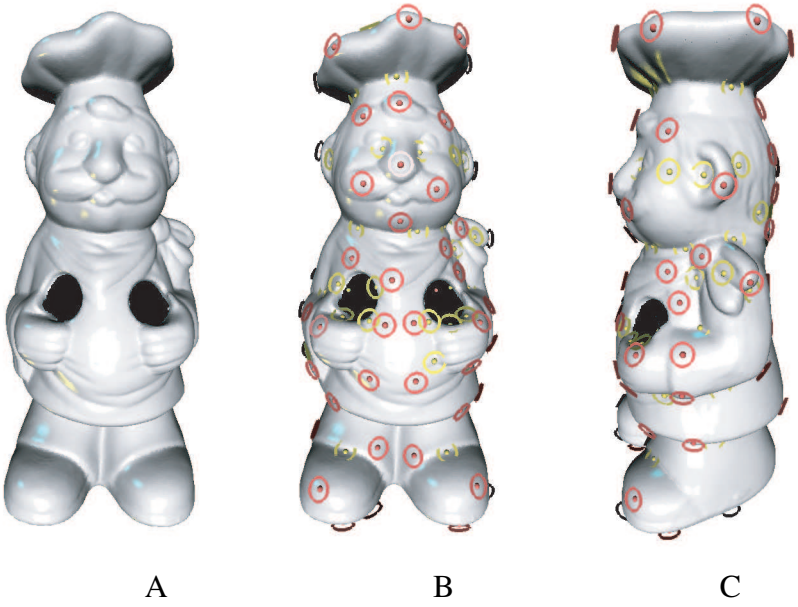


Figure 1.4: An example of feature detection. A: The Chef model; B and C: Feature points on the model are marked with colored spheres and circles. Features are local extreme, and red and yellow show the local maximal and minimal.

To use the detected features, we can assign each feature a point signature, which is a vector that models the surface variation around this point. Applications like surface matching use point signatures to compare 3D shapes [39, 86]. The signatures used in [39, 86] are the histograms of local point position around the associated points. Theoretically, the features described in this section can work with any existing point signatures.

We also proposed a new definition of 3D point signature which is inspired by the key

point descriptor of the SIFT feature for 2D images. As the descriptor of a SIFT feature is computed as a set of orientation histograms, we defined our point signature using the normal orientation of the surface. Given a surface point, a local coordinate system is defined by its position and normal direction as in Figure 1.5. T_1 and T_2 are two orthogonal axes sitting on the tangent plane. To define the signature, on the tangent plane, we sample $m * n$ points uniformly distributed in polar coordinates of the tangent space and compute their normals by averaging (with weight) the normal directions near these sampled points. Then these normals are projected onto the tangent plane and the projected magnitudes are put into a 2D $m * n$ matrix. As this matrix is defined in the local frame, it depends on the selection of T_1 and T_2. To get rid of this ambiguity, we process the $m * n$ matrix using Fourier Transform.

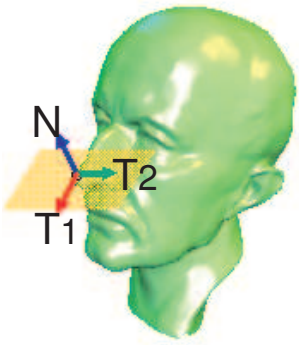


Figure 1.5: local coordinate system: N is the normal direction; T_1 and T_2 are two orthogonal axes on the tangent plane.

Unlike most of the existing signatures, the proposed point signature in this thesis is computed from the surface normals. Since the surface normal is the derivative of position, normals are more capable of capturing the small surface variation. On the other hand, it is more sensitive to noise. However, we assume that the point clouds used in this thesis have reasonable density and relative accurately measured. Additionally, since the point clouds are smoothed during feature detection and signature computation, the proposed method can tolerate noisy data. Experiments in the following chapters show that the detected features and their point signatures work well for real world data sets.

The proposed features along with their signatures are useful for surface matching/registration, multi-view 3D surfaces alignment and object recognition. The following sections will discuss these applications briefly.

1.3 Pairwise surface matching

Pairwise surface matching is the process that compares two 3D surfaces and brings overlapping surfaces into the same coordinate system. It is an important issue in computer vision and computer graphics for applications such as 3D object recognition and classification. We discuss two cases of surface matching. The first one is to match a piece of acquired data to its 3D model as in Figure 1.6. One example of this case is industrial inspection that determines if a manufactured part is consistent with a CAD model. Figure 1.7 shows one setting example of inspection for a CAD model. The second case is aligning the partial overlapping data of a model captured from two different views as in Figure 1.8. One application is multiple surface alignment, which integrates multiple partial surfaces from different views of 3D model into a complete object. In the first case, we assume that the sensed data is a subset of the model. The second case is more complex because we need extra effort to compare surfaces for the unknown overlapping part.

A B

Figure 1.6: Match a scanned surface to its model. A: original position of Stanford Bunny model (purple) and one scanned surface (yellow); B: the result after aligning scanned surface and model.

Most surface matching processes are comprised of two steps. First, an approximate

surface matching algorithm brings two surfaces into the same coordinate system roughly. In the second step, a high-precision alignment like iterative closest point (ICP) [69] is applied to refine the result of approximate matching. Figure 1.8 shows the surface matching process for two scanned surfaces of the Arrigo statue [8]. An automatic process of the first step is the key part of surface matching, which relies on establishing the correspondence between surfaces. In this thesis, we establish the correspondence of two surfaces by matching features extracted from the surfaces, and use the corresponding feature pairs to estimate the rough transformation. However, due to the diversity of acquisition methods and environments in real world, there are difficulties in matching 3D shapes:

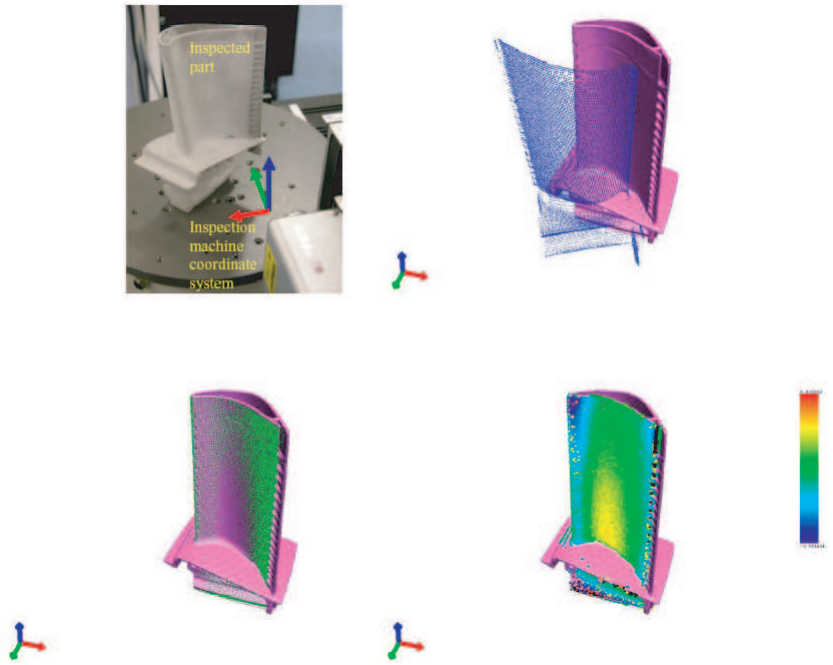


Figure 1.7: An example of industrial inspection. A: The inspected part on the inspection machine. B: the measured data in the machine coordinates and the CAD model in model coordinates. C: the CAD model and the measured data in corresponding coordinate systems after alignment. D: shape deviation from the nominal shape is calculated after alignment and visualized using a color map scale.

- First, the sampling of two surfaces is different. Given a point-based surface, we compute the distance between every point and its nearest point and define the *surface resolution* as the median distance of all points. Even for two surfaces with the same resolution, the sampling points on one surface are not exactly the same as the

points on the other surface. These factors make it impossible to establish the exact correspondence between surfaces.

- Besides dealing with the data sets with different sampling and surface resolution, another difficulty of surface matching is that the captured data is incomplete for complex shapes and big objects that are larger than the device inspection volume, as shown in Figure 1.8. Thus the topology of the surfaces can be different even though they are measurements of the same object. This factor makes the global shape descriptions [86] not suitable for surface matching. In this case, local features and signatures proposed in this thesis are of special interest. We describe the details of surface matching algorithm in Chapter III.

1.4 Multi-view surface alignment

Reconstruction of 3D models from multiple scans is an important practice in computer graphics for applications like animation and virtual museum [19, 8]. It takes scans from different viewpoints and merges them into one compete, non-redundant model. Since scans are acquired from multiple views, this task requires to find the transformations between different views to bring all the scans to the same coordinate system. Using the result of pairwise surface matching introduced in the previous section, multi-view alignment of a set of scans for one object can be automatically performed.

In this thesis, we proposed a simple yet effective algorithm for multi-view surface matching. Given K scanned surfaces, the algorithm runs $K(K-1)/2$ coarse pairwise surface matching to find possible matching between every pair of scans. For every matched surface pair, we compute a weight w that estimates their overlapping part relative to the original surfaces. The pair of surfaces with larger overlap weight has higher confidence to be a good match. Then the results of pairwise surface matching are ordered by their weight. We start merging the pairs from the top of the list and performing ICP registration to align these scans more accurately. Figure 1.9 shows the result of matching 30 scans of Arrigo

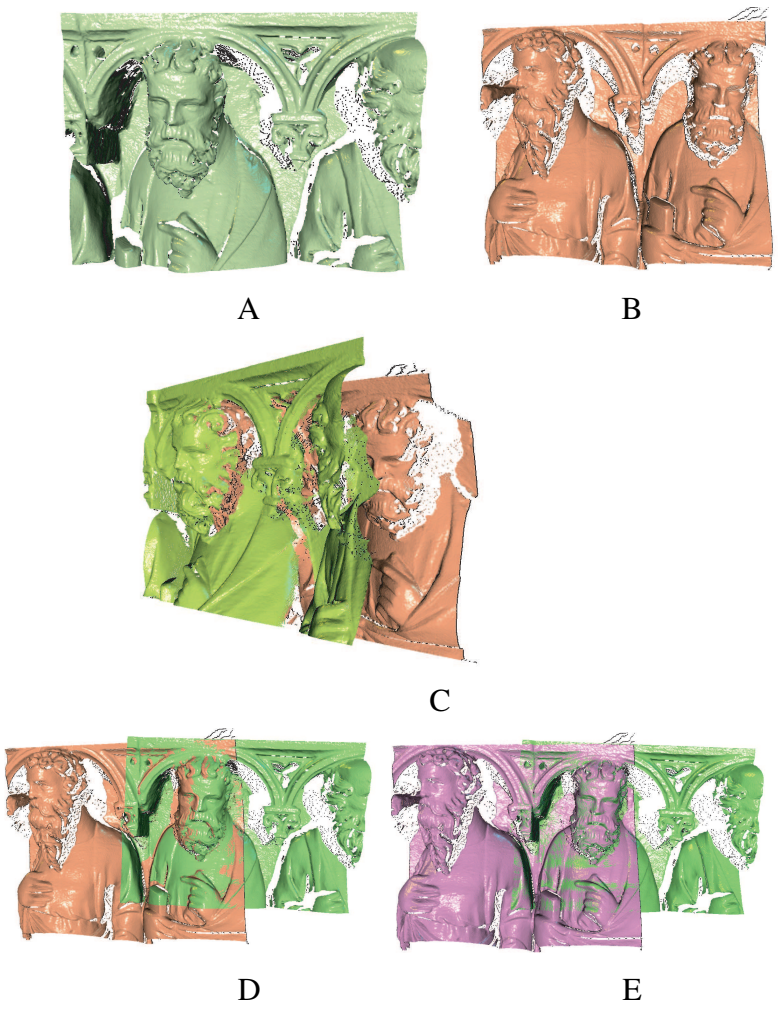


Figure 1.8: Surface matching for surfaces with partial overlapping. A and B: two scanned surfaces from Arrigo data set; C: the relative position of the surfaces before surface matching; D: result after approximate surface matching; E: refined result after applying ICP.

data set captured from 30 different views.

Figure 1.9: An example of multiview surface matching. Arrigo statue reconstructed from 30 scanned surfaces, which are marked with different color.

1.5 Object recognition and other applications

Although human beings recognize objects from 2D images and 3D scenes with little effort, object recognition is still a challenging problem in computer vision and computer graphics. Object recognition is difficult because of the change of illumination, different view points, various object size, as well as partial occlusions and background cluttering. The important issue for object recognition in 2D images is the representation of models in the database and the objects in the scene. Most recognition algorithms over 2D images and video streams use significant points and their local contexts to perform reliable object recognition.

1.5.1 Object recognition

The recognition problem discussed in this thesis is in 3D image domain. It determines whether a predefined object (referred as a model in the database) is present in a specific scene composed of 3D point clouds. Similar to the recognition problem from 2D images, the scene may contain one object or be composed of multiple objects. The illumination variation does not affect the acquisition of 3D point cloud as much as it does in the case of 2D image, but the issue of different view points, partial occlusion and background cluttering still exits, which makes the object recognition a difficult problem. Thus, the proper

presentation of the models and the scene is still a critical issue for 3D object recognition. Since the multi-scale features proposed in this thesis are defined on local geometry properties, they are good candidates to represent 3D surfaces for the recognition problem.

We address two cases of the recognition problem here. In the first scenario, every object in the database is a complete model. The point cloud of the scene has multiple objects, but only covers part of every object due to cluttering and self-occlusion. Since the surface matching method in section 1.3 can deal with surfaces with partial overlap, it provides a feasible solution to this problem. To solve the recognition problem, the scene is compared with every model in the database using the matching algorithm. A model is recognized if there is a match between the scene and this model. Figure 1.10 shows two examples of object recognition for a scene of 4 objects. As you can see, in the first example, the Chicken in the scene are obscured by other models. The small visible portion does not provide enough reliable geometry features for surface matching, and it cannot be recognized by the proposed method.

An alternative case is recognizing shapes from 3D range images of single object. For this problem, we do not have any complete model for every object in the database. Instead, every model has a set of partially overlapping 3D range images that are captured from different views. Together, the range images should cover all the areas of the model. Given a new laser scan, the problem is to decide if it belongs to some model in the database. One solution to this problem is to reconstruct a global model for every object in the database using the scans and then use the solution of the first case for recognition. A similar work was proposed by Mian et al. in [52]. In this document, we introduce a new solution using machine learning techniques.

To explore this recognition problem, we represent every model in the database by a set of scans, and decompose each scan into a set of unordered features. All sets of features are put into a space where they are partitioned into pyramid histogram based on their point signatures. The pairwise similarities between feature sets are computed using a pyramid

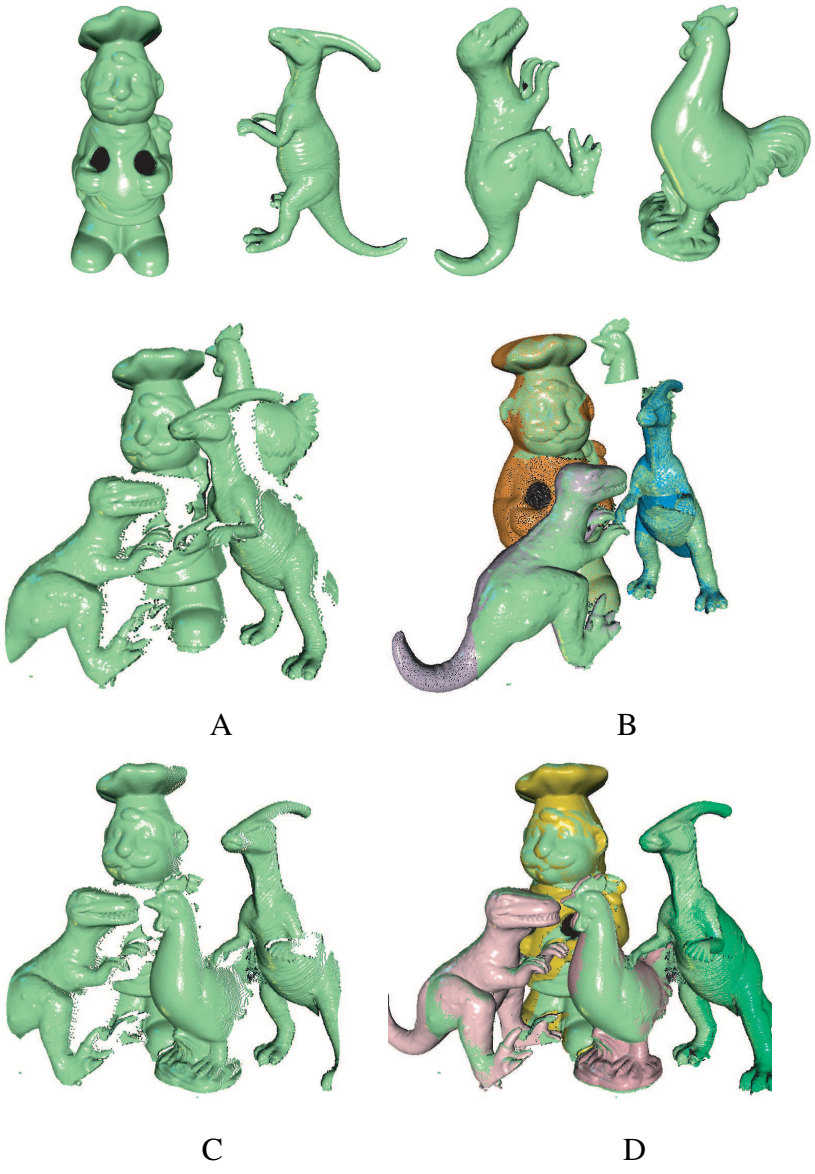


Figure 1.10: Examples of object recognition. Top: four models appeared in the scenes; A and B: one real scene and the results in which three objects are recognized; C and D: another real scene in which all objects in the scene are recognized.

kernel function. Given a set of training images of N objects, N classifiers are learned to separate the set of range images of one object from others. Object recognition means predicting the unknown range images using these classifiers learned. Details about this method will be discussed in Chapter V.

1.5.2 Shape symmetry

Another application of the proposed feature points is to detect shape symmetry. Symmetry is a common and interesting phenomenon in 3D objects. Here we introduce a simple method to detect symmetric shape for 3D point clouds, which works by finding the closest feature pairs on the signature space. Given a 3D shape, we first detect a set of features $\mathcal{F}$. For every $p_i \in \mathcal{F}$, the algorithm finds its closest neighbor $q_i \in \mathcal{F}$ by matching their signatures. The pair of features define a reflective plane D_i. The norm of D_i is aligned with the direction connecting two features $\overrightarrow{p_i q_i}$, and the position of the plane is fixed as the average of two points $(p_i + q_i)/2$. After detecting all the feature pairs, they are merged into groups by their reflective planes. Two feature pairs are gathered into the same group if the normal direction of their planes are close and the distance between their middle points along the normal direction is less than a threshold. The method takes the number of feature pairs that fall into a group as the vote for the group and rank the groups by their votes. Figure 1.10 includes some results of symmetric shape detection. The first row shows the reflective plane with most votes for one partial surface of Chef model. The second row shows the best three reflective planes from a point cloud of the Parasaurolophus model [54].

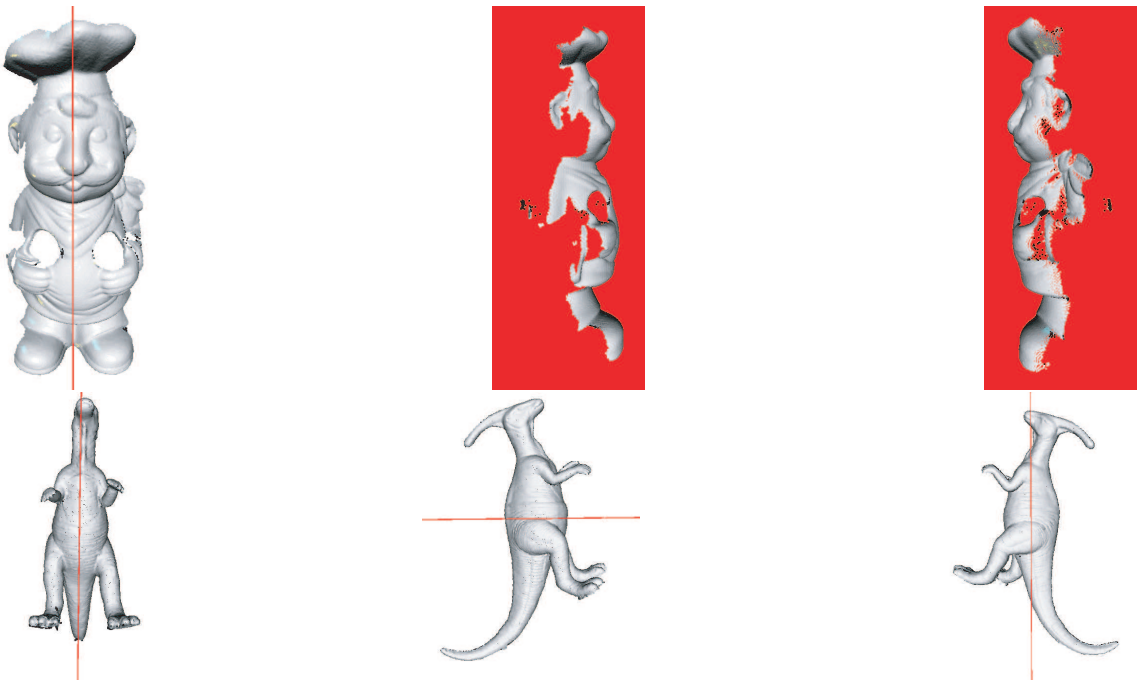


Figure 1.11: Examples of symmetric shape detection. Red lines show the position of the planes. Top: One symmetric plane detected for a Chef model and two parts partitioned by the plane. Bottom: three different planes detected for a Parasaurolophus model surface.

1.6 Overview

This thesis will be arranged like this:

In Chapter II we describe the multi-scale feature detection algorithmss. We start with a robust but slower method using a moving least square (MLS) projection process, and then discuss several variations that speed up the algorithm without losing much on performance. For the introduced feature detection algorithms, we evaluated their robustness in case of surface variation such as sampling and noise.

In Chapter III we define the viewpoint invariant signatures for selected feature points. We also explain the properties of the signatures and how to compare them in the signature space. A pairwise surface matching algorithm is proposed to test the performance of detected features and their signatures. We also introduce a method to reconstruct 3D models from multi-view scans. The experimental results on various real models show the robustness of our matching algorithm.

Chapter IV proposes a feature-based surface alignment algorithm of range scans to their CAD model. Compared with the matching process in Chapter III, it is a simpler process. However, the target surfaces in this application are generally much coarser. In this case, it performs exhaustive search of features from two surfaces without using signatures, which makes it more robust to deal with coarse and noisy datasets.

Chapter V discuss a new method for feature-based object recognition from range images using pyramid matching. We first decompose each shape using a set of features and use a pyramid kernel function to measure the similarity between a pair of surfaces. Then support vector machine (SVM) is used to learn the classifiers that separate one object from the rest. New images are compared with classifiers and labeled as the category with highest response. We evaluate and compare our results with related work, and report the results.

CHAPTER II

FEATURE POINT DETECTION

Complex geometry data are used in many design and manufacturing applications. With the advance of affordable shape acquisition technologies and the growing amount of geometric data there is a need for efficient shape matching and retrieval tools. One way to achieve such efficiency is to augment each shape with a set of its salient features. In this chapter we describe novel approaches for extracting such local feature sets.

