MULTISCALE FEATURE EXTRACTION AND MATCHING WITH APPLICATIONS TO 3D FACE RECOGNITION AND 2D SHAPE WARPING

by

Hadi Fadaifard

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy,

The City University of New York

2011

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Chair of Examining Committee: George Wolberg

Signature_____

Date_____

Executive Officer: Theodore Brown

Signature_____

Date_____

Supervisory Committee:

George Wolberg

_____

Robert M. Haralick

_____

Ioannis Stamos

_____

Jie Wei

_____

H. Quynh Dinh

_____

THE CITY UNIVERSITY OF NEW YORK

Abstract

MULTISCALE FEATURE EXTRACTION AND MATCHING WITH APPLICATIONS TO 3D FACE RECOGNITION AND 2D SHAPE WARPING

by

Hadi Fadaifard

Advisor: Professor George Wolberg

Shape matching is defined as the process of computing a dissimilarity measure between shapes. Partial 3D shape matching refers to a more difficult subproblem that deals with measuring the dissimilarity between partial regions of 3D objects. Despite a great deal of attention drawn to 3D shape matching in the fields of computer vision and computer graphics, partial shape matching applied to objects of arbitrary scale remains a difficult problem.

This work addresses the problem of partial 3D shape matching with no assumptions about the scale factors of the input objects. We introduce a multiscale feature extraction and matching technique that employs a new scale-space based representation for 3D surfaces. The representation is shown to be insensitive to noise, computationally efficient, and capable of automatic scale selection. Applications of the proposed representation are presented for automatic 3D surface registration, face detection, and face recognition. Test results involving two well-known 3D face datasets consisting of several thousand scanned human faces demonstrate that the performance of our recognition system is superior over competing methods.

Estimating differential surface attributes, such as normals and curvatures, plays an important role in the performance of 3D matching systems. Noise in the data, however, poses the main challenge in estimating these attributes. Surface reconstruction methods, such as

Moving Least Squares (MLS), help in minimizing the effects of noise. In this work, we also review the MLS approach for surface reconstruction, and show how the input noise affects the estimated differential attributes of the surface. We demonstrate how these results, together with statistical hypothesis testing, may be used to determine the smallest neighborhood size needed to estimate surface attributes.

MLS reconstruction and the discrete Laplace-Beltrami operator are well-known geometric tools that have a wide range of applications. In addition to their prominent use in our 3D work, we describe a novel use of these tools in a 2D shape deformation system for retargeting garments among arbitrary poses.

Acknowledgments

My sincere thanks to my advisor George Wolberg for his continuous support during my stay at the City University of New York. Thanks for being a fair critic and a great mentor.

I would like to thank Robert Haralick for his technical and inspirational support. I found our research meetings truly rewarding. Thanks for tolerating my stubbornness.

I would like to thank Florian Lengyel for being a great boss and a good friend. Thanks for introducing me to Christian Marks and his sophisticated world of mathematics, philosophy, and physics.

I want to thank the other members of the Imaging Group at CCNY: Siavash Zokai and Gene Yu. I really enjoyed our talks. I also want to thank Ali Assarpour, Dustin Mulcahey, Alexander Rapp, Yi Feng, and Yuqing Tang, with whom I shared an office at the Graduate Center, and worked together on the computational cluster. Thanks for being good comrades!

Finally, and most importantly, my sincere thanks to my parents for being my biggest fans. Mom, Dad, you are my guide, inspiration, and calm. Thanks for all your sacrifices and support.

To my dear mother, Haydeh

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"The beginning is the most important part of the work."

–Plato

The widespread use of 3D data in medicine, engineering, and entertainment has drawn great interest in novel methods to represent, store, search, and process 3D data efficiently. The data represents models obtained from real-world objects using 3D acquisition devices, as well as models created manually with 3D modeling tools. Automating the processing and searching of this data gives rise to a need for *3D shape matching* techniques. 3D shape matching is defined as the process of determining the amount of dissimilarity between 3D shapes [124]. In this work, we study the problem of 3D shape matching and some of its applications. The main applications of 3D shape matching are:

- **3D shape retrieval.** The explosive growth of 3D data has led to the creation of various 3D libraries. For example, Google's 3D warehouse [41] is a large database, which contains 3D models of buildings created by users of Google SketchUp. One of the main difficulties with maintaining such a library is having an efficient way of searching it. The most common ways of searching 3D databases are *query by keywords* and *query by example*. In query by keywords, the user enters a set of words and the system returns the models most relevant to the keywords. In query by

example, the user uploads a 3D model and the system returns similar models in the database; *e.g.*, Princeton's 3D Model search engine [104] allows users to search their database by uploading 3D models or drawing 2D sketches of the model they seek. The need for a good 3D matching technique is clear in query by example, as the query is done entirely using shape information. However, it has also been shown that keyword-based retrieval systems can be augmented with 3D shape matching methods to improve the performance of the search engine [40].

- **3D shape registration.** In order to create precise 3D models from physical objects, active acquisition devices such as 3D laser scanners are generally employed. Due to the limited field of view of these devices, multiple scans of the same object from different views are needed to obtain a complete model. The task of bringing these scans into the same coordinate system and aligning them is known as 3D shape registration. Automatic 3D registration systems make use of shape matching techniques to automatically establish correspondences between different surfaces in order to align them. An interesting application, which employs surface registration is the automatic reassembling of broken objects [45].

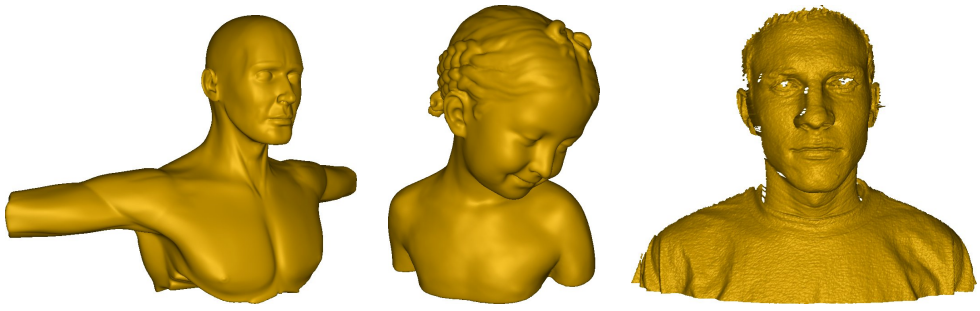
- **3D shape recognition.** Shape recognition refers to the task of identifying the locations and orientations of all instances of a set of previously-known shapes (objects) in a scene. The scenes, which are generally obtained using 3D scanners, are cluttered and contain holes due to occlusions. An automated recognition system needs to employ shape matching techniques to establish correspondences between parts of the scene and a set of models in a known database. One of the applications of shape recognition is in surveillance.

- **3D shape classification.** Given a 3D database containing disjoint sets of models with known classes, the problem of determining to which class a query model belongs is known as shape (object) classification. Shape classification may be required on its

own or it may be used as an initial step in a recognition or retrieval system [30]. Again, the need for 3D shape matching arises in automatic shape classification, since the only information provided to the system is in the form of 3D models.

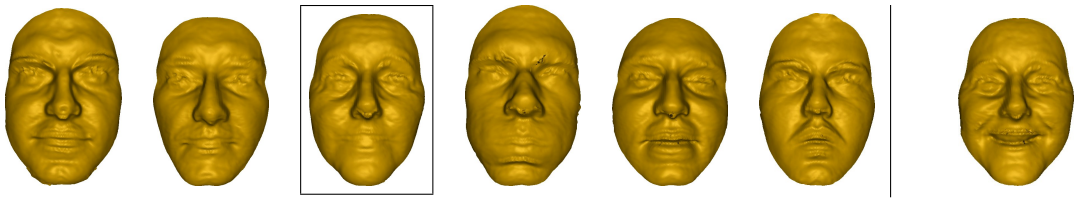
In this work, we address three shape matching problems: automatic surface registration, 3D face detection, and 3D face recognition. In automatic face detection, the objective is to detect the location, orientation, and scale of the human face region on an input 3D model. Such a system needs be versatile enough to handle variations between different faces it may encounter (*e.g.*, compare the models in Fig. 1.1). In 3D face recognition, given a database of 3D human faces with known identities (classes) and a query face model, the objective is to find the individual (class) in the database whose face matches the query model (*e.g.*, see Fig. 1.2). In such a problem, the system must be able to distinguish between minor differences in the surfaces, while at the same time be robust against the presence of expressions in the faces. Note that based on the definitions given above, the problem of automatic face detection falls in the category of shape recognition, while face recognition falls in the category of shape classification.

The only 3D shapes studied in this work are *3D surfaces*. In Sec. 3.2, we provide a list of most common surface representations used in computer graphics and vision. However, in this thesis, we only consider the two main representations produced by 3D acquisition devices, namely, *point clouds* and *polygonal meshes*. Point clouds and polygonal meshes



(a) (b) (c)

Figure 1.1: Possible inputs to a 3D face detection system; the models may have arbitrary scale.



(a) training set (gallery) (b) query (probe)

Figure 1.2: Examples of 3D faces used in our 3D face recognition system; the boxed face in (a) indicates the correct match for the query model shown in (b)

are formally defined and used in Chapter 4 and Chapter 5, respectively. In shape matching, input surface representations such as polygonal meshes are generally converted into intermediate representations, which are more suitable for the intended matching applications. This intermediate representation must be discriminative enough to be able to distinguish between different surface types the system may encounter. At the same time, the representation must be stable against surface perturbations and differences in surface topology (*e.g.*, due to holes in the surfaces). These somewhat contracting requirements make design of shape matching systems challenging and application-dependent. We refer to surface representations specifically used for shape matching as *shape descriptors*. In addition, if these shape descriptors are represented numerically and can be stored in the form of a vector, they are referred to as *feature vectors*.

In Chapter 2, we present and compare a large number of feature-based shape representations and matching approaches, which may be used in the abovementioned problems. As we will see, the majority of the approaches can be categorized into different classes, based on the following criteria:

- **Shape descriptor type.** Shape descriptors may be defined using various surface properties. The main examples are:
  - *Surface frequency content*; *e.g.*, Spherical harmonics [56] and 3D Zernike moments [95].
  - *Statistical surface properties*; *e.g.*, shape distributions [97], where histograms of surface properties such as angles and distances between arbitrary points on

surfaces are computed.

  - *Differential surface attributes*; *e.g.*, surface mean or Gaussian curvatures, Laplacian of curvature, *etc*.

- **Method used to compute shape descriptor.** Surface properties may be computed using various techniques. For example, when surface descriptors are defined in terms of differential surface attributes such normals, and mean/Gaussian curvatures, the performance of the system is heavily influenced by how these entities are estimated; as the differential order of these attributes increases, the sensitivity of the estimated values to noise increases.

- **Dissimilarity measure used to compare shapes.** The similarity/dissimilarity measure used to compare shapes also influences the performance of the matching systems. For example, in Chapter 6, we show that the performance of our proposed face recognition system is improved when the L_1 norm is used instead the L_2 norm.

The choice of the approach used in a shape matching problem is heavily dependent on the target application. For example, in 3D face detection, the shape matching system must be able to distinguish between *different classes* of surfaces, while in 3D face recognition, the system must be able to distinguish between different surfaces from *the same class*, namely human faces. In the latter problem, the employed surface representation must have a high discriminative power. This means the representation should be capable of differentiating between minor differences in the input surfaces; higher discriminative power, in turn, means higher sensitivity to noise in the input data. Therefore, in such an application, the shape matching algorithm must employ a *multiscale* representation that is capable of adjusting its discriminative power based on the input to the system. This issue is investigated in depth in Chapter 6.

As previously mentioned, one of the main problems faced by matching algorithms is the noise in the input data. This noise may be topological (*e.g.*, due to missing or extraneous

data) or non-topological, which generally presents itself in the form of uncertainties in the observed values. Only non-topological noise is considered in this work and, unless stated otherwise, it is simply referred to as noise. The noise in the input data is propagated to the later stages of computations in the matching algorithms. The two most common approaches for dealing with noise are: 1. smoothing the data and 2. making the computations robust to noise. We discuss both approaches in more detail in Chapters 3, 4, and 5.

In Chapter 4, we review the Moving Least Squares (MLS) surface reconstruction technique, and show how the noise in the input data is propagated in the various stages of the computation, up to the surface mean and Gaussian curvatures estimations. We employ statistical hypothesis testing to use the results in deciding on the minimum neighborhood size needed when estimating surface normals and curvatures.

In Chapter 5, we propose a new scale-space based surface representation, which we refer to as the Curvature Scale-Space 3D (CS3). We show that our representation is insensitive to surface noise, computationally efficient, and capable of automatic scale selection. The concept of automatic scale selection, which enables our systems to handle input surfaces with arbitrary scale, is explained in Chapters 3 and 5. We show applications of our multiscale representation in automatic surface registration, 3D face detection and recognition. In Chapter 6, we compare the performance of our face recognition system with state-of-the-art recognition systems, and show its superior performance.

Two main tools used throughout this work are the MLS reconstruction technique and the discrete Laplace-Beltrami operator. Their use, however, is not limited to 3D surface reconstruction and matching applications. In Chapter 7, we show an additional application for these in a 2D shape deformation system for warping garments among arbitrary poses. One of the stages of the approach discussed in Chapter 7 requires automatic extraction of control points on the boundaries of 2D shapes. Again, we employ a scale-space based method for this task.

1.1 Overview of Thesis

1.1.1 Contributions

In this dissertation, we study the problem of 3D shape matching and the main difficulties associated with it. We explore how noise and other variations in the input affect the performance of matching algorithms, and show how their effects can be minimized. The main contributions of this work are as follows:

- We provide a thorough survey and comparison of recent work on feature-based geometric shape matching, with an emphasis on partial shape matching techniques.
- We investigate how noise in the input surfaces is propagated when estimating differential surface attributes. Additionally, we show how statistical hypothesis testing can be used to determine the scale associated with various differential surface operators.
- We present an efficient extension of the scale-space representation of signals in $\mathbb{R}^n$ to 3D surfaces. Our representation is capable of automatic scale selection, and can be used to efficiently and robustly extract feature points/lines on 3D surfaces.
- We show how our proposed scale-space based representation for 3D surfaces can be used in automatic surface registration and face detection. In addition, we demonstrate the discriminative power of our representation by employing it in a 3D face recognition system, which outperforms other state-of-the-art 3D face recognition systems.
- Finally, using some of the tools used for 3D shape matching, we present a new 2D deformation system for warping 2D garments among new target poses.

1.1.2 Outline

The remainder of this thesis is organized as follows:

- In Chapter 2, we review and compare the current literature on feature-based partial and global shape matching techniques.

- In Chapter 3, we briefly describe the scale-space theory for signals in $\mathbb{R}^n$. We then review the current extensions of the theory for 3D surfaces and discuss the pros and cons of each approach.

- In Chapter 4, we discuss surface reconstruction and error propagation on point-sampled surfaces. Additionally, we show how statistical hypothesis testing may be used to determine the scale (neighborhood size) of differential operators on the reconstructed surfaces.

- In Chapter 5, we present our proposed scale-space based representation for 3D surfaces. We show how the representation can be used to extract the locations of geometrically meaningful feature points and lines in an efficient manner. We demonstrate the automatic scale selection capabilities of our representation and apply the results to 3D surface registration.

- In Chapter 6, we employ our scale-space representation for 3D surfaces to automatically identify the location, orientation, and scale of human faces on 3D models with arbitrary scale. We use the results from our automatic face detection procedure in a 3D face recognition system, which again employs our multiscale representation to construct the feature vectors used for recognition. We compare the performance of our 3D face recognition system with other competing methods by running extensive tests on two well-known 3D face databases.

- In Chapter 7, we present a 2D deformation system for retargeting garments among arbitrary poses.

- In Chapter 8, we present a summary of our work and directions for future research.

Chapter 2

Survey of Feature-Based 3D Shape Matching Methods

"If I have seen a little further, it is by standing on the shoulders of giants."

–Isaac Newton

2.1 Overview

In this chapter, we provide a survey and comparison of feature-based 3D shape matching techniques in computer vision and computer graphics. Specifically, we focus on local shape descriptors and their applications in partial shape matching. We also investigate the notions of feature *saliency* and *distinctiveness* and show how they can be used to reduce a surface to a set of representative points or regions that best describe the main properties of the underlying surface.

As mentioned in the introduction, 3D shape matching refers to the process of determining the amount of dissimilarity between 3D shapes [124]. Shape retrieval remains the most common use for shape matching and several surveys have been written on the subject [124, 48, 143, 22]. A large class of 3D matching techniques use *feature-based* geometric

descriptors that represent 3D shapes in a canonical form, which is invariant under different 3D transformations. This representation allows the 3D matching techniques to be applicable to a large class of tasks such as shape retrieval, registration, object recognition and classification. Another advantage of using these descriptors is their applicability to *partial* shape matching. Non-feature-based 3D matching techniques such as skeleton and graph-based methods [123, 14] are generally unable to handle partial matching, and are therefore only applicable to *global* shape matching. In this chapter, we only review feature-based shape matching techniques.

Partial 3D shape matching is a more difficult problem than global shape matching, since the location, orientation, size or overlap of the regions that need to be matched are not known [38]. The main examples where the need for partial shape matching arises are registration of partial 3D scans [38, 39, 50, 49] and 3D object recognition in cluttered scenes [51, 36]. The general approach to matching shapes with partial overlaps involves a combinatorial search for similar regions over the surfaces of the objects that need to be matched. In most cases, the efficiency of partial shape matching is improved by performing the search on a set of representative points on the surfaces, which are generally referred to as *feature points* or *keypoints*.

In the next subsections we briefly describe how feature-based shape descriptors and feature points are used in 3D shape retrieval, registration, object recognition, and object classification. In Table 2.1, we provide a simplified definition for each of these shape matching problems.

2.1.1 3D Shape Retrieval

Given a query model and a database of 3D models, *3D shape retrieval* refers to the process of finding similar objects in the database. The general framework for a 3D shape retrieval system using shape descriptors is given in Figure 2.1. The 3D retrieval system allows the user to search the database of 3D models using a query model. The system, in an offline

Problem	Input	Output/Objective
3D Shape Retrieval	Query model m and a database db of 3D models.	Find models most similar to m in db.
3D Shape Registration	Set of 3D models, $m_1, m_2, ..., m_n$.	Bring all models into the same coordinate frame and align them.
3D Shape Recognition	3D scene S and 3D model m.	Find all instances of m in S.
3D Shape Classification	3D model m and a set of classes $C = \{c_0, c_1, ..., c_n\}$.	Determine to which class m belongs.

Table 2.1: Applications of 3D shape matching

stage, extracts global shape descriptors for each model in the database. The descriptors are either invariant to geometric transformations or all models are first brought into a canonical frame before the descriptors are computed. The descriptors are generally represented in the form of a high-dimensional vector. Hence, the descriptor extraction stage essentially maps each 3D shape to a point in a high-dimensional space, known as the *feature space*. In the online stage, where the user queries the system using a 3D model, the global shape descriptor for that model is computed. The system then returns all the models in the database whose descriptors are close to the query model's in the feature space.

2.1.2 3D Shape Registration

The task of bringing a set of 3D shapes into a common coordinate frame and aligning them is referred to as 3D registration. By far, the most popular technique for pairwise registration of 3D surfaces is the *Iterative Closest Point (ICP)* algorithm [13]. ICP registers two scans by minimizing the sum of the squared distances between the points in one scan (scene) and their closest points in the other scan (model). At the end of the registration process, the rigid 3D transformation that aligns the scene with the model is obtained.

The main drawback of the ICP algorithm is its requirement for the two scans to be initially close to each other with respect to translation and orientation. In most cases,

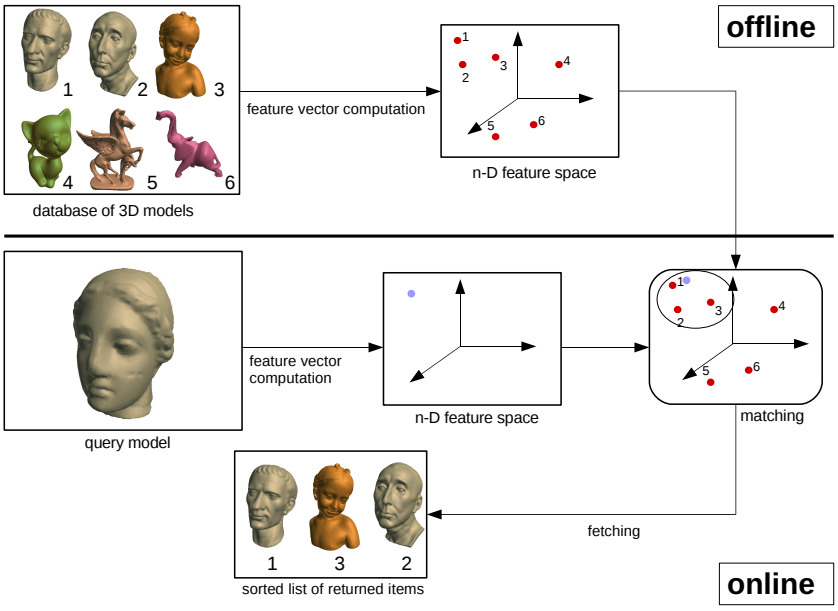


Figure 2.1: Framework for 3D shape retrieval using feature-based descriptor vectors

however, this requirement is not met. As a result, many proposed techniques first try to recover the initial 3D transformation parameters that align the two scans by establishing a set of pairwise correspondence between them [112, 50, 39]. Once this correspondence is established, the two scans can be registered and the initial transformation matrix can be obtained [6, 137]. The registered result can then be used as the input into the ICP algorithm.

Figure 2.2 shows the conceptual steps for registering two partial 3D models using local descriptors. In the first step, a set of *feature points* (or keypoints) are extracted on each 3D surface. The feature point extraction could be done in different ways. The simplest form of feature point extraction is uniformly sampling the 3D surface. We discuss different criteria for picking feature points on surfaces in Sec. 2.4.1. In the next step of registration process, transformation invariant local descriptors are extracted at each feature point. These descriptors are used to establish the initial point-point correspondences between the two surfaces. The correspondences are then used to recover the 3D transformation that brings one of the surfaces into alignment with the other one. A verification step usually follows to ensure the recovered transformation is correct. The output of this process is generally fed

into a local registration algorithm such as the ICP.

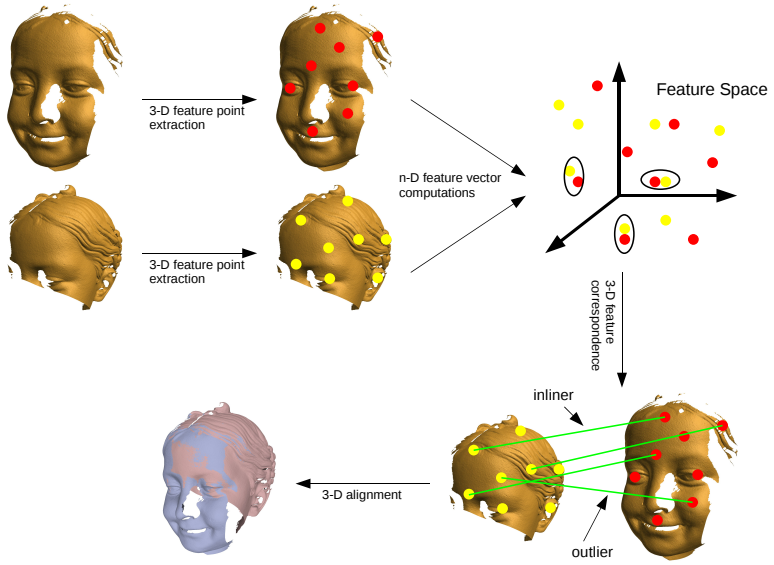


Figure 2.2: Registration pipeline using feature points and local shape descriptor vectors

2.1.3 3D Shape Recognition

In shape (object) recognition, the objective is to find all instances of a set of shapes (objects), which are known from before, in a 3D scene. The 3D approaches to shape recognition are generally similar to the 2D ones: local descriptors are computed at various locations of on the objects in the database and stored. Given a 3D scene, feature points and local descriptors are extracted from it are then used to establish correspondence between points in the scene and the object in the database. A voting scheme such as geometric hashing [62] or RANSAC [16] is generally used to verify the point correspondences by checking that they vote for a consistent pose.

2.1.4 3D Shape Classification

In shape classification, the objective is to identify the class or category of an input shape. Both local and global shape descriptors have been used by matching techniques for auto-

matic shape classification. For example, in [30], the authors employ local descriptors to automatically assign a class to a query model. Figure 2.3 outlines the steps for classifying a query model, q, given a database of models, M, and a set of classes $C = \{c_1, c_2, ...\}$, which partitions the database into disjoint sets of models.

Each model $m_i \in M$ is associated with a single class $C(m_i)$. In an offline stage, m_i is uniformly sampled and semi-local descriptors are computed at each sample point, $p_j \in m_i$. These points are then labeled with class $C(m_i)$. In the online stage, semi-local descriptors are computed at various points r_k on the query model q. Each point r_k is then assigned a class using its associated shape descriptor and its nearest descriptors in the feature space with known classes (which were computed in the offline stage). Finally, a class is assigned to the query model q by identifying the dominant (most common) class associated with its local descriptors, which were computed at sample points $r_k \in q$.

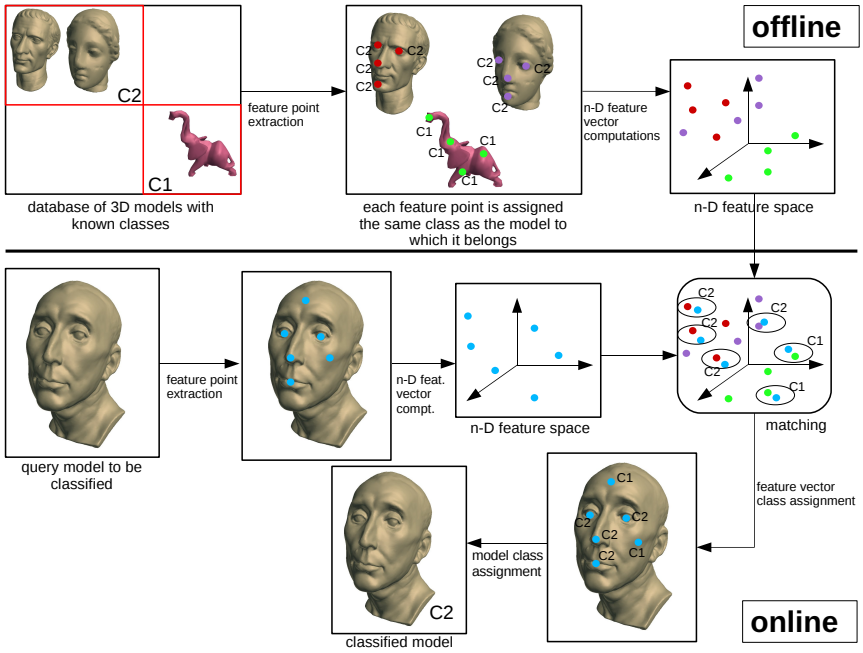


Figure 2.3: Classification pipeline using feature points and local shape descriptor vectors

The remainder of this chapter is organized as follows: in the next section, a definition for feature-based shape descriptors is given and the main considerations for the computa-

tion of these descriptors are discussed. In Sec. 2.3, we classify and review the different shape descriptors employed in the computer graphics and computer vision literature. In Sec. 2.4, we discuss partial shape matching and feature point extraction techniques in more detail. Finally in Sec. 2.5, we provide a summary of this chapter.

2.2 3D Shape Descriptors

As mentioned previously, 3D surfaces are the only 3D shapes discussed in this work. Surfaces can be represented using a variety of techniques. Examples of popular surface representations are polygonal meshes [19], point-set surfaces [3], subdivision surfaces, and Nonuniform Rational B-spline (NURBS) surfaces. Triangular meshes are the most common surface representations used in 3D shape matching applications. Most surface matching algorithms either directly work with or can be modified to work with meshes.

Shape representations and shape descriptors are closely related [81, 75, 140]. Loncaric [75] differentiates between the two by stating that a shape representation produces a *non-numeric* representation of the original shape, such as a graph, which preserves important characteristics of the shape. Whereas, a shape descriptor produces a *numeric* representation of the shape and is generally obtained/computed from shape representations. Bearing in mind that in this work the only shapes that we are concerned with are 3D surfaces, we distinguish between a shape representation and a shape descriptor as follows: a shape *representation* provides a way of encoding the *geometry* of the shape. From this representation different geometric properties (or attributed) of the shape can be derived. On the other hand, a shape *descriptor* encodes one or more geometric *properties* of the shape. In general, these properties cannot be used to recover the original geometry of the shape. Shape descriptors are numeric and represented in the form of a vector of scalar values. In Tangelder *et al.*, these are referred to as *feature-based* descriptors. In the computer vision and pattern recognition literature, terms such as *feature vectors* and *descriptor vectors* are

also used to refer to shape descriptors. We use these terms interchangeably.

3D shape descriptors need to satisfy a few criteria to be useful in shape matching. Since shape representations and descriptors are closely related, the criteria given for the two are very similar. The following lists the main criteria for shape representations:

- *Scope:* The shape representation should be able to handle all classes of shapes. For example, a 3D surface representation cannot be limited to handling only planar or cylindrical surfaces.

- *Uniqueness:* Different shapes should have different representations.

- *Sensitivity*: Small (large) differences in shapes should result in small (large) differences in representations.

- *Stability:* Small fraction of the data perturbed by large amount of noise should produce small changes in the representation.

The above criteria also apply to shape descriptors. An *ideal* shape descriptor satisfies all of the above criteria. Additionally, shape representations and descriptors should satisfy the following practicality criteria [88]:

- *Efficiency:* The representation/descriptor should be computationally and storage-wise efficient.

- *Ease of Implementation:* The representation/descriptor should be relatively easy to implement.

- *Multiscale support:* Most successful shape registration and matching algorithms process the data in a multiscale fashion; for example, by processing the data in a coarse-to-fine manner or vice versa. Therefore, the support for multiscale shape representation/descriptor is always desirable.

- *Computation of shape properties (representation-specific):* It is useful to be able to obtain information about geometric properties of the shape from the representation.

- *Local support (representation-specific):* The shape representation should not depend on global properties of the shape.

A shape descriptor should additionally satisfy the following property [124]:

- *Transformation invariance:* The shape descriptor should be invariant under rigid body transformations, and sometimes scale and reflection.

Satisfying the transformation invariance criterion is very important as the performance of the descriptors is highly affected by it. Translation and scale invariance are generally easier to achieve, compared to rotation invariance. Slight variations in orientations of 3D shapes could result in very different descriptors. However, not all descriptors are inherently transformation invariant. In such cases, an initial *pose normalization* step needs to be performed before computing the descriptors. Pose normalization brings all 3D shapes into a canonical frame, which is invariant under geometric transformations. For example, pose normalization can be achieved by translating each model so that its center of mass is at the origin (translation invariance) and rotating the model so that its principal axes are aligned with the x, y and z $axes$ (rotation invariance). However, it has been shown that transformation invariant descriptors that do not require pose normalization, perform better than other descriptors in 3D model retrieval systems [131].

2.2.1 Locality and Dimensionality

Descriptors, depending on their domain of influence, can be classified as *global* or *local*. Global descriptors describe the 3D shape as a whole and are not affected by small changes in the object and, are therefore, stable against noise. These descriptors are also generally computationally efficient to extract and match, making them ideal for 3D shape retrieval and classification tasks.

Local descriptors on the other hand are computed at various locations on the model and their domain of influence is limited to local regions of the model. These descriptors are more distinctive and more sensitive to noise or surface perturbations and are generally more expensive to compute, store, and match. Local descriptors are mainly used in partial matching tasks such as registration of incomplete 3D scans or object recognition in cluttered scenes. The domain of some local descriptors includes large portions of the model (and sometimes the whole model) [50, 142]. We refer to these descriptors as *semi-local* descriptors.

Feature-based descriptors are represented as vectors of scalar values. This way, each local or global shape represented by the descriptor is mapped (or transformed) to a point in the *feature space* of the descriptor. Due to *transformation invariance*, *uniqueness*, and *invariance* properties of the descriptors, determining the similarity between these feature vectors amounts to computing the distance between points in the feature space. Therefore, the more similar two points are on a surface, the closer their descriptor vectors are in the feature space. The *similarity* between two shapes increases as their feature vectors become closer in the feature space and decreases as the feature vectors become farther away. However, a function, which defines the distance between two descriptor vectors is generally referred to as a *dissimilarity measure*, rather than a similarity measure. This notation is used since the dissimilarity between two shapes *increases* as the distance between the their corresponding feature vectors increases in the feature space.

Depending on the descriptor vector, distance metrics more complex than the Euclidean distance may also be used to measure the dissimilarity. The descriptor and the application at hand dictate which distance metric to use (see section 2.2.2).

Depending on the dimensionality of the descriptor vectors, these vectors can be classified as *high-dimensional* or *low-dimensional*. As one would expect, high-dimensional descriptors tend to be more distinctive as they encode more information about the shape of the object. The domain of applications for each class is different. For example, low-

dimensional descriptors are generally used in *local registration* [39], while, high-dimensional descriptors are mainly used in *global registration* [50].

2.2.2 Shape Similarity Measures

Feature-based 3D shape matching techniques reduce the 3D shapes to a generally high-dimensional feature vector. That is, a 3D shape is transformed to a point in a high dimensional space known as the feature space. The advantage of this approach is that the problem of measuring the similarity between two 3D shapes is reduced to defining a distance function that defines the distance between points in the high-dimensional feature space. A function $d(x, y)$ measuring the distance between two points $x, y \in R^N$ can be classified as a metric, a pseudo-metric or a semi-metric depending on which of the following properties it satisfies [124, 130]:

i. *Positivity:* $d(x, y) \geq 0$

ii. *Identity:* $d(x, y) = 0 \leftrightarrow x = y$

iii. *Symmetry:* $d(x, y) = d(y, x)$

iv. *Triangle Inequality:* $d(x, y) + d(y, z) \geq d(x, z)$

v. *Transformation Invariance:* $d(x, y) = d(g(x), g(y))$, for $g \in G$, where G is a chosen group of transformations.

A distance function is a *metric* if it satisfies properties i, ii, iii, iv, a *pseudo-metric* if satisfies i, iii, iv, and a *semi-metric* if it satisfies i, ii, iii. The following are the most commonly used distance functions in the literature:

- L_p **(Minkowski) Distance:** L_p or the Minkowski distance between two points $x, y \in \mathbb{R}^N$ is defined as:

$$L_p(x, y) = \left(\sum_{i=0}^{N-1} |x_i - y_i|^p \right)^{\frac{1}{p}}. \tag{2.1}$$

L_1 is called the Manhattan or City-block distance, L_2 is the Euclidean distance and L_∞ is the max metric: $L_\infty(x,y) = \max_i(|x_i - y_i|)$. L_p is a metric for all $p \geq 1$ and a semi-metric for $p < 1$, as the triangle inequality does not hold. The Minkowski distance is the most commonly used similarity measure for feature vectors.

- **Correlation Metric:** The correlation metric gives a measure of the angle between two points $x, y \in \mathbb{R}^N$ as follows:

$$C(x,y) = \frac{\sum_{i=0}^{N-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \bar{x})^2 \sum_{i=0}^{N-1}(y_i - \bar{y})^2}}\,, \tag{2.2}$$

  where $\bar{x} = \sum_{i=0}^{N-1} x_i/N$ and $\bar{y} = \sum_{i=0}^{N-1} y_i/N$.

- **Quadratic Form Distance Function:** If the components of the feature space are not independent and the relationship between the components is known, a quadratic distance function Q may be used to measure the similarity between two feature vectors $x, y \in \mathbb{R}^N$:

$$Q(x,y) = (x-y)\cdot A \cdot (x-y)^T = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{ij}(x_i - y_i)(x_j - y_j)\,, \tag{2.3}$$

  where A is a similarity matrix and its components, a_{ij}, represent the similarity between components i and j in the feature space. When A is the identity matrix, Q becomes the Euclidean distance.

Additionally, when the feature vectors correspond to probability distributions or histograms, the following distance measures may also be used:

- **Bhattacharyya Distance:** The Bhattacharyya distance measures the similarity between two discrete probability distributions x and y. It is defined as:

$$B(x,y) = 1 - \sum_{i=0}^{N-1} \sqrt{x_i \cdot y_i}\,, \tag{2.4}$$

where N is the number of bins in the discrete probability distributions.

- χ^2 **Statistic:** χ^2 statistic measures the similarity between two discrete distributions x and y as:

$$\chi^2(x,y) = \sum_{i=0}^{N-1} \frac{(x_i - y_i)^2}{x_i + y_i} \,, \tag{2.5}$$

where N is the number of bins in the discrete probability distributions.

In many cases, the distance between two sets of feature vectors needs to be computed to measure similarity between two objects. Two examples of such distance functions are:

- **Hausdorff Distance:** The Hausdorff distance between two point sets $A = \{a_0, a_1, a_2, ...\}$ and $B = \{b_0, b_1, b_2, ...\}$ is defined as:

$$H(A,B) = \max_{a_i \in A} \min_{b_i \in B} D(a_i, b_i) \,, \tag{2.6}$$

where $D(.,.)$ can be any distance function between two points (e.g., the Euclidean distance).

- **Bottleneck Distance:** The bottleneck distance between two point sets $A = \{a_0, a_1, a_2, ...\}$ and $B = \{b_0, b_1, b_2, ...\}$ is defined as the minimum of the maximum distance of all one-to-one correspondences between A and B. Let F correspond to the set of all one-to-one correspondences between point sets A and B. A function $f \in F$ maps a point in A to B: $f : A \rightarrow B$. The bottleneck distance is then defined as:

$$Bot(A,B) = \min_{f_j \in F} \max_{a_i \in A} D(a_i, f_j(a_i)) \,, \tag{2.7}$$

where $D(\cdot,\cdot)$ can be any distance function between two points.

2.3 Classes of 3D Surface Descriptors

All descriptor vectors can be classified as being *global*, *local*, and sometimes *semi-local*. Global descriptors represent the 3D shape of the whole model in the form of a single descriptor. Whereas, using local descriptors to represent a model, amounts to breaking the model into local patches and representing each patch with a local descriptor. The techniques used to represent 3D shapes using *global* descriptors can be classified into the following categories:

- *Global Shape Properties:* These descriptors encode global shape properties such as volume, surface area, volume to surface area ratio, k-fold symmetry, *etc*.

- *Maps:* Map-based descriptors encode the geometry of the surface or the occupancy volume of the model by storing the positions or orientations of sampled vertices on the surface in the form of maps. These descriptors tend to be more distinctive and intuitive than other descriptors as they encode more information and are generally easier to implement. A major drawback of these methods is that most require a pose normalization of the shape prior to computing the descriptors.

- *Projections:* Projection-based descriptors represent a 3D shape by projecting a function on the model, such as its characteristic function, onto a set of orthogonal basis functions. 3D-DFT, Spherical Harmonics, 3D Zernike Moments are examples of such descriptors. One advantage of using this class of methods is that they can be used to derive inherently geometric invariant descriptors.

- *Feature Distributions and Histograms:* These descriptors are created using histograms of local geometric properties of the shape such as mean or Gaussian curvature sampled over the surface of the object [117]. Other types of histograms that sample relational properties such as distances between sampled points on the surface have also been proposed [97].

Local descriptors are, in general, localized maps. Table 2.2 shows a classification of the 3D descriptor techniques. The following subsections describe each method in more detail.

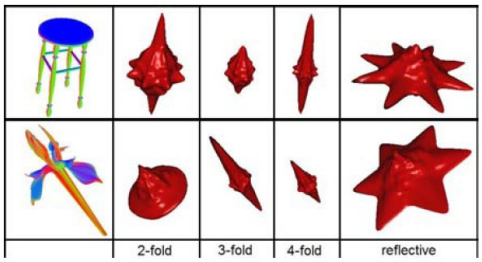
2.3.1 Global Shape Properties

Global shape properties of 3D objects such as hull packing, hull compactness, and global symmetry have been used to describe 3D shapes. This class of low-dimensional descriptors, as one would expect, tend to be very coarse and less discriminative compared to other shape descriptors. However, the main advantage of these descriptors is their stability against noise. The descriptors have also been used to improve the performance of other shape descriptors [58, 57].

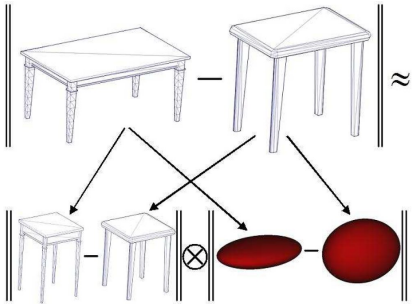
In [144], the authors propose a method for computing global shape features such as volume, moments, and Fourier transform coefficients directly on meshes. The global features are determined by computing the features on elementary shapes such as triangles or tetrahedrons and then summing the values over the surface. The computational cost is proportional to the number of triangles.

The authors of [27], propose and compare different volume and convex-hull based descriptors for use in a 3D search engine. The following dimensionless ratios were used for comparing objects:

- Bounding-box aspect ratio: the ratio of the longest to the shortest edges of the bounding box.
- Hull Crumpliness: the ratio of the object's surface area to the area of its convex hull.
- Hull packing: the percentage of volume of the convex hull of the object not occupied by the object.
- Hull Compactness: the ratio of the cubed surface area of the convex hull of the object to the squared volume of the convex hull.



(a) Visualization of the symmetry descriptor of [58]. The rotational symmetry visualization for each model is created by scaling unit vectors on the spheres proportional to the symmetry of the model about that vector. The reflective symmetry visualization is computed by scaling the unit vectors proportional to reflective symmetry of each model about the plane passing through the center of the model and normal to the vector.



(b) In [57], the anisotropy influenced similarity is computed by first scaling the models so that they are isotropic. The similarity is then defined as the outer product of the differences between the isotropic models and their initial anisotropic scale.

Figure 2.4: Visualizations of (a) the symmetry descriptor [58] and (b) the anisotropy similarity measure [57]. (Images courtesy of Michael Kazhdan)

In [58], the authors introduce the Symmetry Descriptors, which measure the reflective and rotational symmetry of a model with respect to every axis passing through the center of the model. Given a function, f, on the sphere, the k-fold *anti-symmetry distance* of the function with respect to axis p is defined as the average of the correlation of the function with the k different rotations of itself around p. That is,

$$SDist_k(f,p) = \sqrt{\frac{1}{|G_p^k|}\sum_{\gamma \in G_p^k} < f, \gamma(f) >}\,, \tag{2.8}$$

where G_p^k denotes the $k-$fold rotational symmetry group with respect to p. Figure 2.4 (a) shows a visualization of the $2, 3, 4$-fold, and reflective symmetries of two different objects. To efficiently compute the spherical auto-correlation, f is expressed in terms of its spherical harmonic decomposition and the correlation is done in the spherical harmonic domain. The $k-$fold symmetry descriptor is defined as the symmetry and anti-symmetry distance of f with respect to p_i, in which $SDist_k$ is maximal:

$$Sym_k(f) = (s_k(f), \sqrt{\|f\|^2 - s_k(f)^2}), \text{ where } s_k(f) = \max_{p_i \in S^2} SDist_k(f, p_i)\,. \tag{2.9}$$

The authors use the symmetry descriptor to augment their spherical harmonic descriptors [56]. They show that the augmented descriptors perform better in a 3D retrieval system.

In [57], the authors propose a method for augmenting existing shape descriptors by incorporating the anisotropic information about the models into the descriptors. In the proposed method, the model is first scaled so that it becomes isotropic. This is done by scaling the model by multiplying it by the square root of the inverse of the covariance matrix, $C^{-1/2}$, where C is the covariance matrix of the model. The shape descriptors are then computed on the isotropic versions of the models. The similarity, S, between two models, M and N, is defined in terms of the distance between the computed descriptors and their anisotropic difference:

$$S_\gamma(M,N) = \|v_{\tilde{M}}\|^2 + \|v_{\tilde{N}}\|^2 - 2\,\langle v_{\tilde{M}}, v_{\tilde{N}}\rangle\,\langle \lambda_M, \lambda_N\rangle^\gamma \ , \tag{2.10}$$

where $\tilde{M}$ and $\tilde{N}$ are the anisotropically scaled versions of M and N, $v_{\tilde{M}}$ and $v_{\tilde{N}}$ are descriptor vectors of $\tilde{M}$ and $\tilde{N}$. λ_M and λ_N are sorted triplets of eigenvalues of the covariance matrices of M and N. γ controls how much influence the anisotropy of the models should have on the similarity measure. Figure 2.4 (b) shows a visualization of the above similarity formulation. The authors show that the proposed method improves the performance of the existing descriptors in 3D retrieval systems.

2.3.2 Maps

Maps provide the largest class of shape descriptors and have been used as both local and global descriptors. The main attraction of this class of descriptors is their intuitive way of representing shapes, which can also be easily visualized.

Maps discretize the space around the model and store geometric information about the surface such as positions of sampled points, their associated normals or curvatures in the discretized space. Most maps are not rotationally invariant and the object needs to be pose-

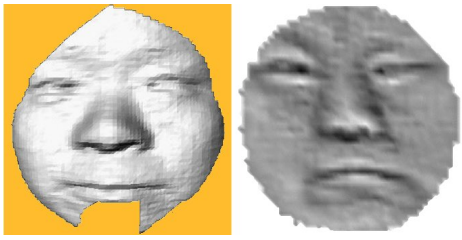


Figure 2.5: Harmonic Shape Image: example of planar parameterization of a 3D surface patch [145]. (Image courtesy of Martial Hebert)

normalized prior to the computation of the descriptor.

The mapping of the discretized space to $1, 2, 3$-D domains is generally done using one of the following methods:

- Voxelization: the 3D space is discretized by directly voxelizing the space in either Cartesian or Spherical coordinates.
- Spherical maps: the 3D positions or orientations are mapped to positions on the sphere.
- Planar maps: the 3D positions are mapped or projected onto a 2D domain by either parameterizing the surface or simply projecting points onto a plane.
- Curves: the local neighborhood is parameterized along geodesic or Euclidean rings around a point of interest.

Figure 2.5 shows an example of a Harmonic Shape Image.

The maps can encode different geometric properties of the surface. For example, each bin could simply contain the number of vertices on the surface that fall into it (encoding positions). The bins could also contain higher order surface properties such as local curvatures or normal orientations.

Voxelization

3D shape contexts and shape histograms[60, 11] constitute a popular set of map-based descriptors. They are extensions of shape contexts from 2D images. Shape contexts discretize the 3D space around each point of interest and count the number of neighboring points that fall into each bin. The 3D space around the point of interest is generally discretized in one of the following three forms:

- *Shells:* space is discretized in the form of concentric shells.
- *Sectors:* space is discretized in the form of sectors originating from the point of interest.
- *Spiderweb:* space is discretized using both shells and sectors.

Only the shells method is rotation invariant. The other two forms of the 3D shape contexts require some of form of pose normalization prior to the descriptor computation. 3D shape contexts have been used both as local [11] and global [5] descriptors.

In [132], the authors voxelize the space around the model by first translating the model so that its center of gravity is as the original (translation invariance) and then rotating and aligning it with its principal axes (rotation invariance). The authors also, attempt to obtain reflection invariance by reflecting the model around each plane (xy, yz, zx) so that the majority of the points of the object lie on the positive side of each plane. Scale invariance is obtained by scaling the model by $s = \sqrt{(s_x^2 + s_y^2 + s_z^2)/3}$, where s_x, s_y, and s_z represent the average distances of object points from the origin along each axis. Once the model is in this canonical frame, the bounding cube of the object is computed and voxelized into an $N \times N \times N$ grid. Each cell in the grid contains the portion of the surface area of the object that lies inside the cell. The authors do not directly use the occupancy grid as a descriptor for the object. Instead, the 3D Fourier transform of the object is computed and the frequency coefficients are used as the descriptors.

The authors of [95] use similar steps to create a voxelization of the model before computing the 3D Zernike descriptors. The model, however, is not aligned with its principal axes as the proposed 3D Zernike descriptors are rotation invariant and prior rotation normalization is not required (see section 2.3.3 for more information on projection-based techniques such as 3DDFT, spherical harmonics, and 3D Zernike descriptors).

Spherical maps

The main advantage of using a function on the sphere to describe a 3D shape is that the function can be decomposed using spherical harmonics and represented in terms of its frequency components. Representing a function in terms of its frequency components has many advantages; the main advantages are: 1) The function can be low-pass filtered by simply dropping the high-frequency components. 2) Rotation invariant descriptors can be derived by using the frequency energies as descriptor components. 3) Even if the function is not decomposed in terms of its spherical harmonics, and a descriptor is derived in the spatial domain, the descriptor matching process can be sped up by performing the distance computations, such as correlation, in the frequency domain. These properties are the reasons for the success and generally superior performance of most spherical descriptors.

Extended Gaussian Images (EGI) [54] are computed by mapping the normal directions of the vertics on the surface to points on the sphere. The obtained function on the sphere effectively becomes a histogram of normal directions. Matching can be done using general template matching techniques. However, methods such as [79] speed up the template matching process by decomposing the function into its spherical harmonics and performing the template matching in the harmonics domain, which can be performed much faster (in the form of a multiplication).

Spherical Extent Function [106] is another example of spherical functions used to describe 3D shapes. The spherical extent function is computed by centering the object inside a bounding sphere and extending rays from the center of the object to sampled points on

the sphere. The value at each sampled point on the sphere is the maximum distance of the surface of the object from the origin along the ray connecting the origin to the point on the sphere. If a ray does not intersect any points on the surface, the sampled value along that direction is set to zero. Mathematically, spherical extent function, f, describing a 3D surface, M, is defined as:

$$\begin{aligned} &f : \mathcal{S}^2 \rightarrow \mathbb{R} \\ &f(\hat{u}) = \max\{r \geq 0 | r\hat{u} \in M \cup \{\mathbf{0}\}\} \ , \end{aligned} \tag{2.11}$$

where $\mathcal{S}^2$ denotes the unit 2-sphere, $\hat{u} \in \mathcal{S}^2$ is a point on the unit sphere and $\mathbf{0}$ is the origin. The authors use spherical harmonic decomposition and geometric moments of the function to derive two different descriptors. They show that the descriptors using the harmonic decomposition of the function perform better in their 3D model retrieval system. The descriptors, however, are not rotation invariant and the approach requires a pose normalization of the model prior to the descriptor computations.

The same authors, in [131], try to improve the performance of the descriptors by using more than one spherical extent function to represent a model. A set of spherical extent functions are used by creating a set of concentric spheres sampling the inside of the model as well (instead of just using one bounding sphere). Their approach is very similar to that of [56]. The object is first voxelized into an $N \times N \times N$ binary grid. A cell has value 1 if it intersects the surface of the model and 0, otherwise. Finally, concentric spheres of varying radii are created inside the voxel space, each sphere containing the spherical extent function of the portion of the voxelized model that falls inside it. The authors use the same technique as [56] to derive rotation invariant descriptors for each spherical extent function (see section 2.3.3 for more details on the invariant descriptors).

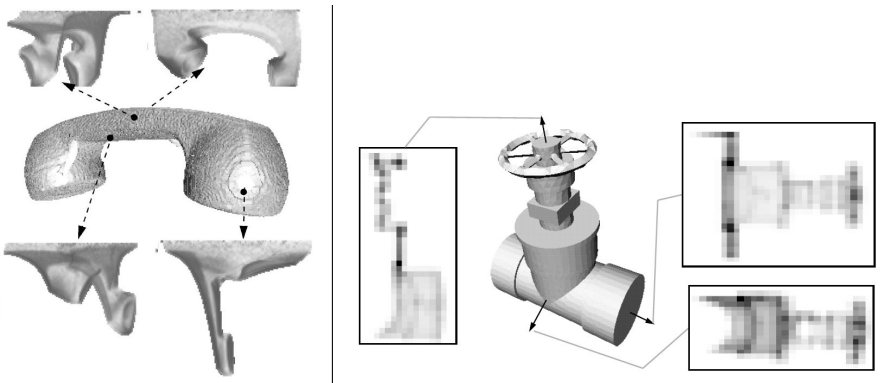
Planar maps

One of the earliest examples of map-based techniques and by far the most popular are spin images [50]. Spin images are local descriptors that are computed at various locations on the surfaces of 3D models. The point where a spin image is constructed is referred to as the *central point*. Assuming the normal direction at the central point is known, a map of the neighboring surface points is computed by rotating a discrete compact plane that passes through the central point and is parallel to the point's normal, around the normal. The plane is represented in the form of a $w \times h$ image. Each bin in the spin image contains the number of neighboring points that fall into it when the plane is rotated around the normal (see Figure 2.6(b)). Spin images are inherently rotation invariant and do not require any pose normalization prior to their computations. Different variations of spin images have been proposed. For example, in [29], the authors propose multi-resolution spin images, where at each central point, spin images of different resolutions and sizes are computed. The authors report improved performance over traditional spin images.

Harmonic Shape Images (HSI) [145] are another example of planar maps. Each HSI is computed at a local patch, D, around a central point, p_c on the surface of the input model. The descriptors use harmonic maps [31] to obtain a planar parameterization of D. This is done by first mapping the boundary vertices of D to the boundary of the unit disk. A map $\phi : D \rightarrow \mathbb{R}^2$, which maps the interior points in D to locations inside the unit disk is then obtained by minimizing the error functional

$$E(\phi) = \frac{1}{2} \sum_{(i,j) \in Edges(D)} k_{ij} \|\phi(i) - \phi(j)\|^2 \,, \tag{2.12}$$

where $\phi(i)$ and $\phi(j)$ are the images of the interior vertices $i, j \in D$. This parameterization can then be used to create a 2D map of different surface properties at the vertices; the authors use surface curvatures in their implementation. Harmonic shape images are not rotation invariant. Therefore, the similarity between two descriptors is measured by com-



(a) (b)

Figure 2.6: Examples of (a) surface signatures [142] and (b) spin images [50] computed at different locations on 3D models. (Images courtesy of (a) Sameh Yamany, and (b) Andrew E. Johnson)

puting the normalized correlations between one of the descriptors and rotated versions of the other one. The rotation which yields the maximum correlation is used as the distance between the two descriptors.

