Geodesics for Point Based Model Processing

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
an der Universität Konstanz

Mathematisch-Naturwissenschaftliche Sektion,
Fachbereich Informatik und Informationswissenschaft

vorgelegt von
Mauro Roberto Ruggeri

Juli 2007

Tag der mündlichen Prüfung: 06.07.2007

Referenten: Prof. Dr. Dietmar Saupe, Universität Konstanz
Prof. Dr. Oliver Deussen, Universität Konstanz

Konstanzer Online-Publikations-System (KOPS)
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2007/3934/
URN: http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-39342

Questa tesi è dedicata alla mia famiglia, che significa tutto per me. In modo particolare, questi anni di studio e lavoro li voglio dedicare a mia madre, per la sua forza di scegliere sempre la strada giusta, sebbene sia la più difficile.

Acknowledgements

For his invaluable guidance and scientific support I would firstly like to thank my supervisor, Prof. Dr. Dietmar Saupe. I am also very grateful to Prof. Dr. Oliver Deussen for his kindness and support during my time at the University of Konstanz.

Many thanks to Prof. Dr. Nahum Kiryati who was most welcoming during my stay at Tel Aviv University and provided much assistance and relevant scientific collaboration. A special thanks goes also to Tal Darom for his important contribution to the thesis and the pleasant and fruitful discussions.

I am very grateful to Prof. Dr. Raouf Hamzoui, Dr. Dejan Vranić, and Dr. Giuseppe Di Fatta who were fundamental in providing me with excellent scientific advices and encouragements to complete the thesis as well as friendship.

I would also like to thank all my colleagues for the pleasant chats and discussions, especially Anna Dowden-Williams and Alina Cleju for proofreading part of this thesis.

This research was partly supported by the DFG Graduiertenkolleg "Explorative Analysis and Visualization of Large Information Spaces" and the Kurt Lion Foundation.

Last but not least, I would like to thank all my friends who supported me with their friendship, esteem, and moral support during the past years.

Konstanz, July 2007

Mauro R. Ruggeri

Abstract

In this thesis we present a framework for analysis and processing of point set surfaces based on geodesics. The first part deals with computing geodesics on point set surfaces. Two main techniques are presented.

The first technique efficiently computes fast approximations of geodesics. It relies on a graph representation of the surface using an extended sphere of influence graph (eSIG). Approximations of geodesics are computed as shortest paths on the eSIG by using Dijkstra-like algorithms.

The second technique computes accurate approximations of geodesics and is robust with respect to noise and outliers. It is a novel technique, which computes piecewise linear approximations of geodesics on point set surfaces by minimizing an energy function defined for piecewise linear paths.

This thesis presents two surface processing/analysis applications, in which geodesics are used for compression of textured point based surfaces of genus-0, and for point based model retrieval.

The compression scheme described in this thesis, is based on a spherical representation of a genus-0 surface represented by a surfel set. It incorporates geometry, texture and other surface attributes in a unified manner. Each surfel, with its attribute vector, is mapped onto a sphere by using a novel and efficient approach for dense surface mapping, which considers rectangular distance matrices. In this approach the geodesic distances between each surfel and a set of key-surfels are optimally preserved in order to improve the resolution and eliminating the need for interpolation that complicates and slows down existing surface unfolding methods. The resulting spherical vector-valued function is then regularly resampled. Its components are decorrelated by the Karhunen-Loève transform, represented by spherical wavelets and encoded using the zerotree algorithm. The compression-induced geometry error is smaller than that of other methods at very low bit-rates.

A system for point based model retrieval is also presented. It comprises: a surfelization technique for converting polygonal mesh models into corresponding surfel based representations, and two shape descriptors for comparing efficiently the geometry and the topology of point based models.

The presented surfelization technique is designed for converting large databases of triangular mesh models into corresponding surfel based representations. It consists of an enhanced triangle rasterization procedure adapted to the geometric features of a triangulated model, which preserves its main visual and geometric characteristics in the resulting surfel based model.

The first shape descriptor focuses on effectively matching the geometry of surfel based models. It is constructed through a pose normalization of surfel based models, and a Fourier analysis of depth-buffer images, which are rendered from different viewpoints. Our depth-buffer image based shape descriptor is able to compare the geometry of several classes of objects. Moreover, it possesses an embedded multi-resolution representation, is invariant with respect to similarity transforms and robust with respect to outliers.

The second shape descriptor is based on geodesics, and is designed for geometry and topology matching of point based models. A normalized geodesic distance matrix is computed from a set of reference points, which are selected randomly on a point set surface. The similarity of two point based models is then computed by comparing the associated distance matrices using a histogram based approach. This technique resulted in being an effective method for the comparison of deformed objects of the same class, since it is invariant with respect to similarity and isometric transformations.

Zusammenfassung

In dieser Dissertation präsentieren wir computergrafische Verfahren für die Analyse und Verarbeitung von Oberflächen, die durch Punktmengen repräsentiert sind. In den Verfahren spielen geodätische Kurven eine zentrale Rolle. Der erste Teil der Arbeit befasst sich mit dem Berechnen von geodätischen Kurven auf punktbasierten Flächen. Es werden zwei Haupttechniken präsentiert.

Die erste ergibt ein Verfahren zur effizienten Berechnung von Approximationen der geodätischen Kurven. Sie basiert auf einer grafischen Darstellung der Fläche durch den sogenannten erweiterten sphere-of-influence-graph (eSIG). Approximationen der geodätischen Kurven werden als kürzeste Pfade des eSIGs berechnet mit Hilfe von Dijkstra-ähnlichen Algorithmen.

Mit der zweiten Technik werden genauere Approximationen der geodätische Kurven berechnet. Sie ist robust in Bezug auf Rauschen und Ausreißer. Stückweise lineare Approximationen von geodätischen Kurven auf einer punktbasierten Fläche sind dabei als Minima einer geeignet definierten Energiefunktion definiert.

In dieser Arbeit werden zwei Anwendungen geodätischer Kurven entwickelt, (1) die Kompression texturierter punktbasierter Flächen von Geschlecht 0 und (2) die Ähnlichkeitssuche in Datenbanken von punktbasierten 3D-Modellen.

(1) Das hier beschriebene Kompressionsverfahren basiert auf einer sphärischen Repräsentation einer Genus-0 Fläche dargestellt durch eine Menge sogenannter Surfel (eine erweiterte punktbasierte Form). Es behandelt Geometrie, Texturen und andere Flächeneigenschaften in einer einheitlichen Art und Weise. Die Surfel werden auf eine Sphäre abgebildet, wobei die geodätischen Abstände auf der Fläche so gut wie möglich erhalten bleiben. Komplexitätsprobleme werden durch eine Einschränkung der in die Berechnung eingehenden Abstände gelöst. Die resultierende

Vektor-bewertete Funktion auf der Sphäre wird dann regelmäßig abgetastet (resampled). Die Komponenten werden dekorreliert durch die Karhunen-Loeve Transformation, dargestellt durch kugelförmige Wavelets und mit dem zerotree Algorithmus kodiert. Der von der Kompression verursachte geometrische Rekonstruktionsfehler ist kleiner als der bei anderen Methoden bei sehr niedrigen Bit-Raten.

(2) Unser System für punktbasiertes Modellretrieval beinhaltet: eine Surfeltechnik für das Umwandeln von polygonalen Netzmodellen (mesh models) in entsprechende surfel-basierte Darstellungen und zwei Formbeschreiber um die Geometrie und die Topologie der punktbasierten Modelle effizient zu vergleichen.

Die Surfeltechnik besteht aus einer Rasterisierung von Polygonen, die den geometrischen Eigenschaften der triangulierten Modelle angepasst ist, so dass die wesentlichen visuellen und geometrischen Eigenschaften im resultierenden surfel-basiertes Modell erhalten bleiben.

Für die erste Formbeschreibung surfel-basierter Flächer wird die Fläche skaliert und aus verschiedenen Blickwinkeln gerendert. Die dabei erzeugten Tiefenpuffer werden Fourier-transformiert und einige der Koeffizienten werden im Outputvektor der Formbeschreibung aufgenommen. Die Formbeschreibung besitzt so eine eingebettete Multiskalen-Darstellung, ist invariant in Bezug auf ähnlichkeitstransformationen und robust in Bezug auf Ausreißer.

Der zweite Formbeschreiber basiert auf geodätischen Kurven und ist für das Abgleichen der Geometrie und Topologie der punktbasierten Modelle bestimmt. Unsere Technik wählt eine Anzahl gleichmäßig verteilter Bezugspunkte auf der punktbasierten Fläche aus. Dann wird die zugehörige normalisierte geodätische Distanzmatrix berechnet. Die Ähnlichkeit von zwei punkt-basierten Modellen wird durch einen Histogramm-basierten Vergleich der beiden zugehörigen Abstands-Matrizen ermittelt. Diese Technik ist effektiv für den Vergleich von 3D Objekten und invariant bezüglich isometrischer Verformungen.

Contents

1 Introduction **1**

1.1 Goals and contributions . . . 3

1.2 Outline . . . 7

2 Computing geodesics on point set surfaces **9**

2.1 Introduction . . . 10

2.1.1 Definition of a geodesic . . . 10

2.1.2 Definition of a surfel . . . 14

2.1.3 Definition of the problem . . . 14

2.2 Related work . . . 15

2.3 Computing fast approximations of geodesics on point set surfaces . . . 17

2.3.1 Extended sphere of influence graph (eSIG) . . . 18

2.3.2 Computing shortest paths . . . 21

2.4 Energy minimizing piecewise linear geodesics on point set surfaces . . . 23

2.4.1 Approximating geodesics by energy minimization . . . 23

2.4.2 Computing an initial path . . . 25

2.4.3 Energy function minimization . . . 25

2.4.4 Multi-resolution approach for energy function minimization . . . 27

2.4.5 Results . . . 29

2.4.6 Experimental convergence and error analysis . . . 30

2.4.7 Robustness with respect to noise . . . 33

3 Geodesics for compressing textured point based models **39**

3.1 Introduction . . . 39

3.2 Spherical mapping . . . 41

3.2.1 Computing geodesic paths on surfel based surfaces . . . 44

3.2.2 Performance evaluation of the spherical mapping . . . 45
3.3 Textured surface compression . . . 48
3.3.1 Representation and compression scheme . . . 48
3.3.2 Special case: geometry only . . . 51
3.4 Results . . . 53
3.4.1 Geometry only compression . . . 53
3.4.2 Compression of textured surfaces . . . 53

4 Point based model retrieval 59
4.1 3D model retrieval . . . 60
4.1.1 3D model retrieval system . . . 60
4.1.2 Evaluating the retrieval performance of a shape descriptor . . . 65
4.2 Related work . . . 68
4.3 Surfelization of triangular mesh databases . . . 73
4.3.1 Previous work . . . 75
4.3.2 Surfelization algorithm . . . 78
4.3.3 Results . . . 80
4.4 Shape descriptor for shape matching of point based models . . . 83
4.4.1 Normalization (pose estimation) . . . 84
4.4.2 Feature extraction . . . 84
4.4.3 Results . . . 89
4.5 Topology matching . . . 92
4.6 Geodesics for topology matching of point based models . . . 99
4.6.1 Sampling a point set surface . . . 102
4.6.1.1 Farthest point sampling . . . 106
4.6.2 Computing a geodesic distance matrix . . . 109
4.6.3 Comparing geodesic distance matrices . . . 109
4.6.4 Results . . . 113

5 Conclusions 119

A Isometric transformation 125

B Color plates of Chapter 1 and 2 127

C Color plates of Chapter 3 133

D Color plates of Chapter 4 **139**

References **161**

Chapter 1

Introduction

(a) (b)

Figure 1.1: Point based representations of the *Butterfly* model by Arius Inc. [10]: (a) point cloud; (b) surfels (splats). **[see Color plate B.1]**

In recent years the complexity of photo-realistic 3D models rapidly increased. Photo-realistic models are mainly generated by using two approaches: (1) via 3D acquisition techniques [68, 90, 144, 156, 172], and (2) via sophisticated 3D modelling techniques [43, 44, 80, 81, 169]. In both cases photo-realistic models are commonly represented by meshes of textured triangles so that they can be efficiently rendered by the graphics systems.

In many fields including culture heritage, reverse engineering, architecture, and medical applications, surfaces in $\mathbb{R}^3$ must be acquired from reality. The current 3D acquisition technologies produce huge sets of 3D point clouds that subsequently have to be processed for different purposes, such as: rendering, surface analysis/processing and modelling. This large input data set increased the practically attainable sizes of triangle meshes to hundreds of millions of triangles.

Regardless of powerful graphics and computer hardware, these huge amounts of data do still generate a bottleneck when managing triangle meshes in interactive/simulation applications such as: interactive modelling/rendering and simulation of physical phenomena (e.g., fracture simulation of solids and simulation of fluids motion in continuum mechanics).

To partially circumvent this problem, points and point based primitives emerged as a representation type complementing triangular meshes. Points offer a flexible representation of surfaces, which is not constrained by connectivity relations, and is suitable for multi-resolution and out-of-core methods. Nevertheless, point based representations have several disadvantages with respect to triangle meshes, especially in the rendering of realistic scene and in CAD modelling. In fact, the advantages and the disadvantages of both surface representations depend on the application.

Triangle meshes encode connectivity explicitly and statically, via edge/vertex/face-adjacency relations. On one hand this fact can be considered as an advantage for geometric processing algorithms that make a strong use of topological information. A constant-time access to neighbored vertices in combination with adequately discretized differential operators, makes meshes well suited for a variety of modelling and surface processing operations. On the other hand in those cases where the topology of a 3D model needs to be updated dynamically and in real time, triangular meshes are not appropriate, and often imply high computation time. This problem arises in interactive modelling and in fracture simulation of solids due to large deformations and fractures, respectively. In such cases point based representations turned out to be more efficient than triangle meshes. The lack of connectivity of point based representation allows to conveniently resample the deformed parts of an object without the need to reconstruct the topological representation on the fly. The drawback of this approach is the cost of resampling. In fact resampling of point clouds is mainly accomplished through continuous surface reconstructions using moving least squares (MLS) [5, 8, 33, 70], Radial basis functions (RBF) [46, 180, 200] and other techniques [48, 73, 151, 159]. In general, these methods need to compute the k-nearest neighborhood of specific subset of point, which requires more elaborate data structures including kd-trees [154], or spatial hashing [142].

Often in rendering applications the geometric complexity and the resolution of triangle meshes are so high that polygons cover an area less than a pixel. In such cases, a point based approach would be more suitable than a triangle based approach, since it implies less computational effort per rendering primitive and requires less memory bandwidth, both for texture and geometry. Powerful rendering paradigms were recently presented for high quality

rendering of point based models. In [187] Pfister et al. presented surfels as a powerful point based surface representation scheme for rendering. Surfels are oriented circular or elliptical disks in $\mathbb{R}^3$ containing color information that describe surface characteristics within the neighborhood of each sample point. Neither connectivity nor other topology information is stored. Surfels are rendered with splatting techniques that achieve a good rendering quality. A splatting technique consists in splatting 3D points onto the image plane according to their geometric and color characteristic. Efficient and photo-realistic splatting techniques to render point based geometry were presented in [5, 31, 199, 257, 269, 270]. Some authors also implemented high-quality ray tracing of point based models [2, 4].

Point based representations showed to be very convenient to model and render complex geometry, like procedural geometry. Point samples were utilized successfully in particle systems for the physically based animation of fluids or smoke. However, for some classes of geometric models like CAD models point based representations require a denser sampling compared to triangle meshes. For high resolution models, this fact can extremely increase the bandwidth requirements between CPU and GPU, which requires a tread-off with the processing speed. Although, currently the graphics hardware is still optimized for triangular meshes, point based graphics architectures might become part of future generation GPUs accelerating the point based rendering speed and complementing triangle rendering. A more detailed review of the advantages and drawbacks of point based representation can be found in [97, 133].

Although, point based primitives have a long history in computer graphics [47, 98, 143, 198], they only recently received strong attention, such that a significant trend in computer graphics has been to develop methods for processing and high-quality rendering of point set surfaces. Common paradigms applied for triangle meshes were and will be transferred to point based geometry and optimized for point based representations [133, 185, 268]. Hence, in the next years the need of algorithms working directly on point based geometry will become prominent in many fields such as: medical and geographical applications, city modelling, cultural heritage, architecture, and physics. For this reason, 3D point based graphics proposes various research areas to further investigate.

1.1 Goals and contributions

This thesis addresses three specific problems in the field of point based graphics by considering geodesics as the key concept of point set surface analysis and processing:

- computing geodesics on point based surfaces;
- compression of textured point based models;
- shape description and retrieval of point based models.

Geodesics, i.e., shortest paths on surfaces, have applications in several surface analysis and processing techniques including: parametrization [104, 209, 265], partitioning [125], flattening [99], modelling [134, 261], segmentation [186], remeshing [222], texture mapping [265], shape matching [252], and compression [165]. Geodesics being naturally related to the intrinsic geometry and the topology of their underlying surface may lead to effective characterizations of surfaces. Moreover, they are invariant with respect to isometric transformations like bending deformations of surfaces (see Appendix A), and represent a useful tool for shape matching and retrieval.

In this thesis we present a framework for analysis and processing of point set surfaces based on geodesics. Considering the lack of topology information in point based representations, the problem of computing exact geodesics on point set surfaces is challenging and relevant for many applications such as: robot motion planning [111, 115], shape analysis [85], geographical applications [182, 224], and for many applications in the medical field [42, 192, 253]. We present two methods to compute approximations of geodesics on point set surfaces. Moreover, our framework provides the basis for two techniques in which geodesics are used for processing and analysis of point based models, namely, for compression of textured point based models, and for point based model retrieval. The contributions of our work in this context can be summarized as follows:

Computing geodesics on point set surfaces (Chapter 2): our system for computing geodesics on point set surfaces comprises two main techniques:

1. a technique to compute fast approximations of geodesics on point clouds and optimized for surfels [66, 67, 203] (Section 2.3);
2. a novel technique to compute accurate piecewise linear approximations of geodesics on point set surfaces that is robust with respect to noise and outliers [203] (Section 2.4).

The first technique relies on a graph representation of the surface using an extended sphere of influence graph, which is optimized for surfels (Section 2.3). Approximations of geodesics are computed as shortest paths on the graph using Dijkstra-like algorithms.

The second technique computes geodesics as piecewise linear paths minimizing an energy function (Section 2.4). This technique produces accurate approximations of geodesics and is robust with respect to noise and outliers. It is also general, since it can be applied to any surface representations, e.g., moving least-squares surfaces, triangle meshes, etc. We implemented our technique for two point based representations: surfels (oriented circular disks with rendering attributes), and moving least squares (MLS) surfaces. Our technique possesses also multi-resolution features, which make it suitable for computing geodesics on large data sets. It can also be extended to other curve representations, e.g., Bezier curves, splines, hierarchical B-splines, curves based on wavelets, etc.

Compressing textured point based models (Chapter 3): our algorithm for computing fast geodesics approximations is used to develop a novel surface compression scheme and spherical representation for genus-0 surfaces represented by surfel sets. This compression scheme, presented by Darom et al. in [66, 67], is described in this thesis. It handles various surface attributes, notably geometry and texture, within a unified representation framework. The input surface is represented by a surfel set. Each surfel, with its attribute vector, is optimally mapped onto a sphere in the sense of geodesic distance preservation. The resulting spherical vector-valued function is then resampled on a regular spherical grid. Its components are decorrelated by the Karhunen-Loève transform, represented by spherical wavelets and encoded using the zerotree algorithm. This compression scheme includes a novel and efficient approach for dense surface flattening/mapping, using rectangular distance matrices where the computation of geodesic distances is crucial. In fact, the geodesic distances between each surfel and a set of key-surfels are optimally preserved in order to improve the resolution and eliminating the need for interpolation that complicates and slows down existing surface unfolding methods. The compression-induced geometry error is smaller than that of other methods at very low bit-rates.

Point based model retrieval (Chapter 4): we present a point based model retrieval system, in which a surfel based model serves as a query and similar objects are retrieved from a collection of surfel based models [204]. The system comprises:

1. a surfelization technique for converting triangular mesh models into corresponding surfel based representations (Section 4.3). Our surfelization technique consists of an enhanced triangle rasterization procedure adapted to the geometric features of the

original triangulated model. It is designed to automatically convert big collection of triangular meshes preserving their main visual and geometric characteristics.

2. a shape descriptor for surfel based models that is based on depth-buffer images (Section 4.4). It is generated through a pose normalization of surfel based models and a Fourier analysis of depth-buffer images that are rendered from different viewpoints. Our depth-buffer image based shape descriptor possesses an embedded multi-resolution representation, is invariant with respect to similarity transforms and robust with respect to outliers.

3. a technique for geometry and topology matching of point based models that is based on geodesics (Section 4.6). Our technique selects N random points evenly distributed on the point set surface. Then, a normalized geodesic distance matrix is computed by storing in each element the geodesic distance between the two corresponding random points. The similarity of two point based models is then computed by comparing the associated distance matrices using a histogram based approach. Our technique resulted in being an effective tool for comparing objects, which are deformed through isometric transformations.

The retrieval effectiveness of the implemented similarity search algorithms for point based models is evaluated with well known tools coming from information retrieval theory.

1.2 Outline

This thesis is organized as follows:

- Chapter 2 introduces geodesics and presents two techniques for computing geodesics on point based surfaces. Experimental results on noisy surfaces with outliers are provided together with a performance evaluation of both techniques.
- Chapter 3 describes a compression scheme and a spherical representation for textured surfel set surfaces of genus-0. Compression performances of the described method are shown by reporting the geometry compression error as a function of the bit-rate. A comparison with other methods is also provided.
- Chapter 4 presents a system for comparing and retrieving point based models. It describes two shape similarity search algorithms for matching geometry and topology of point set surfaces, and a surfelzation technique for a feature preserving conversion of triangular mesh models into corresponding surfel based representations. An introduction to 3D model retrieval is provided together with a brief review of the main matching methods for geometry and topology of 3D models. Tools for evaluating the retrieval effectiveness of similarity search techniques are described and used to assess the retrieval performance of the implemented algorithms.
- Chapter 5 summarizes the thesis and concludes with some directions for future work.

Chapter 2

Computing geodesics on point set surfaces

Point set geometry represents the typical output of the current 3D acquisition techniques, which generate sets of noisy 3D point clouds that subsequently have to be processed for different purposes. The problem of computing geodesics, i.e., shortest paths on surfaces, arises in different applications such as robot motion planning [111, 115], shape analysis [85] and geographical information systems. Furthermore, geodesics have been widely used in surface analysis and processing, including parametrization [104, 209, 265], partitioning [125], flattening [99], modelling [134, 261], segmentation [186], remeshing [222], texture mapping [265], shape matching [252], and compression [165].

This chapter starts providing an introduction to the problem of computing geodesics on point set surfaces. A geodesic and a surfel (the main point based primitive used in this thesis) are defined in Section 2.1. Following, two fundamental techniques that are used to compute approximations of geodesics on point set surfaces, are presented. The first technique computes a fast and efficient approximation of geodesic distance and geodesic paths. It relies on a rough representation of the surface topology by means of an extended sphere of influence graph. The second technique serves to compute accurate approximation of geodesic curves via energy minimizing piecewise linear paths. This technique is robust with respect to noise and outliers, and it can be applied to several surface representations, which are constructed from the original set of sampled, e.g., MLS surfaces, RBF surfaces, triangular meshes, etc. These techniques will then be used in the subsequent parts of this thesis to define more sophisticated point set surface processing algorithms.

2.1 Introduction

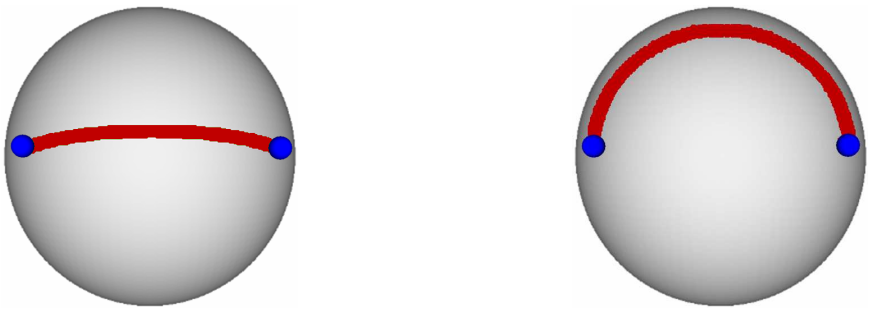
2.1.1 Definition of a geodesic

A *geodesic* is commonly defined as the shortest curve/path between two points on a surface. However, the definition of a geodesic in differential geometry is more general. For smooth parametric surfaces in $\mathbb{R}^3$ a geodesic can be defined as follows. Let $\Psi(u, v)$ be a parametrization of a smooth surface Ψ in $\mathbb{R}^3$, and let $\varphi(s) = \Psi(u(s), v(s))$ be a curve on Ψ parameterized by arc length, where s denotes the arc length parameter. A curve $\varphi(s)$ is called *geodesic* if it is a solution of the following system of differential equations:

$$\frac{d^2u}{ds} + \Gamma^1_{11}\left(\frac{du}{ds}\right)^2 + 2\Gamma^1_{12}\frac{du}{ds}\frac{dv}{ds} + \Gamma^1_{22}\left(\frac{dv}{ds}\right)^2 = 0 \tag{2.1}$$

$$\frac{d^2v}{ds} + \Gamma^2_{11}\left(\frac{du}{ds}\right)^2 + 2\Gamma^2_{12}\frac{du}{ds}\frac{dv}{ds} + \Gamma^2_{22}\left(\frac{dv}{ds}\right)^2 = 0 \tag{2.2}$$

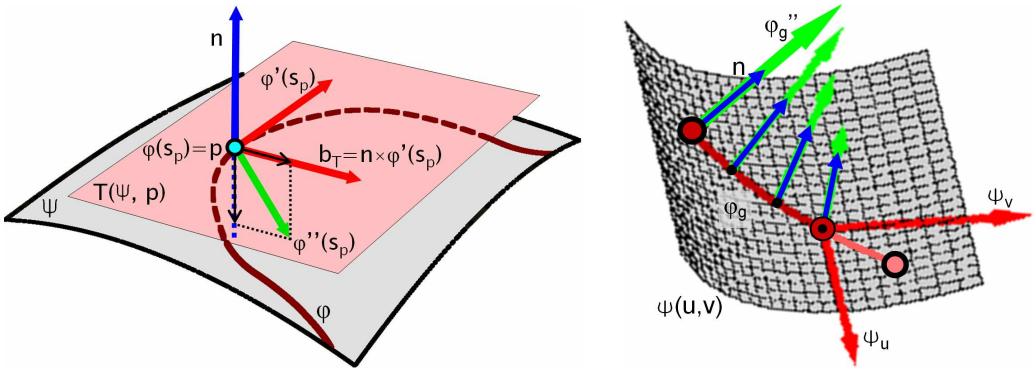
where Γ^i_{jk} are the Christoffel symbols of the second kind, which are derived by the metric of the surface [75, 210]. Geodesics depend on the metric. Different metrics give origin to different geodesics. Figure 2.1 shows a geodesic computed on a sphere by using the Euclidean metric (Figure 2.1a) and the hyperbolic metric (Figure 2.1b).



(a) (b)

Figure 2.1: Dependence of geodesics on the metric. A geodesic is computed on a sphere using two different metrics: (a) Euclidean metric, and (b) hyperbolic metric.

An alternative geometrical interpretation of a geodesic is given in Figure 2.2. Figure 2.2a shows the most relevant geometric properties of a general curve φ on Ψ to understand the definition of a geodesic, while Figure 2.2b shows the geodesic curve φ_g. Let φ'' be the vector of the second derivatives of the curve φ at a point $p = \varphi(s_p)$, the vector φ'' is perpendicular to the tangent vector φ' of the curve φ. φ'' is related to the curvature of φ. The curvature of φ at p is defined as $k(s_p) = |\varphi''(s_p)|$, and indicates how rapidly the curve φ pulls away from the tangent



(a) (b)

Figure 2.2: Geometric properties of a curve and a geodesic. (a) A parameterized curve φ on a surface Ψ. The second derivative vector $\varphi''(s_p)$ (green vector) of φ at the point $p = \varphi(s_p) \in \Psi$ is shown with respect to other geometric properties, such as: the tangent plane $T(\Psi, p)$ of Ψ at p (pink plane); the normal vector n of Ψ at p (blue vector); the tangent vector $\varphi'(s_p)$ of φ at p (red vector 1); the vector $b_T = \varphi'(s_p) \times n$ (red vector 2); the components of the vector $\varphi''(s_p)$ on the direction of the vectors $\varphi'(s_p)$ and b_T (black vectors). (b) A geodesic curve (φ_g) on a parametric surface ($\psi(u, v)$) visualized as a trajectory of a particle. Along the trajectory of the particle, particle's acceleration vectors (φ_g'', in green) are parallel to the surface normals (n, in blue). The vectors spanning the tangent plane of $\psi(u, v)$ at the current particle position are shown as ψ_u and ψ_v (red vectors). **[see Color plate B.2]**

line at s_p in a neighborhood of s_p [75]. Let n be the surface normal vector at p, obviously n is perpendicular to the tangent plane $T(\Psi, p)$ of Ψ (see Figure 2.2a). Let $b_T = \varphi'(s_p) \times n$ be the cross product between the vectors $\varphi'(s_p)$ and n, b_T and $\varphi'(s_p)$ lie on the plane $T(\Psi, p)$, since φ is on the surface Ψ. Thus, the vector φ'' can be defined in terms of its components on the direction of the vectors b_T and n, as $\varphi'' = h_n n + h_b b_T$. These two components are shown as black vectors in Figure 2.2a.

The curve φ on Ψ is a geodesic only if $\varphi'' \cdot \Psi_u = 0$ and $\varphi'' \cdot \Psi_v = 0$, where Ψ_u and Ψ_v denote the first-order derivative vectors with respect to u and v, respectively. Therefore, at each point of a geodesic φ_g the vector φ_g'' is normal to the tangent plane of the surface at that point (spanned by the vectors Ψ_u and Ψ_v). Thus, φ_g'' does not have any component on the tangent plane of the surface ($h_b = 0$), i.e., φ_g'' is aligned to the surface normal n (see Figure 2.2b). Since geodesics do not curve in the tangent space of a surface, they are the *straightest* curves on a surface. It follows that, the shortest path between two points is a geodesic, but a geodesic may not be the shortest path.

On a plane, a geodesic between two points is a straight line segment and it is unique.