2.1 Related works

The use of local features in 2D image processing can be traced back to the interest operator proposed by Moravec [57], which is followed by various feature detectors, e.g. Harris and Stephens[33], Schmid and Mohr[73]. Some features like edges [13, 87], ridges [32], corners [33, 80] show geometry properties of image scenes. Other low level features detectors extract interesting points that are a local maximum of an operator response or a center of gravity.

The idea of using scale-space processing for extracting salient features is well established in image matching and object recognition [84, 47, 51]. Such feature detectors look for stable interesting pixels across all possible scales using a continuous function of scale. Scale invariant features provide a good matching base for images in different size and resolution.

The ability to extract a small but representative set of keypoints on 3D surface is important for applications such as matching, modeling and recognition. To be useful the

extracted features should be repeatable under surface variations (like different surface sampling, rigid transformations) and carry salient surface information as features from 2D image[74]. Therefore, invariant characteristics of the surface such as curvature are often used for feature detection. For instance, in Yamany and Farag's work [86], only a set of high curvature points is considered for signature computation. Chen and Bhanu [17] used points with large shape index computed from principal curvature. Another approach proposed by Wyngaerd and Gool in [85] is to consider a set of bitangent points. An alternative simple method often used in practice is to uniformly distribute the basis points over the shape [39] [23].

A variety of algorithms that find interesting features and regions for various applications from mesh-based surfaces have been proposed recently. Gatzke and Grimm [26] extracted multi-scale features on polygon surfaces using a local shape function based Curvature Map and Graph Cut algorithm. Gal and Cohen-Or [24] presented a method to find salient geometric features that represent interesting parts of a given shape. Shilane and Funkhouser [79] defined distinctive region of 3D surfaces by first placing a random set of points on the surface mesh and then comparing a shape descriptor for the points at multiple scales to find distinctive surface area.

In computer graphics, multiresolution processing of point-based surfaces is an active area of research: robust modeling, rendering, and processing approaches for point and surfel clouds have been recently developed [63, 4, 65, 59]. The work of Pauly et al. [62] extracts line features using a scale-space representation similar to our method proposed here. The method of Gumhold et. al. [31] collects meaningful feature lines from point clouds. In our application, we are not concerned that our feature points represent visually important portions of the model. Rather, our interest is in detecting a stable set of feature points that can be used for surface matching. Our features are built on a multi-scale representation of point-based surface[43]. A related work by Pauly et. al.[64] proposed multi-scale surface representation based on a least squares fitting.

The methods we proposed in this chapter rely on multi-scale filtering of 3D data of the input shape. We build a scale-space representation of the input shape, and use the location of level difference extrema as the salient feature points. This process is similar to finding extrema of the scale-normalized Laplacian of Gaussian in the 2D image case [51], and is easy to implement for point-based surfaces. For detected feature points, we explore their robustness under resampling and present the improved repeatability results.

The input shapes are often represented using hundreds of thousands of points with normals. Therefore, it is very important to pay attention to the computational procedures used to extract salient feature sets. Thus, we present a flexible and simple hierarchical treecode algorithm for efficient feature point extraction that uses a fast evaluation of weighted sums on the octree. The robustness of extracted feature point sets depends on the errors introduced by the filtering operation, therefore one needs to rely on guaranteed error bounds. We present a simple error criteria that can be used to guide the adaptive octree construction and present the experimental data for the runtime analysis of our evaluation procedure. The feature extraction is also made faster with the help of the multiresolution feature extraction that uses simplified point clouds for the coarser levels of the scale-space.

Our treecode approach and its error analysis was inspired by approach of Lindsay and Krasny [48] designed for accelerated simulation of vortex sheet evolution. A similar fast evaluation technique was briefly described in [76] for their evaluation of weighted integrals. Their method works on KD-trees and uses a simple heuristic error criteria. Dey and Sun [22] analyze the reconstruction algorithm for adaptive MLS surfaces, show that the contribution of distant point samples is bounded, and discard the faraway samples in their surface projection algorithm.

In 2D image domain, good features provide reliable matching basis for many applications with regard to affine distortion, change in 3D viewpoint and illumination. Schmid et. al [74] introduced two criteria to evaluate different feature detectors: repeatability rate and information content. In this chapter we discuss the repeatability of our feature detector.

Next chapter, we define a new point signature, which encodes the local content for detected features.

This chapter is organized as following: We first describe the scale-space surface representation using a surface smoothing process by point projection in Section 2.2. Section 2.2.1 presents a scale-space representation for the 3D surface and our feature detection procedure is described in Section 2.2.3. Given the detected feature sets, we examine their repeatability under surface resampling and noise introduction on real datasets in Section 2.2.4. Then we proceed to a new method by smoothing surface normals with faster processing speed in Section 2.3. Finally, we present a fast treecode approach to extract features and study its error analysis in Section 2.4.

2.2 Multi-scale feature detection based on projection

In this section, we describe the feature detection procedure for 3D surfaces, which is originally introduced in our previous work [43]. This method is developed for point cloud, and can be easily extended for mesh-based surfaces.

2.2.1 Scale-space representation by Moving Least Square projection

The scale-space representation is built using on a smoothing procedure, which is based on the point-set surface projection, as described by Amenta and Kil in [5]. Given a surfel cloud $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{n}_i)\}_{i\in\mathcal{I}}$, we define the projection operator $Proj_{[\mathcal{P},h]}$ that can map one 3D point $\mathbf{x}$ onto the smoothed version of the shape defined by the surfel cloud $\mathcal{P}$ and the scale parameter $h > 0$. The projection is found by iterative minimization of the following moving least squares (MLS) error:

$$E_{[\mathcal{P},h]}(\mathbf{x}) := \sum_{i\in\mathcal{I}} \theta_i^h(\mathbf{x}) \langle \mathbf{n}_{[\mathcal{P},h]}(\mathbf{x}), \mathbf{x} - \mathbf{p}_i \rangle^2, \tag{2.1}$$

where $\mathbf{n}_{[\mathcal{P},h]}(\mathbf{x})$ is the smoothed normal defined based on the input cloud normals:

$$\mathbf{n}_{[\mathcal{P},h]}(\mathbf{x}) := \sum_{i\in\mathcal{I}} \theta_i^h(\mathbf{x})\mathbf{n}_i \tag{2.2}$$

and the weights θ^h are normalized Gaussian weights

$$\theta_i^h(\mathbf{x}) := \frac{A_i e^{-d^2(\mathbf{x},\mathbf{p}_i)/h^2}}{\sum_{j\in\mathcal{I}} A_j e^{-d^2(\mathbf{x},\mathbf{p}_j)/h^2}}. \tag{2.3}$$

Here, we introduce the *area weight* A_i for each point: in the case where mesh connectivity information is available, we assign A_i to be the area associated to the vertex (one third of the area of adjacent faces). If no information on the vertex areas is given, we assign $A_i = 1$ for all the points. Introduction of the area weight is important for stable computation of the scale-space for the surfaces with non-uniform point distributions (such as the ones produced by an error-based mesh simplification in [25]), this will be discussed in Section 2.2.4.

Given a particular scale h, we define the smoothed version of the surfel cloud $\mathcal{P}$ by mapping each point $\mathbf{x}_i^h := Proj_{[\mathcal{P},h]}(\mathbf{x}_i)$, and computing its normal as $\mathbf{x}_i^h := \mathbf{n}_{[\mathcal{P},h]}(\mathbf{x}_i)$. We consider a sequence of scales $h_j = h_0 F^j$ for some $F > 1$, where $F = 2^{1/4}$ in this thesis. The smoothed version of $\mathcal{P}$ at the scale h_j is denoted as $\mathcal{P}_j := \left\{(\mathbf{p}_i^{h_j}, \mathbf{n}_i^{h_j})\right\}_{i\in\mathcal{I}}$.

For the efficient computation of $Proj_{[P,h]}$, we perform spatial queries on the surfel cloud data using the KD-tree library of [6] and Octree. We define $b(\mathbf{q}, R) \subset \mathcal{I}$ to be the set of indices such that the corresponding points lie within distance R from the point $\mathbf{q}$. Formally, $b(\mathbf{q}, R) := \{k \in \mathcal{I} : d(\mathbf{p}_k, \mathbf{q}) < R\}$. This query is always run within the original point cloud. The found index set can then be used to access the smoothed versions of the point cloud.

2.2.2 A simpler method for scale-space representation

The smoothing method introduced in 2.2.1 gives an optimized points position for surface smoothing. However the non-linear optimization may cause a slow smoothing procedure. To speed up this process, we can use a simper form as in [88] and [22]

Instead of minimizing the MLS error in Equation 2.1 interactively, the speedy algorithm update the point position on scale i as $\mathbf{x} \leftarrow \mathbf{x} - E_{[\mathcal{P},h]}(\mathbf{x})\mathbf{n}_{[\mathcal{P},h]}(\mathbf{x})$ given a soomthing radius at scale i. Surface normals are computed accordingly using Equation 2.2. By eliminating the non-linear optimization process, this method speeds up the smoothing process for space representation. Row A and row B in Figure 2.7 shows the multi-scale space representation of a sphere with 3 bumps. The difference between two representations with same scale is subtle: the non-linear optimization projection procedure makes the sphere shrink a little more.

2.2.3 Normal difference and feature extraction

We can now describe the salient feature detection procedure. For further processing, we define the normal difference between two adjacent levels of our multi-scale representation: $d_j(i) := \langle \mathbf{n}_i^j, \mathbf{p}_i^j - \mathbf{p}_i^{j-1} \rangle$, for $j \geq 1$ and $i \in \mathcal{I}$. We call a point i on level j to be a *neighborhood maximum* of d, if

$$
\begin{aligned}
&d_j(i) > d_{j-1}(i') \quad \text{for all } i' \in b(p_i, Ch_{j-1}),\\
&d_j(i) > d_j(i') \quad \text{for all } i' \in b(p_i, Ch_j) \setminus \{i\},\\
&d_j(i) > d_{j+1}(i') \quad \text{for all } i' \in b(p_i, Ch_{j+1}).
\end{aligned}
$$

The *neighborhood minimum* of d is defined analogously.

In other words, we compare the value of the normal difference at a point to its neighbors within the level as well as to its neighbors on the two neighboring levels of the scale-space hierarchy. If the difference is larger (smaller) than all of its neighbors within thus constructed neighborhood then the neighborhood maximum (minimum) is detected. The size of the neighborhood grows as we proceed to the coarser scales.

We use the set of neighborhood minima and maxima for a given shape as its salient feature set. They are corresponding to the concave and protruding parts on the surfaces.

The intuition behind using the extrema of differences between consecutive levels of

the scale-space is that the smoothing with a filter of effective radius R will not have much effect on features whose scale is larger than R but will eliminate most of the features at the scales smaller than R. As we consider differences between two versions of the surface smoothed at two scales R_1 and R_2, the most change will happen in the regions that contain features of scales between R_1 and R_2. For more information, see [47].

In Figure 2.7 we show a simple example of a scale-space for a sphere with three differently sized bumps. The surface at each level is colored green or blue depending on the sign of the normal difference between the current and the previous level. The intensity of the color corresponds to the absolute value of change with the brighter parts related to larger surface variations. The circles denote detected extrema cross the scale space. We see that three detected features correspond to the surface bumps. Each feature gets a scale associated with it that is represented by the radius of the circle in Figure 2.7

For most applications addressed in this thesis, the scale-space construction starts with a scale that is a multiple of the median distance between points and their closest neighbors, which is defined as the *surface resolution* in Section 1.3. In most cases, we round the surface resolution r to the smallest number that is a power of two and larger than the median distance. Generally, we processing the surfaces 8-10 level of scales. For real data sets, some point clouds are measured in different units. For example, some surfaces are measured in inch and others are measured in millimeter. However, the scales are parameters that can be set by the users. To deal with point clouds that are captured using same unit at different surface resolution, the smallest scales can be set to be the same for all the point clouds to extract features at the same scale level for comparison. On the other hand, if the models are captured using different units, the smallest scales can be related to be the size of models.

2.2.4 Repeatability of features

Given two finely sampled surfel clouds representing the same surface, we can hope that their scale-space representations will be similar and will result in a similar set of feature

points on scales that are sufficiently large in comparison to the sampling density. We have explored the validity of this assumption by applying several types of "attacks" to input point clouds, and comparing the resulting sets of feature positions.

In order to evaluate the repeatability of our salient feature point set extraction procedure we applied the following transformations to a set of 3D models:

- Uniform resampling: for this purpose, a set of mesh-based data set are selected and subdivided using the method proposed by Loop in [49]. Then we randomly remove a set of points from the refined surfaces until a desired number of points is reached. As shown in the middle of Figure 2.1 for an example, this process makes the points left almost uniformly distributed on the surface.

- Noise addition: to test the robust of features with regards to surface noise, we add white noise on every point's position along it normal direction. We use the mesh-based surface and compute the noise surface normal after adding noise on the positions. For example, the last image in Figure 2.1 shows a Venus model with noise. The magnitude of noise is 0.5% of the bounding box, which is the minimum 3D cube that enclosesthe model. For Bunny and Buddha models [1], each component of the normal vectors is modified by noise uniformly distributed between -0.05 and 0.05.

- Mesh simplification: the mesh-based surfaces are refined using Loop subdivision and then simplified with the quadric algorithm [25] to reduce the number of points to a desired number.

In our experiments, we extract the set of features before and after a particular modification and compare the results as follows: for a feature f with position $\mathbf{x}$ and scale h in one model we see if there exist a feature f' in the other feature set that is on the same or a neighboring level in the scale-space representation (that means that the scale h' of f' is between $2^{-1/4}h$ and $2^{1/4}h$), and whose position x' is within distance $h/2$ from $\mathbf{x}$. We also consider the set of minima and maxima features separately. If such a similar feature exists,

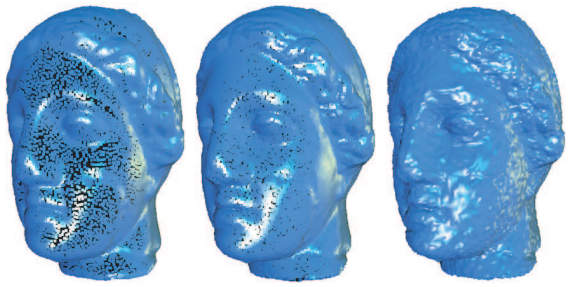


Figure 2.1: Examples of surface variations. Original, resampled, and noisy datasets for the evaluation of feature point detection stability.

we say that the feature f found a correspondence. We then count the number of features in both models that have correspondence and compare it to the overall number of features in both models. The result is the repeatability rate. In the ideal case, we would like to have 100% of all the features find their correspondences after the transformation.

Repeatability of features by projection

Figure 2.2 shows the results of comparing feature sets of original and uniformly resampled models for the venus head, happy buddha, and Stanford bunny datasets. We see that on coarser levels there is a good correspondence between the original and resampled feature sets. Note that the original venus and buddha models we used are already the result of quadrics simplification procedure, hence the distribution of samples in them is rather non-uniform. In this experiment, we did not use any vertex area information so that all area weights were set to 1 (see below for the area-corrected comparisons). Also, the number of vertices in the original and resampled models were roughly the same.

Figure 2.2 shows the comparison between the original and noisy models for the same three datasets. We see that again the percentage of corresponding features rises from about 50% for fine scales to almost 100% for the coarser scales.

Since many scans come with some mesh connectivity information, it is possible to use that mesh information to compute area weights for every vertex. Using these weights is especially important if any error-based mesh simplification algorithm was applied to the model. The error-based simplification typically results in very non-uniform surface

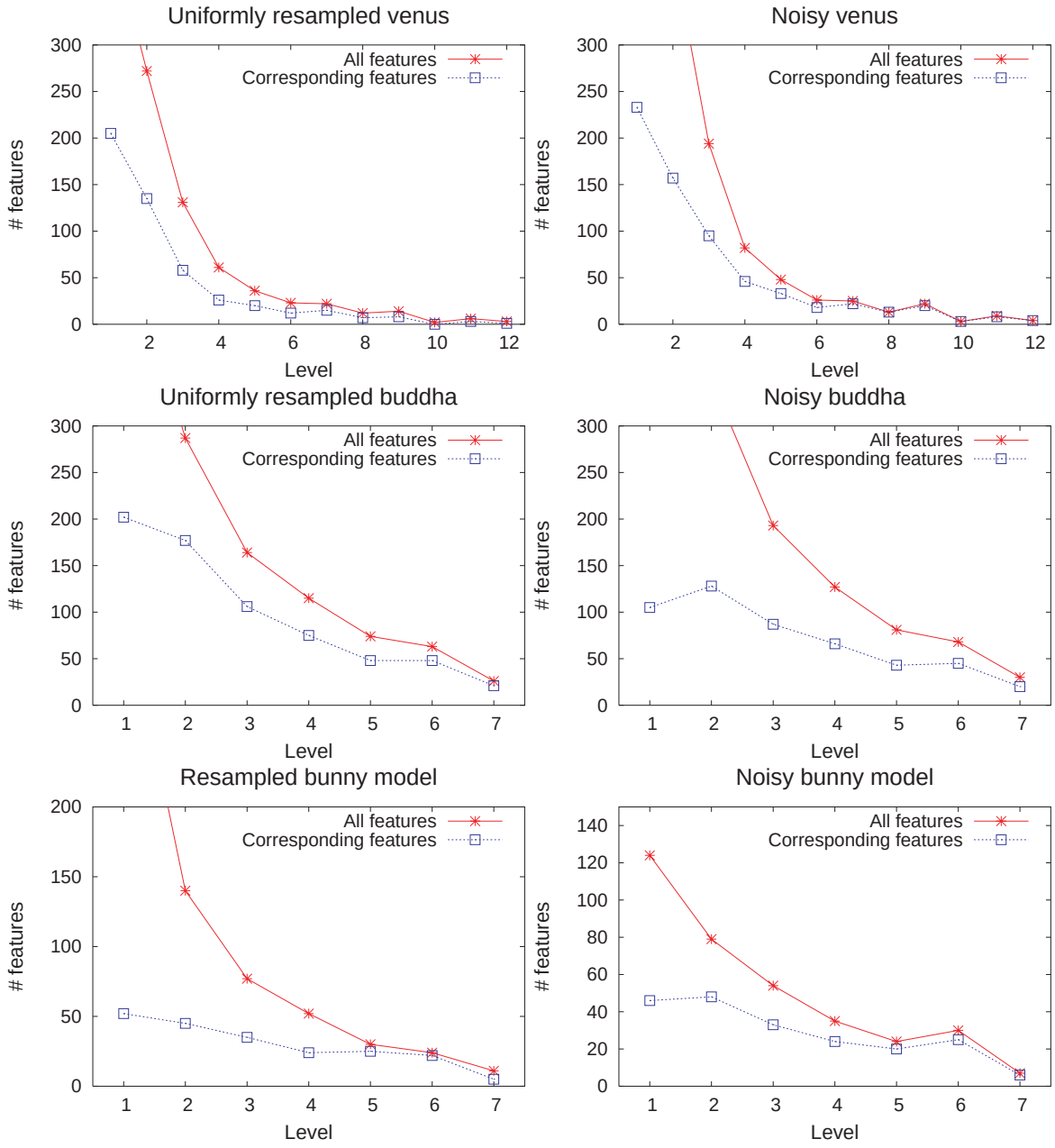


Figure 2.2: Feature set repeatability for the uniform resampling and noise addition transformation applied to Venus, Buddha, and Bunny models. See text for more details.

sampling which negatively affects the repeatability of extracted feature sets. Using area weights however improves the situation significantly, as shown in Figure 2.3 where the original Venus mesh with 50K vertices and quadrics-simplified mesh with 30K vertices are compared first without and then with area weights. The plot on the right shows the percentage of corresponding features. We see that with area correction the percentage of corresponding features improves and gets to almost 100% for coarse scales. See also the sets of extracted features in Figure 2.4 where the corresponding features are shown as yellow circles and non-corresponding features are shown in magenta (note that one has to look not only on the same but also on neighboring scale for finding correspondences).

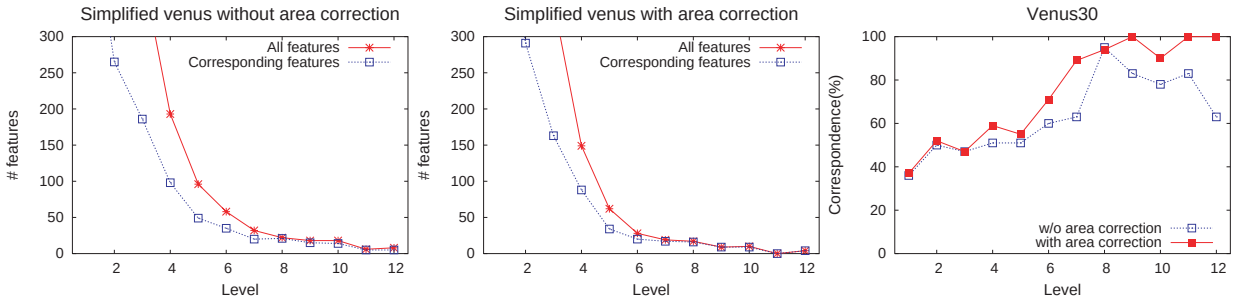


Figure 2.3: Feature set repeatability for the quadric simplification applied to the Venus model. The performance for the area corrected version is evaluated.

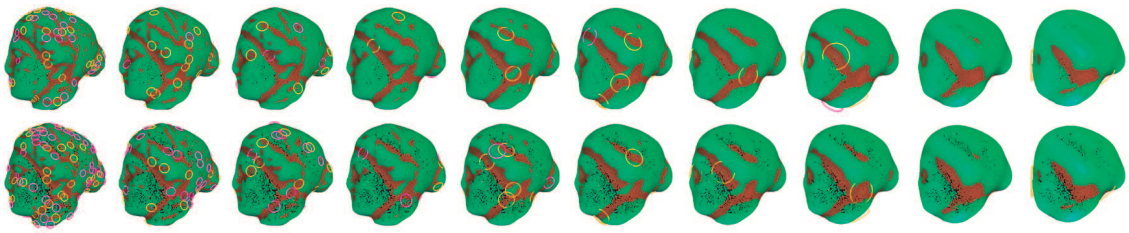


Figure 2.4: Comparison of scale-space features from two differently sampled surfel clouds representing the Venus head. Top: original, bottom: quadrics-simplified to 30K vertices. The features detected at various levels are shown with their normals and scale indicated by the surrounding circle radius. Note that we would typically look not a single level but also at its neighboring levels to find common features (shown by yellow circles). These two models use the area correction. This illustrates the central plot of Figure 2.3.

In our experience, the coarse features hold most significant semantic shape information (see for example the nose and eyes features in Figure 2.4). However, for matching applications, these are not the most useful features since there are not so many of them. Rather, intermediate scale features (of which there are hundreds) play more significant role.

We have also evaluated the area-corrected smoothing in the case when only one model has the connectivity information while the other uses unit area weights. This works well when the distribution of the non-weighted model samples is uniform. Figure 2.5 shows the comparison of area corrected and non-area corrected feature extraction for the resampled happy buddha model.

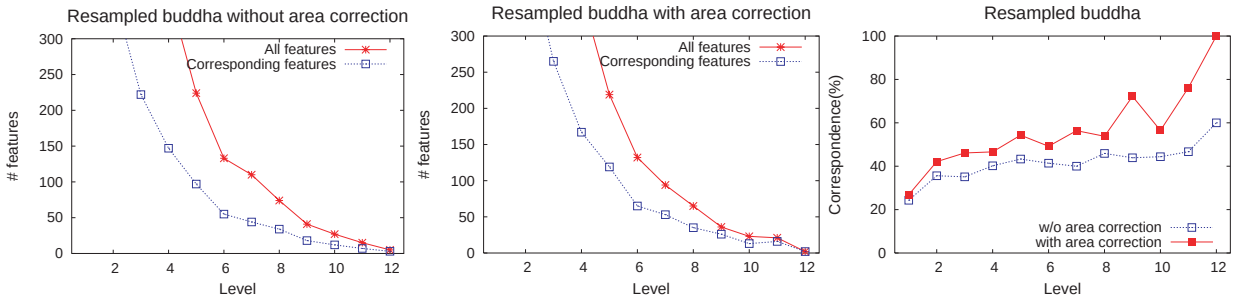


Figure 2.5: Feature set repeatability for the resampled Buddha model. The performance for the area corrected version is evaluated. Only one of the models have the area information.