Surface Signatures [142] encode the geometry of the surface by creating images of the curvature values on the surface seen from a point on the surface. These images are calculated at various locations on the surface. The method is similar to spin images in that it creates rotation invariant descriptor images at various points on the surface by mapping vertices in 3D to a bin in a 2D image. Signature images, however, encode surface curvatures instead of point densities and also use polar coordinates instead of cylindrical coordinates for mapping 3D points to 2D. To create a surface signature at point, p_c, the 2D coordinates of each point, p_i, on the surface is computed using:

$$\begin{aligned} y_i &= \|p_c - p_i\| \\ x_i &= \cos^{-1}\left(\frac{\langle n_{p_c}, p_c - p_i\rangle}{\|p_c - p_i\|}\right) \end{aligned} \tag{2.13}$$

where n_{p_c} is the normal at p_c (Figure 2.6(a) shows examples of surface signatures computed at different locations on a 3D model). To improve the efficiency of the descriptor computations and matching process, the authors only compute the descriptors at points

with absolute mean curvature value greater than a threshold, λ.

Li *et al.* [70] propose a multiscale approach for feature point extraction (section 2.4.1) and descriptor computation for the purpose of 3D registration. The descriptors are local maps of normal directions around each feature point, $\mathbf{p_i}$. The normal map at position $\mathbf{p_i}$ is computed by projecting the normal directions of its neighboring points onto the tangent disk, defined by its normal, $\mathbf{n_{p_i}}$, and radius, r. More specifically, each map consists of an $N \times M$ array of 3D points, $C = \{\mathbf{c_{kl}}\}$, sampled on the tangent disk around $\mathbf{p_i}$:

$$\mathbf{c_{kl}} = \mathbf{p_i} + \frac{2lr}{M}\left(\cos\left(\frac{2\pi k}{N}\right)\mathbf{u} + \sin\left(\frac{2\pi k}{N}\right)\mathbf{v}\right) , \tag{2.14}$$

where $\mathbf{u} \perp \mathbf{v} \perp \mathbf{n_{p_i}}$, and $(\mathbf{u}, \mathbf{v}, \mathbf{n_{p_i}})$ forms an orthogonal frame at $\mathbf{p_i}$. Array of 3D points, C, is then used to create an array of scalars, $S = \{s_{jk}\}$, by projecting the normal, $\mathbf{n_{c_{jk}}}$, at each $\mathbf{c_{jk}}$ onto the direction connecting $\mathbf{c_{jk}}$ to $\mathbf{p_i}$:

$$s_{jk} = \frac{\left\langle \mathbf{c_{jk}} - \mathbf{p_i}, \mathbf{n_{c_{jk}}} \right\rangle}{\|\mathbf{c_{jk}} - \mathbf{p_i}\|} . \tag{2.15}$$

$\mathbf{n_{c_{jk}}}$ is computed by taking the weighted average of the normal directions of points on the surface neighboring $\mathbf{c_{jk}}$. Frequency components of S are then obtained by computing the Discrete Cosine Transform of S in the l-direction, followed by the Discrete Fourier Transform in the k-direction. The final descriptors are the magnitudes of the lower frequency components of S.

Curves

Point signature [26] of the local neighborhood at a point p_c on the surface is computed by creating a 1D distance map of all points in the $r-$radius of p_c. The set of neighboring points is obtained by intersecting a sphere of radius r centered at p_c with the surface. A local plane is then fitted through the neighboring points and a normal direction, N, is obtained. A $1D$ distance map of the neighbors is then computed by measuring the distance of the

neighboring points from the plane with normal N centered at p_c. In order to obtain rotation invariance, the distance map is rotated so that the angle with the maximum distance is at the origin (0). The signature points are extracted at every vertex on the model.

The point fingerprint [121, 122] of a point p_c on the surface is defined using the geodesic circles around p_c. The geodesic circles of different radii are then used to parameterize the local neighborhood of p_c and encode different structural information about the local surface. The authors use the fingerprints to encode normal and radius variations along each geodesic ring around p_c. Other types of information such as curvature, color and other surface properties can also be encoded using the fingerprints. The authors show the application of their local descriptors in a surface registration problem. The drawback of using the geodesic rings to parameterize the local neighborhood of a point is the sensitivity of the geodesic computations to noise. The authors propose to perform a surface smoothing step prior to the descriptor computations.

2.3.3 Projections

Projection-based descriptors represent a 3D shape by projecting a function defining the shape onto a set of orthogonal basis functions. The functions defining the shape of the object are in most cases represented in the form of maps (see section 2.3.2). The main attraction of projection-based methods is that they can be constructed so that they are invariant under rotations. This is generally attained by forming descriptors using the energies of the lower-frequency components of the function. This property makes them ideal for representing general, complete 3D models, which are represented coarsely using a function in a voxelized space. However, the discretization and voxelization of the boundary function of the 3D shapes in this manner makes them less descriptive, hence less ideal, for local matching. 3D Zernike moments [95], Spherical harmonics [56], 3D Fourier Transforms [132] and spherical wavelets [61] are examples of the basis functions that have been used in the literature.

In [95], the proposed 3D Zernike method of [24] is extended for use in 3D shape retrieval. The 3D Zernike moments are computed by projecting the indicator function of the object onto a set of orthonormal functions within the unit sphere. Moments of orders up to 20 are needed to provide a good descriptor for the voxelized 3D shapes. The retrieval performance of these descriptors is comparable to that of Spherical Harmonic descriptors.

Spherical Harmonics [56, 131, 36, 94] provide another popular class of descriptors. These descriptors generally require a large local support and in most applications they are used as global descriptors.

In [56], the authors propose a rotation invariant representation of a function on the sphere using Spherical harmonics. The authors also show how the method can be extended to voxel representations of 3D models and how the descriptors can be used to augment some of the existing non-rotation invariant descriptors.

The spherical harmonic descriptors of [56] represent a spherical function in terms its energies at different frequencies. As shown in the paper, this representation is invariant under rotations. A spherical function, f, can be represented in terms of its spherical harmonics as:

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{m=l} a_{lm} Y_l^m(\theta, \phi) \tag{2.16}$$

The proposed Spherical Harmonic descriptor of f is then defined as:

$$SH(f) = \{\|f_0(\theta, \phi)\|, \|f_1(\theta, \phi)\|, ...\} \tag{2.17}$$

where $f_l(\theta, \phi) = \sum_{m=-l}^{m=l} a_{lm} Y_l^m(\theta, \phi)$.

It can be shown that $SH(f)$ is invariant under rotations. Using this approach any existing shape descriptor represented as a spherical function can be augmented so that it becomes rotation invariant. The authors show how the spherical harmonic descriptors improve the retrieval performance of existing descriptors such as the Spherical Extent Function and Shape Histograms.

Authors of [132] propose a descriptor based on 3D Fourier Transform of a voxelized model. Their descriptors, however, are not rotation invariant, and a pose normalization step needs to be performed prior to computation of the descriptors. The 3D models are pose normalized by moving the center of mass of the objects to the origin and rotating and aligning each object with its principal axes. A scale normalization step is also performed by scaling each model using the average distances of model points from the origin along X, Y, and Z axes. After the pose and scale normalization, a voxel representation of the model is computed. Each voxel cell is weighed using the portion of the mesh surface area that lies in that cell. Finally, discrete 3D Fourier transform of the model is computed. The final descriptor is represented in terms of the absolute values of the Fourier coefficients.

2.3.4 Feature Distributions and Histograms

Distribution-based methods represent 3D shapes in form of histograms of local or global geometric properties of the surface. Examples of local properties include the mean or Gaussian curvatures at sampled points on the surface. Global geometric properties may include Euclidean distances between random points, or angles between edges formed by connecting three random points on the surface. The histograms in most cases are represented in the form of high-dimensional feature vectors.

In [98, 99], a content-based 3D shape retrieval system is introduced. The system allows the user to perform queries using different 3D model properties such as geometry, color and texture. The system performs geometry analysis and similarity search using two types of histograms and a multi-resolution wavelet descriptor. The first type of histogram is computed by creating histograms of the angles between surface normals and the principal axes. The second type of histogram is formed by computing the angles between *cords* and the principal axes. Cords are defined as the vectors connecting the center of the object to the center of mass of each mesh triangle on the surface. A histogram of cord lengths are also used as another shape descriptor. The system allowed the user to search the system

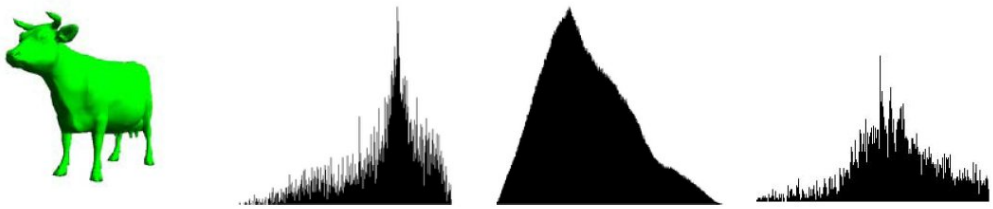


Figure 2.7: Illustration of the three types of histograms used in [129]. From left to right: curvature index, distance and volume histograms of the cow model. (Image courtesy of Jean-Philippe Vandeborre)

using each type of the descriptor.

In [129], three different types of histogram-based descriptors are introduced for describing 3D shapes in a retrieval system. The first histogram is the distribution of the curvature index [59] over the surface. The curvature index at a point, P, on the surface is defined in terms of its principal curvatures as follows:

$$I_P = \frac{2}{\pi} \tan^{-1} \frac{\kappa_P^1 + \kappa_P^2}{\kappa_P^1 - \kappa_P^2}, \text{ with } |\kappa_P^1| \geq |\kappa_P^2| \,. \tag{2.18}$$

The second descriptor is similar to that of [97]. The surface of the model is sampled by taking two random points on each of its faces. To compute the histogram, two random faces are picked and the distances between the sampled points on the faces are computed. The process is iterated N times and a histogram of these distances is obtained. The third descriptor is the histogram of the volume of the tetrahedrons obtained by connecting the center of the object to each triangular face on the model. Figure 2.7 shows illustrations of each type of histogram.

Shape distributions of [97] are computed by sampling the surface of the model with V equally spaced vertices and computing histograms of different shape functions defined in terms of distances between these samples. The following shape functions were introduced:

- **A3:** The angle between three random points on the surface.

- **D1:** The distance of a random point on the surface from a fixed point; the center of the object was used as the fixed point, in their implementation.
- **D2:** The distance between two random points on the surface.
- **D3:** The square root of the area of the triangle defined by three random points on the surface.
- **D4:** The cube root of the volume of the tetrahedron defined by four random points on the surface.

The D2 function performed better than the other functions, in their experiments. The D2 function was extended in [47] by defining three additional types of distances between two sample points:

- **IN distance:** The line connecting the two sample points lies completely inside the model.
- **OUT distance:** The line connecting the two sample points lies completely outside the model.
- **MIXED distance:** The line connecting the two sample points passes both inside and outside of the model.

Using the new distances, three new histograms (besides the D2 histogram) were created. Similar to [97], the histograms were used to define a similarity measure between two 3D shapes in a retrieval system.

2.4 Partial Shape Matching Using Feature Points

Partial matching is generally required in applications such as partial 3D surface registration and object recognition in cluttered scenes. Partial shape matching is a more difficult prob-

lem than global shape matching since no prior knowledge about the orientation, scale, extent or location of the overlapping regions is given. Given two 3D surfaces, partial matching involves detecting the overlapping regions between the two surfaces and also determining the amount of similarity between these regions.

For example, in the case of registering partial 3D scans of a model taken from different views, no prior knowledge is given about the extent or locations of overlap between the scans. Global shape matching techniques cannot be used here as the two shapes are not the same. Therefore, a search for similar region needs to be performed. In such cases, a preferred way to find matching regions is to use local shape descriptors. Local descriptors are extracted at specific locations on the surfaces and a combinatorial search is performed to find correspondences between points on the surface. This is the technique employed by many registration algorithms, which use local descriptors [50, 145, 142, 121].

Therefore, the majority of techniques in the literature perform partial matching in two steps. In the first step, "interesting" features on the surfaces are extracted. Then shape matching is performed between the extracted features using local descriptors. The feature selection step is generally referred to as *feature point extraction*, since in most cases, the features are points or regions represented by points on the surface.

In this section we provide a brief survey of the feature point extraction techniques. The matching process generally uses the same local descriptors discussed in the previous sections. However, since the local descriptors are not distinctive enough, many of the matches are incorrect. Therefore, an additional step is required to verify the consistency of the matches and discard false matches. This step is applied iteratively to improve the matching results. Depending on the application, different methods are used to detect or improve the initial set of matches [39, 51].

2.4.1 Feature Point Selection

The feature point extraction process samples a 3D model by picking points at various locations on, and sometimes around, the model's surface. The general objective is to reduce the geometry of the surface to a set of representative points, in an effort to reduce the computational complexity of the subsequent steps in the matching algorithm. The simplest way to extract feature points is to uniformly sample points over the surface of the model. However, most techniques attempt to pick points at locations, which are geometrically interesting, such as corners, singularities, or regions with high-frequency information. The performance of the feature point extraction process is usually evaluated using the following criteria:

- **Repeatability:** the feature point extraction process should be able to pick a consistent set of points on the same surface under different transformations. That is, it should be stable against noise, small surface perturbations, and different surface resolutions or triangulations. It should also be invariant under geometric transformations.

- **Coverage:** the feature point extraction process should be able to pick points that approximately cover the whole surface. Even though it is desired to pick geometrically interesting feature points, these points should not be only be concentrated at a few locations on the surface.

The repeatability criterion ensures the robustness of the feature extraction process to variations that may be present in different scans of the same surface due to noise, occlusions, and other factors. The coverage criterion ensures that even in cases of small overlap between scans, the extractor picks features, which are present in both scans.

The surface features used in the matching process may be points, lines, curves, or primitive 3D shapes such as spheres and cylinders. The higher the dimension of the feature, the more difficult it will be to use it in later stages of the matching process. This is generally why points are used in most matching applications. Indeed, even in cases when a region

is picked as a feature, it is represented by a point [38]. Therefore, most matching algorithms are interested in feature *point* extraction. Another advantage of point features is that they can be used in a broader range of matching tasks as compared to higher dimensional features.

Feature point extraction techniques can generally be classified into three categories, based on the type of features they use:

- **Geometric features:** the extrema of geometric surface attributes, such as mean and Gaussian curvatures are used as features.
- **Perceptual features:** approaches similar to the scale-space based techniques for 2D images are used to select the surface features.
- **Statistical features:** local shape descriptors are computed at various locations on the model surface, and points with statistically rare descriptors are selected as the features.

In the following subsections, we describe each class of techniques in more detail.

Geometry-Based Feature Extraction Methods

Geometry-based feature extraction techniques select features on 3D surfaces by first computing various geometric surface attributes (*e.g.*, differential surface attributes) over the surface. Subsequently, surface points where these attributes satisfy certain conditions (*e.g.*, are locally maximum) are selected as the surface features. For example, Thirion *et al.* [128] define *extremal points* as the extrema of the crest lines on 3D surfaces. They argue that these points are the equivalents of corners in 2D images. They show an application of their proposed technique for registering 3D medical data.

Yamany *et al.* [142], in an attempt to improve the computational efficiency of their partial shape matching algorithm, subsample the surface prior to the descriptor computations. They achieve this by first estimating the Gaussian curvatures over the input surface.

The surface is subsequently subsampled by selecting points on it where the Gaussian curvature is higher than a preselected threshold. Gains in computation times are achieved by performing descriptor computations and matching at these sample points.

In [38], *salient geometric features* are used to detect and represent important regions on 3D surfaces. The salient regions are defined as clusters of local descriptors that correspond to more interesting (or important) surface regions. The importance measure is based on the surface curvature and its variance in the local region around each point. First, the Gaussian curvature at each vertex on the mesh is computed by fitting a quadratic patch to the local neighborhood and computing the curvature analytically. A region growing technique is then used to create local patches on the surface using the computed curvature values. The center of mass of each patch is then picked as the representative point for that patch. The local patches are then clustered together using a saliency grade defined over the cluster of patches. Initially, each patch constitutes one cluster. Clusters are then grown by iteratively adding the neighboring patches that maximize the saliency grade of the cluster. Finally, the clusters with the highest saliency grade are picked as the salient geometric features of the surface. The saliency grade, S, of a cluster, F, is defined in terms of the Gaussian curvature and its variance in the cluster as

$$S = \sum_{d \in F} W_1 Area(d) Curv(d)^3 + W_2 N(F) Var(F) \tag{2.19}$$

where $Area(d)$ is the area of a patch, $Curv(d)$ is the curvature associated with the patch, $N(F)$ is the number of local curvature extrema in the cluster, and $Var(F)$ is the curvature variance in the cluster.

Perception-Based Methods

Perception-based feature point extraction techniques aim at selection interesting features on 3D surfaces in a manner similar to the human visual system. As discussed in Chapter 3, this

can be achieved by analyzing the surface structures using scale-space based methods. These techniques allow for the recovery of both the location and an associated neighborhood size (scale) of the surface features.

Pauly *et al.* [100] and later, Li *et al.* [70] use the MLS projection operator [3] in a multiscale fashion to extract salient features on point set surfaces. In both cases, the MLS operator is used to obtain a stack of increasingly smoothed versions of the input surface. The amount smoothing at each level is control by the size of the neighborhood size used to compute the MLS projection operator. At each level, the saliency value of each point is then measured by the amount it moves along its normal direction, between the current level and the next. The points whose saliency value is minimum or maximum among their neighbors at the current, previous and next levels are picked as feature points. The level at which each feature point is extracted is used as the scale associated with that point. It is shown that the recovered scales correspond to the size of the underlying structures on the surface. Additionally, the extracted features are shown to be stable in the presence of noise or variations in surface resolutions.

Statistical Methods

Statistical methods first compute local shape descriptors at various locations on surfaces. Surface features are then selected as the points where corresponding descriptors are statistically rare. For example, Johnson *et al.* [50, 49] compute the distribution of their proposed descriptors (Spin Images) and only use points with rare descriptors for 3D matching. Gelfand *et al.* [39] use their proposed Integral Volume Descriptor to only select and use a representative set of points on surfaces when registering 3D scans.

Shilane *et al.* [115, 116] define distinctive regions on an object as regions which are unique to object's class and are not found in other object classes, in a given database. The authors argue that feature extraction methods, which take into account geometric surface properties only on a per-object basis, are not guaranteed to produce discriminative feature

points among different object classes. As a result, it is argued that their proposed distinction measure will perform better in 3D model retrieval or classification systems, which deal with a large number of object classes. A database of 3D models partitioned into disjoint sets of classes is used to compute the distinction measure at points on each model as follows:

1. Each model in the database is sampled by uniformly distributing points on its surface.
2. At each sample point, a semi-local shape descriptor is computed.
3. Each descriptor at a sample point is used as a query key to retrieve similar models in the database. The distinction measure at the sample point is then computed in terms of the percentage of returned models that belong to the same class as the query model. The Discounted Cumulative Gain (DCG) [65] of the returned list is used to compute the distinction measure.
4. A continuous distinction measure over the entire surface of each model is then obtained by assigning each vertex a weighted average of the distinction measures of the neighboring sampled points extracted in step 1.

The authors show applications of the distinction measure to shape matching, mesh simplification, and icon generation.

2.5 Summary

In this chapter, we provided an overview of feature-based shape descriptors used for shape matching. The surveyed descriptors were classified into the following four classes:

- Global shape properties
- Histogram
- Maps

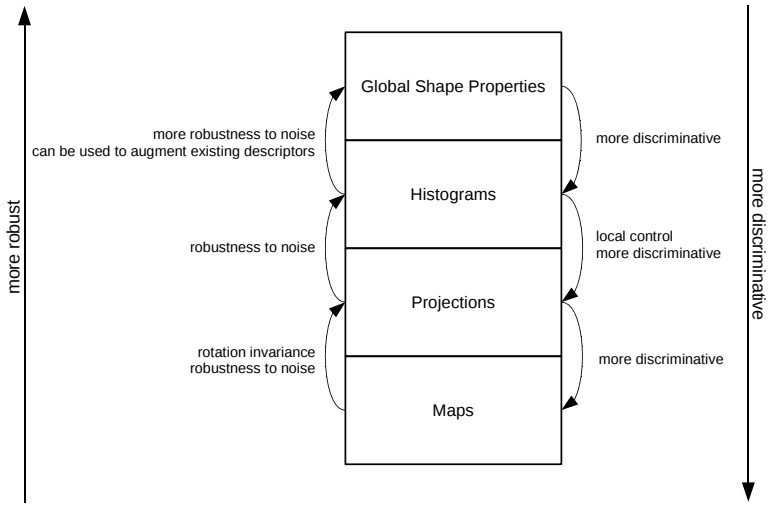


Figure 2.8: Trade-offs of feature-based descriptors

- Projections

Table 2.3 summarizes the major advantages and disadvantages of each class of methods. Figure 2.8 provides a visualization of the relative trade-offs of the feature-based descriptors.

Global shape properties provide a crude description of a shape as a whole. The matching methods employing these descriptors are able to measure similarity between two shapes without gaining any additional information about the relative pose or degree of similarity between sub-parts of the shapes. These methods are robust to noise and shape variations and, in most cases, do not require any pose normalization.

Histograms and shape distributions are able to encode more information than global shape properties and are, therefore, more discriminative. These descriptors perform well in classification applications and retrieval of objects in databases with large number of classes. However, they are not discriminative enough to be used in shape matching applications involving objects of the same class.

Maps, on the other hand, are very discriminative and can be used in shape matching applications involving similar shapes such as 3D registration and recognition. Another advantage of maps is their support for partial matching. Almost all local descriptors, in one way

or another, employ localized maps. Most maps can also be used in a multiscale manner and, hence, can improve the efficiency of the matching process. The major drawback of maps is their sensitivity to noise and, in many cases, their reliance on prior pose-normalization of the objects.

Projection-based methods overcome the main problems with maps, *i.e.*, sensitivity to noise and dependence of pose-normalization. The robustness to noise is achieved by using only low-frequency shape information to form the descriptors. The rotation invariance is achieved by using only information about the energy content of the shape. However, projection-based methods are not as discriminative as maps as they filter out high-frequency information. These methods also suffer from numerical instabilities.

In section 2.4.1, we provided a brief overview of feature point extraction techniques for use in partial shape matching. The two main criteria used for evaluating the performance of a feature point extraction technique were *repeatability* and *coverage* of the extracted points on the surface. Feature point extraction techniques were categorized into three classes, based on how feature points were selected on surfaces.

The first class of feature point extraction techniques consist of the approaches, which use local geometric properties of the surface, such as mean and Gaussian curvatures to select the feature points. The main drawback of these approaches is their dependence of curvature computations, which are generally sensitive to noise.

Perceptual-based feature point extraction techniques process the surface is a multiscale manner and recover information about both the location and scale of the feature points. The multiscale nature of these approaches also makes them stable against noise.

Statistical feature point extraction techniques use local shape descriptors to only select points, which are statistically rare. The domain on which this statistical rarity is defined can be limited to just one model or include an entire database of models. The latter approach has been shown to perform better in retrieval systems.

Class	Subclass	References	Local	Global	Rot. Inv.
Global shape properties	Symmetry	[58]	No	Yes	Yes
	Anisotropy	[57]	No	Yes	Yes
	Convex Hull-based	[27]	No	Yes	Yes
Maps	Voxelization	3D DFT [132]	No	Yes	No
		3D Zernike Moments [95]	No	Yes	No
		3D Shape Histograms [5]	Yes	Yes	No
		3D Shape Contexts [11]	Yes	Yes	No
	Spherical maps	EGI [54]	No	Yes	No
		Spherical Extent Function [106, 131]	No	Yes	No
		Concentric Spheres [131, 37]	No	Yes	No
	Planar maps	Spin Images[50]	Yes	Yes	Yes
		Projection onto tangent disk [70]	Yes	No	No
		Surface Signatures [142]	Yes	Yes	Yes
		Harmonic Shape Images [145]	Yes	No	No
	Curves	Point Fingerprints [122]	Yes	No	No
		Point Signatures [26]	Yes	No	No
Projections	Spherical Harmonics	[94, 106, 131, 37, 133]	No	Yes	Yes
	3D Zernike Moments	[95]	No	Yes	Yes
	3D DFT	[70]	No	Yes	Yes
		[132]	No	Yes	Yes
Histograms	Local properties	Curvature and Volume Index[129]	No	Yes	Yes
	Global feature distributions	Parameterized Statistics [96]	No	Yes	No
		Shape Distributions [97, 47]	No	Yes	Yes
		Cords [98, 99]	No	Yes	Yes

Table 2.2: Classification of methods for computing 3D shape descriptors.

Class	Pros	Cons
Global shape properties	• Robust to noise and shape deformations • Invariance to geometric transformations • Can be used to augment existing descriptors	• Not very discriminative
Histograms	• More discriminative than global shape properties • Robust against noise • Generally, rotation invariant	• Not discriminative enough for measuring similarity between objects in the same class
Maps	• Very discriminative • Local control • Multiscale support • Partial matching capabilities	• Not robust against noise • Generally, not rotation invariant
Projections	• Robust against noise • Built-in low-pass filtering support • Generally, rotation invariant	• Not as discriminative as maps • Could suffer from numerical instabilities

Table 2.3: Comparison of different classes of feature-based descriptors

Chapter 3

Survey of Scale-Space Based 3D Matching Techniques

"I decided that if I could paint that flower in a huge scale, you could not ignore its beauty."
–Georgia O'Keeffe

As mentioned in the previous chapters, in this work we study the problem of partial shape matching on sampled surfaces obtained from 3D acquisition devices. One of the main artifacts of the data obtained from these devices is the noise in the positions of the surface vertices. Depending on the task at hand, the noise in the data may be dealt with using various approaches. For example, in the case of local descriptor computations or feature extraction, the corresponding surface may be smoothed prior to any computations. Alternatively, the feature extraction process may be designed in a manner which is insensitive to noise. For example, integral invariants [80, 39] estimate surface curvatures with the help of integration over a finite region on and around the surface; integration, unlike differentiation, is well-posed and less sensitive to surface noise.

In Chapter 4, we discuss the Moving Least Squares (MLS) approach [67, 3] for reconstructing 3D surfaces and handling the noise in the data by fitting smooth functions to local neighborhoods around surface points. We also show how the noise affects the normal

and curvature computations in the fitting procedure. In this chapter, we review scale-space based representations for 3D surfaces, which serve as another way of dealing with noise in the data. We also study the problem of feature selection on 3D surfaces for use in matching applications.

It is not possible to discuss feature selection (or identification) without addressing the issue of scale selection, which refers to the task of deciding the size of a feature on the surface. This size may vary or be constant across the surface. For example, when computing differential attributes of the surface, such as normals or curvatures, the size of the neighborhood used to compute the entities defines the scale of the operator. The operator size required to compute these entities defines the size of the structures on the surface.

The problems of dealing with noise and scale are not unique to surfaces. They appear in the data obtained from any measurement device. The scale-space theory of images and signals has been developed in an attempt to address these issues. The theory has become quite mature over the past few decades and its applications have been shown in many real-world problems with some impressive results. The main advantages of using such a representation are:

1. **Robustness:** the multiscale nature of the representation ensures the robustness of the approach to noise.

2. **Regularization property:** the use of the Gaussian kernel and its derivatives to generate the representation allows differentiation operations on the signal to be performed in a well-defined manner.

3. **Automatic scale selection:** besides allowing for robust extraction of different differential structures that may be present in the signal, the automatic scale selection principle of the scale-space theory enables one to obtain information about the *size* of the extracted structures.

The scale-space theory has been extensively studied in the case of images (signals) in

(a) (b)

Figure 3.1: Blob detection with automatic scale selection [71]: (a) input image, and (b) detected locations and sizes of blob-like structures in the image.

$\mathbb{R}^n$. It has been shown that besides having nice theoretical properties, it can be implemented efficiently [77, 73, 21]. However, there is a lack of extensions of the scale-space representation to 3D surfaces. This work attempts to fill this gap by proposing a new, simple scale-space representation for surfaces which can in turn be used in matching applications.

3.1 Scale-Space Theory for Signals in $\mathbb{R}^n$

In this section, we provide a brief overview of the scale-space theory for signals in $\mathbb{R}^n$. Sec. 3.1.1 and Sec. 3.1.2 provide motivation and background for scale-space theory of signals. A formal definition of the scale-space representation of signals in $\mathbb{R}^n$ and some of its properties are given in Sec. 3.1.3.

3.1.1 Why Scale-Space?

The scale-space approach to analyzing images is biologically-inspired by the multiscale nature of the human visual system. Given the superior performance of vision in humans and mammals, it is sensible to mimic the same process for analyzing images in computer vision applications. The front-end vision in humans processes images formed in the retina in a multiscale fashion. This is necessary since the size of the real-world structures present in images are not known *a priori*. As a result, front-end vision, as an uncommitted system, needs to consider all possible scales.

Observations in the physical world are measurements done by way of integration. For example, in order to see, one needs to measure the intensity of light received from all sources by the receptor fields in the eye. The same comments apply to the formation of a picture in a camera, which is formed by the integration of photons on a CCD array. The measurements are performed by integrating the perceived light intensity over a finite area. Therefore, the intensities are averaged (blurred) over these finite areas. Larger areas result in smoother perceived signals (images). That is, the receptor fields, in their simplest form, act like low-pass filters. It has been shown that the profile of the receptor fields in the retina resemble Gaussian filters of various sizes and differential orders. The size of a receptor field determines the standard deviation of the Gaussian used to filter the image. In his book [127], Romeny makes the following statement: "we blur by looking". However, we feel an even stronger statement can be made here: *we see by blurring*.

The *inner* and *outer* scales of a measurement device, respectively, correspond to the sizes of the smallest and largest structures it can measure. For example, the inner scale of a camera is determined by the size of each of its CCD elements, and its outer scale is determined by its field of view. The range of structure sizes a measurement device can capture is therefore, limited by its inner and outer scales.

Consider the sunflower field in Fig. 3.1(a). The image contains sunflowers of various sizes. When we look at the image, we are able to immediately extract some crucial information about its contents. For example, at first glance we realize that the image contains many similar blob-like features of varying size. A question that arises is how much of this information is extracted in the earliest stage of our vision system – our front-end vision system [127]. It has been shown that the human front-end vision system is able to extract the information mentioned above by representing the image in a multiscale manner and performing hierarchical feature extraction. This is contrary to earlier beliefs that *a priori* information/experience is needed in order for us to reduce the image to its most elementary components (features). As mentioned earlier, the human visual system obtains this multi-

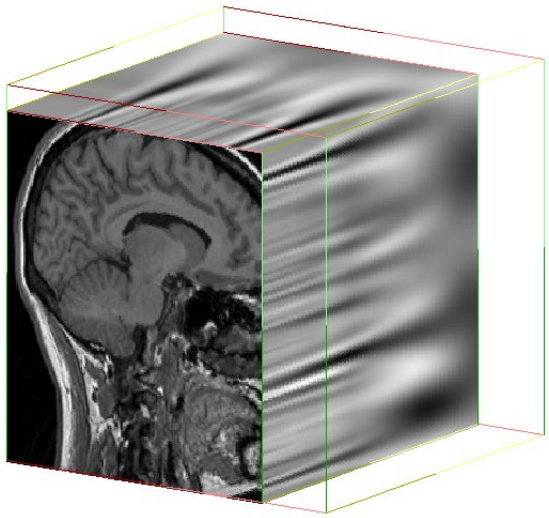


Figure 3.2: Scale-space representation of an image [127]. (Image courtesy of Bart M. ter Haar Romeny)

scale hierarchical representation of images by using receptor fields in the retina of various sizes and shapes.

The individual sunflowers in Fig. 3.1(a) can be extracted using a second order differential operator such as the Laplacian. The main difficulty with using such an operator, however, is choosing the right scale –the finite neighborhood size on which the operator acts. If the scale is too small, small features corresponding to noise may be selected. If the scale is too large, small flowers may fail to be detected. One solution to this problem is to consider all scales simultaneously, and then decide which features to pick. Fig. 3.1(b) shows the result of the application of a multiscale feature detector with automatic scale selection applied to the sunflower field (Sec. 3.1.2).

As the need for multiscale analysis of images becomes apparent, the next question that arises is which smoothing operator to use to build the stack of the increasingly smoothed images. Keeping in mind that the resulting representation should correspond to an "uncommitted" front-end vision system, the representation should satisfy a few axioms. The following *scale-space axioms* have been proposed for signals [127]:

- **Invariance:** the representation should be invariant under various transformations that the signal might undergo; *e.g.*, rotation, translation, scaling.
- **Linearity:** the smoothing process should be independent of any structures that may be present in the signal.
- **Non-spurious detail generation:** no new features should appear at the coarser levels in the stack. Additionally, the local extrema should not be enhanced; the value at local maxima should continuously decrease and the value at local minima should continuously increase.

It has been shown that the Gaussian is the unique kernel that can be used to generate a representation which satisfies these axioms [33, 127]. In fact, the profile of the receptor fields in humans and other mammals resemble Gaussian kernels of various sizes and differential orders.

The scale-space representation of an image adds an additional dimension to the representation, referred to as the scale dimension. As the scale increases, the image becomes coarser (smoother). The smoothing is done by convolving the original image with Gaussian kernels of increasing size. Fig. 3.2 shows an example of the scale-space representation of an image.

3.1.2 Scale-Space Theory, Feature Extraction, and Automatic Scale Selection

Dealing with scale changes in images is an important issue in computer vision. Automatic scale recovery enables design of scale-invariant descriptors. These descriptors, in turn, make the matching process invariant to scale changes in images.

Simultaneously looking at all possible scales of an image has many advantages. The main advantage is the elimination of the need to know the size of the structures we seek in the image. For example, the Laplacian operator may be used to find blob-like structures in

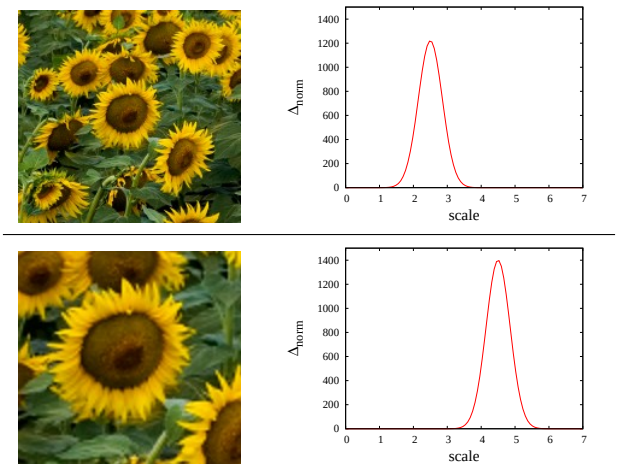


Figure 3.3: Automatic scale selection at an interest point [71].

images. The main problem with using such an operator, however, is choosing its size. If the neighborhood size on which the operator acts is too small, many of the extracted features will correspond to the noise in the image. On the other hand, if an operator is used that is too large, many small features will be missed in the extraction process. However, by using all possible operator sizes at each point and selecting the scale, which yields the largest operator response, a scale (neighborhood size) can be associated to each extracted feature point in the image. Consequently, ignoring all features with small associated scales, will not only yield stable features, but also provide the extra information about the size of the features.

Fig. 3.3 shows an example of the automatic scale selection principle. The graphs plot the magnitude of the scale-normalized Laplacian at the image centers as a function of scale. As the scale is increased, the magnitude of the scale-normalized Laplacian first increases and then decreases. The scale at which the maximum magnitude is reached corresponds to the associated scale at the point of interest. Fig. 3.1 demonstrates the application of the approach to blob detection with automatic scale selection [71] on an image of a sunflower field.

Properties of the scale-space representation of signals has been used in practical ap-

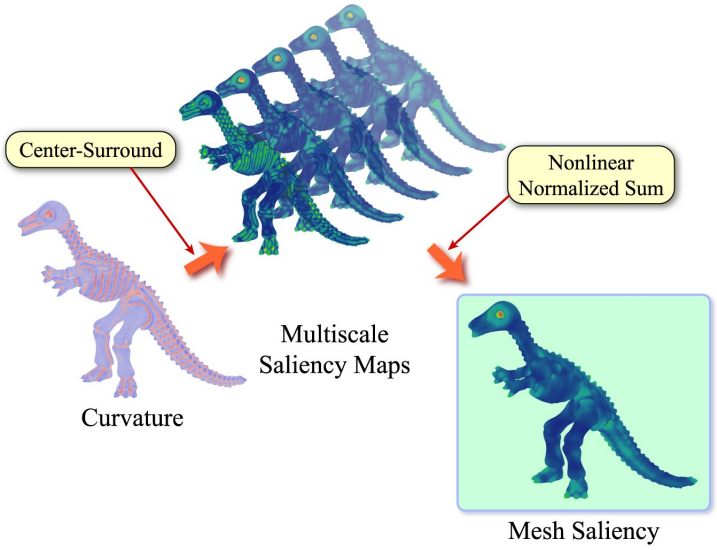


Figure 3.4: Mesh saliency computation [63]. (Image courtesy of Chang Ha Lee)

plication in computer vision with some impressive results [77, 9, 21]. For example, [77] introduces the Scale Invariant Feature Transform (SIFT), which can be used to extract stable feature points and local descriptors on images in an efficient manner. [21] shows an application of SIFT features in automatic stitching of panoramic images.

Feature points can generally be defined in terms of some combination of the derivatives of the input signal. The scale-space representation of signals allows for robust computations of these differential operators. Additionally, these operators can be used for orientation assignment and local descriptor construction.

Applications and extensions of the scale-space theory to 3D surfaces have also produced interesting results. One may find different definitions for the notions of saliency, salient regions and points [134, 38, 35, 63, 111]. Here, we are interested in the perceptual notion of saliency in humans. That is, the regions or points in images (or surfaces) that stand out or attract the attention of the human observer.

In [63], the authors introduce the notion of mesh saliency. They propose that local variations in mean curvature values on surfaces correspond to regions with interesting features. Mesh saliency is then defined in terms of a non-linear sum of the approximate Laplacian

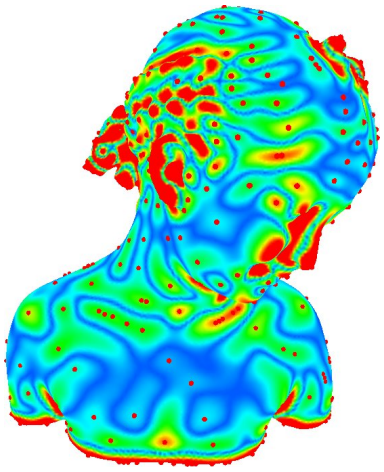
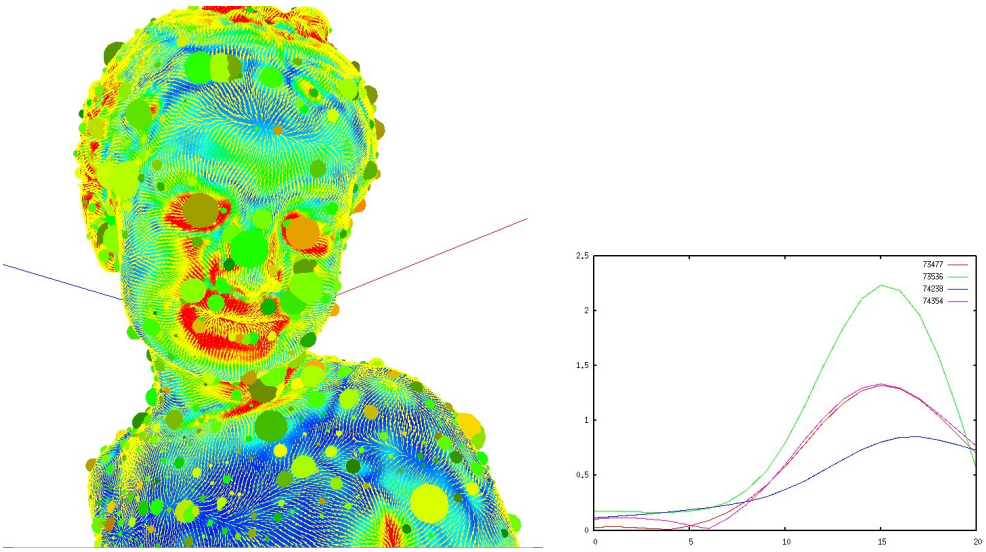


Figure 3.5: Feature extraction on a 3D model using the estimated Laplacian of curvatures.

of vertex curvatures at various scales. The saliency measure is computed by first estimating the surface mean curvatures at mesh vertices. The estimated curvature values are then repeatedly smoothed by computing the Gaussian weighted average in neighborhoods of increasing sizes. This yields a stack of meshes with increasingly smoothed curvatures. The absolute value of the difference between consecutive levels in the stack is then computed to obtain a second stack. Saliency at each vertex on the mesh is defined in terms of a non-linear sum of the values on the second stack (difference of Gaussian) across the scales (Fig. 3.4). The authors show applications of the proposed mesh saliency to mesh simplification and automatic view point selection.

It is noteworthy to mention that since the absolute values of the difference of the Gaussian smoothed curvatures estimate the scale-normalized Laplacian of curvatures, the maxima of these values can be expected to correspond to blob-like structures on the surface. This is confirmed in Fig. 3.5. Additionally, the principle of automatic scale selection may be extended to 3D surfaces to associate a scale to the extracted features on the surface (Fig. 3.6).



(a) (b)

Figure 3.6: Automatic scale selection on 3D surfaces. (a) The radius of the spheres on the model correspond to the associated scale at the feature point. (b) Plot of the estimated scale-normalized Laplacian of mean curvatures at 4 vertices on the mesh. The peaks in the plots correspond to the detected scales at the corresponding vertices.

3.1.3 Scale-Space Representation of Signals in $\mathbb{R}^n$

The linear scale-space representation of a continuous signal $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is defined as the solution to the heat (diffusion) equation [71]

$$\partial_t F = \Delta F = \sum_{i=1}^{n} \partial_{x_i x_i} F \ , \tag{3.1}$$

where $F : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$, with the initial condition $F(x; 0) = f(x)$; $x \in \mathbb{R}^n$ and Δ denotes the Laplacian. It can be shown that the Gaussian is the fundamental solution to the above diffusion equation [71]. That is, the scale-space representation of f can equivalently be defined as

$$F(x; \sigma) = g(x; \sigma) * f(x) \tag{3.2}$$

where $*$ denotes convolution, $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is the n-dimensional normalized Gaussian with standard deviation σ: $g(x; \sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\|x\|^2/2\sigma^2}$. The Gaussian is generally reparametrized

in terms of the scale parameter $t = 2\sigma^2$. Eq. (3.2) then becomes:

$$F(x;t) = g(x;t) * f(x), \tag{3.3}$$

with, $g(x;t) = \frac{1}{(\pi t)^{n/2}} e^{-\|x\|^2/t}$.

The above Gaussian kernel has the following properties:

- *Isotropy:* the kernel acts the same in all spatial directions.
- *Separability:* an nD Gaussian kernel can be obtained from n multiplications of 1D Gaussians:

$$g(x_1, \ldots, x_n; t) = g(x_1; t) g(x_2; t) \ldots g(x_n; t) \; . \tag{3.4}$$

- *Normalization:* the area under a Gaussian with any width (scale) is one:

$$\int_{-\infty}^{\infty} g(x;t) dx = 1 \; . \tag{3.5}$$

- *The Fourier transform of a Gaussian is a Gaussian:* $G(\omega;t) = \frac{t^{n/2}}{(2\pi)^{n/2}} e^{-\|\omega\|^2 t}$, where G denotes the Fourier transform of g.
- *Self-similarity:* convolving two Gaussians with scales t_1 and t_2, yields another Gaussian with scale $t_1 + t_2$: $g(x;t_1) * g(x;t_2) = g(x;t_1 + t_2)$
- *Central limit theorem:* the kernel resulting from the repeated convolutions of *any* kernel with itself, on the limit, approaches a Gaussian.

Additionally, convolution with the Gaussian derivatives has the nice property of regularizing the differentiation operation. Regularization refers to the process of making an ill-posed operation well-posed. A problem is said to be ill-posed if it does not have a stable solution. For example, the task of differentiating discrete data (or any data with discontinuities) is ill-posed. The differentiation operation can become well-behaved by filling the discontinuities in the data; *i.e.*, smoothing the data.

Using the properties of convolution, spatial derivatives of F at scale t can be robustly computed by convolving the signal with Gaussian derivatives at scale t,

$$\begin{aligned} \partial_{x_1{}^{\alpha_1}\ldots\, x_n{}^{\alpha_n}} F(x;t) &= \partial_{x_1{}^{\alpha_1}\ldots\, x_n{}^{\alpha_n}} F(x;t) \\ &= \partial_{x_1{}^{\alpha_1}\ldots\, x_n{}^{\alpha_n}} (g(x;t) * f(x)) \\ &= (\partial_{x_1{}^{\alpha_1}\ldots\, x_n{}^{\alpha_n}} g(x;t)) * f(x)\ . \end{aligned} \tag{3.6}$$

The convolution with Gaussian derivatives regularizes the differential operator rather than the data. This is a desirable behavior as the process does not change the data (*e.g.*, by smoothing it), and instead modifies the operator.

Two additional properties of the scale-space representation of signals are:

1. The local extrema of the signal are not enhanced at coarser levels. That is, in general, the values of the local maxima of the signal (and its derivatives) decrease, while the values of the local minima increase.

2. No new structures are created at the coarser levels. That is, no new local extrema appear at the coarser levels.

The two properties always hold for 1D signals but may not always be true for higher dimensional signals (for more information, refer to [127]).

The non-enhancement property of the scale-space representation of signals, in general, guarantees that the value of the local maxima (minima) decrease (increase) as the signal is smoothed. This is expected since, convolution with a Gaussian attenuates the magnitude of high frequency contents of the signal. However, the amplitude of the spatial derivatives of the signal may be *scale normalized* using the change of variable $v = \frac{x}{t^{\gamma/2}}$, in the original scale-space formulation. With this change of variable, the Gaussian kernel becomes: $g(v;t) = \frac{1}{(\pi t)^{n/2}} e^{-\|v t^{\gamma/2}\|^2/t}$. With $\gamma = 1$, we have $g(v;t) = \frac{1}{(\pi t)^{n/2}} e^{-\|v\|^2}$. That is, this reparametrization of the spatial domain in terms of the scale parameter results in a Gaussian kernel with constant width across scales. Note that the amplitude of the Gaus-

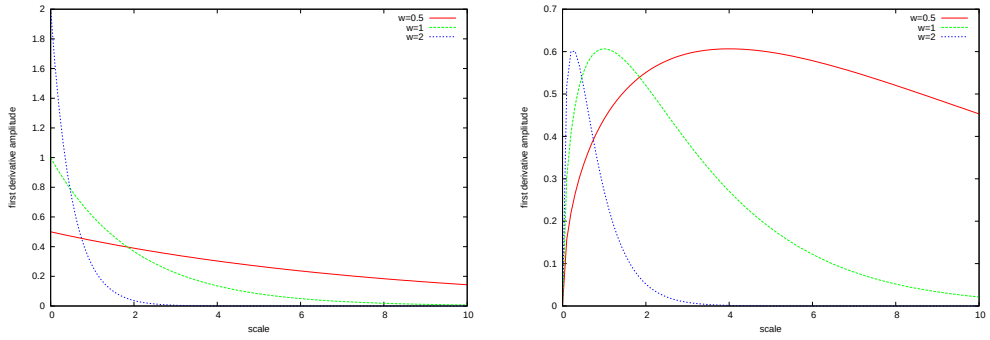
sian still decreases as the scale increases. Intuitively speaking, at each level, the Gaussian kernel smoothes the signal by attenuation the magnitude of the high frequency content of the signal and distributing it among the lower frequencies. That is, in the spatial domain, the information contained near each point is dissipated to a larger neighborhood. With the change of variable, we take larger steps at the coarser scales to contain (keep) the same amount of information that was dissipated by the smoothing operation at the finer scales. This change of variable yields the following scale normalized spatial derivatives of the signal:

$$\partial_{v^m} F_v(v;t) = t^{m\gamma/2} \partial_{x^m} F(x;t) \ , \tag{3.7}$$

m denotes the order of differentiation.

Fig. 3.1.3(a) plots the amplitude of the maximum value of the first derivative of the scale-space representations of sinusoidal signals with frequencies $\omega = 0.5, 1, 2$: $f(x, \omega) = sin(\omega x)$ as functions of scale. As expected, in all cases, the amplitude of the maximum decreases as the scale increases. Fig. 3.1.3(b) shows the plots of the scale-normalized first derivatives of the signals as a functions of scale. Note how the amplitudes of the scale-normalized derivatives first increase and then decrease. The scale at which the maximum amplitude is reached is proportional to the frequency of the signal. The principle for scale selection states [71]:

> *"In the absence of other evidence, assume that a scale level, at which some (possibly non-linear) combination of normalized derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data."*



(a) (b)

Figure 3.7: (a) plots of the amplitudes of the first spatial derivative of a sinusoidal signal as a function of scale; with frequencies $\omega = 0.5, 1, 2$ (b) plots of the amplitudes of the *scale-normalized* derivatives of the same functions [71].

3.2 Scale-space based approaches to 3D surface processing and feature extraction

In this section, we review the current scale-space extensions to 3D surfaces in the literature, which have been proposed for feature extraction and matching. Extending the definition of the scale-space theory to surfaces has proven to be difficult. The main difficulties with such an extension are:

1. **Representation issues:** there are multiple ways of representing 3D surfaces. But no one representation is as simple as the regular grid representation of nD images and none allow efficient ways of developing tools/techniques for frequency analysis on the surface.

2. **Geometric degeneracies:** it is well-known that smoothing the 3D geometry of curves and surfaces, except in special cases, results in degeneracies such as singularities, self-intersections, and shrinkage of the original surface.

The evolution scheme corresponding to the scale-space representation of the surface will depend on the representation of the input surface. Computation and storage constraints

dictated by most 3D matching problems, generally allow these surface representations to be used:

- **Point clouds:** only the 3D coordinates of a set of sampled points on the original surface is provided with no connectivity information.
- **Polygonal meshes:** both vertex positions and connectivity information between neighboring vertices (*i.e.*, faces and edges) is provided.
- **Parametrized surfaces:** position, connectivity information, and also a global parametrization of the surface is provided.
- **Voxelization:** the 3-space around the surface is voxelized, and the surface is represented as a 3D image.
- **Implicit surfaces:** the zero level set of a local or global implicit function is used to represent the surface; *e.g.*, Radial Basis Functions (RBFs).

Over the past decade, different approaches have been proposed, which try to mimic the scale-space representation for 3D surfaces. Depending on how the surface is represented, the scale-space representation may be defined differently. The proposed techniques aim at satisfying some of the properties present in scale-space representation of signals. However, no one technique has been able to successfully extend the scale-space theory of images to surfaces.

The most straightforward approach, as proposed by [94], is to obtain a 3-D image from the original surface by voxelizing the space inside the bounding box of the surface. Each voxel will then have value 1, if the surface passes through it and 0, otherwise. Another alternative is to give each voxel a value in the range $[0, 1]$, depending on its distance from the surface. The voxelized model can then be processed as a 3D image. That is, the scale-space representation of this voxel space is obtained by repeatedly convolving the original image with 3 dimensional Gaussian kernels of increasing sizes. The main drawbacks of this

approach, however, are: 1. loss of precision due to the voxelization. 2. The requirement for the surface to be closed prior to the voxelization. 3. If the method is used for feature point extraction, the extracted points generally do not lie on the surface.

Another approach is to first parametrize the original surface. The parametrization yields a function, $f(u, v) = (x, y, z)$, which maps 2D positions on the plane to 3D points on the surface. f can then be treated as a multi-channel image. Here, each channel corresponds to the positions of the surface vertices along each of the X, Y, and Z axes. Each channel can then be smoothed separately by convolving f with a 2D Gaussian. The problem with this approach, however, is the introduced distortions resulting from the parametrization. To reduce the effects of the parametrization distortion, the authors of [93] propose to use a distortion map which allows estimating the distance between two points (u_1, v_1) and (u_2, v_2), as the geodesic distance between the corresponding 3D points on the surface, $f(u_1, v_1)$ and $f(u_2, v_2)$. In order to incorporate the distortion map in the scale-space representation, the definition of the Gaussian kernel is modified so that it makes use of the distortion map. However, the estimate of the geodesic distance between two points using the distortion map is very rough and inaccurate—the cost of properly computing such a distortion map is $O(n!)$.

The Laplace-Beltrami operator can be used to estimate the Laplacian of a discrete signal of a surface. The scale-space representation of a signal is defined as the solution to the diffusion equation

$$\partial_t F = \lambda \Delta F \quad (0 < \lambda \leq 1) \,. \tag{3.8}$$

Treating the 3D positions of the mesh vertices as the surface signal, the Laplace-Beltrami operator at each vertex gives the positional gradient at the vertex at time t, in the geometric evolution process. Moving the vertices in this manner is the equivalent of Gaussian smoothing for 3D surfaces. Mean curvature flow, which is closely related to surface diffusion, may also be used to smooth the surface. Under mean curvature flow, each vertex is moved along its normal proportional to estimated mean curvature at the vertex. [109]

uses a modification of this approach to obtain a scale-space representation of the original surface and shows how it can be used to perform feature extraction and automatic scale selection on closed 3D models. A major problem with this approach, however, is the geometric degeneracies that generally arise when the smoothing operation is iterated many times. The first type of degeneracy is the convergence of the surface to a point (shrinkage). The second type arises at sharp edges or thin and elongated structures on the surface. Few approaches have been suggested [28, 126, 109] that deal with the shrinkage problem. The modifications result in representations which no longer approximate the linear diffusion process. The degeneracies produced at sharp edges and elongated structures seem to be present in all smoothing algorithms that directly smooth the geometry of the surface. We are not aware of any solutions to this problem.

Instead of smoothing the geometry, functions (signals) defined on the surface may be smoothed. For example, in [63], surface mean curvatures are repeatedly smoothed by computing the Gaussian weighted average of the neighboring vertices. This yields a stack of increasingly smoothed curvatures. The absolute value of the difference between smoothed curvatures at consecutive scales is then computed to obtain a second stack of values. The authors define *mesh saliency* at a vertex, as a non-linear combination of these differences along the scale axis of the stack. They show applications of mesh saliency to mesh simplification and viewpoint selection.