However, a geodesic between two points on a surface may not be unique. This fact can be easily understood with practical examples of geodesics computed on spheres, cylinders, and cones (see Figure 2.3). On these surfaces geodesics are easily visualized and can be efficiently computed analytically. On a surface, a geodesic between two points is the straightest curve connecting them, which can be intuitively seen as a generalization of a straight line segment on that surface. Indeed, if we compute a geodesic on a cylinder(cone) and we unroll that cylinder(cone) into a plane, that geodesic becomes a straight line segment (see the three geodesics on the unrolled cone in Figure 2.3c in comparison to the real cone in Figure 2.3d). On a sphere, a geodesic between two points is a piece of the great circle passing through them (see Figure 2.3a,b). A

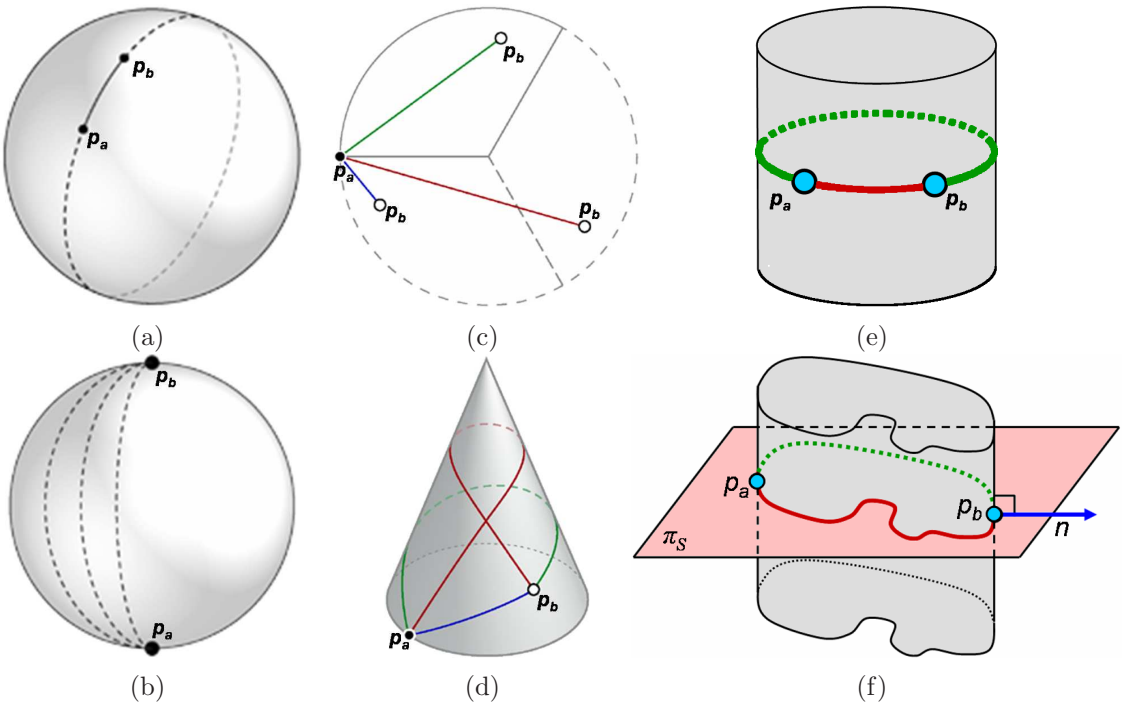


Figure 2.3: A geodesic on a surface may not be unique. On a sphere there are always at least two geodesics between two points, which are two pieces of the great circle passing by those points as shown in (a): the shortest geodesic correspond to the shortest arc of the great circle (a). If the two points are antipodal (b) (e.g., the sphere poles) the number of geodesics is infinite. On a cone, geodesics are straight lines when the cone is unrolled into a plane. In (c) a point on a cone is mapped on three different points onto a plane, after the cone is unrolled. This produces three straight lines in (c), which correspond to three geodesics in (d). Two geodesics between two points are shown on a cylinder (e) and on a general surface (f) as a result of a normal section of a surface, which always generates geodesics.

great circle of a sphere is a normal section of that sphere. Given a unit vector v lying on the tangent plane of a surface Ψ at a point p, the *normal section* of Ψ at p along v is the intersection of Ψ with the plane π containing v and the surface normal n at p (see the normal vector n in Figure 2.3f with respect to the section plane π_S). Any curve contained in a normal section of a surface is a geodesic (see Figure 2.3f).

Figure 2.3 shows different examples stressing the existence of multiple geodesics between two points on a surface. On a sphere, there are always at least two geodesics through two points p_a and p_b: the shortest arc on the great circle passing through p_a and p_b (the shortest geodesic = shortest path), and the other piece of the great circle (see Figure 2.3a). If p_a and p_b are antipodal points (like the North Pole and the South Pole), then there are infinitely many shortest geodesics between them (see Figure 2.3b). A particular example of the non-uniqueness of geodesics is depicted in Figure 2.3c,d, where given two points on a cone, there are three geodesics passing through them.

The second derivative vector of a curve represents the acceleration of a particle following that curve as trajectory. A geodesic on a surface is then interpreted as the trajectory of a particle whose acceleration is normal to the surface, i.e., whose tangential acceleration is null (see Figure 2.2b). Thus, a geodesic is the path that a particle, which is not accelerating on a surface, would follow.

The main geometric properties of geodesics are resumed as follows [155]:

- geodesics are the straightest curves on a surface;
- the shortest path between two points on a surface is a geodesic, but the contrary is not guaranteed;
- a geodesic may not be unique;
- the curvature vector of a geodesic has the same direction as the surface normal vector;
- any normal section of a surface generates a set of geodesics;
- a geodesic depends on the metric, i.e., using different metrics (e.g., Euclidean and hyperbolic in Figure 2.1) we obtain different geodesics between one point pair;

In this thesis we refer to geodesic between two points on a surface as the shortest path between those two points, i.e., the shortest geodesic (with the minimum length).

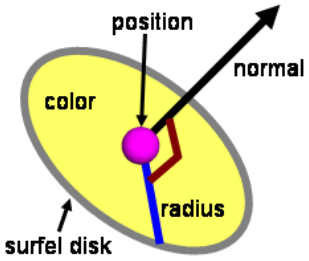


Figure 2.4: Surfel disk with its properties: position, radius, normal vector and color.

2.1.2 Definition of a surfel

We define a surfel as a point based geometric primitive, which is associated to each sample point. Geometrically, a surfel is represented as an oriented circular (or elliptical) disk in $\mathbb{R}^3$ centered onto a sample point [187]. This disk has specific properties, intended to locally describe the underlying surface. The basic geometric properties of a surfel are: the position of its center point, the radius or tangential axis of its circular or elliptical disk respectively, and finally the normal vector defining its disk orientation. For rendering purposes a surfel must have an RGB color, and depending on the particular rendering technique it may include other information (e.g., transparency, specular color, etc.). For our purposes we define a surfel s_i associated to the point p_i as the 10-tuple: $(px_i, py_i, pz_i, nx_i, ny_i, nz_i, r_i, cr_i, cg_i, cb_i)$ where the point $p_i = (px_i, py_i, pz_i)$ is the center of the surfel disk, $n_i = (nx_i, ny_i, nz_i)$ is its normal vector, r_i is the radius of the surfel disk, and $c_i = (cr_i, cg_i, cb_i)$ represents its RGB color. In our implementation of the data structure all the components of the 10-tuple are 32 bit floating point numbers apart from the RGB color that is stored in 24 bits. In order to represent the original surface without holes surfels must overlap each other [187] when rendered on a screen. In literature surfels are also referred to as *splats*, since splatting represents the main technique for point based rendering.

2.1.3 Definition of the problem

Given a surface $\Psi \subset \mathbb{R}^3$, let $\Pi = \{p_i \in \Psi, i = 1, .., N\}$ be a set of N sample points of Ψ. In this chapter we are concerned with computing approximations of geodesic paths and the corresponding geodesic distances between arbitrary point pairs (p_i, p_j) of Π on the surface Ψ.

Geodesics are defined on a surface. In our case the surface is geometrically unknown, and is given only indirectly by the set of sample points Π. Moreover, since surfaces in $\mathbb{R}^3$ are typically

acquired with 3D scanning techniques, we assume that our point set Π includes noise and outliers. Thus, the sample points p_i may not precisely lie on Ψ, but on its neighborhood. In this scenario extracting a precise approximation of the real surface may be difficult and computationally expensive. However, several methods can be used to extract geometric surfaces from point clouds. The most commonly used approaches extract implicit surface representations that locally approximates the real surface either via moving least-squares (MLS) surfaces [5, 8, 33, 70], or via radial basis functions (RBF) [46, 180, 200]. Working directly with local approximations through implicit surfaces allows a rigorous formulation of the problem of computing geodesics on the real surface. Nevertheless, this approach is not suitable for those applications that need to compute such information on the fly, e.g., realtime rendering, compression and retrieval.

Surfels have been invented to provide important visual and geometrical information to 3D points (colors, orientations, radii, as explained in Section 2.1.2) for high-quality rendering of point clouds. Such information can be extracted by analyzing the surface samples in the neighborhood of each point $p_i \in \Pi$ [3, 133, 187, 257]. Thus, a surfel set $S = \{s_i, i = 1, .., N\}$ extends the original point set Π, and can be considered to be a more refined discrete approximation of the real surface Ψ than Π. Since in this thesis we consider surfels as the principal rendering primitives, our framework for computing geodesics on point set surfaces is optimized for surfel sets. However, our techniques are suitable for all point based representations. Indeed, we also implemented them for MLS surfaces.

2.2 Related work

Different algorithms are available for computing geodesics and measuring distances on surfaces. They are typically developed for high-dimensional manifolds and differ with respect to the particular surface representation. One approach to approximate geodesics is to apply a Dijkstra-like algorithm on a graph generated by a refined representation of surfaces (e.g., polyhedron surfaces [52, 122, 123]). Kiryati and Székely in [130] presented an algorithm for geodesic distance estimation on voxel-based surfaces. Kimmel and Kiryati in [128] proposed an algorithm for finding shortest paths in two stages. In the first stage the algorithm proposed in [130] is used to obtain an approximation of the globally shortest path. In the second stage the approximation is refined to a locally optimal path. Polthier and Schnies [189] defined the discrete straightest geodesic between two points on a polyhedral surface, which may differ from a locally shortest geodesic. They proposed an application of the Runge-Kutta method on polyhedral surfaces to

compute a piecewise linear curve approximating the straightest geodesic between two points. Lee at al. [141] proposed a method to compute the straightest path between two vertices on a manifold triangle mesh with boundaries. They generated a cutting plane perpendicular to the manifold at the source point and passing to the destination point, and intersected it with the triangle mesh. Kimmel and Sethian in [197] proposed an efficient algorithm for computing geodesic distances and paths on a triangulated manifold based on a fast marching method. They extended Sethian's fast marching method [215] to triangulated surfaces. Sethian's fast marching method is a numerically consistent distance computation algorithm for solving the Eikonal equation on rectangular grids. A distance function is computed by starting from a source point and propagating outwards the front of the function in an upwind fashion. This numerical method converges to an optimal solution as the numerical grid (triangulation) is refined. The time complexity of this algorithm to compute the geodesic distance between two vertices is $O(n \log n)$, where n is the number of vertices of the triangulated manifold. This approach has been widely used due to its speed and precision. Furthermore, different efficient variations of this method were implemented [129, 176, 222]. Surazhsky et al. in [228] proposed an alternative technique to compute exact and approximate geodesics on triangle meshes. Their technique unfolds a set of adjacent triangles into a common 2D plane and extracts geodesics by reconstructing the distance field associated to each mesh edge.

Geodesics for point set surfaces have been considered before by few authors. Mémoli and Sapiro in [161] constructed an offset band of the point set surface consisting of the union of Euclidean balls centered at the given points. Geodesics were computed by applying the fast marching algorithm on a 3D Cartesian grid built inside the constructed band. The accuracy depends on the resolution of the 3D grid, and the execution time depends on the number of grid points. Their method is robust with respect to noise provided that it is bounded by the radii of the balls defining the offset band.

Klein and Zachmann [131] approximated geodesics as shortest paths on a proximity graph over point clouds. They experimented with Delaunay graphs and spheres-of-influence graphs (SIG) as proximity graphs and proposed an extended version of SIG.

Hofer and Pottmann [111] presented an efficient technique for computing geodesic curves on a point set surface by minimizing an energy functional constrained to a high dimensional manifold. Given two points on a surface Ψ and an initial curve joining the two points and lying on Ψ, the goal is to find the curve, which minimizes an energy functional. The surface Ψ is given by the MLS surface fitting the original point cloud. Their approach consists in

discretizing the initial curve into a piecewise linear path by means of a regular sampling in the arc length parameter space. The sampling produces a path of sample points such that each curve piece connecting two consecutive samples has the same arc length. The M points $\in \mathbb{R}^3$ of the resulting piecewise linear path are then considered as a unique point in $\mathbb{R}^{3M}$. By discretization the problem is reduced to minimizing a quadratic function in a constraint manifold. The constraint manifold consists of these new points and is a surface in $\mathbb{R}^{3M}$. The problem of finding an energy minimizing curve is then turned in the problem of computing the normal footpoint of the a minimizer of the energy functional in $\mathbb{R}^{3M}$ on the constraint manifold. This problem is tackled with an iterative algorithm with geometrically motivated step size control. Under certain conditions this algorithm converges to an optimal solution. Even though this method is efficient and well suited for high dimensional applications, the precision of the final geodesics is sensitive to noise, since it relies on the MLS surface construction.

The first technique proposed in this chapter to compute fast approximation of geodesics is a modification of the Klein and Zachmann's technique for surfels. In particular we proposed an extension of SIG optimized for surfels and different methods to compute shortest paths between two points on a SIG.

The second approach described in this chapter computes precise and robust approximations of geodesic. It is inspired by the method of Hofer and Pottmann [111] in the sense that it is characterized by an energy function minimization. However, we do not constrain our path on a specific surface like in [111], which is basically unknown in a point cloud, but rather we embed this constraint in the function itself, by using surfel disks (or MLS surfaces or other surface representations) as a tool to estimate the proximity of the path to the unknown surface.

2.3 Computing fast approximations of geodesics on point set surfaces

Efficient algorithms to compute geodesic distances and geodesic paths are needed in many applications and surface processing techniques. In this section a technique for a fast approximation of geodesics on point set surfaces is presented. Our technique is designed to provide fast and good estimates of geodesic distances and geodesic paths between a source sample point and n destination sample points of a surface. This problem arises in many applications where n can be arbitrarily large and even equal to the whole number of samples of a point set surface. An

additional requirement of our technique is that the whole computation time should to be feasible for real time surface processing algorithms, e.g., for compression and retrieval applications. One possible approach to this problem is to compute approximations of geodesics by applying the fast marching method on a coarse 3D grid built in an offset band of the point set surface [161]. Another possibility is to construct a weighted graph to describe point connectivity, and use Dijkstra-like algorithms to obtain a set of shortest paths on the graph. Our problem is similar to the well-known problem of finding the shortest paths between one source and all the nodes of a weighted graph. Hence, we chose the second approach, since it is also more flexible than the first approach. In fact, through an appropriate edge insertion/pruning policy a graph representation can be better adapted to the surface topology than the 3D grid proposed in [161]. From the various graphs proposed in the literature for point samples [119], we chose to adapt the spheres-of-influence graph (SIG) [131] to point set surfaces, since it accommodates variations in the point sample density and produces a good description of non-smooth surfaces [119, 131].

2.3.1 Extended sphere of influence graph (eSIG)

The sphere centered at the point sample v_i with radius given by the distance to the nearest neighbor is called the *sphere of influence* of v_i. The SIG is the graph with vertices v_i in which two vertices v_i and v_j $(i \neq j)$ are connected by an edge e_{ij} if the corresponding spheres of influence intersect. Each edge e_{ij} is weighted by the Euclidean distance between the connected vertices. An example of SIG in the two dimensional case is shown in Figure 2.5a.

As shown in Figure 2.5a, in case of noisy or irregularly sampled point clouds, a SIG may produce small clusters of connected points that are inter-cluster disconnected even though they are part of the same surface. To overcome this problem, different approaches that focus on modifying the radii of the spheres of influence can be applied. In [132] the radius of the sphere r_i is set to the distance to the k-th nearest neighbor with $k > 1$, a parameter of the method. This may produce extraneous long edges in the graph requiring an additional edge pruning step (as in Figure 2.5c). As in [132] we treat long edges as local outliers, i.e., observations which are inconsistent with the remainder of that data set. In statistics, a well know approach to discard outliers is to compute the quartiles of a data set and compute an upper threshold from the first and the third quartiles, $Q1$ and $Q3$ respectively. Quartiles are the three values that divide a set of sorted data into four equal parts [16]. For each vertex v_i of the graph the upper threshold $lmax_i$ is computed and all its edges longer than $lmax_i$ are discarded. $lmax_i$

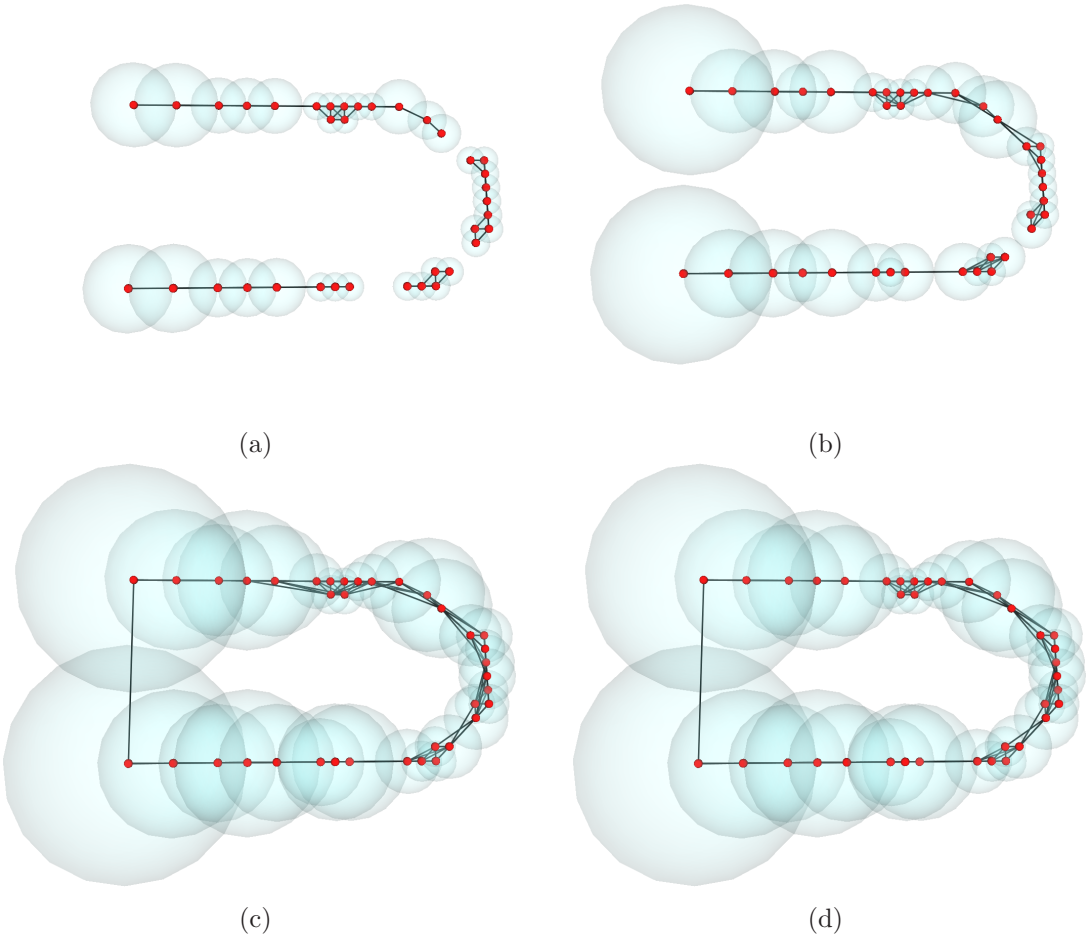
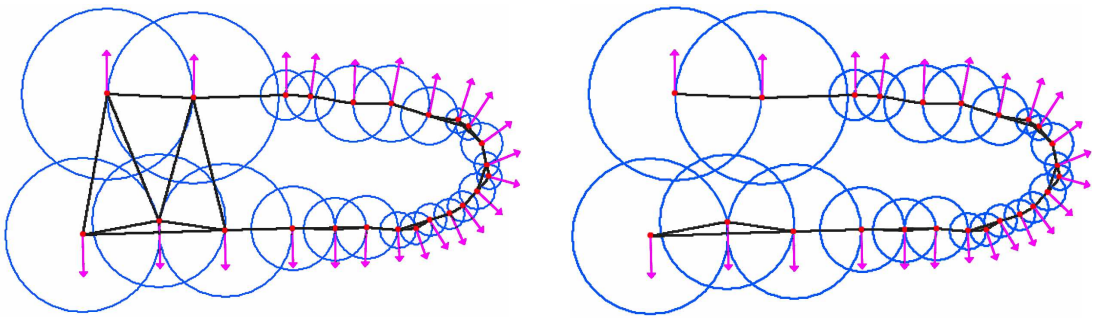


Figure 2.5: Examples of SIGs: (a) Canonical SIG, (b) 2-SIG, (c) 3-SIG, (d) 3-SIG pruned with a statistical detection of outliers based on quartiles.

is calculated as $Q3 + IQR$, where $IQR = Q3 - Q1$ is the interquartile range. Experiments showed that the pruning of edges with length greater than $Q3 + IQR$ produces good results. Even though this method performed well in our case, any other outlier detection algorithm can be used [20, 110]. Experiments also suggested to set k to 4, which on average produced better description of the geometry/topology of the sampled surfaces [132]. Figure 2.5 shows different SIGs computed with $k = 1, 2, 3$. Figure 2.5c and Figure 2.5d show a 3-SIG before and after our pruning technique, respectively. In Figure 2.5d we can notice that long edges have been deleted. However, Figure 2.5d also shows that an undesirable edge connecting the two extremes of the surface is still present. To prune such edges a statistical pruning is not enough, and information about the topology and the geometry of the underlying surface are needed.

The surfel based representation is different from plain point clouds, because surfels are oriented and have a size attribute. When a surfel representation is given, we use these attributes to extend our SIG. We set r_i to the radius of the surfel s_i. In fact, the radius of a surfel disk identifies the contribution of its sample point in representing the underlying surface. Combining this approach with the method described above we can correctly describe noisy surfaces sampled at different densities. In addition, we require that two surfels connected by an edge have approximately the same orientation. We use surfel orientations (their normal vectors) as pre-pruning criterion. We connect two vertices by an edge only if the dihedral angle between their surfel normal vectors is less than a certain threshold angle θ_t (in our implementation, θ_t is set to be slightly larger than a right angle). By using this approach we can discard undesirable edges that would otherwise not be pruned by our statistical pruning, see Fig. 2.6. In this thesis we refer a k-SIG extended with surfel information as *eSIG*.



(a) (b)

Figure 2.6: (a) Canonical SIG, and (b) SIG extended with surfel orientations (eSIG). 2D view.

A SIG constructed from surfels (eSIG) is more accurate than a SIG constructed from plain point clouds. Information about orientation of each sample point of a surface are fundamental in that respect. In those applications dealing with plain point clouds where the accuracy of a SIG is important, two strategies can be chosen. The first is to compute a surfel representation from the point clouds using the various techniques proposed in the literature [3, 29, 133, 187, 257]. The second is to compute an approximation of the surface normal at each point of the point cloud through the same techniques used for the first approach [3, 133, 187, 257]. Excluding the cost of the filtering processes required by the first approach to compute surfels color attributes, both approaches are computationally similar. In fact, the most complicated task is to precisely estimate the surface orientation at one point. However, the first approach has the advantage of allowing for a better rendering quality.

A sphere-of-influence graph can be computed efficiently in time $O(V)$ on average for uniform point sampled models with size V in any fixed dimension [78]. For irregularly sampled surfaces the computation of a sphere-of-influence graph takes $O(|V|\log|V|)$ in time [119]. Table 2.1 shows the execution time for computing our eSIG on surfel based models with different sizes. The algorithm implemented with non-optimized C++ code has been executed on an Intel Pentium 4 2.80 GHz with 2GB of RAM running under a Windows XP Professional system.

Model	# surfels	Time (sec.)
Bunny [236]	35,948	3.50
Max Planck [188]	52,809	3.90
Chameleon [188]	101,685	4.73
Igea [188]	134,345	6.08
Lion [235]	183,408	7.16
Imperia(528KT) [135]	263,908	17.25
Imperia [135]	546,137	33.07
Happy Buddha [235]	1,060,220	46.75
Dragon [235]	1,279,519	56.39

Table 2.1: Execution time for computing weighted graphs.

2.3.2 Computing shortest paths

Approximations of geodesics paths and distances between point samples (surfel positions) are computed by applying Dijkstra's algorithm on the weighted graph. This approach provides fast approximations of the geodesic distances on point/surfel based surfaces. However, these geodesic distance/path estimates are not precise, since the shortest paths on the graph computed with Dijkstra's algorithm must pass through the sample points (surfel centers). This results in an approximation error, which is shown in Table 2.2. We considered surfel based models of surfaces for which the exact geodesic distance can be computed analytically: planes, spheres, and cylinders. We then measured the relative errors between our geodesic distance estimates and the exact geodesic distances. Table 2.2 reports the root mean square of the relative errors (RMS) computed on different surfel based surfaces.

The complexity of computing the geodesic distances estimates between a given surfel and all other surfels in the model using Dijkstra's algorithm is $O(|E| + |V|\log|V|)$, where E is the set of edges [95]. Performances can be improved by using techniques based on the multi-level-bucketing data structure [95], where the average time complexity is linear for uniformly

distributed edge lengths. Furthermore, the worst case complexity is $O(|E| + |V| \log |C|)$, where C denotes the ratio between the length of the largest edge and the length of shortest edge.

Searching for the shortest path between a fixed source and a fixed destination can also be implemented more efficiently [140] than Dijkstra's algorithm. A bi-directional search and A* algorithm provide better performance especially in the average case. Fig. 2.14 shows examples of the geodesic path between selected points on surfel based surfaces computed with Dijkstra's algorithm.

Although this approach, to compute approximations of geodesics, is fast and efficient it is very sensitive to sampling. Its precision strongly depends on the quality of the sampling in the regions of the surface where a geodesic lies. Irregular and non-dense sampling induces an approximation error, while regular sampling induces metrication errors.

An alternative and interesting approach to obtain precise approximation of geodesics is to use the Delaunay graph (DG) instead of an eSIG. A Delaunay graph (DG) can be computed efficiently in $O(N)$ time in 3D for uniform point clouds [7] and allows a reconstruction of a polygonal surface over a point cloud from the Voronoi diagram (which is the dual of the Delaunay graph) also in presence of a reasonable noise level [74, 159]. Since the DG induces a neighborhood relation that also includes far neighborhoods, a statistical pruning policy should also be considered like the case of a eSIG. On the one hand a DG is more sensitive to noise and to surface sampling than a k-SIG. On the other hand a Delaunay graph implicitly defines a Delaunay triangular surface, on which accurate approximations of geodesics curves can be computed efficiently [197, 228]. Although, this approach would reduce the approximation error, it is still limited to the quality of triangulation, which is sensitive to noise and outliers. In Section 2.4 we present a method to reduce the approximation error of computing geodesics on generic representations of a point sampled surfaces via energy function minimization.

Model	# surf.	RMS error
Plane	2500	0.0584
Sphere	4098	0.0148
Cylinder	5002	0.0593

Table 2.2: RMS of the relative errors of approximated geodesic distances on a plane, a sphere, and a cylinder. The plane is characterized by a planar square sampled regularly on a grid. The model of the sphere has been obtained by regularly sampling the unit sphere. The model of the cylinder has been obtained by sampling a cylinder with 50 m diameter and 50 m height, using an equiangular sampling on it circumference and an equidistant sampling on its axis.

2.4 Energy minimizing piecewise linear geodesics on point set surfaces

In many applications computing accurate approximations of geodesics is very important, e.g., in robot motion planning and in medical/geographical applications. This section is concerned with computing accurate approximations of geodesics on point set surfaces with noise and outliers. Noisy surfaces are the typical input of many applications where surfaces are acquired through 3D scanning techniques. In this scenario extracting a precise approximation of the real surface may be difficult and computationally expensive. Although several methods to extract geometric surfaces from noisy point clouds have been presented [5, 8, 33, 46, 70, 180, 200], we are interested in a simple and general algorithm, which can be applied to different data sets and surface representations and in several applications. In this section we present a new method to compute accurate approximations of geodesics on noisy point set surface with outliers.

2.4.1 Approximating geodesics by energy minimization

Let $S = \{s_i, i = 1, .., N\}$ be the surfel set approximating a surface Ψ. Let p_a and p_b be two distinct points in S, our goal is to find a piecewise linear path $P_g = v_0 v_1 .. v_{n+1}$ with $v_0 = p_a$ and $v_{n+1} = p_b$ approximating a geodesic path from p_a to p_b, i.e., the shortest path between p_a and p_b on Ψ. The geodesic distance $g(p_a, p_b)$ between p_a and p_b on Ψ is then approximated by the length of the path P_g.

As the exact surface Ψ is not known and only approximated by the surfel set S, we propose to approximate geodesic curves by curves close to the surfels. We do not require the control points $v_i, i = 1, .., n$ to be on the surfel disks. Thus, we seek a short path that is close to the surfel set and that accommodates the geometry of the underlying surface. For this purpose we propose to minimize an energy function taking these factors into account. We define our energy function $E_S : \mathbb{R}^{3n} \rightarrow \mathbb{R}$ as a sum of three terms:

$$E_S(P) = L(P) + \alpha \cdot D_S(P) + \beta \cdot U_S(P), \tag{2.3}$$

where $P = v_1 .. v_n$ is the sequence of control points of a path between $v_0 = p_a$ and $v_{n+1} = p_b$. The term $L(P)$ is the sum of the squared distances between consecutive points and is related to the path length:

$$L(P) = \sum_{i=0}^{n} \|v_{i+1} - v_i\|_2^2. \tag{2.4}$$

This term serves to shorten the path. Moreover, considering equidistant control points v_i constrained to the ideal underlying surface, the term $L(P)$ can be assumed as the discretization of a functional whose minimization has been demonstrated to lead to a geodesic curve [190]. $D_S(P)$ represents a fidelity term and serves to keep P close to the surfel set S,

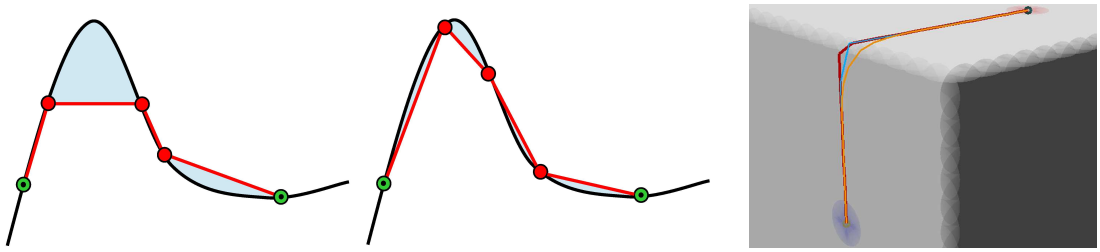
$$D_S(P) = \sum_{i=1}^{n} d_S^2(v_i) = \sum_{i=1}^{n} \min_j (d^2(v_i, s_j)), \tag{2.5}$$

where n is the number of control points v_i, and $d^2(v_i, s_j)$ indicates the squared distance between the 3D point v_i and the disk of the surfel s_j. The term $D_S(P)$ brings control points v_i close to the surfels. To additionally ensure that line segments $v_i v_{i+1}$ are close to surfels (Figure 2.7) so that the path is geometrically correct with respect to the underlying surface, we introduce the term $U_S(P)$.

$$U_S(P) = \sum_{i=0}^{n} d_S^2(\hat{v}_i) \cdot \frac{d_S^2(\hat{v}_i)}{\|v_{i+1} - v_i\|_2^2}, \tag{2.6}$$

where $\hat{v}_i = (v_{i+1} + v_i)/2$ is the midpoint between two consecutive sample points. The squared distance $d_S^2(\hat{v}_i)$ from $\hat{v}_i$ to S serves to keep the line segment $v_i v_{i+1}$ close to S. The ratio between $d_S^2(\hat{v}_i)$ and the squared length of the line segment $\|v_{i+1} - v_i\|_2^2$ puts more weight on line segments where curvature is high. $U_S(P)$ also regulates the distribution of control points v_i over the path P with respect to the path curvature (Figure 2.7a-b). $d_S(\hat{v}_i)$ roughly approximates the mean of the distance to the ideal surface Ψ over the i-th path segment (highlighted area in Figure 2.7a-b). Better estimates can be calculated. However, experiments on noisy surfaces showed that the gain in terms of precision of the whole method was small, while it increased the computational complexity significantly.

The parameters α and β influence the convergence speed of a numerical method for minimization of $E_S(P)$, as well as the quality of the resulting approximation of a geodesic. For paths passing across sharp edges the term $D_S(P)$ and especially the term $U_S(P)$ are crucial in order to achieve geodesic curves with good accuracy. As shown in Figure 2.7c the path obtained with β set to 0 (in orange) is smoother and farther from the surface than the path obtained by setting β to 2.0 (in dark-red). For surfaces with low curvature the term $U_S(P)$ can be almost neglected, since it does not strongly bias the convergence of the algorithm to a good solution. However, experiments suggested to set α and β to 0.7. Also on surfaces with sharp edges, these values have produced paths close to the true geodesic.



(a) (b) (c)

Figure 2.7: (a)(b) 2D view of two piecewise approximations of a geodesic between two points. The highlighted area represents a measure of the path proximity to the underlying surface (black curve). (c) Approximations of geodesic paths computed on a surfel based model with sharp edges: the initial path computed as shortest path on the graph using Dijkstra's algorithm (in light-blue); the path computed with our method using the $U_S(P)$ term with $\beta = 2.0$ (in dark-red) and with $\beta = 0.0$ (in orange). **[see Color plate B.3]**

2.4.2 Computing an initial path

For our algorithm it is important to select a good initial path in order to keep our minimization procedure from falling into a local suboptimal minimum, especially in case of surfaces with high curvature (see Figure 2.7c).