2.3 Multi-scale features by smoothing surface normals

In Section 2.2, we use MLS projection to detect salient points, which operates on both surface position and surface normals. As shown in Figure 2.7, the projection operator smooth the surfaces by 'flatting' the bumps on the sphere. However, with the increasing of the scales, the projection may cause irregular shape, as shown in Figure 2.6.

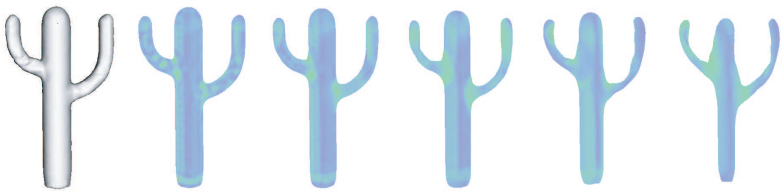


Figure 2.6: An example of scale space representation by surface projection. An cactus model (left) and it multi-scale representation with increasing scales.

Here we introduce the method of surface smoothing by normal filtering. Unlike the projection process proposed in previous section, this method smooth surface normal and leave the position unchanged. The last row of Figure 2.7 gives an example of scale space

representation by smoothing surface normal. For a point-based surface $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{n}_i)_{i \in I}\}$, we build a scale space for the normal data by applying a simple filter of varying scale $h > 0$. The smoothed version of the normal field is defined as:

$$\mathbf{n}_i^h := \frac{1}{W_i^h} \mathcal{N}_i^h, \tag{2.4}$$

where

$$\mathcal{N}_i^h := \sum_k w\left(\frac{|\mathbf{x}_k - \mathbf{x}_i|}{h}\right) \mathbf{n}_k. \tag{2.5}$$

and $w(r)$ is a fast decaying kernel function. One example is the Gaussian kernel function using in Equation 2.3. If we treated the normal as plain data on 3D surface, we use the normalization term W_i^h

$$W_i^h := \sum_k w\left(\frac{|\mathbf{x}_k - \mathbf{x}_i|}{h}\right). \tag{2.6}$$

Another option is to normalize the signal as:

$$\bar{\mathbf{n}}_i^h := \frac{1}{\left|\mathcal{N}_i^h\right|} \mathcal{N}_i^h, \tag{2.7}$$

Then we can define the feature points based on normal smoothing similar as in Section 2.2. Given an input surface, intuitively, the input data is filtered with a sequence of filters of exponentially increasing radius, and the difference between consecutive filtering levels are computed. The local extrema of these difference are then used to define salient feature points.

We use the following sequence of scales $h_j = 2^{j/4} h_0$, where the starting scale h_0 can be set by the user or derived from the parameters of the input data set. In our experiments, we set h_0 to be multiple of the surface resolution: median distance between closest neighbors in a point set. If there is a need to compare several scans measured in same unit (like the surface alignment in Chapter III), a common set of scales is preferable, and the parameter h_0 is set to the same value for all the scans.

There exist several options for converting a vector-valued difference between adjacent levels of normal data to a scalar value necessary for finding extrema. We experimentally

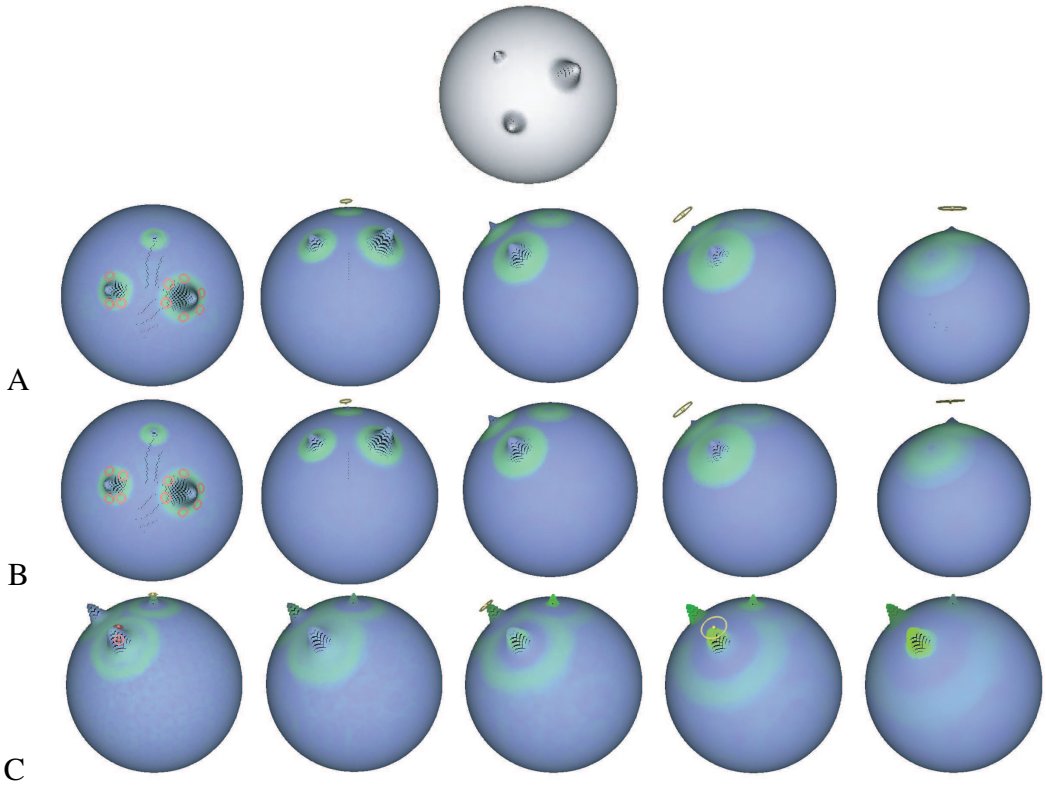


Figure 2.7: Simple examples of multi-scale features by projection A sphere with three bumps (top) and the features corresponding to bumps marked with yellow circles. A: surface representation with projected smoothing procedure. B: surface representation with simpler smoothing procedure. C: surface representation with normal smoothing procedure.

evaluated three methods for computing such a difference, with the feature repeatability results reported in Figure 2.8. Our experiments compare the feature points sets of two differently sampled surface models and are described in detail in the next section. The three compared methods of computing the scalar difference of normals are as follows:

(A) The projection of the averaged normal difference on the normal direction:

$$d_j(i) := \langle \mathbf{n}_i^{h_j}, (\mathbf{n}_i^{h_j} - \mathbf{n}_i^{h_{j-1}}) \rangle \tag{2.8}$$

(B) The length of the averaged normal difference:

$$d_j(i) := \left| \mathbf{n}_i^{h_j} - \mathbf{n}_i^{h_{j-1}} \right| \tag{2.9}$$

(C) The length of the unit normal difference:

$$d_j(i) := \left| \mathbf{n}_i^{h_j} - \mathbf{n}_i^{h_{j-1}} \right| \tag{2.10}$$

Among these three methods, only the first one produces signed values, while the last two always result in a positive value of the difference.

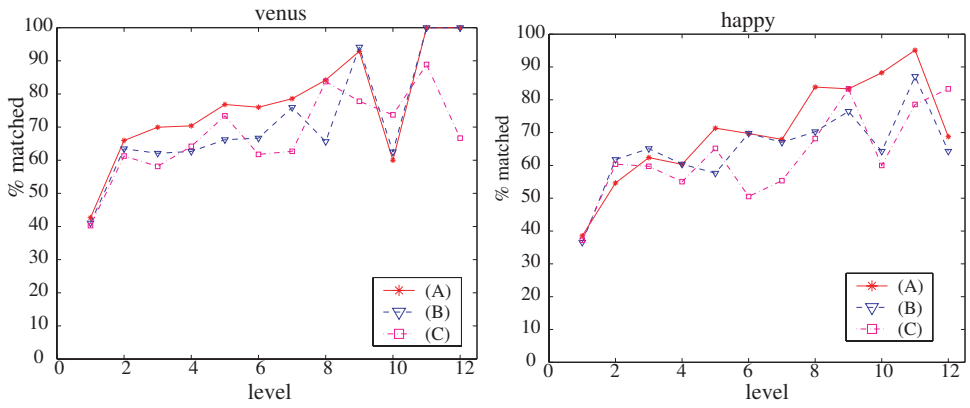


Figure 2.8: Repeatability of features of three choice of scalar difference for two models: the Venus model and Happy Buddha models. Two differently sampled versions of the same model are used for each repeatability experiment.

Once the scalar differences between levels are found, we perform the extraction of its local extrema: for signed values of approach (A) above we extract both maxima and

minima, and for the positive values of the other two approaches with extract the maxima. Our definition of local extrema is the same as described in Section 2.2.3: we are looking for points whose difference value on level j is a local extrema among the points within a ball of the radius equal to $h_j/2$ on the same level and within the similarly defined balls on the previous and the next levels. This definition ensures that the two features at the scale j cannot be closer to each other than $h_j/2$. Based on our repeatability experiments described in the Section2.3.1, we chose the method (A) and use it throughout the paper. Column (C) of Figure 2.7 shows the example of differences using (A) for a surface of sphere with bumps.

2.3.1 Repeatability of normal-based features

In this section, we compare the repeatability of multi-scale features extracted by position projection described in Section2.2.1and by normal filtering described in Section2.3. Figure 2.9 shows this comparison on two models with similar surface variation as in previous section. The same in previous section, the best results were reported when each vertex has an associated area value. We have also augmented our normal filtering procedure with area-weighted values and performed the comparisons. As can be seen in Figure 2.9, the normal-based features are comparable with the position-based features. Thus, the normal-based features can be considered as a good candidate for the salient feature set.

2.4 Treecode evaluation approach

Equations (2.5) and (2.6) require evaluation of sums whose terms are weighted by a fast decreasing function of the distance between the evaluation point and a point of the input surfel cloud. The treecode approach lumps together the summation terms corresponding to surfels lying within a single cell of an octree. The approximate evaluation proceeds by replacing the exact weighting functions by its approximation around the center of the cell.

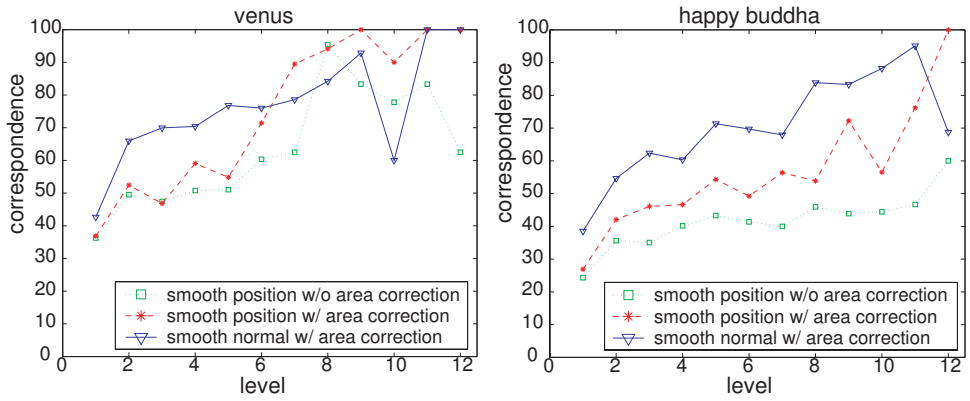


Figure 2.9: The comparison of repeatability of the area-corrected normal-based features introduced in this chapter with the results of the position-based features from [43]. The vertical axis shows the percentage of matched features. Note that the last four levels contain very few features.

We first consider the general form of summation similar to (2.5):

$$A(\mathbf{y}) := \sum_k w(|\mathbf{y} - \mathbf{x}_k| / h) a_k, \tag{2.11}$$

where $w(r)$ is a fast decaying positive differentiable function defined for $r \geq 0$. Consider a spatial decomposition that splits the bounding box B of the point set into a number of non-overlapping cells ω_s so that $B = \cup_s \omega_s$. We denote the set of indices of all the points within a cell ω_s as I_s, that is, $I_s := \{i | \mathbf{x}_i \in \omega_s\}$.

A simple treecode formulation can be obtained by using the Lagrange theorem for the weighting function,

$$w(r) = w(r_c) + w'(\bar{r})(r - r_c), \tag{2.12}$$

where $\bar{r}$ lies between r and r_c. For the error analysis below, we will need the following inequality:

$$|w(|\mathbf{y} - \mathbf{x}_k|/h) - w(|\mathbf{y} - \mathbf{x}_s^c|/h)| \leq \frac{\delta_s}{h} |w'(dist(\mathbf{y}, \omega_s)/h)| \tag{2.13}$$

under the assumption that $\mathbf{x}_k \in \omega_s$ and that the derivative $w'(r)$ is an increasing negative function. Also, we denote by δ_s the maximum distance from a cell's center $\mathbf{x}_s^c$ to any point in the cell ω_s: $\delta_s = \max_{k \in I_s} |\mathbf{x}_k - \mathbf{x}_s^c|$. See Figure 2.10 for illustration.

The summation within a cell ω_s can then be represented as

$$\sum_{k \in I_s} w(|\mathbf{y} - \mathbf{x}_k| / h) a_k = w(|\mathbf{y} - \mathbf{x}_s^c| / h) \sum_{k \in I_s} a_k + R_s(\mathbf{y}), \tag{2.14}$$

where the absolute value of the residual $R_s(\mathbf{y})$ can be bounded from above as follows

$$|R_s(\mathbf{y})| \leq \sum_{k \in I_s} |w'(dist(\mathbf{y}, \omega_s)/h)| \frac{\delta_s}{h}, \tag{2.15}$$

under assumption that $|a_k| < 1$.

We shall use the following approximation for the value of $A(\mathbf{y})$:

$$\tilde{A}(\mathbf{y}) := \sum_s w(|\mathbf{y} - \mathbf{x}_s^c| / h) A_s \tag{2.16}$$

where $A^s := \sum_{k \in I_s} a_k$ can be precomputed for every cell ω_s, and does not depend on $\mathbf{y}$.

2.4.1 Error analysis

The difference between the values of $A(\mathbf{y})$ and $\tilde{A}(\mathbf{y})$ can be expressed as $R(\mathbf{y}) = \sum_s R_s(\mathbf{y})$, and bounded as

$$|R(\mathbf{y})| \leq \sum_s |w'(dist(\mathbf{y}, \omega_s))| \frac{\delta_s}{h} K_s, \tag{2.17}$$

where K_s is the number of surfels within ω_s.

Suppose that $f(r)$ is a function such that

$$\int_{\mathbf{R}^3} f(|\mathbf{r}|) d\mathbf{r} = C_f < \infty \tag{2.18}$$

The lower sum of the above integral over the spatial partition $\{\omega_s\}$ is going to be bounded by the same constant. In order to get a formulation invariant to rescaling of the underlying model we will actually use a scaled version of (2.18), namely

$$\frac{1}{h^3} \int_{\mathbf{R}^3} f(|\mathbf{r}| / h) d\mathbf{r} = C_f < \infty \tag{2.19}$$

Proposition: Suppose that for our spatial partition $\{\omega_s\}$, the following inequality holds for every cell:

$$K_s \left| w'(dist(\mathbf{y}, \omega_s)/h) \right| \leq \epsilon \frac{\delta_s^2}{h^2} \min_{\mathbf{z} \in \omega_s} f(\left| \mathbf{z} - \mathbf{y} \right| /h), \tag{2.20}$$

then the residual $R(\mathbf{y})$ is bounded by $C_f \epsilon$.

The validity of the above proposition can be seen by substitution of (2.20) into the sum of (2.17).

We can now specialize this proposition for the case of our kernel $w(r) = 1/(1 + r^d)$, and for the special type of function $f(r) = 1/r^{3+\Delta}$ where $\Delta > 0$ ensures that (2.18) holds. We are only concerned with the behavior for the large values of r away from zero.

Then taking the derivative of $w(r)$ we obtain $w'(r) \sim r^{-(d+1)}$, and the resulting condition on the cell to achieve the upper bound on residual is:

$$K_s \leq \epsilon \left(\frac{\delta_s}{h} \right)^2 \left(\frac{dist(\mathbf{y}, \omega_s)}{h} \right)^{d-2-\Delta} \tag{2.21}$$

for instance, when $d = 6$ and $\Delta = 1$ we obtain

$$K_s \leq \epsilon \left(\frac{\delta_s}{h} \right)^2 \left(\frac{dist(\mathbf{y}, \omega_s)}{h} \right)^{3} \tag{2.22}$$

This condition does not take into account the fact that the input points lie on a smooth surface. Thus, it will tend to be conservative for large cells. We have also tried another error criteria that does not look inside any cells whose distance to the center of summation is less than a scaled radius of the filter h. This would be similar to the approach considered in [22], except that the faraway data are not thrown away but lumped together and replaced by averages over the cells of a spatial subdivision. We will specify this condition more precisely in the following section, and also present our experiments with the cell refinement approaches.

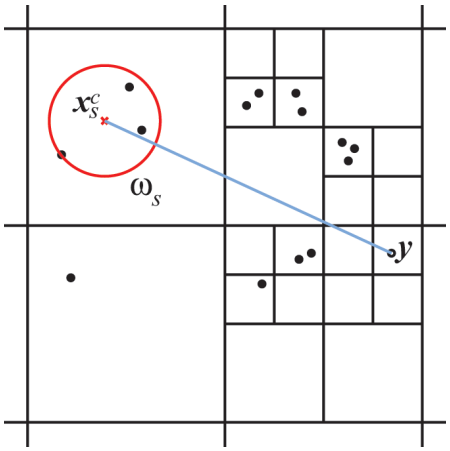


Figure 2.10: Treecode notation. $\mathbf{y}$ denotes the integration center, and the cluster of points in the cell ω_s is enclosed by a bounding sphere with center at $\mathbf{x}_s^c$.

2.4.2 Treecode algorithm implementation

This section describes our implementation of the fast evaluation of normal filtering using octree as the underlying spatial data structure. We separately describe two parts of the algorithm: the initialization stage and the actual computation of filtered data. The initialization stage constructs the octree for the input point cloud and precomputes all the quantities that do not depend on the center of weighted averaging. Then the actual filtering traverses the octree constructed during the initialization and checks an error condition (such as the one defined in (2.21)), and decides whether to continue traversal to the children of the current node or to accumulate the approximation stored in the current node.

Octree initialization

We use the octree data structure [72] as the spatial decomposition that can be adapted for the fast approximate weighted sum computation. The octree is initialized as an empty root cell, and the surfels of the input point cloud are inserted one-by-one and propagated to the appropriate leaf node. We use the octree refinement strategy that stores points at the leaves of the octree and allows no more than some fixed number of points for each leaf node. Thus, if a newly arrived point makes the cell overflow, we subdivide the cell into eight smaller cells and redistribute the points to these new children cells.

After all the points are inserted in the octree, we initialize the following quantities that are stored with each cell s of the octree:

1. the bounding box of the points within the node;
2. the partial sum of normals for the surfels within the cell $\mathcal{N}_s = \sum_{k \in I_s} \mathbf{n}_k$;
3. the number of points K_s within the cell;
4. the average position for the points within the node $\mathbf{x}_s^c = (\sum_{k \in I_s} \mathbf{x}_k)/K_s$ (this serves the role of the *node center*);
5. the radius of the node r_s which serves as an approximation to the radius δ_s of the cell.

All of these values are calculated in the bottom-up manner. Given an occupied leaf node, its properties are computed directly by enumerating the contained surfels. For a non-leaf node, the values are computed from the properties of its children cells. The first four quantities above are computed in a trivial manner. The radius r_s of the node is radius of the sphere with the center at $\mathbf{x}_s^c$ which bounds the similarly defined bounding spheres for its children.

Normals filtering using octree

The computation of scale-space for normal data requires the evaluation of weighted sums of equation (2.4) for every point of the original data. In order to avoid the quadratic run-time, we use the octree data structure described in the previous section together with its precomputed data and compute the averaged normal using the approximate formula (2.16) for both the normal data sum $\mathcal{N}_i^h$ and the sum of weights W_i^h for a point $\mathbf{y} = \mathbf{x}_i$ and a scale $h > 0$. We shall describe the specifics of our approach for the case of the normal data weighted sum $\mathcal{N}_i^h$, and the operation for the sum of weights is similar (but simpler).

Our goal is to compute the following sum:

$$\mathcal{N}_i^h = \sum_s w(|\mathbf{y} - \mathbf{x}_s^c| / h) \sum_{k \in I_s} \mathbf{n}_k, \tag{2.23}$$

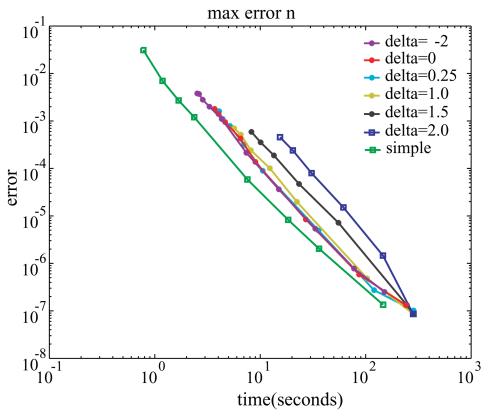


Figure 2.11: Log-log plot for the maximum error of weighted normal averaging computation for various setting of the parameter Δ and for the simple heuristic. Each curve is built by varying the user-specified parameter ϵ and recording the computation time and the maximal error for the Venus head model with 42K surfels for the filter radius $h = 0.03125$. Lower curves are better.

and the only unknown in this equation is a specific spatial decomposition that determines which cells s are included in the above summation. The recursive procedure described below obtains a particular simplified version of the constructed octree, and produces the set of cells to be used in the summation. Note that the values of $\mathcal{N}_s = \sum_{k \in I_s} \mathbf{n}_k$ are already precomputed and stored with the tree. Also note that the precomputation stage does not depend on the scale h or the kernel function w.

We start the sum accumulation process from the root node of the octree and assign the initial sum to be zero. Then we perform the following recursive procedure: if the current node satisfies the error criteria given by equation (2.21), then the sum is augmented by the amount equal to $w(|\mathbf{y} - \mathbf{x}_s^c| / h)\mathcal{N}_s$. Otherwise, the eight recursive calls to children are made, which return the accumulated partial sums within each of their subtrees; these partial sums are then summed up to form the result for the current node. If the node is a leaf, then the direct computation for its individual point contributions is performed.

In our application, we also use a simpler heuristic mentioned above as a substitute of (2.21). Let the distance between the point y and the center ω_s of the node is $dist(y, x^c{}_s)$, and the radius of the node is r_s, then an approximation to the distance between the point and the cell can be approximated as: $dist(y, \omega_s) = |\mathbf{y} - \mathbf{x}_s^c| - r_s$. Then we can use $dist(y, \omega_s) > \epsilon h$ to replace (2.21). This criteria simply uses the cell approximation for all the cells whose

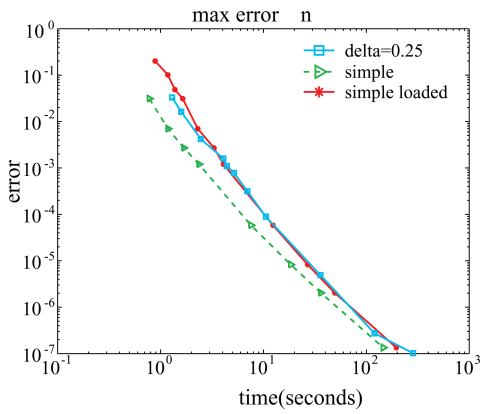


Figure 2.12: Log-log plot for the maximum error of weighted normal averaging computation versus the total runtime. The "loaded" curve corresponds to the simple strategy whose computation time for the condition check was artificially raised to be equal to the condition check for the equation (2.21). Otherwise, the experiment was the same as in Figure 2.11.

points lie outside of the ball of radius proportional to the scale of the filtering.