In [100], the authors define surface variation at a point on a 3D surface as the ratio of the smallest eigenvalue to the sum of all eigenvalues of the covariance matrix of the 3D positions of the local neighborhood:

$$\sigma_n(p_i) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \tag{3.9}$$

where $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the eigenvalues of the covariance matrix, C, at point p_i. n corresponds to the number of nearest neighbors to p_i used to compute C. A stack of surface

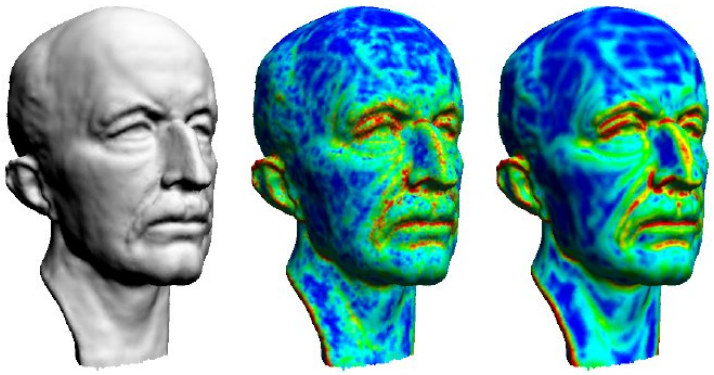


Figure 3.8: Surface variations computed on the Max Planck model at two different scales [100]; the surface variations on the rightmost figure were computed at a coarser level. (Image courtesy of Mark Pauly)

variations can be obtained by increasing n_i. The authors define the local maxima of the surface variations as their features. Larger neighborhood sizes correspond to coarser features. Fig. 3.8 shows an example of the computed surface variations on the Max Planck model at two different scales.

In [101], the authors use the MLS projection operator to repeatedly smooth a point-set surface. In their proposed method, the projection operator is used to project surface points onto the underlying MLS surface. This result is a new set of points whose underlying MLS surface is smoother than the original point set. Repeating this process with increasing neighborhood sizes used to define the projection operator, a series of smoother surfaces is obtained. The set of smoothed point-sets together with the length of the displacement vectors connecting consecutive levels defines the scale-space representation for the surface. The authors show applications of their representation to 3D morphing, surface deformation, and emulated frequency filtering on point-set surfaces. Because of the cost of computing the MLS fit at each vertex, the authors propose the surface be subsampled after each smoothing operation. Despite this, however, the computational cost of this approach is reported to be rather high.

More recently, the Heat Kernel Signature (HKS) [120] has been used in global shape matching tasks involving 3D models that may have undergone isometric deformations. In

this approach, the heat operator or the heat kernel of a surface is studied to infer information about the geometry of the surface. A scale-invariant version of HKS was also introduced in [20] and used for non-rigid 3D shape retrieval. The main drawbacks of the approach are the computation times and their inability to perform automatic scale selection, which is required in partial shape matching tasks. In [120], it is reported that the overall time required to compute the HKS on a surface with $100K$ vertices is approximately 90 minutes. This computation time puts major constraints on the practical applications of the technique.

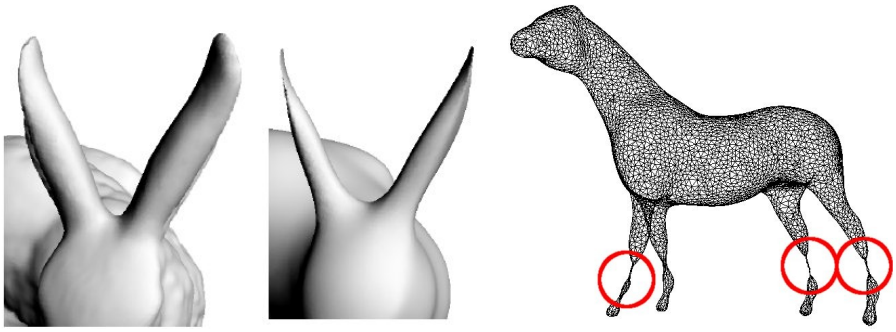
3.3 Summary

In this chapter, we provided motivation for use of the scale-space theory for feature extraction on 3D surfaces. In Sec. 3.1.3, we briefly reviewed the scale-space theory for signals in $\mathbb{R}^n$. In Sec. 3.2, we reviewed related work on the extensions of the scale-space representation to 3D surfaces. In this section, we summarize the shortcomings of the current scale-space representations for 3D surfaces in terms of efficiency, geometric distortions, and scale-space axiom violations.

Efficiency: In order to obtain a complete scale-space stack, the input models need to be smoothed to very coarse levels. Therefore, the efficiency of the representation is heavily influenced by the cost of the smoothing operation used at each level.

The techniques that employ the MLS projection operator as the smoothing operator (*e.g.*, [101, 70]) are very slow since, at each level, a local surface fit needs to be computed at each vertex. The efficiency is improved by subsampling the surface after each smoothing operation. However, despite these improvements, these techniques are generally much slower than the other approaches.

The approaches where the surface is first parametrized [93] or voxelized [94] are the fastest among the mentioned approaches. However, the parametrization approach distorts the initial surface (even though an approximate distortion map is used). The voxelization



(a) (b)

Figure 3.9: Geometric degeneracies developed after smoothing surfaces using a diffusion flow approach. (a) curvatures are increased in the ears regions after smoothing [100] (b) elongated structures becoming too thin (collapsing) [109]. (Images courtesy of (a) Mark Pauly, and (b) Markus Schlattmann)

approach simplifies the surface too much and the approach results in a much fewer features compared to the other techniques.

The limitation on the maximum step size used in each iteration of the Laplacian smoothing of [109] makes the approach slow on large surfaces. This is due the fact that at coarser scales, a large number of smoothing iterations are needed to get any noticeable smoothing effect. The authors report computation times of more than 2 hours for meshes with 10K+ vertices.

Mesh saliency approach of [63] and surface variation approach of [100] both use 3D query data structures such as kD-trees and Octrees to perform nearest neighbor queries. The queries need to be performed at each vertex on the surface, and at each level of the stack. These operations at coarser levels become computationally involved and make the application of these approaches to large surfaces nearly infeasible. Note that the original approaches do not propose to (neither need to) build a complete and densely sampled scale-space stack. The stack is either very coarsely sampled or only a few levels are constructed at the finer scales. However, the intended applications of the approaches are different than ours (feature extraction and matching).

Geometric distortions: When surfaces are evolved in a manner consistent with the diffu-

sion process, the "flow" of the vertices shrinks the area or volume of the surface. Therefore, any *linear* scale-space based approach which directly manipulates the vertex positions will result in shrinkage and/or other types of geometric degeneracies (*e.g.*, see Fig. 3.9). Among the approaches, the mesh saliency [63] and surface variation [100] approaches do not directly smooth the vertex positions and are immune to such problems.

Scale-space axiom violations: The approaches may also be evaluated in terms of which scale-space axioms they violate:

- *non-spurious detail generation:* the geometric degeneracies and creation of new high-frequency content which are present in all geometric surface evolution approaches are in violation of the non-spurious detail generation axiom.

- *linearity:* the MLS projection operator of [101] and volume preserving approach of [109] violate the linearity requirement of the scale-space representation; *i.e.*, they do not approximate the linear diffusion equation.

- *invariance:* the smoothing operations in all the proposed approaches are spatially symmetric—they are not biased towards any specific orientation, position, or scale on the input models.

For a comparison of the different scale-space representations for 3D surfaces, see Table 5.1.

Chapter 4

Error Propagation on Reconstructed 3D Surfaces

"I don't particularly like Husserl's way—long and difficult. He tells us no detailed way about how to do it."

–Kurt Gödel

One may find different definitions for surface reconstruction in the computer graphics literature. Even polygonal meshes are sometimes considered to represent reconstructed surfaces; approaches such as *discrete differential geometry* [15] are applied directly to polygonal meshes to estimate differential surface attributes without any need for obtaining a continuous representation of the surface. In this work, however, by reconstruction we refer to the task of obtaining a $\mathbf{C}^k$ smooth representation of the surface, for some order k. Given such a representation, local surface attributes such as normals and curvature values can be estimated directly.

A crucial question that arises is how the estimated surface properties (attributes) are affected by the noise in the input surface. In this chapter, we study how the noise in the input surface is propagated to the estimated surface normals and curvatures. Additionally, we use this information to verify the implementation of the reconstruction algorithm using

statistical hypothesis testing. Moreover, we show how the hypothesis testing procedure may be used to obtain information about the minimum neighborhood size needed to estimate the differential surface attributes.

We assume the input surface is given in the form of a point cloud. We use the Moving Least Squares (MLS) approach of [67, 3] to obtain a smooth approximation to the underlying surface represented by the input point-set. The main two advantages of using MLS are

- The inherent robustness of the least squares approach to noise
- The ability to easily track the noise in the data across different stages of computation; the term *error propagation* is used to refer to this process.

This chapter is organized as follows. In Sec. 4.1, we review the MLS surface reconstruction and discuss normal and curvature estimation. In Sec. 4.2, we show how the error in the input surface is propagated to the estimated normals and curvature values. Finally, in Sec. 4.3, we discuss how the information we obtain about the propagated error can be used to determine the required neighborhood sizes when reconstructing the surface.

4.1 Moving Least Squares Surfaces

Let $\mathbf{P} = \{\mathbf{p_1}, \cdots, \mathbf{p_N}\}$ denote the input point-set. Under the assumption that the underlying 3D surface represented by the point-set is a 2D manifold, small neighborhoods around each point $\mathbf{p_i} \in \mathbb{R}^3$ on the surface can be parametrized and represented in Monge's form, $z = g(x, y)$; *i.e.*, in the form of a height map. $g : \mathbb{R}^2 \rightarrow \mathbb{R}$ is a smooth function whose domain is the tangent plane at $\mathbf{p_i}$. Function g is estimated in two steps:

1. A local reference frame is constructed at $\mathbf{p_i}$ by minimizing an error functional E_H in the least squares sense.

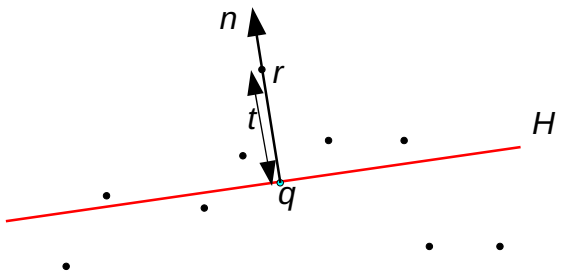


Figure 4.1: Estimated projection plane near interest point, $\mathbf{r}$. $\mathbf{n}$ is a normal vector to H passing through $\mathbf{r}$, t is the signed distance of $\mathbf{r}$ from H, and $\mathbf{q}$ is the projection of $\mathbf{r}$ onto H along $\mathbf{n}$.

2. Using the constructed reference frame, the local neighborhood around p_i is parametrized such that the coordinates of all the neighboring points are represented in the form of a height map. A bi-variate polynomial of degree m is then fitted to the discrete height map to obtain the smooth map g.

In the following sections, we review each step in more detail.

4.1.1 Constructing the Local Reference Frame

The objective of this step is to find a reference frame at an interest point $\mathbf{r} \in \mathbb{R}^3$ in which the coordinates of all the neighboring points on the surface can be represented. Infinitely many such frames exist at each point. However, since the eventual goal is to use the frame to define a height map of the local neighborhood of $\mathbf{r}$, one of the basis vectors must be parallel to the surface normal at the point of interest. This guarantees that the resulting height values have the smallest variance among all possible bases; small variance in the height values, in turn, means that the resulting discrete height map can be more accurately estimated by a smooth function.

The reference frame is constructed by first finding plane H, with unit normal $\mathbf{n}$, near the interest point $\mathbf{r}$. Let the associated neighborhood of vertices near $\mathbf{r}$ be denoted by set $\mathcal{N}(\mathbf{r}) \subseteq \mathbf{P}$. Let $\mathbf{n}$ be the unit normal to H passing through $\mathbf{r}$, t be the signed distance of $\mathbf{r}$

from H, and q be the projection of $\mathbf{r}$ onto H along $\mathbf{n}$ (Fig. 4.1). The unknown parameters of H (unit normal vector $\mathbf{n}$ and center $\mathbf{q}$) are found by minimizing

$$E_H = \sum_{\mathbf{p_i} \in \mathcal{N}(\mathbf{r})} \langle \mathbf{n}, \mathbf{p_i} - \mathbf{q} \rangle^2 \, \phi(\|\mathbf{p_i} - \mathbf{q}\|) \, , \tag{4.1}$$

where $\phi(\cdot)$ is a non-negative, monotonically decreasing weight function. Note that the objective function depends on projection of the interest point onto H, $\mathbf{q}$, which is an unknown. This makes the above formulation a non-linear optimization problem. The reasoning for why the projection of the interest point is used in Eq. (4.1) rather than the interest point itself is given in [67].

By substituting $\mathbf{q} = \mathbf{r} - t\mathbf{n}$ in Eq. (4.1), we obtain

$$E_H = \sum_{\mathbf{p_i} \in \mathcal{N}(\mathbf{r})} \langle \mathbf{n}, \mathbf{p_i} - (\mathbf{r} - t\mathbf{n}) \rangle^2 \, \phi(\|\mathbf{p_i} - (\mathbf{r} - t\mathbf{n})\|) \, . \tag{4.2}$$

The solution to the above minimization problem is estimated by first assuming $t = 0$ (*i.e.*, assuming the plane is centered at $\mathbf{r}$), and solving for the normal direction, $\mathbf{n}$. Once $\mathbf{n}$ is found, it is placed back in Eq. (4.2) and t is found using a gradient descent method.

When $t = 0$, Eq. (4.2) becomes

$$E_H = \sum_{\mathbf{p_i} \in \mathcal{N}(\mathbf{r})} \langle \mathbf{n}, \mathbf{p_i} - \mathbf{r} \rangle^2 \, \phi(\|\mathbf{p_i} - \mathbf{r}\|) \, . \tag{4.3}$$

Let $k = |\mathcal{N}(\mathbf{r})|$, $\mathbf{Q}_{3\times k} = \begin{pmatrix} \mathbf{p_1} - \mathbf{r} \\ \vdots \\ \mathbf{p_k} - \mathbf{r} \end{pmatrix}^\top$, and $\mathbf{W} = (w_{i,j})_{k\times k}$ be a diagonal matrix with $w_{i,i} = \phi(\|\mathbf{p_i} - \mathbf{r}\|)$. Eq. (4.3) is then given in matrix form as

$$E_H = (\mathbf{n}^\top \mathbf{Q})\mathbf{W}(\mathbf{n}^\top \mathbf{Q})^\top = \mathbf{n}^\top \mathbf{Q}\mathbf{W}\mathbf{Q}^\top \mathbf{n} \, . \tag{4.4}$$

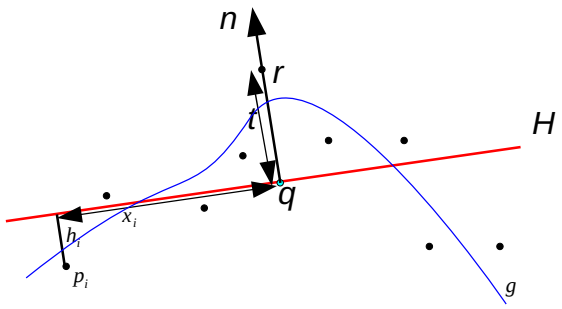


Figure 4.2: Fitting a bi-variate polynomial, g, to the discrete height values in the neighborhood of interest point, $\mathbf{r}$, which were obtained in the local reference frame construction step.

We are looking for $\mathbf{n}$ minimizing E_H, subject to $\mathbf{n}^\top \mathbf{n} = 1$. Taking the partial derivative of E_H with respect to $\mathbf{n}$ and setting the result to zero yields

$$\frac{\partial E}{\partial \mathbf{n}} = 2\mathbf{Q}\mathbf{W}\mathbf{Q}^\top \mathbf{n} = \mathbf{0} \, . \tag{4.5}$$

Therefore, the $\mathbf{n}$ minimizing Eq. (4.3) is the eigenvector corresponding to the smallest eigenvalue of the symmetric matrix $\mathbf{C} = \mathbf{QWQ}$. Note that $\mathbf{C}$ is the weighted scatter matrix of the points in $\mathcal{N}(\mathbf{r})$; *i.e.*, we used Principal Component Analysis (PCA) of the positions of the points on the surface near the interest point to estimate the normal direction for H.

Let $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}$ denote the eigenvectors of $\mathbf{C}$, with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$, respectively. This orthonormal vector set defines the local reference frame at the interest point, with $\mathbf{n} = \mathbf{u_3}$. The coordinates of each point $\mathbf{p_i} \in \mathcal{N}(\mathbf{r})$ in this frame are given as

$$\begin{pmatrix} x_i \\ y_i \\ h_i \end{pmatrix} = \begin{pmatrix} \langle \mathbf{p_i} - \mathbf{q}, \mathbf{u_1} \rangle \\ \langle \mathbf{p_i} - \mathbf{q}, \mathbf{u_2} \rangle \\ \langle \mathbf{p_i} - \mathbf{q}, \mathbf{u_3} \rangle \end{pmatrix} \, . \tag{4.6}$$

4.1.2 Smooth Local Map Construction

The procedure described in Sec. 4.1.1 enables us to represent the neighborhood around each interest point $\mathbf{r} \in \mathbb{R}^3$ in the form of a discrete height map. In this section, we describe how to obtain a smooth height map of the neighborhood by fitting a bi-variate polynomial of degree m. Different choices of m will have different consequences. If the degree of the fitted polynomial is too low, the resulting reconstructed surface will be over smoothed. On the other hand, polynomials with high degrees will suffer from overfitting and the resulting reconstructed surface generally tends to be ripply. Empirically, polynomial degrees of $m = 2$ and $m = 3$ seem to provide the best results [3]; we use $m = 3$.

Therefore, the smooth local map of the neighborhood around $\mathbf{r}$ is defined by the 3^{rd} degree bi-variate polynomial

$$g(x, y) = \beta_1 x^3 + \beta_2 x^2 y + \beta_3 x y^2 + \beta_4 y^3 + \beta_5 x^2 + \beta_6 x y + \beta_7 y^2 + \beta_8 x + \beta_9 y + \beta_{10} \ , \quad (4.7)$$

whose weights $\beta_1, ..., \beta_{10}$ need to be determined. The unknown weights are found by minimizing

$$\begin{aligned} E_g &= \textstyle\sum_{\mathbf{p_i} \in \mathcal{N}(\mathbf{r})} (g(x_i, y_i) - h_i)^2 \phi(\|\mathbf{p_i} - \mathbf{q}\|) \\ &= \textstyle\sum_{\mathbf{p_i}} (g(x_i, y_i) - h_i)^2 w_i \ , \end{aligned} \quad (4.8)$$

where x_i, y_i, h_i are given as in Eq. (4.6). In matrix form, the above equation becomes

$$E_g = (\mathbf{A}\beta - \mathbf{h})^\top \mathbf{W} (\mathbf{A}\beta - \mathbf{h}) \ , \quad (4.9)$$

where

$$\beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_{10} \end{pmatrix}, \ \mathbf{h} = \begin{pmatrix} h_1 \\ \vdots \\ h_k \end{pmatrix}, \ \mathbf{W} = \begin{pmatrix} w_1 & & 0 \\ & \ddots & \\ 0 & & w_k \end{pmatrix}, \quad (4.10)$$

and

$$\mathbf{A} = \begin{pmatrix} x_1^3 & x_1^2 y_1 & x_1 y_1^2 & y_1^3 & x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^3 & x_i^2 y_i & x_i y_i^2 & y_i^3 & x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_k^3 & x_k^2 y_k & x_k y_k^2 & y_k^3 & x_k^2 & x_k y_k & y_k^2 & x_k & y_k & 1 \end{pmatrix} . \tag{4.11}$$

The system of linear equations is generally overdetermined and the unknown coefficients in β are estimated in the least-squares sense. Setting $\frac{\partial E}{\partial \beta}$ to $\mathbf{0}$ results in the following system of normal equations

$$\mathbf{A}^\top \mathbf{W} \mathbf{A} \hat{\beta} = \mathbf{A}^\top \mathbf{W} \mathbf{h} \, . \tag{4.12}$$

The solution vector $\hat{\beta}$ contains the coefficients of the polynomial we seek.

4.1.3 The MLS Projection Operator

Let the (noisy) point-set $\mathbf{P} = \{\mathbf{p_1}, \cdots, \mathbf{p_N}\}$ correspond to observations from an underlying smooth manifold, $\mathcal{M}$. For any point, $\mathbf{r} \in \mathbb{R}^3$ near $\mathcal{M}$, the reconstruction procedure described in the previous sections can be used to locally estimate $\mathcal{M}$ near $\mathbf{r}$. In our formulation the reconstructed surface patch at $\mathbf{r}$ is represented by a reference plane $H_{\mathbf{r}}$, with unit normal $\mathbf{n}$ and center $\mathbf{q} = \mathbf{r} - t\mathbf{n}$, and a bi-variate polynomial $g_{\mathbf{r}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ (t denotes the signed distance of $\mathbf{r}$ from $H_{\mathbf{r}}$; see Fig. 4.2).

Define $\Psi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ as the operator that projects 3D points in the vicinity of $\mathcal{M}$, onto $\mathcal{M}$. In our case, the projection, $\mathbf{r}'$, of a point $\mathbf{r} \in \mathbb{R}^3$ onto $\mathcal{M}$ is given as

$$\mathbf{r}' = \Psi(\mathbf{r}) = \mathbf{q} + \mathbf{n} g_{\mathbf{r}}(0, 0) \, . \tag{4.13}$$

Using the above projection operator, we can define the reconstructed Moving Least Squares (MLS) surface $\mathcal{M}'$, approximating the true underlying surface $\mathcal{M}$, as the set of points in

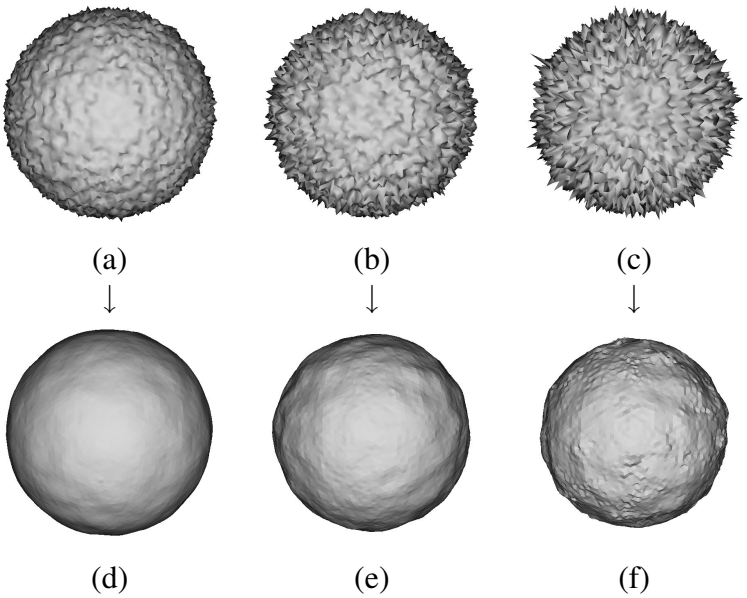
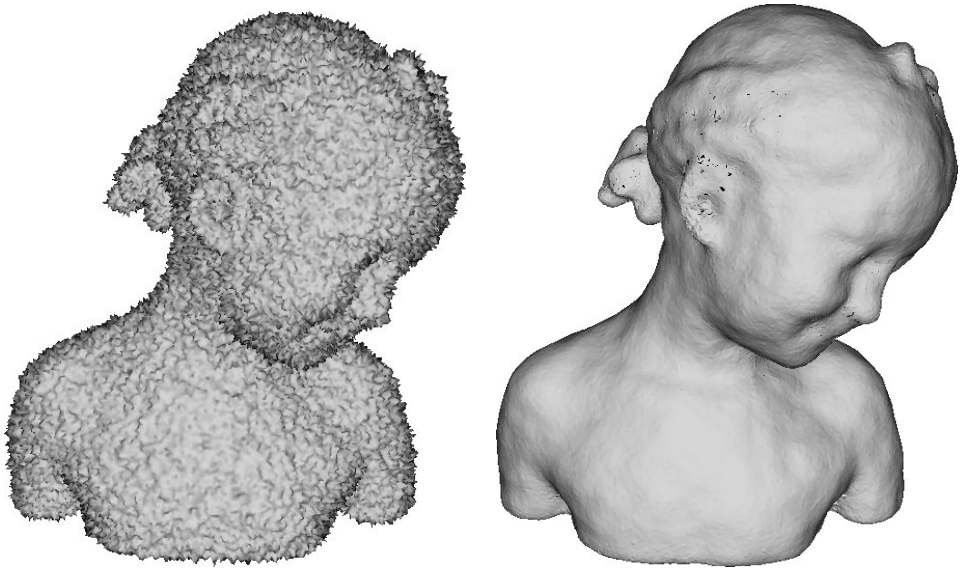


Figure 4.3: Results of MLS smoothing the unit sphere, with ∼40K vertices, perturbed with zero-mean Gaussian noise with varying standard deviations, σ; (a) $\sigma = 0.5\bar{l}$, (b) $\sigma = 1.0\bar{l}$, (c) $\sigma = 1.5\bar{l}$. $\bar{l}$ denotes the average edge length on the original noise-free sphere mesh. Neighborhood size of 100 nearest neighbors was used when computing the MLS projection operator. Flat shading was used when rendering the smoothed surfaces.

$\mathbb{R}^3$ that project onto themselves under Ψ [3]:

$$\mathcal{M}' = \{\mathbf{p} \in \mathbb{R}^3 | \mathbf{p} = \Psi(\mathbf{p})\} \; . \tag{4.14}$$

The MLS projection procedure may be used as a method of denoising (or smoothing) an input point-set. This is done by projecting the original points in $\mathbf{P}$ onto the reconstructed surface. Fig. 4.3 and Fig. 4.4 show the results of MLS smoothing of various surfaces with different levels of noise. In the figures, the original surfaces were given as noisy meshes; $\bar{l}$ denotes the average edge length on the original noise-free mesh surfaces. Note that the connectivity information in the meshes was not used when reconstructing the surfaces.



(a) (b)

Figure 4.4: MLS smoothing of a surface with 100% Gaussian noise ($\sigma = 1.0\bar{l}$); neighborhood of 150 nearest neighbors was used; (a) is the noisy surface, and (b) is the smoothed surface. Flat shading was used in (b) to highlight surface variation on the smoothed surface.

4.1.4 Curvature Estimation on MLS Surfaces

The MLS reconstruction procedure described in the previous section results in a locally parametrized representation of the surface at each interest point near the surface. The representation can be used to compute differential surfaces attributes, such as (improved) normals and curvatures. Let $\mathbf{X} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$,

$$\mathbf{X}(x, y) = \begin{pmatrix} x \\ y \\ g(x, y) \end{pmatrix}, \tag{4.15}$$

denote the parametrized surface at point $\mathbf{r} \in \mathbb{R}^3$; g is given as in Eq. (4.7). The mean curvature, $H_{\mathbf{r}}$, and Gaussian curvature, $K_{\mathbf{r}}$, at $\mathbf{r}$ are given as the trace and determinant of

the shape operator, $\mathcal{S}$, respectively:

$$\begin{aligned} H_{\mathbf{r}} &= \tfrac{1}{2}\text{tr}(\mathcal{S}) \ , \\ K_{\mathbf{r}} &= \det(\mathcal{S}) \ . \end{aligned} \tag{4.16}$$

Let $T_{\mathbf{r}}M$ denote be the tangent plane of surface $\mathcal{M}$ at $\mathbf{r}$. The shape operator at $\mathbf{r}$ is the linear map $\mathcal{S} : T_{\mathbf{r}}M \rightarrow T_{\mathbf{r}}M$, which maps each direction $\mathbf{v} \in T_{\mathbf{r}}M$ to the negative derivative of the surface normal at $\mathbf{r}$: $\mathcal{S}(\mathbf{v}) = -D_{\mathbf{v}}\mathbf{n}(\mathbf{r})$. The shape operator on surface $\mathcal{M}$ parametrized by $\mathbf{X}$ is given as

$$\mathcal{S} = \mathbf{I}^{-1}\mathbf{II} = \begin{pmatrix} \langle \mathbf{X}_x, \mathbf{X}_x \rangle & \langle \mathbf{X}_x, \mathbf{X}_y \rangle \\ \langle \mathbf{X}_x, \mathbf{X}_y \rangle & \langle \mathbf{X}_y, \mathbf{X}_y \rangle \end{pmatrix}^{-1} \begin{pmatrix} \langle \mathbf{n}, \mathbf{X}_{xx} \rangle & \langle \mathbf{n}, \mathbf{X}_{xy} \rangle \\ \langle \mathbf{n}, \mathbf{X}_{xy} \rangle & \langle \mathbf{n}, \mathbf{X}_{yy} \rangle \end{pmatrix} . \tag{4.17}$$

$\mathbf{I}$ and $\mathbf{II}$ are the first and second fundamental forms, respectively. The normal at $\mathbf{r}$ is given as $\mathbf{n} = (\mathbf{X}_x \times \mathbf{X}_y)/\|\mathbf{X}_x \times \mathbf{X}_y\|$.

As a result, to estimate the surface curvatures at a point $\mathbf{r} \in \mathbb{R}^3$ near the surface, first the MLS surface patch at that point is estimated. Then, the determinant and trace of the shape operator are computed as above to estimate the surface curvature at $\mathbf{r}$. The local coordinates of $\mathbf{r}$ in the parametrized surface are given as $(0, 0, g(0, 0))$. The first and second fundamental forms, the shape operator and the mean and Gaussian curvatures at $\mathbf{r}$ are then

variables	x^3	x^2y	xy^2	y^3	x^2	xy	y^2	x	y	1
coefficient	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9	β_{10}
Ideal	.0000	.0000	.0000	.0000	-.5000	.0000	-.5000	.0000	.0000	.0000
$\sigma = 0.5l$	.0204	.0190	.0009	-.0059	-.4814	.0003	-.4750	-.0004	.0000	-.0008
$\sigma = 1.0l$	.0246	.0517	.0961	.0035	-.3938	-.0076	-.3823	-.0011	-.0002	-.0034
$\sigma = 1.5l$	-.0511	.0290	.0658	.0428	-.2674	-.0022	-.2720	.0008	-.0009	-.0071

Table 4.1: Average of the computed polynomial coefficients on the ideal (noise-free) and noisy spheres of Fig. 4.3. Neighborhood size of $N = 100$ was used.

given as

$$
\begin{aligned}
\mathbf{I}_{(0,0)} &= \begin{pmatrix} {\beta_8}^2 + 1 & \beta_8\,\beta_9 \\ \beta_8\,\beta_9 & {\beta_9}^2 + 1 \end{pmatrix}, \\
\mathbf{II}_{(0,0)} &= \frac{1}{({\beta_9}^2+{\beta_8}^2+1)^{\frac{1}{2}}} \begin{pmatrix} 2\,\beta_5 & \beta_6 \\ \beta_6 & 2\,\beta_7 \end{pmatrix}, \\
\mathcal{S}_{(0,0)} &= \frac{1}{\left({\beta_9}^2+{\beta_8}^2+1\right)^{\frac{3}{2}}} \begin{pmatrix} 2\,\beta_5\,{\beta_9}^2 - \beta_6\,\beta_8\,\beta_9 + 2\,\beta_5 & \beta_6\,{\beta_9}^2 - 2\,\beta_7\,\beta_8\,\beta_9 + \beta_6 \\ -(2\,\beta_5\,\beta_8\,\beta_9 - \beta_6\,{\beta_8}^2 - \beta_6) & -(\beta_6\,\beta_8\,\beta_9 - 2\,\beta_7\,{\beta_8}^2 - 2\,\beta_7) \end{pmatrix}, \\
K_{(0,0)} &= \det(\mathcal{S}_{(0,0)}) = \frac{4\,\beta_5\,\beta_7 - {\beta_6}^2}{\left({\beta_9}^2+{\beta_8}^2+1\right)^2}, \\
H_{(0,0)} &= \tfrac{1}{2}\mathrm{tr}(\mathcal{S}_{(0,0)}) = \frac{\beta_5\,{\beta_9}^2 - \beta_6\,\beta_8\,\beta_9 + \beta_7\,{\beta_8}^2 + \beta_7 + \beta_5}{\left({\beta_9}^2+{\beta_8}^2+1\right)^{\frac{3}{2}}}.
\end{aligned}
\tag{4.18}
$$

Table 4.1 shows the average of the estimated polynomial coefficients on the noise-free unit sphere and three noisy spheres with different levels of additive Gaussian noise. In each case, the average was computed over the whole surface. Table 4.2 shows the estimated mean and Gaussian curvature values computed on the noise-free and the noisy spheres. In all cases, the same neighborhood size of 100 nearest neighbors was used to compute the values. As one would expect, the deviation from the ideal (true) values increases as the noise level increases. The main question that arises here is how the noise in the input data affects the estimated shape parameters. We answer this question in Sec. 4.2.

	K	H
Ideal	1.0000	-1.0000
$\sigma = 0.5l$	0.9011	-0.9521
$\sigma = 1.0l$	0.6358	-0.7645
$\sigma = 1.5l$	0.8217	-0.5261

Table 4.2: Average of the computed mean curvature, H, and Gaussian curvature, K, on the ideal and noisy spheres. Neighborhood size of $N = 100$ was used.

4.2 Error Propagation on Reconstructed Surfaces

In the previous section, we saw how the underlying surface corresponding to a (noisy) point-set can be constructed using the Moving Least Squares (MLS) approach. Additionally, it was shown how the shape operator, the mean and Gaussian curvatures at each point on the surface may be estimated using the reconstructed surface. As expected, in the experimental results, we saw that the error in the estimated differential attributes increases as the noise level in the input positions increases.

Assuming the noise is additive and uncorrelated with the data, its effects on the estimated values can be tracked across various stages of the computation. Let point-set $\mathbf{P} = \{\mathbf{p_1}, \cdots, \mathbf{p_N}\}$ contain the 3D positions of the sample points on the ideal (noise-free) surface. The observed point-set, however, is $\mathbf{P}' = \{\mathbf{p'_1}, \cdots, \mathbf{p'_N}\}$, with $\mathbf{p'_i} = \mathbf{p_i} + \epsilon_\mathbf{i}$, and $\epsilon_\mathbf{i} \in \mathbb{R}^3$ denoting the noise added to the data when the observation is made. Each $\epsilon_\mathbf{i}$ is taken to be random variable, and consequently, so is each $\mathbf{p'_i}$. In the following subsections we show how the noise in the positions of the sample points induces error in the estimation of the normal direction, $\mathbf{n}$, of the reference plane, the fitted polynomial coefficients, $(\beta_1, \cdots, \beta_{10})$, and the mean and Gaussian curvatures.

4.2.1 Error Propagation in Local Map Construction

As before, let $\mathbf{r} \in \mathbb{R}^3$ denote the interest point, and $\mathcal{N}(\mathbf{r})$ be the neighborhood point-set associated with $\mathbf{r}$. The local coordinates of each $\mathbf{p_i} \in \mathcal{N}(\mathbf{r})$ in the constructed reference

frame at $\mathbf{r}$ is given by (x_i, y_i, h_i). Throughout, we make the assumption that the noise is only present in the height values; this is a valid assumption [103]. Additionally, to keep the calculations simple, we do not treat the weights used in the weighted least squares fitting as random variables.

Let vector $\mathbf{h}$ contain the ideal (noise-free) height values, and $\hat{\mathbf{h}} = \mathbf{h} + \epsilon$ contain the observed (noisy) values. We seek to determine how the noise is propagated to the estimated polynomial coefficients in the calculations. Let β denote the polynomial coefficients we obtain when the ideal height values are used, and let $\hat{\beta} = \beta + \delta$ be the coefficients obtained when the noisy height values are used; δ is the induced noise arising from the noise in input data, ϵ.

The unknown polynomial coefficients and the height values are related by the following system of linear equations

$$\beta = (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{W}\mathbf{h} \; . \tag{4.19}$$

Therefore, the covariance matrix of the coefficients is related to the covariance matrix of the height values by

$$\begin{aligned}
\Sigma_{\hat{\beta}} &= E[\hat{\beta}\hat{\beta}^\top] - E[\hat{\beta}]E[\hat{\beta}]^\top \\
&= (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{W} E[\hat{\mathbf{h}}\hat{\mathbf{h}}^\top]\mathbf{W}^\top \mathbf{A}(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} \\
&\quad -(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{W} E[\hat{\mathbf{h}}]E[\hat{\mathbf{h}}]^\top \mathbf{W}^\top \mathbf{A}(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} \\
&= (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{W}(E[\hat{\mathbf{h}}\hat{\mathbf{h}}^\top] - E[\hat{\mathbf{h}}]E[\hat{\mathbf{h}}]^\top)\mathbf{W}^\top \mathbf{A}(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} \\
&= (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{W}\Sigma_{\hat{\mathbf{h}}}\mathbf{W}^\top \mathbf{A}(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} \; ,
\end{aligned} \tag{4.20}$$

where $\Sigma_{\hat{\mathbf{h}}}$ denotes the covariance matrix of $\hat{\mathbf{h}}$.

Similarly, the expected covariance matrix of the noise in the computations, Σ_δ, is given by

$$\Sigma_\delta = (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{W}\Sigma_\epsilon \mathbf{W}^\top \mathbf{A}(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} \; , \tag{4.21}$$

with Σ_ϵ denoting the covariance matrix of ϵ. Also, note that since $\hat{\mathbf{h}} = \mathbf{h} + \epsilon$, and $\mathbf{h}$ is not

a random variable, $\Sigma_{\hat{\mathbf{h}}} = \Sigma_\epsilon$. Therefore, $\Sigma_{\hat{\beta}} = \Sigma_\delta$.

Error Propagation using the Abstract Model

When estimating β, since the model relating it to the input, $\mathbf{h}$, is known (and linear) the expected covariance matrices for the parameters and the noise can be obtained easily. However, in most instances, this model is either more complex or simply unknown. In such cases, the expected covariance of the recovered parameters and the noise may still be estimated using the abstract model of [42]. Instead of requiring a model, which establishes a relation between the unknown parameters (output) and the input, the abstract model assumes the unknown parameters are estimated by optimizing an objective function involving the input. Therefore, it requires the objective function $D(\mathbf{h}, \beta)$, which effectively measures the goodness of fit of the estimated parameters β of the fitted model to the input.

As an example, we use the abstract model to estimate the expected covariance matrix of the recovered parameters, $\hat{\beta}$, and the induced noise in the computations, δ. The general approach is as follows: since, β is a minimizer for $D(\mathbf{h}, \beta)$,

$$\mathbf{g}(\mathbf{h}, \beta) = \frac{\partial D(\mathbf{h}, \beta)}{\partial \beta} = \mathbf{0} \, . \tag{4.22}$$

Expanding $\mathbf{g}(\hat{\mathbf{h}}, \hat{\beta})$ about $(\mathbf{h}, \beta)$ up to the first order terms, yields

$$\mathbf{g}(\hat{\mathbf{h}}, \hat{\beta}) = \mathbf{g}(\mathbf{h} + \epsilon, \beta + \delta) \approx \mathbf{g}(\mathbf{h}, \beta) + \epsilon \frac{\partial \mathbf{g}(\mathbf{h}, \beta)}{\partial \mathbf{h}} + \delta \frac{\partial \mathbf{g}(\mathbf{h}, \beta)}{\partial \beta} \, . \tag{4.23}$$

Since $\hat{\beta} = \beta + \delta$ minimizes $D(\hat{\mathbf{h}}, \hat{\beta}) = D(\mathbf{h} + \epsilon, \beta + \delta)$

$$\mathbf{g}(\hat{\mathbf{h}}, \hat{\beta}) = \mathbf{g}(\mathbf{h} + \epsilon, \beta + \delta) = \mathbf{0} \, . \tag{4.24}$$

Additionally, since β minimizes $D(\mathbf{h}, \beta)$, we have $\mathbf{g}(\mathbf{h}, \beta) = \mathbf{0}$, and consequently

$$\epsilon \frac{\partial \mathbf{g}(\mathbf{h}, \beta)}{\partial \mathbf{h}} + \delta \frac{\partial \mathbf{g}(\mathbf{h}, \beta)}{\partial \beta} = 0 \,, \tag{4.25}$$

which yields

$$\delta = -(\frac{\partial \mathbf{g}(\mathbf{h}, \beta)}{\partial \beta})^{-1} (\frac{\partial \mathbf{g}(\mathbf{h}, \beta)}{\partial \mathbf{h}})^{\top} \epsilon \,. \tag{4.26}$$

The above equation estimates the relation between the noise in the input data and the estimated parameters in terms of the partial derivatives of the objective function. The covariance matrix of the induced noise in the computations, Σ_δ, is consequently given by

$$\begin{aligned} \Sigma_\delta &= E[\delta\delta^\top] - E[\delta]E[\delta]^\top \\ &= E[(\tfrac{\partial \mathbf{g}}{\partial \beta})^{-1} (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})^\top \epsilon\epsilon^\top (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}}) (\tfrac{\partial \mathbf{g}}{\partial \beta})^{-\top}] \\ &\quad -E[(\tfrac{\partial \mathbf{g}}{\partial \beta})^{-1} (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})^\top \epsilon] E[(\tfrac{\partial \mathbf{g}}{\partial \beta})^{-1} (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})^\top \epsilon]^\top \\ &= (\tfrac{\partial \mathbf{g}}{\partial \beta})^{-1} (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})^\top (E[\epsilon\epsilon^\top] - E[\epsilon]E[\epsilon]^\top) (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}}) (\tfrac{\partial \mathbf{g}}{\partial \beta})^{-\top} \\ &= (\tfrac{\partial \mathbf{g}}{\partial \beta})^{-1} (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})^\top \Sigma_\epsilon (\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}}) (\tfrac{\partial \mathbf{g}}{\partial \beta})^{-\top} \,. \end{aligned} \tag{4.27}$$

Therefore, to the extent the first order approximation of $\mathbf{g}$ is good,

$$\Sigma_{\hat{\beta}} = E[(\beta + \delta)(\beta + \delta)^\top] - E[\beta + \delta]E[\beta + \delta]^\top = \Sigma_\delta \,. \tag{4.28}$$

In the case of the polynomial least squares fitting problem, D and $\mathbf{g}$ are given by

$$\begin{aligned} D(\mathbf{h}, \beta) &= (\mathbf{A}\beta - \mathbf{h})^\top \mathbf{W} (\mathbf{A}\beta - \mathbf{h}) \,, \\ \mathbf{g}(\mathbf{h}, \beta) &= \tfrac{\partial D(\mathbf{h}, \beta)}{\partial \beta} = 2\mathbf{A}^\top \mathbf{W} (\mathbf{A}\beta - \mathbf{h}) \,. \end{aligned} \tag{4.29}$$

Since the derivative of the objective function is linear in terms of both β and $\mathbf{h}$, its first order Taylor expansion is an exact approximation. Therefore, using the abstract model, we are able to obtain the covariance matrices of the induced noise and the estimated parameters,

exactly:

$$\begin{aligned}\Sigma_{\hat{\beta}} = \Sigma_{\delta} &= (\tfrac{\partial \mathbf{g}}{\partial \beta})^{-1}(\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})^{\top}\Sigma_{\epsilon}(\tfrac{\partial \mathbf{g}}{\partial \mathbf{h}})(\tfrac{\partial \mathbf{g}}{\partial \beta})^{-\top} \\ &= -(2\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}(-2\mathbf{A}^{\top}\mathbf{W})\Sigma_{\epsilon}(-2\mathbf{W}^{\top}\mathbf{A})(-2\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-\top} \\ &= (\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}\Sigma_{\epsilon}\mathbf{W}^{\top}\mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1} \,.\end{aligned} \tag{4.30}$$

Additionally, the variance and covariance matrix of the residual of the fit can be estimated. Let ξ denote the residual. We have

$$\begin{aligned}\xi &= \hat{\mathbf{h}} - \mathbf{A}\hat{\beta} \\ &= (\mathbf{h}+\epsilon) - \mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}(\mathbf{h}+\epsilon) \\ &= [\mathbf{I} - \mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}](\mathbf{h}+\epsilon) \\ &= \underbrace{[\mathbf{I} - \mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}]\mathbf{h}}_{\mathbf{0}} + [\mathbf{I} - \mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}]\epsilon \\ &= \mathbf{Q}\epsilon \,,\end{aligned} \tag{4.31}$$

where $P = \mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}$ and $\mathbf{Q} = \mathbf{I} - \mathbf{P}$.

$\mathbf{Q}\mathbf{h}$ is zero in the above equation since both $\mathbf{Q}$ and $\mathbf{P}$ are orthogonal projection operators, and $\mathbf{P}$ acts as an identity operator on the column space of $\mathbf{A}$ (*i.e.*, $\mathbf{P}\mathbf{A} = \mathbf{A}$). Since $\mathbf{h} \in Col(\mathbf{A})$, $\mathbf{Q}\mathbf{h} = \left[\mathbf{I} - \mathbf{A}(\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}\right]\mathbf{h} = \mathbf{0}$.

With the assumption that $E[\xi] = 0$, the covariance matrix of the residual is then given as

$$\Sigma_{\xi} = E[\xi\xi^{\top}] = E[\mathbf{Q}\epsilon(\mathbf{Q}\epsilon)^{\top}] = E[\mathbf{Q}\epsilon\epsilon^{\top}\mathbf{Q}^{\top}] = \mathbf{Q}\mathbf{E}[\epsilon\epsilon^{\top}]\mathbf{Q}^{\top} \,. \tag{4.32}$$

Assuming the components of the noise vector, ϵ, are uncorrelated and have the same variance, σ^2, we have

$$\Sigma_{\xi} = \mathbf{Q}\mathbf{E}[\epsilon\epsilon^{\top}]\mathbf{Q}^{\top} = \mathbf{Q}\sigma^2\mathbf{I}\mathbf{Q}^{\top} = \sigma^2\mathbf{Q}\mathbf{Q}^{\top} = \sigma^2\mathbf{Q} \,. \tag{4.33}$$

Keeping in mind that $\text{tr}(\mathbf{P}) = K$ and $tr(\mathbf{Q}) = N - K$, where N is the dimension of the

square matrix $\mathbf{P}$, and K is the dimension of the column space of $\mathbf{A}$ (since, rank$(\mathbf{A})$ = rank$(\mathbf{P})$), we can compute the expected variance of the residual as

$$\begin{aligned} E[\xi^\top \xi] &= E[\text{tr}(\xi^\top \xi)] = E[\text{tr}(\xi \xi^\top)] \\ &= \text{tr}(E[\xi \xi^\top]) = \text{tr}(\sigma^2 Q) = \sigma^2 (N - K) \, . \end{aligned} \tag{4.34}$$

Note that tr$(\mathbf{A}) = \sum \lambda_i$, where λ_i are the eigenvalues of $\mathbf{A}$. $\mathbf{P}$ and $\mathbf{Q}$ are projection operators, and therefore, their eigenvalues are only 1 or 0; with the corresponding eigenspaces being the range and kernel of the operators. $\mathbf{P}$ has rank K, therefore, tr$(\mathbf{P}) = K$ and tr$(\mathbf{Q}) =$ tr$(\mathbf{I} - \mathbf{P}) = N - K$.

4.2.2 Error Propagation in Curvature Estimation

In the Sec. 4.1.4, we showed how to compute the mean and Gaussian curvatures at $(0, 0)$, given the polynomial coefficients of the fitted surface at an interest point; $(0, 0)$ denotes the origin of the constructed reference plane at the interest point, and is also the location of the projection of the interest point. Let $\kappa(\beta) = \begin{pmatrix} K_{(0,0)} \\ H_{(0,0)} \end{pmatrix}$, with $\kappa : \mathbb{R}^{10} \rightarrow \mathbb{R}^2$ denoting the non-linear mapping from the polynomial coefficients of the fitted surface to the curvature values at the origin.

Let $\hat{\kappa} = \kappa(\hat{\beta}) = \kappa + \lambda$, with $\hat{\beta} = \beta + \delta$. Again, κ denotes the true curvature values had we used the noise-free input, β, and λ denotes the induced noise in the curvature computations arising from input noise, δ. Expanding $\hat{\kappa}$ up to the first order terms about β yields

$$\hat{\kappa} \approx \kappa + \mathbf{J}\delta \, , \tag{4.35}$$

where $\mathbf{J}$ is the Jacobian of $\hat{\kappa}$ evaluated at β. The covariance matrix of $\hat{\kappa}$ is then estimated as

$$\Sigma_{\hat{\kappa}} \approx E[(\kappa + \mathbf{J}\delta)(\kappa + \mathbf{J}\delta)^\top] - E[(\kappa + \mathbf{J}\delta)]E[(\kappa + \mathbf{J}\delta)^\top] = \mathbf{J}\Sigma_\delta \mathbf{J}^\top \, . \tag{4.36}$$

Consequently, we have

$$\Sigma_{\hat{\kappa}} = \Sigma_{\lambda} \approx \mathbf{J}\Sigma_{\delta}\mathbf{J}^{\top} = \mathbf{J}\Sigma_{\hat{\beta}}\mathbf{J}^{\top} \,. \tag{4.37}$$

For

$$\kappa(\beta) = \begin{pmatrix} \frac{4\,\beta_5\,\beta_7 - {\beta_6}^2}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^2} \\ \frac{\beta_5\,{\beta_9}^2 - \beta_6\,\beta_8\,\beta_9 + \beta_7\,{\beta_8}^2 + \beta_7 + \beta_5}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^{\frac{3}{2}}} \end{pmatrix} \,, \tag{4.38}$$

the Jacobian, $\mathbf{J} = \frac{\partial \kappa}{\partial \beta}$, is given as

$$\mathbf{J} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{4\,\beta_7}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^2} & \frac{{\beta_9}^2 + 1}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^{\frac{3}{2}}} \\ -\frac{2\,\beta_6}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^2} & -\frac{\beta_8\,\beta_9}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^{\frac{3}{2}}} \\ \frac{4\,\beta_5}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^2} & \frac{{\beta_8}^2 + 1}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^{\frac{3}{2}}} \\ -\frac{4\left(4\,\beta_5\,\beta_7 - {\beta_6}^2\right)\beta_8}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^3} & -\frac{\beta_6\,{\beta_9}^3 - 2\,\beta_7\,\beta_8\,{\beta_9}^2 + 3\,\beta_5\,\beta_8\,{\beta_9}^2 - 2\,\beta_6\,{\beta_8}^2\,\beta_9 + \beta_6\,\beta_9 + \beta_7\,{\beta_8}^3 + \beta_7\,\beta_8 + 3\,\beta_5\,\beta_8}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^{\frac{5}{2}}} \\ -\frac{4\left(4\,\beta_5\,\beta_7 - {\beta_6}^2\right)\beta_9}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^3} & -\frac{\beta_5\,{\beta_9}^3 - 2\,\beta_6\,\beta_8\,{\beta_9}^2 + 3\,\beta_7\,{\beta_8}^2\,\beta_9 - 2\,\beta_5\,{\beta_8}^2\,\beta_9 + 3\,\beta_7\,\beta_9 + \beta_5\,\beta_9 + \beta_6\,{\beta_8}^3 + \beta_6\,\beta_8}{\left({\beta_9}^2 + {\beta_8}^2 + 1\right)^{\frac{5}{2}}} \\ 0 & 0 \end{pmatrix}^{\top} . \tag{4.39}$$

4.2.3 Error Propagation in Normal Estimation

The unit normal, $\mathbf{n}$, for the local reference plane at interest point, $\mathbf{r}$, was found as the minimizer for

$$E_H = \mathbf{n}^{\top}\mathbf{Q}\mathbf{W}\mathbf{Q}^{\top}\mathbf{n} \,, \tag{4.40}$$

subject to $\mathbf{n}^\top \mathbf{n} = 1$, where

$$\mathbf{Q}_{3\times k} = \begin{pmatrix} \mathbf{p_1} - \mathbf{r} \\ \vdots \\ \mathbf{p_k} - \mathbf{r} \end{pmatrix}^\top , \tag{4.41}$$

$\mathbf{W} = (w_{i,j})_{k\times k}$ is a diagonal weight matrix, $\mathcal{N}(\mathbf{r})$ denotes the set of neighboring vertices associated with $\mathbf{r}$, and $k = |\mathcal{N}(\mathbf{r})|$. Let $\mathbf{Q}' = \mathbf{Q} + \epsilon$ be the noisy positions vector, with

$$\epsilon = \begin{pmatrix} \epsilon_x^1 & \cdots & \epsilon_x^k \\ \epsilon_y^1 & \cdots & \epsilon_y^k \\ \epsilon_z^1 & \cdots & \epsilon_z^k \end{pmatrix}_{3\times k} , \tag{4.42}$$

and let $\epsilon'_{3k\times 1} = \begin{pmatrix} \epsilon_x^1 & \epsilon_y^1 & \epsilon_z^1 & \cdots & \epsilon_x^k & \epsilon_y^k & \epsilon_z^k \end{pmatrix}^\top$.