An approximation of the initial geodesic path between p_a and p_b is computed using the technique presented in Section 2.4.2, which computes the shortest path between two vertices of a weighted eSIG $v_a = p_a$ and $v_b = p_b$ using Dijkstra's algorithm (see light-blue path depicted in Figure 2.8). Since in this particular case we seek the shortest path between a fixed source and a fixed destination in a weighted graph, we can adopt other more efficient techniques [140]. We experimented with bi-directional search and A* algorithm that can provide better performance in the average case. However, in our case the time needed for computing the shortest path on the graph is very small with respect to the time needed by the energy minimization algorithm. Thus, by applying these techniques we did not notice relevant improvements in the execution time.

2.4.3 Energy function minimization

Our goal is to minimize the energy function $E_S : \mathbb{R}^{3n} \rightarrow \mathbb{R}$, where n is the number of control points of a path P, and $3n$ is the number their coordinates. We considered three numerical optimization methods: the downhill simplex method, the Powell method, and the conjugate gradient method [194]. All methods start from an initial path P_0 of length g_0 obtained by the

method of Section 2.4.2 and result in a path P^* of length g^*. In our experiments all the final paths were very close to the surfel set surface and their lengths were shorter than the initial path P_0. Among these three optimization methods the Polak-Ribiere variant of the conjugate gradient method [194] proved to be the fastest and the most robust with respect to the number of control points. The additional cost to compute derivatives in the minimization process is compensated by the rapid convergence to a minimum especially when the initial solution is close to it, as in our case. Another advantage of this method is the low memory usage. In fact, the Powell method and the downhill simplex method require respectively a matrix of size $(3n)^2$ and $3n(3n+1)$, while the conjugate gradient only requires a vector of size $3n$. Considering its performance, we chose the Polak-Ribiere variant of the conjugate gradient method to minimize our energy function.

The execution time of our energy minimizing method depends mainly on two factors: (1) the number of evaluations of the energy function and its gradient required to find a local minimum, and (2) the number of surfels of the model. The first factor is related to the characteristics of the minimization method, and depends on the proximity of the initial path to a local minimum of the energy function. In our case the conjugate gradient method has been optimized by choosing an adaptive bracketing criterion for Brent's line minimization [152, 194]. The second factor is related to the estimation of the distance $d_S^2(v)$ between the control points and the surfel set surface S (Section 2.4.1). This estimation requires for each control point a search of the closest surfels in the model. This search is implemented through adaptive range-queries on a kd-tree [49] constructed from the point set.

The number of control points involved in the minimization algorithm affects the convergence speed of the minimization method to a local minimum of the energy function as well as the precision of the final path with respect the true geodesic. We apply our method to the shortest path computed on the eSIG with Dijkstra's algorithm by adding one control point in the middle of those path segments that have endpoints in which the path has a curvature higher than a threshold. We also consider the distance $d_S(\hat{v}_i)$ (see Section 2.4.1). We add a control point on the path segment $v_i v_{i+1}$ if $d_S(\hat{v}_i)$ it greater than a threshold, which is statistically computed over all segments of the path. Figure 2.8 shows the result of our technique on the surfel based model of Igea [188]. The path P^* minimizing our energy function (in dark-red) properly fits the underling surface and is shorter than the initial P_0 (in light-blue).

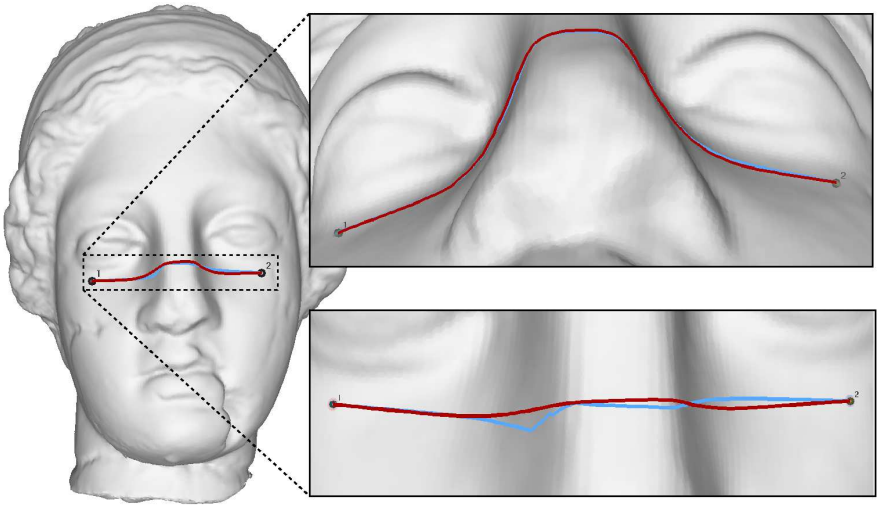


Figure 2.8: Approximations of geodesic path computed with our method (dark-red) with respect to the initial path (light-blue) computed by applying Dijkstra's algorithm on the eSIG (Section 2.4.2). Our path properly fits the underling surface (top-right) and is shorter than the initial path (bottom-right). [**see Color plate B.4**]

2.4.4 Multi-resolution approach for energy function minimization

Our method for energy function minimization possesses useful multi-resolution characteristics. In fact, it can be applied iteratively starting from a low resolution initial path. In this case, the resolution of the subsequent resulting paths can be increased at each iteration to enhance the path quality and precision (see Figure 2.9). This multi-resolution approach can be resumed by the following algorithm:

1. Compute an initial path at low resolution.
2. Apply the energy function minimization algorithm on the current path.
3. If the convergence criterion is satisfied GO TO STEP 9 (END). Two possible criteria are: (a) the approximation error with respect to the exact geodesic (if it is known) is less than a threshold or, (b) the relative improvement of the path distance to the initial path is less than a threshold.
4. Compute for each control point the path curvature $C(v_i)$.
5. If $C(v_i)$ is greater than a threshold C_t, add the midpoints $\hat{v}_i$ and $\hat{v}_{i-1}$ of the respective path segment v_iv_{i+1} and $v_{i-1}v_i$ as new control points of the path (see Section 2.4.1).

6. Compute the path segment $v_i v_{i+1}$ and the distance $d_S(\hat{v}_i)$ (see Section 2.4.1).
7. If $d_S(\hat{v}_i)$ is greater than a threshold D_t add $\hat{v}_i$ as new control points of the path.
8. Repeat - GO TO STEP 2.
9. END;

C_t and D_t are parameters of the algorithm as well as the convergence criterion. Figure 2.9 shows two steps of this algorithm.

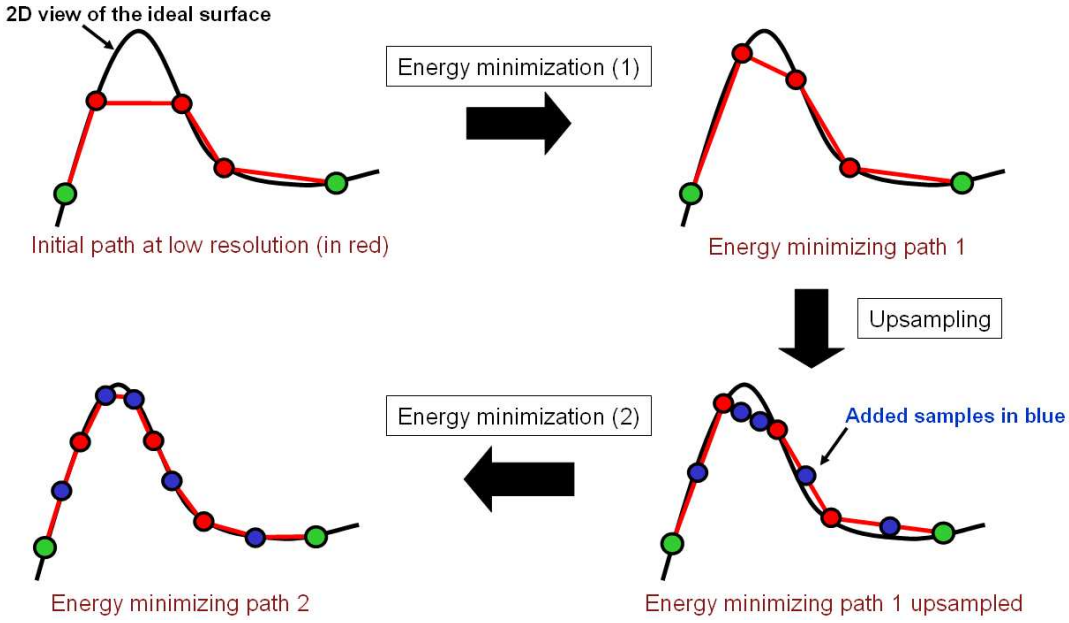


Figure 2.9: Two steps of the multi-resolution approach - 2D view. First the energy function minimization is applied to a low-resolution initial path (upper-left). The resulting path (upper-right) is up-sampled according to the path curvature and the distance of the path segments to the surface (bottom-right). The energy function minimization is applied to the up-sampled path and produces a path (bottom-left), which fits the surface better that the previous path (upper right).

With such multi-resolution approach, we can balance the convergence speed of our minimization method with the precision of the energy minimizing path with respect to the real geodesic. In fact, the first minimization step quickly converges to a local minimum due to the small number of control points of the low resolution initial path. In the subsequent steps the number of points increases according to the up-sampling criterion. However, the convergence of the minimization algorithms is fast, since at each step the minimizations start from an initial

path that is progressively closer to a global minimum. Preliminary experiments showed that this approach is convenient only to compute approximations of long geodesics on surfaces with many sample points (over 500,000 points). In other cases, and especially with our test models, the performance of the multi-resolution approach was less satisfactory than applying the energy function minimization on a high-resolution initial path. This pilot study of the multi-resolution approach is not included in this thesis, since it will be considered and developed in future works. Indeed, further developments of the multi-resolution technique will be considered together with an extension of our technique for multi-resolution curves representations, e.g., curves based on wavelets or hierarchical B-splines.

2.4.5 Results

We present experimental results of our method on surfel set surfaces. Figure 2.15 shows several approximations of geodesics on surfel based models of different sizes and geometry computed with our method. All surfel based models shown in this thesis have been rendered with PointShop3D [268].

We also consider noisy surfaces, whose noise is created by perturbing both surfel positions and surfel normals. Surfel positions are perturbed by shifting them along their surfel normal vectors. The offset of a surfel center is uniformly random in the interval $[-r_i/2, r_i/2]$ or $[-r_i, r_i]$, where r_i is the surfel radius. We identify these two cases as noise level 50% and 100%, respectively. Surfel normal vectors are perturbed by rotating them with a uniformly distributed azimuthal and polar angle. We use a maximal tolerance of 10 degrees for both angles.

Our method has been tailored for surfel based representation, but can be applied to other surface representations, e.g., triangle meshes, MLS/RBF surfaces, etc. In fact, our approach minimizes an energy function relying on estimations of the squared distances between the path sample points and the point set surface $d_S^2(v)$ (Section 2.4.1). We have implemented our technique also for MLS surfaces. The surface S is then given by the MLS surface fitted to the points cloud [5]. The squared distance $d_S^2(v)$ is estimated as the squared Euclidean distance between the control point v and its projection on the MLS surface. The paths computed with this technique on the MLS surface have approximately the same characteristics of the paths computed on surfels (Figure 2.10, 2.11). However, our technique applied to MLS surfaces is slower than our method applied on surfels, due to the large number of MLS surface estimations and projections required by the method. Optimizations of this technique will be considered in future work.

We compare these two techniques with the method presented in [111] to compute piecewise geodesic curves; we call this the *HP method*. For this HP method we take as initial path the same initial path of our methods, i.e., the shortest path computed on the eSIG with Dijkstra's algorithm. We then project the points of this path onto the MLS surface, as is required by the HP method. Figure 2.10 shows a visual comparison of four approximations of geodesic paths obtained with: (1) our method applied on surfels; (2) our method applied on MLS surfaces; (3) the HP method; and (4) the initial path from which all methods started. Both our method on MLS surfaces and the HP method have been executed with the same MLS surface fitting parameters. The same paths are computed on the same models on which we have added noise of level 100% and also outliers (Figure 2.11). Figure 2.11 shows that our methods are robust with respect to noise and outliers. The three zooms on the right show that the paths computed with our technique have not been affected by noise and outliers, while the HP method produced a jagged path when noise/outliers were present. This effect, which we also observed in other experiments, is due to the sensitivity of the MLS projection operator [268] to noise and outliers. This effect can partially be reduced with a proper selection of the MLS surface fitting parameters. Selecting proper and robust parameters is a delicate task. In fact, we noticed that they depend on the surface geometry and especially on the noise. There is no guaranty that a proper setting of these parameters can be found, especially in case of extremely noisy surfaces with outliers, as depicted in Figure 2.11. In order to overcome this problem more sophisticated techniques of MLS surface fitting may be applied [3, 8, 70, 131].

A comparison of time performances of our method with respect to our implementation of HP method is reported in Table 2.3. Both algorithms were executed on an Intel Pentium 4 2.80 GHz with 2GB of RAM running under a Windows XP Professional system. The execution of both methods was stopped as soon as the fractional difference between two consecutive paths was below to a certain threshold. The last column shows the average execution time for computing the initial path from which both algorithms started. This path is computed by applying Dijkstra's algorithm on the extended SIG (eSIG - see Section 2.4.2).

2.4.6 Experimental convergence and error analysis

For an arbitrary surfel set surface it is difficult to establish whether the path computed with our method converges to the exact geodesic, since we need to know the original surface in parametric or implicit form and compare our results with the real geodesic curve. Computing geodesic curves on a general surface is also difficult and may be imprecise, since it requires solving

Model	# surfels	Avg. # Pts	Our M.	HP	D. on eSIG
Igea	134345	74.55	19.37	6.67	0.129
Noisy Sphere	65544	68.55	3.44	1.83	0.190
Noisy Plane	10000	20.58	0.68	1.21	0.038
Noisy Cylinder	12060	37.04	7.11	2.05	0.052
ScanPlane	6580	30.81	0.69	0.57	0.013
Imperia(528K)	263908	99.46	5.70	12.66	0.101
Dragon	1279519	191.62	35.44	19.13	0.218

Table 2.3: Comparison of time performances of our method with respect to the HP method and the initial path computed on the eSIG with Dijkstra's algorithm. The table reports the average execution time taken to compute 100 approximations of geodesic paths between 100 random point pairs. Times are reported in seconds. Column 3 reports the average number of sample points of the computed geodesic paths. The noisy surfaces have a noise level of 50%.

differential equations with numerical methods [75]. However, for some surfaces geodesics can be computed analytically. On a plane a geodesic between two points is a straight line segment. On a cylinder a geodesic is also a straight line segment when the cylinder is unrolled into a plane. On a sphere a geodesic is the shortest arc on the great circle passing through two points.

We tested the convergence and the precision of our method on three surfaces which have been appropriately sampled with surfels at different densities: a plane, a sphere, and a cylinder. The plane has been obtained by regularly sampling a square planar surface on a grid. The sphere has been obtained by regularly sampling a unit sphere with 65544 surfels. We regularly subdivided an octahedron seven times using Loop's scheme [150], and we projected the vertices onto the sphere. The cylinder has been obtained by sampling a cylinder of 30 m diameter and 30 m height using an equiangular sampling on its circumference and an equidistant sampling on its axis.

In order to estimate the precision of our approach we considered two different errors. The first error indicates the precision of the approximated geodesic distance and is defined by

$$e_{geod}(g^*) = \frac{|g^* - g_a|}{g_a}, \tag{2.7}$$

i.e., the absolute relative error with respect to the exact geodesic distance g_a. We also considered the proximity of the computed path with respect to the exact geodesic path,

$$d_{path}(P, \hat{P}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_v^2(v_i, \hat{P})}, \tag{2.8}$$

where $d_v^2(v_i, \hat{P})$ is the squared distance between the control point v_i of the path P and the piecewise linear path $\hat{P}$. The second error is then defined by

$$e_{path}(P^*) = \frac{max(d_{path}(P^*, P_a), d_{path}(P_a, P^*))}{g_a}, \tag{2.9}$$

i.e., the symmetric distance between the approximated geodesic path P^* and the exact geodesic path P_a divided by the length of P_a. In our case P_a is a continuous path, which is properly sampled at high density. This allows to compute $d_{path}(P_a, P^*)$ and to render P_a properly.

We statistically compared these two errors by considering the root mean square of the errors $e_{geod}(g^*)$ (EG) and $e_{path}(P^*)$ (EP). Table 2.4 reports the results of our methods and the HP method on the three test surfel set surfaces described above. The last two columns also show the approximation errors of the initial path of both methods computed with Dijkstra's algorithm on the eSIG. On all these surfaces, our method converged to piecewise linear paths having small EG and EP errors, and that were very close to the exact geodesics. Furthermore, we observed that the MLS projection operator used in the HP method keeps the paths closer to the real surface than our method.

The convergence of our method to a good approximation of the exact geodesics is biased by the sampling of the test surfaces. Surfels linearly approximate a surface. A correct sampling is important to keep the approximation error under a certain threshold, especially for surfaces with high curvature. On the plane our method converges to paths that are closer to the exact geodesics than the paths computed on the sphere and the cylinder (e.g., see Table 2.4). On the sphere and the cylinder the errors $e_{geod}(g^*)$ and $e_{path}(P^*)$ decrease by increasing the sampling density, in case both the sphere and the cylinder are regularly or uniformly sampled. Thus, for numerical comparison, we experimented with five different surfel based approximations of the unit sphere, and compute the errors EG and EP, which are reported in Table 2.5. The five approximation levels of the unit sphere were obtained by regularly subdividing an octahedron using Loop's scheme [150] with vertices projected onto the sphere. We then associated a surfel to each of these vertices. The coarsest model obtained through 3 subdivision levels has 258 surfels. The finest model obtained through 7 subdivision levels has 65538 surfels. Figure 2.12 shows the surfel based models of three unit spheres obtained through different subdivision levels. The path computed with our approach is shown in comparison to the exact geodesic and the initial path.

		Our Met.		HP		Dijkstra on eSIG	
Model	# Surf.	EG%	EP%	EG%	EP%	EG%	EP%
Sphere	65544	2.3e-2	3.1e-2	4.0e-2	1.0e-2	1.66	0.53
Plane	10000	1.6e-5	5.6e-4	2.7e-5	2.7e-7	6.00	1.07
Cylinder	12060	7.3e-3	1.6e-3	4.6e-1	9.0e-2	6.12	1.00

Table 2.4: Comparing the approximation error of our method with the HP method and Dijkstra's algorithm (on the eSIG) on different surfel set surfaces. Root mean square of the errors $e_{geod}(g^*)$ and $e_{path}(P^*)$ are reported in percentage as EG% and EP%, respectively. The approximation error depends on the sampling of the surface (see Table 2.5).

		Our Met.		Dijkstra on eSIG	
# Subdiv. lev.	# surfels	EG%	EP%	EG%	EP%
3	258	0.873	0.509	1.163	2.024
4	1026	0.102	0.322	1.282	1.310
5	4098	0.081	0.054	1.530	0.928
6	16386	0.032	0.052	1.673	0.720
7	65538	0.023	0.031	1.661	0.534

Table 2.5: Approximation error of our method on surfel based models of unit spheres with different samplings. Spheres are sampled regularly using the Loop's scheme at different subdivision levels. Root mean square of the errors $e_{geod}(g^*)$ and $e_{path}(P^*)$ are reported in percentage as EG% and EP%, respectively. The last two columns report the approximation error of the initial path of our method computed with Dijkstra's algorithm on the eSIG.

2.4.7 Robustness with respect to noise

We present experimental results of our method on noisy planes, noisy spheres, and noisy cylinders. The noisy surfaces are created by adding random noise of level 50% or 100% (see Section 2.4.5) to the test surfaces used in the experiments reported in Table 2.4. We also considered a planar surface scanned with a 3D Scanner Minolta VI-900, which presents the typical noise given by laser light-stripe triangulation range-finders.

We compared the precision of our method with respect to the HP method, by computing the errors EG and EP for both methods. Table 2.6 shows the result of our comparison. Figure 2.13 shows the approximation of a geodesic path computed with our method on three noisy surfaces, together with the exact geodesic curve, and the initial path computed on the eSIG. Our method demonstrated a better robustness with respect to noise than the HP method. With our method we obtained non-jagged paths very close to the exact geodesics. However, on both noisy spheres our method revealed an EP error that is greater than the one in the HP method, even though

Model	NL	# Surf.	Our Met. EG%	Our Met. EP%	HP EG%	HP EP%	Dijkstra on eSIG EG%	Dijkstra on eSIG EP%
Sphere	50	65544	0.36	0.27	8.50	0.10	3.26	0.34
Sphere	100	65544	0.88	0.31	22.28	0.15	3.70	0.52
Plane	50	10000	0.019	0.097	0.12	0.24	0.87	0.51
Plane	100	10000	0.023	0.11	0.49	0.48	1.40	0.65
Cylind.	50	12060	0.69	0.46	2.50	0.94	5.56	6.02
Cylind.	100	12060	1.04	0.71	16.38	3.70	1.89	1.29
ScanPlane		6580	0.15	0.25	0.40	0.33	5.62	0.78

Table 2.6: Comparing the approximation error of our method with the HP method and Dijkstra's algorithm (on eSIG) on noisy surfaces. Root mean square of the errors $e_{geod}(g^*)$(EG) and $e_{path}(P^*)$(EP) are reported in percentage as EG% and EP%, respectively. The second column reports the noise level in percentage. Results on the original surfel based models without noise are reported in Table 2.4. ScanPlane is a planar surface scanned with a 3D Scanner Minolta VI-900.

the EG error of the HP method is less significant (Table 2.6). The reason is that the HP method generated jagged paths closer to the exact geodesic curves than the paths computed with our method, even though their lengths were greater than the exact geodesics, and those of our paths.

The EP error in our method depends on the proximity of the initial path to the exact geodesic. In this sense, the quality of the initial paths can be slightly improved by increasing the number of edges of the eSIG through the parameter k (see Section 2.4.2). Other approaches can also be used, e.g., the fast marching method applied on a coarse 3D grid built in an offset band of the point set surface, presented by Mémoli and Sapiro in [161].

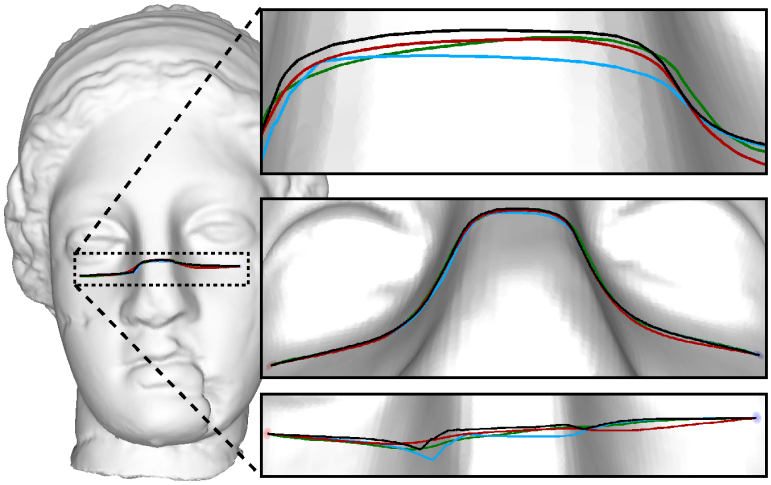


Figure 2.10: Comparison of different approximations of geodesic paths computed on the surfel based model of the Igea [188] with different methods: initial path computed as shortest path on the graph with Dijkstra's algorithm (light-blue), path computed with our method on surfels (dark-red) and on MLS surface (dark-green), and path obtained with the HP method (black). **[see Color plate B.5]**

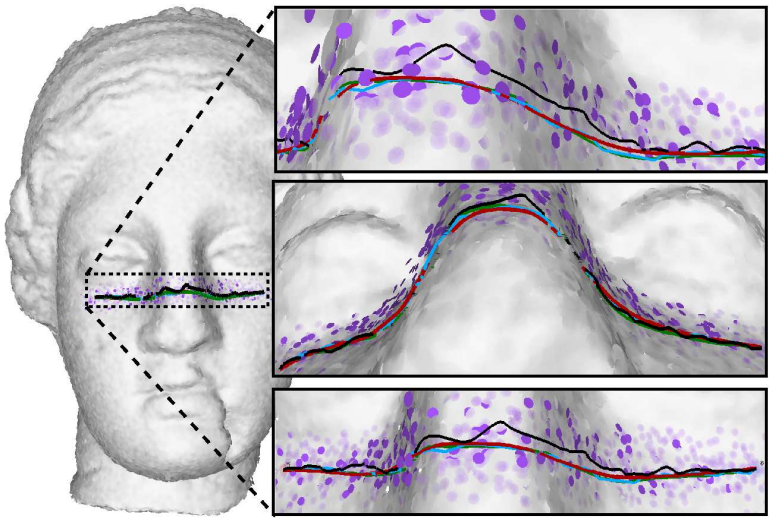
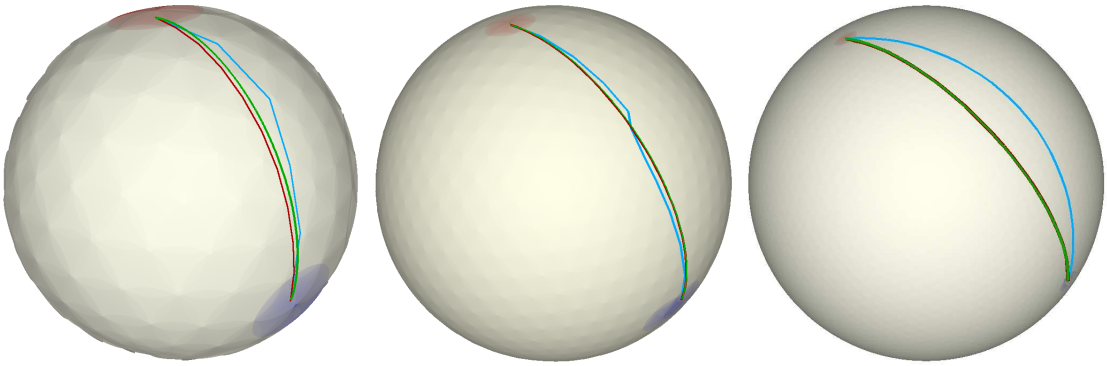
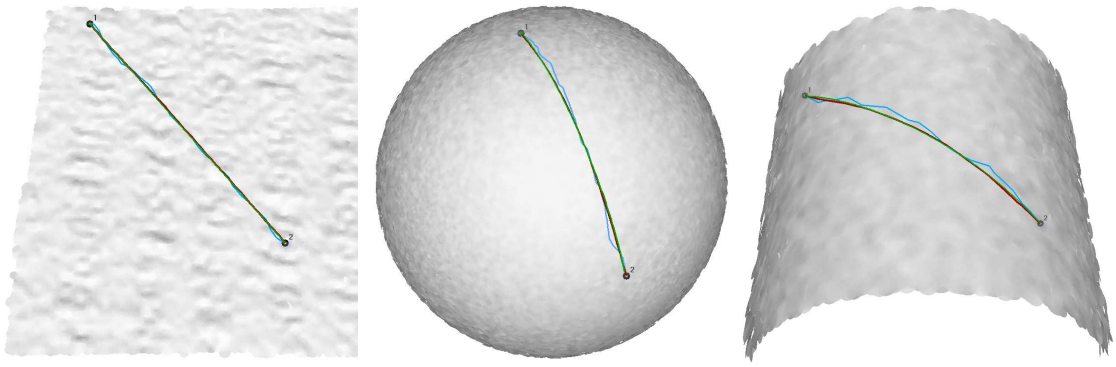


Figure 2.11: Comparison of different approximations of geodesic paths computed on a noisy surfel based model with outliers. Noise of level 100% and outliers (surfels in violet) have been added to the surfel based model of the Igea [188]. Outliers are generated by adding new noisy surfels with noise level 200% to the noisy model. Paths: initial path computed as shortest path on the graph with Dijkstra's algorithm (light-blue), path computed with our method on surfels (dark-red) and on MLS surface (dark-green), and path obtained with the HP method (black). **[see Color plate B.6]**



(a) (b) (c)

Figure 2.12: (a) Result of our method on surfel based models of unit spheres obtained from different subdivision levels using the Loop's scheme (3-(a), 4-(b), 5-(c)). Exact geodesic path computed analytically (green), Dijkstra's shortest path computed on the graph (light-blue), and geodesic path approximation computed with our method (dark-red). [**see Color plate B.7**]



(a) (b) (c)

Figure 2.13: Comparing the exact geodesic path (green) with the geodesic path approximation computed with our method (dark-red), and the initial solution computed on the graph with Dijkstra's algorithm (light-blue) on different noisy surfaces: a plane (a) a sphere (b), and a cylinder (c). [**see Color plate B.8**]

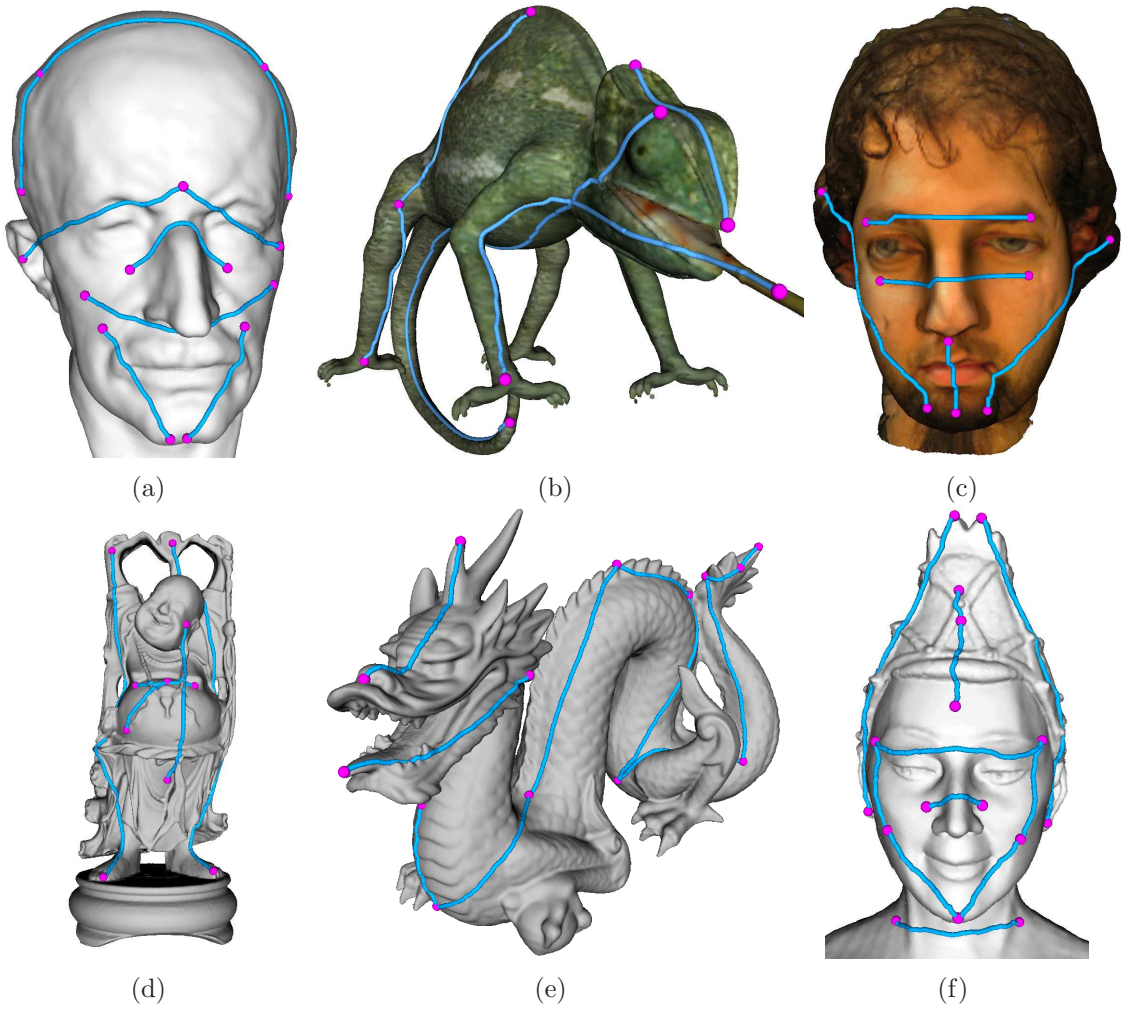


Figure 2.14: (a) Approximations of geodesics on surfel based models computed applying Dijkstra's algorithm on the associated eSIGs: Max Planck [188] 52,809 surfels (a); Chamaleon [188] 101,685 surfels (b); Textured Igea [188] 134,345 surfels (c); Zoom of Imperia [135] 546,137 surfels (d); Happy Buddha [235] 1,060,220 surfels (e); Dragon [235] 1,279,519 surfels (f).