We compare the performance of our fast approximate evaluation method for the point-cloud of the Venus head model with 42K points. Figure 2.11 shows the log-log plot of the total runtime versus the error for various values of the parameter Δ as well as for our simple criteria. Each curve corresponds to one octree refinement strategy, and lower curves are better. We see that lower values of Δ result in a better performance of the method. It is interesting that as Δ becomes negative the curves show small improvement however the relation between the parameter ϵ and the obtained error goes out of control and the samples on these curves shift upwards.

For the uniformly sampled surface of the Venus head model the simple heuristic error condition seems to outperform the more complex error bound of (2.21). We have further explored the question of runtime comparison between a simple heuristic and the condition (2.21), with the conclusion that the time to evaluate the error condition itself has large influence on the overall runtime. We have artificially made the simple heuristic code spend the same amount of time in the evaluation of the condition itself. The results are presented in Figure 2.12. We see that when the time to compute the conditions themselves are equal the two criteria perform very similarly.

2.4.3 Accelerated feature extraction algorithm

We use the fast evaluation of the weighted sums to perform smoothing of the surfel cloud normal data for a sequence of scales h_j and then find the differences between the consecutive levels as described above. The process of finding the local extrema can also be accelerated by storing the extreme values of the differences within each cell of the octree. Additional time savings can also come from using simplified point clouds for feature extraction.

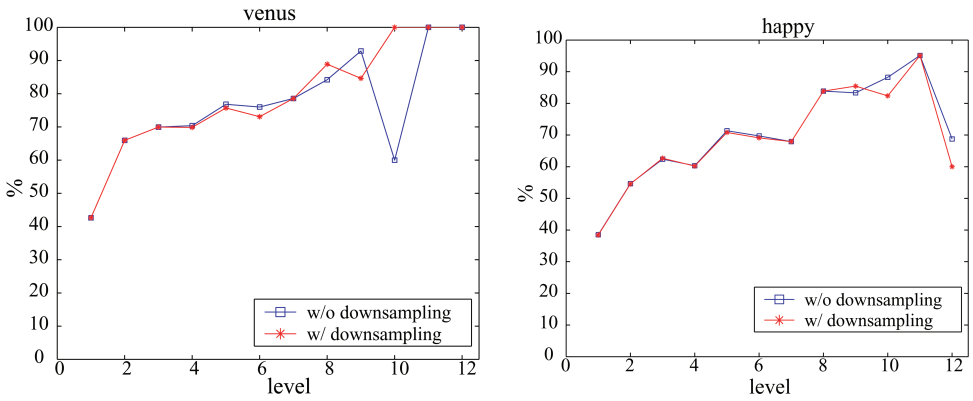


Figure 2.13: Repeatability comparisons for the feature extraction with downsampling. The percentage of matched features is shown on the vertical axis. The last levels contain very few features.

Indeed, as mentioned before, the distance between any two features located at the scale h cannot be smaller than h. Therefore, as the scale grows, the "resolution" at which the feature points are defined also gets coarser, and it makes sense to use a simplified point cloud for the extraction of the feature point set. We use this observation to accelerate processing of large point clouds: we only compute the filtered values of normals and their differences on a simplified point cloud at each scale. The simplified point cloud is built by a greedy approach that ensures that the minimal distance between points in a simplified point cloud is above a scale-dependent threshold, defined in our implementation as $\epsilon_j = h_j/16$.

Note that we always compute the filtered values based on the original point cloud data, using the same octree built during the initialization stage. Therefore, the values of filtered normals stay at the precision defined by the original surfels (we do not compute the filtered

values from the previous level, but always use the original surfels and the corresponding octree with precomputed values).

We compared the repeatability results for the accelerated procedure on with simplification on the venus and happy buddha point sets. Figure 2.13 shows the percentages of matched features and indicate that the simplification does not significantly affect the repeatability of extracted features.

2.5 Summary

In this Chapter, we introduced a novel method to abstract a set of feature points from point-based surfaces. This method can be easily extended for mesh-based surface. In fact, it provides more reliable set of features on mesh-based surface generally because the area weights computed from mesh.

Experiments shows the selected set of features persist under certain surface variation, which makes it an excellent representation for the surfaces. In the following chapters, we will discuss the applications of the features.

CHAPTER III

SIGNATURES FOR FEATURE POINTS

Surface signatures are descriptors that define local or global surface properties and generate possible surface correspondence between points with similar signatures. Signatures for a set of selected points can form an effective representation for a free-form surface for the purpose of fast surface registration and object recognition. In this Chapter, we describe a new definition of surface signatures and show its application for surface matching.

3.1 Related works

There has been a variety of surface descriptors proposed for applications like surface matching and object recogntion[39, 86, 85, 23, 81, 75, 20, 82]. Yamany and Farag[86] created 2D signature images based on curvature information for a set of salient points and use image processing tools to match signatures cross 3D surfaces. Sun and Abidi[82] generated feature descriptors is a set of 2D contours that are the projection of geodesic circles onto the tangent plane. Darbandi et al. [21] define signature for an oriented point cloud using 3D vectors that encode curvature, symmetry and convexity of surface around the point.

There are a set of surface descriptors that are defined based on the density and relative position of surface point position, among which the Spin Image[39] is popular and often compared by others. Frome et al. [23] evaluate various techniques in the context of recognizing noisy range scans. It is observed that signatures based on point distribution such as spin-images[39] and 3D shape contexts [23] are robust to outliers and can handle noisy scans. For automatic reconstruction of good quality 3D scans that contain little noise the

spin-image signatures become less expressive, and may not capture fine variations of the surface detail especially for small scales. On the other hand, curvature-based signatures are better equipped to handle fine surface detail and can use normal data.

Approximate alignment of partially overlapped surfaces is a straightforward practice for the signatures [39, 86, 23, 20]. In this chapter, we introduce a simple algorithm to find the matching between two overlapping surfaces to estimate the transformations between two coordinate systems. The result of rough alignment can be refined using one implementation of ICP algorithm [69]. This pair-wise surface provide important base to reconstruct 3D model with multiple scans of partial overlap.

In this chapter, we define local signatures which are based on the normal field near the surface and are approximately invariant to rigid transformation in Section 3.2. The features and their signatures are used for approximate pairwise surface matching (Section 3.3). Then we propose a surface modeling method as an application of pairwise surface matching in Section 3.4. We use several datasets to evaluate the performance of our signatures and matching algorithms.

3.2 Local surface signatures

Feature points from Chapter II have several attributes (position, normal, and scale) that can be used to perform matching between partial scans of the same shape. However, these attributes are not invariant under rigid transformation or surface scaling. Such transformation can usually relate the coordinate systems for a pair of scans. In order to enable efficient matching between shapes given in different coordinate systems, we endow each feature with a signature that is invariant to such spatial transformation. Given a surfel cloud P and a nearby point $\mathbf{x} \in \mathbf{R}^3$ together with a normal direction $\mathbf{a}$ and a scale $h > 0$, we define a local signature vector $\sigma_P(\mathbf{x}, \mathbf{a}, h)$ as is described below.

3.2.1 Definition of signatures

Given the normal direction $\mathbf{a}$ we define an orthogonal local frame $(\mathbf{u}, \mathbf{v}, \mathbf{a})$. The choice of $\mathbf{u}$ and $\mathbf{v}$ is not unique as they can rotate around the axis defined by $\mathbf{a}$ (we address this ambiguity below by performing Fourier transform to obtain a signature approximately invariant to the choice of $\mathbf{u}$ and $\mathbf{v}$).

Our signature computation starts by defining a $N \times M$ array of 3D points (ξ_{kl}) that sample a disc around point $\mathbf{x}$:

$$\xi_{kl} := \mathbf{x} + \frac{2lh}{M}\left(\cos(\frac{2\pi k}{N})\mathbf{u} + \sin(\frac{2\pi k}{N})\mathbf{v}\right);$$

where $k = 1..N$ and $l = 1..M$. For every sampled point, a normal direction is assigned to it, which is computed from the normal of points near that point. We define the corresponding array of normals as $\nu_{kl} = \mathbf{n}_{[P,h]}(\xi_{kl})$.

We determine a $N \times M$ array of real numbers by projecting the normals onto the direction connecting the signature center and the current sample location as follows (see the figure below for an example of signature samples with colors corresponding to the value of s_{kl}):

$$s_{kl} = \frac{\langle \xi_{kl} - \mathbf{x}, \nu_{kl} \rangle}{\| \xi_{kl} - \mathbf{x} \|} \tag{3.1}$$

We then apply a Discrete Cosine Transform to s_{kl} in the radial l-direction, followed by Discrete Fourier Transform in the k direction. We record the resulting magnitudes into the array $(\tilde{s}_{kl}), k = 1..N, l = 1..M$. The upper left corner of this array contains the filtered values that are approximately invariant to the choice of tangent vectors $\mathbf{u}$ and $\mathbf{v}$ if the

number of samples N around the disk is sufficiently large. We use $N = 32$ and $M = 8$.

We can now define signature vector by extracting the upper-left corner of the array $\tilde{s}$ so that

$$\sigma_P(\mathbf{x}, \mathbf{a}, h) := (\tilde{s}_{kl})_{k=1\ldots N', l=1\ldots M'}.$$

In most of our applications we use 24-dimensional signature vector by choosing $N' = 6$ and $M' = 4$ or 12-dimensional signature vector with $N' = 4$ and $M' = 3$. Based on our experience, the 24-dimensional signatures outperform 12-dimensional one for few models. Our signature contains positive numbers and is invariant to rigid transformations. For the scaling transform, the invariance requires multiplying the point coordinates and the scale of the signature by the same number. Since the absolute value of the Fourier coefficients is taken, the signatures will also be invariant under reflection transformation (for instance, the signatures of features located on the left and the right ears would be similar).

Figure 3.1 visualizes eight "basis" normal vector projection patterns for the 2×4 top-left corner of the signature matrix. We can interpret the four columns of Figure 3.1 as follows:

- The first column corresponds to a local spherical point, either convex or concave.
- The second column represents a non-symmetric component of the signature – the numbers in this column will be often close to zero.
- The third column shows a saddle-like behavior, near a hyperbolic point. It can also be used in a combination with non-trivial first column in order to get an elliptic (non-spherical) point.
- The fourth column would have non-zero entries near a corner of a cube where three creases come together in a symmetric fashion.

We can see that the first and the third columns have some qualitative relation to the curvature at a point (although it is hard to quantify that relation).

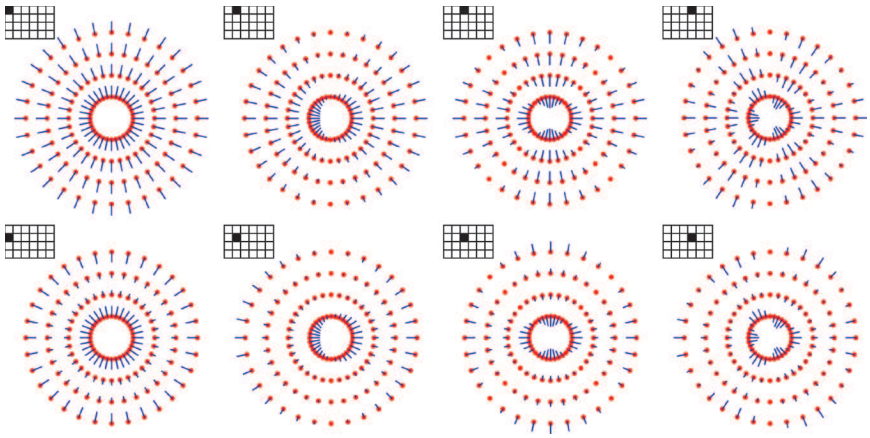


Figure 3.1: Projected normal patterns for eight "basis" signatures. The black box shows which entry of a 6×4 signature matrix is equal to one, white boxes are zeros.

Also, note that while the input normals may be noisy, for the signature computation we sample a smoothed normal field. Additional denoising happens when we exclude the high frequency components of the Fourier transform.

The local signatures introduced in this section can be found for any position, normal, and scale values. In our application, we shall only use signatures computed at the feature points. For a feature $f = (\mathbf{x}(f), \mathbf{n}(f), h(f))$ we denote its signature as $\sigma(f) := \sigma_P(\mathbf{x}(f), \mathbf{n}(f), h(f))$.

3.2.2 Signature stability

In our signature computation we use the normals evaluated on the tangent disk to the surface, hence the actual locations where the normal field is sampled are not on the surface. We have experimented with computing normals at projected surface locations but found that the surface projection operation can be very discontinuous in that small noise in either the tangent plane orientation or the underlying model sampling can introduce huge changes in the computed signature. Figure 3.2 illustrates this by comparing the errors of signatures extracted from the original Buddha model with the signatures computed at the same locations with the same normal and scale but on a resampled version of the model. We see that the projection operation introduces a lot of outliers. Therefore, in our matching application we chose to use signatures computed as described above in Section 3.2.1.

3.2.3 Signature-filtered feature points

The feature-finding procedure of Section 2.2 does not perform any filtering on the found extrema of the scale-space difference function. Thus, in an almost flat region with very small amount of noisy displacement, there will appear a lot of unreliable features. We notice that these "flat" feature points will typically produce local signature vector of a small magnitude. Therefore, we eliminate all the features with signature vector magnitudes below a user-specified threshold. For all the datasets of this paper we remove all the features whose signature vector has length less than 0.2 (note that the signatures are based on a fixed number of unit normals thus no additional normalization is required for this threshold).

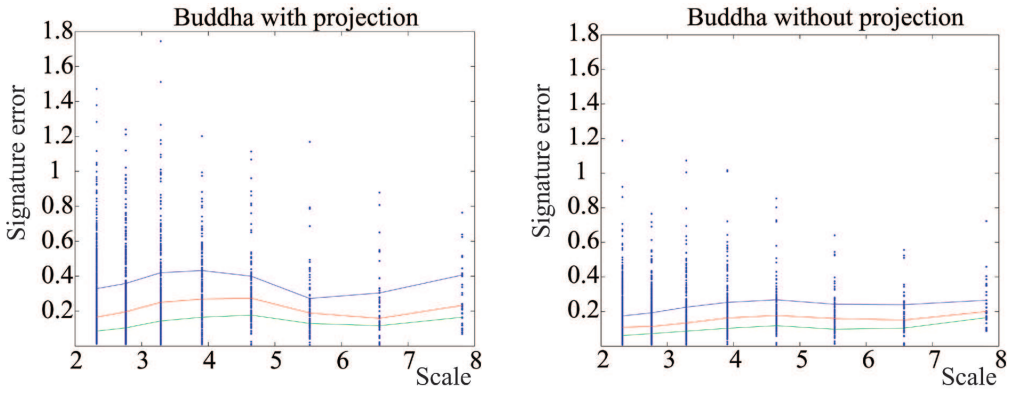


Figure 3.2: Signature stability with and without projection. Error of signatures computed at the same locations on resampled Buddha model. On the left, the signature normals are computed at the projected surface locations, on the right the signatures are computed off the surface (as defined in Section 3.2.1). Each feature point is plotted with its scale on the horizontal axis and its error on the vertical axis. The three color curves show the 25%, 50%, and 75% error percentiles for each scale.

3.3 Pairwise surface matching algorithm

With the detected features and assigned signatures, we are ready to introduced our surface matching algorithms. The input to our pairwise matching algorithm is a pair of surfaces given in different coordinate systems related by an unknown rigid transformation. These could be either two scans of roughly the same size, or a scan and a (partially) reconstructed surface model which can be much bigger. The result of matching is the best candidate for the rigid transformation (R, t) and the estimated percentage of overlap between two

surfaces.

We designate one of the surfaces as the *model* M and the other one as the *scan* S. We compute the feature sets as Ω_M and Ω_S for both surfaces, and compute the signatures at every feature point (also using its normal and scale). Then, for each feature $s \in \Omega_S$ of the scan we find the set $\omega_M(s)$ of $K(s)$ features whose signature vectors are the closest to $\sigma(s)$. The signatures of the model surface $\sigma(\Omega_M)$ are placed in a KD-tree data structure to facilitate fast spatial querying.

The number of model feature points $K(s)$ that are included in the set $\omega_M(s)$ is determined as follows: we first find a fixed constant number K' of top matching neighboring signatures from $\sigma(\Omega_M)$. Denote the feature points corresponding to the found signatures as $m_1, \ldots, m_{K'}$. We compute the distances from each found signature to $\sigma(s)$: $\Delta_k = \| \sigma(m_k) - \sigma(s) \|$, and order the features in the order of increasing Δ_k. Thus, $\Delta_1 \leq \Delta_2 \leq \ldots \leq \Delta_{K'}$ and we find the smallest value of $k > 0$ such that $\Delta_{k+1} > 1.2\Delta_1$, and discard all the features that follow this index k. Thus, at least one match is found for every feature (we use $K' = 10$ so at most ten matches can be found).

The matching problem consists of finding a rotation matrix R and a translation vector t such that after the transformation the two surfaces overlap within some common region. In order to find the transformation, we consider pairs of *scan features*, and for each pair (s, s') we consider all the corresponding pairs of *model features* (m, m') such that $m \in \omega_M(s)$ and $m' \in \omega_M(s')$, and for each match $\mu : (s, s') \mapsto (m, m')$ we estimate the rigid transformation that maps the positions and normals of the scan feature pair onto the corresponding positions and normals of the model feature pair. Thus, a rigid transform (R_μ, t_μ) is found using the normal direction and position correspondences at the matched pair of points by a method similar to unit quaternion method of [36]. Since there is possible false matching between two set of signatures, we need to quantify whether this found transform gives a good overlap between the scan and the model. Since this needs to happen for a large number of matches μ, we cannot use an exact error evaluation. Therefore, we shall use an

approximate voting procedure that takes into account the correspondences established by signature matching.

The vote is computed for a rigid transform (R, t) by summing the individual contributions $V(s)$ of all the scan features $s \in \Omega_S$. The contribution $V(s)$ is computed as the sum of unit votes for each $m \in \omega_M(s)$ which satisfies $\parallel Rx(s) + t - x(m) \parallel < d_1$. Here, d_1 is the threshold that is set to a multiple of the surface resolution (median distance between closest neighbor points in the input data). Thus, only the features that are brought close together by the transformation contribute to the vote, and their contribution is proportional to the quality of their match.

The search for the best rigid transform proceeds until all the feature pairs are considered. The rigid transform with the largest vote is returned as the found solution to the approximate matching problem.

Given the best transform candidate for two surfaces, we compute an approximate overlap percentage $OV(S, M)$ by counting the percentage of points in S that are within a threshold distance d_2 from the model surface M. We choose d_2 to be the smallest scale of the scale-space (based on our experience, we choose the smallest scale as two times the surface resolution of the input point cloud). We only consider point matches that have consistent normals (with angle difference less than ten degrees). This simple estimate works well for uniformly sampled surfel clouds. Note that the best transform candidate only align the two surfaces roughly. To register the model and scan accurately, we run ICP (we use ICP implementation of [71]) after computing the overlap estimate.

3.4 Multi-view surface matching

In this section we describe a procedure that automatically performs multi-view alignment of a set of scans for an object. Given a set of K scans, we run $K(K-1)$ pairwise approximate matching as described in the preceding section and order the resulting pairs of scans by their estimated overlap percentage. Each pair has an estimated (R, t) whose quality will

deteriorate as we proceed down this ordered list to the pairs with low overlap.

After the pairs are ordered, we start merging them proceeding from the top of the list and performing ICP alignment for pairs of scans starting from the estimated approximate (R, t) We keep track of who merged with whom and maintain a disjoint set collection of scan indices (see Figure 3.4 for the illustration). We skip all the pairs that already belong to the same merged component. Each merged component has scans aligned and specified in the same coordinate system. Effectively, we perform a greedy construction of a spanning tree for the set of scans. This is a very simple procedure and we do not perform cross-validation or discrete search as described in [37]. The fact that this simple merging algorithm works well attests to the quality of our pairwise matching.

3.5 Results

We evaluated our approximate matching procedure on several sets of laser scans. These included scan sets for three angel models from the work of Huber and Hebert [37], the scans of bunny and buddha model from Stanford repository, and the Arrigo data set which includes 177 fine scans of a basrelief. The following table enumerates data sets with the number of scans, average number of points per scan, and matching times. The following discussion will give more detail.

Table 3.1: Time for multiple view surface matching for different models

Dataset	Num.of scans	Points per scan	Init time	Matching time
Angel1	17	5K	3m28s	50s
Angel2	20	5K	4m24s	63s
Angel3	18	5K	3m41s	74s
Bunny	10	20K	10m	94s
Buddha	15	35-70K	31m	7m35s
Arrigo(I)	30	50-70K	90m	32m
Arrigo(II)	147	30-70K	see	text

3.5.1 Angels, bunny, and buddha datasets

We reconstruct the three angel models, the bunny model, and the buddha model using the multiview matching procedure described above without any user input. The timing results are given in Table 3.1 and the reconstruction results can be seen in Figures 3.3. Initialization time includes computing the scale-space, extracting features, and computing signatures. The matching time includes all other operations for the multiview matching including ICP alignment. We believe that our method is competitive with the method of [37].

Figure 3.3: Result of automatic alignment of scans of the Angel models: Angel1, Angel2, and Angel3.

3.5.2 Arrigo reconstruction

The above multiview matching procedure requires quadratic time with respect to the number of scans. For the datasets that cover the same area multiple times it makes more sense to perform two-stage processing. The first stage merges a set of scans that cover most of the surface into a single *combined* point cloud model using the multiview matching from Section 3.4, and the set of features and signatures can be computed for this combined model. In the second stage, all the remaining non-merged scans can be matched to this combined model one-by-one. Since the combined model covers most of the surface, it will have non-trivial overlap with each scan and the pairwise matching should work.

The data set of Arrigo basrelief consists of 177 scans separated into four parts each corresponding to a single pass over the whole surface. We took the smallest pass containing

Figure 3.4: Multi-scale surface matching results. Top left: during the merging of the first stage. First 10 pairs merge and form five connected components (visualized here in their final positions, each component in one color). Top right: the combined result of the first stage of the Arrigo basrelief alignment. 30 scans are aligned automatically. Bottom: final reconstruction from 174 scans.

30 scans and run the first stage of our multiview matching that created a combined model consisting of 700K points shown in Figure 3.4.

In the second stage of Arrigo reconstruction we performed pairwise matching of each of the remaining 147 scans to the combined model. As the result, 144 scans were matched successfully and 3 scans could not be matched automatically. Note that this second stage matches a small scan to a big model: this would be impossible to do with a random search initialization. See the final reconstruction result in Figure 3.4.

The three non-matched scans were easy to detect by looking at the maximal error between the initialized point cloud and its points after ICP transformation. All three bad matches had error above 50mm, whereas all the good matches had their error below 25mm as shown in histogram of Figure 3.5.