Define

$$\begin{aligned} D(\mathbf{Q}, \mathbf{n}) &= \mathbf{n}^\top \mathbf{Q}\mathbf{W}\mathbf{Q}^\top \mathbf{n} , \\ g(\mathbf{Q}, \mathbf{n}) &= \tfrac{\partial D}{\partial \mathbf{n}} = 2\mathbf{Q}\mathbf{W}\mathbf{Q}^\top \mathbf{n} , \\ s(\mathbf{n}) &= \mathbf{n}^\top \mathbf{n} - 1 . \end{aligned} \tag{4.43}$$

In the case of constrained optimization, the expected covariance matrix of the induced noise, η, in the normal computation is given as [42]:

$$\Sigma_\eta = \mathbf{A}^{-1}\mathbf{B}\Sigma_{\epsilon'}\mathbf{B}^\top \mathbf{A} , \tag{4.44}$$

where

$$\mathbf{A} = \begin{pmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{n}} & \frac{\partial s}{\partial \mathbf{n}} \\ \left(\frac{\partial s}{\partial \mathbf{n}}\right)^\top & 0 \end{pmatrix}_{4\times 4} , \mathbf{B} = -\begin{pmatrix} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{Q}}\right)^\top \\ 0 \end{pmatrix}_{4\times 3k} . \tag{4.45}$$

The partial derivatives are given as

$$\begin{aligned} \tfrac{\partial \mathbf{g}}{\partial \mathbf{n}} &= 2\mathbf{Q}\mathbf{W}\mathbf{Q}^\top , \\ \tfrac{\partial s}{\partial \mathbf{n}} &= 2\mathbf{n} . \end{aligned} \tag{4.46}$$

The computation of $\frac{\partial \mathbf{g}}{\partial \mathbf{Q}}$ is a bit more involved. Let

$$\mathbf{C} = \mathbf{Q}\mathbf{W}\mathbf{Q}^\top = \sum_{i=1}^{k} w_{i,i} \begin{pmatrix} (p_x^i - r_x)(p_x^i - r_x) & (p_x^i - r_x)(p_y^i - r_y) & (p_x^i - r_x)(p_z^i - r_z) \\ (p_y^i - r_y)(p_x^i - r_x) & (p_y^i - r_y)(p_y^i - r_y) & (p_y^i - r_y)(p_z^i - r_z) \\ (p_z^i - r_z)(p_x^i - r_x) & (p_z^i - r_z)(p_y^i - r_y) & (p_z^i - r_z)(p_z^i - r_z) \end{pmatrix}. \tag{4.47}$$

Then,

$$\mathbf{g}(\mathbf{Q}, \mathbf{n}) = 2\mathbf{Q}\mathbf{W}\mathbf{Q}^\top \mathbf{n} = 2\mathbf{C}\mathbf{n} = \\ 2\sum_{i=1}^{k} w_{i,i} \begin{pmatrix} n_x(p_x^i - r_x)(p_x^i - r_x) + n_y(p_x^i - r_x)(p_y^i - r_y) + n_z(p_x^i - r_x)(p_z^i - r_z) \\ n_x(p_y^i - r_y)(p_x^i - r_x) + n_y(p_y^i - r_y)(p_y^i - r_y) + n_z(p_y^i - r_y)(p_z^i - r_z) \\ n_x(p_z^i - r_z)(p_x^i - r_x) + n_y(p_z^i - r_z)(p_y^i - r_y) + n_z(p_z^i - r_z)(p_z^i - r_z) \end{pmatrix}. \tag{4.48}$$

Let $\frac{\partial \mathbf{g}}{\partial q_{i,j}}$, denote the partial derivative of $\mathbf{g}$ with respect to j^{th} dimension of the i^{th} component of $\mathbf{Q}$, for $i = 1, \cdots, k$, and $j = 1, 2, 3$. For example,

$$\frac{\partial \mathbf{g}}{\partial q_{i,1}} = 2w_{i,i} \begin{pmatrix} 2n_x(p_x^i - r_x) + n_y(p_y^i - r_y) + n_z(p_z^i - r_z) \\ n_x(p_y^i - r_y) \\ n_x(p_z^i - r_z) \end{pmatrix}^\top. \tag{4.49}$$

Then,

$$\left(\frac{\partial \mathbf{g}}{\partial \mathbf{Q}}\right)_{3k \times 3} = \begin{pmatrix} \frac{\partial \mathbf{g}}{\partial q_{1,1}} & \cdots & \frac{\partial \mathbf{g}}{\partial q_{k,1}} \\ \frac{\partial \mathbf{g}}{\partial q_{1,2}} & \cdots & \frac{\partial \mathbf{g}}{\partial q_{k,2}} \\ \frac{\partial \mathbf{g}}{\partial q_{1,3}} & \cdots & \frac{\partial \mathbf{g}}{\partial q_{k,3}} \end{pmatrix}^\top. \tag{4.50}$$

4.2.4 Estimating Noise Level on General 3D Surfaces

Eqs. (4.31), (4.34), and (4.32) relate the residual of fit, its variance and covariance matrix to the noise in the input surface. These equations can be used to estimate the unknown level of noise on a general input surface. To estimate the noise level at a location on a 3D surface,

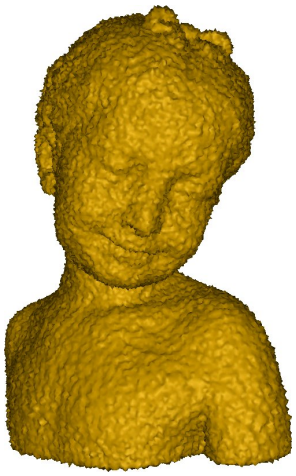


Figure 4.5: Noisy model with 50% additive zero-mean Gaussian noise; the standard deviation of the noise is $\sigma = .5\bar{l} = 2.862 \times 10^{-3}$, where $\bar{l}$ denotes the average edge length on the noise-free model.

a polynomial surface patch is fitted to a small neighborhood around the point of interest as described in previous sections. The residual of the fit can then be used to estimate the noise level on the surface; by using small neighborhoods, we are effectively fitting a model to the surface noise rather than the surface geometry.

As an example, consider the noisy 3D model shown in Fig. 4.5, with approximately $75K$ vertices. Additive zero-mean Gaussian noise was added to the model by moving each vertex on the model along its normal direction by a random amount (value) amount coming from a Gaussian distribution with standard deviation $\sigma = 2.862 \times 10^{-3}$. In order to estimate the standard deviation of the noise on the model, 1000 vertices were randomly picked on the model and a third order bi-variate polynomial was fitted to the neighborhood around each vertex; a neighborhood of 20 nearest neighbors was used for each vertex, when fitting the polynomial patches. Using the *unweighted* residual of fit and Eq. (4.34), an estimate of the standard deviation of the noise at each vertex was obtained. By averaging these values, we were able to obtain an overall estimated standard deviation $\sigma = 2.877 \times 10^{-3}$ for the surface noise, which is very close to the actual noise level.

4.3 Automatic Scale Selection and Software Validation

In the previous sections, we showed how a smooth reconstruction of a point-set surface could be obtained. Additionally, it was shown how the covariance of the noise in the input could be propagated through each stage of the reconstruction. An immediate application of the results from the previous sections are in implementation/software validation. In [42], the authors show how the propagation scheme may be used as a validation procedure to test the implementation of complex vision systems. The general approach consists of computing the expected mean and covariance of the output parameters of the system, analytically, and then comparing the results with the actual mean and covariance of the parameters the system produced, empirically. The comparison, and consequently, acceptance or rejection of the consistency between the empirical and the expected results is done via statistical hypothesis testing.

Another application of the error propagation results is in estimating the minimum neighborhood size required when fitting a model or computing a differential property on the surface. A noisy surface can be thought of as having two underlying structures: the noise and the surface. The structure associated with the noise is considered to have a small scale and high variance. The structure associated with the surface, on the other hand, generally has larger scale and lower variance in a local neighborhood around each point on the surface. Therefore, when the neighborhood used to do the model fitting and consequently compute a differential surface attribute is too small, the recovered parameters tend to correspond to the structure of the noise, rather than the surface. As the neighborhood size increases, the recovered parameters begin to describe the surface rather the noise. We refer to the smallest surface structure on the surface that can faithfully be recovered as the minimum scale of the surface. This scale corresponds to the *inner scale* [127] of the observation device that was used to scan the surface. By automatic scale selection, we refer to the process of determining this inner scale on the noisy surface. The error propagation approach and the statistical hypothesis testing approach, which will be discussed in this section can be used

to obtain an estimate of the inner scale of the surface.

In this section, we first briefly describe the statistical hypothesis testing procedure used in [42] to test our implementation of the MLS surface reconstruction procedure described in the previous sections. We then extend the approach to automatic scale selection on 3D surfaces.

4.3.1 Software Validation

The correct implementation of the system may be validated by comparing the *expected* and *estimated* covariance matrices of the fitted polynomial coefficients and surface curvatures under different levels of noise. Throughout, we use zero-mean additive Gaussian noise with variance σ^2. Let $\bar{l}$ denote the average distance between immediate vertices on the input surface, and let $\mathbf{u}$ denote a unit vector representing the direction along which the noise is added to the surface. Then, we say the surface is perturbed with $x\%$ Gaussian noise, when the position $\mathbf{p}$ of each vertex on the surface is modified according

$$\mathbf{p}' = \mathbf{p} + \delta\mathbf{u}\,, \tag{4.51}$$

where $\delta \sim N(0, \sigma^2)$, and $\sigma = \frac{x}{100}\bar{l}$ denotes the standard deviation of the Gaussian noise. Let $\mathbf{P} = \{\mathbf{p_1}, \cdots, \mathbf{p_N}\}$ be the original, noise-free point-set. A directional Gaussian noise with variance σ^2 and direction $\mathbf{u}$ is added to the point-set, as shown in Fig. 4.6 to obtain a noisy point-set $\mathbf{P'} = \mathbf{P} + \Delta = \{\mathbf{p_1} + \delta_1\mathbf{u} + \cdots, \mathbf{p_n} + \delta_n\mathbf{u}\}$, where $\delta_i \sim N(0, \sigma^2)$.

To test the implementation, we first analytically compute the expected mean and covariance matrix of the polynomial coefficients and curvature values in our surface fitting scheme. We then estimate the polynomial coefficients and surface curvatures of the reconstructed surface under different realizations of the same noise. Let μ_0 and Σ_0 correspond to the expected mean and covariance matrix of the fitted coefficients which are computed analytically. Let μ_β and Σ_β correspond to the estimated mean and covariance of the coeffi-

cients computed empirically. Under the assumption that the implementation is correct, the two distributions corresponding to (μ_0, Σ_0) and $(\mu_\beta, \Sigma_\beta)$ should be the same. Therefore, we hypothesize that the mean and the covariance matrix of the estimated coefficients are the same as the expected mean and covariance matrix; that is,

$$H_0 : \mu_\beta = \mu_0 \text{ and } \Sigma_\beta = \Sigma_0 \, . \tag{4.52}$$

Various test statistics may be used to test the above null hypothesis. We use the following uniformly most powerful test, T:

$$T = -2\log(\lambda) = n(\text{tr}(\Sigma_\beta\Sigma_0^{-1}) + (\mu_\beta - \mu_0)^\top \Sigma_0^{-1}(\mu_\beta - \mu_0) - \log(\det(\Sigma_\beta\Sigma_0^{-1})) - p) \tag{4.53}$$

where

$$\lambda = (e/n)^{pn/2} \det(n\Sigma_\beta\Sigma_0^{-1})^{n/2} \exp(-(\text{tr}(n\Sigma_\beta\Sigma_0^{-1}) + n(\mu_\beta - \mu_0)^\top \Sigma_0^{-1}(\mu_\beta - \mu_0))) \, . \tag{4.54}$$

p is the dimension of β (in the case of the estimated polynomial coefficients $p = 10$), and n is the number of times the polynomial coefficients were estimated under different realizations of the noise to compute the covariance matrix, Σ_β (*i.e.*, the number of estimates). The distribution of T under true null hypothesis is [42]:

$$T \sim \chi^2_{p(p+1)/2+p} \, . \tag{4.55}$$

The distribution of the alternative hypothesis (*i.e.*, $H_A : \mu_\beta \neq \mu_0$ or $\Sigma_\beta \neq \Sigma_0$) is unknown.

In the case of polynomial fitting, $p = 10$, and consequently, the corresponding χ^2 distribution has 65 degrees of freedom. In the case of curvature estimation, $p = 2$ and the corresponding distribution has 5 degrees of freedom.

The *expected* covariance matrices for the polynomial coefficients and curvatures, un-

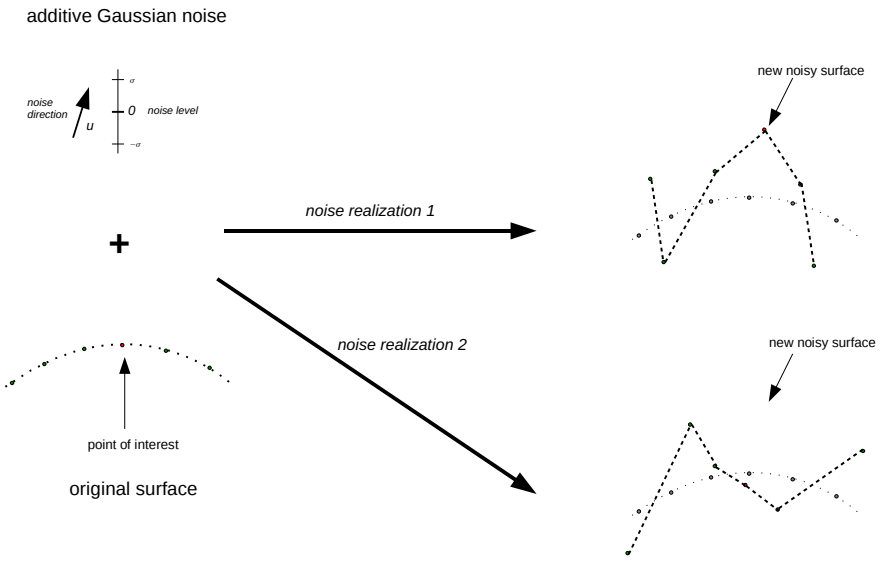


Figure 4.6: Adding Gaussian noise to a local neighborhood of an interest point.

der a known level of noise, are computed according Eqs. (4.30) and (4.37), respectively. Let β be the expected polynomial coefficients, κ the expected curvatures, Σ_β the expected covariance matrix of the coefficients, and Σ_κ the expected covariance matrix of the curvatures (evaluated at point $\mathbf{p}$). M different noisy realizations of the surface are then obtained using the procedure outlined in Fig. 4.6. Let β_i and κ_i denote the estimated polynomial coefficients and mean and Gaussian curvatures at $\mathbf{p}$, under the i^{th} realization of the noisy surface. The *estimated* mean and covariance matrices for the polynomial coefficients and curvatures are then computed as

$$\begin{aligned} \hat{\beta} &= \tfrac{1}{M}\textstyle\sum_{i=1}^{M}\beta_i\ , \quad \hat{\Sigma}_\beta = \tfrac{1}{M}\textstyle\sum_{i=1}^{M}(\beta_i-\hat{\beta})(\beta_i-\hat{\beta})^\top\ , \\ \hat{\kappa} &= \tfrac{1}{M}\textstyle\sum_{i=1}^{M}\kappa_i\ , \quad \hat{\Sigma}_\kappa = \tfrac{1}{M}\textstyle\sum_{i=1}^{M}(\kappa_i-\hat{\kappa})(\kappa_i-\hat{\kappa})^\top\ . \end{aligned} \tag{4.56}$$

The expected and estimated mean and covariances are then compared using the test statistic of Eq. (4.53) to obtain T_j. This process is repeated L times to obtain $\mathbf{T} = \{T_1, \cdots, T_L\}$. Finally, we use the Kolmogorov-Smirnov test to confirm that the distribution of $\mathbf{T}$ is $\chi^2_{p(p+1)/2+p}$.

In the following, we summarize the steps involved in validating the implementation of the reconstruction algorithm by fitting a polynomial patch to the surface at a vertex $\mathbf{p}$, estimating the surface curvatures at $\mathbf{p}$, and running the statistical hypothesis testing procedure, which was described above:

1. Pick an interest point, $\mathbf{p}$, noise direction $\mathbf{u}$, noise level σ^2, and neighborhood size k. Define $N_k(\mathbf{p}) = \{\text{k nearest neighbors of } \mathbf{p}\}$.

2. Repeat the following steps L times to obtain $\mathbf{T}_\beta = \{T_1(\beta), \cdots, T_L(\beta)\}$ and $\mathbf{T}_\kappa = \{T_1(\kappa), \cdots, T_L(\kappa)\}$:

   (a) Repeat the following steps M times to obtain $\mathbf{B} = \{\beta_1, \cdots, \beta_M\}$ and $\mathbf{K} = \{\kappa_1, \cdots, \kappa_M\}$:

      i. Obtain a noisy realization of surface patch defined by the set of points in $N_k(\mathbf{p})$ with directional additive Gaussian noise $(\mathbf{u}, N(0, \sigma^2))$.

      ii. Compute the polynomial coefficients, β_j, on this surface patch.

      iii. Compute the Mean and Gaussian curvatures, κ_j, from β_j at $\mathbf{p}$.

   (b) Compute the mean and covariance matrices of $\mathbf{B}$ and $\mathbf{K}$ to obtain: $\hat{\beta}, \hat{\kappa}, \hat{\Sigma}_\beta, \hat{\Sigma}_\kappa$.

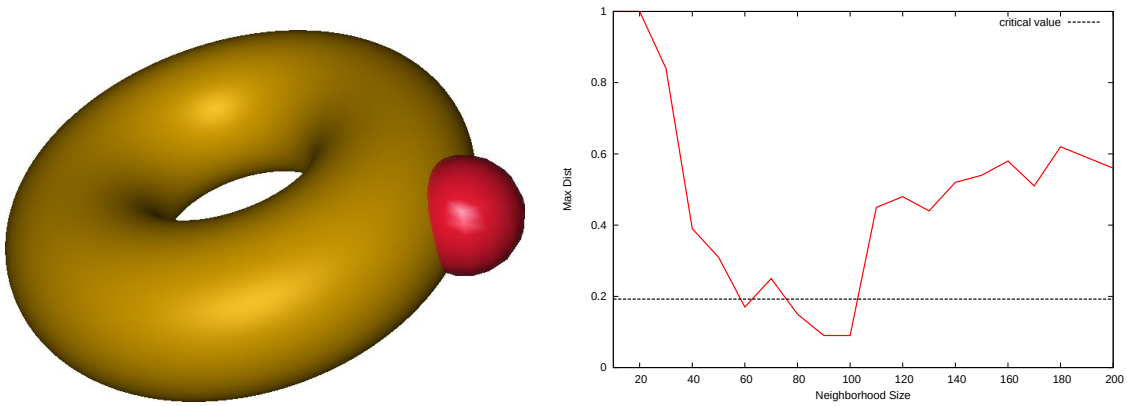
   (c) Compute the expected β, κ, Σ_β, and Σ_κ as follows:

      - Compute β on the original noise-free surface.
      - Obtain κ from the β computed in the previous step.
      - Compute expected Σ_β according Eq. (4.30).
      - Compute expected Σ_κ according Eq. (4.37).

   (d) Using the expected and estimated mean and covariance matrices of β and κ compute the test statistics $T_l(\beta)$ and $T_l(\kappa)$ using Eq. (4.53).

3. The null hypothesis is that the distributions of $\mathbf{T}_\beta$ and $\mathbf{T}_\kappa$ are χ^2_{65} and χ^2_5, respectively; use the Kolmogorov-Smirnov (KS) test to reject or tentatively accept the null hypothesis.

In the above procedure, tentative acceptance of the null hypothesis validates the implementation of the reconstruction algorithm. The Kolmogorov-Smirnov test statistic is a



(a) (b)

Figure 4.7: Validating implementation using statistical hypothesis testing; the red sphere in (a) indicates the location on the surface where the surface was reconstructed and the curvatures estimated. The graph in (b) plots the Kolmogorov-Smirnov test statistic as a function of the neighborhood size used to reconstruct the surface. The horizontal line shows the critical value for significance level $\alpha = 0.05$; in the experiments, 20% Gaussian noise was used.

distance measure between the cumulative distribution functions of two sample sets, which are being compared. In the case where the correctness of the estimated surface curvatures is being tested, the two distributions are χ^2_5 and the distribution of the samples in set $\mathbf{T}_\kappa$. In our experiments, we repeated step 2 in the above procedure 100 times ($L = 100$). For a significance level of $\alpha = 0.05$ and $L = 100$ experiments in the KS test, the critical (cut-off) value is $D = 0.19$. In Fig. 4.7, we show how the KS test statistic for the estimated curvature values, at a point on a torus, with 20% Gaussian noise (not shown) varies as the neighborhood size used to reconstruct the surface is increased. As can be seen, for small neighborhood sizes the KS test fails. However, as the neighborhood size is increased, the value of the KS test statistic decreases and eventually falls below the critical value and the null hypothesis is accepted. For large neighborhood sizes the value of the test statistic increases again until the null hypothesis is continually rejected.

In our experiments, we noticed that the above procedure does not work on all surface types or surfaces with high levels of noise. Therefore, in the experiments presented in the remainder of this chapter, we use the following modified approach: we assume the

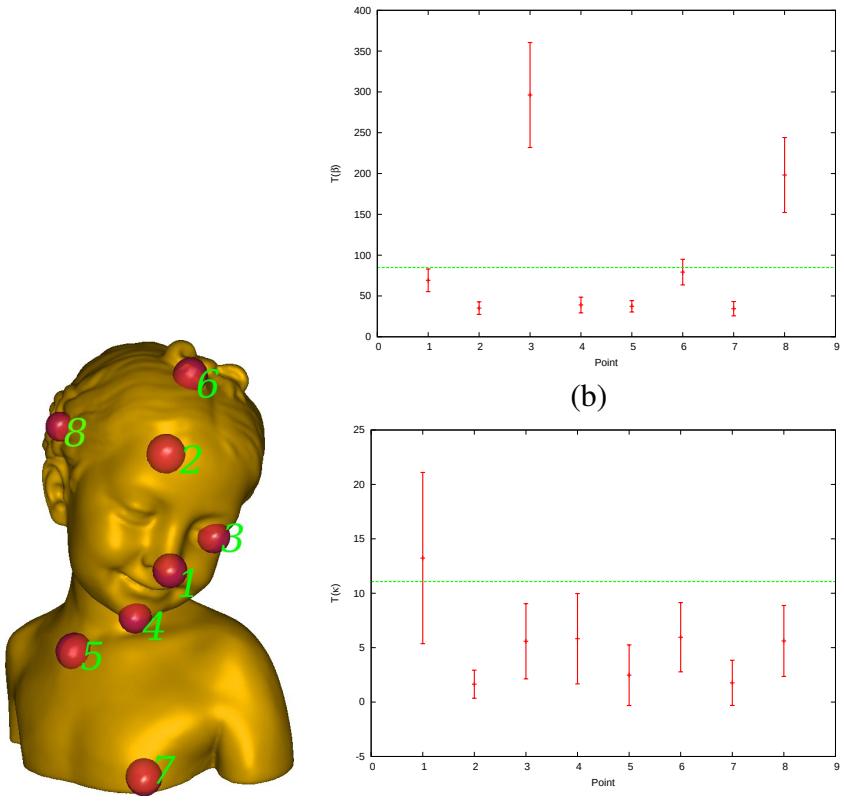
	probability of exceeding the critical value				
	.10	.05	.025	.01	.001
χ^2_5	9.236	11.070	12.833	15.086	20.515
χ^2_{65}	79.973	84.821	89.177	94.422	105.988

Table 4.3: Upper critical values of χ^2_5 and χ^2_{65} distributions.

samples in $\mathbf{T}_\beta$ and $\mathbf{T}_\kappa$ come from *noncentral* χ^2 *distributions* with 65 and 5 degrees of freedom (and unknown noncentrality parameters), respectively. As the neighborhood size used to reconstruct the surface is changed, the corresponding distributions of $\mathbf{T}_\beta$ and $\mathbf{T}_\kappa$ are shifted and scaled. The tests in experiments where the means of sample sets in $\mathbf{T}_\beta$ and $\mathbf{T}_\kappa$ are smaller than a preselected critical value are accepted. In Table 4.3, we show the critical values for χ^2_{65} and χ^2_5 distributions for a few significant levels. In Fig. 4.8, we show the results of running the described test procedure at a few vertices on the model for a fixed neighborhood size of 100; the vertical lines indicate the standard deviations of the samples in $\mathbf{T}_\beta$ and $\mathbf{T}_\kappa$. As can be seen, for the given neighborhood size, the tests fail only in regions with somewhat complex structures. However, as the neighborhood size is increased, the tests for the more complicated structures also pass. In Sec. 4.3.2, we use these results to decide on the minimum neighborhood size needed when reconstructing noisy surfaces.

Implementation Issues

Computing the test statistic involves computing the inverse of the expected covariance matrices of the polynomial coefficients and curvatures. In many cases, however, when the shape of the surface is simple, some of its corresponding polynomial coefficients (especially, the higher order terms) vanish. This causes the matrix inverse computations to become unstable. For example, false coloring is used in Fig. 4.9 to show the distribution of the computed test statistic over the surface of the Bimba model; red (hot) colors correspond to large T's and blue (cold) colors correspond to small values of T. As can be seen, in the

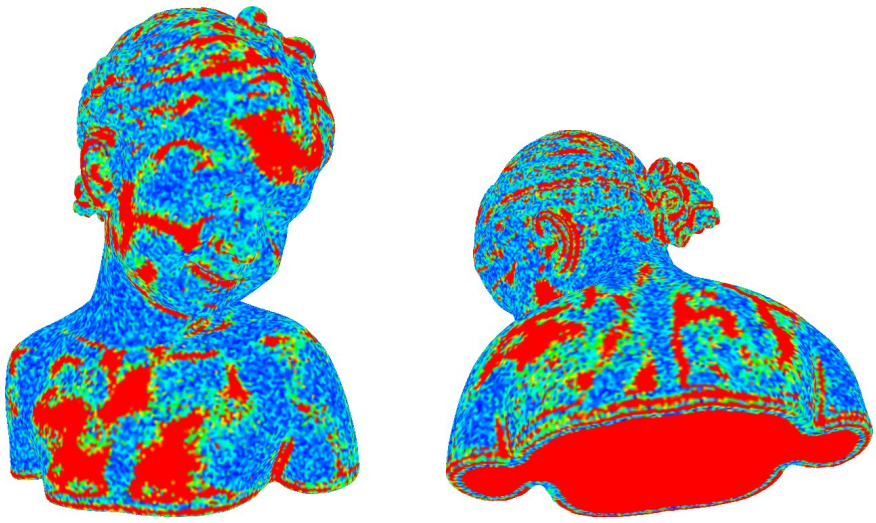


(a) (b) (c)

Figure 4.8: Computed test statistics for (b) the estimated polynomial coefficients and (c) curvature values at various points on the Bimba model with 10% Gaussian noise (not shown). The dashed green line shows the significance level $\alpha = 0.05$ for (b) χ^2_{65} and (c) χ^2_5 distributions; a neighborhood of 100 nearest neighbors was used at each point.

areas where at least one of the principal curvatures vanishes, the computed T's are very large.

Let $\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$ denote the eigendecomposition of the symmetric matrix we are trying to invert. The diagonal matrix $\mathbf{\Lambda}$, with diagonal entries λ_i, contains the eigenvalues of $\mathbf{M}$ is descending order. $\mathbf{\Lambda}'$ is obtained from $\mathbf{\Lambda}$ by setting the eigenvalues smaller than some ϵ to zero. Let $\mathbf{M}^+ = \mathbf{Q}\mathbf{\Lambda'}^{-1}\mathbf{Q}^\top$ denote the pseudo-inverse of $\mathbf{M}$ obtained in this manner. $\mathbf{M}^+$ can be used in place of $\mathbf{M}^{-1}$ in Eq. (4.53) to compute T. However, the determinant of the product of a matrix $\mathbf{A}$ with the pseudo-inverse is computed differently. Let K denote the number of non-zero eigenvalues in $\mathbf{\Lambda}'$, and $\mathbf{J}$ be a $p \times p$ diagonal matrix with diagonal entries $j_{ii} = 1$ for $i \leq K$ and $j_{ii} = 0$ for $i > K$, $\mathbf{P} = \mathbf{QJ}$, and $\mathbf{A}' = \mathbf{P}\mathbf{P}^\top\mathbf{A}\mathbf{P}\mathbf{P}^\top$.



(a) (b)

Figure 4.9: Distribution of $T(\kappa)$ over the surface of the Bimba model when the test statistic is naively computed; red (hot) colors correspond to large T's and blue (cold) colors correspond to small T's.

Let Δ be the diagonal matrix, with diagonal entries δ_i, containing the eigenvalues of $\mathbf{A}'$ in descending order. Then, $|\mathbf{A}'| \sim \Pi_{i=1}^{K} \delta_i$, $|\mathbf{M}^+| \sim \Pi_{i=1}^{K} \frac{1}{\lambda_i}$, and

$$|\mathbf{A}'\mathbf{M}^+| \sim \Pi_{i=1}^{K} \frac{\delta_i}{\lambda_i} \,. \tag{4.57}$$

$|\mathbf{A}'\mathbf{M}^+|$ is used in place of $|\mathbf{A}\mathbf{M}^+|$ when computing T. Additionally, since the degrees of freedom have been reduced to K in the above computations, $p = K$ should be used both when computing T and testing the null hypothesis.

4.3.2 Automatic Scale Selection

In this section, we address the problem of choosing the right neighborhood size (scale) when reconstructing the surface or estimating a differential property on a possibly noisy surface. As one would expect, the smallest neighborhood size needed to correctly estimate the local geometry of the surface is directly related to the signal-to-noise ratio (SNR) of the surface, which is being reconstructed. Additionally, as will be shown later in this section,

k	$\hat{T}(\beta)$	std$(T(\beta))$	$\hat{T}(\kappa)$	std$(T(\kappa))$
30	709.818216	44.851792	53.897816	35.286950
40	864.035673	53.119690	13.236161	10.596596
50	950.213714	50.130063	9.141995	11.161460
60	517.719319	44.482952	6.596001	9.981383
70	37.622512	8.247707	4.681548	5.842763
80	38.541797	9.996436	2.848392	2.451173
90	38.072085	10.612102	2.565717	2.906478
100	35.778368	8.410462	2.562647	2.219588
110	36.784341	7.723637	2.471866	2.019013
120	36.351695	7.790615	2.365497	2.757388
130	37.348674	7.727040	2.506941	2.545995
150	36.934627	8.133453	2.948145	2.518951
200	37.994565	9.159700	2.865477	3.439013
250	36.066682	7.995311	3.919198	3.244758

Table 4.4: The mean and standard deviation of the test statistics computed at a vertex on a noisy sphere with 50% Gaussian noise; k is the neighborhood size used at the vertex to compute T. The highlighted row shows the first neighborhood size which passed both tests for the polynomial coefficients and the curvatures at the 5% significance level.

the neighborhood size is also dependent on the local geometry of the surface. Therefore, each point on a general surface could have a different scale. We use the statistical hypothesis testing approach discussed in the previous section to obtain an empirical link between the SNR and the inner scale of the surface.

We start with the simple cases where the input surface is locally the same everywhere (*e.g.*, planar, cylindrical, and spherical surfaces) and the noise level is known. Table 4.4 shows the values of $T(\beta)$ and $T(\kappa)$ computed at a vertex on the unit sphere with 50% additive Gaussian noise; the original, noise-free sphere mesh had $\sim 40K$ vertices with average edge length of ~ 0.02. The highlighted row shows the first neighborhood size which passed both statistical tests for the polynomial coefficients and the curvatures. By applying the same procedure to the same surface with different levels of noise we are able to get an estimate of the minimum required neighborhood size for each surface. This is demonstrated in the results shown in Table 4.5. As can be seen, as the noise level increases, the required minimum neighborhood size increases.

Noise level	minimum k	$\hat{T}(\beta)$	std$(T(\beta))$	$\hat{T}(\kappa)$	std$(T(\kappa))$
20%	30	35.433801	8.892575	8.288684	9.987461
50%	70	37.622512	8.247707	4.681548	5.842763
80%	100	39.023673	8.889890	4.084890	5.376168
100%	130	42.181347	9.881063	4.034220	4.433164
150%	210	44.718942	10.052251	8.288253	5.930069

Table 4.5: The estimated minimum neighborhood sizes required to compute the MLS patch at a vertex; k denotes the required number of nearest neighbors.

The next question that arises is the following: assuming we are dealing with surfaces with the same sampling rate and noise level but with different geometry, how does the recovered inner scale vary with different surface geometry? In other words, does the local geometry bear any weight on the associated scale at a point? Table 4.6 shows the results for the recovered inner scales on a plane, cylinder, sphere, and multiple points on a torus under different levels of noise. The surfaces have approximately the same sampling rate of ~ 3200 points per unit area and average edge length of ~ 0.02.

In the case of the torus, the selected locations correspond to three different classes of points (see Fig. 4.10): Hyperbolic, Parabolic, and Elliptic. At a *hyperbolic point*, the principal curvatures have opposite signs, or equivalently, the Gaussian curvature is negative. At a *parabolic point*, exactly one of the principal curvatures is zero, or equivalently, the Gaussian curvature is zero but the shape operator is non-zero. At an *elliptic point*, both principal curvatures have the same sign, or equivalently, the Gaussian curvature is positive. Additionally, a point is said to be *planar* if both principal curvatures at the point are zero, or equivalently, both the Gaussian curvature and the shape operator are zero.

In Fig. 4.10, the bounding spheres enclosing 400 nearest neighbors to each of the points are shown. As can be seen, the neighborhood sizes in terms of the number of nearest neighbors have different meanings in the spatial domain. Table 4.7 shows the results of the same set of experiments as in Table 4.6, except in terms of the radii of the bounding spheres enclosing the minimum required number of neighbors when reconstructing the local surface

patches at the interest points. As can be seen, the required minimum neighborhood sizes (scales) differ for different surface types.

Additionally, we run the same tests on the torus model with quarter of the sampling rate of the original torus; the average edge length on the new, low-resolution torus, is $\bar{l} =\sim 0.04$, with ~ 820 vertices per unit area. In this experiment, however, the average edge lengths used to define and add the directional noise are taken to be from the higher resolution torus used in the previous experiments (with $\bar{l} = 0.02$). Tables 4.8 and 4.9 show the results. As can be seen, compared to the higher resolution surface, the recovered scales have almost halved in terms of the number of nearest neighbors. On the other hand, the radii of the bounding spheres have increased on the low resolution noisy torus; *i.e.*, larger neighborhood sizes are needed when the SNR is higher.

In summary, based on the results shown in Tables 4.6, 4.7, 4.8, and 4.9 the recovered scale at point on a noisy surface is dependent on the SNR and surface type.

It is noteworthy to mention that using the above-mentioned automatic scale selection process, we can also determine the maximum amount of noise that may be added to a surface until its curvatures cannot be reliably estimated—no matter how large the size of the used neighborhood is grown. As an example, Table 4.10 demonstrates what happens when the noise levels become too high, when attempting to recover the scale at different points on the low-resolution torus model. As can be seen, once the noise level exceeds a certain threshold, larger neighborhood sizes do not pass both statistical hypothesis tests for the polynomial coefficients and the curvature values. Fig. 4.11 shows a visualization of the noisy torus where all the tests in Table 4.10 fail.

4.4 Summary

In this chapter, we reviewed the Moving Least Squares (MLS) surface reconstruction method. In addition, we showed how the noise in the input surface is propagated to the estimated

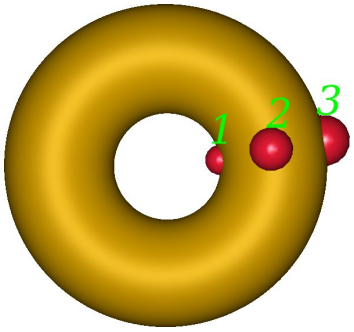


Figure 4.10: Three different types of points picked on the torus: (1) Hyperbolic point ($K(p) < 0$), (2) Parabolic point ($K(p) = 0$), and (3) Elliptic point ($K(p) > 0$); $K(p)$ denotes the Gaussian curvature at interest point, p. In all three cases, the spheres are the bounding spheres enclosing 400 nearest neighbors for each interest point.

		Noise Level						
Surface	Point Type	20%	40%	60%	80%	100%	120%	140%
Plane	Planar	50	60	80	110	130	160	180
Cylinder	Parabolic	110	170	200	240	280	310	350
Sphere	Elliptic	40	70	80	100	130	150	180
Torus	Hyperbolic	70	100	130	160	200	250	260
Torus	Parabolic	70	100	140	170	180	200	250
Torus	Elliptic	70	100	130	160	190	190	250

Table 4.6: Automatic scale selection on simple surfaces: the table shows the estimated minimum neighborhood sizes needed when computing the polynomial fit and estimating curvatures on different surfaces with varying levels of noise. The average edge length on the original noise-free surfaces were $\bar{l} =\sim 0.02$. The sampling rate on all the surfaces was approximately the same, with ~ 3200 points per unit area uniformly sampled on each surface. A noise level of $n\%$ indicates a zero-mean Gaussian with standard deviation $\sigma = \frac{n}{100}\bar{l}$. The direction of the noise was picked as the normal direction at the interest point were the test was run.

		Noise Level						
Surface	Point Type	20%	40%	60%	80%	100%	120%	140%
Plane	Planar	.0736	.0757	.0892	.1071	.1143	.1262	.1359
Cylinder	Parabolic	.1070	.1298	.1437	.1535	.1683	.1784	.1896
Sphere	Elliptic	.0477	.0634	.0689	.0783	.0921	.1015	.1096
Torus	Hyperbolic	.0566	.0713	.0793	.0890	.0991	.1115	.1135
Torus	Parabolic	.0808	.0948	.1130	.1248	.1279	.1347	.1508
Torus	Elliptic	.1058	.1188	.1378	.1587	.1673	.1673	.1925

Table 4.7: The above table shows the same results as in Table 4.6, except that the neighborhood sizes are given in terms of the radii of the bounding spheres of the neighbors rather than the number of nearest neighbors.

	Noise Level						
Point Type	20%	40%	60%	80%	100%	120%	140%
Hyperbolic	30	40	60	80	100	120	140
Parabolic	30	50	50	70	80	90	100
Elliptic	30	50	60	70	70	90	90

Table 4.8: Automatic scale selection results on a torus with quarter of the sampling rate of the torus used in Tables 4.6 and 4.7; the average edge length on the new, low-resolution torus, is $\bar{l} = 0.04$, with ~ 820 vertices per unit area. In the experiments run on this torus, the average edge lengths used to define and add the directional noise are taken to be from the higher resolution torus used in the previous experiments (with $\bar{l} = 0.02$).

	Noise Level						
Point Type	20%	40%	60%	80%	100%	120%	140%
Hyperbolic	.0780	.0873	.1103	.1260	.1403	.1490	.1651
Parabolic	.1034	.1343	.1343	.1591	.1683	.1770	.1870
Elliptic	.1256	.1715	.1812	.2079	.2079	.2321	.2321

Table 4.9: The above table shows the same results as in Table 4.8, except that the neighborhood sizes are given in terms of the radius of the bounding spheres of the neighbors rather than the number of nearest neighbors.

	Noise Level						
Point Type	40%	80%	120%	160%	200%	240%	280%
Hyperbolic	50	80	120	180	-	-	-
Parabolic	50	70	90	110	140	170	-
Elliptic	40	70	90	100	150	-	-

Table 4.10: Automatic scale selection results on the low-resolution torus of Table 4.8. As the noise level becomes too high, the statistical hypothesis tests for the computed polynomial coefficients and the curvatures fail to pass—no matter how large the neighborhood size is grown. As in the previous case, $\bar{l} = 0.02$ was used when adding the noise.

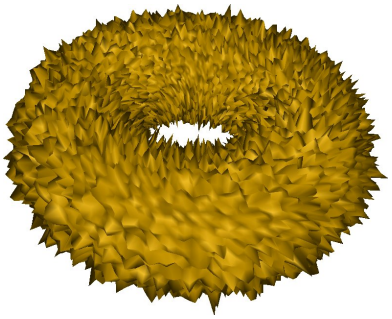


Figure 4.11: Mean and Gaussian curvatures on the above noisy torus can not reliably be estimated using any neighborhood size. By "reliably" we mean: in a manner in which the amount of uncertainty in the estimated values can be known.

surface normals and curvatures. We showed how the results can be used in a statistical hypothesis testing procedure to validate the implementation of the reconstruction algorithm. More importantly, we used the results to decide on the minimum neighborhood size needed when reconstructing the surface. We referred to this minimum size at a point $\mathbf{p}$, as the *scale* of the surface structure at $\mathbf{p}$; we referred to the process of finding this scale as *automatic scale selection*.

Chapter 5

Curvature Scale Space 3D

"How did Biot arrive at the partial differential equation [the heat equation]? ...
Perhaps Laplace gave Biot the equation and left him to sink or swim for a few years in trying to derive it."

–Clifford Truesdell

Scale-space techniques are now widely used for signals in $\mathbb{R}^n$ [72], with the theory having become quite mature over the past few decades, especially for 2D images. Beyond having nice theoretical properties, the scale-space representation of images has been shown to be realized efficiently with impressive practical results [77, 21]. Currently, scale-space matching techniques, such as SIFT [77] and SURF [8], are the *de facto* standards in many 2D matching applications. Despite finding widespread use in 2D signal processing, scale-space techniques have not been widely applied to 3D surfaces.

There are two major difficulties with extending scale-space representations to 3D surfaces. These difficulties are due to representation issues and the estimation of the scale parameter necessary for automatic scale selection. The lack of grid-like structures that are present in 2D images and the non-Euclidean geometry of surfaces make development of precise and efficient representations difficult. The scale parameter, which in the case of signals in $\mathbb{R}^n$ is defined in terms of the variance of the smoothing kernel, may not be readily

available for 3D surfaces.

In this chapter, we extend the use of scale-space theory to 3D surfaces for the purpose of shape matching. The main contribution is a new scale-space representation for 3D surfaces that addresses the two major difficulties outlined above. The new representation is shown to be insensitive to noise, computationally efficient, and capable of automatic scale selection.

The few current scale-space based surface representations (see Chapter 3) can be categorized into two classes based on how a signal is derived from the surface and consequently evolved. First, surface positions may be treated as the signal and therefore the surface geometry is modified during the evolution process. Second, a signal may be defined and evolved over the surface while the geometry of the surface remains unchanged. It is well-known that the evolution of surface positions generally leads to geometric problems such as shrinkage and foldovers [126, 28]. Therefore, we opt for the second approach whereby we define our scale-space representation in terms of the evolution of the surface curvatures.

This chapter in organized as follows: in Sec. 5.1, we present our proposed scale-space representation for 3D surfaces, which we refer to as Curvature Scale Space 3D (CS3). In Sec. 5.2, we show how CS3 can be used for feature extraction with automatic scale selection. In Sec. 5.3, we show how the noise in the input surface affects the estimated surface curvatures at each level in our representation. In Sec. 5.4, we compare the properties of our proposed CS3 representation with other scale-space based surface representations. In Sec. 5.5 and Sec. 5.6, we present applications of CS3 in automatic surface registration and crest line extraction on noisy 3D surfaces. Finally, in Sec. 5.7, we provide a summary of this chapter.

5.1 Scale-Space Representation for 3D Surfaces

As mentioned in Chapter 3, the scale-space representation of a signal in $\mathbb{R}^n$ adds a new *scale* dimension to the original representation of the signal. The finest scale corresponds to

the original signal, while the coarser (higher) scales correspond to the Gaussian smoothed versions of the signal. As the scale is increased, high frequency contents of the signal are removed, and the signal becomes smoother until it converges to a constant value.

The majority of the extensions of the scale-space representation to 3D surfaces, that were reviewed in Chapter 3, treat surface positions as the signal, and consequently build the scale-space stack by smoothing the positions. The two main differentiating factors between the approaches are: 1) the surface representation used, and 2) the method used to define/apply the smoothing operator on the surface.

On the other hand, approaches such as the Heat Kernel Signature (HKS) [120, 20] obtain a multiscale representation of the surface by encoding information about the heat kernel at various scales (times). The main factor, which separates this approach from others is the *indirect* inference of information about surface structures through the study of the heat kernel. However, both scale-space and HKS based approaches are related, as the heat equation and its fundamental solution are central in both types of representations.

We define our scale-space formulation of the surface in terms of the evolution (diffusion) of signals on the surface with the help of the Laplace-Beltrami operator. We analyze the surface structure by directly studying the *signal* as it evolves on the surface. More specifically, we take the signal to be the surface curvatures, which are derived from the surface geometry. The main advantages of this approach over HKS are gains in computation efficiency and the ability to estimate the size of the surface structures. Additionally, our representation enables us to robustly and efficiently estimate the Laplacians of surface curvatures, which result in a rich set of features useful in the subsequent matching tasks.

The advantage of smoothing surface curvatures, instead of positions, is that (at least for genus 0 surfaces) the possibility of developing undesirable geometric artifacts during the smoothing process is reduced. Additionally, all the relevant local geometric information about the surface is encoded by the principal directions and curvatures. The surface corresponding to the smoothed curvatures may also be reconstructed using techniques such as

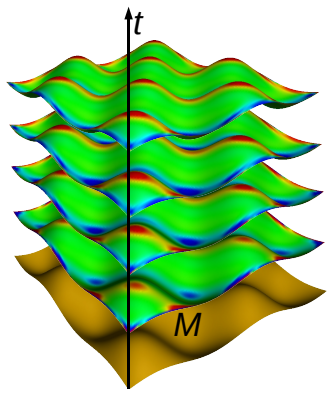


Figure 5.1: Scale-space representation of a surface signal

[32]. Moreover, geometric feature extraction may be performed directly in this curvature-domain. On the other hand, the surface noise is amplified more when the curvature is used instead of the positions. Nevertheless, assuming a zero-mean noise (with any distribution), the effects of surface noise is expected be dramatically reduced at the higher scales in this representation (see Sec. 5.3).

We define the scale-space representation, $F : \mathcal{M} \times \mathbb{R} \rightarrow \mathbb{R}^n$, of a continuous signal $f : \mathcal{M} \rightarrow \mathbb{R}^n$ on surface $\mathcal{M}$ as the solution to the heat (diffusion) equation

$$\frac{\partial}{\partial t} F(\mathbf{x}; t) = \Delta_{\mathcal{M}} F(\mathbf{x}; t) \ , \tag{5.1}$$

with the initial condition $F(\mathbf{x}; 0) = f(\mathbf{x})$; $\Delta_{\mathcal{M}}$ denotes the Laplace-Beltrami operator. Fig. 5.1 shows an example of the scale-space representation of a signal. Similar to the case of signals in $\mathbb{R}^n$, the representation may equivalently be defined using convolution with Gaussians,

$$F(\mathbf{x}; t) = f(\mathbf{x}) *_{\mathcal{M}} G(\mathbf{x}; t) \ , \tag{5.2}$$

where $\mathbf{x}$ is a point on $\mathcal{M}$, $G(\mathbf{x}; t)$ is a Gaussian with scale t. $*_{\mathcal{M}}$ denotes convolution on

$\mathcal{M}$, and is defined as

$$f(\mathbf{x}) *_{\mathcal{M}} G(\mathbf{x}; t) = \int_{-\infty}^{\infty} f(\tau) g(D_{\mathcal{M}}(\mathbf{x}, \tau); t) d\tau \ . \tag{5.3}$$

$D_{\mathcal{M}}(\mathbf{a}, \mathbf{b})$ measures the geodesic distance between points $\mathbf{a}$ and $\mathbf{b}$ on $\mathcal{M}$, and $g(r; t)$ corresponds to the 1-dimensional Gaussian with scale t.

We define the *Curvature Scale Space 3D (CS3)* representation of a surface as the scale-space representation of its curvatures. The curvature function may be the mean, Gaussian, or the curvature tensor. In majority of this work (except in Sec. 5.6), we use mean surface curvatures to define the CS3 representation. In this formulation, a stack of Gaussian smoothed surface curvatures is obtained, which can be used directly in multiscale feature extraction and descriptor computations. CS3 is not to be confused with Curvature Scale-Space (CSS) of Mokhtarian [88, 87], which is a multiscale representation for closed plane curves and is used for 2D shape matching.

The CS3 representation of a surface is constructed by first estimating the curvature tensor and computing the principal curvatures at each point on the surface. The surface curvatures are subsequently evolved according to Eq. (5.1) to obtain a stack of increasingly smoothed curvatures. An alternative, and more accurate, approach is to estimate the curvature tensor at the finest level and then smooth the *curvature tensor* to build the stack. The principal curvatures (and directions) are then obtained from the smoothed curvature tensors at each level. Besides improving the accuracy, this approach results in principal directions, whose orientations are more consistent between neighboring vertices. This approach, however, is slower than the original formulation and may not be necessary in matching applications. We use this approach in later sections, when extracting crest lines on noisy surfaces; everywhere else in this work, the former approach is used to construct the CS3 representation.

The curvature tensor may be estimated using a variety of approaches. The interested

reader is referred to [102] for a survey and comparison of different methods in the literature. In our implementation, we use Taubin's method [125] to estimate the curvature tensors at the finest level in CS3. This approach was selected since it provides a good balance between accuracy and speed.

In the next subsections we describe how the CS3 representation of a *discrete* surface represented by a polygonal mesh may be constructed in more detail. In Sec. 5.1.1, we first state the relation between the scale-space representation of a surface signal and the heat kernel and heat operator. In Sec. 5.1.2, we describe how a discrete surface signal may be efficiently smoothed in a manner consistent with the scale-space representation of signals. In Sec. 5.2, we show how the representation may be used for feature point extraction with an automatic scale selection mechanism.

5.1.1 Relation of Scale-Space Representation of Surface Signals to the Heat Kernel

The scale-space representation of a surface signal, $f : \mathcal{M} \rightarrow \mathbb{R}^n$, given by Eq. (5.1) can equivalently be defined with the help of the *heat operator*, H_t, which evolves any signal on $\mathcal{M}$ according to the heat equation. That is,

$$H_t f(x) = F(x,t) \ , \tag{5.4}$$

where F denotes the scale-space representation of f.

H_t and the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ are related by the exponential [120]:

$$H_t = e^{t\Delta_{\mathcal{M}}} = \sum_{i=0}^{\infty} \frac{(t\Delta_{\mathcal{M}})^i}{i!} = I + t\Delta_{\mathcal{M}} + \frac{t^2}{2}\Delta_{\mathcal{M}}^2 + \dots \ , \tag{5.5}$$

where I denotes the identity operator. Throughout this work, $-\Delta_{\mathcal{M}}$ is assumed to be constructed such that it is positive semi-definite. Let $\lambda_i \leq 0$ and ϕ_i denote the i^{th} eigenvalue

and eigenfunction of $\Delta_{\mathcal{M}}$, respectively. Then, Eq. (5.5) implies that the i^{th} eigenfunction of H_t is ϕ_i (*i.e.*, the same as $\Delta_{\mathcal{M}}$), and its corresponding eigenvalue is given as $e^{\lambda_i t}$. H_t can be written in integral form [120]

$$H_t f(x) = \int_{\mathcal{M}} k_t(x, y) f(y) dy \ , \tag{5.6}$$

where $x, y \in \mathcal{M}$ and dy denotes the volume form on $\mathcal{M}$. $k_t(x, y)$ is known as the *heat kernel*, and is the fundamental solution to the heat equation. The main properties of the heat kernel are

- k_t is C^∞ smooth in $\mathcal{M} \times \mathcal{M} \times \mathbb{R}$.
- k_t is positive.
- k_t is symmetric:

$$k_t(x, y) = k_t(y, x) \ . \tag{5.7}$$

- k_t satisfies the heat equation:

$$\frac{\partial}{\partial t} k_t = \Delta_{\mathcal{M}} k_t \ . \tag{5.8}$$

- k_t satisfies the semi-group identity:

$$k_{t+s}(x, y) = \int_{\mathcal{M}} k_t(x, z) k_s(z, y) dz \ . \tag{5.9}$$

For a compact manifold $\mathcal{M}$, $k_t(x, y)$ has the following eigendecomposition

$$k_t(x, y) = \sum_{i=0}^{\infty} e^{\lambda_i t} \phi_i(x) \phi_i(y) \ . \tag{5.10}$$

In $\mathbb{R}^n$, the heat kernel is given as the Gaussian

$$k_t(x, y) = \frac{e^{-\|x-y\|^2/4t}}{(4\pi t)^{n/2}} \ . \tag{5.11}$$

In our scale-space formulation of surface signals given in Eq. (5.3), we effectively used

$$k_t(x, y) = \frac{e^{-D^2(x,y)/4t}}{(4\pi t)^{n/2}} , \tag{5.12}$$

in place of the heat kernel; $D : \mathcal{M} \times \mathcal{M}$ was defined as the function measuring the geodesic distance between two points on $\mathcal{M}$. We extend the definition of D to measure the Euclidean distance when in $\mathbb{R}^n$. From these observations, we can see that the heat kernel $k_t : \mathcal{M} \times \mathcal{M} \times \mathbb{R}$ in Eq. (5.6) can be replaced by $k'_t : \mathbb{R} \to \mathbb{R}$,

$$k'_t(r) = \frac{e^{-r^2/4t}}{(4\pi t)^{n/2}} , \tag{5.13}$$

to obtain

$$H_t f(x) = \int_{\mathcal{M}} k_t(x, y) f(y) dy = \int_{\mathcal{M}} k'_t(D(x, y)) f(y) dy . \tag{5.14}$$

This reveals that the right hand side of Eq. (5.6) is in fact a convolution, and confirms that k_t is the fundamental solution for the heat operator.

5.1.2 Scale-Space Representation of Discrete Surface Signals

In the previous sections, we presented the scale-space formulation for a continuous surface signal. In this section, we show how the definition can be extended to discrete surfaces represented by polygonal meshes. Let our discrete surface be represented by the polygonal mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$, and $\mathcal{E} = \{e_{ij} | v_i \text{ is connected to } v_j\}$ denote the vertex and edge sets, respectively. The scale-space representation of a mesh signal $f : \mathcal{V} \to \mathbb{R}$ can be obtained by integrating Eq. (5.1) using the forward Euler method as

$$F^{l+1} = F^l + \partial_t F^l = F^l + \lambda \Delta_{\mathcal{M}} F^l , \tag{5.15}$$

with the initial condition $F^0(\cdot) = f(\cdot)$, and time step $0 < \lambda < 1$. The Laplacian, $\Delta_{\mathcal{M}}$, of the signal at vertex v_i at level l, is defined as [126, 28]:

$$\Delta_{\mathcal{M}} F^l(v_i) = \sum_{j \in \mathcal{N}(i)} w_{ij} (F^l(v_j) - F^l(v_i)) \,, \tag{5.16}$$

where $\mathcal{N}(i)$ denotes the 1-ring vertex set of v_i, and w_{ij} is a positive weight associated with the edge connecting vertices v_i and v_j. The edge weights are selected such that $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$. A simple choice for w_{ij} is the inverse of valence of the vertex v_i; *i.e.*, $w_{ij} = \frac{1}{|\mathcal{N}(i)|}$.

The constraint on the step size makes the application of the above scale-space formulation to large meshes nearly infeasible. To overcome this limitation, we employ the implicit surface fairing approach of Desbrun *et al.* [28], which is a backward Euler integration method, to construct the scale-space representation of discrete surface signals: let $F^l : \mathcal{V} \to \mathbb{R}$ denote the smoothed discrete surface signal at level l, and define

$$\mathbf{F}^l = \begin{pmatrix} F^l(v_1) & \cdots & F^l(v_N) \end{pmatrix}^\top \,. \tag{5.17}$$

The smoothed surface signal at level $l+1$ (*i.e.*, $\mathbf{F}^{l+1}$) is obtained by solving the following sparse system of linear equations

$$(\mathbf{I} - \lambda_l \mathbf{L}) \mathbf{F}^{l+1} = \mathbf{F}^l \,, \tag{5.18}$$

where $\lambda_l > 0$ is a time step, and $\mathbf{L}$ and $\mathbf{I}$ denote the $N \times N$ Laplacian and identity matrices, respectively. The elements of the Laplacian matrix $\mathbf{L} = (w_{ij})_{N \times N}$ are given as

$$w_{ij} = \begin{cases} -1 & \text{for } i = j \,, \\ \frac{1}{|\mathcal{N}(i)|} & \text{for } j \in \mathcal{N}(i) \,, \\ 0 & \text{otherwise.} \end{cases} \tag{5.19}$$

The Laplacian matrix may also be populated with other types of weights, such as cotan weights of [28]. The linear system in Eq. (5.18) can be solved efficiently using the Conjugate Gradient method. Note that in the above formulation, the limit on the maximum time step size is lifted.