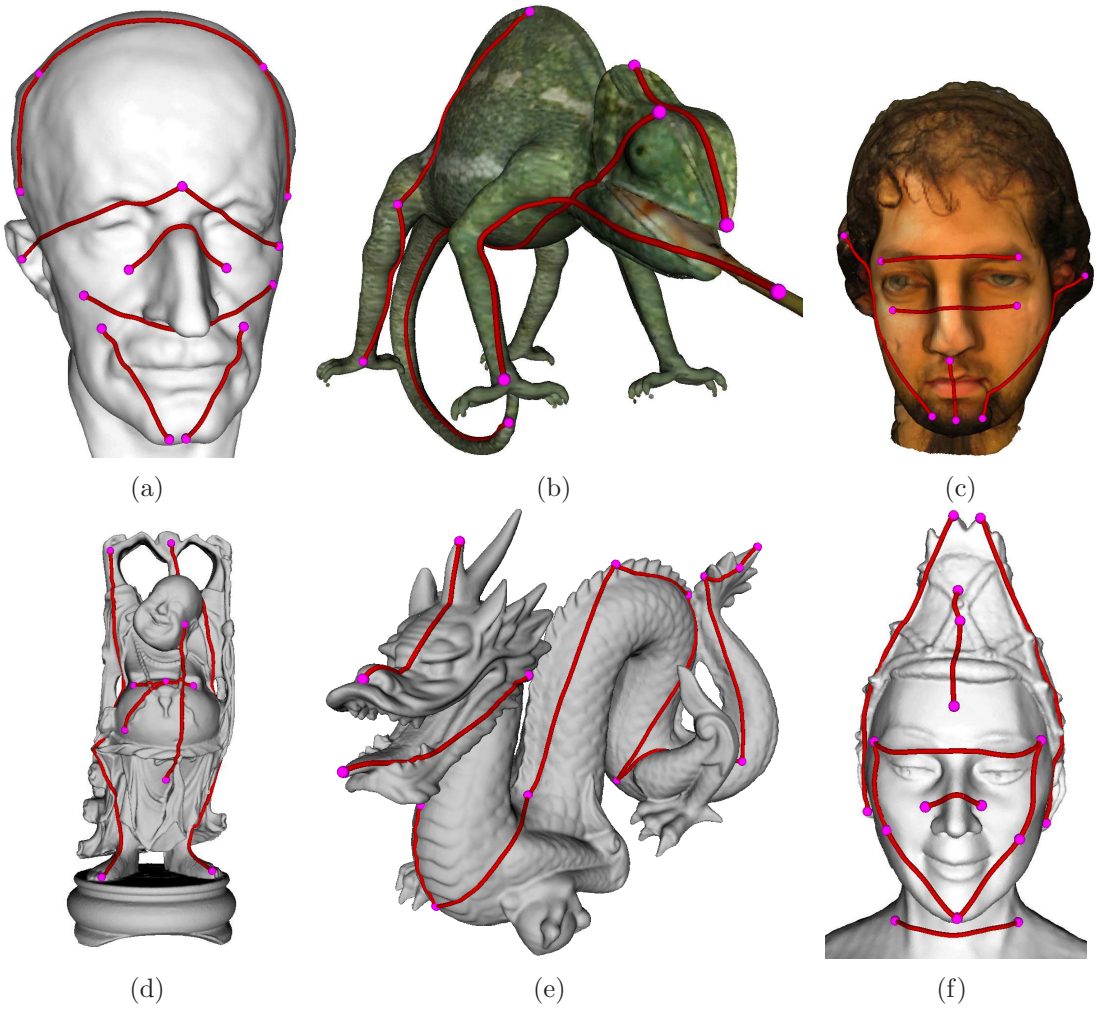


Figure 2.15: (a) Approximations of geodesics computed with our method on surfel based models: Max Planck [188] 52,809 surfels (a); Chamaleon [188] 101,685 surfels (b); Textured Igea [188] 134,345 surfels (c); Zoom of Imperia [135] 546,137 surfels (d); Happy Buddha [235] 1,060,220 surfels (e); Dragon [235] 1,279,519 surfels (f).

Chapter 3

Geodesics for compressing textured point based models

In this chapter we describe a framework that uses geodesics for generating a spherical representation and compressing textured point set surfaces. This framework, presented by Darom et al. in [66, 67], relies on a computationally efficient procedure for geodesic-distance preserving spherical multi-dimensional scaling (MDS), which is described in section 3.2. The compression scheme is described in section 3.3. Geodesic distance estimation on surfaces represented by surfels is considered in section 3.2.1. Experimental results are presented in section 3.4.

3.1 Introduction

Textured three dimensional models are of common use in video games, television, movie studios, virtual shops, computer-aided design, telemedicine and medical image databases. Textured 3D models can be acquired from reality or generated by techniques for 3D modelling and computer-aided design [44, 81]. In the first case they are acquired through 3D scanning techniques [144], which use laser scanners and/or structured light range finders [207, 208]. Other techniques compute 3D textured models from multiple images [68, 172] and videos [169], by using epipolar geometry [90], photogrammetry [82] and algorithms from computer vision (e.g., shape from shading [191, 264]) and computational stereo [15, 37]. Textured models are also obtained by segmentation of 3D medical images, acquired using CT, MRI or other imaging modalities [1, 151].

Real time transmission and compact storage of detailed 3D models require compression of textured 3D models. In video games, large relevant part of virtual worlds stored on servers need

to be transmitted to the player side for visualization. Telemedicine applications and efficient indexing in large 3D medical image databases also call for convenient representation of 3D models.

Point based models are gaining popularity in several fields [97, 133] since are the natural output format for important classes of 3D scanners (laser and structured light, shape from shading and texture, photometric stereo, tactile). Improvements in 3D scanning technologies led to huge point based models of the order of millions of 3D points that need to be compressed for storage space saving and transmission. This chapter presents a compression scheme for both geometry and texture of surfaces represented by surfel sets (Section 2.1.2). It relies on a spherical parametrization of surfel set surfaces where geodesics play an important role to minimize the distortion of the parametrization.

For triangle-mesh surfaces, compression of both geometry and texture was considered in [102]. Geometry compression schemes applicable to surfel based surfaces were presented by Fleishman *et al* [91] and by Ochotta and Saupe [177]. The second method consists of division of the surfaces into patches and compression of the patch height field above a suitable reference plane. Compression of both the geometry and texture of point set surfaces was presented by Waschbüsch *et al* [254]. Their compression scheme is based on multiresolution predictive encoding, which separately encodes geometry and color attributes of a point set surface.

The lossy compression scheme presented in this chapter is designed for textured surfaces represented by surfel sets. This algorithm relies on the observation that in many applications, notably in the important task of 3D face representation, the surface can be mapped to the sphere as a whole. This is convenient for indexing in surface databases, and naturally leads to invariance to geometric transformations.

Lossy surface compression schemes introduce a geometry error. The geometry error induced by the presented compression algorithm is measured using an extension for point set surfaces of the well known approach introduced by the Metro tool [54] . Such an error measurement procedure, introduced by Pauly et al. [184] and used by Ochotta and Saupe [177], measures the geometry error via the Moving Least Squares surface (MLS surface) [5] representation of the surfel based surfaces.

Spherical mapping, i.e., the mapping of a given surface onto a sphere, is a major building block in the suggested surface representation and compression scheme. Spherical mapping has various statistical interpretations and applications [63], such as: face recognition [36], and texture mapping [84]. In general, a spherical mapping takes as input a matrix quantifying the

dissimilarity between points in the data set. In geometric surface analysis applications [36, 66, 84] dissimilarity is measured by geodesic distance. Spherical parametrization distortion is reduced when geodesic distances are preserved. Thus, a good spherical mapping algorithm should preserve geodesic distances, as much as possible.

Cox and Cox in [63] presented the spherical multi dimensional scaling (Spherical MDS) algorithm, which was designed for optimal preservation of dissimilarity values by *Euclidean* distances between points on the sphere.

Multi dimensional scaling (MDS) is a set of data analysis techniques that given a set of observations in a n-dimensional space detect meaningful underlying dimensions, which allow to study observed similarities or dissimilarities (distances) in the data [30]. The main idea is to map measurements of similarity or dissimilarity among pairs of observations into distances between feature points with given coordinates in a small-dimensional Euclidean space. This approach is often used in data visualization to explore dis/similarities and geometric structures in the data. In the case of [63] the final target space of the MDS is a 3D sphere.

Elad and Kimmel [86] extended the algorithm of Cox and Cox [63] for optimal preservation of the geodesic distances between points on an input surface by considering distances between the respective points on the sphere. Since in practical applications the dimension of the input dissimilarity matrix is limited by memory requirements [63, 86], a surface must be down-sampled prior to the spherical MDS procedure. Thus, a spherical mapping of the full-resolution input surface can only be accomplished by interpolation. This implies that the loss induced by the down-sampling cannot be properly recovered.

For an high quality representation and compression of textured surfaces via spherical mapping an accurate mapping of the input surface elements at the full resolution is needed. The algorithms presented in [63, 86] are inadequate for this purpose. In Section 3.2 we describe an algorithm, which tackles the problem of dense spherical mapping through a computationally feasible solution. A fundamental part of the suggested approach consists of fast computation of geodesic distances (see Section 3.2.1). Estimates of geodesic distances on surfel based surfaces are computed with the approach presented in Section 2.3.

3.2 Spherical mapping

Spherical mapping is the process of mapping a set of points on a surface to a set of points on a sphere. It is used in several applications to parameterize a surface in a spherical domain.

This allows to apply several signal processing and numerical algorithms, once a surface on the spherical domain is resampled on a regular grids. Since many applications often deal with geometric models represented by closed and genus-zero surfaces, the sphere is the most natural and convenient domain where to map these surfaces. Unlike planar parametrization, which were extensively studied for texture mapping an surface flattening, a spherical parametrization does not require a surface to be cut into charts or disks. However, for a highly deformed surface a spherical parametrization maybe be difficult to compute. Praun and Hoppe [193] mapped vertices of a triangle mesh sequentially, minimizing a local stretch measure. Global relaxation steps at preset intervals complement the process. This method was later used as part of a compression scheme for surfaces represented as triangle meshes [112]. Zwicker and Gotsman [267] considered spherical mapping of a point set by constructing a k-nearest neighbor graph and iteratively reducing the corresponding Laplace-Beltrami operator on the sphere. The technique described in this section considers a generalization of the spherical mapping algorithm that was originally developed by Cox and Cox [63], and later extended and applied to surface analysis by Elad and Kimmel [86].

Cox and Cox [63] define a stress functional for a spherical mapping $\chi : \Pi \rightarrow \Sigma$

$$S_C = \left[\frac{\sum_{j=1}^{N} \sum_{i=1}^{N} (d_{ij} - \delta_{ij})^2}{\sum_{j=1}^{N} \sum_{i=1}^{n} \delta_{ij}^2} \right]^{\frac{1}{2}} \tag{3.1}$$

where $\Pi = \{p_1, .., p_N\}$ is a set of N points in an Euclidean space of fixed dimension, Σ is a sphere $\in \mathbb{R}^3$, δ_{ij} is the dissimilarity between two points $p_i, p_j \in \Pi$, d_{ij} is the Euclidean distance between their mappings onto the sphere, i.e., $\chi(p_i)$ and $\chi(p_j)$ respectively. The stress functional S_C measures the quality of the match between distances on a spherical domain and dissimilarities in the Euclidean space. Following earlier work on surface flattening via multidimensional scaling [99, 265], Elad and Kimmel [86] used (normalized) geodesic distance on the input surface as the dissimilarity measure (δ_{ij}). They redefined d_{ij} as the (normalized) arc length on the destination sphere. The functional thus quantifies the discrepancy between corresponding geodesic distances on the input surface and on the sphere. The method described in this section follows the same approach of Elad and Kimmel [86], and considers a surfel set as input surface. Hence, in our case the set of data points Π is related to a surfel set, as the points p_i and p_j correspond to the centers of the surfels s_i and s_j respectively (see Section 2.1.2).

Computing and minimizing this functional requires $\Theta(N^2)$ storage of the dissimilarity matrix. Practical surfaces may contain millions of points. Processing of such surfaces requires a

pre-processing step of surface simplification to at most a few thousands of points. In the context of surfaces represented by triangle meshes, Elad and Kimmel [86] minimize the functional using the conjugate gradient algorithm.

Surface simplification by sub-sampling (decimation) is incompatible with the preservation of details. *Dense spherical mapping*, i.e., the mapping of *all* the surface points onto the sphere, without interpolation, is therefore necessary. For surface flattening onto the plane, Bankirer and Kiryati [14] suggested relying on the dissimilarity between a small set of n reference points to all N points. This is accomplished by measuring the geodesic distance from each reference point to all points on the surface at the full resolution. The resulting non-square $n \times N$ dissimilarity matrix is used as input to a modified classical scaling algorithm [14] (compare with [30]). Thus, with this approach the mapping of the $N - n$ non-reference points is not computed by interpolation of the mapping of the n reference points, but it is determined considering their geodesic distances to all reference points. These distances are obtained without additional computational cost, because the geodesic distances from each reference point to the non-reference points are calculated anyway while computing the geodesic distances from the reference point to the other reference points.

The concept of [14] is adapted to the spherical domain as follows. A subset of n points (surfels centers) are evenly selected out of all the N points by using the farthest point strategy [89]. The resulting $n \times N$ dissimilarity matrix is used to define the following modified stress functional:

$$S_D = \frac{\sum_{i=1}^{n} \sum_{j=1}^{N} (d_{ij} - \delta_{ij})^2}{\sum_{i=1}^{n} \sum_{j=1}^{N} \delta_{ij}^2} \tag{3.2}$$

The functional is minimized in five steps:

1. Given the $n << N$ reference points (surfels), they are mapped to the sphere by minimizing functional (3.1) using the conjugate gradients algorithm [195].

2. The mapping of all N points is obtained by interpolation [93], and used to initialize the next step.

3. The functional (3.2) is minimized by moving only non-reference points, using the conjugate gradients method.

4. A relaxation step takes place, in which the functional is minimized using the conjugate gradients method.

5. The solution is rotated by 90^o along the meridian, and the functional is minimized again, without moving points that are close to the sphere poles.

Steps 3-4, and the minimization in step 5, can be iterated. The fifth stage is necessary, because the numerical solution is unstable near the poles. In principle, this problem can be traced to [63] and [86], but is more noticeable here, because of the density of the points near the poles. By minimizing this functional, each point (surfel) is mapped onto the sphere, eliminating the need for surface simplification, sub-sampling or interpolation.

3.2.1 Computing geodesic paths on surfel based surfaces

As previously discussed in section 3.2, estimates of geodesic distances are needed in our method in order to reduce the parametrization distortion. This procedure requires the computation of $n \cdot N$ geodesic distances, where n is the number of reference surfels and N is the number of surfels. Thus, we need a fast and efficient algorithm to estimate geodesics between surfels over all the surfel based model that have to be preserved in the spherical parametrization. Considering the execution time requirements of our compression scheme we adopt the technique presented in Section 2.3, specifically tailored for surfel sets. An eSIG is computed from the surfel set and the geodesic distance estimates between surfel pairs selected by the parametrization algorithm are computed by applying Dijkstra‘s algorithm on the eSIG.

As mentioned in Section 2.3 this approach is fast but its precision is strongly biased by the surface sampling, since the shortest paths on the graph computed with Dijkstra's algorithm must pass through surfel centers. To improve the precision of the geodesics's estimates one can use our technique based on function minimization (see Section 2.4). However, considering its execution time and the big number of estimates that we need to compute to parameterize a surfel based model on a sphere, this technique is feasible only for models with a small number of surfels, which actually do not need compression. A compromise between precision and algorithm's speed may be found by reducing the convergence threshold of our minimization technique. An alternative approach is to run few iterations of the Hofer and Pottmann method [111] to slightly improve the precision of the shortest path computed with Dijkstra's algorithm. We also tried an hybrid algorithm, which runs few iterations of the Hofer and Pottmann method [111] before applying our minimization technique with a high convergence threshold. This approach moves the initial path closer to a local minimum of the function to minimize (see Section 2.4) and the minimization procedure rapidly converges to a local minimum after few iterations. However,

all the suggested approaches to slightly improve the precision of the geodesics's estimates did not produce visible improvements of the compression quality in terms of geometric distortion induces by the whole compression procedure.

Improvements of the execution time of the graph based algorithm can be instead achieved with more sophisticated techniques. The complexity of computing the geodesic distance estimates between a given surfel and all other surfels in the model using Dijkstra's algorithm is $O(|E| + |V| \log |V|)$, where V and E are the sets of surfels (vertices) and the set of edges, respectively [95]. This procedure must be executed n times, where n is the number of reference surfels. Performances can be improved by using techniques based on the multi-level-bucketing data structure [95], where the average time complexity is linear for uniformly distributed edge lengths. Furthermore, the worst case complexity is $O(|E| + |V| \log |C|)$, where C denotes the ratio between the length of the largest edge and the length of shortest edge.

3.2.2 Performance evaluation of the spherical mapping

In order to compare and visualize the resulting mapping error, a mapping error is defined for each surfel:

$$\epsilon_i = \left[\sum_{\forall s_j \in S_r} \frac{(d_{ij} - \delta_{ij})^2}{{\delta_{ij}}^2} \right]^{\frac{1}{2}} \tag{3.3}$$

where S_r is a set of 100 randomly selected surfels.

Table 3.1 shows the RMS values of the surfel mapping errors obtained using the dense spherical mapping method, for typical numbers of reference points n. Table 3.2 is provided as reference, and presents the corresponding performance of the algorithm of [86] followed by interpolation. The number n of reference points needed to achieve a certain error level using the suggested method is smaller by about two orders of magnitude with respect to the method of [86]. The respective color mappings of the error are shown in figures 3.1 and 3.2. The mapping accuracy gain made possible by the proposed method is evident.

reference surfels (n)	RMS mapping error
10	0.1334
50	0.1326
100	0.1311

Table 3.1: Mapping quality with the suggested dense spherical mapping method

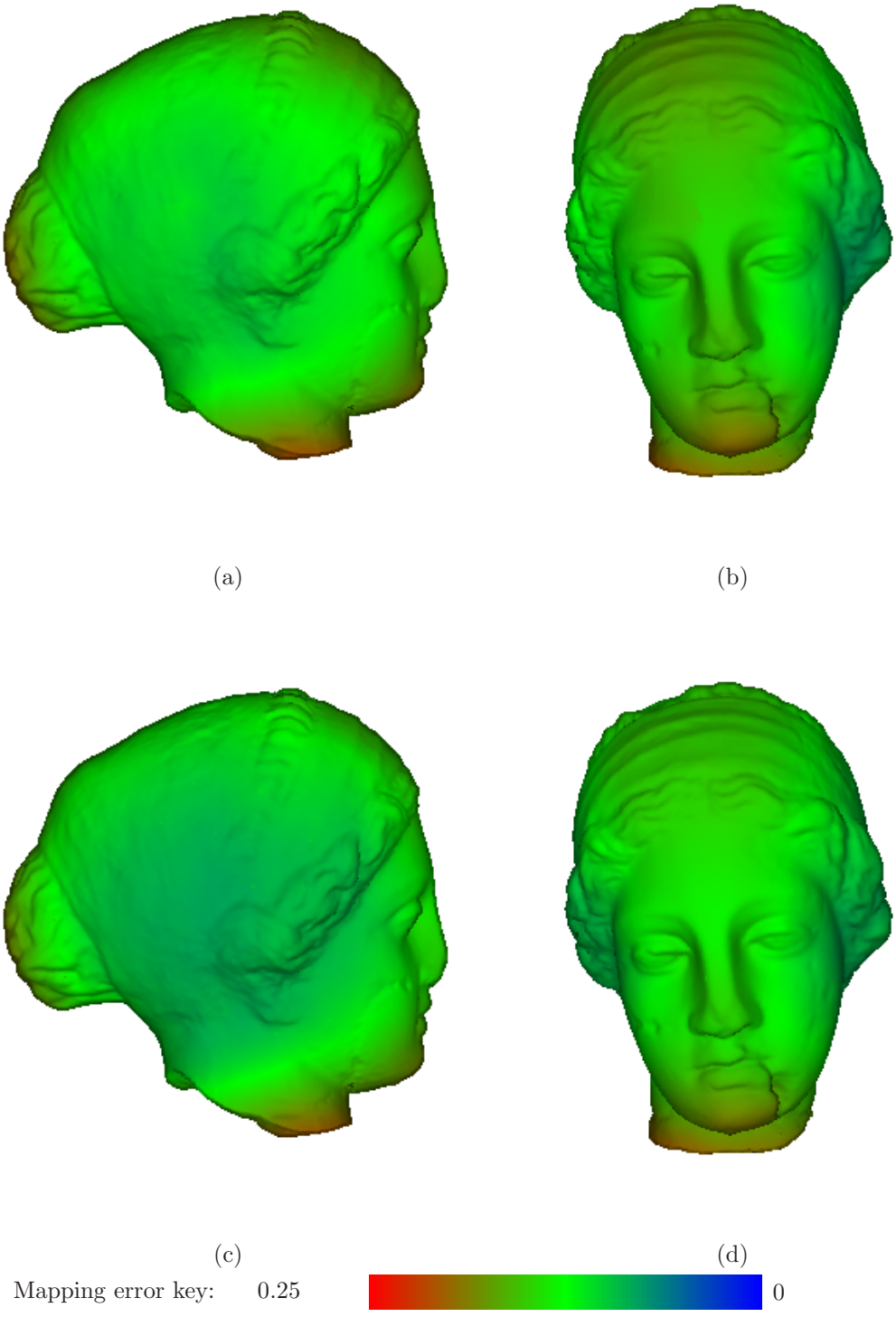


Figure 3.1: Dense spherical MDS: The mapping error of each surfel in the Igea model [188] is color-coded. (a), (b): Using 10 reference surfels. (c), (d): Using 150 reference surfels. **[see Color plate C.2]**

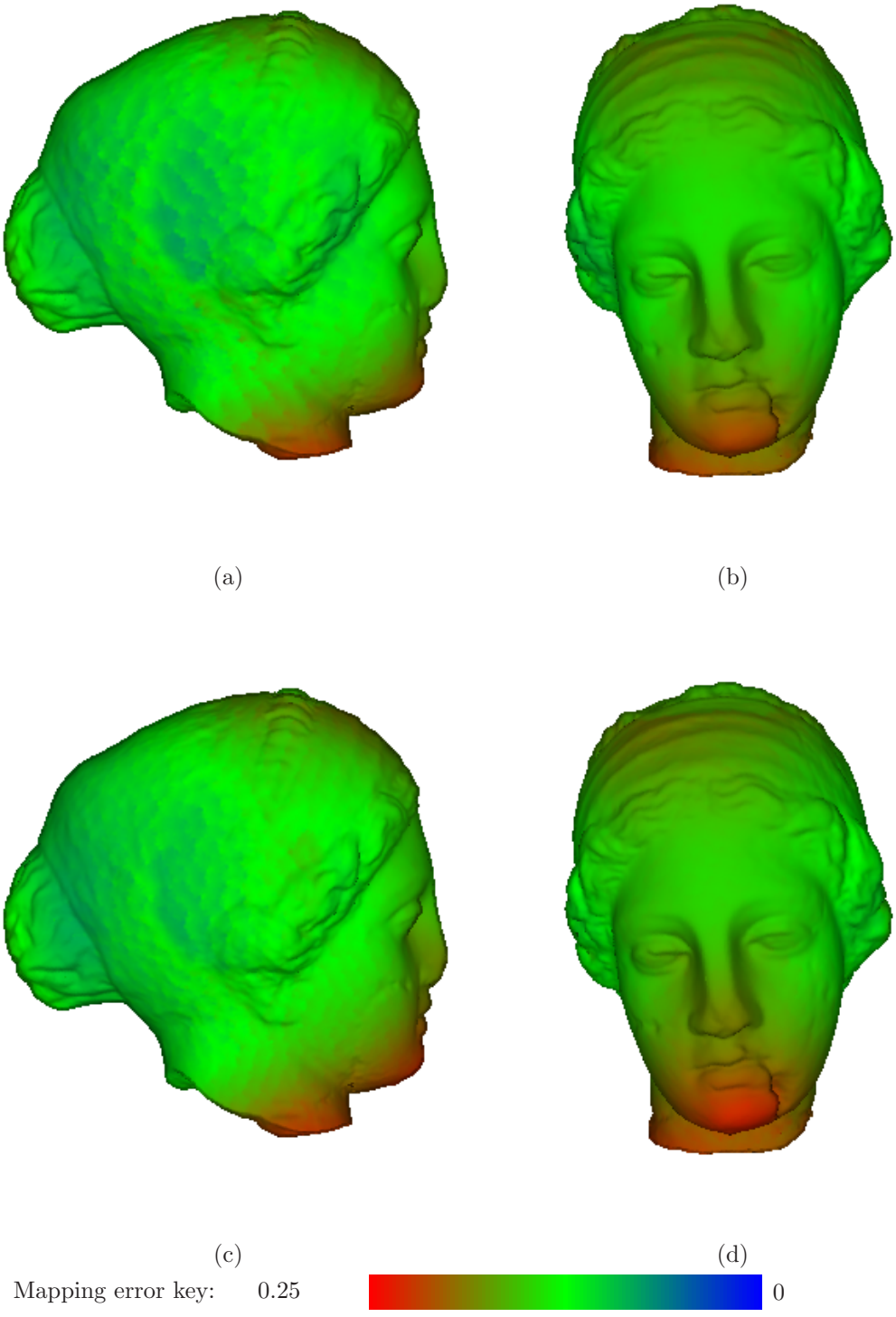


Figure 3.2: Constrained spherical MDS [86] followed by interpolation. The mapping error of each surfel in the Igea model [188] is color-coded. (a), (b): Using 1000 reference surfels. (c), (d): Using 2000 reference surfels. **[see Color plate C.3]**

3.3 Textured surface compression

The compression of surfaces that are represented by triangle meshes has received significant attention, see e.g. [102, 127, 239]. In some cases, the preservation of the mesh connectivity is an important consideration [239]. In others, remeshing takes place as part of the compression-reconstruction cycle [102]. In surfel based surface representation there is no mesh, hence there is no connectivity that needs to be preserved. Nevertheless, in analogy to the remeshing notion, the surface can be "resurfeled" as part of the process, meaning the approximate representation of the surface by a new surfel set. The technique described in this section focuses on lossy compression, that maintains good visual quality at high compression ratios.

3.3.1 Representation and compression scheme

In the suggested method for representation and compression of textured surfel based surfaces, the underlying continuous surface is optimally mapped onto the unit sphere, as described in section 3.2. The mapping of each surfel includes its spatial coordinates and color components as a vector. Additional attributes, such as normal directions and surface reflectance properties can also be mapped. On the sphere, the mapped vectors are regarded as samples of a continuous vector valued (multilayered) function. This function is represented by spherical wavelets [211, 229]. The wavelet coefficients are quantized, encoded and interleaved. The resulting bitstream is then losslessly compressed using an entropy encoder.

In the reconstruction stage, resurfeling takes place: a surfel is created for each spherical grid point. Its position and color attributes are obtained from the coded data. In the examples shown in this chapter, the orientation and radius are derived from the spatial position using neighborhood considerations. The proposed compression pipeline is illustrated in Figure 3.3. A compression result is shown in Figure 3.4.

reference surfels (n)	RMS mapping error
1000	0.1459
2000	0.1453
5000	0.1444

Table 3.2: Mapping quality using constrained spherical MDS [86] with interpolation

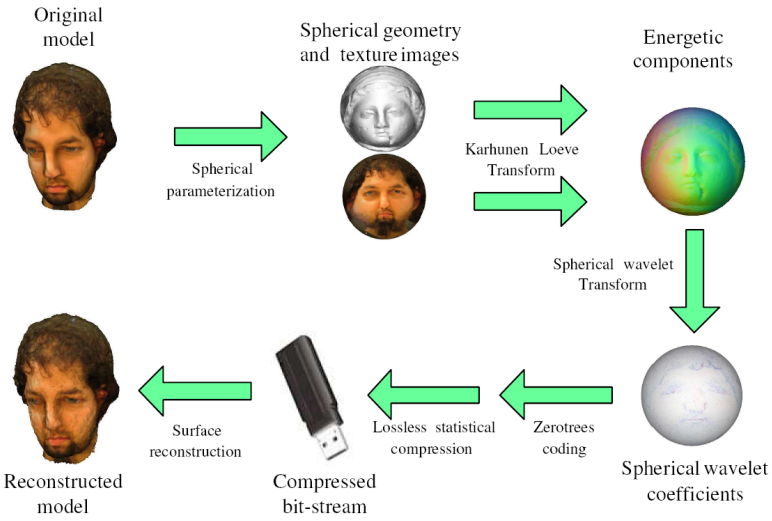


Figure 3.3: The proposed compression pipeline.

Hierarchical spherical grid

Consider the problem of determining a uniform sampling grid on a sphere. Dutton [77] obtained uniform spherical grids by tightly enclosing a regular polyhedron within the sphere, and taking the vertices that touch the sphere as the spherical grid points. Only for a tetrahedron, octahedron and icosahedron, the resulting spherical grid can be triangulated by identical equilateral spherical triangles. This initial regular mesh can be recursively subdivided to obtain a hierarchical spherical sampling grid. As suggested by [211], an icosahedron-based grid subdivision scheme is used. The resampling is at the first available resolution greater than the original resolution, i.e., at the first subdivision level containing more grid points than the original number of surfels. Resampling requires interpolation of the available scattered function values at the spherical grid positions. This is accomplished using the method of Franke and Nielson [93], that is based on proximity-weighted neighbor averaging.

Spherical wavelets representation

The resampling of the surface subsequent to the dense spherical mapping generates a multi-layered (vector-valued) attribute function sampled on the hierarchical spherical grid. This representation is characterized by spatial redundancy as well as by inter-layer redundancy.

To remove the inter-layer redundancy, the Karhunen Loève transform (KLT) is employed. The vector-valued samples of the attribute function corresponding to the given surface are

(a) (b)

Figure 3.4: Compression of a textured Igea model [188] (134345 surfels). (a) Original textured model. (b) Compression to 3.56 bits/surfel. **[see Color plate C.4]**

used to compute the covariance matrix. Each vector-valued sample is represented using the Karhunen Loève basis vectors derived from the covariance matrix. Note that the dimension of the KLT is very low, equal to the number of layers, i.e., to the total number of surface attributes that are carried into the representation (three spatial coordinates, three color components, etc.). Thus, its computation is cheap, and the overhead due to its inclusion in the compressed file is negligible.

To remove the spatial redundancy, each layer is then represented by spherical wavelet decomposition via the lifting scheme. This method was presented by Sweldens [229], and later adapted for functions on spherical domains by Schröder and Sweldens [211]. In the lifting scheme, the wavelet basis function is not formulated explicitly, but is implicitly 'lifted' from level to level, using a 'mother' wavelet function. In this work the butterfly subdivision function [79] is used as a down sampling 'mother' function. A lifting stage creates a down-scaled version of the original function, and a set of wavelet coefficients that can be later used for reconstruction. The resulting wavelet coefficients are encoded using the zerotree algorithm [217], and then losslessly stored using a statistical coder.