The processing for the Arrigo scans were split into two stages: the preprocessing stage and the actual matching stage. In a preprocessing stage each scan was first simplified to approximatedly 50K points, and all of the processing happened with these simplified surfel clouds. For each scan, a scale-space representation was built by smoothing the model for

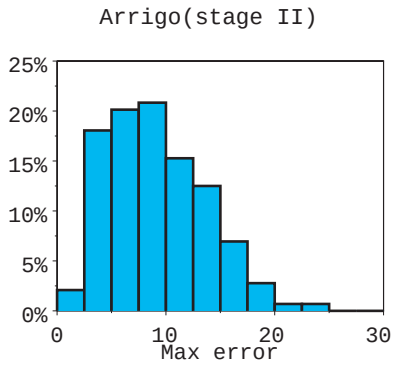


Figure 3.5: Errors of surface alignment of 144 scans to the partially reconstructed Arrigo model. Histogram of maximal difference between the initialized point cloud and the exactly aligned version after ICP. The histogram is given for 144 scans matched in the second stage to the combined data from the first stage. Horizontal axis is in millimeters (the statue height is 600mm).

10 levels. This was the most time consuming of the preprocessing steps and took about two minutes (all the times will be given for a 50K point scan running on a 2.8GHz Pentium 4 PC). In the remaining two steps, the extrema of the inter-scale difference function was found (5 seconds), and the signatures were computed for each extracted feature point. The number of features varied greatly depending on the complexity of any given scan. For a 50K point model, the number of found features were on the order of 1% (or 500). We used the parameter value $C = 1/\sqrt{2}$ for the feature set extraction of Section 2.2.3 (this reduces the number of features and makes the matching perform faster). The signature computation time for these feature points was about 30 seconds. The multiview matching of the first stage was performed in about 32 minutes. The resulting combined pointcloud was simplified to 700K points and the features and signatures were computed for it in about 35 minutes. The second stage processing for matching each of the 147 scans to the combined model took about 16 seconds per scan (4 seconds for approximate matching and 12 seconds for ICP final alignment). The current bottleneck of our processing is the smoothing stage, this time should be significantly improved by using fast multipole-like methods.

3.5.3 Pairwise surface matching with unknown rescaling factor

When the scaling factor between two surfaces is known, we only need to match signatures of features on the nearby levels of the scale-space representation. In our application, when processing a scan feature s from the scale h_j, we only match the model features at levels $j-1$, j, and $j+1$.

In some applications, it is important to be able to perform matching when there is an unknown rescaling factor between two surfaces. In this case, each feature of the scan surface is matched to features on every level of the model surface. The ratio of the scales of found matched pairs gives an initial estimate of the rescaling factor, which is further refined by the transform estimation step. An example of scale matching is given in Figure 3.6.

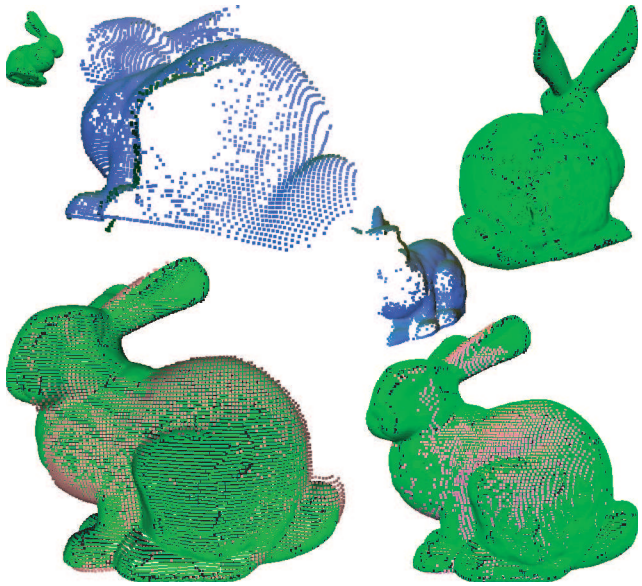


Figure 3.6: Surface matching with unknown rescaling factor. Two examples of matching a noisy scan to a known model with unknown rescaling factor.

The quality of scans used in this experiment was significantly worse (coarser and noisier) than in the previous examples (the scans were acquired by a moire-based scanning technique applied to a 3D-printed replica of the Stanford bunny). Our matching method was successful, and found the rescaling factor and the rigid transform that brought these scans into correspondence with the bunny model. No ICP was run in this case.

3.5.4 Compare with Spin Image on multi-view surface matching

In this section, we compare the matching performance of our signatures with Spin Image. For this purpose, we compute both Spin Image and the signature proposed in this document for a set of selected feature from a given shape. Unlike our signature, which is computed from surface normals, Spin Image is defined as the histogram of relative point position around the selected points. It has been successfully used for the purpose of surface matching and object recognition.

One Spin Image around an oriented point near the surface is computed by projecting the 3D point of the point's neighborhood area onto 2D image. The oriented point defines a local frame with the z-axis along the normal direction and the radial direction on the tangential plan. The 3D space around the point is divided into bins and the size of 2D image depends on the number of bins along the radial and normal direction. The number of points that fall into a particular bin is accumulate to generate the histogram and to be used to compute the 2D image. The Spin Image is treated as a high dimensional vector with 100-400 elements for applications like image matching. The high dimensional vector is not efficient to perform spatial query. Thus, we us Principal Component Analysis to reduce the high dimensional signatures to be a 12-dimensional vector.

We used both signatures for multi-view surface alignment as proposed in this chapter and tested them on the 177 Arrigo scans and 36 Armadillo scans. The results show that the our signatures succeed for both data sets. The Spin Image align all the 36 scans for Armadillo model and 14 out of 177 Arrigo scans are misplaced. Figure 3.7 depicts the result of multiple view scans of Armadillo model and two sample scans that failed for the Arrigo model. All the misplaced Arrigo scans are part of the bottom of the statue (refer to Figure 3.4 that contains relatively smooth repeated features from clothes creases. These features are less distinctive compared with the features from the upper parts like faces and different poses of hands and arms.

Figure 3.7: Comparison of the proposed signature and Spin Image on multiple surface alignment.

3.6 Summary

We defined a new point signature and introduced a rough pairwise surface alignment method by matching signatures of a set of selected features from two surfaces. We then employed the pairwise surface matching algorithm for reconstructing a model from multiple scan fragments acquired from arbitrary view points. Experiments show our method can handle real-life data sets. In the following chapters, we will continue to discuss the applications of the proposed features and signatures.

CHAPTER IV

ALIGNMENT OF RANGE DATA TO CAD MODEL

In recent years, 3D surface scanning technology has found multiple industrial applications. One of these applications is in the area of dimensional metrology (or error measurement). In a typical metrological setup, a manufactured part needs to pass quality control to assure its conformance to the design specifications. These days, a digital CAD surface model is usually available which precisely describes the desired shape. Multiple laser scans of the actually produced object are therefore compared against this CAD model. Often, the scanning facility will have the special fixture for physically positioning the model in a prescribed position, so that both the measured scans and the specified CAD model are given in the same coordinate system. Then, after refining the mutual alignment with an error-based ICP algorithm the error can be directly measured and reported.

In some applications however it is inconvenient or even impossible to use the positioning fixture. Therefore, the initial alignment step needs to be performed to find the rough alignment of the partial surface scan and the corresponding CAD model. A similar problem also appears in the surveillance and shape recognition applications, where the positioning of the acquired object (or of the scanning equipment) is not always under full control of the acquisition system user.

Our focus in this chapter is an initial approximate alignment of a partial and noisy scan to a given CAD model. This *scan-to-model* alignment (Figure 4.2) is a somewhat easier task than the *scan-to-scan* alignment proposed in Chapter III because we can assume that the scanned surface is (at least approximately) the subset of the full object acquired in a

single scanning pass. It is often unknown whether they have any overlap at all for the scan-to-scan problem, and any algorithm for initial rough alignment of such partial scans should find ways to dealing with this problem. Note that modern laser scanners can include the dynamic "scanning head" positioning, so that the produced partial scan is not necessarily taken from a single viewing position, and may actually wrap around the scanned object. Even so, there is no need to estimate the overlap area between the scan and the model, and one can expect to obtain simpler matching algorithms.

The initial idea of this chapter is proposed for automated inspection approach as reported in [46]. Figure 4.1 shows an example of the setting of an inspection system. In this chapter, we extended this approach for a simple feature-based algorithm for approximate scan-to-model alignment and consider several modifications for making it more efficient and robust. We evaluate the feature point selection strategy and compare it to the random point selection. We improve robustness of surface alignment by excluding detected boundary points from matching. We also use simple curvature-based signatures to achieve faster performance. The content of this chapter appeared as a conference paper[45] and two journal papers [42].

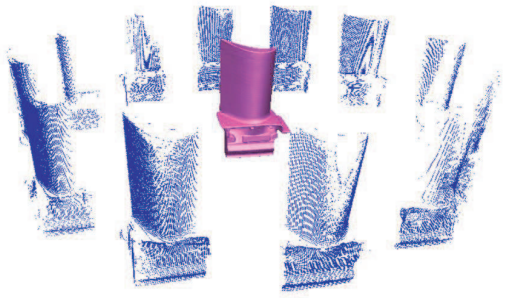


Figure 4.1: Multi-view scans of one object and their nominal CAD model. The CAD model of Turbine Vane is shown in the middle. Twelve scans obtained from a manufactured Turbine Vane are shown from corresponding points of view.

4.1 Related works

Traditional inspection usually a calibration process where known features and locators [58]are used to establish the reference coordinate system. When non-contact technology is employed, prior knowledge of the shape of the inspected part is not necessarily required for the measurement acquisition process.

After digitization, there are computation methods for inspection and alignment [18, 67]. Most approximate alignment algorithms rely on establishing correspondences between sets of points on the surfaces being matched, and then use these correspondences to estimate the rigid transformation that brings these sets of points into the approximate alignment. Two kinds of approaches are used for this purpose: feature-based and random sampling selection of representative surface point set. We mention several feature-based approaches. In the work of Yamany and Farag [86], points with high curvature on both the surface and the model are detected to find the corresponding pairs. Gelfand et al. [27] used points with distinctive integral volume descriptor on the scan and handle corresponding search on the entire model. Li and Guskov [43] used multi-scale features that are robust to surface resampling. The random-based approach was used in the spin-image work of Johnson and Herbert [39], where small subsets of points on surfaces are selected randomly, the spin image signature is computed for every selected point, and a search is performed over all the pairs of feature points that are close in the signature space. A similar approach was used in the automatic surface reconstruction work of Huber and Hebert [37]. In another approach, Chen et al [16] select a small set of points (the minimal number is three) on the scan and slide them over the model performing exhaustive search.

In this chapter we evaluate different strategies of choosing the representative point set. In particular, we compare the popular method of randomly choosing the points to be matched [23] and the method for selecting salient surface features presented in [43], as well as its single-scale version. For any method of selecting a representative point set there is a tradeoff between the number of selected points and the reliability of the resulting matching

procedure. Larger feature point sets tend to increase the matching time but produce better alignment results. We explore the performance of different feature point selection methods by varying the feature set size and comparing the performance of our alignment algorithm on a large number of scans.

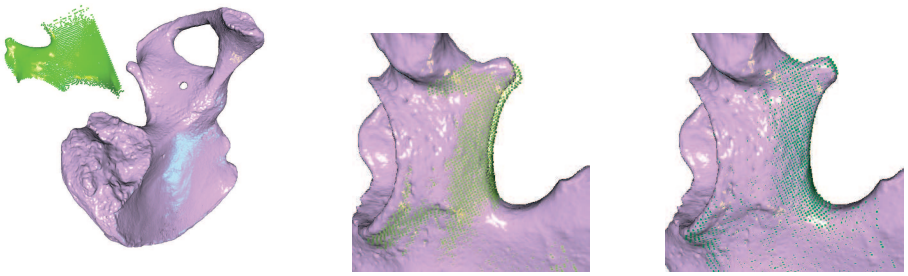
The runtime performance of feature-based surface matching methods depend both on the overall number of feature points used in the algorithm and on the particular method of estimating the rigid transform aligning two shapes. Various methods use different number of point matches to estimate the transformation. This also depends on the information available at each feature point. For instance, when only feature point positions are considered, the estimation procedure needs to use at least three matching pairs of points to find the transformation [16]. Using a branch-and-bound algorithm that checks the consistency of a set of matched pairs, larger sets of potential matches can be built for robust estimation of alignment transformation, as shown by Gelfand et al. [27]. In some applications, the quality of the scanned data does not allow such precise association of feature pairs due to higher noise in the measured surface data. In such applications, it makes sense to use a larger set of potential matches that are not necessarily very reliable. In this case, enumerating all the possible triples of feature matches can become excessively slow.

One strategy to reduce the amount of processing time is to use more information available at a surface point. For instance, in [43] the approximate alignment algorithm uses two pairs of matched positions and their normals to estimate the transformation using the unit quaternion method. In this paper, we explore an algorithm that considers only single matched pair of features for each estimation of the rigid matching transform. In order to do so, we need to fix a directional degree of freedom in the tangent plane at each matched point. This can be done either by exhaustively trying a dense set of direction, or by locally estimating major curvature direction at a point. A direction in the tangent plane, together with the position and the normal can be used to estimate the rigid transformation.

In order to further improve robustness of our method perform a boundary classification

process, and exclude the near-boundary points from matching. Adamson and Alexa [3] define smooth and natural boundary on a non-oriented point-based surface. In this paper, we assume that the surface points are associated with normals, and we apply a simple algorithm to find boundary points which based on the angle distribution of neighbors of a point. In addition, we accelerate the feature correspondence search by using curvatures as simple signature values.

This chapter is organized as following. We describe the initial approximate matching algorithm in Section 4.2 and evaluate different strategies of choosing the representative point set in Section 4.3. Section 4.4 discussed how to improve the robustness of the alignment algorithm, including finding the principle tagent direction and excluding detected boundary feature points from matching. Some Example of alignment results are given in Section 4.5



a b c

Figure 4.2: An example of matching a scanned surface to a Hipbone model. (a) is the relative position of scanned surface (green) and the CAD model (purple); (b) shows the zoomed-in view of the result of initial matching and (c) shows the zoomed-in view of the result after ICP algorithm.

4.2 Initial approximate alignment algorithm

The goal of approximate scan-to-model alignment algorithm is to find the rigid body transformation $T = [R\,\mathbf{t}]$ that would align the scan surface $\mathcal{S}$ to the model surface $\mathcal{M}$. If the match is perfect, we would achieve $T\mathcal{S} \subset \mathcal{M}$; however for the approximate point cloud data, we would like to find the transformation T that minimizes the distance between the transformed scan and the model surface $d(T\mathcal{S}, \mathcal{M})$. If each point on the scan and model surfaces is associated with the normal direction and a reference tangent direction (e.g. the

estimate of principal curvature direction), then a single point correspondence between the scan and the model is enough to estimate the rigid transform. Indeed, suppose that a scan point $\mathbf{p} \in \mathcal{S}$ corresponds to the point $\mathbf{p}' \in \mathcal{M}$ on the model. The local orthogonal frame $(\mathbf{n}, \mathbf{k}, \mathbf{m})$ at $\mathbf{p}$ is given by the normal $\mathbf{n}$, and the two directions $\mathbf{k}$ and $\mathbf{m}$ on the tangent plane with $\mathbf{m} = \mathbf{k} \times \mathbf{n}$, and the corresponding local frame at $\mathbf{p}'$ is given by $(\mathbf{n}', \mathbf{k}', \mathbf{m}')$. Given this information, the transformation $T_{pp'} = [R_{pp'}\ \mathbf{t}_{pp'}]$ can be estimated as follows:

$$R_{pp'} = \begin{bmatrix} \mathbf{n}' & \mathbf{k}' & \mathbf{m}' \end{bmatrix} \begin{bmatrix} \mathbf{n} & \mathbf{k} & \mathbf{m} \end{bmatrix}^T \tag{4.1}$$

$$\mathbf{t}_{pp'} = \mathbf{p}' - R_{pp'} * \mathbf{p} \tag{4.2}$$

We assume that the normal information is given as the input data for the point cloud; the point with associated normal is also called a surfel. Given a cloud of surfel data, we find two directions $\mathbf{k}$ and $\mathbf{m}$ using the method described in Section 4.4.1 or Section 4.4.2. We assume that both the input CAD model and surface scans are represented as surfel clouds; therefore most of our processing algorithms will work in essentially the same way on both the scan and the model.

In practice, the correspondences between the scan and the model points are not known, and a variant of exhaustive search over all possible correspondences is applied by selecting a representative set of surface positions from both the scan and the model surfaces. The particular choice of the representative sets will be discussed in the next section. We denote the representative sets for the scan and the model surfaces as $\Omega(\mathcal{S})$ and $\Omega(\mathcal{M})$ correspondingly. For each surfel s from $\Omega(\mathcal{S})$, we can consider every surfel s' from $\Omega(\mathcal{M})$, and check the quality of the estimated transformation: we use an error computed on a random sampling of points of the scan to approximate the L^∞-error from the scan surface after transformation to the model surface. The detailed matching algorithm is shown in Table 4.1, and the procedure to verify a transformation matrix is shown in Table 4.2.

We use the simple approximate scan-to-model matching algorithm as the basis for our explorations in this chapter. In order to make it more efficient and robust, we consider

Table 4.1: Initial matching algorithm

SurfaceMatching(S,M)

Initialize the return value T_{best};
Select a set of surfels $\Omega(M) \in I_M$;
Select a set of surfels $\Omega(S) \in I_S$;
Initialize Δ_{max};
For every surfel $m \in \Omega(M)$
    For every surfel $s \in \Omega(S)$
        Compute the transformation $T = \begin{bmatrix} R & t \end{bmatrix}$ from m and s using
            equation 4.1 and 4.2;
        Δ = Verify(T, Δ_{max});
        If $\Delta > \Delta_{max}$
            continue;
        Else
            Δ_{max} = Δ;
            $T_{best} = T$;
        EndIf
    EndFor
EndFor
Return T_{best};

various modifications of this basic algorithm. These modifications are described in the following sections and are concerned with the selection of the representative point sets on matched surfaces (Section 4.3) and the selection of directions $\mathbf{k}$ and $\mathbf{m}$ (Section 4.4.1 and Section 4.4.2). We find that the scanned range data contain a lot of noise near the boundary; hence we exclude the points near the boundary from the matching process; the details are described in Section 4.4.3 below.

4.3 Selection of points

It is time-consuming and unnecessary to use all the points on the scan and model surfaces to find corresponding points for approximate matching. We compared the efficiency of the approximate scan-to-model matching algorithm for two different strategies of selecting the representative sets of points: the random sampling and the feature point selection. Two ways of selecting feature points are used: the first one is a multi-scale procedure of Li

and Guskov [43], and the other one is its simplified single-scale version described in this section.

Table 4.2: Verify transformation matrix T=[R t]

Verify(T, Δ_{max})
$\Delta = 0$;
$\Delta_{local} = 0$;
Select a set of $N_S = 100$ points $\Psi(S) \in I_s$ on the scan surface.
For every point p in $\Psi(S)$
Compute the position after transformation p' = $Rp + t$;
Find the closest point q of p' on 3D CAD model surface.
Δ = $d\,(p', q)$
If $\Delta > \Delta_{local}$
$\Delta_{local} = \Delta$;
EndIf
If $\Delta > \Delta_{max}$
Return Δ
EndIf
EndFor
Return Δ_{local}

4.3.1 Random point selection

The random selection of the representative point set has been often used before [39, 23]. In our work we adapt the method of [23]: we divide the space into uniform voxel grid and select at most one surface point for each voxel that intersects with the surface. The selected point is chosen randomly from the set of surface points contained in a voxel. If the selected point is on the boundary (see Section 4.4.3 for details), this point is discarded and a new random point is tried. The number of tries is bounded by the number of surface points within a voxel.

4.3.2 Multi-scale feature-based point selection

The feature-based point selection approach relies on the assumption that the scan point cloud is a partial noisy approximation of the prototype model surface, and therefore it has a similar set of geometric features (away from the boundary). It is should therefore be possi-

ble to use that set of point features for the matching search. Of course, this only works if we have a feature extraction procedure that is stable under resampling perturbation. One option is the multi-scale features described in Section2.2.3. These multi-scale features are especially useful for surface matching with unknown uniform rescaling which can be useful for shape recognition applications. However, in the scan-to-model matching applications there is typically no unknown rescaling between the scanned surface and the prototype CAD model. Therefore, we introduce a simpler single scale procedure of feature point extraction which does not require construction of multiple levels of the scale-space, only two levels of smoothing are used(rather than eight-ten levels used for multi-scale procedure[43])

4.3.3 Single-scale feature-based point selection

In order to compute the single-scale feature points, we build two adjacent levels of smoothing $j-1$ and j for some fixed j and compute the normal difference d_j as described in Section 2.2.3. Then, the local extrema of this discrete function are used as feature points. Using notations of the Section 2.2.3 we call a point i on level j a neighborhood maximum if $d_j(i) > d_j(i')$ for all $i' \in N'_j(i)$ where $N'_j(i) \stackrel{def}{=} \{k : |\mathbf{p}_k - \mathbf{p}_i| < h_j\}$. The neighborhood minima are defined analogously. We use a larger radius for detecting single-scale features (twice the radius we used for the multi-scale features) so that the resulting number of features does not become too large. The single scale condition is generally weaker and results in more features detected, and in order to obtain an efficient matching procedure we would like to limit the number of selected feature points.

Thus, the local features defined in this fashion belong to two classes (minima and maxima), and two features from the same class cannot be closer to each other than the radius of the neighborhood used to define them. In addition, since the features are detected using only the smoothed versions of the pointclouds the effect of noise is mostly removed.

Given two finely sampled point clouds representing the same surface, we expect their single scale feature sets to be similar. In order to evaluate how similar the feature sets

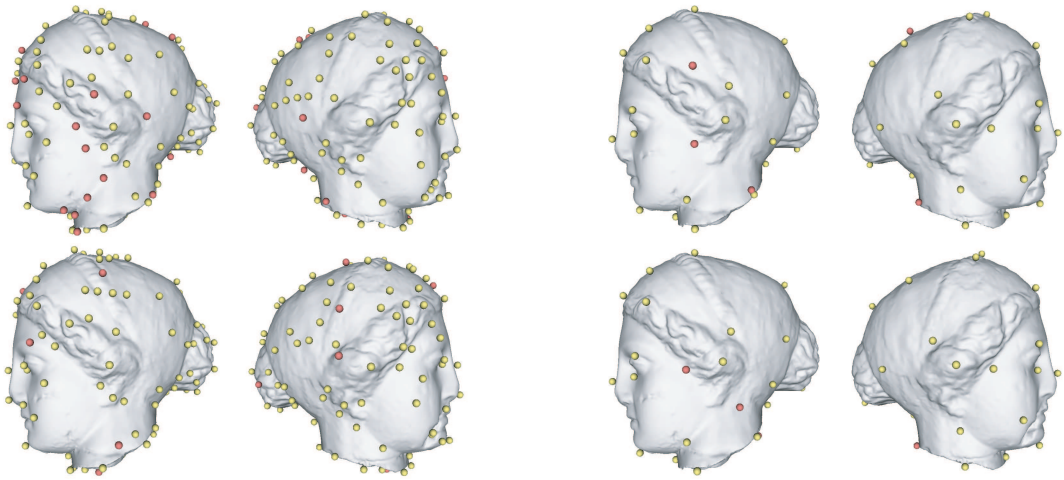


Figure 4.3: Single scale feature points on two representations of a Venus model. Features are detected for two scales and marked with spheres. Yellow spheres show corresponding features and red spheres show the features which don't have correspondence (same scale) on the other model. Top: features on venus with 30K points; Bottom: features on venus with 50K points.

are we perform the repeatability experiments first introduced for multi-scale features [43]. Given two representations of a surface model, we extract a set of features on each of them for each scale, and find the correspondent feature sets. Figure 4.3 illustrates the process: two point clouds of the Venus head model are considered (with 30K and 50K surfels correspondingly), and the feature points are extracted and color-coded according to whether there exists a close corresponding feature in the other feature set, that is within $h_j/2$ distance. The set of features with correspondence is shown in yellow, and features without correspondence are shown in red. The repeatability here is defined as the number of yellow features divided by the total number of features.