Consequently, the scale-space representation of surface signal $\mathbf{f}$ is given by the sequence $(\mathbf{F}^0, \ldots, \mathbf{F}^{L-1})$, which is obtained iteratively using

$$\mathbf{F}^l = \begin{cases} (\mathbf{I} - \lambda_{l-1}\mathbf{L})^{-1}\mathbf{F}^{l-1} & \text{if } l > 0 \\ \mathbf{f} & \text{if } l = 0\ , \end{cases} \tag{5.20}$$

for $l = 0, \ldots, L-1$; $(\lambda_0, ..., \lambda_{L-2})$ denotes the sequence of time steps taken at each level in the stack.

The resulting transfer function of the implicit Laplacian smoothing in Eq. (5.18) is

$$h(\omega) = (1 + \lambda_l \omega^2)^{-1}\ , \tag{5.21}$$

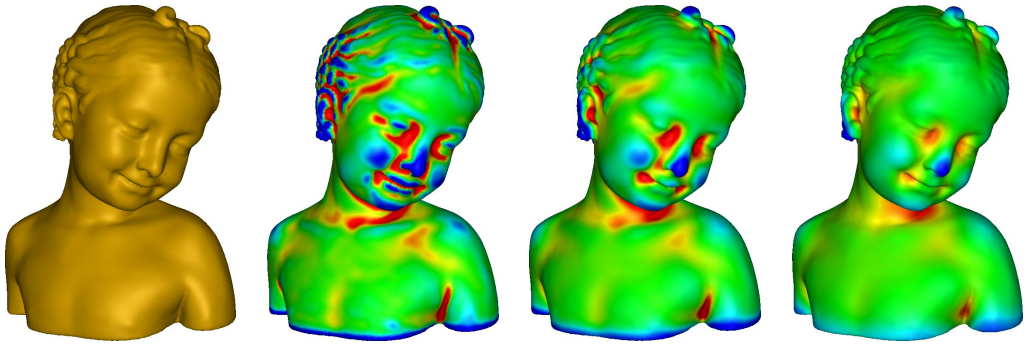
where ω denotes surface signal frequency [28]. When a stack of smoothed signals with L levels is constructed according Eq. (5.20), with corresponding time steps $(\lambda_0, \ldots, \lambda_{L-2})$, the transfer of the filter at level $L-1$ is given by

$$h_{L-1}(\omega) = \prod_{l=0}^{L-2} (1 + \lambda_l \omega^2)^{-1}\ . \tag{5.22}$$

Note that the representation needs to be defined in an iterative manner, since the transfer function of the filter defined by Eq. (5.18) is not a Gaussian. On the other hand, the resulting transfer functions of the iterative formulation approach Gaussians, as L grows.

In our implementation, the time steps are selected as

$$\lambda_l = \lambda_{l-1}\delta = \lambda_0 \delta^l\ , \tag{5.23}$$



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

Figure 5.2: (a) original model; the CS3 representation of the model at scales (b) $t = 3.0$, (c) $t = 7.5$, (d) $t = 13.8$.

where λ_0 denotes an initial time step and $\delta > 1$ is a constant. Fig. 5.2 shows a 3D model and its corresponding CS3 representation at various scales.

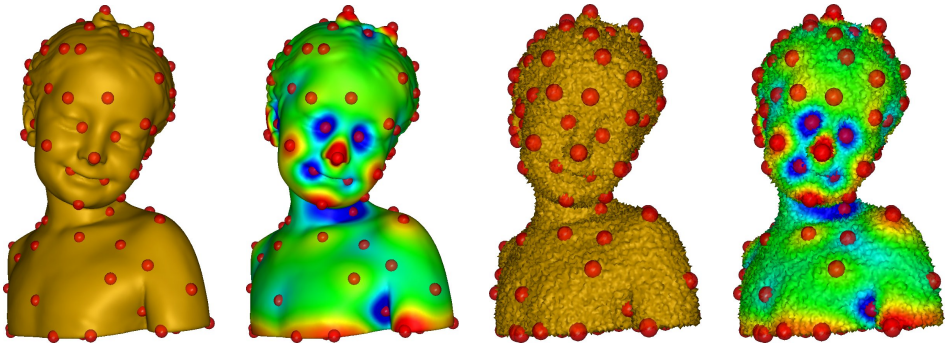
Estimating the Scale Parameter in CS3

The time steps λ_l do not have the same meaning as the scale parameter, t, in the original scale-space representation of signals as given by Eq. (5.1) and Eq. (5.2). To overcome this issue, at each level l, we fit a Gaussian to the transfer function of the smoothing filter for that level, and define the scale of the smoothed signal as the variance (scale) of the fitted Gaussian. This is done by sampling h_l over its domain and obtaining the set of pairs $\Gamma = \{(\omega_j, h_l(\omega_j))\}_{j=0}^{J-1}$. These pairs are then used to estimate the scale, t_l, of a fitted Gaussian $g_l(\omega, t_l) = e^{-\omega^2 t_l}$, in the least-squares sense:

$$t_l = \frac{-\sum_{j=0}^{j<|\Gamma|} \omega_j^2 \ln(h_l(\omega_j))}{\sum_{j=0}^{j<|\Gamma|} \omega_j^4} = \frac{\sum_{j=0}^{j<|\Gamma|} \omega_j^2 \sum_{k=0}^{k<l-1} \ln(1+\lambda_k \omega_j^2)}{\sum_{j=0}^{j<|\Gamma|} \omega_j^4} \,. \tag{5.24}$$

Alternatively, the scale parameter t_l we seek for each level l could be defined in terms of the variance of the transfer function at that level:

$$t_l = \frac{\int_{-\infty}^{\infty} \left(\prod_{k=0}^{l-1} (1+\lambda_k \omega^2)^{-1} \right) d\omega}{\int_{-\infty}^{\infty} \left(\omega^2 \prod_{k=0}^{l-1} (1+\lambda_k \omega^2)^{-1} \right) d\omega} \,. \tag{5.25}$$



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

Figure 5.3: Extracted features on (a) original, and (c) noisy Bimba models at $t = 21.7$; the false-colors in (b) and (d) reflect the response of the Δ^{si} (Eq. (5.31)) at each vertex on the original and noisy models, respectively. The models in (c) and (d) contain 80% Gaussian noise.

Since the transfer function at each level is analytic and only depends on the known sequence of time steps, λ_l, its variance can be precomputed numerically. In this work, we use Eq. (5.24) to estimate the scale parameter. The obtained sequence of scales, $(t_0, \ldots, t_{L-1})$, together with the stack of smoothed signals, $(\mathbf{F}^0, \ldots, \mathbf{F}^{L-1})$, define the CS3 representation of the surface.

5.2 Feature Extraction with Automatic Scale Selection

The CS3 representation of a 3D surface may be used directly for feature extraction. Let $\Phi(\mathcal{M}) = (\mathbf{F}^0, \ldots, \mathbf{F}^{L-1})$ and $\Psi(\mathcal{M}) = (t_0, \ldots, t_{L-1})$ correspond to the CS3 representation of surface $\mathcal{M}$. The difference between the smoothed signals at two consecutive levels l and $l+1$ can be used to approximate the Laplacian of the signal at level l. This difference can be stated in terms of convolution of the original signal with Gaussian filters as

$$\mathbf{F}^{l+1} - \mathbf{F}^{l} \approx \mathbf{F}^{0} * \left(g(\cdot; t_{l+1}) - g(\cdot; t_l)\right), \tag{5.26}$$

where $*$ denotes convolution defined over the surface, and $g(\cdot; t_l)$ is a Gaussian with scale t_l. Noting that $\frac{\partial g}{\partial t} = 0.5\Delta g$, we have

$$\frac{\partial g}{\partial t} = 0.5\Delta g \approx \frac{g(\cdot; t_{l+1}) - g(\cdot; t_l)}{t_{l+1} - t_l} , \tag{5.27}$$

and consequently,

$$\mathbf{F}^{l+1} - \mathbf{F}^l \approx 0.5(t_{l+1} - t_l)\mathbf{F}^0 * \Delta g . \tag{5.28}$$

Therefore, the estimated Laplacian of $\mathbf{F}^0$, at level l, which we denote by $\Delta\mathbf{F}^l$, is approximated by

$$\Delta\mathbf{F}^l \approx \frac{2(\mathbf{F}^{l+1} - \mathbf{F}^l)}{t_{l+1} - t_l} . \tag{5.29}$$

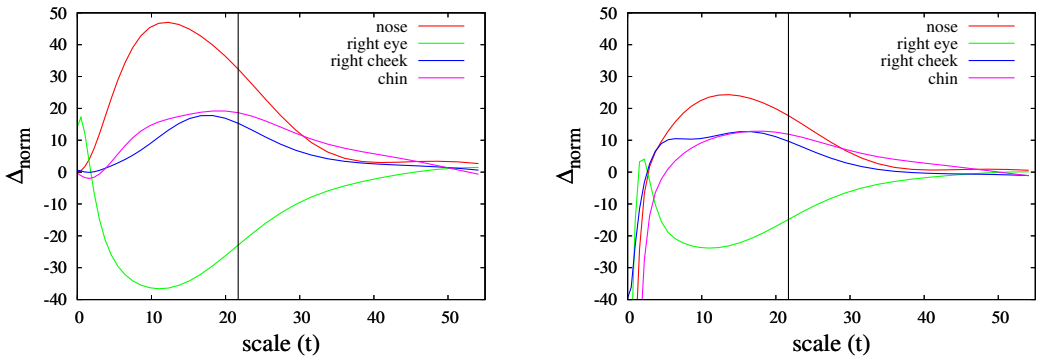
We define the *scale-normalized* Laplacian of the surface signal at scale t_l as

$$\Delta_{norm}\mathbf{F}^l = t_l\Delta\mathbf{F}^l = \frac{2t_l(\mathbf{F}^{l+1} - \mathbf{F}^l)}{t_{l+1} - t_l} . \tag{5.30}$$

Throughout this work, we assume the surface signal corresponds to the surface mean curvatures. $\Delta_{norm}\mathbf{F}$ then corresponds to the scale-normalized Laplacian of mean Curvatures (LoC).

The local extrema of $\Delta_{norm}\mathbf{F}$ could be used to define feature points on a 3D model. For example, Fig. 5.3 depicts the computed scale-normalized Laplacian of mean curvatures on a 3D model and its noisy counterpart, at level $l = 20$ (scale $t = 21.7$); the red spheres indicate the locations where LoC is locally maximum or minimum at the displayed level. As seen in the figure, the detected locations of the extrema of LoC, despite their high differential order, are robust against noise and may be used for extraction of stable and well-localized feature points. Additionally, note that the extracted features cover the entire surface.

The plots in Fig. 5.4 show the computed LoC values at a few selected vertices on the models in Fig. 5.3 as a function of scale. As expected, the values for both the noisy and



(a) original (b) noisy

Figure 5.4: Plots of LoC values of a few vertices on the surfaces in Fig. 5.3. The vertical black lines indicate the location of the displayed scale ($t = 21.7$) in Fig. 5.3.

noise-free models converge at the higher scales. However, the corresponding LoC values of the vertices at the scale shown in Fig. 5.3 are not the same between the two models due to the noise. To alleviate this, we introduce the *scale-invariant* LoC as

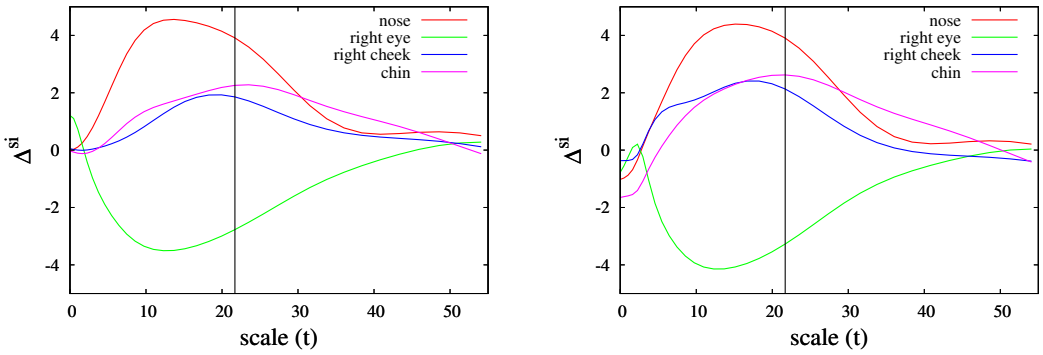
$$\Delta^{si}\mathbf{F}^l = \frac{\Delta\mathbf{F}^l - \bar{\mathbf{F}}^l}{\sigma_l} , \tag{5.31}$$

where

$$\bar{\mathbf{F}}^l = \frac{1}{N}\mathbf{1}^\top\Delta\mathbf{F}^l\mathbf{1}, \quad \sigma_l = \frac{1}{\sqrt{N}}\|\Delta\mathbf{F}^l - \bar{\mathbf{F}}^l\| , \tag{5.32}$$

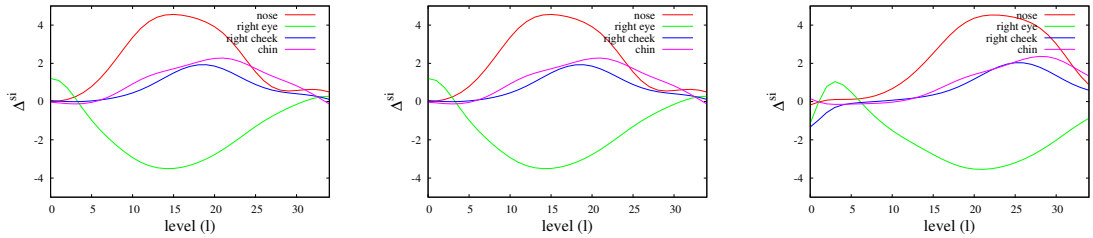
denote the vector-form mean, and standard deviation of the LoC values at level l, respectively; N is the total number of vertices in $\mathcal{M}$, and $\mathbf{1}$ is an N-dimensional vector of all 1's.

Fig. 5.5 shows the scale-invariant LoC plots of the same vertices as in Fig. 5.4. As can be seen, the LoC curves of the two surfaces begin to converge at a much finer scale, and look more similar. The scale-invariant LoC is resilient to changes in resolution, spatial scaling, and additive i.i.d. noise. Additionally, Fig. 5.6 provides a comparison between the Δ^{si} plots on the original, scaled, and higher resolution versions of the same model as in Fig. 5.3(a). The higher resolution version of the model was obtained by applying one iteration of Loop's subdivision scheme, which increases the number of mesh vertices by a



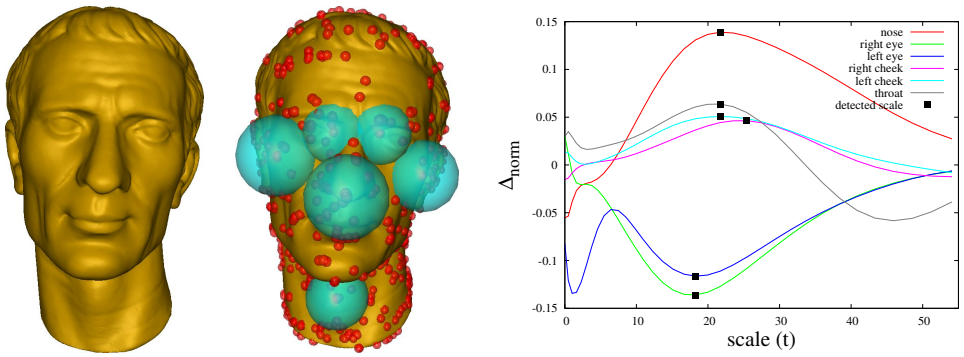
(a) original (b) noisy

Figure 5.5: Plots of the *scale-invariant* LoC values of a few vertices on the surfaces in Fig. 5.3.



(a) original (b) spatial scaling: $\times 100$ (c) sampling resolution: $\times 4$

Figure 5.6: Comparison of scale-invariant LoC plots of the Bimba model (Fig. 5.3(a)) with different spatial scales and sampling resolutions. Plot in (b) is identical to the plot for the original model, shown in (a), while (c) has been shifted to the right by approximately 7 levels. Note that, unlike the plots in Figs. 5.4 and 5.5, the x-axis denotes the level in the CS3 stack, and not the scale.



(a) (b)

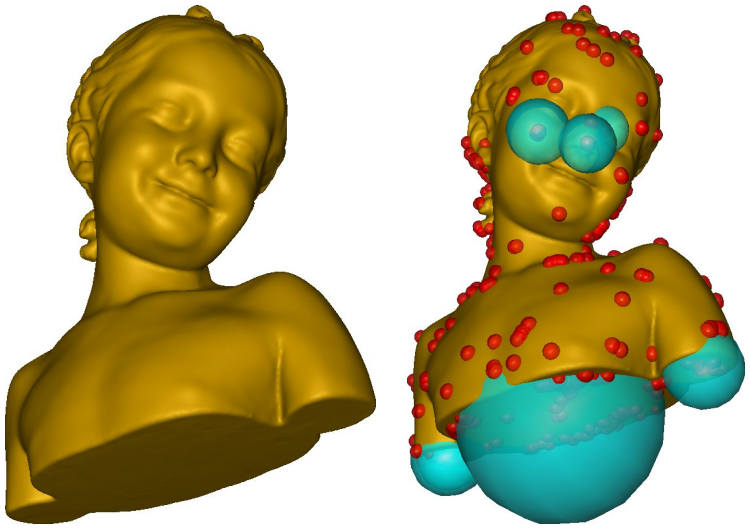
Figure 5.7: Automatic scale selection on the Caesar model. (a) Original model. (b) Estimated scales at a few locations on the model; the radii of the blue spheres are computed using Eq. (5.33). (c) Plots of the scale-normalized Laplacian of the surface mean curvatures at the selected vertices as functions of scale; the locations of the filled squares on the scale-axis indicate the detected scale for the keypoints.

factor of 4. As can be seen, spatial scaling of the model has no effect on the plotted Δ^{si} curves, while the increase in the resolution of the surface shifts the curves to the right.

According to the principle of automatic scale selection [72], the scale(s) where $\Delta_{norm}\mathbf{F}_i$ becomes a local extremum across scales can be expected to correspond to the size of surface structures at vertex v_i. This is visually verified in Figures 5.7 and 5.8, where the size of the blue spheres indicate the computed spatial scale (neighborhood size) at a few selected keypoints. An approach similar to Lowe's [77] was used to select the keypoints (shown as red spheres) on the models, in the two figures. The keypoints were selected as the vertices that were local extrema among their immediate neighbors, both on the current level and two adjacent levels on the stack: let set $Q^l(i) = \{\mathbf{F}_j^{l+k}\} \cup \{\mathbf{F}_i^{l-1}, \mathbf{F}_i^{l+1}\}$, for $k = -1, 0, 1$, and $j \in \mathcal{N}(i)$. Then, vertex v_i, at level l, is selected as a keypoint if $\mathbf{F}_i^l > q_j, \forall q_j \in Q^l(i)$ or $\mathbf{F}_i^l < q_j, \forall q_j \in Q^l(i)$. Let t_l be the scale associated with level l. t_l then defines the scale of the detected keypoint v_i. The radius of each blue sphere in Figures 5.7 and 5.8 was computed using

$$r = t_l \bar{e}\ , \tag{5.33}$$

where $\bar{e}$ is the average edge length in the surface mesh.



(a) (b)

Figure 5.8: Automatic scale selection on the Bimba model. (a) Original model; (b) estimated scales at a few locations on the model.

The graph in Fig. 5.7(c) shows the plots of LoC values at the few selected keypoints (blue spheres) on the model in Fig. 5.7(a). The filled squares on the curves indicate the location of the detected scale for each keypoint.

5.3 Error Propagation in Scale-Space Representation of Surface Signals

In this section we attempt to determine how the error or uncertainty in the surface curvature values is propagated during the smoothing process. Let the relation between the true and estimated curvature values at each vertex i be given as

$$\hat{F}_i = F_i + A_i X_i \,, \tag{5.34}$$

where F_i is the noise-free (true) curvature value at vertex i, A_i is a scale factor, and X_i is a random variable with probability density function δ_i which describes the additive noise;

i.e., $X_i \sim \delta_i(x)$. δ_i is assumed to have zero mean and standard deviation σ_i.

In the explicit smoothing case, one iteration of the Laplacian smoothing at vertex i at level l is given as

$$\hat{F}_i^{l+1} = \hat{F}_i^l + \lambda \Delta \hat{F}_i^l \ , \tag{5.35}$$

where $0 < \lambda < 1$ is the time step size, and Δ again denotes the discrete Laplacian computed at i. Expanding the above equation yields

$$\begin{aligned} \hat{F}_i^{l+1} &= F_i^l + A_i X_i + \lambda \left(\textstyle\sum_{j \in \mathcal{N}(i)} w_j (F_j^l - F_i^l + A_j X_j - A_i X_i) \right) \\ &= F_i^l + \lambda \textstyle\sum_j w_j (F_j^l - F_i^l) + \left[A_i X_i + \lambda \textstyle\sum_j w_j (A_j X_j - A_i X_i) \right] \\ &= \underbrace{F_i^l + \lambda \sum_j w_j (F_j^l - F_i^l)}_{G_i^l} + \underbrace{\left[(1-\lambda) A_i X_i + \lambda \sum_j w_j A_j X_j \right]}_{Z} \ , \end{aligned} \tag{5.36}$$

where $Z \sim \psi(x)$ is a new random variable. Note that

$$G_i^l = F_i^{l+1} = F_i^l + \lambda \sum_j w_j (F_j^l - F_i^l) \ . \tag{5.37}$$

That is, G_i^l corresponds to the smoothed *ideal* signal at level $l+1$.

Under the assumption that all X_i and X_j's are independent, $\psi(x)$ is given as

$$\psi(x) = \underbrace{\lambda^J (1-\lambda) A_i \prod_{j \in \mathcal{N}(i)} w_j A_j}_{B_i^l} \underbrace{\left[\delta_i(x) * \bigotimes_{j \in \mathcal{N}(i)} \delta_j(x) \right]}_{\phi_i^l(x)} \ , \tag{5.38}$$

where $*$ denotes convolution and $J = |\mathcal{N}(i)|$; we define $\bigotimes_{j=1}^{J} \delta_j(x) = \delta_1(x) * \cdots * \delta_J(x)$. By introducing random variable $Y_i^l \sim \phi_i^l(x)$, the smoothed curvature value at vertex i is given as

$$\hat{F}_i^{l+1} = G_i^l + B_i^l Y_i^l \ . \tag{5.39}$$

Note that even though the convolution process that produces $\phi_i^l(x)$ results in a distribu-

tion with a larger standard deviation, the associated scale factor B_i^l is small and $B_i^l Y_i^l$ is fast decreasing as a function of l, and $\lim_{l\to\infty} B_i^l Y_i^l = 0$. This implies that after "large enough" number of iterations, n, we have the expected behavior: $\hat{F}_i^n \approx F_i^n$; again, F_i^n denotes the value of the *ideal* smoothed curvature value at vertex i, and $\hat{F}_i^n$ denotes the smoothed *noisy* curvature values after n iterations.

In the implicit Laplacian smoothing case, the smoothed values at level $l+1$ are given as the solution to the linear system

$$(I - \lambda L)F^{l+1} = F^l \ . \tag{5.40}$$

The relation can equivalently be written as

$$\begin{aligned} F^{l+1} &= (I-\lambda L)^{-1} F^l \\ &= (I + \lambda L + \lambda^2 L^2 + \lambda^3 L^3 + \lambda^4 L^4 + \cdots) F^l \\ &= (I+\lambda L)F^l + (\lambda^2 L^2 + \lambda^3 L^3 + \lambda^4 L^4 + \cdots) F^l \ . \end{aligned} \tag{5.41}$$

We note that the implicit smoothing is equivalent to the explicit smoothing plus an infinite sum of higher order Laplacians of the signal. The non-zero elements in row i of the first order Laplacian matrix correspond to vertex i and its 1-ring neighbors. The non-zero elements in row i of matrix L^2 correspond to vertex i and all its 1 and 2-ring neighbors. As the order of the Laplacian matrix increases, the number of non-zero elements in the matrix increases, since more neighbors are included. Therefore, one iteration of the implicit Laplacian is a global smoothing operation, and the corresponding $B_i^l Y_i^l$ in Eq. (5.39) decreases even faster than the explicit smoothing case.

By making the assumption that the probability density function of the noise at each vertex, i, is given by a normal distribution with zero mean and variance σ_i^2 (*i.e.*, $X_i \sim N(0;\sigma^2)$), the resulting probability density function of the noise at vertex i after one smoothing iteration is given as $\phi_i^l(x) = N(0, \sigma_i^2 + \sum_{j\in\mathcal{N}(i)} \sigma_j^2)$. The random variable de-

scribing the noise at the smoothed vertex i, is $Y_i^l \sim \phi_i^l(x)$ and is scaled by B_i^l as described by Equations 5.38 and 5.39.

In a matching application, where two similar surfaces with the same resolution by different levels of noise are given, the above noise propagation scheme can be used to determine how much smoothing of the surface curvatures (or some other function) is needed. Let S and T denote the two surfaces that we are processing. Let σ_S^2 and σ_T^2 (with $\sigma_S^2 \geq \sigma_T^2$) denote the variance of the noise in the estimated curvature values, at each vertex, on surfaces S and T, respectively. Then both surfaces need to be smoothed up to level n, until $E[B_S^n Y_S^n] < \epsilon$, for some small ϵ.

5.4 Discussion

In the previous sections, we presented a scale-space representation for 3D surfaces which was directly defined on the polygonal surface and did not require surface parametrization or voxelization. We showed how the representation may be used for feature extraction and automatic scale selection and that the extraction process is robust to noise. We called the representation the Curvature Scale Space 3D (CS3). The CS3 representation has additional properties, which are not all present in other proposed scale-space extensions (see Chapter 3) for 3D surfaces. That is,

- the representation does not result in geometric distortions or degeneracies such as shrinkage or self-intersections.[1]
- the representation is computed directly on the polygonal surface without the need for parametrization or voxelization. Additionally, the smoothing operation is performed in a consistent manner with the metric tensor over the surface.
- the representation provides a good estimate of the linear diffusion process.

[1]The representation does not directly smooth the geometry. However, if the surface corresponding to the smoothed curvatures is reconstructed, these degeneracies will be absent or minimal for genus 0 surfaces.

Representation	Parametrization/ Voxelization free	Efficient (scalable)	Distortion free	Auto-scale capable	Approximates Linear SS
CS3	Yes	Yes	Yes	Yes	Yes
Parametrization [93]	No	Yes	No	Yes	Yes
Voxelization [94]	No	Yes	No	Yes	Yes
Mean curvature flow [109]	Yes	No	No	Yes	No
Multiscale MLS [101, 70]	Yes	No	No	No	No
Mesh saliency [63]	Yes	No	No	Yes	Yes
Surface variation [100]	Yes	No	No	Yes	Yes

Table 5.1: Comparison of the different scale-space representations for 3D surfaces.

- the representation is capable of automatic scale recovery.
- the representation is efficient.
- the representation is stable against noise.

Table 5.1 provides a summarized comparison of the CS3 representation with other scale-space representations for 3D surfaces.

Applications. The CS3 representation of a surface has many applications. We have already seen its application in robust feature extraction. Additionally, the representation enables robust local descriptor computations based on differential attributes of the surface. These, in turn, lead to applications in shape matching tasks such as registration, recognition, retrieval, and classification. In the next section, we discuss such an application in 3D surface registration. In Chapter 6, we show another application of our proposed representation in 3D face recognition. CS3 also has applications outside the realm of computer vision. For

example, using reconstruction technique of [32], CS3 may be used for multiscale surface fairing without the artifacts that are generally present in techniques that directly smooth the geometry. Additionally, the representation may be used to implement the mesh saliency of [63] more efficiently. This will expand the range of applications to mesh simplification and automatic viewpoint selection. CS3 together with curvature-domain processing technique of [32] may also be used for 3D shape morphing.

5.5 Application: Surface Registration

The feature extraction procedure described in the previous section together with the CS3 representation of the surface can be used in 3D matching tasks such as surface registration and 3D object recognition. Given two surfaces, A and B, with arbitrary positions and orientations in 3D, the goal is to align the two surfaces with each other such that the total distance between corresponding points on the two surfaces is minimized. If one knows the correct point correspondences between the two surfaces (*i.e.*, which point on surface B corresponds to a given point p on surface A), then registering the two surfaces is trivial; the surfaces can be aligned with each other with at least three correct correspondences, but more point correspondences will yield better registration results [44, 6]. However, in most cases, no initial point correspondence is known between surfaces and this needs to be established automatically. In this section we describe how the CS3 representation can be used to establish such correspondences automatically, and ultimately register the two surfaces.

5.5.1 Outline of the Method

Given two input surfaces A and B with arbitrary position and orientation, our proposed registration algorithm consists of the following stages:

1. **CS3 Computation.** The CS3 representations of the two surfaces are obtained.

2. **Feature Point Extraction.** The CS3 representation of each surface is used to extract geometric feature points on each surface.

3. **Principal Frame Computation.** Principal Component Analysis (PCA) is used to assign a local coordinate system, referred to as the *principal frame*, to each feature point on the two surfaces.

4. **Local Descriptor Computation.** Local descriptors are computed at the extracted feature points using the estimated LoC values in the neighborhood of the feature points.

5. **Registration.** Finally, correspondences are established between points on the two surfaces using the computed local descriptors and principal frames. These correspondences are then used to register the two surfaces.

In this section, we skip the description of the first step of the algorithm since it has already been discussed in the previous sections.

5.5.2 Feature Point Extraction

After the CS3 stack is built for each input surface, the local extrema of the estimated Δ_{norm} of surface curvatures (cf. Eq. (5.30)) are selected as candidate feature points at each level in the stack. The features, which are the most persistent across scales are selected as the features that are used in the registration process. We assign a scale to each feature point by picking the highest level i where the Δ_{norm} of the curvature response is greater than the response at levels $i-1$ and $i+1$. If no such level can be found, the feature point is assigned the maximum possible scale.

Fig. 5.9 shows the extracted features and their associated scales on the Bimba model and its noisy version. Note the robustness of the extraction and scale assignment process to noise, despite the high differential order of the features. The associated scale at a feature

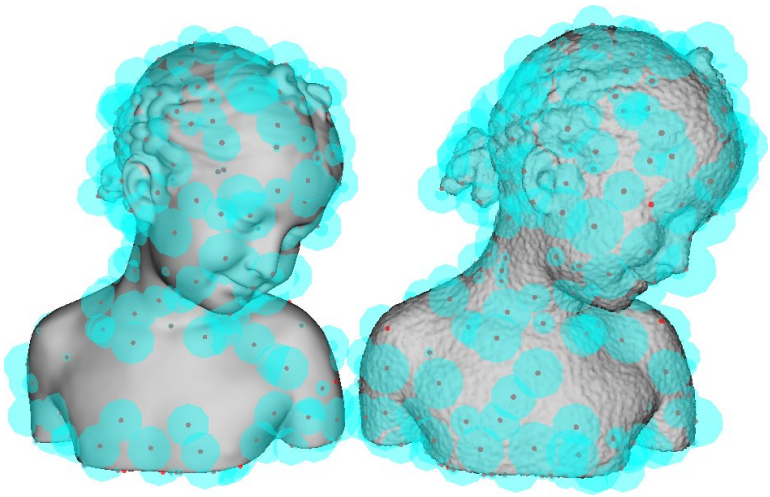


Figure 5.9: Feature extraction and scale assignment on the Bimba model and its noisy version. The small red spheres indicate the position of the extracted features and the size of the transparent blue spheres are used as indicators of the detected scale at the features; the radii of the transparent spheres were picked proportional to the detected scales at the features.

determines the level in the CS3 stack where its descriptor is computed in step 4 of the registration algorithm. The scale assignment makes the point-point matching process robust to noise and variations in mesh resolutions between the models.

5.5.3 Principal Frame Computation

In matching applications, having the ability to assign a transformation-invariant local coordinate frame to an interest point is essential, though not always necessary. This extra information can dramatically improve the performance (robustness and time-efficiency) of the matching application at hand (*e.g.*, see [62, 38, 45]).

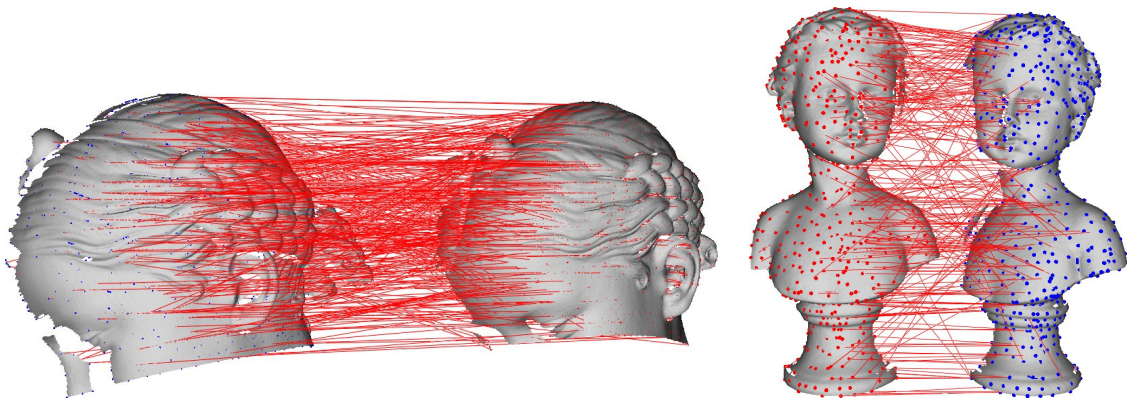
In 2D images, the gradient of image intensities at the pixel of interest is generally used to assign an orientation to the point. The equivalent of such an approach for 3D surfaces is to compute the gradient of the signal at the mesh vertices. We computed the gradients of the smoothed signal at various scales and then used a weighted averaging scheme similar to [77] to assign a tangent orientation to the point of interest. This approach, however, was

not robust to large levels of noise. We speculate the problem mainly lies in the way the gradients are computed at the vertices; in the implementation, we used a simple weighted finite difference method to compute the gradient of the signal at the vertices of the mesh. For 2D images, however, the gradient field of the input image is generally obtained by convolving the image with a Gaussian derivative. The CS3 representation may also be used for robust directional derivative computations—this is left for future work.

The robustness of the LoC map to noise and its coherence to the local geometry of the 3D shape make it a good candidate for use in local shape description and consistent orientation assignment. In our final implementation, we used the principal component analysis of the local neighborhood around the feature points to recover the principal directions. The neighborhood size was selected proportional to the detected scale at the features and the covariance matrix was weighted using the LoC values of the neighboring vertices. This weighting scheme dramatically improved the matching performance (as opposed to using uniform or Gaussian weighting). The eigenvectors of the weighted covariance matrix provide a set of three orthonormal directions, which can be used to construct a local frame at the point of interest on the surface. This local frame is referred to as the principal frame. The principal directions, however, provide only direction and not the orientation of the vectors. The information about the normal orientation at the vertex can be used to resolve the orientation ambiguity along the normal direction. The ambiguity of the other two directions on the tangent plane, however, cannot be easily resolved. We show how to deal with the remaining orientation ambiguity when we discuss point-point matching.

5.5.4 Local Feature Descriptors

Both the curvature and LoC maps of the local neighborhoods of the feature points on the surface provide a relatively robust and discriminative description of the local geometry at the points. The descriptors which use the LoC values are more distinctive than those that use the curvature information. On the other hand, the estimated LoC values in the



(a) (b)

Figure 5.10: Point-point matching using local shape descriptors; for each feature point on the left models in (a) and (b), a point with the most similar descriptor was found as its correspondence on the other model.

CS3 representation may be dependent of the mesh resolution and perform poorly when the resolution of the two surfaces are dramatically different. This is why the scale assignment step in the feature extraction process is crucial.

The local descriptors in our final implementation were defined in terms of the weighted histogram of the LoC values in the neighborhoods of the feature points. Given an associated neighborhood size with a feature point, we compute the LoC histogram of all its neighboring vertices. Each histogram bin is then scaled by the corresponding bin's LoC value. The histograms are represented in the form of high-dimensional descriptor vectors. The L_2 distance of these vectors is then used to measure the similarity between the feature points.

5.5.5 Matching and Registration

Using the extracted feature points and their corresponding descriptor vectors, we perform point-point matching which is then used to register the models. Even if the vertex matching process is not accurate, this enables us to build a vertex priority queue based on the descriptor distances which is then used to pick points to estimate the transformation between the

models. The extracted feature points and the their corresponding descriptors are then used to automatically register two 3D surfaces.

Let's assume M_1 is the reference scan and M_2 is the second scan that needs to be brought into the same coordinate frame as M_1. Our registration algorithm proceeds as follows:

First, feature points are extracted on both M_1 and M_2. This yields the point sets $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$ for M_1 and M_2, respectively. Let r_i be the associated neighborhood size for each feature point p_i. Using PCA, a local coordinate frame is then assigned to each feature point on both scans. We then extract local descriptor vectors at each feature point and store them in a high-dimensional kd-tree data structure such as the Approximate Nearest Neighbor (ANN) [91].

For each point $p_i \in P$, we find the set of its K-nearest neighbors in Q. That is, for each $p_i \in M_1$, we obtain a set of possible candidate matches $L_i = \{l_1, \ldots, l_K | l_i \in Q\}$. In our implementation we used $K = 5$. Fig. 5.10 shows an example of point-point matching between pairs of partial scans. For each feature point on the left model, in both (a) and (b), the figure shows the first match in L_i that was found on the right model. Note that, even though, some of the matches are not correct, a correct match may be present in the remaining candidates in L_i.

At least three points are required to align M_2 with M_1. At this stage, we use the associated local frames for each feature point $p_i \in P$, to obtain two additional points on M_1. Let v_{i1} and v_{i2} be the two principal directions tangent to the surface at p_i. Two intermediate points $g_j = p_i + v_{ij} r_i$ (with $j = \{1, 2\}$) are computed. The two additional points for p_i on M_1 are then obtained as the closest feature points on the mesh to g_1 and g_2. For each possible match, $l_j \in L_i$, for p_i, we use the same procedure to obtain two other points near l_j on M_2. This yields a three point-pair correspondence between M_1 and M_2 and is used to recover the missing transformation, T, between the models. Notice that, since we don't know the orientation of v_{ij}, the process needs to be repeated with $g_j = p_i - v_{ij} r_i$.

The point-point matching procedure described in the previous paragraphs is used to obtain a set of possible transformations $\mathbf{T} = \{T_1, \ldots, T_k\}$ that may be used to bring M_2 into alignment with M_1. For each $T_k \in \mathbf{T}$, we then compute the error functional

$$E_k = \sum_{p_i \in P} \|p_i - T(q_i)\| \,, \tag{5.42}$$

where q_i is a feature point on M_2 which is closest to $T^{-1}(p_i)$. Finally, the T_k which yields the smallest E_k is used to register M_2 with M_1.

We tested our proposed algorithm in a few pairwise surface registration tasks. Fig. 5.11 shows the registration results without use of a local registration algorithm such as the Iterative Closes Point (ICP).

5.6 Application: Line Extraction with Automatic Scale Selection

In this section, we briefly describe how the CS3 representation may be used to robustly extract crest lines [128, 43] on 3D models. Additionally, we show preliminary results for using the scale selection mechanism to detect the varying scales on the extracted lines.

Let κ_0, κ_1 (with $|\kappa_0| \geq |\kappa_1|$) be the principal curvatures with corresponding principal directions $\vec{t}_0, \vec{t}_1$, at a point on the surface. κ_0 is referred to as the *maximal principal curvature* (or simply *maximal curvature*). The extremality coefficient e is defined as the directional derivative of the maximal curvature in the corresponding principal direction: $e = \langle \vec{\nabla}\kappa_0, \vec{t}_0 \rangle$. Depending on the sign of κ_0, e is referred to as the minimal (e_{min}, when $\kappa_0 < 0$) or maximal (e_{max}, when $\kappa_0 > 0$) extremality coefficient. The zero-crossings of e_{max} correspond to ridges on the surface, and the zero-crossings of e_{min} correspond to valleys. Here, we do not distinguish between ridges and valleys and only work with the absolute value of κ_0. Therefore, in short, crest lines are defined as the locus of points

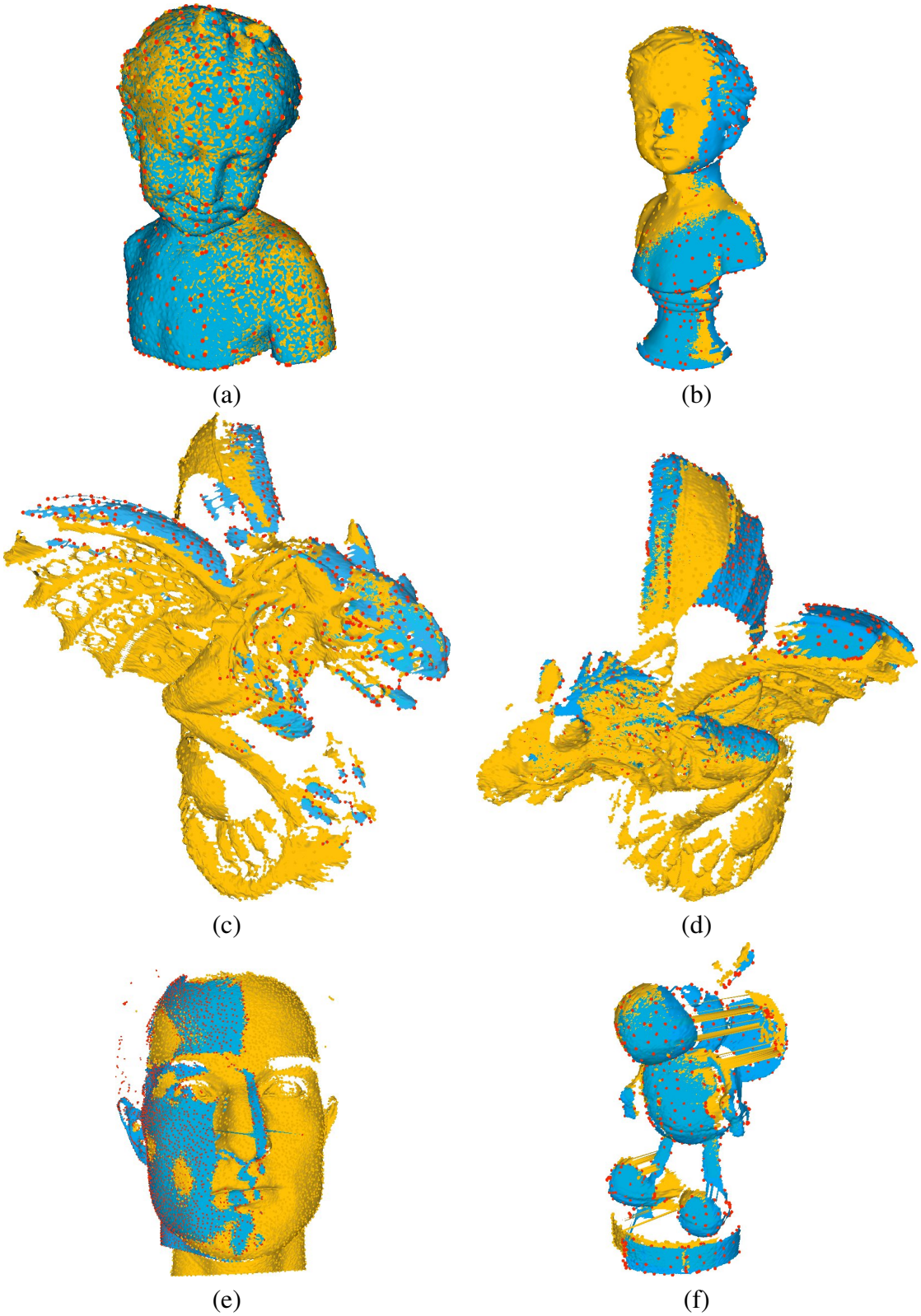
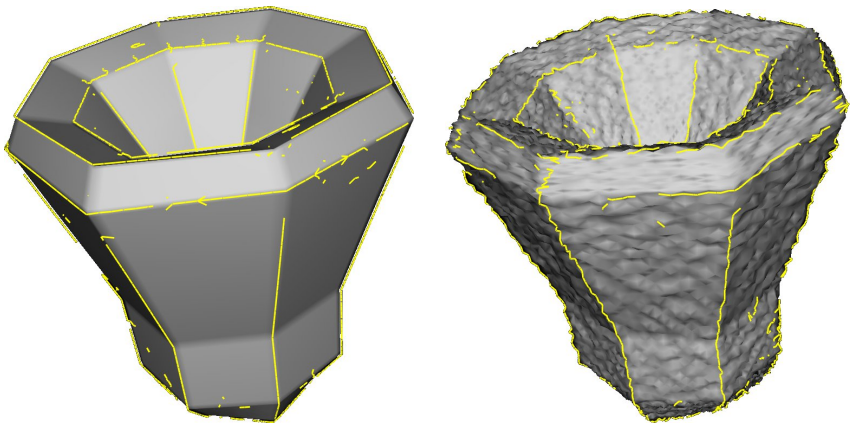


Figure 5.11: Results of the proposed pairwise registration algorithm on a few sets of scans: (a) complete Bimba model with its noisy version, (b) two partial scans of a buste model, (c) two partial scans of the Gargoyl model, (d) different scans of the Gargoyl model, (e) two partial facial scans, (f) partial scans of a bee model.



(a) (b)

Figure 5.12: Extracting crest lines on (a) a cup model, and (b) its noisy version.

where:

$$e = \langle \vec{\nabla}\kappa_0, \vec{t_0} \rangle = 0, \qquad \text{and} \quad \langle \vec{\nabla}e, \vec{t_0} \rangle < 0 \; . \tag{5.43}$$

Starting from an initial estimation of the curvature tensors at the vertices on the surface, we proceed by first building the CS3 representation of the surface. Since, we also need the principal directions, when computing the extremality coefficients, the CS3 stack is obtained a bit differently from the original formulation. Instead of smoothing the principal curvatures on the surface, we smooth the curvature tensor itself. The principal curvatures and directions are subsequently extracted from the smoothed curvature tensors at each level.

Finally, the extremality coefficients and their derivatives are computed using the smoothed curvatures. The crest lines are extracted by finding the edges on the polygonal mesh where the zero-crossings of the coefficients occur. Fig. 5.12(a) shows the extracted crest lines on a cup model. Fig. 5.12(b) shows the result of the same extraction procedure applied to a noisy version of the same model. As can be seen, despite the fact that extracting the crest lines requires up to the 4^{th} order derivative computations, the crest line extraction procedure is stable in the presence of noise.

The scale selection mechanism may be used to recover the scale along each of the extracted crest lines. We define the edge strength at a point p on the surface as $s(p) =$

(a) (b) (c)

Figure 5.13: Crest line extraction with automatic scale selection: (a) extracted crest lines on the Bimba model, (b,c) recovered scales at the lines; the thickness of the lines is chosen proportional to the recovered scales at the individual points on the lines.

$|\kappa_0(p)| - |\kappa_1(p)|$. Consequently, we use the following scale-normalized edge strength for scale selection:

$$s_{norm}(p) = t_i^{1/2} s(p) = t_i^{1/2}(|\kappa_0(p)| - |\kappa_1(p)|) \; . \tag{5.44}$$

For each point on an extracted crest line, we look at the scale-normalized edge strengths, across scales. The first level where a local maximum is found is assigned as the scale associated with the point. Fig. 5.13 shows the extracted crest lines and their recovered scales on a 3D model. This scale selection mechanism may be used to combine all the extracted lines at the various scales in the stack to obtain one representative set of crest lines for the model. This is left as future work.

5.7 Summary

In this chapter, we introduced the Curvature Scale Space 3D (CS3) representation for surfaces. We showed its clear advantages over the previous scale-space based methods. The proposed representation is computationally efficient, insensitive to noise, and capable of automatic scale recovery. We showed how the CS3 representation may be used to perform

feature extraction on 3D models with automatic scale selection. Applications of the CS3 representation and the proposed feature extraction technique were discussed for various shape matching tasks. We showed how CS3 may be used in an automatic surface registration system. Even though, the presented registration algorithm was crude, it produced acceptable results due to the robustness of the CS3 representation.

In future work, we will improve the efficiency of the CS3 computation by performing mesh decimation at the coarser levels of the CS3 stack. Additionally, we did not perform any post-processing of the extracted features. Some of the features belonged to regions with low surface variations or lied along surface edges. The removal of these features will improve the performance in the shape matching tasks. We are investigating how the CS3 representation may be used for robust directional derivative computations on the surface. Having such a capability will help in robust extraction of other types of features, such as crest lines, on 3D surfaces.

Chapter 6

3D Face Recognition using CS3

"If I make the lashes dark
And the eyes more bright
And the lips more scarlet, ...
I'm looking for the face I had
Before the world was made."
–William Butler Yeats

In this chapter, we present applications of our proposed CS3 representation in automatic 3D face detection and matching. In Sec. 6.1, we show an application of our approach to a partial shape matching task involving the detection of human faces on 3D models without any assumptions about the scale of the models. In Sec. 6.2, we use the outputs of our face detection system in a 3D face recognition system and show how our multiscale representation can be used to optimize performance of the recognition system.

Fig. 6.1 shows examples of input surfaces on which we seek to find the location and

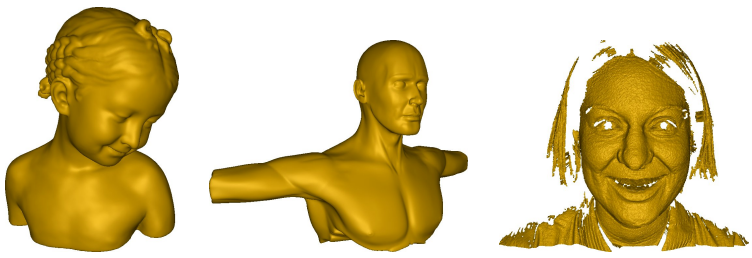
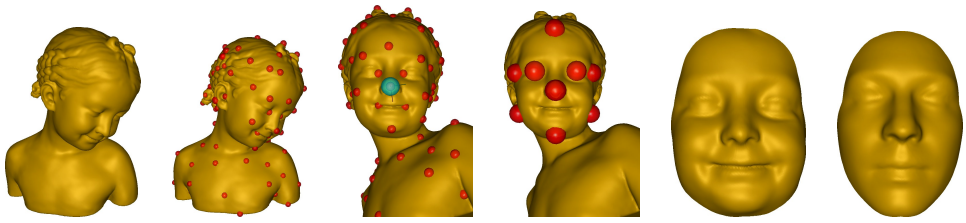


Figure 6.1: Examples of possible inputs to our face detection system.



(a) (b) (c) (d) (e) (f)

Figure 6.2: Multiscale 3D face detection pipeline. (a) original model; (b) extracted keypoints; (c) detected location, orientation, and scale of the face region; (d) auxiliary facial features extracted from the face region; (e) cropped and remeshed face region; (f) reference face.

extent of a human face. Note that these surfaces come from different sources, have arbitrary scale, and may also contain facial expressions. To provide an insight into our face detection approach, we first present the general steps involved in the automatic detection of human face on arbitrary 3D models. In Fig. 6.2, we show the general steps involved in the detection task: first, a set of interesting keypoints are extracted on the surface of the model (Fig. 6.2(b)). Using these extracted keypoints, the relative location, orientation, and scale of the model (Fig. 6.2(c)) is obtained with respect to a reference face model (Fig. 6.2(f)). Using this information, the locations of other important facial features are identified (Fig. 6.2(d)). Finally, using the new set of facial features in Fig. 6.2(d), the face region is extracted from the input model (Fig. 6.2(e)).

The face extraction process generates a remeshed version of the cropped face in a manner consistent between all models. Therefore, the output of the face detection process can directly be fed into a 3D face recognition algorithm. Note that unlike the majority of automatic 3D face detection systems [84, 7], the proposed detection scheme does not make any assumptions about the relative pose or scale of the input models. In Sec. 6.2, we show how the CS3 representation can be used to form feature vectors, which are used in the face recognition system. We also show that due to the stability and discriminative power of the CS3 representation, our recognition system is capable of outperforming the current state-of-the-art 3D face recognition systems.

6.1 Application: Automatic Face Detection

In this section, we describe how our proposed CS3 representation can be used to detect the face region on 3D surfaces containing human faces with arbitrary scale, orientation, and translation. The detection system consists of two main stages. In the offline (training) stage, a classifier is trained with a set of 3D human faces, which consists of faces both with and without expressions. The online (matching) stage involves the actual detection of the face on an input 3D surface. Sec. 6.1.1 describes the training stage, while the matching stage of the algorithm is described in Sec. 6.1.2.

6.1.1 Face Detection: Training

The objectives of the training phase are:

- *Facial feature identification and AFM computation:* special facial features (*e.g.*, the nose tip, eyes, chin, *etc*) are manually specified by a user on the 3D models in the training set. These features are used to align the 3D face models in the training set with each other. The aligned models are then cropped so that only the face region is included and consequently an Average Face Model (AFM) is computed by averaging all the cropped faces in the training set. The AFM is later used in the matching stage of the face detection algorithm.

- *Face region descriptor computations:* local shape descriptors are computed on the nose tips of the faces in the training set. These descriptors are used to define the distance (dissimilarity) measure between 3D surfaces that is needed to detect the face region in the matching stage. This is done by converting the descriptors into feature vectors and estimating the class conditional probability density function (pdf) of the face class in the feature space.