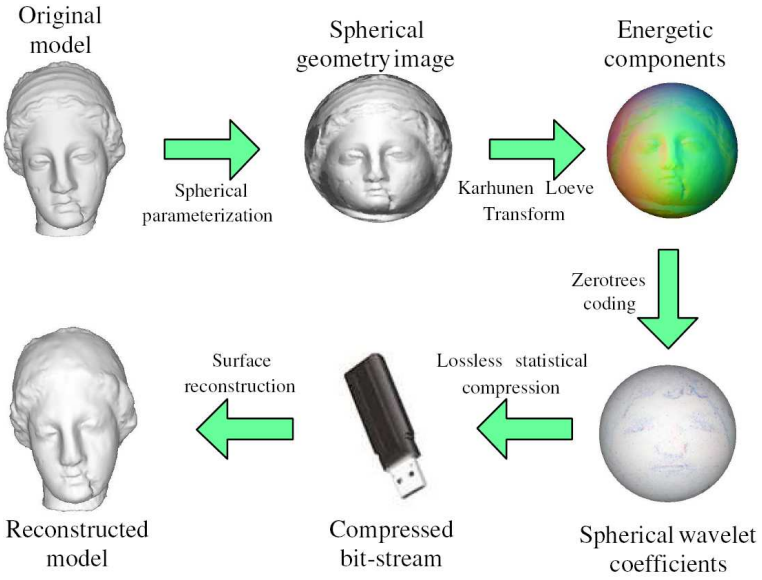


Figure 3.5: Geometry only compression pipeline.

3.3.2 Special case: geometry only

The compression of purely geometric (textureless) surface data is of practical importance, and has received much greater attention than the more general problem considered in Section 3.3.1, see [127, 177] and references therein.

The presented compression scheme can be reduced to the case of pure geometry compression in a straightforward manner. Beyond its significant practical value, this exercise creates the possibility of quantitative comparison of the approach presented in this chapter to related methods, for which performance data is available.

The geometry-only compression pipeline is shown in Figure 3.5. The three geometry layers $\{X, Y, Z\}$ are resampled on the hierarchical grid, decorrelated using the Karhunen-Loève Transform and coded using the zerotree algorithm followed by statistical encoding.

Evaluation of geometry error

In order to evaluate the geometry error due to the compression-reconstruction process, an error measure is needed. In the case of triangle mesh surfaces, the Metro tool [54], that defines an RMS error between two mesh surfaces, is commonly used. For surfaces represented by surfel sets, we follow the error estimation algorithm introduced by Pauly et al. [184] and used by

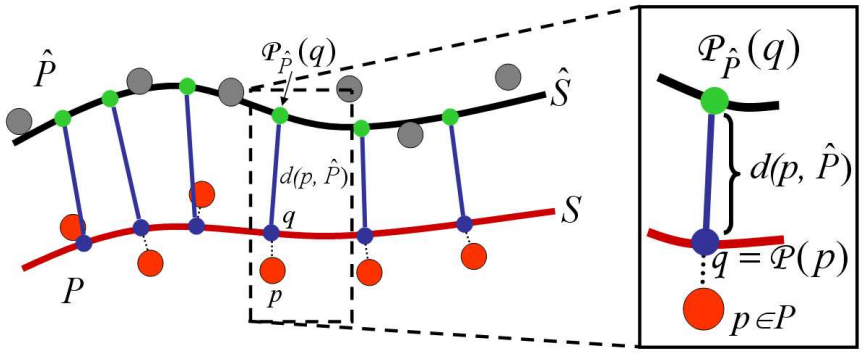


Figure 3.6: Measuring the distance between two surfaces S (red curve) and $\hat{S}$ (black curve) represented by two point sets P (red dots) and $\hat{P}$ (grey dots), respectively. Each point $p \in P$ is projected to the point $q = \mathcal{P}_P(p) \in S$ (blue dot). The point $q \in S$ is then projected onto the surface $\hat{S}$, resulting in the point $\mathcal{P}_{\hat{P}}(q) \in \hat{S}$ (green dot). The distance $d(p, \hat{P})$ is the Euclidean distance between these two points, $\left\|q - \mathcal{P}_{\hat{P}}(q)\right\|$. [**see Color plate C.5**]

Ochotta and Saupe [177]. The error is estimated as the RMS distance between the Moving Least Squares (MLS) representations [5] of the original and reconstructed surfaces.

For a given set of points P (the surfel centers in this case), that represents a surface S, Alexa *et al* [5] define a projection operator $\mathcal{P}_P(q)$ to project a nearby point q onto the surface. Given two point sets, P and $\hat{P}$, containing the centers of the surfels representing surfaces S and $\hat{S}$ respectively, the distance between a point $p \in P$ and the representation $\hat{P}$ of $\hat{S}$ is defined as

$$d(p, \hat{P}) = \left\|q - \mathcal{P}_{\hat{P}}(q)\right\| \tag{3.4}$$

where $q = \mathcal{P}_P(p)$ (see Fig 3.6), and the RMS error between S and $\hat{S}$ is

$$d(S, \hat{S}) = \left[\frac{1}{\|P\|}\sum_P d^2(p, \hat{P})\right]^{\frac{1}{2}} \tag{3.5}$$

Since this error measure is not symmetric, i.e., $d(\hat{S}, S) = d(S, \hat{S})$ is not generally true, a symmetric error is defined:

$$E_{RMS} = \max\left\{d(\hat{S}, S), d(S, \hat{S})\right\} \tag{3.6}$$

The RMS error is expressed in $\%d_{bb}$ units, where d_{bb} denotes the diagonal of the intersection of the (nearly identical) bounding boxes of the two surfaces. Specifically, for a surface tightly bounded by the unit cube, $d_{bb} = p3$, and $\%d_{bb}$ denotes percents of that number. The measure

E_{RMS} quantifies the surface geometry error introduced by the compression process, and does not take texture into account. It allows to evaluate the dependence of the geometry distortion on the compression ratio in the proposed scheme, and enables comparison with other surfel based compression methods with respect to geometric distortion. A geometry error analysis example is presented in Figure 3.7. Figure 3.7c is the visualization of the local discrepancy between the outcome of the compression process (Figure 3.7b) and the original surface (Figure 3.7a). The global error in this example is $E_{RMS} = 3.01 \cdot 10^{-4}\ d_{bb}$.

3.4 Results

3.4.1 Geometry only compression

Experimental results are presented for the *Igea* model, originally taken from the Cyberware 3D samples library and converted to surfel based representation by the Pointshop3D team at ETH Zurich [188]. Pointshop3D software [268] was also used to render the results throughout this thesis. The texture-free 3D *Igea* model, consisting of 134345 surfels, technically at 96 bits/surfel, is shown in Figure 3.9a. Figure 3.9b–d show the outcome of the proposed compression scheme with 5.17 bits/surfel, 1.22 bits/surfel and 0.42 bits/surfel respectively. The corresponding point-wise reconstruction error $d(p, \hat{P})$ is visualized in Figure 3.9e–g. The respective overall error values E_{RMS} are $3.48 \cdot 10^{-4}\ d_{bb}$, $6.66 \cdot 10^{-4}\ d_{bb}$ and $9.99 \cdot 10^{-4}\ d_{bb}$. Figure 3.8 shows E_{RMS} as a function of the bit-rate and gives a quantitative comparison to the results of Ochotta and Saupe [177], Waschbüsch *et al* [254] and Fleishman *et al* [91]. The method presented in this chapter is superior to the method of [177] at the lower bit-rates. The method of [91], that provides excellent results at higher bit-rates, is inapplicable at very low bit rates due to its need to encode a base model [177]. Waschbüsch *et al* [254] do not report results at very low bit-rates. Low bit-rate compression results for the *Max Planck* model [188] are presented in Figure 3.12.

3.4.2 Compression of textured surfaces

Since the *Igea* model [188] is provided without texture, it was mapped with a facial texture using Pointshop3D [268]. The textured *Igea* model, technically with 120 bits/surfel, is shown in Figure 3.10a. Figure 3.10b is the result of compression of the original surface to 3.56 bits/surfel. Figure 3.10c is the result of compression to 0.89 bits/surfel. An additional example of textured 3D model compression is presented in Figure 3.11 for the Trilobite model from Arius3D Inc.

This model consists of 726027 surfels, technically 120 bits/surfel (shown in Figure 3.11a). Figure 3.11b shows the result of compression to 6.3 bits/surfel, while Figure 3.11c shows the result of compression to to 3.77 bits/surfel.

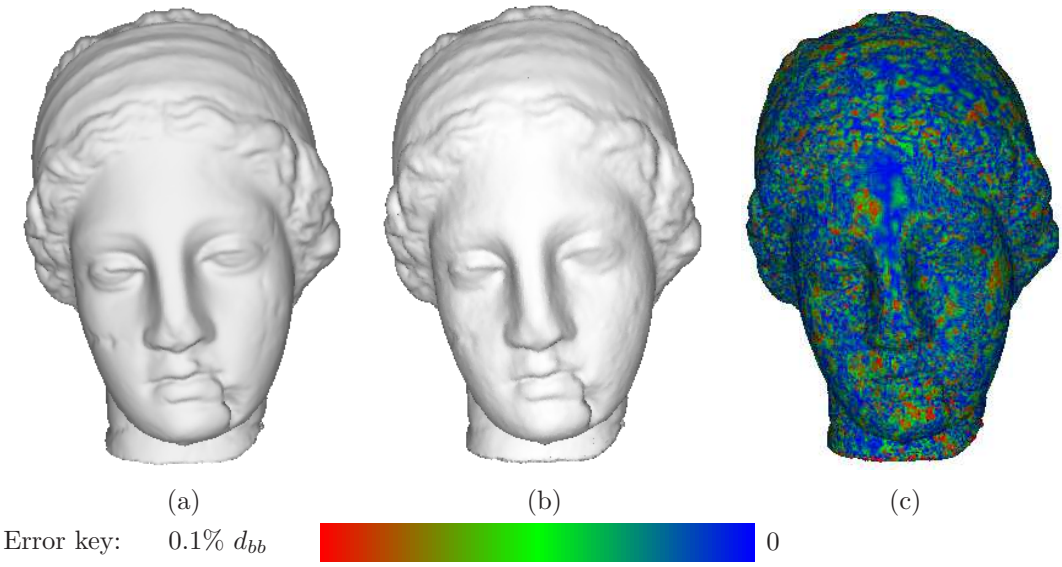


Figure 3.7: Example of error measurement: (a) original *Igea* model [188]. (b) Compressed model. (c) Error mapped on the compressed model, total $E_{RMS} = 3.01 \cdot 10^{-4}$ d_{bb}. **[see Color plate C.1]**

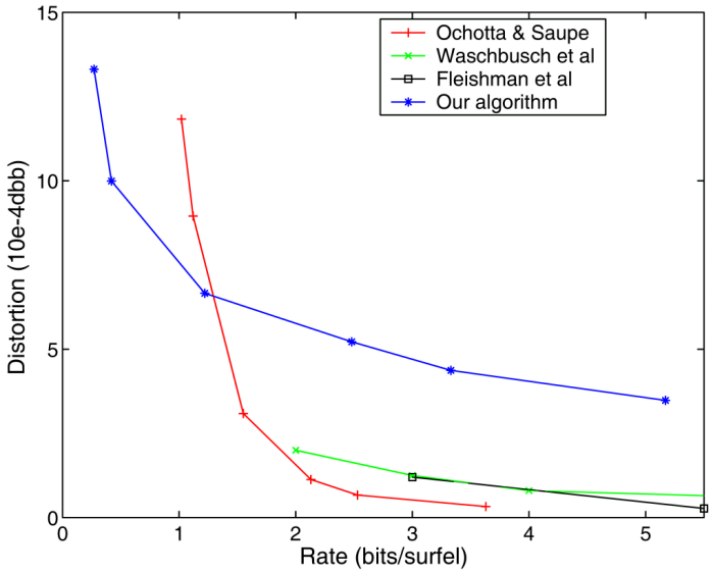


Figure 3.8: A comparison of the geometry compression error E_{RMS} as a function of the bit-rate in the compression of the texture-free *Igea* model [188].

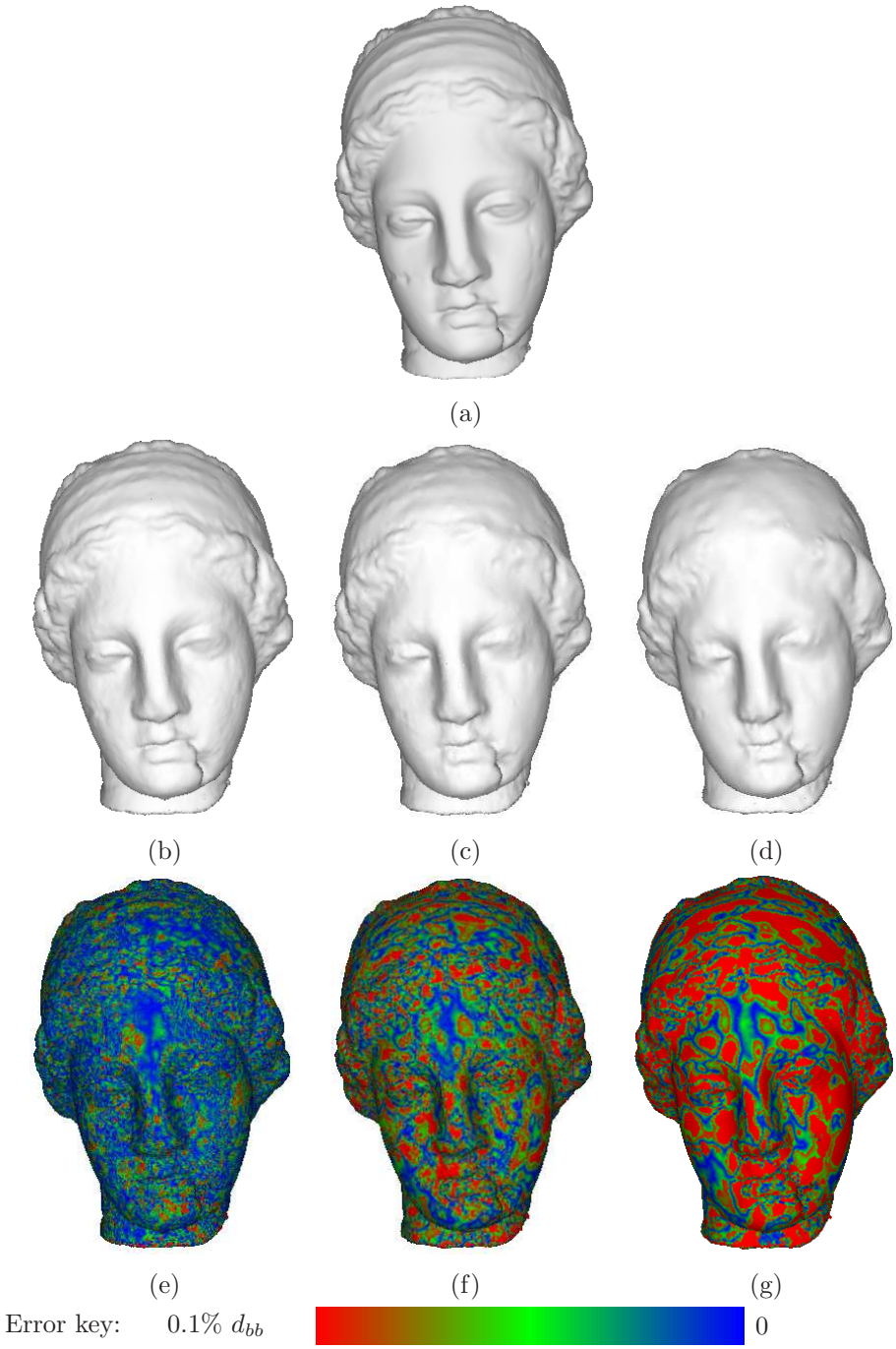


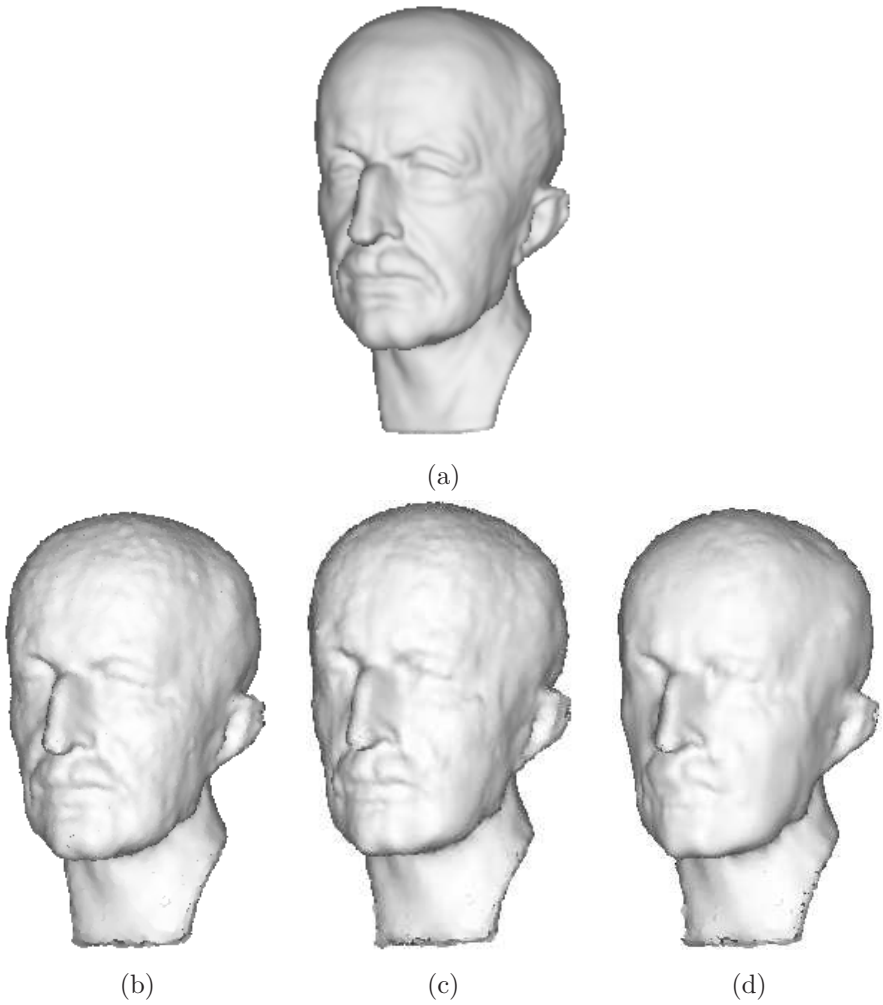
Figure 3.9: Compression of the *Igea* model (geometry only). (a) Original model. (b) Compressed to 5.17 bits/surfel. (c) Compressed to 1.22 bits/surfel. (d) Compressed to 0.42 bits/surfel. (e) The point-wise reconstruction error in (b). (f) Error in (c). (g) Error in (d). [**see Color plate C.6**]

(a) (b) (c)

Figure 3.10: Compression of the *Igea* 3D model [188] with texture. (a) Original model, 134345 surfels, technically 120 bits/surfel. (b) Compression to 3.56 bits/surfel. (c) Compression to 0.89 bits/surfel. **[see Color plate C.7]**

(a) (b) (c)

Figure 3.11: Compression of the textured *Trilobite* model (3D model courtesy of Arius3D Inc. [10]). (a) Original model, 726027 surfels, technically 120 bits/surfel. (b) Compression to 6.3 bits/surfel. (c) Compression to 3.77 bits/surfel. **[see Color plate C.8]**



(a)

(b) (c) (d)

Figure 3.12: Low bit-rate compression of the *Max Planck* model [188] (geometry only). (a) Original model. (b) Compressed to 3.50 bits/surfel, the error is $E_{RMS} = 14.1 \cdot 10^{-4}\ d_{bb}$. (c) Compressed to 1.98 bits/surfel, $E_{RMS} = 16.5 \cdot 10^{-4}\ d_{bb}$. (d) Compressed to 0.98 bits/surfel, $E_{RMS} = 18.4 \cdot 10^{-4}\ d_{bb}$.

Chapter 4

Point based model retrieval

In this chapter we present a framework for point based model retrieval. We introduce a system in which an object represented by a point based model serves as a query, and similar objects are retrieved from a collection of point based models. We consider the problem of matching both the geometry and the topology of point based surfaces. This problem is tackled with two different matching techniques. In Section 4.4 we present a technique based on depth-buffer images, which is suitable to discriminate the geometry of point based models. In Section 4.6 we present a technique based on geodesics, which is particularly effective in comparing the topology of point based models, especially those models representing objects isometrically deformed (see Appendix A).

Point based representations of 3D objects have recently received a lot of attention by the computer graphics community and the industry, due to the diffusion of several 3D scanning techniques. Shape similarity search of point based models has not been studied in depth yet, even though the literature provides a rich variety of shape similarity search methods for polygonal mesh models.

The number of point based models and applications dealing with point based representations is swiftly increasing. In the near future large and complex databases of point based models will be generated. Such databases will require powerful 3D model retrieval systems (see Section 4.1.1) in order to perform appropriate effective and efficient queries. However, such databases are not available yet, and producing suitable point based representations of 3D objects is still a difficult task. We consider surfels as suitable point based primitives both for rendering and for geometry processing (see Section 2.1.2). A surfel based representation of a 3D model can be mainly generated with the two following approaches:

- acquisition: computing surfels directly from 3D points clouds obtained by 3D scanning systems and 3D reconstruction procedures;
- surfelization: extract surfels from a generic geometric representation of a 3D model (triangular mesh, implicit surface, parametric surfaces, CSG, etc. [43]).

The nature of these two approaches is different, and when both are applicable the acquisition process is commonly more complicated and time-consuming than the surfelization process. For this reason, our framework includes a surfelization technique to convert retrieval benchmark databases of triangular mesh models [17, 219] into databases of surfel based models, which are needed to execute our retrieval experiments. In Section 4.3, we present a surfelization technique, which preserves the geometric and visual features of the triangular meshes.

4.1 3D model retrieval

This section provides a brief introduction to 3D model retrieval. The characteristics of a 3D model retrieval system are presented in Section 4.1.1. Section 4.1.2 presents a set of tools to evaluate the retrieval performances of a retrieval system, like the precision and the retrieval effectiveness of its matching algorithms.

4.1.1 3D model retrieval system

A 3D model retrieval system (3DMRS) [17, 92, 94, 117, 173, 230, 246] is a system where a 3D model serves as a query, and a set of similar 3D models are retrieved from a collection or a database of 3D models. In this thesis as well as in the major related applications, a 3D model represents a 3D object through a geometric representation, e.g., a polygon/triangle mesh, surface patches, or a point based representation. In this thesis, we call point based model retrieval system a 3DMRS, where 3D models are given in a point based representation. Figure 4.1 shows how a 3D model retrieval system is organized. In the following sections its main components are described in detail.

Query interface

A user can access a 3DMRS by submitting a query through a query interface to search for a 3D object. A query interface allows a user to browse a database of 3D objects and select a query object among the 3D models stored in the database. It also allows to search for an object by

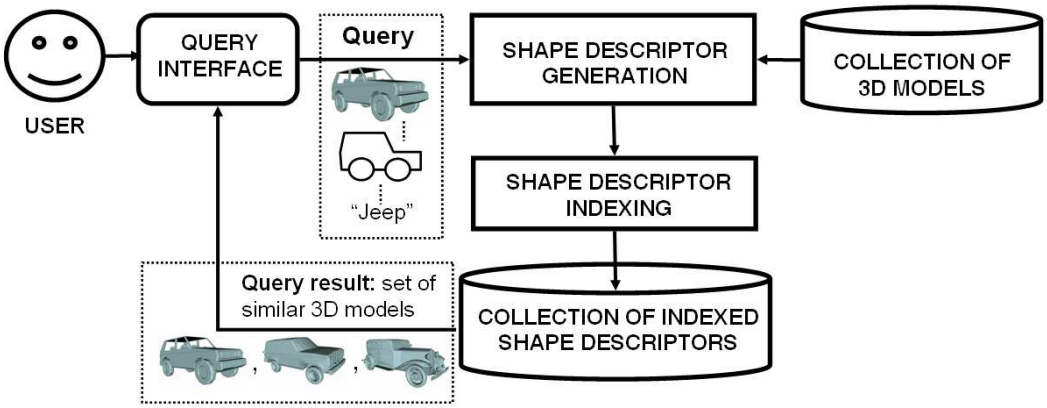


Figure 4.1: Basic scheme of a 3D model retrieval system

example (*query by example*), i.e., by submitting a 3D model to a 3DMRS and search for similar 3D models in the database. In this case a 3D model can be uploaded in a 3DMRS as new object to query. Some query interface [92, 94, 173, 230] also allows a user to sketch an object either in 3D with simple and user friendly 3D modelling tools [116, 260] or in 2D by drawing the 2D shape of an object from different viewpoints.

Text queries are also supported by some 3DMRS to refine a shape-based query. In fact, several 3D models are described in textual formats and may contain useful characteristics of the objects that they represent (e.g., VRML, OFF, PLY etc.). However, such information can be misleading, thus a separated textual descriptor is commonly associated to each model [94].

Shape descriptor generation

Once a query has been submitted, a 3DMRS is required to efficiently perform a similarity search. For this reason, as many other content-based information retrieval systems, a 3DMRS extracts a *shape descriptor* (SD) for each 3D model, and measures the similarity between two 3D models by comparing the corresponding SDs. In this context, a similarity search technique aims at computing the similarity (or dissimilarity) between the shapes of two 3D models. Thus, it is called *shape similarity search*. A shape similarity search technique mainly consists in extracting meaningful features from an object to generate a shape descriptor. The result of a shape similarity search is a set of objects of the database which are usually ordered ascendingly considering the similarity measures of their SDs with respect to the SD associated with the query object. Obviously, in this scenario the SD must be accompanied by a *metric* or more

generally by a *dissimilarity function* (distance function). A *dissimilarity function* on a set O is a non-negative definite function $\delta : O^2 \rightarrow \mathbb{R} \cup 0$ with the following properties:

1. *Identity*: $\forall x \in O, \delta(x, x) = 0$.
2. *Positivity*: $\forall x \neq y \in O, \delta(x, y) > 0$.
3. *Symmetry*: $\forall x, y \in O, \delta(x, y) = \delta(y, x)$.
4. *Triangle inequality*: $\forall x, y, z \in O, \delta(x, z) \leq \delta(x, y) + \delta(y, z)$.
5. *Transformation invariance*: for chosen transformation group G $\forall x, y \in O, g \in G, \delta(g(x), g(y)) = \delta(x, y)$.

When only the first four properties hold, the function δ is called *metric*. Transformation invariance is important when the comparison and the extraction process of SDs have to be independent of the place, orientation and scale of the object in its Cartesian coordinate system [232].

A good SD should meet the following important requirements:

- *efficiency:* an SD should be efficient to compute, since in case of queries by example it must be computed on the fly.
- *invariance with respect to transformation, rotation, scaling and reflection of a 3D model:* since 3D models are collected from different sources and created with different technologies (3D scanning, 3D modelling, etc), they may have arbitrary position, scale, orientation and reflections in their coordinate system. Thus, an SD should be invariant with respect to those transformations.
- *robustness with respect to noise, outliers, and topological degeneracies in the 3D model:* 3D models acquired through 3D scanning techniques present noise and outliers, and may have topology degeneracies produced by the surface reconstruction algorithms. An SD should not change significantly in presence of those noisy characteristics.
- *robustness with respect to different level-of-details and surface samplings:* this property of an SD is important, since a 3D model may be stored in a database at different level of detail. Moreover, depending on the surface representation, the same object may be represented with different tessellations (for polygonal meshes/patches) or sampled with different techniques (e.g., point based surfaces acquired from different scanners).

- *invariance to isometric transformation (e.g., bending deformations):* this property is important for topology matching and for recognizing objects in those applications where different instances of an object are stored in a database after a geometric deformation, e.g., models of human bodies or animals in different postures.

Several algorithms for computing a shape similarity between two 3D polygonal models have been presented in the last decade (see Section 4.2). Those algorithms follow different approaches and generate SDs of different nature, which can be broadly summarized in the following four categories.

1. *Feature vector approach*: this approach consists in computing a *feature vector* (FV), i.e., is a high-dimensional vector storing a compact description of the shape of an object [40]. Conceptually an FV may be built by storing different type of shape features such as: spectral coefficients, spatial curvature, visual signatures, etc. Feature vectors are compared with a suitable metric, which is usually defined by a Minkowski distance. The dimension of an FV depends on the application and the matching algorithm. It affects the retrieval effectiveness of a shape descriptor and the efficiency of the overall similarity search [41]. In fact, to perform efficient queries some similarity search algorithm requires loading the overall database of FVs in the main memory, or in the memory of a graphics card [38]. Thus, the dimension of an FV may be fundamental for the efficiency of a 3D model retrieval system. In practice, it is usually set as a parameter of the shape similarity algorithm by the user, who finds the right trade-off between storage space and effectiveness of a FV. Moreover, in practical applications the dimension of an FV has usually a saturation point, where adding more dimensions to an FV only adds noise to the shape description [40].

2. *Histogram approach*: this approach relies on matching methods that statistically examine geometric features of randomly selected geometric entities, which are defined on the 3D model surface [103, 147, 181, 202]. Such methods generate histograms that are compared with suitable metrics or alignment methods. In general, matching histograms is more complicated and time consuming than matching FVs, since it often requires a dynamic scaling or a rough alignment before matching the histogram bins (see Section 4.6.3).

3. *Graphs approach*: this approach describes the structure of an object by means of a graph, which is related to its topology [22, 107, 237, 240]. Topologically meaningful parts of an

object are usually extracted and represented by the nodes of a graph, while the graph edges describe how they are connected to each other. The geometric characteristics of an object part is associated to the corresponding node for geometric matching. Other approaches use a graph to describe the skeleton of an object. Computing a similarity between graphs is not as efficient as for feature vectors, since in some case it may require combinatorial algorithms [22, 107]. Furthermore, graph based approaches may be sensitive to small changes in the 3D objects, which may lead to large changes in the corresponding structural graph [107, 227]. For this reason they are suitable only for particular applications and not for solving a general retrieval problem.

4. *Approach based on abstract representations*: this approach is mainly based on extracting a set of individual features in a low dimensional transformed space [45, 57], e.g., n-dimensional points or curves on a canonical spaces computed with principal component analysis or multidimensional scaling [87, 234]. These features require an alignment procedure before being compared with a suitable metric. Computationally, this approach is similar to the graph based approach and it is less efficient than the feature vector approach.

Among the classes of SDs mentioned above, the feature vector approach is the most commonly adopted by the major 3D model retrieval systems, since it resulted in being the most efficient, robust, and easy to implement [40, 94, 219, 232, 249]. Using a feature vector approach, for each object of the database an FV is generated and stored with the other FVs in a particular structure which facilitates the retrieval, as described in the next section.

Indexing shape descriptors

In a 3D model retrieval system a similarity search is typically performed through a k nearest neighbors (NN) query on the stored collection/database of shape descriptors. A *k-NN-query*, retrieves the k closest objects to the query object. Another type of search which is usually performed on multimedia databases is the range-query. Given a radius r, a *range query* retrieves those objects that are inside the radius r of the query object with respect to a metric.

Using a feature vector approach, a database of feature vectors can be modelled as a *vector space*, usually of high dimension. A vector space of dimension d is a set V of vectors $\vec{x} = (x_1, .., x_d)$ with two operations (called vector addition and scalar multiplication), which satisfy a set of axioms [226]. A vector space accompanied with a metric defines a metric space [262]

(see also Appendix A). Let $\vec{x} = (x_1, .., x_d)$ and $\vec{y} = (y_1, .., y_d)$ be two vectors of dimension d, the *Minkowski distance of order p* $L_p(\vec{x}, \vec{y})$ between $\vec{x}$ and $\vec{y}$ is defined as

$$L_p(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p} \tag{4.1}$$

In a similarity search different orders of the Minkowski distance are usually taken as suitable metrics for a vector space, e.g., the Manhattan distance (1-order or L_1) and the Euclidean distance (2-order or L_2).

In a metric space space, similarity searches through k-NN-queries and range-queries can be efficiently performed with several algorithms [23, 28, 38, 41, 255, 262]. These algorithms rely on efficient indexing techniques, where each feature vector of the database is associated to an index, which is stored in a suitable index structure to optimize the similarity searches. An index structure serves to efficiently discard irrelevant objects (or complete zones of the search space), in order to avoid costly sequential scans of the entire database for each search. An detailed overview of indexing methods can be found in [262]. Efficient indexing techniques for shape similarity search on multimedia databases modelled with feature vectors are presented in [38, 41].