Figure 4.4 compares the repeatability of single-scale and multi-scale feature extraction procedures on three different models with five different scales. "All features" curves show the overall number of features detected in a pair of point clouds and "Corresponding features" curves show the overall number of features matched between these two points clouds. The three models are Venus, Happy Buddha and Stanford bunny. For every model, we have one original point cloud and create two modified versions in the same way as was reported in [43]. One modification introduced resampling and the other modification added

noise to the surface data in both the position and normal. The plots in Figure 4.4 report the repeatability results under these two modifications. We see that the repeatability of single-scale features is competitive with the original multi-scale features.

We compared the random-based method and feature-based methods on two sets of surfaces: Stanford bunny model with 10 scans and three angel models with 55 scans [37] are used for this purpose. By adjusting the size of voxels, we choose different number of points from the scans and models, and compare the initial matching results using these randomly selected points with the results using multi-scale feature points and single scale features. Section 4.5.1 describes details of our comparison experiments, and Figure 4.8 summarizes the comparison results.

4.4 Resolving tangent plane orientation ambiguity

In order to enable the approximate alignment procedure that only uses a single pair of correspondent points, we need to establish a local orthogonal frame of reference at both positions. The normal direction can be reliably estimated even for fairly noisy scan data and is used as one coordinate direction. The second vector orthogonal to the normal direction can sometimes be found by the estimation of the major curvature axis. However, this estimate can be sensitive to the noise and resampling, especially for near-umbilic surface points. An alternative approach is proposed to extends the matching algorithm by exhaustively searching through a number of possible sampled orientation for each point correspondence. In this section, we evaluate both approaches. We first describe our curvature direction estimation procedure, and then give details of direction sampling.

4.4.1 Principal curvature direction estimation

Curvature tensor estimation is a well-studied problem for triangular meshes, and there are several good algorithms widely used in practice [83, 70]. For the point-based surfaces, the curvatures are also important for the surface modeling and denoising applications [63, 40].

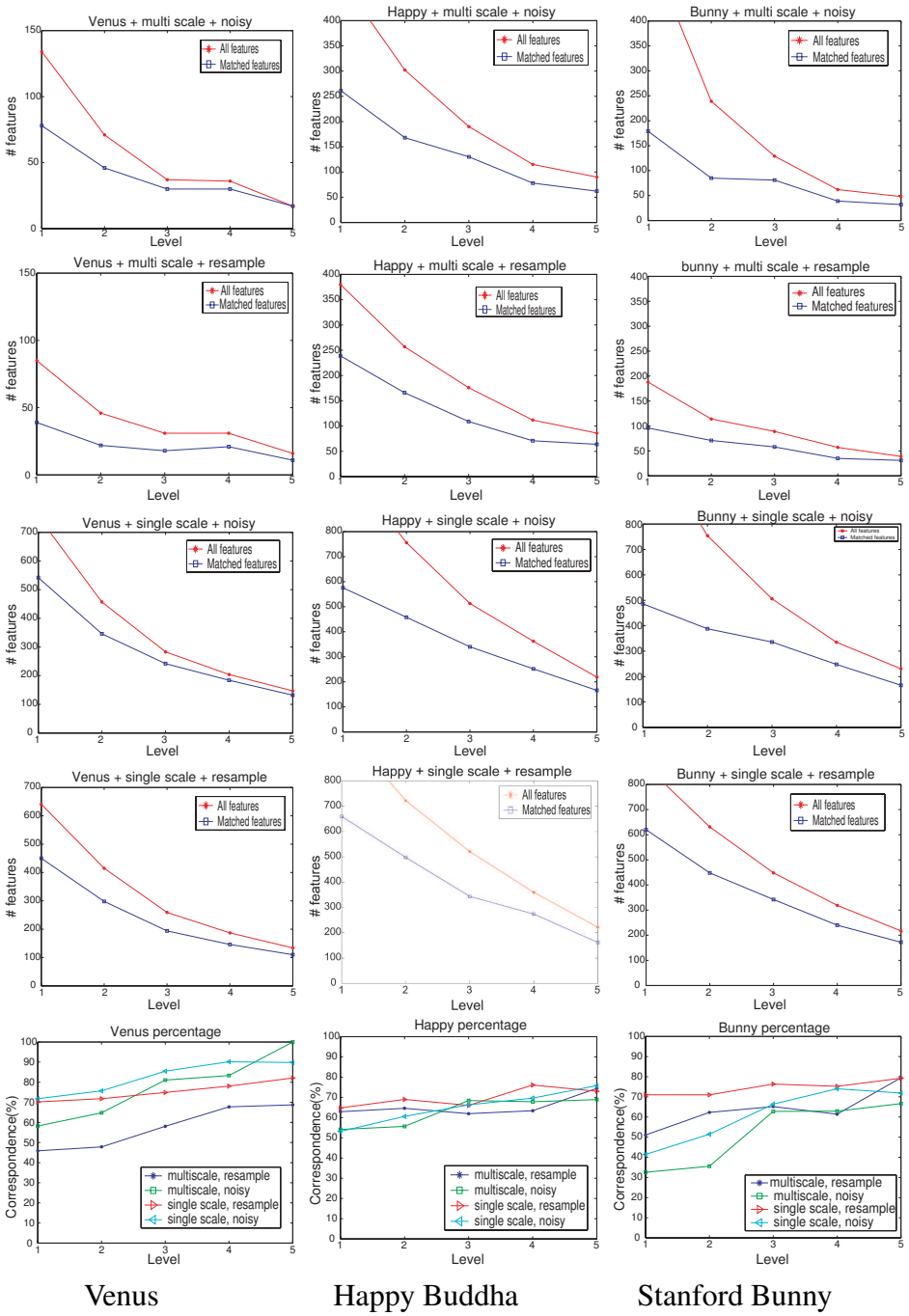


Figure 4.4: Repeatability of single-scale feature points. Bottom shows the percentage of matched features over all features.

We adapt an approach that is a point-based version of the curvature estimation algorithm of Rusinkiewicz [70].

Given a surfel $\mathbf{s} = (\mathbf{p}, \mathbf{n})$ and one of its neighbors $\mathbf{s}_i = (\mathbf{p}_i, \mathbf{n}_i)$, there is an approximate constraint on the second fundamental tensor II at point $\mathbf{s}$:

$$\mathrm{II} \begin{pmatrix} \mathbf{e} \cdot \mathbf{u} \\ \mathbf{e} \cdot \mathbf{v} \end{pmatrix} = \begin{pmatrix} (\mathbf{n}_i - \mathbf{n}) \cdot \mathbf{u} \\ (\mathbf{n}_i - \mathbf{n}) \cdot \mathbf{v} \end{pmatrix} \tag{4.3}$$

here $\mathbf{u}$ and $\mathbf{v}$ are two orthonormal basis vectors on the tangent plane at $\mathbf{s}$; naturally $\mathbf{n} = \mathbf{u} \times \mathbf{v}$, and $\mathbf{e}$ is the vector from $\mathbf{p}$ to $\mathbf{p}_i$. Then, let

$$\mathrm{II} = \begin{pmatrix} E & F \\ F & G \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} E \\ F \\ G \end{pmatrix} \tag{4.4}$$

From 4.3 and 4.4 we obtain two constraints on the value of $\mathbf{a}$:

$$\begin{pmatrix} \mathbf{e} \cdot \mathbf{u} & \mathbf{e} \cdot \mathbf{v} & 0 \\ 0 & \mathbf{e} \cdot \mathbf{u} & \mathbf{e} \cdot \mathbf{v} \end{pmatrix} \mathbf{a} = \begin{pmatrix} (\mathbf{n}_i - \mathbf{n}) \cdot \mathbf{u} \\ (\mathbf{n}_i - \mathbf{n}) \cdot \mathbf{v} \end{pmatrix} \tag{4.5}$$

We estimate the curvature tensor by combining the constraints in the least-square sense using the weights w_i depending on the distance between points $\mathbf{p}$ and $\mathbf{p}_i$ as follows:

$$w_i = e^{-d^2(\mathbf{p}, \mathbf{p}_i)/h^2} \tag{4.6}$$

After finding the second fundamental matrix, we compute its eigenvalues k_1 and k_2 (these are the principal curvatures), and the corresponding eigenvectors x_1 and x_2. Without loss of generality, we can assume that $k_1 > k_2$, and define the mean curvature and the Gaussian curvature of the surfel s as follows:

$$H = (k_1 + k_2)/2, \quad K = k_1 k_2.$$

a b c

Figure 4.5: Mean curvature distribution on the Stanford bunny model and a scanned surface. (a): sampled surface of Stanford Bunny. (b) a partial scan of Stanford Bunny. Green points have negative mean curvatures and red points have positive mean curvatures. The color distribution on the model and scans are consistent, which illustrates the robustness of our curvature computation procedure. Note that the model has a finer resolution than the scan. View (c) shows the major curvature axis direction on the scanned surface.

The principal curvature directions can be determined as:

$$\mathbf{y}_1 = x_{11}\mathbf{u} + x_{12}\mathbf{v}, \quad \mathbf{y}_2 = x_{21}\mathbf{u} + x_{22}\mathbf{v}.$$

An example of curvatures computed with the described method is given in Figure 4.5.

Table 4.3: Estimation of principal curvature direction at point s_i.

$s_i = (p_i, n_i) \in S$, N is a set of k neighbors of s_i
u is a reference direction on the tangent plane of s_i
$v = u \times s_i$
For every points $s_j = (p_j, n_j) \in N, j = 1 \ldots k$
$w = e^{-\lvert p_j - p_i \rvert^2 / d^2}$
Accumulate the constraint into the least-squared solution weighted by w.
EndFor
Use Least squares method to solve the second fundamental tensor at s II.
Use two eigenvalues of II as the principal curvatures
Use the eigenvectors to compute the principal curvature directions.

4.4.2 Sampled direction approach

The goal of using principal curvature directions in Equation 4.1 is to make the two directions on the tangent planes to be consistent. However, the computation of principal directions is sensitive to noise and surface re-sampling especially for near-umbilic surface regions. Instead, we can select a random direction $\mathbf{k}$ at a point $\mathbf{p} \in \mathcal{S}$, with $\mathbf{m} = \mathbf{n} \times \mathbf{k}$. To

find a direction correspondent to $\mathbf{k}$ on the other surface, we try 36 directions $\mathbf{k}'$ distributed evenly on the tangent plane of point $\mathbf{p}' \in \mathcal{M}$, and compute $\mathbf{m}' = \mathbf{n}' \times \mathbf{k}'$. If $\mathbf{p}$ and $\mathbf{p}'$ are corresponding points, $\mathbf{k}$ will be approximately consistent with one of 36 $\mathbf{k}'$.

4.4.3 Boundary detection

Scanned range surface data are often full of holes and ragged boundaries. We find that the normal and the principle curvature directions cannot be estimated reliably near the boundary, and therefore we perform a boundary detection step before surface matching for both the scan surface and the model point cloud. The points near the detected boundary are excluded from the matching process.

We decide if a point is on the boundary by comparing the relative positions between this point and its neighbors. Given a surfel $\mathbf{s} = (\mathbf{p}, \mathbf{n}) \in \mathcal{S}$ and a set of neighboring points $N_r(\mathbf{s}) \in \mathcal{S}$ that fall into a ball of small radius r, for every point $\mathbf{s}_{i,i=0\ldots k-1} = (\mathbf{p}_i, \mathbf{n}_i) \in N_r(\mathbf{s})$ we define $\overrightarrow{\mathbf{ss}_i} := \mathbf{p}_i - \mathbf{p}$ and project $\overrightarrow{\mathbf{ss}_i}$ onto the tangent plane at $\mathbf{s}$. The projected direction can be marked as $\mathbf{a}_{i,i=1\ldots k-1}$, and ordered in the counterclockwise order using the angle values $\angle \mathbf{a}_0\mathbf{a}_{i,i=0\ldots k-1}$. Then we check the angle between every pair of neighboring directions: $\phi_{i,i=0\ldots k-2} = \angle \mathbf{a}_i\mathbf{a}_{i+1}$, and if ϕ_i is larger than a threshold ϕ, we will mark s as a boundary points. In this chapter we use the threshold ϕ equal to two radians. In this chapter, the radius of the ball for the boundary detection is set to be five times the median distance between nearest-neighbor point pairs (this gives a measure of the sampling density over a given surface).

Surface points close to boundary points are called a near-boundary points. For every non-boundary surfel $\mathbf{s} \in \mathcal{S}$ for which any of its κ (*near-boundary* parameter) nearest neighbors is a boundary point, $\mathbf{s}$ is called a *near-boundary* point. Figure 4.6 shows an example of detecting boundary points and near boundary points for the Stanford bunny model and a scanned surface. For a surface $\mathcal{S}$, we denote the set of points near boundary to be $N_{\mathcal{S}}$ and the set of (inner) points that are not near the boundary to be $I_{\mathcal{S}}$.

We shall present the experimental evaluation of the influence of the boundary on the curvature estimation in the next section.

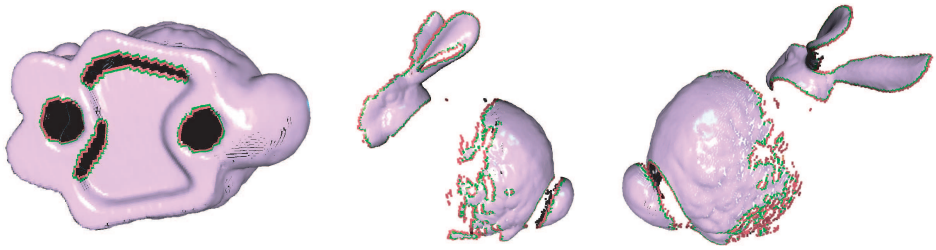


Figure 4.6: An example of the boundary and near-boundary points on bottom of Stanford bunny model (left) and one scanned surface from different views (center, right); Red points are boundary points, green points are near boundary points, and purple color shows the inner points

4.4.4 Evaluation of curvature estimation procedures

In this section, we present the experimental evaluation of our curvature estimation procedure for noisy point cloud data. We first compare the point cloud estimation with the corresponding mesh-based curvature estimation. Then, we perform a comparison between curvature quantities computed on the CAD model and the registered scans.

Comparison with mesh-based curvature estimation

We compared the performance of the curvature estimation method on the point cloud data for the Stanford bunny model with the original curvature estimation method of [71]. Table 4.4 shows the standard deviations between the curvature and principal curvature directions estimated for the internal and near-boundary vertices of the model separately, as being dependent on the size of the boundary parameter κ (near-boundary parameter as in section 4.4.3). The left two columns show the standard deviation of the mean curvature, and the right two columns show the corresponding data for the principal curvature direction in degrees. The curvature deviation is normalized by the mean curvature of the sphere whose radius is equal to the half of the diagonal of the bounding box for the bunny model. That is, if l_{diag} is the length of the bounding box diagonal, we can define H_{diag} as the mean curvature of a sphere with radius R = $\frac{l_{diag}}{2\sqrt{3}}$. We have $H_{diag} = \frac{1}{R}$, and the curvature error in

the table is normalized by that quantity. We see that the quality of the estimation deteriorates near the boundary. At the same time, excluding additional points near the boundary from matching may deteriorate matching results especially for coarse lower-quality scans.

In Table 4.4, as we can see, the standard deviation of inner points (δ_{IM} and $\delta_{\angle_{IM}}$) is smaller than their corresponding data for near-boundary points (δ_{NM} and $\delta_{\angle_{NM}}$). However, the change of the deviation become smaller with the increment of near-boundary parameter κ. In this chapter, we choose $\kappa = 8$

Table 4.4: Comparison with mesh-based curvature estimation

κ	δ_{IM}	δ_{NM}	$\delta_{\angle_{IM}}(^\circ)$	$\delta_{\angle_{NM}}(^\circ)$
2	1.0874	2.7044	14.8507	19.7460
4	0.9861	2.5178	14.7037	18.6613
6	0.8905	2.4097	14.5148	18.5039
8	0.8672	2.2590	14.3458	18.5386
10	0.8461	2.1360	14.3401	17.9558
12	0.8286	2.0268	14.2718	17.6719

Curvature comparison between the model and the scan

In order to obtain a reliable matching procedure, we need to have similar estimated curvatures and principal curvature directions from the model and from the scanned data. We perform this comparison on a scan registered with the ICP method onto the model of the Stanford bunny, so that they are given in the same coordinate system. Note that in this case, the reported error reflects both the difference in the measured data and the error of the estimation procedure; the resulting combined error should be indicative of the performance for the matching method.

For every surfel $\mathbf{s} = (\mathbf{p}, \mathbf{n}) \in \mathcal{S}$, we calculate its mean curvature $H_{\mathbf{s}}^{scan}$ and the maximal principal curvature direction $\mathbf{d}_{\mathbf{s}}^{scan}$ on the scan. We also estimate the mean curvature $H_{\mathbf{s}}^{model}$ and the maximal principal curvature direction $\mathbf{d}_{\mathbf{s}}^{model}$ at the same location using the model surface data. To estimate $H_{\mathbf{s}}^{model}, \mathbf{d}_{\mathbf{s}}^{model}$, we first find its k neighbors $N(\mathbf{s})$ from the model,

and estimate the new normal using these neighbors as: $\mathbf{n}_{\mathbf{s}}^{model} = \sum w_i \mathbf{n}'_{ii \in N(\mathbf{s})}$; Then we can estimate its curvature using the algorithm in Table 4.3 given $\mathbf{n}_{\mathbf{s}}^{model}$, $\mathbf{p}$ and $N(\mathbf{s})$.

After estimating the mean curvature and principal direction for surfel s, we compute $\Delta(H) := H_s^{scan} - H_s^{model}$ and $\Delta(\angle d) := cos^{-1}\left(\mathbf{d}_s^{scan} \cdot \mathbf{d}_s^{model}\right)$. Furthermore, we can normalize $\Delta(H)$ as described in the preceding section.

Table 4.5 shows the deviation of $\Delta(H)$ for inner points (δ_I) and near boundary points (δ_N). Table 4.6 shows the deviation of principal curvature directions $\Delta(\angle d)$ and normal vectors $\Delta(\angle n)$ for inner points ($\delta_{\angle(d)_I}, \delta_{\angle(n)_I}$) and boundary points ($\delta_{\angle(d)_N}, \delta_{\angle(n)_N}$). The integer near-boundary parameter κ control the number of points in the near-boundary set N_S on the scan as described in Section 4.4.3. The following formula is used to compute the standard deviation of δ_I:

$$\delta_I = \frac{1}{H_{diag}} \sqrt{\frac{1}{\#(I_S) - 1} \sum_{s \in I_s} (\Delta(H))^2}$$

where $\#(I_S)$ is the cardinality of set I_S. Similarly, σ_N shows the results for the set N_S. We can compute the other elements of Tables 4.5 and Table 4.6 in a similar way. We see that the match between estimated principal curvature directions is worse than the match between the normal directions. We also see that near the boundary the difference between the local basis directions deteriorates significantly. Therefore, the approximation introduced by a sampled directions approach is not worse than the errors resulting from principal curvature estimation procedures. This will also be confirmed by our evaluation of the actual matching algorithm in the next section.

Table 4.5: Standard deviations of mean curvatures

κ	δ_I	δ_N
2	1.3344	5.8029
4	1.1566	5.4162
6	1.0937	5.1158
8	1.0471	4.8291
10	0.9946	4.6701
12	0.9475	4.4987

Table 4.6: Standard deviations of angles (principal curvature direction and normal) (in degrees$^\circ$)

κ	$\delta_{\angle_I}$	$\delta_{\angle_{IS}}$	$\delta_{\angle_N}$	$\delta_{\angle_{NS}}$
2	21.5740	6.3939	39.7218	15.3090
4	21.2025	5.9968	37.9804	16.9588
6	20.9973	5.9231	37.7735	15.7693
8	20.8623	5.7816	36.7476	15.5321
10	20.7451	5.7597	36.0057	14.6237
12	20.6331	5.6241	35.2647	14.5438

4.5 Experimental evaluation of matching algorithms

This section includes results of experimental evaluation of our algorithms introduced in this chapter on scanned data.

4.5.1 Evaluating matching performance

In this section, we compare the feature-based and random sampling methods of selecting representative point sets discussed in Section 4.3, as well as strategies for fixing the tangent plane direction described as Section 4.4.1 and 4.4.2. For this comparison we use two data sets: the first dataset is the bunny model and 10 scans from Stanford repository. For this dataset the model contains 35,947 surfels and the average number of points on the scans is about 22K. The second dataset used for comparison is the three angel models and their 55 scans (these are coarser and contain about 5K surfels on average, also see Figure 4.7). The angel scans also contain more noise, and the angel models are reconstructed from scans as described in [37].

For random-based point selection, we vary the set of points for every trial, and perform 20 trials for every scan. This amounts to $55 \times 20 = 1100$ total trials for angel data set and $10 \times 20 = 200$ trials for the bunny data set. Figure 4.8 shows the results comparing with multi-scale features and random-based method, and the results with single scale features and random-based method. The vertical axis records the percentage (out of 1100 for the angel data set, and out of 200 for the bunny data set) of trials that succeeded to find a good

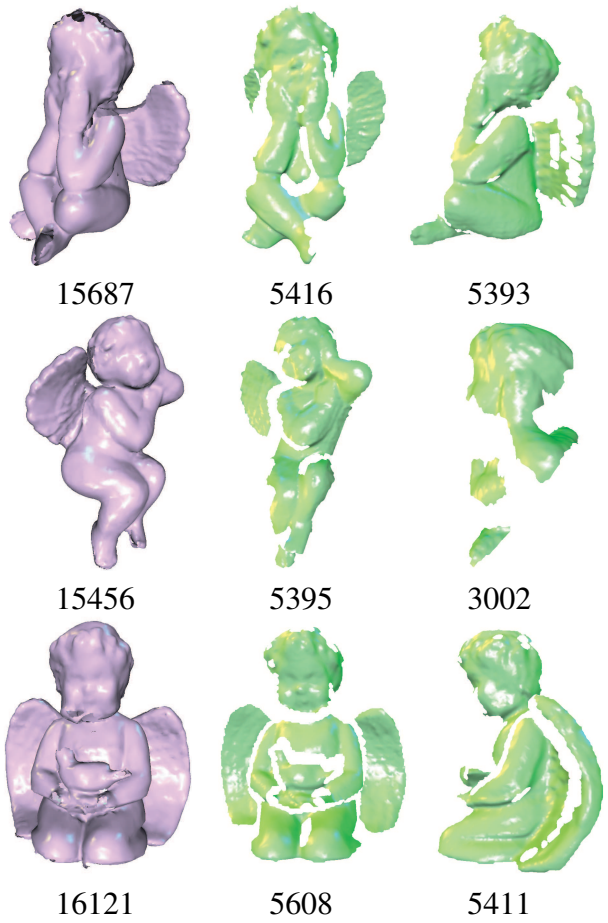


Figure 4.7: Three angel models (purple) and some of their scans (green); The numbers show the number of points on the surfaces

initial matching. For the feature-based method, the feature points are selected deterministically and the reported percentage again corresponds to the number of cases that matched successfully (out of 55 for the angels, and out of 10 for the bunny). The horizontal axis shows the percentage of the points selected as the representative sets on both the model and the scan. We see that feature points outperform the randomly selected points in these experiments, so that a smaller percentage of representative points can be used to do the initial rough alignment. When principal curvature direction used, as can be seen in Figure 4.8, for the Angel model the multi-scale feature-based algorithm can match all scans to their models if 1% of features are used from the scans and models. Random-based algorithm have 1 out 1100 tries failed if 2% of inner points are selected. For the Bunny model, with

0.3% of feature points selected, all scans can be matched to the model. If the points are selected randomly, 31 out of 200 tries failed to match if 0.3% points are selected, and all out of 200 tries are matched to the model if 0.5% points are selected. We have evaluated the performance of single scale feature point selection and random point selection when the sampled direction selection version of the algorithm was used, with the conclusion that the single scale features give more robust results for surface matching. Results in Figure 4.8 also show that sampled direction selection outperforms the principal curvature direction version based on the experiments with random point selection method.