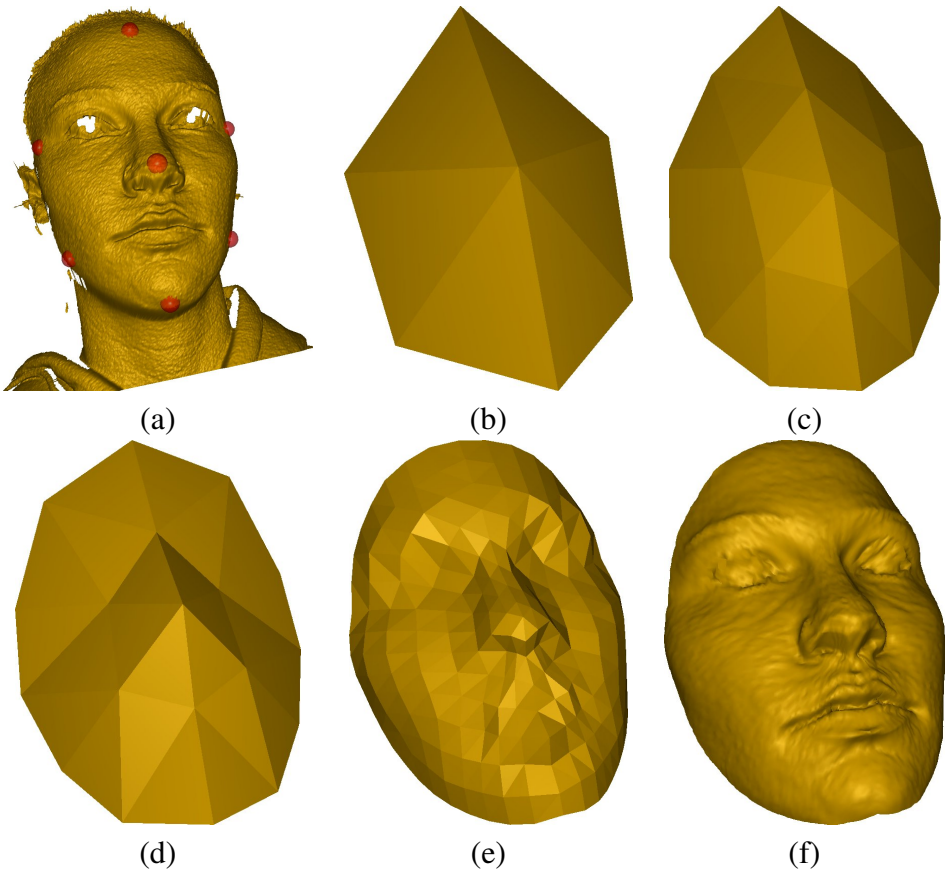
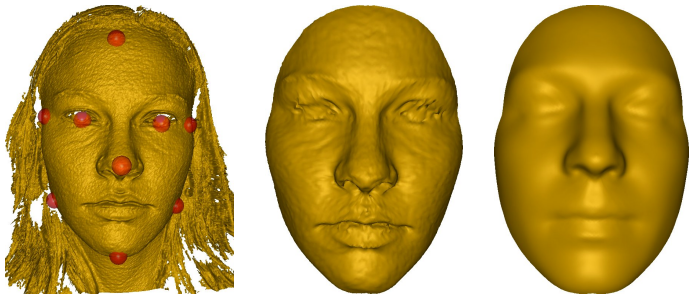


Figure 6.3: Face mask generation steps; (a) the initial points defining the domain mesh are selected on the input 3D model; (b) the initial domain mesh is obtained; (c) the domain mesh after 1 butterfly subdivision; (d) projection of the subdivided domain mesh onto the input face model; (e,f) the results of the subdivision and projection procedure applied to the initial domain mesh after 3 and 6 iterations, respectively.

Facial Feature Identification and AFM Computation

In this step, a user manually selects a predefined set of facial feature points on each 3D face in the training set. Fig. 6.4(a) shows the locations of the features used by our detection system. The facial features are used to obtain a cropped version of the input surfaces such that only the face region is contained. We use a similar approach to [69] to obtain the cropped and remeshed surfaces. The resulting cropped surface is referred to as the "face mask". In Fig. 6.3, we summarize the steps involved in creating the face mask from an input surface. The facial feature points are additionally used to register all the face masks with each other using the approach of [6], followed by an Iterative Closest Point (ICP) [13] iteration to refine the alignment. The registered face masks are then averaged together to obtain the AFM. An example of the AFM is shown in Fig. 6.4(c).



(a) (b) (c)

Figure 6.4: (a) Facial feature selection on a 3D surface; (b) cropped face region; (c) average face model obtained by averaging a few of the cropped faces in the training set.

Face Region Descriptor Computations

Local shape descriptors are used to detect the face region on an input 3D surface. In our implementation, the descriptors are height maps, which are in each case centered at a point with an associated set of basis vectors. In Sec. 6.1.2, we describe how these descriptors and their corresponding feature vectors are computed. In this section, we show how these local descriptors are used to estimate the probability that a given point and its neighboring region

on a 3D surface correspond to a human face; the face region is assumed to be centered at and represented by the nose tip.

Let c_f denote the "face class" in the feature space. The objective of this step in the training phase is to estimate the class conditional pdf $p(\mathbf{z}|c_f)$, where $\mathbf{z} \in \mathbb{R}^D$ denotes a point in the feature space. This information is then used to define a dissimilarity metric in the feature space, which measures the distance of all feature points from the face class. Throughout, it is assumed that $p(\mathbf{z}|c_f)$ can be estimated by a D-dimensional Gaussian with mean μ_{c_f} and covariance matrix $\mathbf{\Sigma}_{c_f}$. Under this assumption, the mean and covariance matrix of the distribution is estimated from the feature vectors in the training set as

$$\mu_{c_f} = \frac{1}{M} \sum_{\mathbf{z} \in \mathbf{Z}_{c_f}} \mathbf{z}_m \ , \quad \mathbf{\Sigma}_{c_f} = \frac{1}{M} \sum_{\mathbf{z} \in \mathbf{Z}_{c_f}} (\mathbf{z}_m - \mu_{c_f})(\mathbf{z}_m - \mu_{c_f})^\top \ , \tag{6.1}$$

where $\mathbf{Z}_{c_f} = \{\mathbf{z}_m\}_{m=1}^M$ denotes the set of feature vectors from the face class in the training set. In cases, where it is known that the distribution is multi-modal, and the above Gaussian assumption does not hold, more complicated models such as mixture of Gaussians may be used [86, 105].

Under the Gaussian assumption, the squared Mahalanobis distance,

$$d^2_{c_f}(\mathbf{z}) = (\mathbf{z} - \mu_{c_f})^\top \mathbf{\Sigma}^{-1}_{c_f} (\mathbf{z} - \mu_{c_f}) \ , \tag{6.2}$$

is the sufficient statistic for $p(\mathbf{z}|c_f)$. Therefore, in the matching stage, given an input 3D surface, $\mathcal{M}$, the goal is to find a point $\mathbf{p} \in \mathcal{M}$, with corresponding feature vector $\mathbf{z} \in \mathbb{R}^D$, which minimizes Eq. (6.2).

The high dimensionality of the feature vectors generally prevents one from directly computing Eq. (6.2). Additionally, in most cases, the number of samples in the training set is much smaller than the dimension of the feature space ($D \gg M$). This results in a singular covariance matrix $\mathbf{\Sigma}_{c_f}$, and therefore, Eq. (6.2) cannot be computed reliably. Even if the covariance matrix is non-singular, evaluating Eq. (6.2) is computationally expensive,

as it involves two D-dimensional matrix-vector multiplications.

The most popular class of approaches for dealing with the above-mentioned problems is to use subspace projection methods such as the Principal Component Analysis (PCA). In these approaches, the major principal components of the data in the training set are found and the data from both the training and test sets are projected onto the subspace defined by these components. The main problem with this class of approaches is that too much information is discarded. In [86], the authors propose a PCA-based probabilistic approach that retains information about the less significant principal components of the data, which is generally discarded in other PCA-based approaches. They show this modification, as expected, improves the performance of the detection system. We follow a similar approach in our detection system.

Let

$$\mathbf{\Sigma}_{c_f} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \tag{6.3}$$

denote the eigendecomposition of $\mathbf{\Sigma}_{c_f}$, where orthogonal matrix $\mathbf{U} = (\mathbf{u}_1 \cdots \mathbf{u}_D)$ and diagonal matrix $\mathbf{\Lambda} = (\lambda_{i,j})_{D\times D}$ contain the eigenvectors and the corresponding eigenvalues of $\mathbf{\Sigma}_{c_f}$, respectively. The column vector $\mathbf{u}_d$ denotes the d^{th} eigenvector of $\mathbf{\Sigma}_{c_f}$ and its corresponding eigenvalue is given by $\lambda_d = \lambda_{d,d}$. The eigenvalues in $\mathbf{\Lambda}$ are assumed to be ordered in the descending order; *i.e.*, $\lambda_1 \geq \cdots \geq \lambda_D$. If $\mathbf{\Sigma}_{c_f}$ is full rank, then the eigenvectors $\mathbf{u}_d$ span the feature space, and any point $\mathbf{z} = (z_1 \ldots z_D)^\top \in \mathbb{R}^D$ in this space can be given new coordinates

$$\mathbf{y} = \mathbf{U}^\top(\mathbf{z} - \bar{\mathbf{x}}_{c_f}) = \mathbf{U}^\top\hat{\mathbf{z}}\,, \tag{6.4}$$

where $\hat{\mathbf{z}} = (\mathbf{z} - \bar{\mathbf{x}}_{c_f})$, and $\mathbf{y} = (y_1 \ldots y_D)^\top \in \mathbb{R}^D$. The squared Mahalanobis distance of Eq. (6.2) is given in terms of $\mathbf{y}$ as

$$d^2_{c_f}(\mathbf{z}) = \hat{\mathbf{z}}^\top\mathbf{\Sigma}^{-1}_{c_f}\hat{\mathbf{z}} = \hat{\mathbf{z}}^\top\mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^\top\hat{\mathbf{z}} = \mathbf{y}^\top\mathbf{\Lambda}^{-1}\mathbf{y} = \sum_{d=1}^{D}\frac{y_d^2}{\lambda_d}\,. \tag{6.5}$$

To reduce the dimensionality of the data, and as a result increase the computational efficiency and stability of the detection system, the data is projected onto the subspace spanned by the K major eigenvectors of $\mathbf{\Sigma}_{c_f}$, namely, $\mathbf{u}_d$ for $d = 1, ..., K$. The squared Mahalanobis distance, $d^2_{c_f}(\mathbf{z})$ in Eq. (6.5), can then be decoupled in the form

$$d^2_{c_f}(\mathbf{z}) = \hat{\mathbf{z}}^\top \mathbf{\Sigma}^{-1}_{c_f} \hat{\mathbf{z}} = \mathbf{y}^\top \mathbf{\Lambda}^{-1} \mathbf{y} = \sum_{d=1}^{D} \frac{y_d^2}{\lambda_d} = \sum_{d=1}^{K} \frac{y_d^2}{\lambda_d} + \sum_{d=K+1}^{D} \frac{y_d^2}{\lambda_d} \,. \tag{6.6}$$

Since a partial eigendecomposition of $\mathbf{\Sigma}_{c_f}$ with K major eigenvalues is obtained, the first summation in the above equation can be computed in a straightforward manner. The second summation can only be approximated since the remaining eigenvalues are not recovered. Define $\epsilon^2(\mathbf{z})$ as the squared Euclidean distance between $\mathbf{z}$ and its projection onto the principal eigenspace,

$$\epsilon^2(\mathbf{z}) \;\; = \textstyle\sum_{d=K+1}^{D} y_d^2 = \|\hat{\mathbf{z}}\|^2 - \sum_{d=1}^{K} y_d^2 \,. \tag{6.7}$$

It can be shown [86] that the second summation in Eq. (6.6) is approximated as

$$\sum_{d=K+1}^{D} \frac{y_d^2}{\lambda_d} \approx \frac{1}{\rho} \sum_{d=K+1}^{N} y_d^2 = \frac{\epsilon^2(\mathbf{z})}{\rho} \,, \tag{6.8}$$

where

$$\rho = \frac{\sum_{d=K+1}^{D} \lambda_d}{D - K} \,. \tag{6.9}$$

λ_d, for $d = K + 1, \cdots, D$, are estimated by fitting a function of the form $f(x) = 1/x$ to the known principal eigenvalues.

Using the above observations, given a feature vector $\mathbf{z}$, the dissimilarity measure of Eq. (6.2) is approximated as

$$d^2_{c_f}(\mathbf{z}) \approx \sum_{d=1}^{K} \frac{y_d^2}{\lambda_d} + \frac{\epsilon^2(\mathbf{z})}{\rho} \,. \tag{6.10}$$

As mentioned previously, in the matching stage, the candidate point whose feature vector

yields the smallest dissimilarity is picked as the nose tip.

6.1.2 Face Detection: Matching

The main goal of the online stage of the algorithm is to identify the location, orientation, and scale of the face region on an input surface. This region is centered at the nose tip because the nose bridge helps establish orientation and symmetry, and its high curvature is easily detected using our keypoint extraction method. This is accomplished in the following steps:

1. *Keypoint extraction:* the CS3 representation of the surface with L levels is constructed. The keypoints at each level are selected as the mesh vertices whose LoC values are locally maximum or minimum. Additionally, a local scale is assigned to each extracted keypoint.

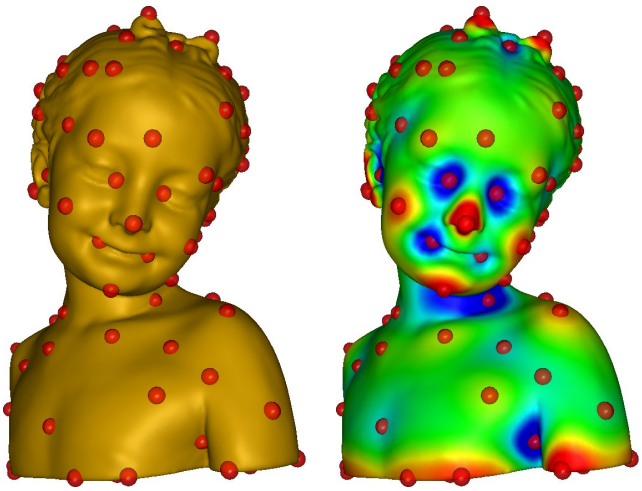
2. *Scale alignment:* using the extracted keypoints and their corresponding scales, the relative scale of the input surface with respect to a reference face model in the training set is found, and consequently adjusted.

3. *Local descriptor computation:* the local geometry of the surface and its CS3 representation are used to construct multiple local coordinate systems and shape descriptors at each extracted keypoint.

4. *Detection:* a classifier is used to determine which keypoint with its associated local shape descriptor corresponds to the most likely location of the face region on the input surface.

Keypoint Extraction and Local Scale Estimation

Keypoint extraction is performed by first building the CS3 representation of the surface and consequently finding the local extrema of the estimated Laplacian of surface curvatures



(a) (b)

Figure 6.5: Extracted keypoints on a 3D model at scale $t = 21.7$; the false-colors in (b) reflect the response of the Δ^{si} (Eq. (5.31)) at each vertex on the model.

(LoC) in the constructed stack (see Eq. (5.30)). An approach similar to Lowe's [77] is used to select the keypoints on the input model. See Sec. 5.2 for more details on the approach.

Note that since the signal that is being smoothed in our representation is the estimated surface mean curvatures, the extracted keypoints correspond to locations of blob-like features on the surface; this is visually verified in Fig. 6.5. Since we are interested in identifying the location of the nose tip, these features provide an ideal choice for this task. Additionally, the size of each blob is estimated by the associated scale of the keypoint at that location using Eq. (5.33).

Global Scale Adjustment

One advantage of estimating the size of the surface structures, as described in Sec. 5.2 and Sec. 6.1.2, is that the computed values are intrinsic to the surface. By intrinsic we mean that scaling (resizing) or changing the resolution (number of vertices per unit area) of the surface will not significantly affect the detected radii of the structures *relative* to the actual structure sizes. For example, if the surface in Fig. 6.6(a) is scaled by a factor of 10, the size of the nose on the surface as well as the *automatically detected* radius of the keypoint on

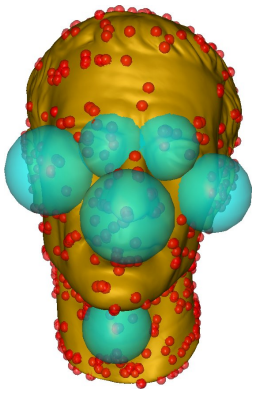


Figure 6.6: Automatic scale selection at a few locations on the surface of the Caesar model.

the nose tip as estimated by Eq. (5.33) will be scaled by the same factor.

The relative scale of an input 3D surface with respect to the 3D face models in the training set can therefore be estimated by comparing the detected sizes of the surface structures. To do this, the median radius of the surface structures on the surfaces in the training set are computed offline. Similarly, the median radius of the structures on the input 3D surface are computed. The ratio of these radii are then used to estimate and then adjust the relative scale of the input model with respect to the models in the training set.

Local Shape Descriptor Computation

Since we are interested in detecting human faces on a wide range of 3D models, instead of using the LoC values—which are of high differential order—we choose to use the positional information of the vertices. Given a keypoint and the estimated size of the face region (obtained from the faces in the training set), we construct a local coordinate system at the point and represent the positions of its neighboring vertices in the form of a height map. We use the Multilevel B-Splines Algorithm (MBA) [64], to obtain a continuous map of the local neighborhood around the keypoint, which is then turned into an image; the MBA helps fill the gaps in the image, which are present due to missing data in the input surface. The resulting height image defines the local descriptor employed by our detection system. Fig. 6.7 shows examples of the constructed height maps at two locations on a 3D

model.

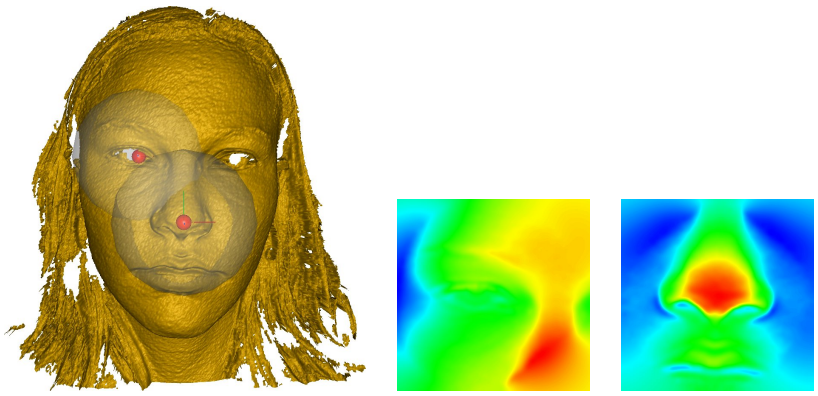
The local coordinates at the keypoints need to be constructed in a manner which is invariant to any transformation the 3D model may undergo. We use PCA of the local neighborhood around each point of interest to construct the local coordinate system. Let p_0 denote the keypoint where the local coordinate system is being constructed, and let $k_1 \geq k_2 \geq k_3 \geq 0$, u_1, u_2, u_3 denote the eigenvalues and unit eigenvectors of the covariance matrix of the vertex positions in the neighborhood around p_0, respectively. u_3 approximates the normal direction, while u_1 and u_2 span the tangent plane at p_0. u_3 defines the z direction of the constructed local coordinate system, and u_2 and u_1 define the directions of the x and y axes, respectively.

Since the eigenvectors provide only information about direction and not orientation, we make the following adjustments: u_3 is adjusted so that it has the same orientation as the surface normal at p_0, while the orientations of u_1 and u_2 are changed such that $u_2 \times u_1 = u_3$. u_1 and u_2 may still be rotated by 180 degrees. To overcome this ambiguity, we create two height maps using the two possible orientations. In the detection stage, both maps are used, and the one with the worst performance is discarded.

In our implementation, the covariance matrix was weighted by the LoC values at each vertex. The weighting scheme helped the two major eigenvectors of the covariance matrix to be better-aligned with the underlying elongated structures of the surface at each keypoint.

Detection

The goal of the detection stage is to find the location of the face region on an input 3D surface by using the computed local descriptors at the extracted keypoints. Each local descriptor is converted into a feature vector by concatenating the rows in its height map image. Let $\mathbf{z}_i$ denote the feature vector associated with keypoint i, and let $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{I}$ be the set of all such feature vectors extracted on the input surface. The ideal objective of detection would then be to identify all $\mathbf{z}_i$'s which correspond to a face region—and as a



(a) (b) (c)

Figure 6.7: Computed height maps at two locations of a 3D model. The large transparent spheres in (a) show the radii of the neighborhoods used when computing the height maps. The descriptors for the eye and nose regions are shown in (b) and (c), respectively.

result, having the ability to identify multiple faces on the same model. However, in this work, we seek to solve a simpler problem, namely, finding the most likely keypoint that belongs to the face region. The detection task is therefore to find $\mathbf{z}^*$, minimizing $d^2_{c_f}(\mathbf{z}_i)$ among all $\mathbf{z}_i \in \mathbf{Z}$, where $d^2_{c_f}$ denotes the estimated squared Mahalanobis distance as defined by Eq. (6.2).

Once the most likely location of the face region is identified on the input surface, we use its location and associated local coordinate system to obtain an initial guess for the relative pose of the input with respect to the AFM in the training set. The estimated position and orientation are then iteratively improved using the ICP algorithm [13]. In each iteration, we also improve the initial guess for the scale by finding s that minimizes

$$E = \sum_{\mathbf{p}_i \in M_{afm}} \min_{\mathbf{q}_j \in M_{in}} \|\mathbf{p}_i - s\mathbf{q}_j\|^2 , \tag{6.11}$$

where M_{in} and M_{afm} denote the point-sets corresponding to the input surface and the AFM, respectively. Both surfaces are assumed to have been translated so that the nose tip is at the origin. This prevents the scaling operations from changing the relative positions of the nose tips between the two surfaces.

We iteratively minimize Eq. (6.11) by first finding

$$\mathbf{q}_i = \underset{\mathbf{q}\in\mathcal{M}_{in}}{\arg\min} \|\mathbf{p}_i - s\mathbf{q}\|^2 \,, \tag{6.12}$$

for each $\mathbf{p}_i \in \mathcal{M}_{afm}$, where s denotes the current guess for the scale. Eq. (6.11) then becomes

$$E_2 = \sum_{i\in|\mathcal{M}_{afm}|} \|\mathbf{p}_i - s\mathbf{q}_i\|^2 \,, \tag{6.13}$$

where $|\mathcal{M}_{afm}|$ denotes the size of the point-set corresponding to surface $\mathcal{M}_{afm}$. Since E_2 is quadratic in s, the s^* minimizing it can be found by taking its partial derivative with respect to s and setting the result to zero:

$$s^* = \frac{\sum_{i\in|\mathcal{M}_{afm}|} \mathbf{p}_i^\top \mathbf{q}_i}{\sum_{i\in|\mathcal{M}_{afm}|} \mathbf{q}_i^\top \mathbf{q}_i} \,. \tag{6.14}$$

The resulting s^* then replaces the current guess for scale s in Eq. (6.11) and the above steps are iterated K times.

Once the input surface has been aligned with the AFM, the locations of auxiliary facial features, which were specified in the training phase, are identified on the surface. Fig. 6.8(a) shows the locations of these features, which were automatically detected by our system on an input model. These features are then used to crop and remesh the face region using a subdivision scheme similar to [69] (see Fig. 6.3 for a summary of the steps involved). Fig. 6.8(b) shows an example of the final result produced by our face detection procedure. Note that in [69], the locations of the facial features on the input model were specified manually, whereas our system detects them automatically.

6.1.3 3D Face Detection Results

We tested the performance of our detection system on a dataset of 1068 models, which included artificial models with human faces and more than 1000 3D scanned human faces

(a) (b)

Figure 6.8: (a) Automatically detected facial features, and (b) the resulting extracted face region.

from the FRGC dataset [34]. The majority of the FRGC scans where from Spring 2003, which included large portions of clothing. All models were randomly scaled (by a factor in the range $(0, 1000]$) and rotated prior to detection. The correct detection rate of the system was 92.13%. The required time to build the CS3 representation with 35 levels for a model with approximately $113.5K$ vertices was 32secs on a 2.4GHz CPU. The overall detection time on the same model was 180secs.

As mentioned previously, the output of the detection system, which results in normalized and compatibly remeshed 3D surfaces, can directly be used in a 3D face recognition system. Additionally, the extracted 3D faces may be used in automatic 3D processing tasks such as swapping of facial features between 3D faces. We employed the differential coordinates approach of [2] to enable a user to automatically specify and swap facial features between various face models. For example, Fig. 6.9 shows an example of a warping task involving three different faces. Our system allowed the user to easily specify which facial features on the source face (Fig. 6.9(c)) were to be replaced by the ones on the target faces (Fig. 6.9(a) and Fig. 6.9(b)); Fig. 6.9(d) shows the resulting hybrid face. Since the faces were normalized and the meshing was consistent between the surfaces, the user only needed to specify the location and region of influence of the features on the source face.

The resulting hybrid faces may, in turn, be used to generate new faces and extend the size of a training set in a face recognition system.

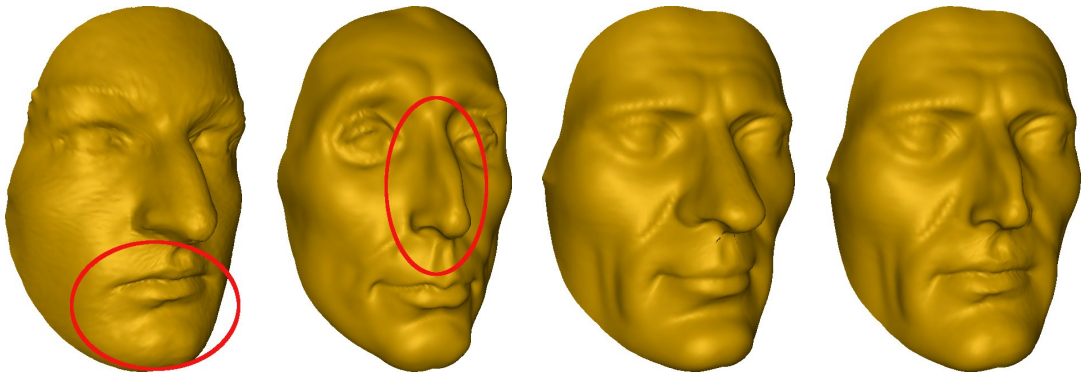


Figure 6.9: Automatic 3D face warping results. (a) target mouth; (b) target nose; (c) source face; (d) resulting hybrid face.

6.2 Application: 3D Face Recognition

We tested the discriminative power of our proposed scale-invariant Laplacians of surface curvatures in a simple PCA-based 3D face recognition system. The face regions on the input 3D surfaces were automatically detected and cropped using the procedure described in the previous section. In cases where the automatic face detection failed, a manual procedure was used to extract the face region from input surfaces.

As mentioned previously, the face mask extraction process results in consistently remeshed face regions. By "consistent," we mean that all meshes have the same number of vertices and a one-to-one correspondence between the vertices on the meshes is known. This property simplifies the process of converting the meshes into feature vectors, which are used for recognition. In the previous section, we used the resulting face masks, to generate hybrid 3D faces. However, the main application in mind for the face masks, as first proposed in [69], is 3D face recognition.

A large body of literature exists on 2D and 3D face recognition [108, 1]. For example,

approaches such as the Bayesian face recognition of [85], or face recognition using sparse representation [141, 69] may be used for this task. However, in our face recognition system, we employ the most well-known approach of eigenfaces [118]. We choose this approach due to its simplicity and ease of implementation. More importantly, this choice enables us to attribute the superior performance of our recognition system (see Sec. 6.2.2) to its feature extraction component, rather than the classifier it employs.

In Sec. 6.2.1, we discuss the steps involved in the training and recognition phases of our algorithm. In Sec. 6.2.2, we present the recognition results of our system on different datasets and compare its performance against other state-of-the-art 3D face recognition techniques.

6.2.1 3D Face Recognition using CS3 Representation

In both the training and recognition stages of our algorithm, each face mask f_m with N vertices is converted into an N-dimensional feature vector by first constructing its CS3 representation and computing the scale-invariant Laplacians of Curvatures (si-LoC) at some level l in the CS3 stack (see Eq. (5.31)); the optimal choice of l is discussed in the results section. The feature vector is then formed by arranging the si-LoC values of the mesh vertices into an N-dimensional vector. Since the face masks for all faces are obtained using the same procedure, the vertices in all meshes have the same ordering and, as a result, the constructed feature vectors are consistent. We denote the feature vector corresponding to face mask f_m by vector $\mathbf{x}_m \in \mathbb{R}^N$.

Each vector $\mathbf{x}_m$ corresponds to a point in the N-dimensional feature space. Under the assumption that the feature vectors are constructed judiciously, multiple feature vectors corresponding to different 3D scans of the *same individual* are expected to form a cluster in the feature space. The objective of the training phase is then to obtain information about the characteristics (*e.g.*, the shapes) of these clusters. In the recognition phase, this information is used to decide to which cluster a given input feature vector belongs. Therefore, in this

paradigm, face recognition is treated as a feature classification problem.

In this work, the eigenfaces approach that we employ is a simple nearest neighbor classifier. As mentioned previously, we show that despite this choice, our system is capable of outperforming most state-of-the-art 3D face recognition techniques. We argue that this good performance is due to both the *discriminative power* of our feature vectors and their *resilience to noise*. In this application domain, the noise may be due to *surface perturbations* or *facial expressions* in the input faces.

3D Eigenfaces

Let $\mathbf{X} = \{(\mathbf{x}_m, c_m)\}_{m=1}^{M}$ be the training set; $\mathbf{x}_m \in \mathbb{R}^N$ and c_m denote the m^{th} feature vector and its associated class in the training set, respectively. Let

$$\mu = \frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\ , \quad \mathbf{\Sigma} = \frac{1}{M}\sum_{m=1}^{M}(\mathbf{x}_m - \mu)(\mathbf{x}_m - \mu)^\top \tag{6.15}$$

denote the mean vector and covariance matrix of the feature vectors in $\mathbf{X}$. The eigendecomposition of $\mathbf{\Sigma}$ is given as $\mathbf{\Sigma} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$, where the $N \times N$ matrices

$$\mathbf{\Phi} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \dots & \mathbf{u}_N \\ | & & | \end{pmatrix} \text{ and } \mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{pmatrix} \tag{6.16}$$

contain the eigenvectors and eigenvalues of $\mathbf{\Sigma}$, respectively. It is assumed that the eigenvalues are ordered in the descending order; *i.e.*, $\lambda_1 \geq \ldots \geq \lambda_N \geq 0$.

In the case of face recognition, where each $\mathbf{x}_m$ is derived from a face model, the K-major eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ are referred to as "eigenfaces". The eigenfaces span a K-dimensional subspace of the feature space with the smallest total orthogonal distance from the feature points in the training set. Therefore, the projection of the feature vectors onto the subspace spanned by the eigenfaces results in dimensionality reduction of the feature

vectors, with the minimal loss of variance. We refer to the subspace spanned by the eigenfaces simply as the *eigenspace*. When $N \gg K$, this dimensionality reduction enables efficient processing of the data, which would otherwise be computationally prohibitive.

In the eigenfaces approach, both the training and test sets are projected onto the eigenspace, and the classification tasks are performed in this space. Let $\mathbf{x}' \in \mathbb{R}^K$ denote the projection of feature vector $\mathbf{x} \in \mathbb{R}^N$ onto the eigenspace:

$$\mathbf{x}' = \mathbf{U}^\top (\mathbf{x} - \mu) \,, \tag{6.17}$$

where

$$\mathbf{U} = \begin{pmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_K \end{pmatrix}_{N \times K} \,. \tag{6.18}$$

Let set $\mathbf{X}' = \{(\mathbf{x}'_m, c_m)\}_{m=1}^M$ denote the transformed training set obtained by projecting the feature vectors in the training set onto the eigenspace. In the recognition phase, classification is performed by assigning each test feature vector $\mathbf{x}_t \in \mathbb{R}^N$ to class c^* of $\mathbf{x}^* \in \mathbf{X}$, which satisfies

$$(\mathbf{x}^*, c^*) = \underset{(\mathbf{x}'_m, c_m) \in \mathbf{X}'}{\arg\min} \|\mathbf{x}'_t - \mathbf{x}'_m\|_p \,, \tag{6.19}$$

where $\mathbf{x}'_t = \mathbf{U}^\top (\mathbf{x}_t - \mu)$, and $\| \cdot \|_p$ denotes the L_p-norm in $\mathbb{R}^K$, for some $p \geq 1$. The optimum choice of p is discussed in Sec. 6.2.2.

6.2.2 3D Face Recognition Results

We tested the performance of our proposed 3D face recognition system on the publicly available GavabDB dataset [89], which contains 3D face scans of 61 individuals. This dataset is much noisier and of lower resolution than the FRGC dataset [34]. The scanned faces for each individual in GavabDB contains the following poses and expressions: 1 scan looking up, 1 scan looking down, 2 frontal scans, 1 scan with random gesture, 1 scan with laughter, and 1 scan with smile.

Pose	Acc.	Group	Group acc.	Overall
Looking down	95.08%	Neutral	96.31%	95.42%
Looking up	93.44%			
Frontal 1	98.36%			
Frontal 2	98.36%			
Random gesture	88.52%	Non-neutral	94.54%	
Laughter	96.72%			
Smile	98.36%			

Table 6.1: Face recognition results on GavabDB. The pose column indicates which class of scans was taken as the test set while the remaining scans were used as the training set. Column 3 indicates which scans contained facial expressions. Columns 4 and 5 show the neutral/non-neutral and overall correct recognition rates of the system, respectively. Level $l = 10$ of the CS3 stack was used to form the feature vectors, and the L_1-norm was as the distance metric in the feature space.

In the first set of tests, we followed the same leave-one-out cross-validation procedure as in [69] to test the correct accuracy rate of our recognition system. In each trial, one class of faces (*e.g.*, scans looking up) were used as the test set and the remaining faces in the dataset were used as the training set. The recognition accuracy was measured as the percentage of times the system returned the correct individual for each query face from the test set. Table 6.1 shows the correct recognition rates of our system for each test set. In the experiments, si-LoC values at level 10 of the CS3 stack were used to form the feature vectors for both the training and test sets. Additionally, the L_1-norm was used in the matching stage of the algorithm, when searching for nearest neighbors in the eigenspace. The test sets in Table 6.1 have been grouped together into two categories of "neutral" and "non-neutral", to indicate which of the sets contained facial expressions. The accuracy rates for the two groups (column 4), and the overall accuracy of the system (column 5) were computed by averaging their associated rows in column 2 of the table.

Since CS3 is a multiscale representation, a level l from the CS3 stack must be selected in order to construct the feature vectors which are used for training and matching. Therefore, l is an unknown parameter whose optimal value must be estimated using the training set. Other parameters that will also influence the performance of the system are the initial time step, λ_0, and the factor, δ, by which the step size is increased at each level in the CS3

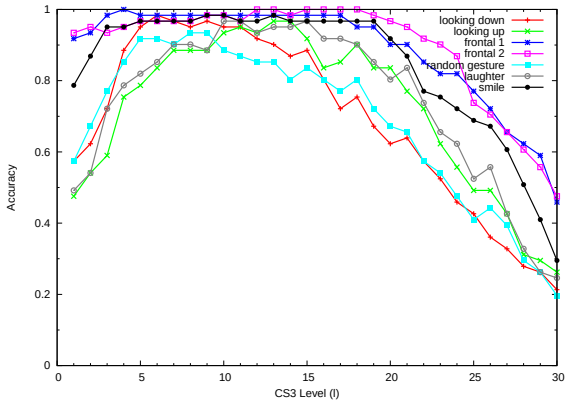


Figure 6.10: Plots of the correct recognition rates of the system for different classes of test sets as functions of the CS3 level. At each level, the si-LoC values were used to form the feature vectors. In these experiments, the L_1-norm was used to find the nearest neighbors in the eigenspace.

stack (see Eq. (5.23)). Throughout this work, we used the following values for these two variables: $\lambda_0 = 1.0$, $\delta = 1.2$. However, the value of l must be picked more carefully as it has a larger influence on the performance of the system.

In Fig. 6.10, we show how the performance of our recognition system, is affected by the choice of l, for each class of test sets. In all cases the accuracy first increases and then decreases. Additionally, Table 6.2 uses the data from Fig. 6.10 to show how the recognition rates for the neutral/non-neutral groups of test sets are influenced by the choice of the CS3 level. As can be seen, again level $l = 10$ yields the optimal performance for both classes of tests.

We define the optimal CS3 level for recognition, as the level where the overall accuracy of the system is maximal. Procedure `FindOptimalLevel` in Alg. (1) summarizes the steps involved in finding the optimal level l^*, using only the training set. The procedure may be iterated a number of times to obtain a set of values for l^*; the arithmetic mean or median of these values may then be used to select the optimal level.

The choice of the distance function used by the classifier is another issue that needs to be investigated. We tested the performance of our system using different distance metrics in Eq. (6.19). Fig. 6.11 and Table 6.3 show the detection results of the same set of experiments

Level	Neutral	Non-neutral	Overall
1	0.7254	0.6174	0.6714
2	0.7622	0.6939	0.7281
3	0.8073	0.8142	0.8107
4	0.8975	0.8633	0.8804
5	0.9221	0.9016	0.9118
6	0.9426	0.9125	0.9275
7	0.9508	0.9234	0.9371
8	0.9467	0.9344	0.9405
9	0.9549	0.9344	0.9446
10	*0.9631**	*0.9454**	*0.9542**
11	0.9631	0.9344	0.9487
12	0.9590	0.9180	0.9385
13	0.9631	0.9289	0.9460
14	0.9508	0.9071	0.9289
15	0.9467	0.9234	0.9351
16	0.9057	0.8961	0.9009
17	0.8893	0.8852	0.8872
18	0.9016	0.8907	0.8961
19	0.8606	0.8469	0.8538
20	0.8319	0.7978	0.8148
21	0.8155	0.7868	0.8012
22	0.7663	0.6939	0.7301
23	0.7172	0.6502	0.6837
24	0.6762	0.6065	0.6413
25	0.6065	0.5409	0.5737
26	0.5696	0.5573	0.5635
27	0.5163	0.4754	0.4959
28	0.4549	0.3770	0.4159
29	0.4262	0.3114	0.3688
30	0.3524	0.2459	0.2991

Table 6.2: The accuracy rates of the system for different choices of the CS3 level where the si-LoC values are picked to form the feature vectors; the L_1-norm was used for matching.

Algorithm 1 `FindOptimalLevel`

Input: Training set $\mathbf{T} = \{(f_m, c_m)\}_{m=1}^{M}$, $p \geq 1$
Output: Optimal CS3 level l^*

Build CS3 stack with L levels
for each level $l \in \{1, \ldots, L\}$ **do**
  for $m = 1$ to M **do**
    Obtain feature vector $\mathbf{x}_m$ from f_m using CS3 values at level l
  end for
  $\mathbf{X} \leftarrow \{(\mathbf{x}_m, c_m)\}_{m=1}^{M}$
  $\mu \leftarrow \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m$
  $\mathbf{\Sigma} \leftarrow \frac{1}{M} \sum_{m=1}^{M} (\mathbf{x}_m - \mu)(\mathbf{x}_m - \mu)^\top$
  Build $\mathbf{U}$ from eigendecomposition of $\mathbf{\Sigma}$ (Eq. (6.18))
  Partition $\mathbf{X}$ into two sets $\mathbf{X}_{train} = \{(\mathbf{x}_r, c_r)\}_{r=1}^{R}$ and $\mathbf{X}_{test} = \{(\mathbf{x}_s, c_s)\}_{s=1}^{S}$
  for each $\mathbf{x}_r \in \mathbf{X}_{train}$ **do**
    $\mathbf{x}'_r \leftarrow \mathbf{U}^\top (\mathbf{x}_r - \mu)$
  end for
  $\mathbf{X}'_{train} \leftarrow \{(\mathbf{x}'_r, c_r)\}_{r=1}^{R}$
  for each $\mathbf{x}_s \in \mathbf{X}_{test}$ **do**
    $\mathbf{x}'_s \leftarrow \mathbf{U}^\top (\mathbf{x}_s - \mu)$
  end for
  $\mathbf{X}'_{test} \leftarrow \{(\mathbf{x}'_s, c_s)\}_{s=1}^{S}$
  $\text{correct}_l \leftarrow 0$
  for each $(\mathbf{x}'_s, c_s) \in \mathbf{X}'_{test}$ **do**
    $(\mathbf{x}^*, c^*) \leftarrow \underset{(\mathbf{x}'_r, c_r) \in \mathbf{X}'_{train}}{\arg\min} \|\mathbf{x}'_s - \mathbf{x}'_r\|_p$
    if $c_s = c^*$ **then**
      $\text{correct}_l \leftarrow \text{correct}_l + 1$
    end if
  end for
  $\text{correct}_l \leftarrow \frac{\text{correct}_l}{S}$
end for
$l^* \leftarrow \underset{l \in \{1,\ldots,L\}}{\arg\max} \text{correct}_l$

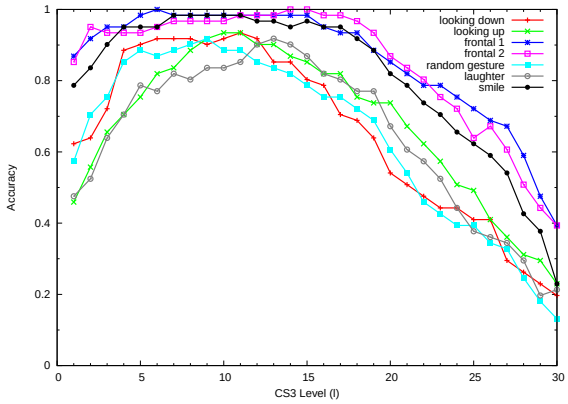


Figure 6.11: Plots of the correct accuracy rates of our recognition system for different classes of test sets as functions of the level in CS3 stack. The same experiments as in Fig. 6.10 were run, except that here the L_2-norm was used for matching, instead of the L_1-norm.

as the ones shown in Fig. 6.10 and Table 6.2, except that the L_2-norm (Euclidean distance) was used by the classifier in the matching stage of the algorithm. As can be seen, the accuracy rate is slightly lowered when the Euclidean distance is used. Our experiments also show that the Mahalanobis distance performs worse than both the L_1 and L_2 norms when used in Eq. (6.19).

In each experiment shown in Table 6.1, the training and test sets contained 366 and 61 meshes (each with 3169 vertices), respectively. The overall time required to run each experiment was approximately 230secs on a 2.0GHz Intel CPU: 110secs to read the meshes in the training and compute the feature vectors, 100secs to solve the resulting eigensystem, and 20secs to read and match all the 61 faces in the test set (approximately 0.33secs to read and match each 3D face).

In the following we compare the performance of our system on GavabDB, against competing methods in the literature. Unfortunately, different methods used different testing procedures when reporting their results. In order to provide a fair comparison, in each case, we use the same testing procedure as the one used by the method against which we are comparing our system.

Mahoor *et al.* [78] and Berretti *et al.* [12] use only one of the frontal scans as the train-

Level	Neutral	Non-neutral	Overall
1	0.7008	0.6120	0.6564
2	0.7663	0.6885	0.7274
3	0.8155	0.7650	0.7903
4	0.8688	0.8360	0.8524
5	0.8934	0.8743	0.8838
6	0.9221	0.8633	0.8927
7	0.9262	0.8961	0.9112
8	0.9385	0.8961	0.9173
9	0.9426	*0.9126**	0.9275
10	0.9508	0.9016	0.9262
11	*0.9590**	0.9071	*0.9331**
12	0.9467	0.9071	0.9269
13	0.9303	0.9071	0.9187
14	0.9262	0.8907	0.9084
15	0.9098	0.8743	0.8920
16	0.8852	0.8415	0.8633
17	0.8606	0.8360	0.8483
18	0.8360	0.8032	0.8196
19	0.7991	0.7814	0.7903
20	0.7500	0.6994	0.7247
21	0.7090	0.6448	0.6769
22	0.6721	0.5901	0.6311
23	0.6393	0.5519	0.5956
24	0.6065	0.4972	0.5519
25	0.5655	0.4644	0.5150
26	0.5450	0.4316	0.4883
27	0.4836	0.4043	0.4439
28	0.4180	0.3224	0.3702
29	0.3606	0.2513	0.3060
30	0.3032	0.1912	0.2472

Table 6.3: The accuracy rates of the system for different choices of the CS3 level where the si-LoC values are picked to form the feature vectors; the L_2-norm was used for matching.

Pose	This Work	Mahoor [78]	Berretti [12]
Frontal	95.08%	95.0%	94%
Smile	93.44%	83.6%	85%
Laughter	80.33%	68.9%	81%
Random gesture	78.69%	63.4%	77%
Looking down	88.52%	85.3%	80%
Looking up	85.25%	88.6%	79%
Overall	86.89%	82.83%	84.29%

Table 6.4: Detailed comparison of the performance of our recognition system on GavabDB with competing methods. The training set in all cases consisted of only one scan per subject taken from the set of frontal scans not used in the test sets. Note that in [78], the faces in test sets with expressions were cropped such that only the eyes and nose regions were used.

ing set, while using the remaining scans as test sets. In Table 6.4, we compare our results with theirs. As can be seen, because of the reduction in the number of scans per subject in the training set, the performance of our system has dramatically reduced when compared to our results in Table 6.1. However, the overall recognition rate of our system is still slightly better than the other approaches. Also, note that in [78], the faces in the test set, which contained expressions were cropped such that only the eyes and nose regions were used in matching. This of course, requires one to know, in advance, if an input 3D face contains facial expressions (which is not very practical). Moreover, the two approaches do not report on how the performance of their systems are affected when more samples per subject are provided. Therefore, there is no indication that the performances of their systems improve as the number of scans per subject in the training set is increased. However, we show that the performance of our system improves greatly as more samples are provided.

In most real-world applications, more than one sample per subject is provided in the training set. In fact, the majority of face recognition approaches (*e.g.*, Fisherfaces [10], SVM [90, 92], Bayesian face recognition [85], sparse representation [69, 141]) require more than one sample per subject, in order to estimate information about the distribution of the class associated with each subject in the feature space.

Moreno *et al.* [90] use two types of experiments to evaluate the performance of their PCA and SVM-based 3D face recognition systems. In the "controlled" setting, the test set

Approach	Controlled	Non-controlled
This work	98.36%	96.02%
Moreno (PCA)	82.00%	76.20%
Moreno (SVM)	90.16%	77.90%

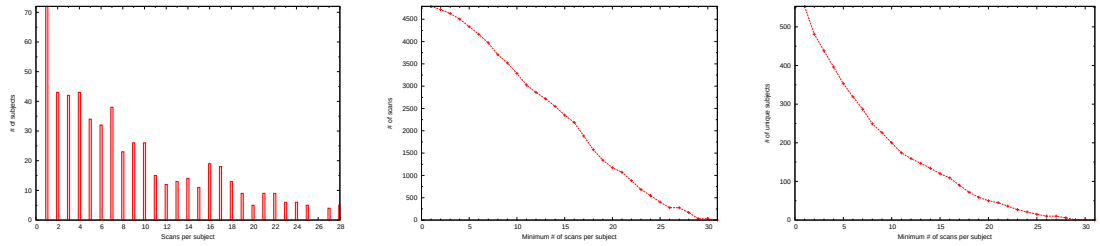
Table 6.5: Comparison of the recognition accuracies of our system with those of Moreno *et al.* [90]

consists of one frontal scan per subject, while the training set consists of the remaining scans in the dataset. Therefore, the sizes of the test and training sets are 61 and 366, respectively. In the "non-controlled" setting, they create the test set by randomly selecting two (out of 7) scans for each subject, and using the remaining scans for the subjects in the training set. As a result, the test and training sets contain 122 and 305 scans, respectively. We use the same procedure to compare the performance of our system with theirs, and show the results in Table 6.5. In the non-controlled setting, we repeated the experiment 7 times and the results in Table 6.5 show the average of the experiments; the best and worst performances were 99.18% and 92.62%, respectively. As can be seen, in all cases, our system outperforms the method of [90].

In [69], the authors test the performance of their 3D face recognition system on GavabDB. However, in their experiments, they extend the size of the dataset by adding 59 additional 3D faces from the FRGC dataset, while omitting the "looking up" and "looking down" scans from the Gavab dataset. The performance of the system was then tested by running 5 different sets of experiments. In each experiment, the test set consisted of 61 scans (1 scan per subject) from the 5 different groups of scans (two sets of frontal scans, 1 set with random gestures, 1 set with laughter and another set with smile), while the training set consisted of the remaining scans in the dataset. The recognition results were then grouped into two classes. The frontal scans formed the "neutral" class, while the other scans (with random gesture, laughter, and smile) formed the "non-neutral" class. The recognition accuracy for each class was then computed as the average of the recognition results of its members. In Table 6.6, we compare the performance of our system with [69].

Approach	Search space:	# scans (Gavab)	# scans (FRGC)	Total # scans	Neutral acc.	Non-neutral acc.	Overall acc.
This work	120 subjs	61 subjs (4 s/s)	59 subjs (6 s/s)	598	97.54%	95.08%	96.07%
This work	614 subjs	61 subjs (4 s/s)	553 subjs (1-30 s/s)	5032	98.36%	92.90%	95.08%
Li [69]	120 subjs	61 subjs (4 s/s)	59 subjs (4 s/s)	480	96.67%	93.33%	94.68%

Table 6.6: Comparison of the correct accuracy rate of our system with the approach of [69]; s/s stands for scans per subject.



(a) (b) (c)

Figure 6.12: (a) Histogram of the number scans per subject for the set of 3D faces from FRGC dataset, which were used in our experiments; (b) dataset size as a function of the minimum number of scans per subject; (c) unique subject count in the dataset as a function of the minimum number of scans per subject.

Note that we conducted two sets of experiments. In the first set of experiments, we followed the same procedure as in [69], but added two additional scans for each subject from the FRGC dataset. This increased the size of the search space by 118 scans from faces *not* in the test set, and hence made the recognition task even more difficult. The first row of Table 6.6 shows the results for this set of experiments. As can be seen, our approach outperforms the approach of [69]. To make the recognition task even more challenging, we added 4788 scans from the FRGC dataset to the training set, while keeping the number of scans from the Gavab dataset the same as before, and followed the same procedure as before to measure the correct recognition rate of our system. The second row of Table 6.6 shows the results of this experiment. As can be seen, the overall correct accuracy rate of our system is still higher than that of [69].

We also tested our recognition system on 4788 3D faces from the FRGC dataset, which corresponded to 553 unique individuals. Unlike GavabDB, in the FRGC dataset, the number of scans for all individuals (subjects) was not the same. In Fig. 6.12(a), we plot the histogram of the number of scans per subject in the dataset, which we used. For example,

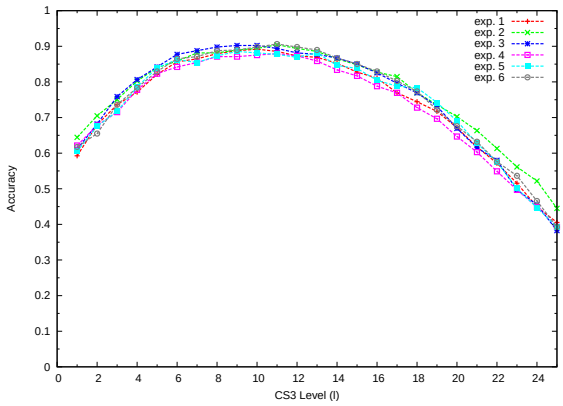
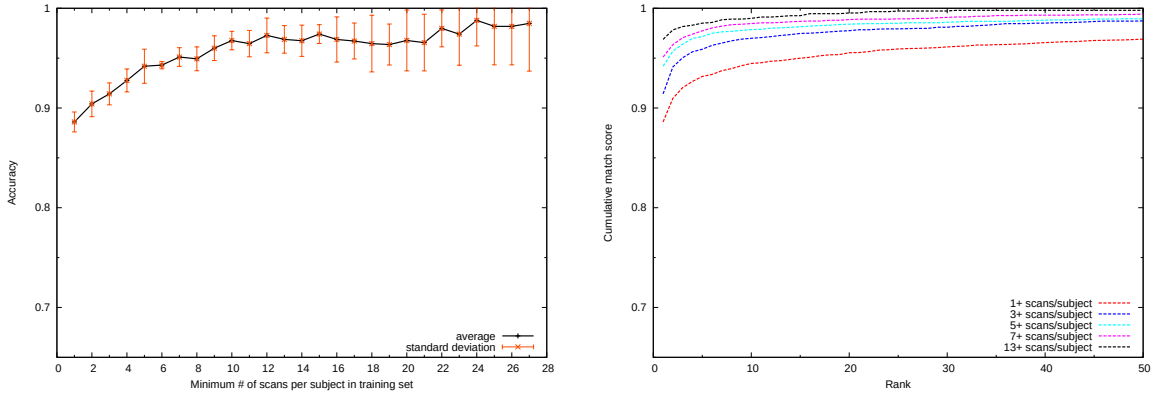
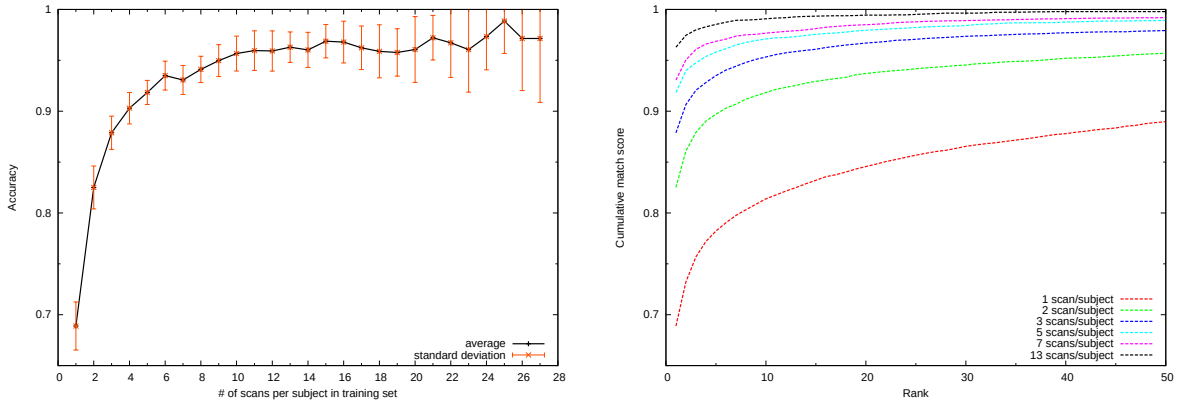


Figure 6.13: Finding the optimal CS3 level for the FRGC dataset: the curves plot the accuracy rate of the system for different number of CS3 levels used to construct the feature vectors.