4.1.2 Evaluating the retrieval performance of a shape descriptor

This chapter focuses on generating effective shape descriptors for retrieving point based models. The retrieval effectiveness of a shape descriptor can be measured with different tools used in information retrieval theory [11] to evaluate similarity search strategies. One of the most commonly used tool for comparing the retrieval effectiveness of competing shape descriptors is the *precision-recall diagram*. Given a query object q, a similarity search algorithm retrieves a set R of similar objects from a collection/database of objects. Usually, among the retrieved objects only a portion of them are relevant to the query. A *relevant object* is an object that belongs to the same class (category) of the query object. To establish the relevance of a retrieved object we need to define a classification of the objects in the database. Let O be the set of all objects in a database, we call *classification* of the database O a set C of n disjunctive subsets C_i of O:

$$C = \bigcup_{i=1}^{n} C_i \quad : \quad C_i \subset O \;\; and \;\; C_i \cap C_j = \emptyset \;\; \forall \, i \neq j. \tag{4.2}$$

A database may contain unclassified objects. Thus, let U be the set of unclassified objects, $U \cup C = O$. Given a query object q belonging to the class C_q (i.e., the set of relevant objects

to q in the database) we define $C_q^- = C_q \setminus \{q\}$ as the set of all objects belonging to the class C_q without the query object q.

The *precision* of a similarity search algorithm is defined as the portion of retrieved objects that are relevant, while the *recall* is defined as the portion of relevant objects actually retrieved. Formally, let R_q be the set of retrieved objects for the query q, and Ω_q be the set of relevant objects retrieved excluding the query object ($\Omega_q = R_q \cap C_q^-$), the *precision* and the *recall* of a similarity search algorithm for the query q are defined as:

$$precision_q = \frac{|\Omega_q|}{|R_q|}, \quad recall_q = \frac{|\Omega_q|}{|C_q^-|}. \tag{4.3}$$

Thus, $precision_q$ is the ratio of the number of relevant objects retrieved ($|\Omega_q|$) to the number of all objects retrieved ($|R_q|$), and $recall_q$ is the ratio of the number of relevant objects retrieved ($|\Omega_q|$) to the number of all relevant objects stored in the database excluding the query object ($|C_q^-|$). Precision and recall are usually expressed in percentage. Obviously, the ideal value for precision is 100%. Figure 4.2 shows the result of a query on a collection of objects, which are classified in eleven classes depicted with different colors. The set of retrieved objects is visualized with a blue contour. The relevant objects are those in the class C_2, i.e., the same class of the query object (highlighted with a red contour).

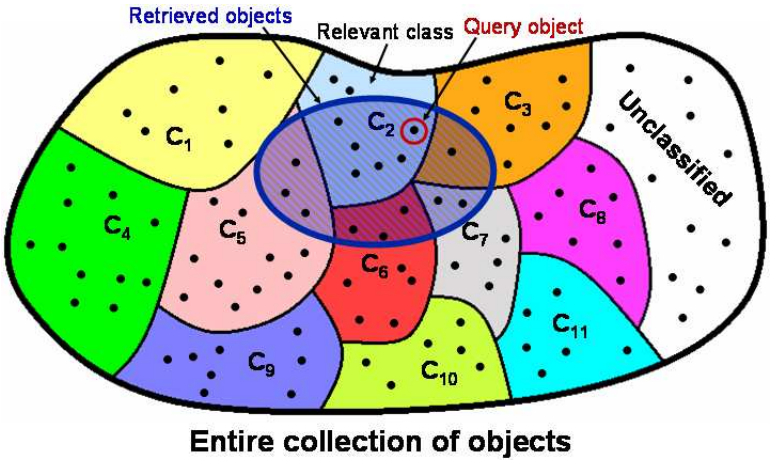


Figure 4.2: Principle of a precision-recall diagram. Given a query object, usually only a portion of the retrieved objects is relevant to the query, i.e., those objects belonging to the same class of the query object (C_2). [**see Color plate D.1**]

Other measures of retrieval effectiveness commonly used to evaluate search strategies are based on ranking the retrieved objects with respect to their similarities, i.e., the objects are

ordered in ascending order according to their distance (dissimilarity value) to the query object. The most representative evaluation measures are listed below.

***Nearest neighbor (NN)*:** it indicates the percentage of the k best (closest to the query object) retrieved objects that are relevant. The value of k is selected by the user, and usually is set to 1.

***R-precision (RP) (or first tier)*:** it indicates the percentage of all relevant objects that are retrieved among the $|C_q^-|$ best ranked objects. That is the precision when the number of objects retrieved is $|C_q^-|$:

$$RP = \frac{|\Omega_q|}{|R_q|}, \quad when \quad |R_q| = |C_q^-| \quad (precision_q = recall_q). \tag{4.4}$$

Since, the R-precision is that value of precision when $precision_q = recall_q$, it can also be interpreted as the recall for the smallest number of retrieved objects that could possibly include 100% of the objects in the query class.

***Bull's Eye Performance (BEP)(or second tier)*:** it indicates the percentage of all relevant objects that are retrieved among the $2 \cdot |C_q^-|$ best ranked objects. It is similar to the R-precision but is twice as big. The BEP is adopted by MPEG-7 [100] to evaluate the retrieval performance of descriptors for visual content [101].

***Average precision over recall range 50% and 100% ($\bar{P}_{50}$ and $\bar{P}_{100}$)*:** they represent the average precision over half of the recall range, i.e., $\frac{1}{G}\% \leq recall \leq 50\%$ ($\bar{P}_{50}$), and over the whole recall range, i.e., $\frac{1}{G}\% \leq recall \leq 100\%$ ($\bar{P}_{100}$), where G is the discrete recall resolution used for interpolating the precision values. As proposed by Vranić in [249], these values are interesting to compare the overall performance of different shape descriptors.

***Distance matrix image*:** it is an image, which visualizes a dissimilarity matrix (or distance matrix). A *dissimilarity matrix* of a set of M objects is a squared matrix of size $M \times M$, in which each element m_{ij} of its ith row and jth column stores the dissimilarity value (distance) between the ith object and the jth object. A dissimilarity matrix is symmetric and has all the elements of the main diagonal equal to 0. The *distance matrix image* is then an image of M^2 pixels, where lightness of each pixel (i, j) is proportional to the magnitude of the element m_{ij} of the corresponding dissimilarity matrix. Thus, a white pixel indicates the maximum distance, while a black pixel a distance equal to 0. The pixel on the main diagonal are all black. A distance matrix image is a useful tool to visually

and intuitively evaluate and analyze the retrieval effectiveness of a shape descriptor. In fact, models are usually grouped by class along each axis, so that the optimal result is visualized as a set of darkest blocks of pixels (as large as their class size) along the main diagonal. This indicates that every object matches the objects within its class better than ones in other classes. Moreover, it also allows to visualize the relation between two classes, e.g., off-diagonal blocks of dark pixels indicate that two classes match each other well.

The evaluation measures defined above are related only to a single query. To compute a general and realistic evaluation of the retrieval effectiveness of a similarity search method, those measures are statistically analyzed. The usual approach is to average the measures over all query models (*micro-average*) or to average the averages computed for each class (*macro-average*). All the evaluation measures expressed in this chapter consider the micro-average.

4.2 Related work

Point based model retrieval has not yet been explored in depth. So far, only few matching techniques have been proposed for point based models. Mémoli and Sapiro in [162] compared point clouds representing samplings of manifolds by using an estimate of the Gromov-Hausdorff distance. This distance is computed by considering geodesic distance matrices, which are constructed from meaningful subsets of the point clouds (for more details see Section 4.5).

Dey et al. [72] compared noisy point clouds, by matching signatures extracted from segmented parts of the point clouds. A point cloud is first segmented by making use of the Morse theory (see Section 4.5), their Voronoi vertices and their Delaunay tetrahedra. A signature of a shape is derived from its segmentation considering a weighted point and its volume. A point cloud is described by the signatures of few significant segments. To compare two point clouds their set of signatures are aligned and compared in their space. This approach is not effective for deformed objects.

Collins, Carlsson et al. [45, 57] compared point clouds by comparing barcode descriptors computed by exploiting the persistence homology theory. Their algorithm uses differential geometry and algebraic topology. From a point cloud they constructed an approximation of its filtered tangent complex, starting from an approximation of the tangential space of the point cloud. This representation is constructed to capture, with high precision, differences in the geometry and topology of curves and surfaces. A barcode shape descriptor is then build

by applying a persistence homology algorithm [266]. Two barcode descriptors are compared by recasting their matching as a maximum weight bipartite graph matching problem. They experimented their algorithm with 2D point based curve, and proposed an extension for surfaces in $\mathbb{R}^3$.

Many techniques were invented for shape matching of polygonal models. An extensive overview of such methods is provided in [232]. According to [232], they can be broadly summarized as follows:

Feature based methods: these approaches compare the similarity of 3D models by extracting suitable geometric and topological features, and storing them in a feature vector. This class of methods can be further subdivided in four categories:

- **global features:** these methods extract global features that characterize the global shape of the entire model such as: statistical moments of the boundary/volume of the model, volume-to-surface ratio, the Fourier transform of the volume, and the boundary of the shape [62, 183, 263].
- **global feature distributions:** these methods compare distributions of global features instead of the global features directly, e.g., distributions that measure properties based on distance, angle, area and volume measurements between random points of a surface [118, 178, 181] (see also Section 4.6).
- **spatial map distributions:** these approaches extract spatial locations like physical locations or sections of an object, and arrange them in a manner that preserve the relative positions of the features in an object [9, 94, 126, 247, 248].
- **local features:** these approaches consider local surface shape characteristics extracted in the neighborhood of points on the boundary of the shape [138, 259].

graph based methods: these methods aim to extract geometric features of a 3D shape by using a graph showing how shape components are linked together. Graph based methods can be divided into three broad categories according to the type of graph used: model graphs [83, 158], Reeb graphs [26, 107], and skeletons [26].

image based methods: these approaches consider two models similar if they look similar from any different angle. They typically extract a feature vector from a set of images obtained by projecting a 3D model onto planes defined by different view directions [51, 149, 153, 204, 243] (see also Section 4.4).

Volumetric error based methods: these approaches calculate a volumetric error between two voxelized models by computing their transportation distance, which measures how many voxels have to move and how far to change one shape into another [223].

Weighted point set based methods: these approaches compare shapes by comparing weighted point sets, which are sampled from the models with different metrics and waiting functions (e.g., curvature of local surface) [45, 71, 233].

Deformation based methods: such methods compare two 2D shapes by measuring the amount of deformation required to register the shapes exactly [18].

Comparisons of shape descriptors were presented by several authors. Shilane et al. in [219] presented the Princeton shape benchmark, which consists of a classified database of polygonal models collected from the World Wide Web. It also provides a set of tools for evaluating and comparing the performance and the retrieval effectiveness of different shape matching algorithms with respect to different classification models.

According to the experiments presented in [219] the light field descriptor presented by Chen et al. in [51] resulted in having the best retrieval effectiveness. The main idea of this approach is to extract 100 silhouette images from 10 different viewpoints (20 uniformly distributed viewpoints excluding symmetry) extracted from the 20 vertices of the bounding dodecahedron of a 3D object. The 100 images are generated by rendering with an orthogonal projection the 10 images associated to each of the 10 light fields constructed on a sphere. Each image is encoded with a descriptor considering Zernike moments (region-based) and Fourier descriptors (contour-based). In total they stored 35 coefficients for the Zernike moment descriptor and 10 coefficients for the Fourier descriptor, both quantized to 8 bits. The resulting coefficients are used to compute the similarity between two objects. To make the method robust with respect to rotation, the similarity is computed only between corresponding images of two objects computed by checking a set of possible rotations for each light field.

Bustos et al. in [40] presented an experimental comparison of feature-based 3D retrieval methods for polygonal meshes, where a shape descriptor based on depth-buffer images obtained the best average retrieval effectiveness. This method, presented by Heczko et al. in [106] and refined by Vranić in [249], builds six depth-buffer images by orthogonally projecting a polygonal mesh onto the six faces of its canonical bounding cube. Such a bounding cube is constructed after aligning the polygonal mesh to its principal axis computed using Principle Component Analysis (PCA). The 3D model is then scaled with respect to the main diagonal of its bounding

cube. A shape descriptor is constructed by considering the magnitude of the low-frequency Fourier coefficients, which are extracted from the 2D Fourier spectrums of the six depth-buffer images.

Image-based shape descriptors then demonstrated better discrimination (on average) than other shape descriptors on different databases and with several classifications [40, 219]. However, both [40] and [219] experiments showed that no single shape descriptor performs absolutely best for all classifications, and no single classification provides the best evaluation of all shape descriptors. In fact, the relative ranking of an algorithm may vary significantly for different classifications. Some matching algorithm performs better than others for particular classifications. An example of this fact is the shape descriptor based on an extended Gaussian image (EGI) [114, 124], that is defined by a spherical function giving the distribution of surface normals. Experiments in [219] showed that EGI performed best of all the other methods in discriminating between man-made objects and natural objects, even though for other classifications it performed worse than the other tested shape descriptors.

Similar results were shown in experiments conducted in [40], where the shape spectrum feature vector presented in [259] achieved the best retrieval performance in discriminating models of humans, while scored the worst on average. This matching method considers the distribution of local curvature of an object surface. It is defined as the distribution of the shape index for points on the surface, which is a function of the two principal curvatures at the respective surface points. Since it considers the distribution of local curvature, it was able to retrieve human models having different postures, while the other feature vectors retrieved only those models having roughly the same posture.

Although from these results it turns out that there is no absolute best shape descriptor for all classifications, a study conducted by Bustos et al. in [39] showed how to improve the retrieval effectiveness of a similarity search on a database of 3D models. The idea is to automatically combine shape descriptors with different characteristics. Different feature vectors can be automatically and dynamically combined for each query object by weighting them with a weight function. Such a function is computed considering an a priori estimation of individual FVs performances on a training set of classified objects. Two weight functions are considered. The first weight function considers a purity measure of each FV for a given query object. The purity of an FV indicates the maximum number of objects that belong to the same model class in the first k positions of each ranking. This value aims at measuring the coherence of the retrieved objects using an FV. The second weight function measures an entropy impurity of

each FV for a given query object. The entropy impurity of an FV is 0 if all the objects in a ranking belong to the same class, or is a positive real number that reaches its maximum value when the number of classes covered in a ranking is maximal (all classes are equally represented). Experiments showed that with a dynamically combination of FVs with both weight functions the overall retrieval effectiveness of a similarity search can be improved of about 40% in terms of R-precision with respect to the best single FV.

Since the goal of this research is to match point based models, we are particularly interested in those shape descriptor conceived for polygonal model that are suitable to be easily extended to point based representations. Among the shape descriptor classes mentioned above, the following three are the most suitable to be extended to point based models:

image-based methods: these methods are based on images generated by rendering 3D models. They do not depend on the characteristics of the data representation. Considering the high retrieval performance obtained by this class of shape descriptors in experiments run in [40] and in [219], the shape descriptors of this class are the most suitable to be converted to point based models. In Section 4.4 we present an extension of the depth-buffer image based feature vector for point based models.

feature-based methods: with these approaches feature values are typically computed by sampling 3D shapes. Thus, they can be extended to point set surfaces that, if needed, can be re-sampled according to specific requirements. An example of this method is the shape distribution technique presented by Osada et al. in [181]. This technique is based on an uniform point sampling of a triangular surface, from which the probability density function (PDF) of a shape function defined by the Euclidean distance between sample points is constructed. The matching between two models is then accomplished by comparing two PDFs with a minimization technique. Similarly, in the method presented by Ohbuchi and Takei in [179], a multiresolution alpha shape representation is constructed from a point set. The point set is obtained by sampling a triangular surface with a Monte-Carlo method. A 3D histogram, representing the final SD, is generated by combining the multiresolution alpha shape representation and an absolute-angle-distance (AAD) descriptor constructed as in [178]. The AAD method considers sample points and their normal vectors to build a 2D histogram of distances and mutual orientations of the sampled 3D points.

weighted point set based methods: since these methods are based on weighted points computed from geometric features of triangular surfaces, they can be extended to point set

surfaces as in [45, 57]. Tangelder and Veltkamp in [233] proposed a shape descriptor for polyhedral models using a weighted point set. A polyhedral model is aligned to its principal axes after applying PCA. A 3D grid enclosing the model is constructed, and for each non-empty cell a weighted point is computed. Three different methods are presented for computing weighted points. The first method considers the Gaussian curvature at each vertex of a cell, and takes the vertex with the highest curvature as weighted point. The second method computes an area-weighted mean point of the vertices of a cell, and weights it with its normal variation, which is calculated from the normal vectors of the adjacent faces. The third method computes a weighted center of mass of a cell, which is weighted with the unit. This weighted point expresses perceptual importance of the cell according to the technique presented in [201]. This technique computes for each vertex of the cell the inverse of the maximum angle between all pairs of incident edges, and the length of the longest edge among the incident edges. Two weighted point sets are then compared by using a modification of the Earth Mover's Distance.

Similar weight functions can be computed for point based models using the algorithms presented in [184, 185].

4.3 Surfelization of triangular mesh databases

Surfelization is the process of extracting surfels from a generic geometric representation of a 3D model (e.g., triangular mesh, implicit surface, parametric surfaces, CSG, etc. [43]).

In recent years different surfelization techniques were proposed (see Section 4.3.1). The main surfelization techniques used in computer graphics are designed to convert triangle meshes constructed from scanned data into a surfel set. These approaches rely on the fact that the resolution of a triangular mesh is similar to the resolution of the original point cloud obtained from the scanning process. The main idea is to associate a surfel to each vertex of a triangle mesh and compute its orientation, radius, and color by averaging or filtering the information extracted from the adjacent triangles [187, 206]. This approach works properly with dense triangular meshes and with smooth triangular surfaces. If a triangular mesh has several sharp features (e.g., crease and boundary edges, and spikes, see Figure 4.4) or discontinuities in the triangular surface, this approach may generate intersecting surfels, which cut the original triangular surface without capturing its geometric feature. This effect is shown in Figure 4.3 where

two triangular meshes (Figure 4.3a) are converted into the corresponding surfel based representations (Figure 4.3b) by using the approach proposed by Rusinkiewicz and Levoy in [206]. Their approach consists in assigning a sphere to each vertex, large enough to touch neighbor spheres. Each sphere is then associated to a surfel, and its properties are computed from the triangulation in the neighborhood of the corresponding vertex. Notice that with this approach the model of the bunny, which does not have sharp edges and has a high-resolution triangulation, is properly converted; whereas the model of the ant, which presents many crease edges and a low-resolution triangulation, is wrongly converted. In fact, the resulting surfels have over-estimated radii, uncorrected normals, and do not capture the real shape of the original triangular mesh.

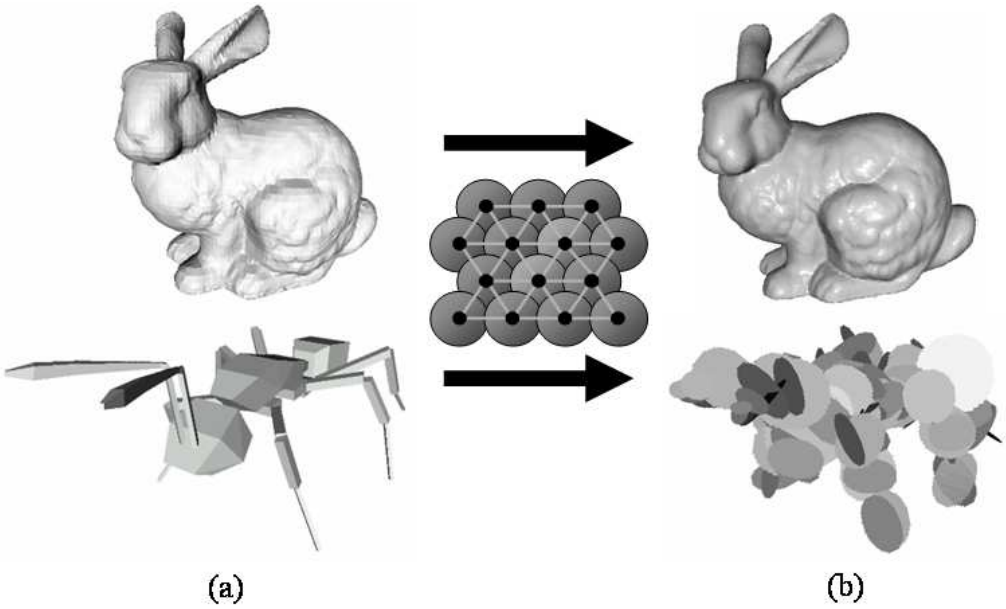


Figure 4.3: Result of the surfelization implemented by Rusinkiewicz and Levoy in [206]: the original triangulated models (a) and the corresponding surfel based representations (b). A sphere is assigned to each vertex. Each sphere should be large enough to touch neighbor spheres. The model of the bunny and the ant are respectively the triangle mesh *m0* and *m112* of the Princeton database [219]. The "Bunny" triangular mesh has 7691 vertices and 15294 triangles, and is converted into 7691 surfels. The "Ant" triangular mesh has 298 vertices and 492 triangles, and is converted into 298 surfels. The surfel based model in (b) are rendered using the point based rendering system QSplat [206].

3D models with sharp features are typical of many applications such as: CAD and architectural applications, city modelling, etc. Moreover, they represent the majority of the models

available on internet, which are commonly decimated to optimize their storage and transmission over the network. 3D model retrieval benchmark databases [17, 92, 117, 173, 219, 230] are mainly composed of these types of models, which were gathered from internet. Thus, in the context of this research it is important to conceive a surfelization method, which properly preserves the geometric characteristics of such models.

We present a surfelization technique, which aims at preserving the geometric properties of the original triangle mesh. Thus, the converted models have the same shape characteristics as the originals. This is important during retrieval tests for comparing the effectiveness of shape descriptors for point based models with similar shape descriptors developed for polygonal meshes. The basic idea of our surfelization method is to follow the triangulation of the original triangulated model, so that all the geometric structures of the model can be appropriately sampled. In fact, we do not know a priori what the source of a triangular mesh is. Therefore, by following this approach we ensure a correct geometric conversion. We sample triangles based on specific geometric features that can be easily extracted on the fly. These features are: a linear approximation of the surface flatness, and the edges generated by orientation and depth discontinuities. Each triangle is sampled on a regular square grid. The sampling pattern is adapted near the features to their shape in order to minimize the approximation error between the original triangular surface and the corresponding surfel based description (see Section 4.3.2). Our algorithm is suitable to be implemented in the current graphics hardware, due to its similarity to the rasterization process and for the simplicity of the geometric operations involved. Our surfelization technique is fast, and is therefore suitable to quickly convert large databases of triangulated 3D models.

4.3.1 Previous work

Surfelization is strictly related to the problem of sampling geometric models, which were studied in different settings. Different point sampling techniques were proposed in the past. They depend on the geometric representation of the 3D models (polygonal mesh, implicit surface, parametric surface, procedural geometries) and the target context (i.e. sampling or resampling for interactive and fast rendering, for the editing of point based models, for point set surfaces simplification, etc.). Another important factor is the execution time. Some technique requires complicated computations in the geometry and/or image space. The larger complexity typically leads to a finer approximation of the original surface geometry and texture at the cost of a high computation time. Thus, such approaches are well suited to be used when preprocessing

time is not critical. Grossman and Dally in [98] sampled triangular meshes on a set of 32 orthographic depth images computed from 32 different directions. They sampled these images on an equilateral triangle lattice. The density of this lattice is determined by a tolerance angle with respect to the view direction and by the size of the pixel of the target screen resolution. The final set of point samples is obtained by combining all the depth image samples and eliminating redundant information. This is a complex task and the overall approach tends to work better with smooth and convex surfaces. Pfister et al. in [187] started from the assumption that in a surfel representation the object sampling is aligned to the image space and must match the expected output resolution of the image. Thus, their sampling technique is based on a concept recalled from image based rendering: the Layered Depth Cube (LDC) [148]. The LDC consists of three orthogonal Layered Depth Images (LDI) [216]. The LDI is an image of depth pixels, and each of them can store multiple surfels. For each surfel a multiresolution texture color (called surfel mimap) is also stored. Each LDI is computed by using ray-tracing on the original geometric model from three different and orthogonal directions, which produce three orthogonal LDIs. The resulting LDC represents the first set of point samples that is refined by eliminating redundancies. The advantage of the techniques presented in [98, 187] is that they produce a compact, fine and multiresolution representation both of the geometry and texture color information of the original model. However, these methods have two main limitations. The first limitation is the processing time. Both methods may require running times of approximatively hours, but the technique of Pfister saves time in the elimination of the redundancies due to a more compact representation. The second limitation is that the sampling grid resolution does not adapt to the size of details and features in the model. Thin structures under the sampling resolution may be neglected (i.e. hairs, wires etc.).

Moenning and Dodgson in [166, 167] proposed a sampling method which potentially could overcome these limits. They extended the concept of farthest point sampling of images to triangle meshes. The basic idea of a farthest point sampling is to progressively place sample points in the middle of the least known area of the sampling domain (see also Section 4.6.1.1). They exploited the fast marching method to incrementally construct bounded Voronoi diagrams across triangular surfaces. Their sampling method, which takes time of the order of seconds, produces good visual results. Other related technique for resampling point set surfaces are based on an estimation of the local curvature of the ideal underlying surface or a local parametric approximation. These approaches are typically used for simplification of point based models and interactive point based editing.

The method of Alexa et al. [5] computes a local bivariate polynomial approximations of the surface in a neighborhood of an initial set of points, by minimizing a weighted least squares error. The points are then projected on this new surface called Moving Least Square (MLS) surface. They proposed an efficient algorithm to increase and decrease the resolution of the point samples on the MLS surface. This technique is well suited for point set surfaces where an approximation of the underlying implicit surface is necessary.

Zwicker et al. in [268] showed how to resample a point set surface with respect to a texture. Their approach is based on fitting local point samples to a polynomial function or to a tangent plane. They used a reconstruction kernel based on radially symmetric Gaussian to blend the resampled local patches with respect to a parametrization of the texture.

Related approaches apply physically-based particle systems to sample implicit surfaces [256] and isosurfaces [64]. Adapting these techniques to triangular mesh would require further approximation of the model surfaces from its vertices, which would introduce smoothing and an approximation error.

Rusinkiewicz and Levoy [206] generated an efficient hierarchy of bounding spheres structure of surfels directly from a triangular mesh for point based rendering and progressive transmission. Their algorithm assigns a sphere to each vertex, large enough to touch neighbor spheres. Each sphere is associated to a surfel. Then, the local triangulation in the neighborhood of each vertex is analyzed to compute the properties of the associated surfel. This approach works well when the original mesh is computed from range data (see Figure 4.3).

A finer approach directly based on the current triangle mesh was presented by Turk in [241]. He computed a level of detail representation of a triangular mesh by randomly spreading points on it surface. Then, he applied a relaxation procedure based on the repelling force of each point with respect to its neighbors. His method uses an approximation of surface curvature to weight the number of points that should be placed on a given area of the surface. The same relaxation method is used by Pauly et al. in [185] to dynamically reposition new point samples after surface distortion operations.

Other techniques are conceived to be performed on the fly exploiting graphics hardware. Tobor et al. in [238] and Coconu et al. in [55] used geometry rasterization from different viewpoints to minimize the pre-processing time of their rendering systems exploiting hardware acceleration. Tobor et al. proposed an algorithm using repeatedly hardware Z-buffer rendering from different directions (like in [98]) to compute surfel properties (position, orientation and color). This algorithm uses a hash table structure to efficiently clean the initial point cloud from

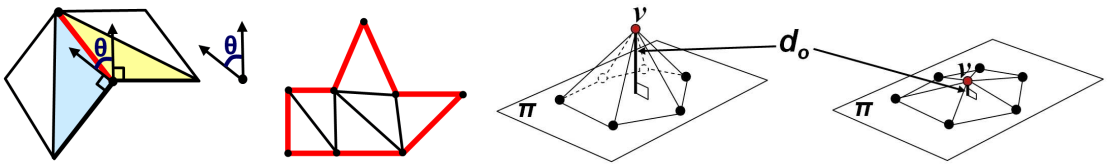
redundancies. Wand et al. in [251] proposed an efficient and real-time sampling algorithm for rendering complex triangulated models through an extension of the z-buffering technique. Their algorithm consists in randomly sampling a variable number of points of small triangles on the fly and rendered them in sampling order. In [250] Wand also proposed an interesting theoretical analysis of the sampling and rendering process, where upper bounds for the rendering time were established. Cohen et al. in [56] presented a triangle rasterization procedure that, given the distance between adjacent samples, covers the entire triangle surface with square tiles adapted to their edges. The triangle plane is then tiled with disks circumscribing the original square tiles. They also proposed a criterion to compute the distance between adjacent samples in order to keep the number of samples under a certain threshold. Stamminger and Drettakis in [225] presented a fast hierarchical sampling scheme in a 2D parametric space for procedural and complex geometric objects. The algorithm iteratively samples the parametric plane on a grid through two spawning vectors, and adaptively refines the grid where further details are needed. The resulting fractal-like sampling pattern is further adapted to reach the boundary of the region to be sampled.

Our algorithm is inspired by this last class of methods since we use a triangle rasterization process, which is driven by an approximation of the triangular surface curvature and by other geometric features. We choose to base our sampling only on the information we can extract from the original mesh, without producing further surface approximation. As in [241] we mainly spread our sample points where the relevant features are found. We do not use a relaxation process, but we directly sample the points along the features that we identify. The rasterization nature of our sampling algorithm makes our surfelization technique easy to implement and flexible. With our technique we are able to capture all the structures present in the original triangular mesh because we strictly follow its triangulation.

4.3.2 Surfelization algorithm

Our surfelization technique aim at sampling an arbitrary triangular surface with surfels by preserving both its geometric properties and its contour properties. This will ensure a correct shape approximation of the original triangulated model. The main idea is to follow the original triangulation by sampling triangles with respect to the following geometric features:

- crease edges and boundary edges;
- flatness of the triangular surface at a vertex.



(a) (b) (c)

Figure 4.4: Geometric features to preserve. Edges to preserve (in red): crease edges (a) and boundary edges (b). (c) Non-flat (left) and flat (right) triangular surface. **[see Color plate D.2]**

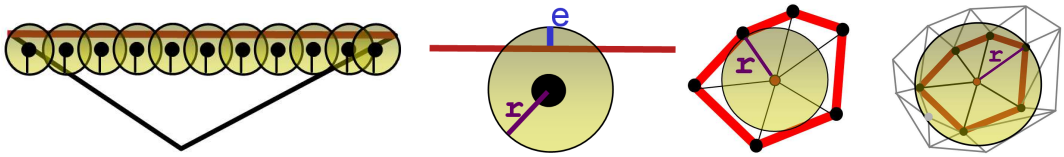
An edge is considered as a *crease edge*, if the dihedral angle θ between two adjacent triangles exceeds a predefined threshold θ_t. An edge is detected as a *boundary edge* if it belongs to only a single triangle. We call these two types of edges *preserved edges*. For each vertex v, we consider all triangles sharing the vertex v as our local triangular surface Ψ. We then compute the average plane π, formed by all the vertices belonging to Ψ [212]. The surface Ψ is considered as *non-flat*, if the orthogonal distance between the vertex v and the plane π exceeds a threshold d_t. The average plane π at the vertex v is computed with respect to the m triangles sharing this vertex. The plane π is defined by its normal vector $\vec{N}$ and the scalar d, which are computed by the formulae:

$$d = \vec{N} \cdot \vec{C} \ , \qquad \vec{N} = \frac{\vec{n}}{|\vec{n}|} \ , \qquad \vec{n} = \frac{\sum_{i=1}^{m} \vec{n}_i A_i}{\sum_{i=1}^{m} A_i} \ , \qquad \vec{C} = \frac{\sum_{i=1}^{m} \vec{c}_i A_i}{\sum_{i=1}^{m} A_i} \tag{4.5}$$

where A_i, c_i and n_i are respectively the area, the centroid and the normal vector of the triangle i. The literature provides other methods to estimate fitting planes (e.g., MLS) and the local surface curvature at a vertex [241]. However, our approach requires only simple floating-point operations. Furthermore, the normal vector and the centroid of each triangle are computed once and reused to estimate the flatness of neighbor vertices, when it is necessary.