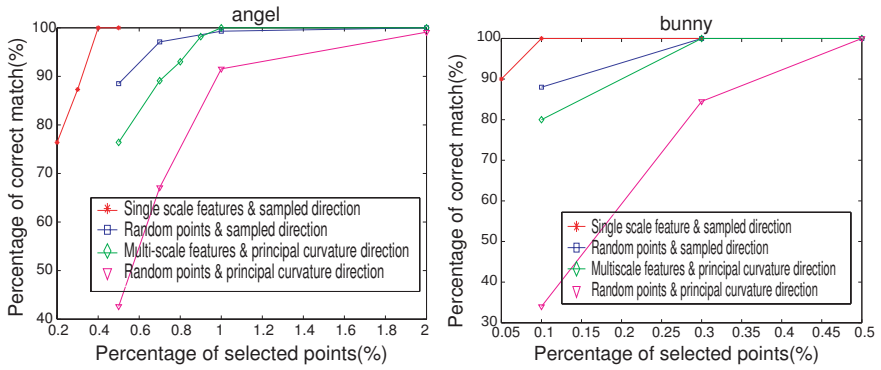


Figure 4.8: Comparison of feature-based matching and random-based matching approaches. The plot shows the percentage of correct matches for the scan-to-model alignment. See text for more details.

We have also used our multi-scale feature based method for initial alignment of 12 scans of the turbine blade model. We check the quality of the initial alignment as well as the quality of our approximate transformation verification procedure by computing the L_∞ distance between the model and each scan after the initial alignment (but before the ICP) and then after the ICP has aligned the models. Figure 4.9 shows the approximate distance used for verification (as in algorithm in Table 4.2), the L_∞ distances before ICP (for inner and boundary points), and L_∞ distances after ICP. The approximate sampling is the subset of inner points and provides a fairly good estimate on the match quality. The results of the approximate matching are shown in Figure 4.10

Figure 4.11 shows the results of surface matching for 4 models and 75 scans from Mian et. al. [52] using sampled directions and the single scale features proposed in section 4.3.3.

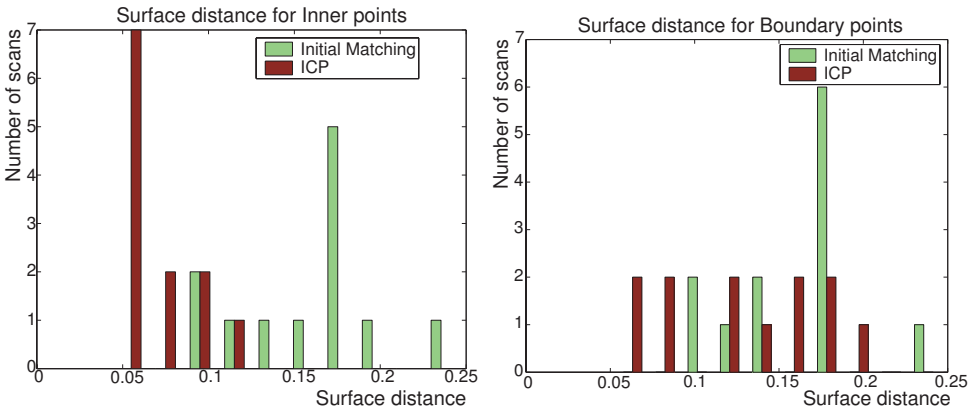


Figure 4.9: Histogram of surface distances between the Turbine blade model and 12 scans after matching. The distances for the inner points (top) is smaller than the distances for the boundaries (bottom). The distance after ICP algorithm (red) is smaller than the distance after initial matching (green), which is relative small compared to the diagonal of bounding box (8.8071) of the model.

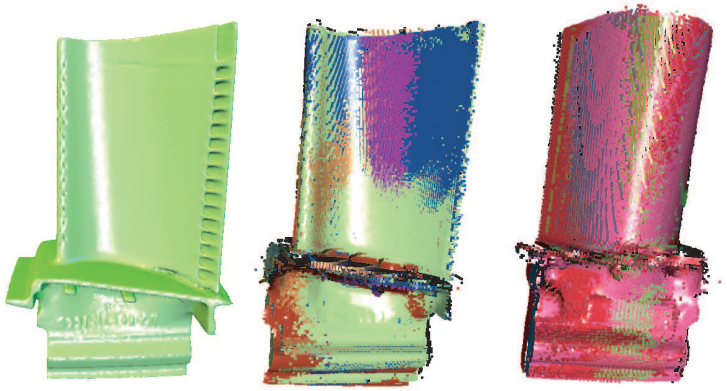


Figure 4.10: Matching result of the Turbine model. Left: the Turbine model. Center and right: 12 scans of the Turbine model after initial matching.

To save time on building scale representation and extracting feature, we resample the original surfels into smaller point clouds, which keep about 22% vertices from the original point clouds. We use the the point clouds with coarse resolution for the approximate matching and the original surfaces for ICP algorithm. The repeatability of features between coarse surfaces and original surfaces as described in Section 4.3.3 is shown in Table 4.7. For this dataset the total time of initializing the multi-scale features for all the models and scans (with principal curvature direction version) was about 58 minutes, and the total runtime of initial matching algorithm was about 40 minutes. With the new single scale features and sampled direction method, we improved the total initialization time to about 13.5 minutes and the total initial matching time (75 scans to their respective models) to about 14.5

minutes. All the computed matches were correct.

Chef Chicken Parasaurolophus T-rex

Figure 4.11: Results of surface matching for 4 complex models

Table 4.7: Repeatability of features of models in Figure 4.11

Model	Chef	Chicken	Parasaurolophus	T-rex
All features	557	255	323	307
Matched features	370	214	222	220
Percentage(%)	66.4	83.9	68.7	71.7

4.5.2 Comparison with two matched pairs alignment

The initial approximate alignment algorithm introduced in section 4.2 uses a single pair of corresponding points to compute the transformation. The algorithm can be easily modified to use two pairs of corresponding points for estimating the transformation matrix using both position and normal information as in [43]. The modified algorithm is shown in Table 4.8. If M points are selected from the scan and N points are selected from the model, the time complexity of the algorithm in Table 4.1 is $O(M \times N)$ and the time complexity of the algorithm in Table 4.8 is $O(M^2 \times N^2)$

We use the angel data set to verify the time complexity and robustness of these two matching algorithms for the case when 1% of representative points are randomly selected from both the scan and the model. As described before, we run 20 trials for every scan

(1100 trials overall). The total time for 1100 trials is 4.0 seconds for one-pair matching algorithm and 167 seconds for two-pair matching algorithm. However, the matching rate (percentage of correct alignment out of 1100 trials) for two-pair matching algorithm (70%) is lower than the rate for one-pair algorithm (91%).

Table 4.8: Initial surface matching process with two pairs of corresponding points

SurfaceMatching(S,M)

Initialize the return value T_{best};
Select a set of surfels $\Omega(M) \in I_M$;
Select a set of points $\Omega(S) \in I_S$;
Initialize Δ_{max};
For every surfel $s_{i1}, s_{i2} \in \Omega(S)$
    For pair of surfels $m_{j1}, m_{j2} \in \Omega(M)$
        If $abs(d(m_{j1}, m_{j2}) - d(s_{i1}, s_{i2})) > d_1$ continue;
        Compute $T = \left[\begin{array}{cc} R & t \end{array} \right]$ from m_{j1}, m_{j2}, s_{i1} and s_{i2} as in [43];
        Δ = Verify(T, Δ_{max});
        If $\Delta > \Delta_{max}$
            continue;
        Else
            Δ_{max} = Δ;
            $T_{best} = T$;
        EndIf
    EndFor
EndFor
Return T_{best};

4.5.3 Accelerating matching with simple signatures

Surface matching algorithms [39, 43, 27] often use surface signatures to establish correspondences between scans and models. In this section, we use mean curvature as a very simple signature to accelerate the matching algorithm. By using signatures, for every selected feature point from the scan, we select its k points from the model multi-scale feature set that have the nearest mean curvature values. Then, the correspondences are build from the point and its k neighbors instead of all the selected points from the model. We have tested the mean curvature signatures on the bunny data set, with the result that all 10 scans can be approximately matched to the model when $k = 40$, and the average (per-scan)

time for matching is reduced to be 3 seconds for the procedure using signatures versus 14 seconds of the plain matching procedure.

4.6 Summary

In this section, we have introduced a new scan-to-model matching procedure and explored its various enhancements, focusing on representative point selection. We have shown that the feature-based representative point selection results in a better matching performance than the random sampling of the surface.

CHAPTER V

OBJECT RECOGNITION

Object recognition is important in many practical applications of computer vision. The traditional techniques of object recognition from 2D images are sensitive to changes in illumination and shadowing. When the range input data is available, the recognition procedure does not have to deal with these problems. On the other hand, the presence of clutter, occlusion and change of viewpoint still remains a challenge, which can be alleviated by using a collection of locally defined surface descriptors.

The technique of representing 2D or 3D images by a collection of unordered features shows impressive performance with various applications [38, 53, 28, 43, 41]. In this chapter, we propose a new method to recognize object from range images using local surface descriptors: given several groups of classified range images, we consider the problem of labeling an unknown image with the most related group (or category) label. In this case, each object is represented by a group of range images, and every range image is represented by its set of features. Our method does not require any prior knowledge of the test images, and is invariant to 3D rigid transformation.

Most previous works recognize 3D objects using surface matching methods: unknown range surfaces are compared with the models in the database and recognized as the one with the smallest metric distance [7, 38, 52]. These methods require a global model, which needs an extra surface modeling process from multiple range images as in [52]. In this chapter, we propose an alternative solution using supervised learning which measures the similarity between range images using their feature sets.

Our process works as follows: for every range image, we first select a set of points which are distinctive and carry salient geometry information. Every salient point is combined with a surface descriptor (invariant to 3D trasformation and occlusion) defined on the local surface patch near the point. After detecting salient points and computing their local signatures, we represent the range images as a set of unordered surface descriptors. The pyramid match kernel function proposed by Grauman and Darrell [28] is used to measure the similarity between sets of unordered features (range images). Finally, given a set of n classes of labeled images, we learn n classifiers using the pairwise similarities between these images, where every classifier separates one class of images from all the others. An input image can then be run against these classifiers and recognized as the most related category. Our experiments on two datasets show high recognition rates.

The rest of this chapter is organized as follows: after reviewing the related work in the next section, we describe the salient points detection algorithm in Section 5.2. In Section 5.3, we show how to define the range signatures on single point and pair of points. Section 5.4 reviews the pyramid match kernel used to learn the classifiers. Section 5.5 shows some recognition results on two datasets.

5.1 Related works

Keypoints have been proven to be very useful for object recognition from 2D images [50]. In 3D domain, the detected keypoints must be invariant to 3D rigid transformation, which can be achieved, for instance, by considering points with high curvature values as proposed by Chen and Bhanu [17]. Several object recognition methods [38, 23, 52] use a set of randomly selected points to compute surface descriptors. In Chapter II, we describe methods to detect a set of salient points by building a scale space representation of the 3D surface, based on our original work in Li and Guskov [43]. The feature points were shown to be robust under some surface variations. In this chapter, we use a similar process to detect salient points; our modifications include the use of linear projection for smoothing and a

single-scale version of the multi-scale features [43].

Object recognition with local and global surface characteristics has been an active research topic in computer vision community, for a detailed survey, please refer to [12]. Recent works show that local surface descriptors are useful tools for object recognition when clutter and occlusion happens. One popular example is a spin image signature introduced by Johnson and Hebert [38], which is a 2D histogram defined at an oriented point location. Spin images and their modifications, spherical spin images by Correa et. al. [68] and 3D shape contexts by Frome et. al. [23] perform well on the problem of object recognition from sensed data. In Chapter III, we proposed normal-based point descriptors that reflect the normal variation of local patches. In this chapter, we compare the performance of the normal-based descriptor and spin images for the purpose of object recognition.

Though most of local surface descriptors are defined on single selected point [38, 68, 23, 43], Mian et al. [52] have recently proposed a tensor-based surface representation defined on pairs of oriented points. Their descriptors are high dimensional surface histograms (hundreds to thousand elements in one descriptor) that measure the variation of surface position. In this chapter, we use both the signatures defined at a single point and signatures defined on a pair of salient points. We compare their performance on object recognition using a face dataset in Section 5.5.2.

To measure the similarity of surfaces represented by a set of descriptors, a common technique [68, 38] is to search the nearest neighbors to find the closest points from the database for every input descriptor, and [52] use a voting process to find pairs of tensors with high overlap ratio. Correlation coefficient of these roughly matched descriptors are computed and only pairs of features with high correlation are chosen as candidate correspondences between surfaces. These methods are usually followed by a surface registration process for the recognition purpose. Grauman and Darrell [28] proposed a multi-resolution histogram to represent a set of descriptors. They introduced the pyramid kernel function that provides a partial matching mechanism to compute the similarity between two set of

unordered features. In our work, we use this kernel function to compute the similarity between input range images for object recognition without performing surface alignment. The content of this chapter appeared as a conference paper [44].

5.2 Salient points detection and feature pairs

In this section, we describe our approach to selecting a set of representative points for given 3D range images.

5.2.1 Salient points detection

In Section 2.2, we introduced a multi-scale feature detection algorithm, which produces approximately the same sets of salient points for two differently sampled models of the same 3D surface. The multi-scale features can be useful for matching surfaces with unknown uniform rescaling. In the object recognition from 3D range images, the scale of the objects are known, hence it makes sense to use a faster single-scale salient point detection procedure, that uses only two levels from the scale space of the surface. The basic to detect features has been presented in previous chapters, here we summarize the techniques briefly.

In order to detect salient points on surface $\mathcal{S} = \{(\mathbf{p}_i, \mathbf{n}_i)_{i \in I}\}$, a scale space representation of $\mathcal{S}$ is built by a projection operation that maps a 3D point with the normal $\mathbf{s} = (\mathbf{p}, \mathbf{n})$ onto the smoothed version of $\mathcal{S}$. To avoid the nonlinear optimization, we use a simple smoothing procedure. Given a point $\mathbf{s}$ and a scale $h > 0$, we define a residual operator

$$E_{[S,h]}(\mathbf{s}) := \sum_{i \in N_i^h} \theta_i^h(\mathbf{s})(\mathbf{s} - \mathbf{p}_i), \tag{5.1}$$

and a smoothed normal direction

$$\mathbf{n}_i^h := \sum_{i \in N_i^h} \theta_i^h(\mathbf{s})\mathbf{n}_i \tag{5.2}$$

where N_i^h is the a set of neighbors for point i, and θ_i^h are normalized Gaussian weights:

$$\theta_i^h(\mathbf{s}) := \frac{e^{-d^2(\mathbf{s},\mathbf{p}_i)/h^2}}{\sum_{j \in N_i^h} e^{-d^2(\mathbf{s},\mathbf{p}_j)/h^2}}.$$

Given a scale h, the smoothed version of $\mathcal{S}$ is computed by mapping every point $\mathbf{s}$ to the position:

$$\mathbf{p}_i^h := \mathbf{p} - E_{[S,h]}(\mathbf{s})\mathbf{n}_i^h$$

We build the scale space representation of $\mathcal{S}$ with two different scales h_0 and h_1 ($h_1 > h_0$). Then we proceed to compute the normal difference between these two different levels: $d(i) := \mathbf{n}_i^1 \cdot (\mathbf{p}_i^1 - \mathbf{p}_i^0), \forall i \in I$. For every point i on the normal difference, we call it to be a neighborhood maximum if:

$$d(i) > d(i') \qquad \forall i' \in N_i^h;$$

The neighborhood minimum can be detected analogously. The set of neighborhood maxima and minima are used as detected salient point sets. One example is given in Figure5.1.

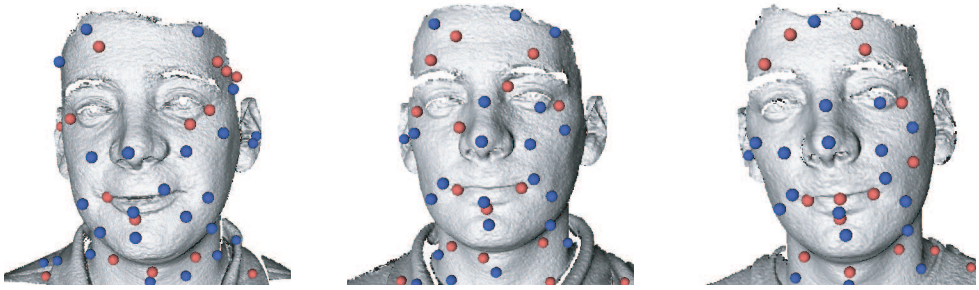


Figure 5.1: Features detected on range images of one person with different viewpoints and expressions. Maxima and minima are marked with different colors.

5.2.2 Feature pairs

Compared with single point, pair of points may provide more information like their distance and angle between their normal directions. In this chapter, we use two salient points (two local maxima or two local minima) as a feature pair if the distance between them is at

least $3h_1$ and at most $5h_1$. Figure 5.2 shows examples of feature pairs matched from two different 3D images. Section 5.3 will discuss how to compute the surface descriptor for a pair of points and Section 5.5.2 compares the recognition results using descriptor defined on single points and pair of points using the face dataset.

5.3 Signatures

The surfaces are compared by matching their signatures. We use two definition of signatures: spin images proposed by Johnson and Hebert[38] and the point signature proposed in this thesis. Since the latter reflects the variation of surface normals, we refer it as normal-based signatures (NBS). In this section, we will review how to compute the signature near a feature point briefly, and extend the definition for feature pairs.

5.3.1 Spin Image

Spin image is a popular surface descriptor and it was proven to be useful in both image registration and object recognition from clustered scenes. Spin image is defined on a local patch centered at a selected oriented point (a point with normal direction). We adapt the definition of spin images for salient points $\mathbf{x}$ with the support area of the spin image a cylinder of radius r and height r. By dividing the support area to $K \times L$ cells, we can compute the $K \times L$ dimension signature based on the histogram of the points in the support area. For efficient matching, we compress the high-dimension histograms as lower dimension vectors using Principal Component Analysis.

5.3.2 Normal-based signature

The definition for normal-based signatures is the same as the one for multi-scale features. Given a point s near the surface S with attributes: position $\mathbf{p}$, normal $\mathbf{n}$, and feature scale h, we compute its signature by first defining $N \times M$ points that sample a disc around point

s on its tangent plane:

$$x_{ij} := \mathbf{p} + \frac{2ih}{N}(cos(\frac{2\pi j}{M})\mathbf{u} + sin(\frac{2\pi j}{M})\mathbf{v})$$

where $i = 1, \ldots, N$, $j = 1 \ldots, M$, $\mathbf{u}$ and $\mathbf{v}$ are two orthogonal directions on the tangent plane. For every sampled point x, we compute a corresponding normal which is a weighted sum of the surface normals near x, using Equation 5.2. An $N \times M$ array of real numbers $R_{N\times M}$ can be determined by projecting the array of normals onto the tangent plane of s. The signature is computed by first applying a Discrete Cosine Transform to $R_{N\times M}$ in the N direction and then a Discrete Fourier Transform in the M direction. This results in a new array, and the absolute values of $n \times m$ elements from upper left corner are used in the signature vector. In this chapter, we use $N = 8$, $M = 36$, $n = 3$ and $m = 4$, which makes the dimension of signature $d = 12$. By definition, the NBS contains positive numbers. We note one limitation of NBS that it can not distinguish between local convex and concave spherical points. It is not a problem for surface matching in Chapter III because we match two sets of features separately. However, for the recognition problem, we pub all the features into the same set, thus we need to differentiate the minima and maxima.We solve this confusion by assigning negative sign to the signatures of local minima features.

5.3.3 Signature for pair of points

Given a pair of points together with their positions ($\mathbf{p}_1$, $\mathbf{p}_2$), normals ($\mathbf{n}_1$, $\mathbf{n}_2$) and scale s, we compute their average position $\mathbf{p} = 0.5(\mathbf{p}_1 + \mathbf{p}_2)$, a scale $d = 0.5|\mathbf{p}_1 - \mathbf{p}_2|$, and one direction $\mathbf{n}$ which is defined as following: let $\mathbf{t} = \mathbf{n}_1 \times \mathbf{n}_2$ and $\mathbf{m} = \overline{\mathbf{p}_1 - \mathbf{p}_2}$, then $\mathbf{n} := \mathbf{m} \times \mathbf{t}$. With the computed position $\mathbf{p}$, normal $\mathbf{n}$, and radius $\mathbf{d}$, the 12-dimensional NBS signature of Section 5.3.2 and the spin image can be defined for this pair of surface accordingly. Moreover, if we include two extra elements $d_p = d/s$ and $d_n = \mathbf{n}_1 \cdot \mathbf{n}_2$ in the NBS for pairs, we have a new 14 dimensional signature, which is noted as 14-d NBS. We compare the performance of 12-d NBS and 14-d NBS in Section 5.5.2.

5.4 Pyramid matching

In this section, we will describe the pyramid matching proposed by Grauman and Darrell [28]. In this chapter, it is used to measure the pairwise surface similarity with sets of features.

Let two sets of vectors $X_1 = \{f_1, \ldots, f_{n_1}\}$ and $X_2 = \{f_1, \ldots, f_{n_2}\}$ represent two 3D surfaces $\mathcal{S}_1$, $\mathcal{S}_2$, where f_i is a d-dimensional surface signature. The signature space is divided as multi-resolution histograms with $L+1$ levels, such that there are 2^l bins along each dimension at level $l \in \{0, \ldots, L\}$. This makes for a total of $b_l = 2^{dl}$ bins at level l.

As in [28], we define

$$\mathcal{I}_l(X_1, X_2) = \sum_{n=0}^{b_l} min(H_l(X_1^n), H_l(X_2^n))$$

Here $H_l(X)$ is the histogram over X at level l, and $H_l(X_i^n)$ is the number of vectors from X_i falling into bin n; $\mathcal{I}_l(X_1, X_2)$ is the number of vectors matched at level l (vectors are matched if they fall into the same bin), which also includes all of the vectors matched at level $l-1$. Then the *newly* matched features found at level l is $\mathcal{I}_l(X_1, X_2) - I_{l-1}(X_1, X_2)$. The pyramid matching kernel that measures the similarity of two surfaces can be calculated as:

$$\mathcal{K}(X_1, X_2) = \sum_{l=0}^{L} \frac{1}{2^l}(\mathcal{I}_l(X_1, X_2) - \mathcal{I}_{l-1}(X_1, X_2))$$

The weight coefficient in the kernel function is reciprocal to the size of grid, reflecting the similarity between the matched features at that level. This pyramid matching kernel provides an efficient way to compute the similarity between two sets of surface descriptors. Given a set of input images with N classes, we can compute the pairwise similarity by matching their descriptors using this kernel function, which results in a positive-definite symmetric matrix. In the next section, we will show how to use this symmetric matrix to learn the classifiers for object recognition and report the results on two data sets.