(a) (b)

Figure 6.14: (a) The average accuracy rate of the system as a function of the minimum number of scans per subject in the training set; the red bars show the standard deviation of the results in the experiments. (b) Cumulative Match Characteristic curves for different minimum numbers of scans per subject.



(a) (b)

Figure 6.15: (a) Accuracy rate of the system as a function of the number of scans per subject in the training set; (b) Cumulative Match Characteristic curves for different numbers of scans per subject.

72 subjects had only 1 scan, and 43 subjects had 2 scans. In Fig. 6.12(b), we show how the size of the dataset decreases as we increase the required minimum number of scans per subject. For example, the total number of scans in the dataset decreases to 4716, when only subjects with at least two scans are considered, while the dataset size becomes 3286, when only subjects with at least 10 scans are kept. In Fig. 6.12(c), we show how the number of unique subjects in the dataset decreases as the required minimum number of scans per subject is increased. For example, the number of unique subjects decreases to 481 and 200, when the minimum required number of scans are set to 2 and 10, respectively. As is shown in the following experiments, the minimum number of scans per subject used in the training set affects the performance of our recognition system, even though our PCA-based system does not explicitly attempt to recover information about the class conditional probability density function of each face class in the feature space.

In Fig. 6.13, we show how the accuracy of the system for the FRGC dataset is influenced by the choice of the CS3 level used when constructing the feature vectors. As can be seen, again the optimal performance is achieved approximately at level $l = 10$ (with average accuracy rate of 89.05%). Therefore, throughout all our experiments, we used 10 CS3 levels to construct the required feature vectors. In the 6 experiments conducted to obtain

the plots in Fig. 6.13, we used a subset of the scans in the dataset, which contained at least two scans per subject. This enabled us to partition the dataset into two disjoint sets to obtain the training and test sets. In each experiment, the test set was constructed by randomly selecting one scan for each individual in the dataset, and the remaining scans were used as the training set.

We argue that the decreased accuracy of the system for the FRGC dataset (compared to GavabDB) is due to the large number of subjects in the training set with small number of scans, and that the increased size of the search space has a smaller influence on the performance of the system. All (100%) of the subjects in the GavabDB experiments had 6 scans in the training set, whereas in FRGC only 287 out of 553 (51.9%) of the subjects had at least that many scans. In Fig. 6.14(a), we show how the performance of the system is improves as the minimum number of scans per subject is increased. The graph plots the average and standard deviation of the accuracy rate of the system for a set of 11 experiments where disjoint training and test sets were randomly constructed. As can be seen, the accuracy of the system increases as the minimum required number of scans per subject is increased. The average accuracy rate of the system on a subset of the FRGC dataset where each subject has at least 6 scans in the training set is 94.30%. The training set for this subset contains 287 unique subjects, and a total of 3683 scans. The performance of system is further increased to 96.00% when all subjects in the training set have at least 9 scans (200 unique subjects, and a total of 3086 scans in the training set). Fig. 6.14(b) shows the averaged Cumulative Match Characteristic (CMC) curves of our system for different minimum number of scans per subject in the dataset.

In Fig. 6.15, we plot the average accuracy rates and CMC curves of our system for a set of 35 experiments. However, for these experiments, a *fixed* number of scans per subject was used in each experiment—instead of a minimum number of scans; *e.g.*, the red curve in Fig. 6.15(b), plots the average CMC curve of 35 experiments, where in each experiment, the training set was constructed by randomly selecting *only one* scan from each subject in

the FRGC dataset and using the remaining scans in the test set. As the graphs show, the performance the system improves dramatically as the number of scans per subject increases in the training set. For example, increasing the number of scans for each subject in the training set from 1 to 3, improves the accuracy rate of the system by approximately 20%.

6.3 Summary

In this chapter, we presented an application of our proposed CS3 representation for 3D surfaces in a partial 3D shape matching task involving detection of 3D faces on input surfaces with arbitrary scale, orientation, and translation. The output of our detection system could directly be fed into a 3D face recognition system. We showed other applications involving automatic processing of 3D faces, such as the generation of hybrid faces. Moreover, we presented a new face recognition system, which employed our proposed scale-invariant Laplacian of surface curvatures to form the feature vectors used for recognition. We tested the performance of the recognition system on two well-known 3D face datasets, and showed its superior performance over state-of-the-art 3D face recognition systems.

Chapter 7

2D Warping for Retargeting Garment Poses

"Art is significant deformity."
–Roger Fry

7.1 Introduction

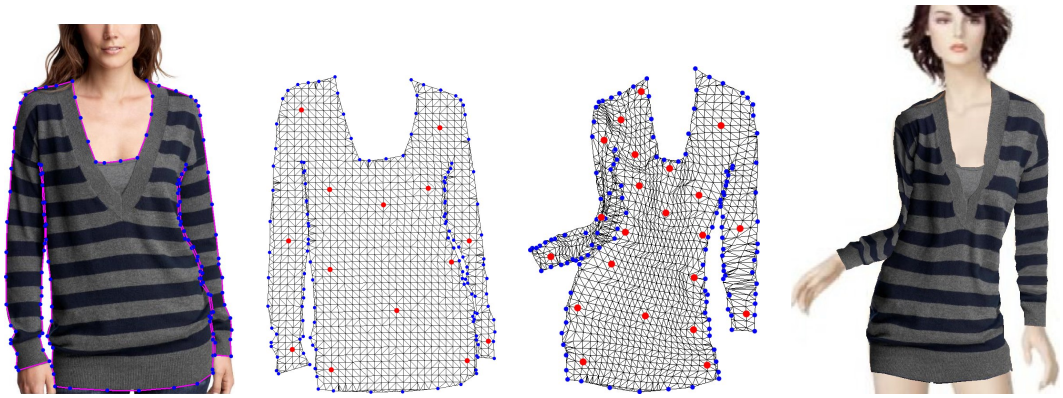
The online experience in shopping for garments has remained largely unchanged over the years. Consumers are typically presented with a myriad of clothing images, with each image displayed on a mannequin or flat surface. The ability to drag and drop clothing onto a single mannequin to configure and visualize outfits is considered to be highly useful. A few companies, such as *looklet*® [76] and *schway*® [110], already help users design custom looks by allowing them to virtually dress model mannequins with clothing items available on their websites. To facilitate this capability, all clothing items must either be placed on the mannequins when obtaining their pictures, or be deformed from different poses to fit the target mannequins. The former approach is generally more expensive and involves more manual labor than the latter. Additionally, the second approach allows showcasing any

clothing item, which may have been previously photographed for other purposes. Therefore, having a system which allows designers to manipulate images of clothing items to fit a target shape is desirable. Retargeting of images of garments among arbitrary poses, however, is a challenging problem, since the images come from disparate sources, including mannequins and human models in many poses, sizes, and shapes.

This chapter describes a deformation system to warp images of garments from any pose onto a target mannequin upon which a fashion ensemble can be created by the user. We shall treat this problem as an exercise in 2D shape deformation that is governed by user-specified control points. Various image deformation techniques exist in the graphics literature. However, to the best of our knowledge, none provide a general framework capable of handling the challenges that arise when designing a system for this problem. The main difficulties faced by the system are allowing the user to perform the deformations with as few operations as possible, while providing fine control over the shapes of the deformed images. Additionally, allowing the user to assign depth values to different parts of the items is another crucial feature needed in such a deformation system.

We build upon recent advances in Moving Least Squares (MLS) and As-Rigid-As-Possible (ARAP) shape manipulation [4, 46, 119, 107]. We extend MLS from image-space domain to the object-space domain to handle the concave shapes that are prevalent in this application. We also exploit ARAP deformations to yield more natural warps that improve the MLS results.

Fig. 7.1 shows the deformation pipeline of our system. We treat the garment to be a textured triangulated 2D mesh that is derived from the input image. If the image contains an alpha channel, the outer contour of the largest connected component of non-zero alpha values is taken to be the garment contour that defines the mesh boundary. Otherwise, the user must manually trace a closed polyline in the image to define the garment contour. The closed region defined by the contour is then triangulated to form a mesh. This is achieved by applying a constrained conforming Delaunay triangulation [114] to the boundary points



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

Figure 7.1: Deformation pipeline: (a) the user traces the contour of the garment to define a closed region; (b) the triangulated mesh of the region with user-specified interior (red) and exterior (blue) control points; (c) the user deforms the mesh to align it with a target mannequin by inserting, deleting, and moving the control points; (d) the final result of the garment retargeted onto the desired mannequin.

and a uniform sampling of the interior. The resulting triangles are constrained in terms of their minimum angle to prevent undesirable thin triangles. The quality of the triangulation is further improved by smoothing the resulting mesh using Laplacian smoothing [28].

The deformation system operates in two modes: *coarse* and *fine*. The coarse mode makes use of interior control points to adjust the overall shape of the object's mesh, while the fine mode uses exterior control points to enable the user to make finer adjustments to the mesh boundary. The interior control points may be placed by the user anywhere inside the mesh. The exterior control points consist of user-drawn polyline vertices or critical points that lie on the contour extracted from the alpha channel of the garment image. Both sets of control points are moved to deform the mesh. When operating in coarse mode, the exterior control points are treated as an additional set of points that moves alongside the deformed object. When operating in fine deformation mode, the converse happens. That is, the scattered set of interior control points are treated to be a part of the object and move as the boundaries of the shape are deformed.

The contributions of this work include the extension of the image-space MLS deformation approach of [107] to object-space. This enables our system to correctly deform

objects with concave boundaries and overlapping regions. Additionally, we show that even though the resulting deformation is not "as-rigid-as-possible" in the sense of [4, 119], it can be used as a good initial estimate to an iterative scheme that improves the rigidity of the deformation at interactive rates.

Another contribution of this work includes a more direct derivation for the rigid MLS formulation than originally given in [107]. By using Lagrange multipliers, we avoid the need to derive a solution for the similarity MLS deformation prior to solving for the rotation matrix in the rigid deformation.

An additional feature of our 2D deformation system is the capability of assigning arbitrary depth values to the control points, which can then be used to determine the depth values at all mesh vertices. This feature is required when rendering deformed objects with overlapping parts. To interpolate these depth values across the mesh requires us to solve a scattered data interpolation problem. We again employ Moving Least Squares for this purpose and efficiently incorporate the geodesic distance as the default metric in the computations to accommodate arbitrary shapes.

Finally, we show how the interior and exterior control points enable the system to furnish intuitive local-global control over the shape of the deformed mesh. As mentioned earlier, exterior points accommodate the fine local detail along the boundary, while interior points perform coarser global shape adjustments.

The proposed deformation system was designed to be easy to use and operate at interactive rates. The behavior of the system adheres to the following criteria:

- *Ease of use:* an iterative ARAP procedure is used to require minimal user input in creating natural deformations.

- *Fast response time:* our object-space MLS deformation algorithm furnishes a good initial estimate for the ARAP procedure so that only a few iterations are required, thereby achieving deformations at interactive rates.

- *Fine control:* exterior control points provide the user with fine control over the deformations.
- *Global control:* interior control points provide the user with global control over the deformations.
- *Scalability:* the linear complexity and parallelism of ARAP facilitates efficient handling of large meshes.

7.2 Related Work

Various 2D deformation approaches exist in the graphics literature. These approaches can be categorized into two classes:

- *Image-space techniques:* space warping techniques in which all points in the 2D Euclidean space are affected by the deformation. Historically, this class of techniques comes from the image morphing literature [18, 138, 64, 139].
- *Object-space techniques:* the domain of the warp function does not extend to the whole 2D Euclidean space. Instead, the warp function is defined over a closed region in the space, where an object resides. Additionally, a metric other than the Euclidean distance is generally used in the deformation to cope with the general, non-convex shape of the object. These techniques have their roots in the 3D modeling literature [4, 46, 119].

The origins of image-space 2D deformations can be traced back to the early days of image morphing. Image morphing refers to the fluid transformation between two images. The development of morphing techniques gained widespread attention mainly due to their applications in the film industry. The most important element in image morphing is the warp function, which maps the 2D positions in the image domain into new ones. The process of performing this mapping on the image is referred to as *image warping*. The

applications of image warping techniques, however, are not limited to image morphing. For example, one of the early uses of warping techniques was to correct the effects of lens distortion in images. The main disadvantage of an image-space deformation technique is that it does not respect the boundaries of the objects in the scene; deforming one object may influence other objects in the scene. Additionally, these techniques are unable to handle the cases, where parts of the deformed objects overlap.

Object-space deformation techniques, on the other hand, are capable of deforming objects with complex shapes, and independently of other objects that may be present in the scene. These techniques may be classified as a generalization/extension of space warping techniques to non-Euclidean or non-convex spaces. The main drawback of these approaches is the need to first define the shape of the object or the deformation space. This is generally done by specifying the object boundary and tessellating the closed region inside. In this work, we are mainly interested in modeling garments which tend to be heavily textured, non-convex, and may develop overlapping elements as a result of the deformation. In such cases, an object-space deformation approach seems like the logical choice.

A large number of image deformation techniques exist in the graphics literature. However, starting with the works of Alexa *et al.* [4], a considerable attention has been given to deformation techniques which are "as-rigid-as-possible" (ARAP). Even though the rigidity requirement for the deformation may not be suitable for all applications, it is argued that ARAP deformations are more "natural" when manipulating physical objects. These techniques have the additional benefit of requiring only few set of handles or control points to guide the deformation.

The original work in [4] is intended for morphing between two (compatibly) triangulated 2D objects by deciding on the most natural paths for the mesh vertices from the source to target shapes. Igarashi *et al.* [46] use the ARAP criterion as the main constraint in designing a deformation system, which enables manipulating 2D objects in real-time. However, they report slow performances for meshes with more than 300 vertices on a 1GHz

system. In [107], Shaffer *et al.* propose an MLS-based image-space deformation system which is capable to deforming images with thousands of pixels in real-time. They derive closed-form solutions for affine, similarity and rigid control points and lines, which are used to guide the deformation.

In [119], Sorkine *et al.* propose an ARAP system for deforming 3D triangular surfaces. Starting with an initial guess for the deformation, their approach iteratively improves the deformation by minimizing the non-rigidity of the deformation in vertex cells over the whole mesh. In [55], the authors propose a more general deformation scheme for deforming images using a similar "local-global" scheme to the one used in [119]. Their method is more general as it allows incorporating information about the image content (energy) in the set of allowable transformations the deformation may employ. They show applications of their system in content-aware image resizing and 2D image deformations.

The main shortcomings of the MLS-based deformation approach of [107] are lack of support for non-convex objects and not being truly "as-rigid-as-possible" as defined in [4]. We show how their MLS approach can be extended to handle non-convex objects. In recent work [55, 135, 136], the shortcomings of the MLS approach compared to other approaches have been solely attributed to it use of Euclidean distance as the driving metric in the formation. However, empirically, we show that, besides the issues arising from the employed metric, the MLS approach enforces a weaker rigidity constraint and that its deformation results can be improved.

7.3 Deformation Framework

The main goal of the shape manipulation system described in this work is to easily deform 2D objects with minimal user interaction. The system is designed to produce intuitive deformations using only a small number of control points, or handles. The deformation is governed by the movement and rotation of these control points. We desire to realize a

deformation that meets user expectations of a "natural" warp. It has been argued in [4, 46, 119, 107, 55] that such deformations should be "as rigid as possible." This facilitates the decoupling of underlying rigid transformations from the more elastic local transformations. First, an initial guess for the most rigid deformation is derived using the changes in control point positions and orientations. The final mesh is obtained by iteratively minimizing the non-rigidity energy of the deformed mesh.

To enforce the rigidity constraint, we follow the local-global approach of recent works by Sorkine *et al.* and Karni *et al.* [119, 55]. Their algorithms alternate between local and global energy minimization stages. The local stage operates at each vertex v and its associated vertex cell, which consists of all triangles incident with v. A small optimization problem is solved at each vertex cell c to derive an unknown rigid transformation between c in the source and deformed mesh. This maintains rigidity at each cell. However, no consistency is maintained *between* cells, which leads to conflicting demands on the positions of each vertex. Therefore, a global optimization is needed to evaluate the final positions of the vertices while minimizing the total non-rigidity energy of the deformation. The global stage minimizes an error functional by solving a large sparse linear system of equations, which yields new positions for all vertices on the updated mesh.

The above procedure may be iterated a number of times until convergence is reached. A good initial guess for the rigid deformation is important because it reduces the number of required iterations, thereby facilitating deformations at interactive rates, and decreases the possibility of converging to a local minimum. We obtain the initial guess for the deformation by using a modification of the Moving Least Squares approach in [107]. We address a key shortcoming of that work in which an image-space deformation technique was limited to deforming convex objects. Our simple modification, which modifies the metric used in the deformation, is capable of handling objects with arbitrary geometry. We refer to this new approach as the *object-space MLS*. Our iterative local-global optimization stage is motivated by the 3D deformation approach of [119]. However, we achieve computational

savings by exploiting a closed-form result for our 2D case that was originally presented in [107].

Additionally, although the overall shape of the deformed object may look correct, one generally needs to make finer adjustments to the contour of the shape. To enable this task, our system allows the user to manipulate a set of additional control points which are automatically placed near the boundaries of the shape.

In Sec. 7.3.1, we describe our object-space MLS deformation scheme, which yields an initial guess for the deformation. In Sec. 7.3.2, we describe the ARAP procedure, which improves the initial deformation. In Sec. 7.3.3, describe the user may rotate parts of the mesh by assigning rotation angles to each control point. To handle overlapping parts in the deformed mesh, we enable the user to assign depth values to each interior control point on the mesh. In Sec. 7.3.4, we describe how the assigned depth values may be interpolated across the whole mesh. Finally, in Sec. 7.3.5, we describe the process used to automatically extract exterior control points and show how to they can be used to make fine adjustments to the shape of the mesh boundary.

7.3.1 The Initial Deformation

The input to our deformation system is a 2D object represented by triangular mesh $M = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_n\}_{n=1}^{N}$ is the set of mesh vertices, and $\mathcal{E} = \{e_{ij} \in \mathcal{V} \times \mathcal{V} | v_i \text{ is connected to } v_j\}$ is the set of edges that connect the vertices. We will assume that the topology of the shape remains the same during the deformation and only the geometry is changed. Let ordered sets $P = \{p_n\}_{n=1}^{N}$ and $Q = \{q_n\}_{n=1}^{N}$ contain the 2D positions of the vertices in the original and deformed meshes, respectively. Similarly, let ordered sets $A = \{a_m\}_{m=1}^{M}$ and $B = \{b_m\}_{m=1}^{M}$ respectively contain the source and target positions of the interior control points. The goal is to determine a warp function $\psi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that yields a natural deformation based on the movements of the control points from a_m to b_m. In order to achieve a desirable deformation, ψ must satisfy the following properties [107]:

- *Smoothness:* The warp function must be smooth to guarantee that the deformation process does not introduce discontinuities in the geometry or texture of the deformed object; *i.e.*, $\psi \in C^k$, for $k \geq 1$.

- *Interpolation:* The warp function must be interpolating; *i.e.*, $\psi(a_m) = b_m, \forall a_m \in A$. This guarantees that each control point $a_m \in A$ is mapped to the user-specified location $b_m \in B$ as a result of the deformation.

- *Identity:* When none of the control points are moved, the function must leave all the points in the original mesh unchanged; *i.e.*, if $b_m = a_m,\ \forall a_m \in A$, then $q_n = \psi(p_n) = p_n,\ \forall p_n \in P$.

- *Rigidity:* The warp function should satisfy an "as-rigid-as-possible" property to reduce the burden of the user in specifying the deformation. This means, locally, $\forall x \in \mathbb{R}^2$, the Jacobian of the warp function, $\mathbf{J}(x) = \psi'(x)$, should be a rotation matrix; *i.e.*, $\mathbf{J}(x) \approx \mathbf{R}$, where $\mathbf{R}$ denotes a 2D rotation matrix.

Globally, satisfying the rigidity constraint for all arbitrary configurations of source and target control points is not feasible. Instead, the constraint is enforced on small, local neighborhoods near each point in the domain of ψ. Therefore, we seek to reconstruct the warp function at each mesh vertex based only on the movement of the control points by minimizing an error functional in the least-squares sense. In particular, for each vertex v_n we find a separate local warp function $\psi_n : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that minimizes the error functional

$$E_n = \sum_{m=1}^{M} \theta(d(a_m, v_n)) \|b_m - \psi_n(a_m)\|^2 , \tag{7.1}$$

where $d : \mathbb{R}^2 \times \mathcal{V} \rightarrow \mathbb{R}$ is a function measuring the geodesic distance between a point in $\mathbb{R}^2$ and a mesh vertex, and $\theta : \mathbb{R} \rightarrow \mathbb{R}$ is a non-negative monotonically decreasing weight function. This approach for deforming 2D shapes using Moving Least Squares is based on the work in [107]. One important difference is that the work in [107] was limited to convex

shapes, whereas our use of geodesic distance extends our technique to arbitrary shapes.

The geodesic distance between any two vertices on the mesh is defined as the length of the shortest path that runs along the edges between the two vertices. Let $g : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ be a function that returns the geodesic distance between two mesh vertices. Then, the geodesic distance between a mesh vertex v_n and an arbitrary point $p \in \mathbb{R}^2$ is computed as $d(p, v_n) = \|p' - p\| + g(v', v_n)$, where v' denotes the mesh vertex nearest to p, and p' refers to that vertex position. In our case, p refers to the *source* position of a control point $a_m \in A$. As a result, the geodesic distance only needs to be computed from each control point $a_m \in A$ to all mesh vertices. Note that this computation can be performed offline, as shown in Alg. (2), since it is specified in terms of the anchored source positions, and not the varying target locations. We use Dijkstra's shortest paths algorithm to efficiently compute these distances.

In order for ψ to be local and interpolating at the control points, the following respective conditions must hold: $\theta(r)$ must be rapidly decreasing as $r \rightarrow \infty$ and $\theta(0) = \infty$[66]. These two conditions reduce the set of possible choices for θ to asymptotic functions whose asymptotes coincide with the x and y axes. The hyperbolic cosecant function $\theta(r) = \text{csch}(r/\alpha)$, and the rational function $\theta(r) = r^{-\alpha}$, for some $\alpha > 0$, are two examples of good candidates for the weight function. In our experiments, we noticed that weight functions with a faster rate of decay result in more desirable (*i.e.*, rigid) deformations. For any $\alpha_1, \alpha_2 > 0$, $\text{csch}(r/\alpha_1)$ decreases more rapidly than $r^{-\alpha_2}$ as $r \rightarrow \infty$. This, theoretically, makes $\text{csch}(r/\alpha_1)$ a more suitable choice as the weight function for the deformation. However, the fast rate of decay of csch introduces numerical instabilities. Therefore, $\theta(r) = r^{-\alpha}$ is used as the weight function in all calculations that follow. The same choice of weights was also used in the original MLS formation of [82] and MLS-based deformation scheme of [107].

Since we are looking for a rigid deformation, ψ_n is restricted to belong to the class of rigid transformations. Therefore, the warp function at each vertex v_n is given as

$$\psi_n(x) = R_n x + T_n \ , \tag{7.2}$$

where R_n is a rotation matrix, and T_n is a vector representing translation. Consequently, the error functional of Eq. (7.1) becomes

$$E_n = \sum_{m=1}^{M} \theta_m^n \|b_m - R_n a_m - T_n\|^2 \ , \tag{7.3}$$

where $\theta_m^n = d(a_m, v_n)^{-\alpha}$ for some $\alpha > 0$. It is important to note that the MLS formulation in Eq. (7.3) attempts to derive a rigid transformation based exclusively on the source and target positions of the control points. This is considered to impose a weaker rigidity constraint because it does not explicitly incorporate the local neighborhood about each mesh vertex v_n.

It is shown in Appendix A that the T_n minimizing Eq. (7.3) is given as

$$T_n = \bar{b} - R_n \bar{a} \ , \tag{7.4}$$

where

$$\bar{a} = \frac{\sum_{m=1}^{M} \theta_m^n a_m}{\sum_{m=1}^{M} \theta_m^n} \ , \qquad \bar{b} = \frac{\sum_{m=1}^{M} \theta_m^n b_m}{\sum_{m=1}^{M} \theta_m^n} \ , \tag{7.5}$$

denote the weighted centroids of the original and new positions of the control points, respectively. Substituting the recovered T_n back into Eq. (7.3), we obtain

$$E_n = \sum_{m=1}^{M} \theta_m^n \|\hat{b}_m - R_n \hat{a}_m\|^2 \ , \tag{7.6}$$

where $\hat{b}_m = b_m - \bar{b}$ and $\hat{a}_m = a_m - \bar{a}$. The R_n minimizing Eq. (7.6) is given as

$$R_n = \frac{1}{\mu} \sum_m \theta_m^n \begin{pmatrix} \hat{b}_m & -\hat{b}_m^\perp \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} , \tag{7.7}$$

where $\mu = \sqrt{\left(\sum_m \theta_m^n \hat{b}_m^\top \hat{a}_m\right)^2 + \left(\sum_m \theta_m^n \hat{b}_m^\top \hat{a}_m^\perp\right)^2}$. $\perp$ is an operator, which rotates 2D vectors by 90 degrees counterclockwise:

$$v^\perp = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}^\perp = \begin{pmatrix} -v_2 \\ v_1 \end{pmatrix} . \tag{7.8}$$

See Appendix A for the full derivation of R_n.

Therefore, the warp function at vertex v_n is given by

$$\begin{aligned} \psi_n(x) &= R_n x + T_n \\ &= R_n x + (\bar{b} - R_n \bar{a}) \\ &= \tfrac{1}{\mu} \textstyle\sum_m \theta_m^n \begin{pmatrix} \hat{b}_m & -\hat{b}_m^\perp \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} (x - \bar{a}) + \bar{b} \, . \end{aligned} \tag{7.9}$$

However, for each warp function ψ_n we are only interested in evaluating it at p_n, which corresponds to the position of mesh vertex v_n. Therefore, $\psi_n(p_n)$ becomes

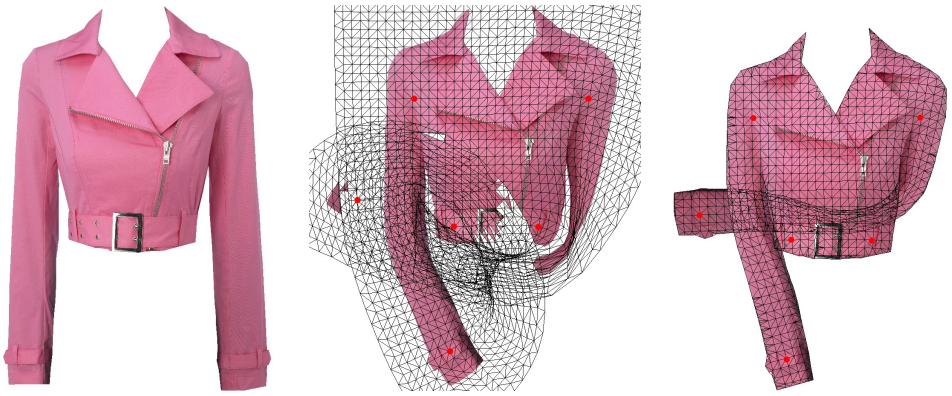
$$\psi_n(p_n) = \frac{1}{\mu} \sum_m A_m^n \hat{b}_m + \bar{b} \, , \tag{7.10}$$

where

$$A_m^n = \theta_m^n \begin{pmatrix} (p_n - \bar{a}) & -(p_n - \bar{a})^\perp \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} . \tag{7.11}$$

See Appendix B for the full derivation of Eq. (7.10).

Since each matrix A_m^n only depends on mesh vertex position p_n and the source positions of all the control points, A_m^n may be precomputed. On the other hand, μ, cannot be precomputed as it depends on b_m. The computation of μ, however, can be circumvented by



(a) original image (b) image-space MLS (c) object-space MLS

Figure 7.2: Comparison of image-space and object-space MLS deformations. The object-space deformation results are preferred since the movement of control points on one sleeve does not induce extreme foldover.

exploiting the fact that rotation preserves lengths, as suggested in [107]. Indeed, the vector $p_n - \bar{a}$ is rotated by R_n in Eq. (7.10). Therefore, its rotated length must be preserved:

$$\|p_n - \bar{a}\| = \frac{1}{\mu}\|\sum_m \theta_m^n \begin{pmatrix} \hat{b}_m & -\hat{b}_m^\perp \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} (p_n - \bar{a})\| = \frac{1}{\mu}\|\sum_m A_m^n \hat{b}_m\| \,. \quad (7.12)$$

From this we obtain $\mu = \frac{\|\sum_m A_m^n \hat{b}_m\|}{\|p_n - \bar{a}\|}$. Therefore, $\psi_n(p_n)$ can be computed more efficiently as

$$\psi_n(p_n) = \|p_n - \bar{a}\| \frac{\sum_m A_m^n \hat{b}_m}{\|\sum_m A_m^n \hat{b}_m\|} + \bar{b} \,. \quad (7.13)$$

Note that both A_m^n and $\|p_n - \bar{a}\|$ can be precomputed, as shown in Alg. (2). The new positions of the vertices in the deformed mesh are given as $q_n = \psi_n(p_n)$.

Fig. 7.2 compares the results for image-space and object-space MLS deformations. Notice that the object-space MLS deformation results are superior to the image-space results because the movement of control points in one part of the shape does not affect unrelated mesh vertices that may lie nearby. For instance, the movement of control points on the sleeve only affects the sleeve region and does not affect the position of mesh vertices on the nearby torso.

7.3.2 As-Rigid-As-Possible Deformation

As noted earlier, the object-space MLS deformation procedure described in Sec. 7.3.1 is not really "as rigid as possible" in the sense of [4, 119]. The downfall of the approach is due to the fact that a different rotation matrix is obtained for each vertex in the input mesh and no attempt is made to ensure the matrices are locally the same. However, the approach results in a good approximation and can be used as an initial guess for an iterative ARAP procedure that refines the deformation. In this section, we adapt the ARAP 3D deformation method of [119] to iteratively improve the 2D results of Sec. 7.3.1. Again, to achieve deformations at interactive rates, special care is taken to ensure that computations that can be done offline are decoupled from those needed online.

The ARAP procedure iteratively minimizes the total non-rigidity energy of our initial MLS deformation. This energy is defined as the sum of the local non-rigidity energies in the vertex cells covering the whole mesh. Each vertex cell C_n is defined using vertex n and its 1-ring neighbors, given by set $\mathcal{N}(n) = \{k|e_{nk} \in \mathcal{E}\}$.

Each iteration of the minimization in the ARAP procedure consists of two steps. In the first step, for each vertex cell C_n in the original mesh, we solve for a rotation matrix that best maps, in a least-squares sense, the positions of the vertices in C_n onto their corresponding positions in the deformed mesh. The sum of squared differences in these positions defines the local non-rigidity in the cell. In the second step, the recovered rotation matrices are used to construct and solve a sparse linear system to recover vertex positions that minimize the total non-rigidity energy of the deformation over the whole mesh. The two steps can be iterated to improve the deformation results. It has been shown that this iterative algorithm is guaranteed to converge to a local minimum [55]. In the following sections, we describe each step of the ARAP procedure in more detail.

Step 1: Recovery of Local Non-Rigidity Energy

Let $P = \{p_n\}_{n=1}^N$, $Q = \{q_n\}_{n=1}^N$ denote the positions of the mesh vertices before and after the initial MLS deformation. If the deformation is locally rigid in cell C_n, as shown in top row of Fig. 7.3 for $n = 0$, then a rotation matrix R_n exists such that

$$\hat{q}_k = R_n \hat{p}_k \ , \qquad \forall k \in \mathcal{N}(n) \ , \tag{7.14}$$

where $\hat{p}_k = p_k - p_n$, $\hat{q}_k = q_k - q_n$, and n denotes the central vertex of the cell. If the deformation is not rigid, as shown in the bottom row of Fig. 7.3, a rotation matrix R_n can be found that minimizes the following non-rigidity energy for cell C_n,

$$E_n = \sum_{k \in \mathcal{N}(n)} w_{nk} \|\hat{q}_k - R_n \hat{p}_k\|^2 \ , \tag{7.15}$$

where w_{nk} are edge weights associated with each edge e_{nk}. These weights may be defined in any manner. But in the derivations that follow, it has been assumed that the edge weights are symmetric; *i.e.*, $w_{kn} = w_{nk}$. We shall discuss several possibilities for the edge weights and show that uniform weights work best.

Note that the above minimization problem, which is required in the ARAP procedure, is exactly the same as the one given in Eq. (7.6) that was used for the MLS case. This means that the closed-form solution for the rotation matrix that we derived earlier in Eq. (7.7), can be used for this case. Therefore, the minimizing R_n is similarly given as

$$R_n = \frac{1}{\mu} \sum_{k \in \mathcal{N}(n)} w_{nk} \begin{pmatrix} \hat{q}_k & -\hat{q}_k^\perp \end{pmatrix}^\top \begin{pmatrix} \hat{p}_k & -\hat{p}_k^\perp \end{pmatrix} \ , \tag{7.16}$$

where $\mu = \sqrt{\left(\sum_k w_{nk} \hat{q}_k^\top \hat{p}_k\right)^2 + \left(\sum_k w_{nk} \hat{q}_k^\top \hat{p}_k^\perp\right)^2}$.

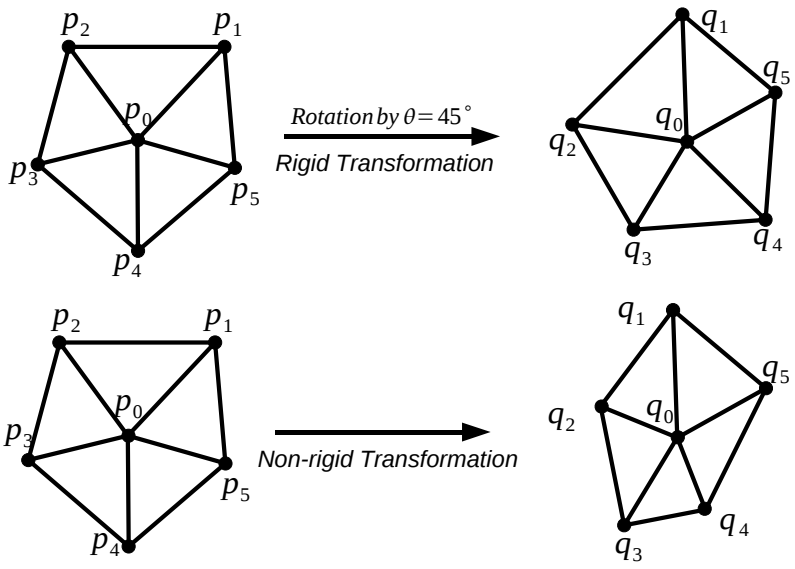


Figure 7.3: Rigid and non-rigid deformations in vertex cell C_0.

Step 2: Improving the Deformation by Minimizing Total Non-Rigidity

In Step 1, Eq. (7.15) defined the non-rigidity of the deformation in each vertex cell. We obtained a best-fit rotation matrix for each vertex cell by minimizing that energy. Since each vertex is subject to multiple rotations due to its membership in several cells, the final position of the vertex is computed by minimizing the total non-rigid energy of the deformation over the whole mesh. This total energy can be defined, using the result in Eq. (7.15), as the weighted sum of the local energies

$$\begin{aligned} E &= \textstyle\sum_{n=1}^{N} w_n E_n \\ &= \textstyle\sum_{n=1}^{N} w_n \sum_{k\in\mathcal{N}(n)} w_{nk} \|(\hat{q}_k - R_n \hat{p}_k\|^2 \\ &= \textstyle\sum_{n=1}^{N} w_n \sum_{k\in\mathcal{N}(n)} w_{nk} \|(q_k - q_n) - R_n(p_k - p_n)\|^2 \,, \end{aligned} \tag{7.17}$$

where w_n are vertex weights, w_{nk} are edge weights, and R_n is given by Eq. (7.16).

Whereas our ARAP procedure began with an initial MLS deformation to warp the image, the subsequent iterations attempt to improve the deformation by further modifying the positions of the mesh vertices. We therefore seek to find these unknown positions q_n, for

all n, by minimizing E. The partial derivative of E with respect to q_n is given as

$$\begin{aligned}
\frac{\partial E}{\partial q_n} &= \frac{\partial}{\partial q_n}\left(\sum_k w_n w_{nk}\|(q_k - q_n) - R_n(p_k - p_n)\|^2\right. \\
&+ \left.\sum_k w_k w_{nk}\|(q_n - q_k) - R_n(p_n - p_k)\|^2\right) \\
&= 2\sum_k w_{nk}\left(-w_n(q_k - q_n) + w_n R_n(p_k - p_n)\right. \\
&+ \left. w_k(q_n - q_k) - w_k R_k(p_n - p_k)\right) \\
&= 2\sum_k w_{nk}\left(-(w_k + w_n)(q_k - q_n)\right. \\
&+ \left.(w_k R_k + w_n R_n)(p_k - p_n)\right) \ .
\end{aligned} \tag{7.18}$$

Note that the first (second) term in the first line of Eq. (7.18) refers to the non-rigidity energy measured across the edges emanating from (towards) q_n. Setting the derivative to zero yields

$$\begin{aligned}
\sum_{k\in\mathcal{N}(n)} w_{nk}(w_k + w_n)(q_k - q_n) &= \\
\sum_{k\in\mathcal{N}(n)} w_{nk}(w_n R_n + w_k R_k)(p_k - p_n) &\ .
\end{aligned} \tag{7.19}$$

The left hand side of the above equation corresponds to computing the weighted discrete Laplacian of the unknown positions at q_n. The corresponding system of equations for the unknown positions can be written in matrix form as

$$\mathbf{L}\mathbf{q} = \mathbf{h} \ , \tag{7.20}$$

where $\mathbf{L}$ is the discrete Laplace operator [83], $\mathbf{q}$ is a vector containing the unknown positions q_n, and vector $\mathbf{h}$ contains the corresponding equalities for each q_n on the right hand size of Eq. (7.19). Each element of $\mathbf{L}$ is given as

$$L_{nk} = \begin{cases} -\sum_{j\in\mathcal{N}(n)} w_{nj}(w_j + w_n), & \text{if } k = n \ , \\ w_{nk}(w_k + w_n), & \text{if } k \in \mathcal{N}(n) \ , \\ 0 \ , & \text{otherwise.} \end{cases} \tag{7.21}$$

Vector $\mathbf{h}$ contains the corresponding equalities for each q_n on the right hand size of Eq. (7.19)

$$h_n = \sum_{k \in \mathcal{N}(n)} w_{nk}(w_n R_n + w_k R_k)(p_k - p_n) \; . \tag{7.22}$$

The system of equations in Eq. (7.20) is sparse and can be solved efficiently using the preconditioned conjugate gradient method. We use the Jacobi preconditioner in this work.

It is important to note that the $N \times N$ matrix $\mathbf{L}$ is only of rank $N - 1$. This implies that the system of equations is underdetermined and yields a solution that is unique up to translation. This ambiguity is easily resolved by fixing the position of one vertex. Fortunately, the control points impose their own constraints by fixing the positions of some deformed mesh vertices: $q_c = b_c$. This serves to guarantee that the control points are mapped to their user-specified target positions, as well as resolving the rank deficiency problem.

Each constraint can be easily added to the system by removing the corresponding row and column from the Laplacian matrix $\mathbf{L}$, removing the corresponding element in vector h, and properly updating the affected elements in $\mathbf{h}$. For example, to add the constraint $q_m = b_m$, the following term is subtracted from the m^{th} element in vector $\mathbf{h}$ for all $k \in \mathcal{N}(m)$:

$$r = w_{mk}(w_m + w_k)b_m \; . \tag{7.23}$$

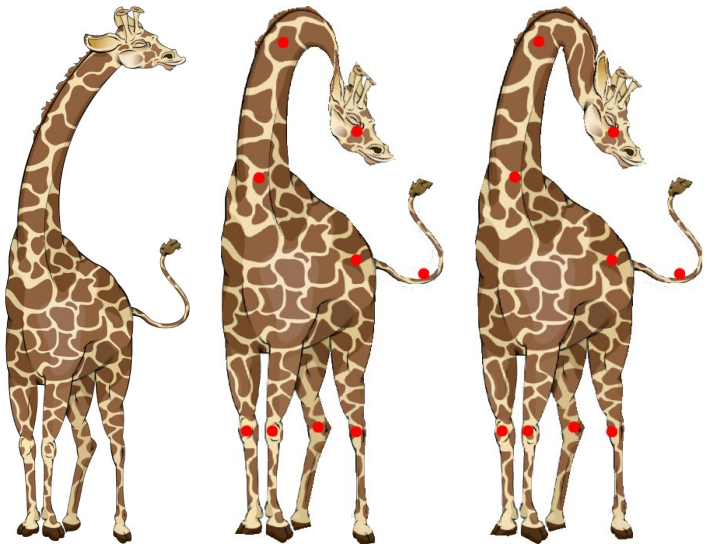
Row and column m is then removed from $\mathbf{L}$ and element m is removed from $\mathbf{h}$. Note that since the points q_n are two dimensional, the system needs to be constructed and solved twice—once for each dimension.

Discussion

Fig. 7.4 and Fig. 7.5 compare MLS and as-rigid-as-possible MLS (ARAP MLS) deformation results. The deformation results corresponding to ARAP MLS are clearly more rigid and desirable, particularly around the sleeve. The giraffe example in Fig. 7.5 is chosen to show the superior performance of ARAP MLS over the object-space MLS under extreme

MLS ARAP MLS

Figure 7.4: Comparison of MLS and ARAP MLS deformation results. Notice that the MLS version induces excessive distortion in the sleeves. This result can be improved, however, by using exterior control points (see Sec. 7.3.5).



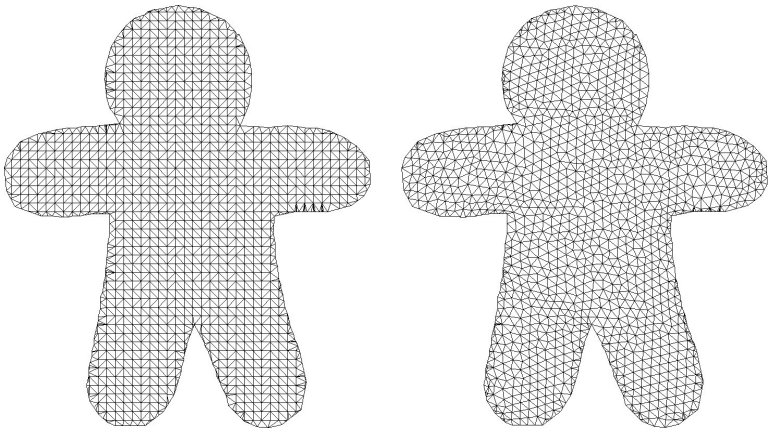
original MLS ARAP MLS

Figure 7.5: Comparison of MLS and ARAP MLS deformation results. Notice that MLS version suffers from excessive distortion along the neck near the head.

deformations. In both Fig. 7.4 and Fig. 7.5, only one iteration of ARAP MLS procedure was applied. Increasing the number of iterations of the ARAP MLS procedure improves the results at the cost of slowing down the deformations. However, increasing the number of iterations does not produce significant improvement since the initial guess was already close to the optimal solution. This a very desirable property of ARAP MLS as it does not require many iterations for the optimizer to converge to the local minimum. This allows the system to perform the deformations at interactive rates.

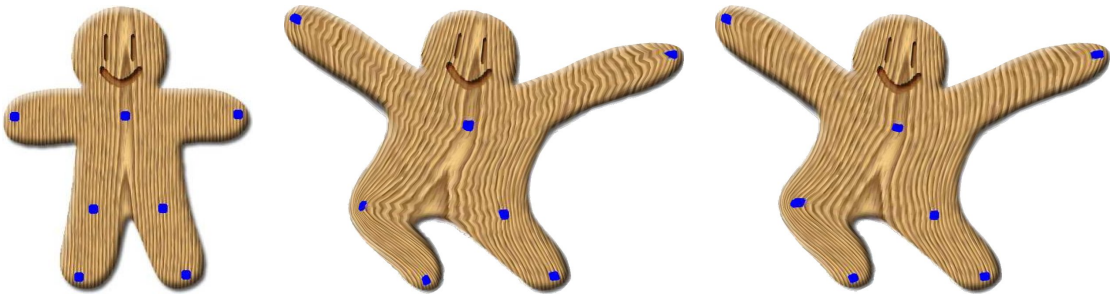
When computing R_n and consequently building the linear system of Eq. (7.20), we use uniform edge weights; *i.e.*, $w_{nk} = 1, \forall e_{nk} \in \mathcal{E}$. In [119], the authors show better results with cotan weights [83] when rigidly deforming 3D objects. However, in our 2D system we observed significantly better results with uniform weights. A comparison of deformations with uniform and cotan weights is shown in Fig. 7.8. When using uniform edge weights, variations in triangle sizes and angles in the original mesh will have an adverse effect on the resulting deformations. To reduce these effects, when creating the mesh, we use Laplacian smoothing [28] to refine the initial triangulation, which is obtained using constrained Delaunay triangulation of the interior of a closed region. This step causes each vertex in the mesh to approximately move to the centroid of its cell. Fig. 7.6 shows the initial and the refined triangulation using Laplacian smoothing. Fig. 7.7 shows the effects of using the two triangulations on the texture of a deformed object. As can be seen, the refined triangulation eliminates the problematic areas in the texture of the deformed object.

Uniform vertex weights w_n were used everywhere, except in the neighborhood around the control points. The mesh vertex closest to a control point and all vertices in its 1-ring neighborhood were given higher weights. This effectively required the 1-ring neighborhood at each control point to be more rigid than the other locations of the mesh. This prevented the local neighborhood around the control points from protruding too much. Additionally, the vertex weights could be set such that they correspond to the magnitude of image gradient at the vertices. However, as expected [55], this does not improve the quality



(a) initial triangulation (b) refined triangulation

Figure 7.6: Initial and refined triangulation of the original mesh used in the deformation.



(a) (b) (c)

Figure 7.7: Comparison of the deformation results using textured meshes in Fig. 7.6: (a) Original; deformation results using (b) non-refined mesh, and (c) refined mesh. Notice that the excessive distortion in (b) is virtually eliminated in (c).

(a) original (b) cotan edge weights (c) uniform edge weights.

Figure 7.8: Deformation results with cotan and uniform edge weights. Notice the undesirable distortion along the sleeve in (b). These artifacts are eliminated by using uniform edge weights.

of the texture in the regions with extreme deformations.

7.3.3 Rotation About Control Points

In some instances, the user may wish to deform the input mesh by rotating sections of the object around a control point. To allow this, we associate a rotation angle to each control point. Rotation about the control points is performed in two steps. First, an initial guess for the deformation is obtained. Second, the rigidity of the deformation is iteratively improved using the same approach as in Sec. 7.3.2.

The initial deformation is obtained by rotating each mesh vertex v_n by the rotation angle associated with its nearest interior control point c_m. Let p_n denote the position of v_n, and R_m be the rotation matrix associated with c_m. The new position q_n of v_n is then obtained as

$$q_n = R_m(p_n - c_m) + c_m \ . \tag{7.24}$$

The initial deformation, however, results in undesirable artifacts in the deformed mesh, as shown in the top of Fig. 7.9. The result can be improved by applying the ARAP procedure as described in the Sec. 7.3.2. In this case, since the initial deformation is poor, more ARAP iterations are required to improve the results. This considerably increases the computation time. However, as seen in the top row of Fig. 7.9, the initial deformation

generally only modifies the positions of a few vertices on the mesh. Therefore, the ARAP deformation procedure only needs to be applied to the vertex cells where the residual of the initial deformation is large. The modified ARAP procedure for rotations about control points is given as follows:

1. Using the initial deformation results, for each vertex cell C_n, obtain rotation matrix R_n as described in Sec. 7.3.2.

2. For each cell C_n, compute the residual

$$r_n = \sum_{k \in \mathcal{N}(n)} w_{nk} \| (q_k - q_n) - R_n (p_k - p_n) \|^2 . \tag{7.25}$$

3. Construct and solve the sparse linear system

$$\mathbf{L}\mathbf{q} = \mathbf{h} , \tag{7.26}$$

   as discussed in Sec. 7.3.2. In order to speed up the computation, add all vertices with residual $r_n < \epsilon$, for some $\epsilon \geq 0$, as constraints to the system.

4. Repeat the above steps K times.

The bottom row of Fig. 7.9 shows the results of the above ARAP procedure applied to the initial deformations results of rotations about control points, shown in the top row of Fig. 7.9.

7.3.4 Handling Overlapping Elements

An important feature of the deformation system is the ability to handle shapes with overlapping elements (Fig. 7.10). We resolve these overlaps by considering each mesh vertex to have a depth value, that is initialized via user-specified control points. This is an exercise in

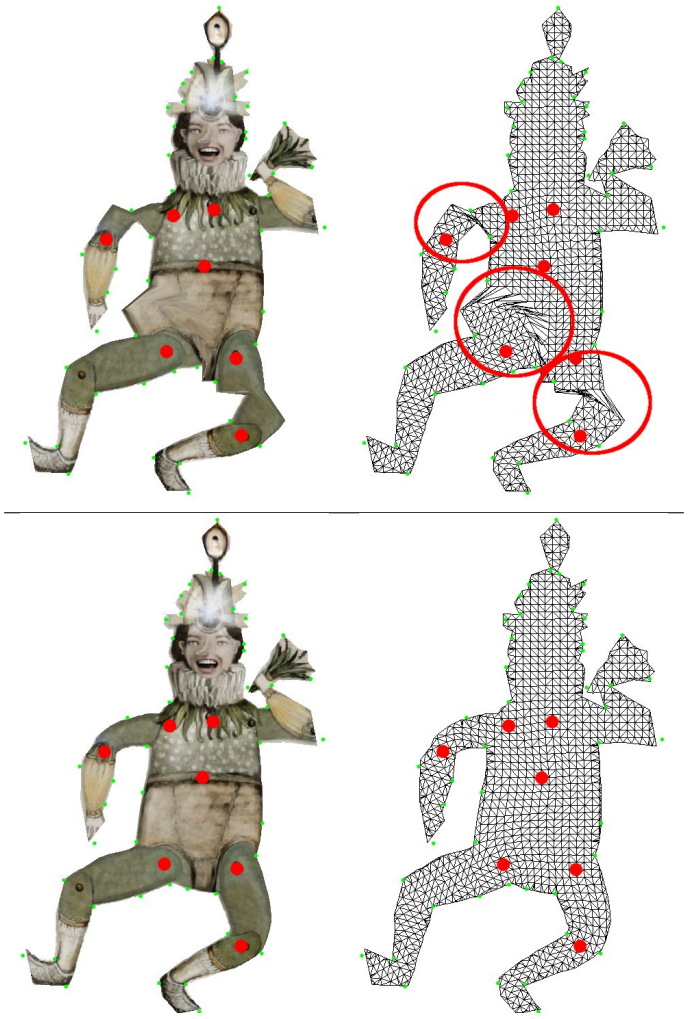


Figure 7.9: Rotation about control points. Top row: initial guess. Bottom row: improved deformation after 6 ARAP iterations.

(a) (b)

Figure 7.10: (a) incorrect and (b) correct handling of overlapping elements.

scattered data interpolation, whereby a smooth depth function is fitted through the control point values.

Let $A = \{a_m\}_{m=1}^{M}$ denote the sparse set of control point positions and let $Z = \{z_m\}_{m=1}^{M}$ denote their associated depth values. Our objective is to find a smooth function $\xi : P \rightarrow \mathbb{R}$ that maps the mesh vertices in P to depth values, with the following properties:

- *Smoothness:* the function must be smooth everywhere, i.e., be at least C^1-continuous.
- *Interpolation:* the depth values at the control points must be the same as the user-specified values; *i.e.*, $\xi(a_m) = z_m$, for $m = 1, ..., M$.
- *Identity:* if all control points have the same depth value, then the mesh should remain flat; *i.e.*, if $z_m = K,\ \forall m \in |Z|$, then $\xi(p_n) = c, \forall p_n \in P$, where c is a constant value.

Note that the desired function has exactly the same properties as the warp function we sought in Sec. 7.3.1. Therefore, we apply the MLS technique to solve for the depth func-

tion. At each mesh vertex v_n we seek to find $\xi_n : \mathbb{R}^2 \rightarrow \mathbb{R}$, minimizing

$$E_n = \sum_{m=1}^{M} \theta(d(a_m, v_n))(z_m - \xi_n(a_m))^2 \,, \tag{7.27}$$

where $\theta(r) = r^{-\alpha}$ for some $\alpha > 0$ and $r \in \mathbb{R}$, and $d(a_m, v_n)$ returns the geodesic distance between mesh vertex v_n and control point a_m, as defined in Sec. 7.3.1. We know the function ξ_n cannot depend on the initial depth values of the control points, since they are all initially placed on the flat mesh and, as a result, have the same values everywhere. Therefore, the local function we are looking for is of the form $\xi_n(x) = c_n$; *i.e.*, a constant function. The error functional becomes

$$E_n = \sum_{m=1}^{M} \theta_m^n (z_m - c_n)^2 \,. \tag{7.28}$$

where $\theta_m^n = d(a_m, v_n)^{-\alpha}$. The minimizing c_n for the energy functional is found by taking the partial derivative of E_n with respect to c_n and setting the result to zero:

$$\frac{\partial E_n}{\partial c_n} = -2 \sum_m \theta_m^n (z_m - c_n) = 0 \,, \tag{7.29}$$

we obtain

$$c_n = \frac{\sum_m \theta_m^n z_m}{\sum_m \theta_m^n} \,. \tag{7.30}$$

Therefore, the depth value of mesh vertex v_n can be efficiently computed as $\xi_n(p) = c_n$ for all $p \in \mathbb{R}^2$. Recomputation is necessary only when the user modifies the height value of a control point. This method enables the user to represent shapes with complex overlapping elements, as shown in Fig. 7.11.