The sampling algorithm starts from a seed vertex and samples the underlying local triangular surface near the features with a specific pattern. The remaining planar surface is sampled on a regular grid. Triangles sharing a preserved edge are sampled exactly along the edge line (Figure 4.5a). Surfel disks are then placed along the edge line. The radius and the position of each surfel are selected in order to keep the over edge error e (Figure 4.5b) under a threshold e_{max}. Furthermore, surfels radii must be in a predefined range $[r_{min}, r_{max}]$. When a triangular surface is locally flat, a surfel is centered on each of its vertices parallel to the average fitting plane [212] of its neighbor vertices. To select the radius of the surfel we consider two cases. In the first case where at least one triangle sharing the vertex has a preserved edge (see Figure 4.5c),

the radius of the surfel is set to the minimum length of the connected edges (r in Figure 4.5c). In the other case we set the radius to the maximum length of the connected edges (r in Figure 4.5d). In this last case, we also have to ensure that preserved edges, which belong to some neighbor triangle do not overlap the surfel disk (see Figure 4.5d). In both cases the radius r is cropped to be in the range $[r_{min}, r_{max}]$, and the remaining unfilled area is sampled on a regular grid. Beside the two special cases mentioned above, the remaining triangles are sampled on a regular squared lattice without covering the already sampled area near the preserved features. Basically, this is a rasterization process, where surfel disks are placed onto the center of each output pixel with a radius equal to half the pixel diagonal. This technique is parallelizable and can be easily implemented exploiting the rasterization process of graphics hardware.

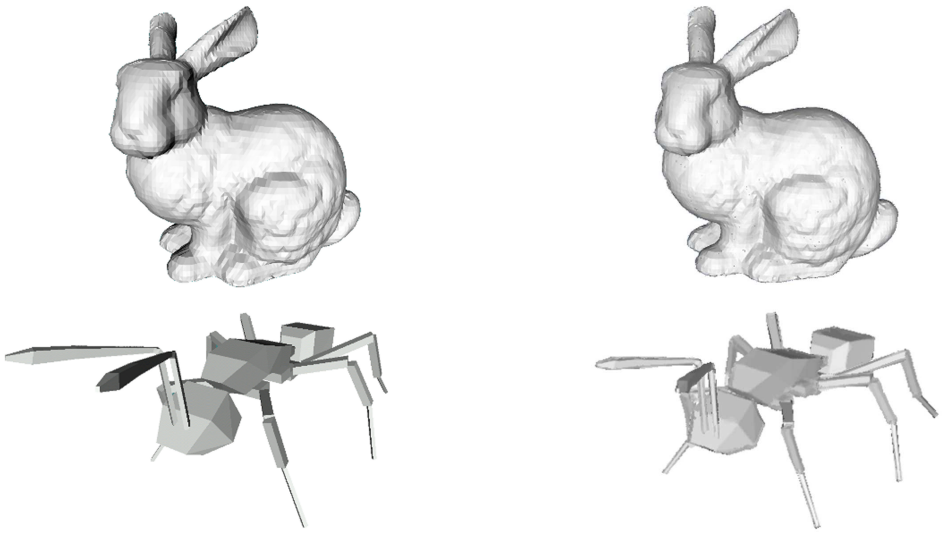


(a) (b) (c) (d)

Figure 4.5: Surfelization near preserved edges (a, b) and of a flat local surface (c, d).

4.3.3 Results

We have converted two retrieval benchmark databases of triangular mesh models. The first database consists of 1814 3D polygonal models provided by the Princeton Shape Benchmark repository [219]. The second database consists of 1841 polygonal models provided by the CCCC repository [17]. These databases contain triangulated 3D models at different resolutions and with very different numbers of triangles. Both databases have been converted in less than two hours including the input/output time. The time required for the surfelization was 98.28 minutes for the first database, and 100.74 minutes for the second database, on a 2.4 GHz Intel Pentium 4 system with 2 GB RAM using a non-optimized C++ version of our program, and software only rasterization. Table 4.1 shows the execution time (in seconds) and the number of surfels produced by our surfelization technique together with the number of geometric primitives (vertices and triangles) of the original triangular mesh. The fourth column shows the number of the geometric features extracted from the triangulated models. All the converted triangular meshes belong to the Princeton database [219].



(a) (b)

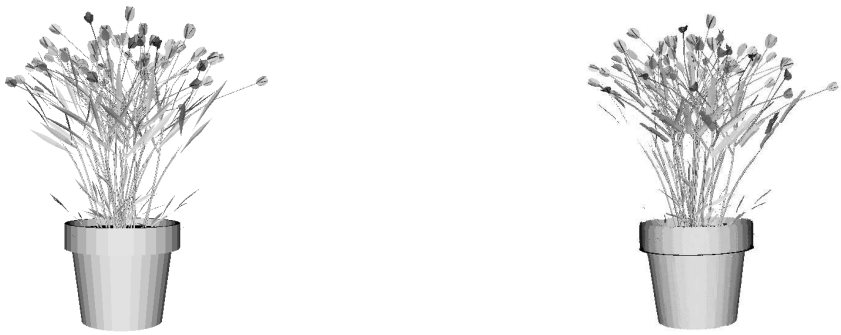
Figure 4.6: Result of the surfelization: the original triangulated models (a) and the corresponding surfel based representations (b). The model of the bunny and of the ant are respectively the triangle mesh *m0* and *m112* of the Princeton database [219]. The information about the triangular meshes and the surfel based model illustrated herein are reported in Table 4.1.

Our surfelization technique is then able to convert large databases of triangle mesh models into databases of surfel based models in a reasonable period of time preserving the geometric features of the original models. Figure 4.6 illustrates the result of our surfelization technique (Figure 4.6b) applied to two different triangular mesh models (Figure 4.6a). The model of the bunny and the ant are respectively the triangle mesh *m0* and *m112* of the Princeton database [219]. Notice that the geometric features of the original triangular meshes (Figure 4.6a) have been preserved in the resulting surfel based models (Figure 4.6b).

Model	Vertices	Triangles	Features	Surfels	Time (sec.)
Bunny	7691	15294	13800	283604	2.18
Ant	298	492	347	14202	0.09
Fish	18409	35270	31563	325851	5.16
Airplane	13908	27087	24185	470660	7.57
Vase of flowers	16590	21551	31102	2008608	25.65

Table 4.1: Surfelization performance. Triangular models from the Princeton DB [219]

Our surfelization technique does not consider the texture properties of a triangle mesh, since the currently available retrieval benchmark databases consist of non-textured models.



(a) (b)

Figure 4.7: Surfelization applied to a model of a vase of flowers: (a) the original triangulated models (model *m1019* of the Princeton database [219]), and (b) the corresponding surfel based representations.

Indeed, shape matching algorithms do not consider texture information of an object, but rather information about its geometry, topology, and in few cases its reflectance. Although textures are not relevant to shape analysis and retrieval, they are fundamental for realistic surfel based model visualization. Future work could be then oriented to capture textures and preserve texture properties, which will also imply to consider the possibility to implement adaptive non-uniform sampling to reproduce special rendering techniques using surfels.

Another drawback of our method is that can generate too many surfels, which are impractical to manage. This problem arises for those triangle meshes, which have many boundary/crease edges, as in the case of the model of a vase of flowers (model *m1019* of the Princeton database [219]) illustrated in Figure 4.7 and reported in the last row of Table 4.1. For 3D models with many sharp edges (e.g., CAD models and models used in architecture) point based representations are not suitable for both rendering and modelling. However, as long as databases of point based models will not become available, we need to work with databases converted from databases of triangular meshes. Thus, to overcome the problem of over-sampling of triangular meshes, we plan to investigate in depth the trade-off between the approximation quality and the number of surfels produced by the surfelization method in relation to the time complexity.

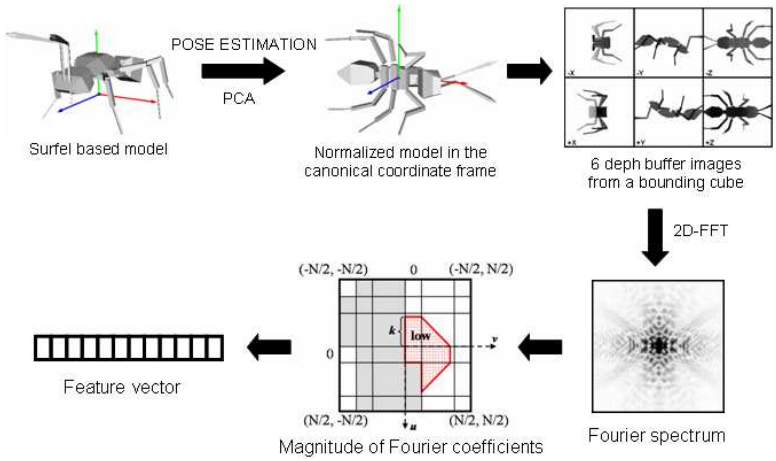


Figure 4.8: Construction scheme of the shape descriptor based on depth-buffer images.

4.4 Shape descriptor for shape matching of point based models

Our retrieval framework aims at efficiently querying a database of point based models. An arbitrary point based model serves as a query key to retrieve similar models from a collection of point based models.

Considering the high retrieval performance obtained by image based shape descriptors in experiments run in [40, 219], and the suitability of this approach to be extended to surfel based models, we propose an image based feature vector for surfel based models. We consider surfels as point based rendering primitives. Our shape descriptor is an extension of the method presented by Heczko et al. in [106] and refined by Vranić in [249]. It is constructed through the algorithm resumed in Figure 4.8. A surfel based model is first transformed into a canonical coordinate frame, which is computed via a pose estimation procedure using Principle Component Analysis (PCA). Six depth-buffer images are then rendered by orthogonally projecting the surfel based model onto the six faces of an appropriate bounding cube. The 2D-FFT algorithm is then applied to all the resulting images to compute their Fourier spectra. The magnitudes of the low-frequency Fourier coefficients of each image are then combined to form a compact feature vector of real numbers.

4.4.1 Normalization (pose estimation)

Surfel based models are given in arbitrary units of measurement and undefined positions and orientations. The normalization procedure aims to transform a model into a *canonical coordinate frame*, so that if one chooses a different scale, position, rotation, or orientation of a model, then its representation in such coordinates remains the same. Moreover, the normalized representations corresponding to different levels of detail of the same model should be similar as much as possible. We normalize each surfel based model by applying a modification of the Principle Component Analysis (PCA) to the set of surfels. Following the usual PCA procedure we compute the covariance matrix C through the following formulae:

$$C = \frac{1}{A(S)} \sum_{i=1}^{N} (\vec{p}_i - \vec{g})(\vec{p}_i - \vec{g})^T A(s_i), \quad \vec{g} = \frac{1}{A(S)} \sum_{i=1}^{N} \vec{c}_i A(s_i), \quad A(S) = \sum_{i=1}^{N} A(s_i), \tag{4.6}$$

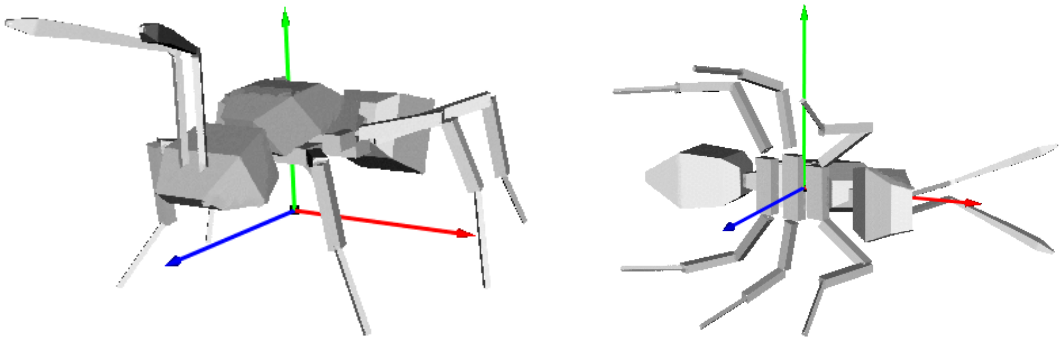
where $\vec{g}$ is the center of gravity and $A(S)$ is the surface area of the entire model, N is the number of surfels, $A(s_i)$ and $\vec{p}_i$ are the area and the center of the ith surfel s_i, respectively. C is a 3x3 symmetric real matrix. The eigenvalues of C are computed and sorted in decreasing order. From C we build the rotation matrix R, which has the normalized eigenvectors as rows with nonnegative elements on the main diagonal. First, we center the surfel based model onto the point $\vec{g}$ by translating the surfel centers $\vec{p}_i$. Then, we apply the transformation defined by the rotation matrix R to the surfel centers and to the surfel normal vectors $\vec{n}_i$. The resulting surfel based model is now expressed in canonical coordinates, which are invariant to translation and rotation. Figure 4.9 shows the surfel based model of an ant (converted from the triangle mesh *m0* of the Princeton database [219]) before and after the normalization procedure.

4.4.2 Feature extraction

Our goal is to extract a set of features describing the shape of an object, which is represented by a surfel based model. These features are stored as a feature vector (FV) of fixed dimension.

We adapted the approach proposed in [249] in which a shape descriptor is extracted from a model in the canonical coordinate frame, through a set of orthographic depth-buffer images.

A *depth-buffer image* is an image, which stores in each pixel a depth value. The depth value of each pixel is computed as the distance between the pixel center and its closest point p_s on the model surface. The point p_s is the closest intersection between the surface and the ray starting from the pixel center and directed further away from the image plane. When an orthographic projection is adopted, the rays are perpendicular to the image plane. In case a perspective



(a) (b)

Figure 4.9: Normalization (pose estimation). Surfel based model of an ant rendered before (a) and after (b) the normalization procedure from the same viewpoint. Notice that in (b) the depicted axis represent the axis of the canonical coordinate frame, which has it origin at the center of gravity of the model.

projection is considered, the rays passing through the center of each pixel start from a unique point behind the image plane, called center of projection, where all rays converge. We adopt an orthographic projection, meaning that the depth value of each pixel represents the distance of the closest surface point to the pixel center. The depth value of a pixel is then coded as a gray scale color. A white pixel codes the maximum distance computed, while a black pixel codes the closest distance computed.

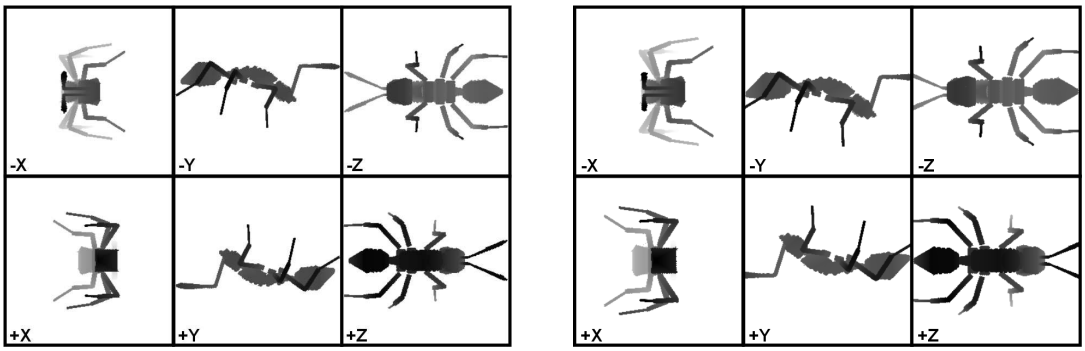
We compute six depth-buffer images by rendering a surfel based model using orthogonal projection on the six faces of a cube establishing an appropriate *region-of-interest*. The size of each depth-buffer image is 256×256 pixels (see Figure 4.10). This operation can be efficiently performed exploiting a graphics hardware z-buffer.

The choice of the *region-of-interest cube* (RIC) is very important and may affect the retrieval performance. We define a canonical bounding cube (CBC) of a surfel based model as a cube whose edges have length $l_c = max\{(\tilde{x}_{max} - \tilde{x}_{min}), (\tilde{y}_{max} - \tilde{y}_{min}), (\tilde{z}_{max} - \tilde{z}_{min})\}$, where $\tilde{x}_{max}, \tilde{x}_{min}, \tilde{y}_{max}, \tilde{y}_{min}, \tilde{z}_{max}, \tilde{z}_{min}$ represent the maximum and the minimum coordinates in the canonical coordinate frame. An object may contain outliers, which affect the dimension of the CBC, and consequently the description power of the depth-buffer images. In fact, suppose considering a model of a car with a long antenna. The antenna will lead to a large CBC, and the resulting depth-buffer images will show the main shape of the car only on a small area of them. Thus, important shape structures of the car may be visualized only by few pixels or even neglected.

To reduce the influence of outliers on the shape descriptor we use a cube, which is aligned with the three canonical axes and centered at the center of gravity $\vec{g}$ of the model (defined in Formula 4.6). The size of the cube, d_f, is proportional to the average distance d_{avg} of the surfels to the center of gravity, $\vec{g}$

$$d_{avg} = \frac{1}{A(S)} \sum_{i=1}^{N} \|\vec{p}_i - \vec{g}\|_2 A(s_i), \quad d_f = f \cdot d_{avg}, \tag{4.7}$$

where f is a fixed ratio, $A(S)$ is the surface area of the entire model, N is the number of surfels, $A(s_i)$ and $\vec{p}_i$ are the area and the center of the ith surfel s_i, respectively. In our experiments we have used $f = 3.6$. Our experience suggests that this approach is superior to using the standard bounding cube, which may sensitively depend on outliers of the model. Thus, this approach leads to a shape descriptor that is more robust with respect to outliers than using a CBC. Figure 4.10 shows two sets of six depth-buffer images computed from the surfel based model of an ant (also shown in Figure 4.12 and 4.6). The depth-buffer images illustrated in Figure 4.10a are computed from a canonical bounding cube (CBC), while the images in Figure 4.10b are computed from our region of interest cube (RIC). Notice that in Figure 4.10b the two antennae of the ant are partially clipped by the region of interest cube.



(a) (b)

Figure 4.10: Depth-buffer images clipped on a canonical bounding cube (a) and on a region of interest cube (b) after the normalization procedure.

The six depth-buffer images are coded by storing a fixed number of Fourier coefficients. The depth-buffer images could also be directly used as a feature vector in the spatial domain, but the resulting feature vector will have a poor effectiveness when compared with other feature vectors using the L_1 or L_2 norm. In fact, components of n-dimensional vectors in the spatial domain are not correlated, and an effective comparison of such vectors requires the definition

of a suitable metric [249]. Thus, the problem of comparison of two images in the spatial domain is transferred to the spectral domain by computing the 2D Fourier spectrum of each depth-buffer image. This approach has several advantages: (1) the uncorrelated components of the spatial features of the images will be correlated in the spectral domain; (2) the distance between vectors of Fourier coefficients can be efficiently computed with the L_1 or L_2 norm, instead of using other complicated distance functions [249]; (3) the Fourier coefficients can also be arranged to generate an FV with an embedded multi-resolution representation.

We apply the 2D discrete Fourier transform (2D-DFT) to each depth-buffer image to obtain the corresponding Fourier spectrum. Let $f(x, y)$ be a 2D function $f : [0, .., N-1]\times[0, .., M-1]\rightarrow I$ representing a gray scale image of $N\times M$ pixels, so that $x = 0, .., N-1$ and $y = 0, .., M-1$. The 2D discrete Fourier transform of $f(x, y)$ is given by the equation

$$F(u, v) = \frac{1}{NM} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) e^{j2\pi(\frac{ux}{N} + \frac{vy}{M})}. \tag{4.8}$$

The coefficients $F(u, v)$, with $u = 0, .., N-1$ and $v = 0, .., M-1$, are complex numbers and are called Fourier coefficients. Thus, they are composed of a real and an imaginary part, i.e., $F(u, v) = Re(u, v) + jIm(u, v)$. The 2D-DFT can be computed in $O(MN(M+N))$ operations, and if N and M are power of 2, its computational complexity can be reduced to $O(MN \log MN)$ by applying the 2D fast Fourier transform (2D-FFT) [96]. Therefore, by keeping our depth-buffer images of size $N\times N$ with N power of 2, we can speed up the computation of their Fourier coefficients.

The feature vector is then built by storing, for each processed image, the magnitudes of low frequencies Fourier coefficients $|F(u, v)| = [Re^2(u, v) + Im^2(u, v)]^{1/2}$, since they incorporate the most meaningful information about the main shape of an image [96]. We consider only a small number of Fourier coefficients by following the method described in [249]. This approach reduces the storage space of the feature vectors and makes their comparison efficient.

The structure of the Fourier spectrum of an $N\times N$ image is shown in Figure 4.11. The Fourier spectrum is commonly centered in order to have the origin of its (u, v)-coordinate system at the center of the spectral image. This facilitates it comprehension [96]. Figure 4.11a shows the location of the low-frequency and the high-frequency coefficients of a Fourier spectrum. Notice that the low-frequency coefficients, in which we are interested, are located in the center of the image. Moreover, the structure of the magnitudes of the Fourier coefficients for a squared image is symmetric. This is a consequence of the fact that a complex number and its complex conjugate have the same magnitude. Figure 4.11b shows the distribution of symmetrical magnitudes of the

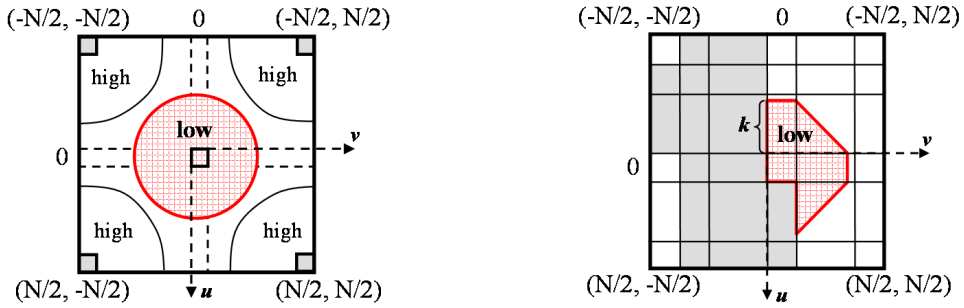
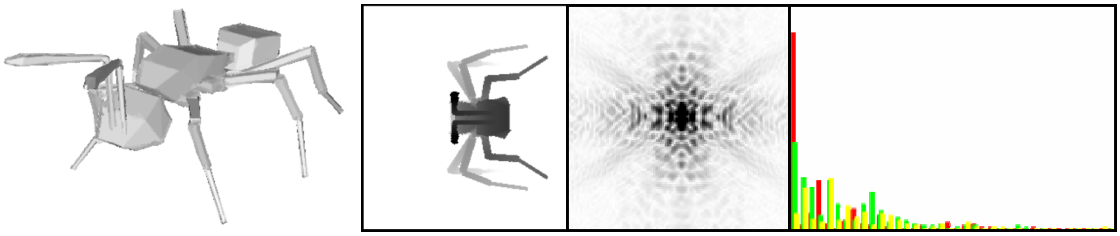


Figure 4.11: Structure of the Fourier coefficients in a centered Fourier spectrum of a $N \times N$ image. (a) The location of the low and high frequency coefficients are denoted by the bounded areas marked with "low" and "high", respectively. (b) Distribution of symmetrical magnitudes of the Fourier coefficients. The area delimited by the polygonal boundary represents the set of coefficients stored in the feature vector. The coefficients depicted in gray color can be computed by using the symmetry property of the 2D-DFT.

Fourier coefficients of an $N \times N$ image. The coefficients depicted in gray color can be computed by using the symmetry property of the 2D-DFT. The area delimited by the polygonal boundary represents the set $\Omega_k^{(i)}$ of coefficients stored in the feature vector for the depth-buffer image i. The total number of coefficients contained in $\Omega_k^{(i)}$ is $k^2 + k + 1$. Thus, our feature vector will contains $6(k^2+k+1)$ magnitudes of low-frequency coefficients. There is a trade-off between the required storage space, computational complexity, and the resulting retrieval performance [249]. According to experimental results [249] we consider $k = 8$, which results in a feature vector of dimension 438 (73 real numbers for each image). Figure 4.12 shows the surfel based model of an ant with a depth-buffer image, its Fourier spectrum image, and the magnitude of its Fourier coefficients.

An important property of an FV is to embed a multi-resolution feature representation. Given two FVs $\vec{v} = (v_1, .., v_n)$ and $\vec{\hat{v}} = (\hat{v}_1, .., \hat{v}_m)$ of different dimensions (suppose $n > m$) an *embedded multi-resolution feature representation* holds if the vector $\vec{v}$ contains all the lower-dimensional vectors of the same type, i.e., $v_i = \hat{v}_i \; \forall \; i : 1 \leq i \leq m$. This property allows to compare an FV of high resolution ($\vec{v}$) with an FV of low resolution ($\vec{\hat{v}}$) without the need to separately store a low resolution vector of the same dimension. In fact, the comparison can be performed considering the first m components of $\vec{v}$.

To obtain a multi-resolution feature representation we arrange the selected 438 Fourier coefficients of the six depth-buffer images as follows. The main idea is to pack together the Fourier



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

Figure 4.12: Surfel based model (a) and the extracted features (b, c, d). From a surfel based model (a) six depth-buffer images are extracted from its region-of-interest cube. The Fourier spectrum (c) of each depth-buffer image (b) is then computed. A set of magnitudes of its low-frequency coefficients (d) is then stored as feature vector.

coefficients of different depth-buffer images, which are associated to the same pixels. Considering the central Fourier spectrum of each depth-buffer image, we start from the central pixel $|F(0,0)| = c^{(1)}_{0,0}$ of the first image, and we store the first vector of six components $(c^{(1)}_{0,0}, .., c^{(6)}_{0,0})$ in the FV. Then, we consider the next selected pixel $|F(u,v)| = c^{(1)}_{u,v}$ and we enqueue the corresponding vector of six coefficients $(c^{(1)}_{u,v}, .., c^{(6)}_{u,v})$ to the FV, until all the selected 73 pixels are processed.

The resulting feature vector is invariant with respect to similarity transforms (rotation, translation, and scaling - see Section 4.4.1), robust with respect to outliers and possesses an embedded multi-resolution feature representation. Furthermore, our feature vector allows to achieve an efficient and effective shape similarity search by using the L_1 or L_2 norm.

4.4.3 Results

The retrieval effectiveness of the presented method for matching shape of surfel based models is evaluated by using a precision-recall diagram (see Section 4.1.2). We compare our method with other retrieval methods developed for polygonal mesh models. The retrieval experiments are run on the polygonal models of the original Princeton Shape Benchmark test database [219], and on the corresponding surfel based representation. The considered feature vectors are associated to a label and have the following characteristics:

DBD: it is based on depth-buffer images [249]. The algorithm to extract this FV is similar to our algorithm presented in Section 4.4.2, but applied to a triangle mesh. In our experiment we consider depth-buffer images of 256×256 pixels and FV of dimension 438 ($k = 8$, see Section 4.4.2).

RSH: it is a ray based feature vector with spherical harmonics representation [247] . This method is based on a function $r(\vec{u})$ defined on a unit sphere, which associates a distance to a sampled direction $\vec{u}_i$. This distance is calculated from the origin of the canonical coordinate frame to the farthest intersection point between the triangulated surface and a ray emanated from the origin in the direction $\vec{u}$. The function $r(\vec{u})$, reconstructed by using spherical harmonics as basis functions, is properly sampled in order to apply the 3D fast Fourier transform. The magnitude of the computed low-frequency Fourier coefficients are then combined to form an FV, which embeds a multi-resolution representation.

EDTorig: it is based on exponentially decaying Euclidean distance transform (EDT) [94]. This method describes an exponentially decaying Euclidean distance function define on voxel grid with spherical harmonics. A triangle mesh is voxelized in a normalized 3D grid of dimension $2R \times 2R \times 2R$. The triangle mesh is moved, so that the center of mass is placed at the point (R, R, R). Then, the model is scaled so that the average distance of occupied voxels to the center of mass is $\frac{R}{2}$. An exponentially decaying Euclidean distance function is defined on a set of points of the voxel grid with length $l \leq R$. The resulting function is expressed in spherical coordinates and restricted to a set of spherical functions, each associated to a radius. Each spherical function is then described as a sum of spherical harmonics, each identifying a frequency. The FV is constructed by storing a combination of the low frequency band energies of the extracted spherical functions. The resulting FV is by construction invariant with respect to rotation, translation and scaling.

SIL: it is based on 2D silhouettes [106, 249], which are extracted from images rendered with orthogonal projection on the faces of the canonical bounding cube (CBC - defined in Section 4.4.2). A triangle mesh is first normalized using PCA. Then, three images are rendered using orthogonal projection onto the three faces of the CBC, each orthogonal to one of the principal axes. The silhouette images are computed by filling the interior of the projected triangles. The contours of the silhouette images are then extracted and described as contour functions. A contour function computes for each sample point (equally spaced) on the contour its distance to the origin of the silhouette image. The FV is computed by storing the magnitudes of low-frequency Fourier coefficients of the three extracted contour functions.

DBSBM: it represents our FV extracted from surfel based models by using our region-of-interest cube. In our experiment we consider depth-buffer images of 256×256 pixels and FV of dimension 438 ($k = 8$, see Section 4.4.2)

FV	Dim	$\bar{P}_{50}$	$\bar{P}_{100}$	BEP	RP	NN
DBD	438	53.6	38.6	45.0	34.5	60.9
EDTorig	544	49.6	35.1	42.4	31.7	56.8
SIL	300	50.1	36.5	43.9	32.7	55.7
RSH	136	41.4	28.0	35.5	25.8	52.6
DBSBM	438	51.6	36.9	43.7	33.2	60.2

Table 4.2: Comparison of the retrieval effectiveness of different feature vectors (FVs). The column labelled as "Dim" reports the dimension of the FVs. The other columns report different retrieval effectiveness measures defined in Section 4.1.2.

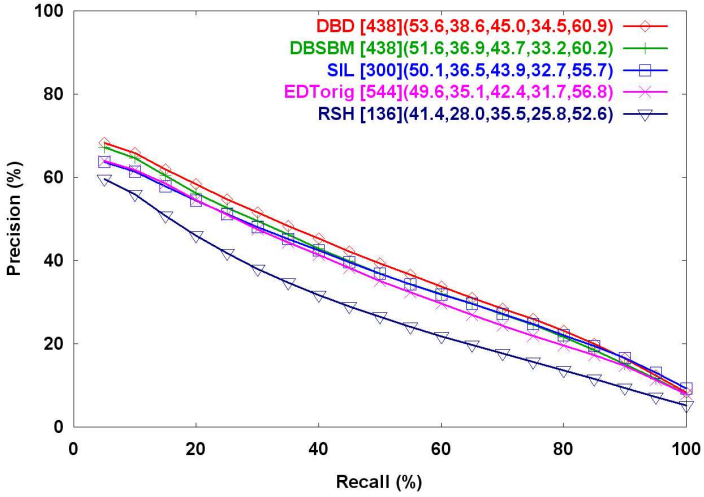


Figure 4.13: Precision-recall diagram computed for different feature vectors (DBD, EDTorig, SIL, RSH) of the original triangles meshes in comparison to proposed feature vector extracted from the corresponding surfel based models (DBSBM). The dimensions of the feature vectors are given in brackets. Between round parentheses different retrieval effectiveness measures defined in Section 4.1.2 are reported, i.e., ($\bar{P}_{50}$, $\bar{P}_{100}$, BEP, RP, NN). The same measure are clearly reported in Table 4.2. [**see Color plate D.3**]

The retrieval effectiveness of the FVs mentioned above is compared through the evaluation measures defined in Section 4.1.2. These measures are reported in Table 4.2. Figure 4.13 shows the precision-recall diagram illustrating the comparison of the retrieval effectiveness curves of our FV computed for surfel based models with other FVs extracted from the original triangu-

lated models. The diagram shows that the retrieval performance of the depth-buffer image-based descriptor for surfel based models is approximately the same as the performance of the corresponding descriptor for polygonal mesh objects. The slight difference between these two FVs of the same nature (but applied to different model representations) can be explained with the presence of approximation artifacts in the surfel based representation. In fact, it is well known that point based representations are not suitable for both rendering and modelling models with many sharp edges (e.g., CAD models and models used in architecture), which are numerous in the Princeton database. Moreover, even though our surfelization method (Section 4.3) preserves the geometric features of a triangular mesh, for models with many sharp edges it may generates a huge number of surfels, which are still not manageable by the adopted point based rendering system (PointShop3D [268]). In order to reduce the total number of surfels, our surfelization algorithm can be configured to produce a simplified version of surfel based models, which may have surfels with large radii. This effect produces visualization artifacts, which may affect the quality of the rendered depth images and of the final feature vector. This problem can be overcome either by improving the attainable size of surfel based models, or by applying an advanced simplification method [184] to the high resolution surfel based models of the database. However, the latter approach is very expensive in terms of execution time. Improvements of the surfelization algorithm can also help to obtain an optimal ratio between rendering quality and number of surfels. This problem will be considered in future work.