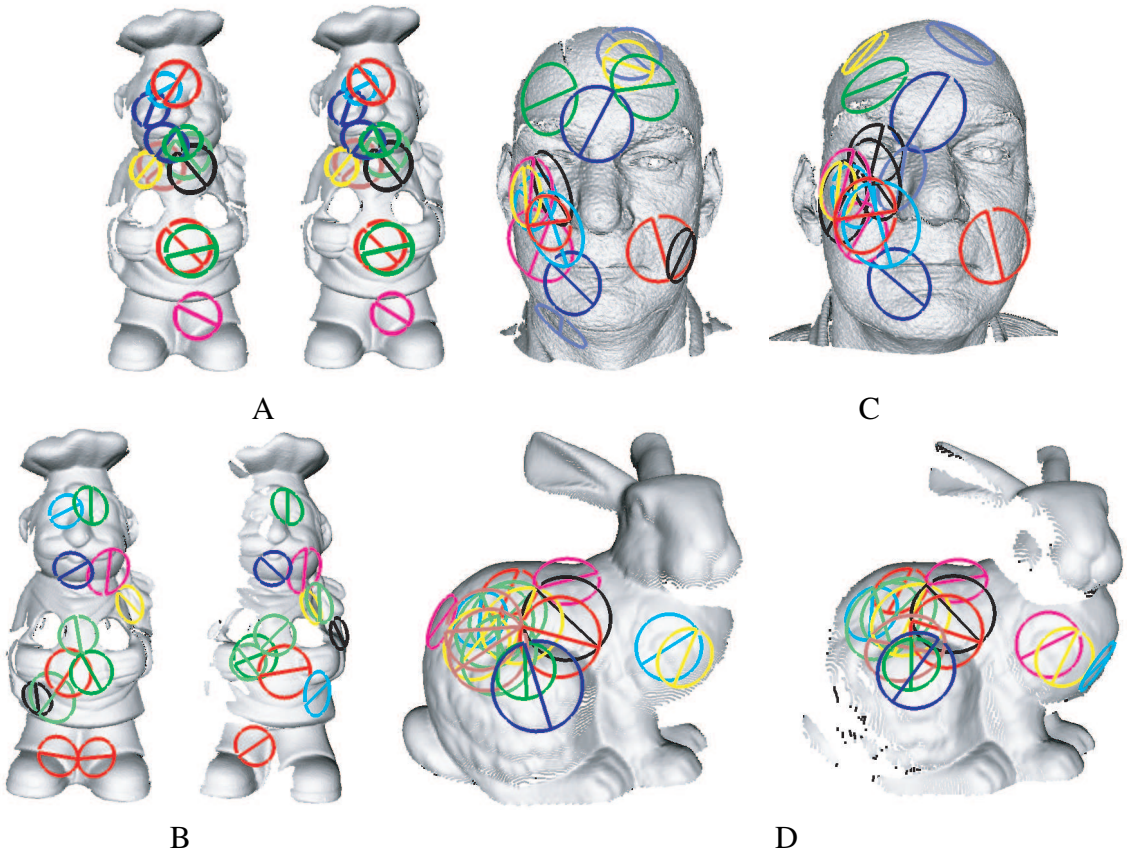


Figure 5.2: Correspondences of feature pairs between 4 pairs of surfaces with high confidence computed using the kernel function. Signatures falling into the same bin are marked with the same color. The colored line connects one pair of features. (A) shows the matched pairs between two different samplings of the same surface (the number of points on the right surface is 3.6 times of the left one); (B, D) shows the matched pairs between two surfaces of the same object from different viewpoints. (C) shows an example of face data with slightly different expressions and viewpoints.

5.5 Experiments

Support vector machine (SVM) is an effective tool for discriminating classes by finding an optimal separating hyperplane between classes. SVM can be trained on given kernel values of all pairs of training images. The kernel function described in Section 5.4 produces a positive-definite matrix which guarantees the convergence of SVM function. Given examples, we compute a kernel matrix that measures the pairwise similarity between examples and learn the classifiers from this matrix by SVM. For this purpose, we us an implementation of SVM [14] in our recognition experiments. For a set of labeled 3D range images of N classes, an SVM classifier is trained for each class to separate it from the rest. The

values between a test image and the training examples are computed using same the kernel function. After being compared with all classifiers, the test image is labeled as the category with the highest response.

Since the time complexity of the kernel function is linear to the dimension of signatures [28] and the high-dimensional spin images contain redundant information [38], we compress the spin images to compute the kernel matrix efficiently. For this purpose, we first find a set of k bases by applying PCA on the training set, and then reduce the dimension of spin images (from both training set and test set) to k-d vectors using this set of bases, similar to the approach of Johnson and Hebert [38]. In this chapter, we select $k = 12$ to make it equal to the dimension of normal-based signatures.

5.5.1 CAD model dataset

The dataset used in this section includes the range images of 30 models used by Hetzel et al.[34]. Every model has 258 range images which are acquired from different views distributed evenly over the whole viewing sphere. Figure 5.3 shows examples of range images for all the objects, please note that it includes 3 car models with similar shapes. All range images are available at [2].

As in [34], for every object 66 images are selected to train the classifiers and 192 images are used for testing. The numbers of feature points detected from these range images vary from 4 to 213, depending on the complexity of input shape. Every feature is assigned two signatures: a 12-d normal-based signature and a 100-d spin image (compressed to 12-d). We train two sets of classifiers using two signatures individually. The overall recognition rates are 98.02% using normal-based signature and 97.53% using spin images. The results are better than the 93% recognition rate reported by Hetzel et al. [34] using the same database. Please note that they reported the results of the best three matches while we use the best response.

The range images in this dataset are acquired synthetically with small difference of

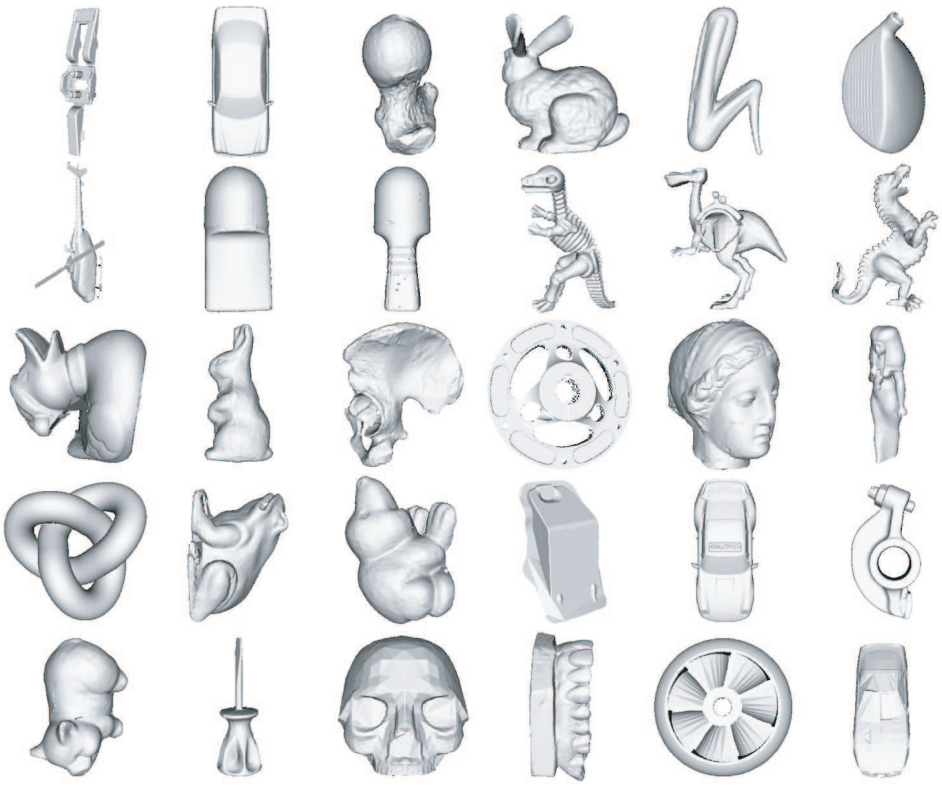


Figure 5.3: Examples of range images of 30 models used in Section 5.5.1.

viewpoints. In the next section we use a noisy dataset of different people's faces to test our recognition method.

5.5.2 Face dataset

Face recognition from 2D images or 3D surfaces is an active research topic in computer vision and pattern recognition [11]. In this section, we use the FRGC face database [66, 15], which includes the 3D range images of hundreds of different persons. From the gallery, we use the data of first 18 persons who have 20 or more usable range images. The overall number of usable images is 430. However, the images are acquired with different facial expressions, and the viewpoints are different as well. Figure 5.4 shows all the 20 images of one person used in this section. As you can see, there are holes due to self-occlusion and the limitations of the acquisition method. Also, clothes are captured in a lot of surfaces. We use the range images as they are, without preprocessing as reported in [15].

Figure 5.4: Example of all 20 faces of one person used in the face database.

The size of each raw 3D image is 640×480, with approximately 100k vertices per image marked as foreground. We use the triangles available from the original data to compute the surface normal for every vertex, which is the average of normals of the triangles adjacent to the vertex. The number of features detected from each range image is different due to the size of range images and the complexity of surfaces, e.g. the number of feature pairs for 196 images varies from 21 to 345. For these 430 images, we compute the 12-d NBS for single points as in Section 5.3.2, and normal-based signatures (12-d NBS and 14-d NBS) for pairs of points as in Section 5.3.3. We also compute 400 dimensional spin images for feature pairs and compress them to 12-d to compute the kernel matrix efficiently.

To learn the classifiers, we randomly select k images for every class as the training examples, and use the rest of images for testing. Figure 5.5 shows the recognition rate of the $(430 - 18k)$ test images with different k. To achieve a statistical estimation of recognition rate, we run 20 trials for every k. As one can see, with $k = 6$, the recognition rate of this 18 classes dataset is about up to 90% if normal-based signatures on feature

pairs are used to compute the kernel matrix. It is also clear that the pairwise signatures outperform the single point signatures significantly.

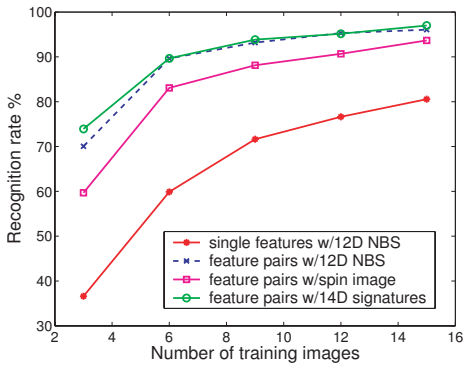


Figure 5.5: Face recognition rate of 18 persons' range images. The horizontal axis shows the number of training range images per person.

5.6 Summary

In this chapter, we propose a new method for object recognition using local features from range images. Our method does not require surface modeling as most previous methods. Since we use salient points and local signatures, it is invariant to 3D transformation and robust to occlusion. The recognition rate reported in this chapter is competitive with the results from previous work using same datasets.

CHAPTER VI

CONCLUSIONS AND FUTURE WORKS

This thesis proposes novel algorithms to extract a set of feature points from a given 3D point cloud and defined a new kind of point signature for these features. These features and their signatures give a new representation for 3D surfaces. In this thesis, we have demonstrated their success using applications like surface matching and object recognition.

6.1 Contributions

The major contribution of this thesis is a new representation for 3D surfaces: signatures for a set of selected salient features. The success of this representation is based the fact that the definition of features and signatures are local, invariant to view points and robust to the problem of some surface variations. We use point-based surface for the experiments in this document, and the proposed algorithms for feature detection and the feature's applications can be generalized to other surface representations, such as polygonal meshes with little effort.

Feature points and signatures are view independent The proposed features and signatures are rigid-transformation invariant, which is the fundamental requirement of features to be useful when comparing shapes. In this thesis, we detect features based on a smoothing procedure using the relative geometric information of surface points. Thus, the difference between smoothed surfaces along the normal direction stay unchanged under rigid geometrical transformation, which makes the features view-independent.

The normal-based signature is simple to construct. It is defined and based on a local coordinate centered at the point to which it is attached. To compute the signature for an feature, we sample points on the tangent plane and calculate the points' normals. Then we compute the signatures using the projection of these normals on the tangent plane, which stay the same with regard to rigid transformation. Thus, the signatures are also invariant to geometrical transformation too.

Feature points and signatures are scale-invariant The size of a shape could change for different usages, like being shrunk or enlarged by a scale. One property of the feature detection algorithms is to approximately select the same set of features for the same object, but of different sizes. Every detected feature point has a scale indicating the size of the salient geometry part detected as the feature near this point. For two surfaces of different sizes of the same object, the ratio of the scales of features detected from the same location should be consistent with the ratio of the size of the surfaces. Since the scale of the feature determines the local area that defines the signature, the signature is invariant to scale.

The advantage of scale-invariant of features is that we can compare surfaces of the same object cross-scale. We demonstrated their effectiveness for surface matching at different scales. We can extend the applications such as object recognition in future work.

Feature points and signatures can deal with partial surfaces The localization of the feature detection and definition of surface signatures makes the features appropriate for matching partial surfaces, or matching partial scans to their nominal models. We have explored applications in surface alignment and object recognition. All the scans we used in the experiments have cluttering, which prove the effectiveness of the features and signatures to compare scenes with significant occlusion parts, although it could fail for some cases with large occlusion portions, as shown in Figure 1.10.

One concern of our methods proposed in this thesis is that the feature detection algorithm employs a smoothing procedure, which makes is slow to process every vertex on the

surfaces. However, we only need to perform feature detection for a particular surface once, and then save the results for further use. One way to compromise for a dense surface is to detect features on its sparser version and use the same set of features to compute signatures on the denser surface.

6.2 Future work

In this thesis, we have shown how the proposed features and signatures can be applied in applications like surface alignment and object recognition. However, features and their signatures give a general surface representation. In this section, we discuss their possible applications other than the ones addressed in this document.

Shape similarity We have presented a simple method to detect shape symmetry at the beginning of this thesis in Section 1.5.2. It uses the localization property of the features and a simple process that matches signatures of a set of features detected from one surface. Since the features proposed in this thesis capture the invariant salient geometry properties on 3D surfaces, we can use them for other applications such as shape similarity

In Chapter V, we introduced the object recognition algorithm by comparing surfaces with sets of unordered features detected from these surfaces. For this application, we first compute the similarity between two surfaces using the local normal-based signatures proposed in this thesis. This would provide a good starting point to find shape similarity for partial scans.

For shape analysis applications, whole models are used most often. Since our features can work with any point signatures proposed in previous literature, we can combine our features with other global surface signatures that are invariant of rigid transformation (like Spin Images). Since we do not need to consider cluttering for global models, we expect more robust results.

Object classification The object recognition algorithm in Chapter V assumes that objects being processed are in the database, and the problem is in finding the best fit between the object and given models. Object classification is a more difficult problem because it requires recognition of unknown objects. One idea for classifying 3D objects could be similar as its 2D counterpart proposed by Grauman and Darrel [29]. Grauman and Darrell computed the 'distance' between objects by matching features using a pyramid kernel, and then used normalized cuts [77] to partition the objects into specified number of groups.

Instead using a set of unordered features, one alternative is to combine the unordered features with their spatial information. For this purpose, one object will be partitioned into several cubes. A pyramid histogram proposed by Grauman and Darrell in [28] will be constructed for each cube, and the shapes will be compared using each single cube from different surfaces. Using this technique, we expect to have new method for object recognition and classification from a clustered scene.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] The stanford computer graphics laboratory. The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep/.

[2] Stuttgart range image database. The University Of Stuttgart, 2001. http://range.informatik.uni-stuttgart.de/htdocs/html/.

[3] A. Adamson and M. Alexa. Approximating bounded, non-orientable surfaces from points. In *SMI*, pages 243–252, 2004.

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE TVCG*, 9(1):3–15, 2003.

[5] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, 2004.

[6] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[7] A. P. Ashbrook, R. B. Fisher, C. Robertson, and N. Werghi. Finding surface correspondance for object recognition and registration using pairwise geometric histograms. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II*, pages 674–686, London, UK, 1998. Springer-Verlag.

[8] C. Baracchini, A. Brogi, M. Callieri, L. Capitani, P. Cignoni, A. Fasano, C. Montani, C. Nenci, R. P. Novello, P. Pingi, E. Ponchio, and R. Scopigno. Digital reconstruction of the Arrigo VII funerary complex. In Y. Chrysanthou, K. Cain, N. Silberman, and F. Niccolucci, editors, *AST 2004: The 5th International Symposium on Virtual Reality, Archaeology and Inteligent Cultural Heritage*, pages 145–154, 2004.

[9] A. Baumberg. Reliable feature matching across widely separated views. *cvpr*, 01:1774, 2000.

[10] F. B. Blais. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13:231–240, 2004.

[11] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Three-dimensional face recognition. *Int. J. Comput. Vision*, 64(1):5–30, 2005.

[12] R. J. Campbell and P. J. Flynn. A survey of free-form object representation and recognition techniques. *Comput. Vis. Image Underst.*, 81(2):166–210, 2001.

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

[14] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[15] K. I. Chang, K. W. Bowyer, and P. J. Flynn. An evaluation of multimodal 2d+3d face biometrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(4):619–624, 2005.

[16] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng. Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. *IEEE Trans. PAMI*, 21(11):1229–1234, 1999.

[17] H. Chen and B. Bhanu. 3d free-form object recognition in range images using local surface patches. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3*, pages 136–139, Washington, DC, USA, 2004. IEEE Computer Society.

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

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

[20] C. S. Chua and R. Jarvis. Point signatures: A new representation for 3d object recognition. *Int. J. Comput. Vision*, 25(1):63–85, 1997.

[21] H. Darbandi, M. Ito, and J. Little. Surface signature-based method for modeling and recognizing free-form objects. In *International Symposium on Visual Computing*, pages II: 447–458, 2007.

[22] T. K. Dey and J. Sun. An adaptive mls surface for reconstruction with guarantees. In *Symposium on Geometry Processing*, pages 43–52, 2005.

[23] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proc. of ECCV*, pages 224–237, 2004.

[24] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25(1):130–150, 2006.

[25] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH*, pages 209–216, 1997.

[26] T. Gatzke and C. Grimm. Feature detection using curvature maps and the min-cut/max-flow algorithm. In M.-S. Kim and K. Shimada, editors, *GMP*, volume 4077 of *Lecture Notes in Computer Science*, pages 578–584. Springer, 2006.

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

[28] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1458–1465, Washington, DC, USA, 2005. IEEE Computer Society.

[29] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 19–25, Washington, DC, USA, 2006. IEEE Computer Society.

[30] J. P. Grossman and W. J. Dally. Point sample rendering. In *Rendering Techniques*, pages 181–192, 1998.

[31] S. Gumhold, X. Wang, and R. MacLeod. Feature extraction from point clouds. In *Proceedings of the 10^{th} IMR*, pages 293–305, Oct. 2001.

[32] R. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, 22(10):28–38, 1983.

[33] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *4th ALVEY Vision Conference*, pages 147–151, 1988.

[34] G. Hetzel, B. Leibe, P. Levi, and B. Schiele. 3d object recognition from range images using local feature histograms. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 2, pages 394–399, 2001.

[35] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[36] B. Horn, H. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am.*, 5:1127–1135, 1988.

[37] D. Huber and M. Hebert. Fully automatic registration of multiple 3D data sets. *Image and Vision Computing*, 21(7):637–650, July 2003.

[38] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433 – 449, May 1999.

[39] A. E. Johnson and M. Hebert. Surface registration by matching oriented points. In *Proceedings 3DIM*. IEEE Computer Society, 1997.

[40] C. Lange and K. Polthier. Anisotropic fairing of point sets. *Special of CAGD 2005*, 2005.

[41] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[42] X. Li, J. Barhak, I. Guskov, and G. W. Blake. Automatic registration for inspection of complex shapes. *Virtual and Physical Prototyping*, 2, 2007.

[43] X. Li and I. Guskov. Multiscale features for approximate alignment of point-based surfaces. In *Symposium on Geometry Processing*, pages 217–226, 2005.

[44] X. Li and I. Guskov. 3d object recognition from range images using pyramid matching. pages 1–6, 2007.

[45] X. Li, I. Guskov, and J. Barhak. Robust alignment of multi-view range data to cad model. In *IEEE International Conference on Shape Modeling and Applications*, pages 98–107, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[46] X. Li, I. Guskov, and J. Barhak. Feature-based alignment of range scan data to cad mdoel. *International Journal of Shape Modeling(IJSM)*, 13, 2007.

[47] T. Lindeberg. *Scale-Space Theory In Computer Vision*. Kluwer, Dordrecht, 1994.

[48] K. Lindsay and R. Krasny. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *J. Comput. Phys.*, 172:879–907, 2001.

[49] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, The University of Utah, August 1987.

[50] D. G. Lowe. Object recognition from local scale-invariant features. pages 1150–1157, 1999.

[51] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[52] A. S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1584–1601, 2006.

[53] A. S. Mian, M. Bennamoun, and R. A. Owens. Matching tensors for pose invariant automatic 3d face recognition. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, page 120, Washington, DC, USA, 2005. IEEE Computer Society.

[54] A. S. Mian, M. Bennamoun, and R. A. Owens. A novel representation and feature matching algorithm for automatic pairwise registration of range images. *Int. J. Comput. Vision*, 66(1):19–40, 2006.

[55] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[56] N. J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328, New York, NY, USA, 2003. ACM.

[57] H. Moravec. Rover visual obstacle avoidance. In *proceedings of the seventh International Joint Conference on Artificial Intelligence*, pages 785–790, August 1981.

[58] p. w. MORONI GIOVANNI, giusarma s. Inaccuracy prediction due to six-point locating principle. In *Proceedings of the 4th CIPR International Seminar on Intelligent Computation in Manufacturing Engineering (ICME 2004)*, Sorrento, Italy, 2004.

[59] S. Pankanti, C. Dorai, and A. K. Jain. Robust feature detection for 3d object recognition. In *In Proceedings, SPIE Conference on Geometric Methods In Computer Vision II (Vol. 2031*, pages 366–377, 1993.

[60] M. Pauly and M. Gross. Spectral processing of point-sampled geometry. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 379–386, New York, NY, USA, 2001. ACM.

[61] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

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

[63] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[64] M. Pauly, L. P. Kobbelt, and M. Gross. Point-based multiscale surface representation. *ACM Trans. Graph.*, 25(2):177–193, 2006.

[65] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00*, pages 335–342, 2000.

[66] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 947–954, Washington, DC, USA, 2005. IEEE Computer Society.

[67] L. R. Prieto F., Redarce T. and B. P. An automated inspection system. volume 19, London, U.K., June 2002. Springer-Verlag.

[68] S. Ruiz-Correa, L. G. Shapiro, and M. Meila. A new signature-based method for efficient 3-d object recognition. In *CVPR (1)*, pages 769–776, 2001.

[69] S. Rusinkiewicz. trimesh. http://www.cs.princeton.edu/gfx/proj/trimesh2/.

[70] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, 2004.

[71] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proc. 3DIM 2001.*, 2001.

[72] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[73] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[74] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

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

[76] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph.*, 23(3):896–904, 2004.

[77] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, 2000.

[78] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[79] P. Shilane and T. Funkhouser. Distinctive regions of 3D surfaces. *ACM Transactions on Graphics*, 26(2):Article 7, June 2007.

[80] S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995.

[81] F. Stein and G. Medioni. Structural indexing: Efficient 3-D object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):125–145, 1992.

[82] Y. Sun and M. A. Abidi. Surface matching by 3d point08s fingerprint. *iccv*, 02:263, 2001.

[83] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902, Washington, DC, USA, 1995. IEEE Computer Society.

[84] A. P. Witkin. Scale-space filtering. In *Proc. of 8th IJCAI*, pages 1019–1022, 1983.

[85] J. V. Wyngaerd and L. V. Gool. Automatic crude patch registration: toward automatic 3d model building. *Computer Vision Image Understanding*, 87(1-3):8–26, 2002.

[86] S. M. Yamany and A. A. Farag. Surfacing signatures: An orientation independent free-form surface representation scheme for the purpose of objects registration and matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1105–1120, 2002.

[87] D. Ziou and S. Tabbone. Edge detection techniques: an overview. *International Journal on Pattern Recognition and Image Analysis*, 8(4):537–559, 1998.

[88] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 322–329, New York, NY, USA, 2002. ACM Press.