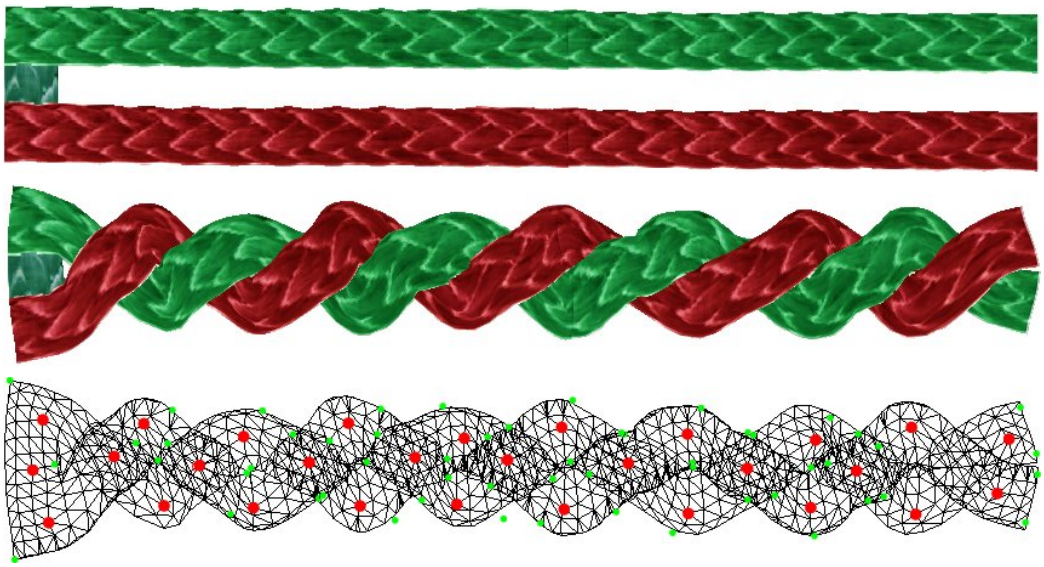


Figure 7.11: Example of complex overlapping elements. Top row: original ropes; Middle and bottom rows: interlaced ropes and their underlying mesh structure with control points shown in red.

7.3.5 Boundary Deformation

As shown earlier, the ARAP MLS approach can be used to deform 2D shapes in a natural manner with few control points. However, the user may still want to further refine the deformation, particularly along the boundaries. This is achieved by enclosing the boundary with a control polyline. The polyline vertices define the exterior control points. By allowing the user to move these points, fine control is given over the shape of the object's boundary. In particular, each line segment in this polyline is an MLS control line. A closed-form solution for deforming the object using these lines is given in [107]. Alternatively, these control lines may be densely sampled to yield MLS control points that will govern the deformation.

In order for the interior and exterior control points to interact with each other in a consistent and natural manner, the system has two deformation modes: coarse and fine. In the coarse (fine) deformation mode, the user can deform the interior (boundary) by moving the interior (exterior) control points. When one set of points are moved, the other set of

points are subject to the deforming forces and follow the shape in tandem.

Since the exterior control points are not part of the triangulated mesh but move with the mesh as it deforms, their coordinates are given relative to the mesh vertices. Let $p \in \mathbb{R}^2$ denote the position of an exterior control point and let $p_m, p_n \in P$ denote the positions of the two closest vertices on the undeformed mesh to p, such that $\|p_m - p_n\| > \epsilon$, for some small $\epsilon > 0$. Then, the relative coordinates of p with respect to the mesh are given in terms of the following inner products:

$$\begin{aligned} p_u &= \langle (p - p_n), \hat{u} \rangle \ , \\ p_v &= \langle (p - p_n), \hat{v} \rangle \ , \end{aligned} \tag{7.31}$$

where $\hat{u} = \frac{p_m - p_n}{\|p_m - p_n\|}$ and $\hat{v} = \hat{u}^{\perp}$ form an orthonormal basis at p_n. The global coordinates of p can be obtained from $\hat{u}$ and $\hat{v}$ by

$$p = p_u \hat{u} + p_v \hat{v} + p_n \ . \tag{7.32}$$

Let p'_m and p'_n denote the new positions of p_m and p_n on the deformed mesh, respectively. The new position of the control point on the deformed mesh is given as

$$p' = p_u \hat{u}' + p_v \hat{v}' + p'_n \ , \tag{7.33}$$

where $\hat{u}' = \frac{p'_m - p'_n}{\|p'_m - p'_n\|}$, $\hat{v}' = \hat{u}'^{\perp}$.

When the deformation mode is changed from coarse to fine, the original mesh is set to be the deformed mesh and the interior control points are subjected to the mesh deformation. The exterior control points define the endpoints of control lines that can be used to adjust the shape of the boundary. Note that as a result of the coarse deformation, the control lines which initially enclosed the original shape may cross into the shape. This, however, does not seem to degrade the deformation results in the fine adjustment phase.

Fig. 7.12 shows an example where a dress is deformed to fit a target model. Note that

the deformation results of ARAP MLS (Fig. 7.12(a)), though close to the desired shape, still require finer adjustments. This is especially evident in the regions near the wrists and shoulders of the model. The shortcomings of the ARAP MLS in those regions are *not* a result of the failure of the technique in enforcing the rigidity constraint; the shape of the underlying mesh near the armpits prevents the arms from correctly rotating around the shoulder joints without distorting the geometry and texture in the surrounding regions. Figures 7.12 (b) and 7.12 (d) show the improved deformation results with the additional help of exterior control points.

Automatic Exterior Control Point Extraction

Manual specification of the exterior control points is undesirable as it may be too tedious for complex shapes. Instead, we would like the system to be able to pick the most appropriate locations for these control points while keeping their numbers at a minimum. In our system, the control points are picked at the locations on the contour curve where the unsigned curvature is locally maximum.

In the continuous setting, the unsigned curvature at a point on an arc-length parametrized regular curve, $S(t), S : [0, 1) \rightarrow \mathbb{R}^2$, is given as

$$\kappa(t) = \| \frac{\partial \hat{T}}{\partial t}(t) \| \, , \tag{7.34}$$

where $\hat{T}(t) = \frac{\partial S}{\partial t}(t)$ is the unit tangent vector at t. The goal is to reliably and quickly estimate the curvature values in the discrete setting, whereby a regularly sampled 2D curve is represented with the ordered point-set $R = \{r_i\}_{i=1}^{I}$, where $r_i \in \mathbb{R}^2$ and I is the number of points on the curve. Our approach is based on the curvature scale-space theory [87], where the derivatives are computed by convolving the sample data with Gaussian derivatives. We obtain the tangent vectors of the sample points by performing circular convolution of the

Figure 7.12: Deforming a dress to match a target shape. Deformation using (a) interior control points only, and (b) interior and exterior control points. Notice that (b) requires exterior control points to perform finer adjustments to the shape boundary. The red and blue dots denote the interior and exterior control points, respectively. (c) and (d) show the original shape and the final deformation results.

data with the first-order Gaussian derivative

$$T = R * G^1_\sigma \,, \tag{7.35}$$

where G^1_σ denotes the discrete first-order Gaussian derivative kernel with standard deviation σ and $*$ is the discrete circular convolution operator. Note that the vectors $t_i \in T$ obtained this way do not necessarily have unit length and their orientations may be incorrect. Therefore, we obtain new normalized tangent vectors $\hat{t}_i$ with correct orientations by

$$\hat{t}_i = s\frac{t_i}{\|t_i\|} \,, \tag{7.36}$$

where $s = \left\langle t_i, r_{(i+1)\%I} - r_i \right\rangle$. The vector-set $\hat{T} = \{\hat{t}_i\}_{i=1}^{I}$ contains the normalized vectors with correct orientations. The normal directions at the data points are then computed by convolving $\hat{T}$ with a first-order Gaussian derivative, in the same manner as before. Let $N = \{n_i\}$ be the vector-set obtained in this manner. The unsigned curvature at data point r_i is given as $\kappa_i = \|n_i\|$.

Sample points whose unsigned curvatures are greater than their immediate neighbors' are then picked as the exterior control points. Note that the robustness of the curvature computations and the number of extracted control points can be adjusted by changing the standard deviation of the Gaussian kernel, σ, used in the derivative computations. We use a default value for $\sigma = \frac{I}{\beta}$, for some $\beta > 0$, which produces acceptable results for all 2D shapes. However, β is a parameter which the user can adjust to alter the number of exterior control points which are automatically extracted. Fig. 7.13 shows an example of the automatically extracted exterior control points on the boundary of a mesh corresponding to a jacket.

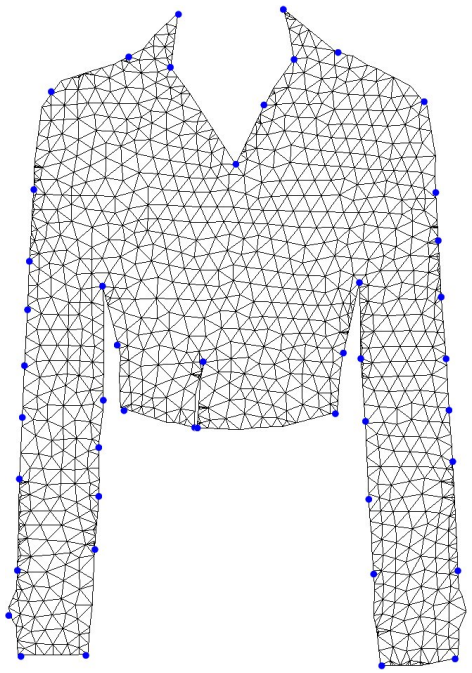


Figure 7.13: Automatically extracted exterior control points on a mesh model of a jacket.

7.4 Implementation

In this section, we briefly discuss the implementation details of the deformation system. The main stages involved are:

- *Input shape specification and triangulation:* an input 2D shape is read and triangulated. The object boundary is either automatically extracted from an image with transparencies or is specified by the user. The exterior control points are also placed on the boundary of the object, at this stage. These exterior control points may, however, be removed or new ones may be added later.

- *Insert/Delete mode:* the user can add or remove interior or exterior control points. Once the user exits this mode, deformation parameters that do not depend on the new target positions of the control points are precomputed to reduce the required computations during the deformation process.

- *Coarse deformation mode:* the user moves the interior control points and the object

is deformed using the ARAP MLS procedure described in the previous sections.

- *Fine deformation mode:* the user makes adjustments to the shape of the boundary of the object by moving the exterior control points.

In the following subsections, we review the steps involved in each of the abovementioned stages of the deformation. In what follows, ordered point-set $P = \{p_n\}_{n=1}^{N}$ represents the original positions of the mesh vertices and $Q = \{q_n\}_{n=1}^{N}$ represents the positions of the vertices in the deformed mesh. $A = \{a_m\}_{m=1}^{M}$ and $B = \{b_m\}_{m=1}^{M}$ represent the source and target positions of the interior control points, respectively. $D = \{d_i\}_{i=1}^{I}$ and $H = \{h_i\}_{i=1}^{I}$ represent the source and target positions of the exterior control points. $S = \{s_m\}_{m=1}^{M}$ represents the depth values of the interior control points, and $T = \{t_n\}_{n=1}^{N}$ represents the depth values of the mesh vertices.

7.4.1 Input Specification and Triangulation

The system allows the user to manipulate 2D shapes which are defined in the form of triangular meshes. The only information needed to generate these meshes is the contour of the shape. Once the contour is defined, the interior of the closed region defined by the contour is triangulated. The user may load an image into the system and manually define the contour of the interest shape in the image using polylines. The system also allows the user to load images with transparencies. In such cases, all the connected components in the image with non-zero alpha values are extracted. The contour of the largest component is used as the input. Given the object's contour, the interior is filled with regularly spaced internal points and a constrained conforming Delaunay triangulation [114] of the interior of the shape is obtained. We use Triangle [113] to compute the triangulation. The triangulation is further improved by applying Laplacian smoothing [28] to the non-boundary vertices of the mesh.

If the shape contour was specified by the user, the locations of the vertices in the contour

polyline are used to define the exterior control points. On the other hand, if the contour was extracted from the alpha channel (transparency) of an image, the locations of the exterior control points are determined automatically as described in Sec. 7.3.5.

7.4.2 Insert/Delete Mode

There are two update modes for adding/removing interior control points and for adding/removing the exterior control points.

Insert/Delete interior control points. Each newly inserted control point is assigned the default depth value of zero. However, these values can be changed by the user. Once the user is done inserting/deleting interior control points, the MLS parameters θ_m^n, l_n, and A_m^n, are computed as per Alg. (2). Additionally, ARAP deformation parameters and vertex depth values are computed as per Alg. (3) and Alg. (4), respectively.

Algorithm 2 Precompute MLS parameters

Input: Mesh vertices $\mathcal{V} = \{v_n\}_{n=1}^N$, the corresponding position set $P = \{p_n\}_{n=1}^N$, and control points $A = \{a_m\}_{m=1}^M$.
Output: MLS parameters θ_m^n, l_n, and A_m^n.

for each control point position $a_m \in A$ **do**
  $v_m \leftarrow$ the nearest vertex to a_m
  $p_m \leftarrow$ position of v_m
  $d_E \leftarrow \|p_m - a_m\|$
  for each vertex $v_n \in \mathcal{V}$ **do**
    $d \leftarrow d_E +$ `GeodesicDistance` (v_m, v_n)
    $\theta_m^n \leftarrow d^{-\alpha}$, for some default $\alpha > 0$
  end for
end for
for each vertex $v_n \in \mathcal{V}$ **do**
  $\bar{a} \leftarrow \frac{\sum_{m=1}^M \theta_m^n a_m}{\sum_{m=1}^M \theta_m^n}$
  $l_n \leftarrow \|p_n - \bar{a}\|$
  for each control point position $a_m \in A$ **do**
    $\hat{a}_m \leftarrow a_m - \bar{a}$
    $A_m^n \leftarrow \theta_m^n \begin{pmatrix} (p_n - \bar{a}) & -(p_n - \bar{a})^\perp \end{pmatrix}^\top \begin{pmatrix} \hat{a}_m & -(\hat{a}_m)^\perp \end{pmatrix}$
  end for
end for

Algorithm 3 Precompute ARAP parameters

Input: Mesh vertices $\mathcal{V} = \{v_n\}_{n=1}^{N}$ and control points $A = \{a_m\}_{m=1}^{M}$.
Output: ARAP parameters w_{ij}, w_n, and $\mathbf{L}$.

for each edge $e_{ij} \in \mathcal{E}$ **do**
  $w_{ij} \leftarrow 1$
end for
for each vertex $v_n \in \mathcal{V}$ **do**
  $w_n \leftarrow 1$
end for
for each control point position $a_m \in A$ **do**
  $v_m \leftarrow$ the nearest mesh vertex to a_m
  $w_m \leftarrow \beta$, for some $\beta \geq 2$
  for each $k \in \mathcal{N}(m)$ **do**
    $w_k \leftarrow \beta/2$
  end for
end for
$\mathbf{L} \leftarrow \mathbf{0}$
for each vertex $v_n \in \mathcal{V}$ **do**
  $\mathbf{L}_{nn} \leftarrow -\sum_{j \in \mathcal{N}(n)} w_{nj}(w_j + w_n)$
  for each $j \in \mathcal{N}(n)$ **do**
    $\mathbf{L}_{nj} \leftarrow w_{nj}(w_j + w_n)$
  end for
end for

Algorithm 4 Compute vertex depth values

Input: Mesh vertices $\mathcal{V} = \{v_n\}_{n=1}^{N}$, control points $A = \{a_m\}_{m=1}^{M}$, depth of A: $Z = \{z_m\}_{m=1}^{M}$, and weights θ_m^n.
Output: Vertex depth values, $Z' = \{z'_n\}_{n=1}^{N}$.

for each vertex $v_n \in \mathcal{V}$ **do**
  $z'_n \leftarrow \frac{\sum_{m=1}^{M} \theta_m^n z_m}{\sum_{m=1}^{M} \theta_m^n}$
end for

Insert/Delete exterior control points. Once the user is done inserting/deleting exterior control points, the following parameters are updated:

- For each exterior control point i, compute its relative coordinates with respect to the mesh using the procedure described in Sec. 7.3.5 and obtain p_u^i, p_v^i.

7.4.3 Coarse Deformation Mode

In the coarse deformation mode, the user selects and drags interior control points to continuously deform the underlying mesh to the desired shape. The deformation is done in two stages. First, we use Moving Least Squares to obtain an initial deformation, as outlined in Alg. (5). Then, the rigidity of the deformation is improved using the as-rigid-as-possible approach described in Sec. 7.3.2 and outlined in Alg. (6).

Algorithm 5 Perform MLS Deformation

Input: Mesh vertices $\mathcal{V} = \{v_n\}_{n=1}^N$, target control points $B = \{b_m\}_{m=1}^M$, and MLS parameters θ_m^n, l_n, A_m^n.

Output: Vertex positions $Q = \{q_n\}_{n=1}^N$ of deformed mesh.

for each vertex $v_n \in \mathcal{V}$ **do**
  $\bar{b}^n \leftarrow \frac{\sum_{m=1}^M \theta_m^n b_m}{\sum_{m=1}^M \theta_m^n}$
  $\hat{b}_m^n \leftarrow b_m - \bar{b}^n$
  $q_n \leftarrow l_n \frac{\sum_{m=1}^M A_m^n \hat{b}_m^n}{\| \sum_{m=1}^M A_m^n \hat{b}_m^n \|} + \bar{b}^n$
end for

Algorithm 6 Perform ARAP Deformation

Input: Mesh vertices $\mathcal{V} = \{v_n\}_{n=1}^N$, vertex positions $P = \{p_n\}_{n=1}^N$ and $Q = \{q_n\}_{n=1}^N$ of input and output meshes, and ARAP parameters w_{ij}, w_n, $\mathbf{L}$.

Output: Refined vertex positions $Q = \{q_n\}_{n=1}^N$.

for each mesh vertex $v_n \in \mathcal{V}$ **do**
  $\mu \leftarrow \sqrt{\left(\sum_k w_{nk} \hat{q}_k^\top \hat{p}_k\right)^2 + \left(\sum_k w_{nk} \hat{q}_k^\top \hat{p}_k^\perp\right)^2}$
  $R_n \leftarrow \frac{1}{\mu} \sum_{k \in \mathcal{N}(n)} w_{nk} \begin{pmatrix} \hat{q}_k & -\hat{q}_k^\perp \end{pmatrix}^\top \begin{pmatrix} \hat{p}_k & -\hat{p}_k^\perp \end{pmatrix}$
end for
Build the sparse linear system $\mathbf{Lq} = \mathbf{h}$ with constraints $q_c = b_c$ as described in Sec. 7.3.2.
Solve the system to obtain the improved vertex positions q_n in the deformed mesh.

7.4.4 Fine Deformation Mode

When the deformation mode is changed from coarse to fine, the state of the deformed mesh is saved by setting $p_n = q_n$ for $n = 1, ..., N$. Each pair of consecutive exterior control points define a control line on the mesh boundary. The required parameters associated with each control line are precomputed, as described in [107]. The user can change the positions and orientations of the control lines by moving the exterior control points. As these lines are moved, the positions q_n of the mesh vertices and the positions of the interior control points are updated as described in [107]. Once the user is done and the deformation mode is changed from fine to coarse, we set $p_n = q_n$ for $n = 1, ..., N$ to save the state of the mesh after the fine deformation mode.

7.5 Results

Fig. 7.14 shows snapshots of our deformation system during various stages of the warping process for retargeting garments among various poses. The top row of Fig. 7.14 depicts images of two dresses that need to be warped to fit different poses. These dresses are shown overlaid upon target models in the second row of the figure. It is clear that major adjustments are needed. The third row of Fig. 7.14 shows the deformation results using only interior control points. Further refinements to the shape boundary are made by adding and moving exterior control points, as shown in the fourth row of the figure. Final results of the deformation process are shown in the bottom row of Fig. 7.14.

The total time required for a user to warp and obtain the final result for each dress in Fig. 7.14 was approximately three minutes. The underlying mesh in each dress consists of approximately 1700 triangular faces, and each ARAP MLS deformation iteration, on average, took 56ms on a single 2Gz Intel® CPU core. In each iteration, the computation time is dominated by the ARAP deformation step, which requires solving a sparse linear system. We use a CPU implementation of the preconditioned conjugate gradient method to solve

the system. However, the timing results can be dramatically improved if a GPU implementation of the conjugate gradient method is used [17, 25]. This performance enhancement is left for future work.

7.6 Summary

In this chapter, we described a deformation system for warping images of garments onto target mannequins of arbitrary poses. The rationale for this work is that input imagery of garments may come from varied sources. However, in order to create a compelling online shopping experience, it is useful for a consumer to drag and drop images of clothing onto a target mannequin to visualize a customizable fashion ensemble. This can only be achieved if the images available to the user have already been warped to be readily aligned on the target mannequin. This warping/alignment problem was treated as an exercise in 2D shape deformation that is governed by user-specified control points. A balance was maintained to allow the user to perform the deformations with as few operations as possible, while, at the same time, providing fine control over the shapes of the deformed images.

We built upon recent advances in MLS and ARAP shape manipulation, and extended MLS from the image-space to the object-space domain to handle the concave shapes that are prevalent in this application. We also exploited ARAP deformations to yield more natural warps that improve the MLS results. In addition, our system enabled the user to correctly deform objects with overlapping elements. Finally, we showed how the interior and exterior control points enable the system to furnish intuitive local-global control over the shape of the deformed mesh.

The work described in this chapter has led to a deformation tool for retargeting images of garments onto mannequins of arbitrary poses. It supercedes commercially available tools such as the Puppet Warp module in Adobe® Photoshop® CS5, which enables deformations using only interior control points. The absence of exterior control points in Puppet Warp

Figure 7.14: Snapshots of our garment retargeting system. Each column of the five rows respectively depicts an input dress, an overlay of the dress on a target model, partial deformation results using interior control points, additional refinement using exterior control points, and the final result.

makes it difficult to perform local shape adjustments, as required for carefully retargeting images of garments. Our use of interior and exterior control points for coarse and fine shape editing modes facilitates a high-speed workflow to handle vast image collections.

In future work, we will attempt to simplify making finer adjustments to the boundary of the deformed object using fewer control points. We will experiment with cage-based deformation techniques that use mean value coordinates [53], harmonic coordinates [52], Green coordinates [74], and complex barycentric coordinates [135]. In future work, we will also incorporate a GPU implementation of the conjugate gradient method to achieve better performance gains.

Chapter 8

Conclusions and Future Work

"What we call the beginning is often the end.
And to make an end is to make a beginning."
–T. S. Eliot

8.1 Summary

In Chapter 2, we provided an overview of *feature-based geometric* 3D surface matching techniques. These techniques are defined as the set of approaches, which encode information about the geometry of (complete or partial) 3D surfaces into numeric representations. The resulting representations are generally stored in the form of high dimensional vectors known as *feature vectors*. We categorized the feature vectors or descriptors employed by these methods into four classes:

- *Global shape properties,* which provide crude description of a surface as a whole.
- *Histograms and shape distributions,* which encode statistical information about surfaces to form the feature vectors.
- *Projections-based methods,* which use frequency analysis tools, such as spherical harmonics, to form the feature vectors.

- *Maps,* which create (mostly local) maps of the positions, normals, curvatures and higher order differential attributes of the surfaces to form the feature vectors.

In Chapter 2, we also briefly reviewed various feature point (keypoint) extraction techniques. These techniques effectively reduce the computational cost of matching algorithms by selecting a small number of points on (or near) the input surfaces. We saw that keypoint extraction techniques are most heavily used in *partial shape matching* tasks, such as automatic surface registration and recognition. These tasks also use *local* shape descriptors at the selected keypoints to establish correspondences between 3D surfaces. We showed that the majority of local descriptors are maps or histograms.

In Chapter 3, we first provided an overview of the scale-space representation and theory for signals in $\mathbb{R}^n$, and then reviewed the extensions of the theory to 3D surfaces. We argued for the use of multiscale surface representations and descriptors for 3D shape matching. In particular, we advocated the extension and use of scale-space theory of signals for 3D surfaces. The motivation was based on the inherent capabilities of the scale-space representation in handling the noise in the data—especially when employing differential operators. Another advantage of the scale-space representation is in its ability to automatically recover information about the size of (differential) structures in the data, which is referred to as "automatic scale selection."

In Chapter 4, we reviewed the Moving Least Squares (MLS) surface reconstruction approach. The MLS approach was used to obtain a smooth representation of the underlying surface described by an unorganized set of 3D points. We showed how the noise in the input is propagated in the various stages of the computation. The error propagation results and statistical hypothesis testing was used to estimate the minimum neighborhood size needed when reconstructing the surface and estimating surface curvatures at each point on the surface. We used the same terminology as in the scale-space theory and referred to the smallest neighborhood size needed to estimate the surface normals or curvatures as the local *scale* of the surface structures. We showed an example where, due to high level of noise in

the input data, the surface curvatures could not be estimated reliably. The main motivation for discussing MLS reconstruction and error propagation was to provide another approach for dealing with the noise in the data. However, we devoted the majority of the thesis to discussing our proposed scale-space representation for 3D surfaces and its applications to various shape matching tasks. One of the advantages of our representation compared to the presented MLS approach was its computational efficiency.

In Chapter 5, we presented our proposed multiscale representation for 3D surfaces, which we referred to as the 3D Curvature Scale Space (CS3). The representation was obtained by Gaussian smoothing the surface curvatures in a manner consistent with the scale-space representation of signals. This enabled us to assign a "scale" to each level in the stack of smoothed surface curvatures, and to define the scale-normalized Laplacian of Curvatures (LoC). We showed that the local extrema of LoC can serve as keypoints, which are stable against noise in the input surface, and are also distributed throughout the surface. The scale normalization enabled us to associate a scale with each extracted keypoint, which in turn, gave an estimate of the neighborhood size associated with each keypoint. We used these results in a crude automatic surface registration algorithm.

In Chapter 6, we employed the proposed CS3 representation for 3D face detection and recognition. The scale selection capabilities of our representation enabled us to perform face detection on models with arbitrary scales. We also demonstrated the discriminatory power of the scale-invariant Laplacian of Curvatures (si-LoC) in a face recognition system, which outperformed the current state-of-the-art 3D face recognition systems.

In Chapter 7, we described a deformation system for warping images of garments onto target mannequins of arbitrary poses. We built upon recent advances in MLS and ARAP shape manipulation, and extended MLS from the image-space to the object-space domain to handle the concave shapes that are prevalent in this application. We also exploited ARAP deformations to yield more natural warps that improve the MLS results. In addition, our system enabled the user to correctly deform objects with overlapping elements. Finally, we

showed how the interior and exterior control points enable the system to furnish intuitive local-global control over the shape of the deformed mesh.

8.2 Future Work

In Chapter 5, we presented a surface registration algorithm, which employed the proposed CS3 representation. However, no thorough error analysis was done on the performance and results of the algorithm. This is left for future work. Additionally, our 3D face recognition algorithm in Chapter 6 did not explicitly make use of the information provided by multiple scans per subject in the training set. In future work, we will augment our recognition system to use more modern recognition algorithms, such as the sparse representation of [23, 141]. Moreover, the face recognition system together with the automatic face warping method, which were discussed in Chapter 6, can easily be used in a system for automatic enhancements of input 3D faces; *e.g.*, see [68]. This is also left for future work.

Appendix A

Recovering MLS Parameters T_n and R_n

Since the error functional is quadratic in T_n, we can find a unique solution for T_n that minimizes E_n. Taking the partial derivative of E_n with respect to T_n and setting the result to zero yields

$$\frac{\partial E_n}{\partial T} = 2 \sum_{m=1}^{M} \theta_m^n (T_n + R_n a_m - b_m) = 0 \; . \tag{A.1}$$

This results in the following expression for T_n:

$$T_n = \bar{b} - R_n \bar{a} \; , \tag{A.2}$$

where

$$\bar{a} = \frac{\sum_{m=1}^{M} \theta_m^n a_m}{\sum_{m=1}^{M} \theta_m^n} \; , \qquad \bar{b} = \frac{\sum_{m=1}^{M} \theta_m^n b_m}{\sum_{m=1}^{M} \theta_m^n} \; , \tag{A.3}$$

denote the weighted centroids of the original and new positions of the control points, respectively. Substituting the recovered T_n back into Eq. (7.3), we obtain

$$E_n = \sum_{m=1}^{M} \theta_m^n \| \hat{b}_m - R_n \hat{a}_m \|^2 \; , \tag{A.4}$$

where $\hat{b}_m = b_m - \bar{b}$ and $\hat{a}_m = a_m - \bar{a}$.

Even though the objective function in Eq. (A.4) is quadratic in R_n, the nonlinear con-

straints $R_n^{-1} = R_n^\top$ and $\det(R_n) = 1$ that guarantee R_n is a rotation matrix make the process of finding R_n more difficult than T_n. Various approaches exist in the photogrammetry literature for finding R_n that minimizes Eq. (A.4) [44, 6], although the majority are formulated for the more complicated 3D case. However, as shown in [107] for the 2D case, a closed-form solution for R_n can be obtained in a manner that allows one to precompute as much information as possible. We adopt this solution to speed up the deformation. We now show how to obtain R_n using Lagrange multipliers in a different derivation than given in [107].

We find R_n by minimizing Eq. (A.4), subject to $\det(R_n) = 1$ and $R_n^\top R_n = I$, where I is the identity matrix. In 2D,

$$R_n = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_1^\perp \end{pmatrix} , \tag{A.5}$$

where θ is a rotation angle and $\perp$ is an operator, which rotates 2D vectors $v = (v_1, v_2)^\top$ by 90 degrees counterclockwise: $[(v_1, v_2)^\top]^\perp = (-v_2, v_1)^\top$. Note that the 2×2 matrix R_n can be defined by a 2D row vector R_1. In addition, the constraints for R_n to be a rotation matrix reduce to $R_1 R_1^\top = 1$. Multiplying R_n with a position vector $a = (a_1, a_2)^\top$ yields

$$\begin{aligned} R_n a &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} a_1\cos\theta - a_2\sin\theta \\ a_1\sin\theta + a_2\cos\theta \end{pmatrix} \\ &= \begin{pmatrix} a_1 & a_2 \\ a_2 & -a_1 \end{pmatrix} \begin{pmatrix} \cos\theta \\ -\sin\theta \end{pmatrix} = \underbrace{\begin{pmatrix} a & -a^\perp \end{pmatrix}}_{2\times 2 \text{ matrix}} R_1^\top . \end{aligned} \tag{A.6}$$

This result allows us to express the objective function as

$$E_n = \sum_m \theta_m^n \| \hat{b}_m - \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} R_1^\top \|^2 , \tag{A.7}$$

subject to the constraint $R_1 R_1^\top = 1$. Using Lagrange multipliers, we include the constraint into the objective function in Eq. (A.7) and obtain

$$E = \lambda(R_1 R_1^\top - 1) + \sum_m \theta_m^n \|\hat{b}_m - \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} R_1^\top\|^2 \,, \tag{A.8}$$

where λ is the introduced unknown Lagrange multiplier. The partial derivative of E with respect to R_1 is given as

$$\begin{aligned}
\tfrac{\partial E}{\partial R_1} &= \tfrac{\partial}{\partial R_1}(\lambda(R_1 R_1^\top - 1) + \textstyle\sum_m \theta_m^n (\hat{b}_m - \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} R_1^\top)^\top (\hat{b}_m - \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} R_1^\top)) \\
&= \tfrac{\partial}{\partial R_1}(\lambda(R_1 R_1^\top - 1) + \textstyle\sum_m \theta_m^n (\hat{b}_m^\top \hat{b}_m + R_1 \underbrace{\begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix}^\top \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix}}_{\hat{a}_m^\top \hat{a}_m I} R_1^\top \\
&\quad -2R_1 \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix}^\top \hat{b}_m)) \\
&= \tfrac{\partial}{\partial R_1}(\lambda(R_1 R_1^\top - 1) + \textstyle\sum_m \theta_m^n (\hat{b}_m^\top \hat{b}_m + \hat{a}_m^\top \hat{a}_m R_1 R_1^\top - 2R_1 \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix}^\top \hat{b}_m)) \\
&= 2(\lambda R_1^\top + \textstyle\sum_m \theta_m^n (\hat{a}_m^\top \hat{a}_m R_1^\top - \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix}^\top \hat{b}_m)) \,.
\end{aligned} \tag{A.9}$$

The following system of equations is obtained by setting the partial derivatives of E with respect to the unknowns (R_1 and λ) to zero:

$$\begin{cases}
\tfrac{\partial E}{\partial R_1} &= 2\left(\lambda R_1^\top + \textstyle\sum_m \theta_m^n \left(\hat{a}_m^\top \hat{a}_m R_1^\top - \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix}^\top \hat{b}_m\right)\right) &= 0 \,, \\
\tfrac{\partial E}{\partial \lambda} &= R_1 R_1^\top - 1 &= 0 \,.
\end{cases} \tag{A.10}$$

The first equation yields

$$R_1 = \frac{1}{\lambda + \sum_m \theta_m^n \hat{a}_m^\top \hat{a}_m} \sum_m \theta_m^n \hat{b}_m^\top \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} \,. \tag{A.11}$$

The second equation enforces the original constraint that R_1 must be of unit length. Since

the length of R_1 is given as

$$\|R_1\| = \frac{\mu}{\lambda + \sum_m \theta_m^n \hat{a}_m^\top \hat{a}_m} , \tag{A.12}$$

where $\mu = \sqrt{\left(\sum_m \theta_m^n \hat{b}_m^\top \hat{a}_m\right)^2 + \left(\sum_m \theta_m^n \hat{b}_m^\top \hat{a}_m^\perp\right)^2}$, we obtain λ by requiring $\|R_1\| = 1$:

$$\lambda = \mu - \sum_m \theta_m^n \hat{a}_m^\top \hat{a}_m . \tag{A.13}$$

Substituting the recovered λ back into Eq. (A.11), yields

$$R_1 = \frac{1}{\mu} \sum_m \theta_m^n \hat{b}_m^\top \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} . \tag{A.14}$$

Finally, since $R_n = \begin{pmatrix} R_1 \\ R_1^\perp \end{pmatrix}$, we obtain

$$\begin{aligned} R_n &= \tfrac{1}{\mu} \textstyle\sum_m \theta_m^n \begin{pmatrix} \hat{b}_m & -\hat{b}_m^\perp \end{pmatrix}^\top \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} \\ &= \tfrac{1}{\mu} \textstyle\sum_m \theta_m^n \begin{pmatrix} \hat{b}_m & -\hat{b}_m^\perp \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^\perp \end{pmatrix} . \end{aligned} \tag{A.15}$$

Appendix B

Evaluation of $\psi_n(p_n)$

For each input position p_n, the deformed output position q_n is given as $q_n = \psi_n(p_n)$, where ψ_n is the warp function associated with the mesh vertex at p_n. We attempt to accelerate the computation of $\psi_n(p_n)$ by separating out the terms that are independent of the target control point positions b_n, and precomputing as many terms as possible. Therefore, $\psi_n(p_n)$

becomes

$$
\begin{aligned}
\psi_n(p_n) &= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} \hat{b}_m & -\hat{b}_m^{\perp} \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^{\perp} \end{pmatrix} (p_n - \bar{a}) + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} \hat{b}_{m,1} & \hat{b}_{m,2} \\ \hat{b}_{m,2} & -\hat{b}_{m,1} \end{pmatrix} \begin{pmatrix} \hat{a}_{m,1} & \hat{a}_{m,2} \\ \hat{a}_{m,2} & -\hat{a}_{m,1} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} \hat{b}_{m,1} & \hat{b}_{m,2} \\ \hat{b}_{m,2} & -\hat{b}_{m,1} \end{pmatrix} \begin{pmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{pmatrix} \begin{pmatrix} \hat{a}_{m,1} \\ \hat{a}_{m,2} \end{pmatrix} + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} u_1 & u_2 \\ u_2 & -u_1 \end{pmatrix} \begin{pmatrix} \hat{b}_{m,1} & \hat{b}_{m,2} \\ -\hat{b}_{m,2} & \hat{b}_{m,1} \end{pmatrix} \begin{pmatrix} \hat{a}_{m,1} \\ \hat{a}_{m,2} \end{pmatrix} + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} u_1 & u_2 \\ u_2 & -u_1 \end{pmatrix} \begin{pmatrix} \hat{a}_{m,1} & \hat{a}_{m,2} \\ \hat{a}_{m,2} & -\hat{a}_{m,1} \end{pmatrix} \begin{pmatrix} \hat{b}_{m,1} \\ \hat{b}_{m,2} \end{pmatrix} + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} u & -u^{\perp} \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^{\perp} \end{pmatrix} \hat{b}_m + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m \theta_m^n \begin{pmatrix} (p_n - \bar{a}) & -(p_n - \bar{a})^{\perp} \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^{\perp} \end{pmatrix} \hat{b}_m + \bar{b} \\
&= \tfrac{1}{\mu}\textstyle\sum_m A_m^n \hat{b}_m + \bar{b}\,,
\end{aligned}
\tag{B.1}
$$

where $u = (p_n - \bar{a}) = (u_1, u_2)^{\top}$, $\hat{a}_m = (\hat{a}_{m,1}, \hat{a}_{m,2})^{\top}$, $\hat{b}_m = (\hat{b}_{m,1}, \hat{b}_{m,2})^{\top}$, and

$$
A_m^n = \theta_m^n \begin{pmatrix} (p_n - \bar{a}) & -(p_n - \bar{a})^{\perp} \end{pmatrix} \begin{pmatrix} \hat{a}_m & -\hat{a}_m^{\perp} \end{pmatrix}. \tag{B.2}
$$

Note that all terms in A_m^n are independent of target control point positions b_m and can therefore be precomputed offline.

Bibliography

[1] Andrea F. Abate, Michele Nappi, Daniel Riccio, and Gabriele Sabatino. 2d and 3d face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885 – 1906, 2007. Image: Information and Control.

[2] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, V19(2):105–114, May 2003.

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[5] Ankerst, Kastenmuller, Kriegel, and Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *SSD: Advances in Spatial Databases*. LNCS, Springer-Verlag, 1999.

[6] K. S. Arun, T. S. Huang, and S. D. Blostein. Least squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:698–700, 1987.

[7] V.R. Ayyagari, F. Boughorbel, A. Koschan, and M.A. Abidi. A new method for automatic 3d face registration. pages 119 –119, June 2005.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded-up robust features. In *9th European Conference on Computer Vision*, Graz, Austria.

[10] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:711–720, July 1997.

[11] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell*, 24(4):509–522, 2002.

[12] Stefano Berretti, Alberto Del Bimbo, and Pietro Pala. 3d face recognition by modeling the arrangement of concave and convex regions. In Stphane Marchand-Maillet, Eric Bruno, Andreas Nrnberger, and Marcin Detyniecki, editors, *Adaptive Multimedia Retrieval: User, Context, and Feedback*, volume 4398 of *Lecture Notes in Computer Science*, pages 108–118. Springer Berlin / Heidelberg, 2007.

[13] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):239–258, February 1992.

[14] Biasotti, Marini, Mortara, Patane, Spagnuolo, and Falcidieno. 3D shape matching through topological structures. In *DGCI: International Workshop on Discrete Geometry for Computer Imagery*, 2003.

[15] Alexander I. Bobenko, Peter. Schrder, and John M. Sullivan. *Discrete differential geometry*. Birkhuser, 2008.

[16] R. C. Bolles and M. A. Fischler. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Image Understanding Workshop*, pages 71–88, 1980.

[17] Bolz, Jeff, Farmer, Ian, Grinspun, Eitan, and Peter Schröoder. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 917–924, New York, NY, USA, 2003. ACM.

[18] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.

[19] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Rössl. Geometric modeling based on polygonal meshes video files associated with this course are available from the citation page. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 1, New York, NY, USA, 2007. ACM.

[20] M.M. Bronstein and I. Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. pages 1704 –1711, June 2010.

[21] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision*, 74(1):59–73, 2007.

[22] Bustos, Keim, Saupe, Schreck, and Vranic. Feature-based similarity search in 3D object databases. *CSURV: Computing Surveys*, 37, 2005.

[23] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, Feb. 2006.

[24] N. Canterakis. 3D zernike moments and zernike affine invariants for 3D image analysis and recognition. In *Scandinavian Conference on Image Analysis*, page Pattern Recognition I, 1999.

[25] Ali Cevahir, Akira Nukada, and Satoshi Matsuoka. Fast conjugate gradients with multiple GPUs. In Gabrielle Allen, Jaroslaw Nabrzyski, Edward Seidel, Geert van Albada, Jack Dongarra, and Peter Sloot, editors, *Computational Science ICCS 2009*, volume 5544 of *Lecture Notes in Computer Science*, pages 893–903. Springer Berlin / Heidelberg, 2009.

[26] C. S. Chua and R. Jarvis. Point signatures: A new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85, October 1997.

[27] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks, and R. Macleod. Coarse filters for shape matching. *Computer Graphics and Applications, IEEE*, 22(3):65–74, May/Jun 2002.

[28] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[29] H. Q. Dinh and S. Kropac. Multi-resolution spin-images. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages I: 863–870, 2006.

[30] R. Donamukkala, D. Huber, A. Kapuria, and M. Hebert. Automatic class selection and prototyping for 3-D object classification. In *3DIM05*, pages 64–71, 2005.

[31] J. Eells and L.H Sampson. Harmonic mappings of riemannian manifolds. *Amer. J. Math.*, 1964.

[32] Michael Eigensatz, Robert W. Sumner, and Mark Pauly. Curvature-domain shape processing. *Comput. Graph. Forum*, 27(2):241–250, 2008.

[33] Luc M. J. Florack, Bart M. ter Haar Romeny, Jan J. Koenderink, and Max A. Viergever. Scale and the differential structure of images. *Image Vision Comput.*, 10(6):376–388, 1992.

[34] Patrick J. Flynn. FRGC database v2.0, 2003. http://bbs.bee-biometrics.org/.

[35] S. Frintrop, A. Nuchter, H. Surmann, and J. Hertzberg. Saliency-based object recognition in 3d data. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 3:2167–2172 vol.3, Sept.-2 Oct. 2004.

[36] A. Frome, D. F. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision*, pages Vol III: 224–237, 2004.

[37] Thomas A. Funkhouser, Patrick Min, Michael M. Kazhdan, Joyce Chen, Alex Halderman, David P. Dobkin, and David P. Jacobs. A search engine for 3D model. *ACM Transactions on Graphics*, 22(1):83–105, January 2003.

[38] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, January 2006.

[39] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 197, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[40] Corey Goldfeder and Peter Allen. Autotagging to improve text search for 3d models. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 355–358, New York, NY, USA, 2008. ACM.

[41] Google. 3D Warehouse. http://sketchup.google.com/3dwarehouse/.

[42] Robert M. Haralick. Propagating covariance in computer vision. In *In Proc. Workshop on Performance Characteristics of Vision Algorithms*, pages 493–498, 1996.

[43] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 85, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[44] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[45] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, 2006.

[46] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[47] Cheuk Yiu Ip, Daniel Lapadat, Leonard Sieger, and William C. Regli. Using shape distributions to compare solid models. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 273–280, New York, NY, USA, 2002. ACM.

[48] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509 – 530, 2005. Geometric Modeling and Processing 2004.

[49] A. Johnson. Surface landmark selection and matching in natural terrain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-00)*, pages 413–421, Los Alamitos, June 13–15 2000. IEEE.

[50] A. E. Johnson and M. Hebert. Surface registration by matching oriented points. In *3DIM97*, pages 4–View Registration, 1997.

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

[52] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[53] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph*, 24:561–566, 2005.

[54] Sing Bing Kang and Katsushi Ikeuchi. The complex EGI: A new representation for 3-D pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7), July 1993.

[55] Z. Karni, D. Freedman, and C. Gotsman. Energy-based image deformation. In *SGP '09: Proceedings of the Symposium on Geometry Processing*, pages 1257–1268, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[56] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 156–164, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[57] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Shape matching and anisotropy. *ACM Transactions on Graphics*, 23(3):623–629, August 2004.

[58] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Symmetry descriptors and 3D shape matching. In Dieter Fellner and Stephen Spencer, editors, *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP-04)*, pages 117–126, Aire-la-Ville, Switzerland, July 8–10 2004. Eurographics Association.

[59] Jan J. Koenderink and Andrea J. van Doorn. Surface shape and curvature scales. *Image Vision Comput.*, 10(8):557–565, 1992.

[60] Kortgen, Park, Novotni, and Klein. 3d shape matching with 3d shape contexts. 2003.

[61] Hamid Laga, Hiroki Takahashi, and Masayuki Nakajima. Spherical wavelet descriptors for content-based 3d model retrieval. *smi*, 0:15, 2006.

[62] Y. Lamdan and H. J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision*, pages 218–49, Tampa, FL, 1988.

[63] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Transactions on Graphics*, 24(3):659–666, July 2005.

[64] Seungyong Lee, George Wolberg, and Sung Yong Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3:228–244, 1997.

[65] George Leifman, Sagi Katz, Ayellet Tal, and Ron Meir. Signatures of 3d models for retrieval. 2003.

[66] David Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67:1517–1531, 1998.

[67] David Levin. Mesh-independent surface interpolation. In *Advances in Computational Mathematics*. in press, 2001.

[68] Tommer Leyvand, Daniel Cohen-Or, Gideon Dror, and Dani Lischinski. Data-driven enhancement of facial attractiveness. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), August 2008.

[69] Xiaoxing Li, Tao Jia, and Hao Zhang. Expression-insensitive 3d face recognition using sparse representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[70] Xinju Li and Igor Guskov. Multi-scale features for approximate alignment of point-based surfaces. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 217, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[71] T. Lindeberg. Scale-space theory in computer vision, kluwer, dordrecht. *Monograph 1994*, 1994.

[72] Tony Lindeberg. Scale-space theory in computer vision, 1994.

[73] Tony Lindeberg and Lars Bretzner. Real-time scale selection in hybrid multi-scale representations.

[74] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 78:1–78:10, New York, NY, USA, 2008. ACM.

[75] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[76] Looklet.com. Looklet.com. http://looklet.com/.

[77] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[78] Mohammad H. Mahoor and Mohamed Abdel-Mottaleb. Face recognition based on 3d ridge images obtained from range data. *Pattern Recogn.*, 42:445–451, March 2009.

[79] A. Makadia, A. I. Patterson, and K. Daniilidis. Fully automatic registration of 3D point clouds. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages I: 1297–1304, 2006.

[80] Siddharth Manay, Byung woo Hong, and Anthony J. Yezzi. Integral invariant signatures. In *In ECCV04, volume LNCS 2034*, pages 87–99. Springer, 2004.

[81] D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Royal Society of London*, B 200:269–294, 1978.

[82] D. H. McLain. Drawing Contours from Arbitrary Data Points. *The Computer Journal*, 17(4):318–324, 1974.

[83] M. Meyer, M. Desbrun, P. Schröder, and A.H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics*, 3(7):34–57, 2002.

[84] Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. Automatic 3d face detection, normalization and recognition. pages 735–742, 2006.

[85] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33(11):1771–1782, 2000.

[86] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):696 –710, jul. 1997.

[87] F. Mokhtarian and M.Z. Bober. *Curvature scale space representation: theory, applications, and MPEG-7 standardization*. Kluwer Academic Pub, 2003.

[88] Farzin Mokhtarian and Alan K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. Pattern Anal. Mach. Intell*, 14(8):789–805, 1992.

[89] A.B. Moreno and A. Sanchez. GavabDB: A 3D Face Database. *Proc. 2nd COST Workshop on Biometrics on the Internet: Fundamentals, Advances and Applications*, pages 77 – 82, 2004.

[90] A.B. Moreno, A. Sanchez, J. Velez, and J. Diaz. Face recognition using 3d local geometrical features: Pca vs. svm. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 185 – 190, 2005.

[91] David Mount and Sunil Arya. Ann - approximate nearest neighbors. http://www.cs.umd.edu/ mount/ANN/.

[92] M.H. Mousavi, K. Faez, and A. Asghari. Three dimensional face recognition using svm classifier. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 208 –213, May 2008.

[93] J. Novatnack, K. Nishino, and A. Shokoufandeh. Extracting 3D shape features in discrete scale-space. In *3DPVT06*, pages 946–953, 2006.

[94] Marcin Novotni, Patrick Degener, and Reinhard Klein. Correspondence generation and matching of 3d shape subparts. Technical report, Universitt Bonn, 2005.

[95] Marcin Novotni and Reinhard Klein. 3d zernike descriptors for content based shape retrieval. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 216–225, New York, NY, USA, 2003. ACM.

[96] R. Ohbuchi, T. Otagiri, M. Ibato, and T. Takei. Shape-similarity search of three-dimensional models using parameterized statistics. *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pages 265–274, 2002.

[97] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3D models with shape distributions. In Bob Werner, editor, *Proceedings of the International Conference on Shape Modeling and Applications (SMI-01)*, pages 154–166, Los Alamitos, CA, May 7–11 2001. IEEE Computer Society.

[98] E. Paquet and M. Rioux. Nefertiti: a query by content software for three-dimensional models databases management. *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 345–352, May 1997.

[99] E. Paquet and M. Rioux. The mpeg-7 standard and the content-based management of three-dimensional data: A case study. *icmcs*, 01:9375, 1999.

[100] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. In P. Brunet and D. Fellner, editors, *Proceedings of Eurographics 2003*, volume 22(3) of *Computer Graphics Forum*, pages 281–289. Blackwell Publishing Inc, September 2003.

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

[102] Sylvain Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Comput. Surv.*, 34(2):211–262, 2002.

[103] Helmut Pottmann, Johannes Wallner, Qi-Xing Huang, and Yong-Liang Yang. Integral invariants for robust geometry processing. *Computer Aided Geometric Design*, 26(1):37 – 60, 2009.

[104] Princeton. 3d model search engine. http://shape.cs.princeton.edu/.

[105] R. Redner and H. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM Review*, 26(2):195–239, 1984.

[106] Dietmar Saupe and Dejan V. Vranic. 3d model retrieval with spherical harmonics and moments. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 392–397, London, UK, 2001. Springer-Verlag.

[107] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25(3):533–540, 2006.

[108] Alize Scheenstra, Arnout Ruifrok, and Remco Veltkamp. A survey of 3d face recognition methods. In Takeo Kanade, Anil Jain, and Nalini Ratha, editors, *Audio- and Video-Based Biometric Person Authentication*, volume 3546 of *Lecture Notes in Computer Science*, pages 891–899. Springer Berlin / Heidelberg, 2005.

[109] M. Schlattmann, P. Degener, and R. Klein. Scale space based feature point detection on surfaces. *Journal of WSCG*, 16(1-3), February 2008.

[110] Schway.net. Schway.net. http://schway.net/.

[111] Ling Shao and Michael Brady. Invariant salient regions based image retrieval under viewpoint and illumination variations. *J. Vis. Comun. Image Represent.*, 17(6):1256–1272, 2006.

[112] Gregory C. Sharp, Sang Wook Lee, and David K. Wehe. ICP registration using invariant features. *IEEE Trans. Pattern Anal. Mach. Intell*, 24(1):90–102, 2002.

[113] J. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *Applied Computational Geometry Towards Geometric Engineering*, pages 203–222, 1996.

[114] J.R. Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry*, 22(1-3):21–74, 2002.

[115] Philip Shilane and Thomas Funkhouser. Selecting distinctive 3d shape descriptors for similarity retrieval. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, page 18, Washington, DC, USA, 2006. IEEE Computer Society.

[116] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3d surfaces. *ACM Trans. Graph.*, 26(2):7, e 07.

[117] T. Sikora. The mpeg-7 visual standard for content description-an overview. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):696–702, Jun 2001.

[118] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.

[119] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[120] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Eurographics Symposium on Geometry Processing (SGP)*, 2009.

[121] Y. Y. Sun and M. A. Abidi. Surface matching by 3D point's fingerprint. In *International Conference on Computer Vision*, pages II: 263–269, 2001.

[122] Y. Y. Sun, J. K. Paik, A. F. Koschan, D. L. Page, and M. A. Abidi. Point fingerprint: A new 3-D object representation scheme. *IEEE Trans. Systems, Man and Cybernetics*, 33(4):712–717, August 2003.

[123] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. *Shape Modeling International, 2003*, pages 130–139, May 2003.

[124] Tangelder and Veltkamp. A survey of content based 3d shape retrieval methods. In *SMI '04: Proceedings of the Shape Modeling International 2004*, pages 145–156, Washington, DC, USA, 2004. IEEE Computer Society.

[125] 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.

[126] Gabriel Taubin. Eurographics 2000 star state of the art report abstract geometric signal processing on polygonal meshes, 2000.

[127] Bart M. ter Haar Romeny. Front-end vision and multi-scale computer vision theory and applications, written in mathematica, 2003.

[128] Jean-Philippe Thirion and Serge BENAYOUN. Image surface extremal points, new feature points for image registration. 1993.

[129] J. P. Vandeborre, V. Couillet, and M. Daoudi. A practical approach for 3D model indexing by combining local and global invariants. In *3DPVT02*, pages 644–647, 2002.

[130] R.C. Veltkamp. Shape matching: similarity measures and algorithms. *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 188–197, May 2001.

[131] D. V. Vranic. An improvement of rotation invariant 3D-shape descriptor based on functions on concentric spheres. In *International Conference on Image Processing*, pages III: 757–760, 2003.

[132] D. V. Vranic and D. Saupe. 3d shape descriptor based on 3d fourier transform. In *Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001) (editor K. Fazekas)*, pages 271–274, Budapest, Hungary, September 2001.

[133] D.V. Vranic, D. Saupe, and J. Richter. Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics. *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pages 293–298, 2001.

[134] K. N. Walker, T. F. Cootes, and C. J. Taylor. Locating salient facial features using image invariants. In *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, page 242, Washington, DC, USA, 1998. IEEE Computer Society.

[135] Ofir Weber, Mirela Ben-Chen, and Craig Gotsman. Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2), 2009.

[136] Ofir Weber and Craig Gotsman. Controllable conformal maps for shape deformation and interpolation. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers*, pages 78:1–11, New York, NY, USA, 2010. ACM.

[137] Gaojin Wen, Dengming Zhu, Shihong Xia, and Zhaoqi Wang. Total least squares fitting of point sets in m-d. In *CGI '05: Proceedings of the Computer Graphics International 2005*, pages 82–86, Washington, DC, USA, 2005. IEEE Computer Society.

[138] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[139] George Wolberg. Image morphing: A survey. *The Visual Computer*, 14:360–372, 1998.

[140] R. J. Woodham. Stable representation of shape. Technical Report TR-87-05, Department of Computer Science, University of British Columbia, February 1987. Mon, 21 Jul 1997 19:42:09 GMT.

[141] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210 –227, feb. 2009.

[142] Sameh M. Yamany and Aly A. Farag. Surface 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.

[143] Y. Yang, H. Lin, and Y. Zhang. Content-based 3-D model retrieval: A survey. *IEEE Trans. Systems, Man and Cybernetics*, 37(6):1081–1098, November 2007.

[144] C. Zhang and T. Chen. Efficient feature extraction for 2D/3D objects in mesh representation. pages 935–938.

[145] D. M. Zhang and M. Hebert. Experimental analysis of harmonic shape images. In *3DIM99*, pages 209–218, 1999.