4.5 Topology matching

Topology matching of surfaces has been considered by several authors. In fact, it is particularly useful for interactively searching 3D objects, since the result of a topological similarity search well fits human intuition. The topological features of an object are related to its shape properties that do not change under certain type of deformations [69, 105]. Allowed deformations imply isometric transformations (see Appendix A), which prevent a deformation from cutting segments and piercing holes. Isometric transformations are length preserving transformations, meaning that they not generate shearing, stretching or shrinking on the surface of object during deformation. In some cases, deformations of objects can be approximated by isometric transformations, e.g., bending limbs of human body models, moving components of articulated objects, and varying face expressions. We call *bending deformations* those isometric transformations, which deform an objects by bending its surface.

Bending deformations are interesting in many applications since they are strictly related to the human perception of similarity between objects. In fact, imagine to search on a big collection of objects of various types those that are similar to a query model of a human body. Our natural perception of similarity will lead us to pick those models with the same topological characteristics of the query model. Thus, models of human/humanoid bodies, even though represented in different postures with respect to the query model (e.g., bent down bodies or bodies with bent limbs), will be preferred to other geometrically similar models. In this context, geodesics play a prominent role, due to their invariance with respect to isometric transformations. For this reason geodesics have been widely used as a basis for various topology matching and object recognition techniques.

The *Morse theory* is used to analyze the topology of a manifold. It is based on studying a suitable differentiable function, called *Morse function*, defined on a manifold. In particular, it is focused on the extraction of critical points of this function with specific characteristics, which directly reflect specific types of changing in the topology of a manifold. The Morse theory allows to handle decompositions of manifolds and to obtain substantial information about their homology [69, 72, 220]. It is also used to conceive techniques for identifying topographical features, control their simplification [12], and organize them into a multiresolution hierarchy [32]. Formally, let $f : M \rightarrow \mathbb{R}$ be a real-valued smooth function on a differentiable manifold M, a point $p \in M$ is called *critical point* if $\nabla f(p) = 0$, and $f(p)$ is called *critical value*. A critical point p is called *degenerate critical point* if the Hessian matrix in p (the matrix of second partial derivatives of f) is singular, and *non-degenerate critical point* if it is non-singular. The function f is called *Morse function* if it has no degenerate critical points. Suitable Morse functions for surface topology analysis are: the height function, the geodesic distance function [107], and extended versions of the Euclidean distance function [13]. However, the choice of a suitable Morse function depends on the application, and is a delicate task. In fact, an arbitrarily chosen Morse function can yield a complicated arrangement of many critical points, which can be impractical for many applications. A discussion of this problem can be found in [175]. Critical points carry important topological information, because they describe how the topology of a manifold change. There are three types of critical points called: *minima, saddles, and maxima.* They are associated to the indices, 0, 1, and 2, respectively. The index of a critical point p is defined as the number of negative eigenvalues of the Hessian matrix of the Morse function f in p. Intuitively, this index identifies the number of independent directions around p in which f decreases. The critical points of a Morse function are used to extract topological graphs, which

compactly describe the topology of a manifold. For example, Nackman in [171] presented a critical point graph (CPG) to compactly describe critical points of a two-dimensional Morse function. A more detailed review of similar techniques can be found in [27].

The most commonly used structure related to the Morse theory that is used for topology analysis and matching is the Reeb graph [13, 25, 27, 50, 107, 237, 240], which is a particular CPG. A suitable Reeb graph provides a compact graph description of the topological skeleton of a manifold, and can be used for topology matching [107], compression, simplification and segmentation [258]. Let $h : M \rightarrow \mathbb{R}$ be a continuous function on a manifold M. Let $M/\sim$ be the quotient space of M defined by the equivalence relation $\sim$, i.e., the set $M/\sim = \{[p] : p \in M\}$ of equivalent classes $[p]$ generated by $\sim$, where $[p] = \{q \in M : q \sim p\}$. The *Reeb graph* of M with respect to the function h is defined as the quotient space of $M \times \mathbb{R}$ defined by the equivalence relation $\sim$, such that $\forall\ x_1, x_2 \in M,\ (x_1, h(x_1)) \sim (x_2, h(x_2))$ holds if and only if:

- $h(x_1) = h(x_2)$, and
- x_1 and x_2 are in the same connected component of $h^{-1}(h(x_1))$.

In practice, a Reeb graph is commonly defined with respect to a Morse function h, and its nodes denote the critical points of h, and its edges represent the connected components of h^{-1} linking the critical points of h. Reeb Graphs and extended multiresolution Reeb graphs have been extensively used for topology matching and analysis [13, 27, 50, 107, 237, 240].

Hilaga et al. [107] presented a technique to match the topology of triangulated models, by comparing Multiresolutional Reeb Graphs (MRGs). A MRG is constructed by using a suitable discrete approximation of the function $\mu(v) = \int_{p \in S} g(v, p) dS$ defined on the triangulated surface S of the model, where $v, p \in S$, and $g(v, p)$ is the geodesic distance between v and p on S. The geodesic distance $g(v, p)$ is computed by applying Dijkstra's algorithm on a graph constructed from the vertices and the edges of the mesh. To reduce the approximation error, which is affected by the triangulation resolution, the triangles are resampled until all edge lengths are less than a selected threshold. Moreover, additional shortcut edges and vertices are added in order to connect neighbor vertices that are not directly connected via an edge of the mesh. The function $\mu(v)$ is discretized, and then normalized with respect to its minimum and maximum value. The construction of the MRG begins with the construction of a Reeb graph at the finest resolution (selected by the user). A Reeb graph is constructed by first dividing the domain of the function $\mu(v)$ into k μ-ranges. Then, each triangle lying over the boundaries of the μ-range is split in sub-triangles, so that every triangle of the mesh belongs to only one μ-range. In

each μ-range a set of connected triangles constitutes a node of the Reeb graph. Two triangles are connected if there exists a path of edges (belonging to the same μ-range) connecting one of their vertices. Two nodes belonging to different μ-ranges are connected by an edge of the Reeb graph if one of their triangles shares a vertex. A MRG is then constructed in a bottom-up fashion, by starting from the finest resolution Reeb graph, and progressively unifying adjacent nodes at each new coarse level. The algorithm for matching two MRGs adopts a complicated coarse-to-fine strategy, which searches with a heuristic the node pairs providing the largest value of similarity while maintaining topological consistency. The similarity between two nodes is then computed by comparing their attributes, i.e., the areas and the lengths of the triangles associated to the nodes.

Heuristics for comparing graph based representations extracted from 3D model are of particular interest in CAD applications, where graph descriptions of CAD models embedding topological and geometrical information are used for both global matching and partial matching of object components [22, 43, 53, 83, 118, 221]. For example, in [83] an attributed graph representing the topology of a 3D object is constructed from its surface parts, and their edges. The surface parts define the nodes of the graph and their edges define the link of the graph. Each node of the graph stores the geometric characteristic of the associated surface part.

Skeleton representations of 3D models turned out to be effective tools for comparing topology of 2D/3D objects [26, 60, 214, 227, 231, 245]. In particular, they are useful for both global and partial matching of the topology/geometry of objects, and in shape modelling [26]. Sundar et al. in [227] presented a technique, which compares 3D models by comparing directed acyclic graphs (DAGs) obtained by 3D skeletons. A 3D skeleton is extracted from a triangular mesh, by using a volumetric thinning approach. The skeleton is then simplified by using a clustering procedure, which makes it robust with respect to small perturbations in the geometry. A DAG is then constructed from the simplified skeleton, by ensuring that the shape information carried by the skeleton is preserved. This procedure is achieved by avoiding that minor changes in the position of the skeletal points do not produce graphs with big differences. Each node of the DAG corresponds to a segment of the skeleton. The edges correspond to joints in the skeleton. Each node incorporates a topological feature vector coding information about the associated point cloud. This point cloud is obtained from the volume thinning used to generate the skeleton segment associated to the node. The resulting feature vectors are used for partial matching, and to compare nodes in the global matching. The problem of matching skeletal DAGs is tackled by reformulating it as bipartite graph matching problem. A heuristic is developed to constrain

the bipartite matching algorithm to hierarchical relationships in the skeletal DAGs. As in [107], their approach consists of a coarse-to-fine strategy. Similarities computed at higher levels of the tree are used to constrain similarities computed at lower levels.

Other similar techniques based on 3D skeletons were presented for topology matching and analysis [61]. In particular, methods based on *curve skeletons* matching obtained promising results in retrieving 3D objects [60]. We refer to [61] for a theoretical description and an extensive review of these methods.

Bending deformations invariant shape descriptors were conceived for matching both geometry and topology of 3D objects [18, 21, 85, 87, 103, 145, 162]. They have applications in retrieval of articulated models [18, 145], in medical applications for comparing human organs, in 3D face recognition [34], and in object recognition and classification [21, 103, 170].

Elad and Kimmel [85, 87] proposed a canonical bending invariant representation for isometric triangulated surfaces. Their technique consists of a transformation process to turn the bending invariant matching problem into a rigid object matching problem in the Euclidean space. The transformation process is composed of two main steps. In the first step, they constructed a geodesic distance matrix considering the geodesic inter-distances between n random vertices uniformly distributed on the surface [87]. In [85] the n random vertices are selected by using a farthest point sampling. Starting from a random vertex an iterative algorithm chooses at each iteration the vertex with the longest geodesic distance from the already selected vertices. Geodesic distances are measured using the fast marching method applied to a triangulated domain [197]. In the second step, they computed the canonical coordinates of the surface in a low dimensional Euclidean space $\mathbb{R}^m$. This process is achieved by applying multi-dimensional scaling (MDS) techniques on the geodesic distances matrix. The canonical coordinates are uniformly scaled to a unit bounding box, centered and oriented using the second order moments (the eigenvalues in the classical MDS). The matching problem is then reduced to the problem of matching rigid objects represented in canonical coordinates in the Euclidean space $\mathbb{R}^m$. In the Euclidean space the canonical surfaces are then matched by considering the first few moments (less than 15) of each surface. The Euclidean distances between each pair of such moment vectors is then calculated to obtain the final dissimilarity measure between two objects [88].

Mémoli and Sapiro in [162, 163] compared point clouds (representing dense samplings of manifolds) by computing a distance function, which bounds the Gromov-Hausdorff distance between two point clouds. Let X and Y be two compact subsets of a metric space (Z, d_z), the

symmetric *Hausdorff distance* between X and Y is defined as:

$$d_H^Z(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d_z(x,y), \sup_{y\in Y}\inf_{x\in X} d_z(y,x)\}. \tag{4.9}$$

The Gromov-Hausdorff distance is an extension of the symmetric Hausdorff distance for comparing isometric metric spaces and it is defined as follows:

$$d_{GH}(X,Y) = \frac{1}{2}\inf_{\substack{Z\\ f:X\to Z\\ g:Y\to Z}} d_H^Z(f(X),g(Y)) \tag{4.10}$$

where $f : X \to Z$ and $g : Y \to Z$ are isometric embeddings into the metric space Z. Thus, the Gromov-Hausdorff distance measures how far X and Y are from being isometric. In fact, if X and Y are isometric, $d_{GH}(X,Y) = 0$. For compact metric spaces the Gromov-Hausdorff distance can also be formulated in terms of distortion of transformations [35, 163]. Consider the two compact metric spaces (X, d_X) and (Y, d_Y). The *distortion* of a transformation $\phi : X \to Y$ between X and Y is given by

$$dis(\phi) = \sup_{x,x'\in X} |d_X(x,x') - d_Y(\phi(x),\phi(x'))|. \tag{4.11}$$

The Gromov-Hausdorff distance $d_{GH}(X,Y)$ between X and Y can then be rewritten as follows [35, 163]:

$$d_{GH}(X,Y) = \frac{1}{2}\inf_{\substack{\phi:X\to Y\\ \psi:Y\to X}} \{diss(\phi), diss(\psi), diss(\phi,\psi)\}, \tag{4.12}$$

$$diss(\phi,\psi) = \sup_{\substack{x\in X\\ y\in Y}} |d_X(x,\psi(y)) - d_Y(y,\phi(x))|, \tag{4.13}$$

where $diss(\phi,\psi)$ measures how far the transformations ϕ and ψ are from being the inverse of each other.

Given a point cloud X obtained by sampling uniformly a manifold, Mémoli and Sapiro [162, 163] computed a reference subset $X_m \subset X$ of lower cardinality $m = |X_m| \leq |X|$ that covers the manifold through the union of balls centered at each point of the subset X_m. X_m is computed with a farthest point sampling technique (see Section 4.6.1.1), which considers estimates of geodesic distances between the points of X_m. The estimates of the geodesic distances are computed on a neighborhood graph extracted from the point cloud [234]. The points of X_m have a fixed separation between them, so that they are not too close to each other. A geodesic distance matrix (GDM) $M_X = \{m_{ij}^X = d_X(x_i,x_j) : x_i, x_j \in X_m, 1 \leq i,j \leq m\}$ is computed by considering the geodesic distance $d_X(x_i,x_j)$ between each possible point pair $(x_i,x_j) \in X_m^2$.

The two manifolds represented by the point clouds X, Y are then compared by calculating an approximation of $d_{GH}(X, Y)$, which neglects the term $diss(\phi, \psi)$ in Equation 4.12 and is defined as the *permutation distance*

$$d_p(X_m, Y_m) = \frac{1}{2} \min_{\pi \in \mathcal{P}} \max_{1 \leq i,j \leq m} \frac{1}{2} |m_{ij}^{(X_m)} - m_{\pi_i \pi_j}^{(Y_m)}| \tag{4.14}$$

between their reference subset X_m, Y_m of m points, where $m_{ij}^{(X_m)} = d_X(x_i, x_j)$ is the element of the GDM constructed from X_m, $m_{\pi_i \pi_j}^{(Y_m)} = d_Y(y_{\pi_i}, y_{\pi_j})$ is the element of the GDM constructed from Y_m selected according to the permutations π. π belongs to the set $\mathcal{P}$ of all permutations of matrix indices $\{1, .., n\}$. Then, the function $\tilde{d}_p(X_m, Y_m) = \max(d_p(X_m, Y_m), d_p(Y_m, X_m))$ is computed to have a symmetric distance function, which bounds the Gromov-Hausdorff distance between the two original point clouds X and Y. To reduce the computational complexity of this approach the authors proposed a heuristic minimizing the approximation error point-wise, by considering sub-matrices of geodesic distances properly extracted from the main GDMs. This algorithm progressively constructs approximations of $\tilde{d}_p(X_m, Y_m)$ using subsets of X_m and Y_m (i.e., sub-matrices of their GDMs) constructed with the farthest point sampling strategy.

Bronstein et al. [35] proposed a method to approximate the Gromov-Hausdorff distance between two smooth surfaces X and Y, which consists in finding the minimum-distortion map between X and Y. This problem was approached in a MDS-like spirits by sampling, as in [162, 163], the two surfaces X and Y into two sets of m points, X_m and Y_m respectively, and minimizing the function:

$$F(y'_1, .., y'_m) = \sum_{i,j} w_{ij} |d_X(x_i, x_j) - d_Y(y'_i, y'_j)|^p, \tag{4.15}$$

where d_X and d_Y indicate the geodesic distances computed on the triangular surfaces respectively approximating X and Y by using the Fast Marching method presented in [197]. w_{ij} are weights representing the set of consistent pairs (for partial matching), and p is a parameter chosen by the user, which influences the robustness of the computation. The minimum of the function $F(y'_1, .., y'_m)$ was searched over the points y'_i lying on the triangle mesh induced by the sample points in Y_m. The geodesic distances $d_Y(y'_i, y'_j)$ on the triangle mesh were approximated from the distances $d_Y(y_i, y_j)$ computed between the sample points in Y_m. The minimum of $F(y'_1, .., y'_m)$ was then computed by applying a multi-resolution approach based on the steepest descent method, which was intended to avoid local minimal, though this is not guaranteed. This approach was also used to compute an approximation of the following *non-symmetric*

partial embedding distance

$$d_{PE}(X, Y') = \frac{1}{2} \inf_{\psi: Y \to X} \sup_{x,x' \in C} |d_Y(y, y') - d_X(\psi(y), \psi(y'))|, \tag{4.16}$$

where $C = \{(y, y') : d_{Y'}(y, y') = d_Y(y, y')\} \subseteq Y \times Y$ contains only the pairs of points for which the metric $d_{Y'}$ defined on the patch Y' is consistent with that defined on Y (d_Y). Intuitively, $d_{PE}(X, Y')$ indicates how similar a patch Y' of the surface Y is to the surface X.

Hamza and Krim [103] proposed a statistical approach for comparing 3D objects, which is based on a shape descriptor describing the distribution of the global squared geodesic distance function on the surface of an object. Similarly to Hilaga et al. [107], they considered a discrete approximation of the global squared geodesic distance function $\mu(v) = \int_{p \in S} g(v, p)^2 dS$ defined on the triangulated surface S of the model, where $v, p \in S$, and $g^2(v, p)$ is the squared geodesic distance between v and p on S. In order to reduce the computational cost of the algorithm, they computed $\mu(v)$ by considering as points $v, p \in S$ only the centroids of a subset $S' \subset S$ of m triangles of the mesh S. S' was obtained by reducing the number of triangles of S to m while preserving the its overall shape. The geodesic distance $g(c_j, c_j)$ was then approximated by $g_{S'}(c_j, c_j)$, which was computed by applying Dijkstra's algorithm on a neighborhood graph constructed from the m triangle centroids $(c_i, c_j, 1 \leq i, j \leq m)$, the vertices and the edges of the mesh S'. dS was approximated as the area of the triangle containing c_i. $\mu(c_i)$ were then assumed as random variables with a common probability density function ρ, which was considered as statistical shape descriptor. ρ was estimated as the sum of m Gaussian kernels each centered at $\mu(c_i)$. The dissimilarity between two objects was then calculated by computing the Jensen-Shannon divergence between the corresponding statistical shape descriptors.

4.6 Geodesics for topology matching of point based models

The study of surface topology has a relevant impact in shape analysis and understanding. It has application in different fields such as: object recognition, surface modelling and compression. Intuitively, topology is related to shape properties that are invariant with respect to isometric transformations, i.e., those properties which do not change under a deformation, like bending.

This section focuses on matching two point based models with a similarity function, which is able to consider both topological and geometric similarities of the two models. In particular,

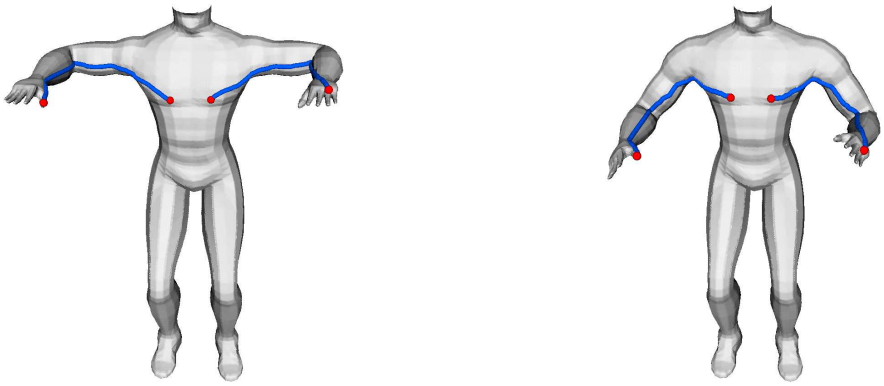
we aim at conceiving a method that is able to capture the similarities between the geometric structures of two objects that are related to their intrinsic topological features.

The problem of comparing two objects is related to the problem of finding the best alignment between their geometric and topological features. The goal is to find the rigid transformation (rotation and translation) minimizing a dissimilarity function, which considers the geometric/topological features of the two objects. This problem was studied in depth in 3D scanning and 3D reconstruction [65, 90, 144, 205, 218]. Most of those techniques focus on a geometric comparison of objects by aligning 3D points, geometric primitives, and geometric features with rigid body transformations.

A geometric comparison of two objects can be achieved by considering the Euclidean distances between pairs of points, which are densely sampled on their surfaces for capturing all their geometric structures. Multidimensional scaling (MDS) methods [30] using this approach were proposed for object recognition [85, 87, 234]. In practice, the problem of comparing two objects, can be rephrased as the problem of matching the corresponding distance matrices, which store the inter-distances between points appropriately sampled on their surfaces. In this context, the comparison of distance matrices can be formulated as an aligning problem, but in a metric space of reduced dimension. In fact, the goal of a matrix comparison is to find a permutation between the indices of the rows and the columns of the matrices, which minimizes a dissimilarity function defined on the matrix elements.

If we consider the Euclidean distance, the distance matrices only provide information about the rigid similarity of the objects. Therefore, if two distance matrices of the same size are equal up to a permutation of their row and column indices, it means that a rigid isometry between the two associated point clouds minimizing a dissimilarity function can be found, thus the two objects are similar. This concept of similarity can be extended to make it invariant to isometric transformations by using geodesic distances instead of Euclidean distances.

Geodesics are invariant with respect to isometric transformations of surfaces. This concept is illustrated in Figure 4.14, where two geodesic curves are computed on two surfel based models representing the same human body in different postures. The geodesic distances, i.e., the lengths of the depicted geodesic curves, did not change despite of changing in the model geometry. Notice that the topology of the model did not change. Thus, geodesics can be exploited to capture the topological structure of a surface and to conceive a topology matching algorithm for surfaces.



(a) (b)

Figure 4.14: Invariance of geodesics with respect to bending deformations. Figure (a) and (b) show two geodesics computed on two surfel based models representing the same human body in different postures. The geodesic distances do not change despite of changing in the model geometry.

Mémoli and Sapiro in [162] demonstrated that using geodesic distance matrices (GDMs) computed from set of densely sampled point clouds it is possible to bound the Gromov-Hausdorff distance between two bounded surfaces, with a distance function defined on the elements of the GDMs (see also the review in Section 4.5). Thus, it is possible to define a similarity value, which considers both geometric and topological features of the objects. The theoretical foundations presented in [162] for bending invariant recognition of point clouds, contributed to the motivation of our method for matching point set surfaces.

We present a method for comparing the topology and the geometry of two point based surfaces by analyzing the distribution of the geodesic distances between random point pairs computed on the two surfaces. The main idea of our technique is expressed by the scheme shown in Figure 4.15.

A set of M random points evenly distributed on a point set surface are selected as reference points (see Section 4.6.1). We experimented with two sampling approaches to produce points with a Poisson disk distribution: (1) a dart-throwing algorithm, and (2) the farthest point sampling algorithm.

Once a set of reference points is selected, a normalized GDM is computed by storing in each element the geodesic distance between the two corresponding reference points (see Section 4.6.2). The geodesic distances are computed with the algorithm presented in Section 2.3.

The similarity of two point based models is then computed by comparing the associated

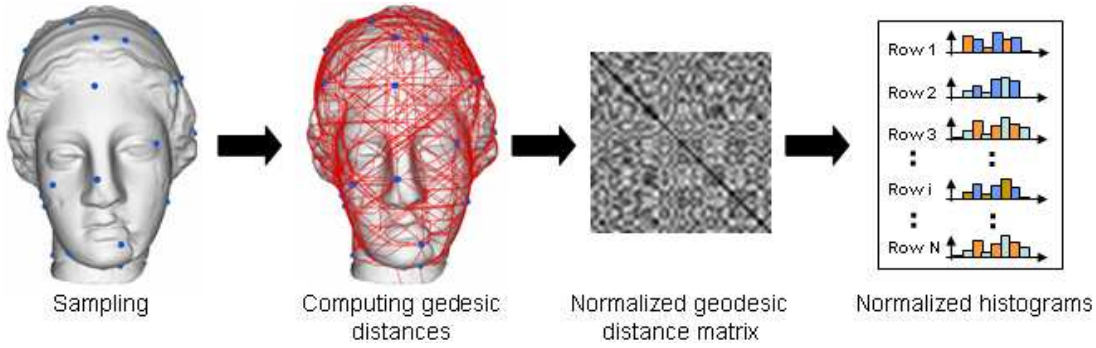


Figure 4.15: Construction scheme of the shape descriptor based on geodesic.

GDMs by using a histogram based approach (see Section 4.6.3). A histogram of the geodesic distances for each row of the matrix is generated. Similar histograms in the two matrices are compared with a suitable metric, which is used to calculate the dissimilarity value between two objects.

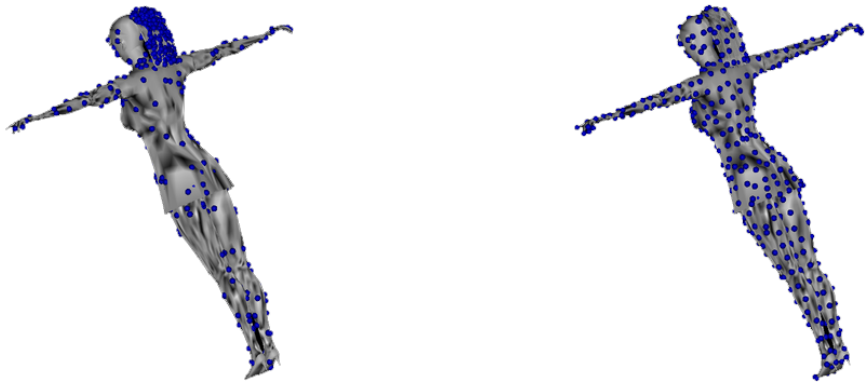
Our geodesic based matching algorithm demonstrated a good retrieval effectiveness for classes of deformed and isometric objects. Moreover, unlike the image based method presented in Section 4.4, it does not require any pose normalization step, since the invariance with respect to similarity transforms is already provided by the definition of the shape descriptor. In fact, the geodesic distances between selected points on the surface of an object do not vary after similarity transforms.

4.6.1 Sampling a point set surface

The first step of our matching algorithm is to select a set of M random points from a point set surface. These random points will serve as *reference points* to generate a GDM $M \times M$, storing the geodesic distances between them. Since we aim at capturing the geometrical and topological structures of a surface, we require the reference points to be properly distributed over the surface.

The problem of sampling a surface in $\mathbb{R}^3$ is well studied in computer graphics (see also Section 4.3), e.g., in point based rendering [98, 250, 251], in painterly rendering systems [160], and in mesh optimization [6, 113] and simplification [56, 241] . Surfaces in $\mathbb{R}^3$ are also sampled in several matching methods [87, 120, 174, 179, 181, 232, 233]. In these methods, a surface is commonly approximated by a triangular mesh that is usually sampled through an uniform sampling as a preliminary step of a shape features extraction algorithm. In general, this uniform

sampling strategy consists in picking random triangles with probabilities proportional to their area, and generate random sample points inside them with equal probability over their areas. Uniform sampling is widely adopted in many methods, since it is simple to implement and fast to execute. However, it is not always the optimal choice for creating shape descriptors. In fact, a uniform sampling method generates samples that may not be evenly distributed on the space of the surface structures, since it is purely random and independent of the shape structure. For shape analysis and matching it is important that the spacial distribution of points captures the shape of the sampled object, so that a shape descriptor has a chance to describe it. Indeed, if a sampling method misses some relevant part of an object, in general, this part has a little probability to be captured. This effect is illustrated in Figure 4.16 where the model's hair is so detailed that most of its area is concentrated around it. Notice that the uniform sampling strategy shown in Figure 4.16a places the majority of samples in the model's hair, and consequently the sampling does not completely represent the shape of the model. Figure 4.16b shows the result of the evenly spaced sampling described in [174], which has a high probability to capture all the parts of the model.



(a) (b)

Figure 4.16: Uniform sampling vs. evenly spaced sampling of a triangle mesh (courtesy of Nehab and Shilane [174]). (a) An uniform sampling considering a probability proportional to the areas of the triangles, and (b) a stratified sampling generating sample points evenly distributed in space [174].

Nehab and Shilane in [174] studied the problem of sampling in relation to shape matching and painterly rendering. They proposed a stratified sampling algorithm for triangular meshes that generates point samples evenly spaced over a triangular surface with a minimum and maximum distance constraint between adjacent samples. Their algorithm is based on a voxelization

of a triangle mesh. From each voxel a triangle intersecting the voxel bounding box is chosen. The resolution of the voxelization is specified by the user and controls the total number of samples. A triangle is sampled randomly according to an exponential probability distribution defined on the distance (relative to the edge length of the enveloping voxel) between the sample point and the center of the originating voxel. With the rate parameter of the exponential distribution they could control the variation in the points placement with respect to the voxel center, and generated sampling patterns from highly jittered to highly regular (points close to the voxel centers). This results in a nonuniform sampling pattern with samples evenly distributed over a triangular surface, and with a spectral characteristic similar to the blue noise Fourier spectrum [242].

Nehab and Shilane in [174] applied their stratified sampling algorithm to a set of suitable shape descriptors (SDs). They compared their retrieval effectiveness with the same SDs obtained with an uniform sampling and a regular sampling generated from a voxel grid. The stratified sampling resulted to outperform the uniform sampling, and slightly improve the performance of the regular sampling on a voxel grid. This result motivated our research to adopt a sampling strategy, which generates a set of reference points non-uniformly and evenly distributed over a point set surface.

Nonuniform sampling strategies have a long history and a great importance in computer graphics as practical solutions to aliasing artifacts in rendering [58, 164]. Indeed, nonuniform sampling has the property of producing random noise that replaces spurious low frequencies. This noise is masked by the human visual system and generates better visual results [242]. For this properties nonuniform sampling methods have been applied in several applications, e.g., in: dithering [242], realistic rendering [58, 164], image processing [89], halftoning, and painterly/non-photorealistic rendering [109, 136, 160].

In computer graphics is well known that a good sampling should have blue noise spectral characteristic [58, 89, 164, 242], i.e., its power spectrum does not have power peaks at any non-zero frequency (leading to no aliasing artifacts), and most of the noise power is concentrated at the high frequency domain, to which the human eye is less sensitive. The most used sampling strategies that meets blue noise requirement implement a Poisson disk distribution. The generated sample points are randomly and non-uniformly distributed with the constraint that the distance between every point exceeds a certain threshold (called *Poisson disk radius*). The most commonly used technique to generate a Poisson disk distribution is the *dart throwing* algorithm. This technique consists in progressively generating points by randomly "throwing a

dart" on a surface as long as a requested number of points are collected. A new random point is discarded when it is generated inside a disk of a fixed radius and centered at a point already selected by the algorithm. Such techniques are known as *dart throwing techniques*.

With our graph based framework to compute geodesics (see Section 2.3), we implemented a dart throwing method to select a set Q of points on a point set surface S represented as an eSIG (see Section 2.3.1). The resulting distribution of the selected points is a Poisson disk distribution on S with respect to the geodesic distance function, i.e., the geodesic distance between every pairs of selected points on S is lower-bounded by a value G_{min}. Thus, G_{min} can be seen as the radius of the Poisson geodesic disk fixed by the method. The implemented algorithm is composed of the following steps:

1. Select randomly an initial point q_0, and insert it in the set Q.
2. Let $g(v) = g(q_0, v)$ be the geodesic distance of the eSIG's vertex v to q_0. Initialize $g(q_0) = 0$ and $g(v_i) = \infty \ \forall v_i \in eSIG$.
3. Let SELECTABLES be the list of all selectable vertices of the eSIG. Fill the list SELECTABLES with all the vertices of eSIG, apart from q_0.
4. Push q_0 into the priority queue MINPQUEUE, in which a vertex v will be ordered ascendingly with respect to the value $g(v)$.
5. Pop the vertex v from MINPQUEUE (v has the smallest $g(v)$).
6. For each vertex v_a adjacent to v: if $g(v) + length(v, v_a) < g(v_a)$, update $g(v_a) = g(v) + length(v, v_a)$. Then, if $g(v_a) < G_{min}$, push v_a into MINPQUEUE and remove v_a from SELECTABLES.
7. If MINPQUEUE is not empty go to step 5.
8. If $|Q| < M$, select and remove a random vertex v_i from SELECTABLES, insert v_i in the set Q, push v_i into MINPQUEUE, and go to step 5.

This algorithm is efficient when knowing the right value of G_{min}. On one hand, if G_{min} is too large, the algorithm cannot terminate at all, or if we reduce G_{min} after a consequent number of empty iterations, it can slowly converge to a set Q. On the other hand, if G_{min} is too small, the resulting points of Q might not be evenly spaced on the surface, as required. To overcome this problem various techniques based on relaxing sample points with Lloyd's

method were conceived [108, 157]. Recently, efficient techniques to approximate a Poisson disk distribution in $O(N \log N)$ time for a pattern of N random points were presented [76, 121]. Dunbar and Humphreys [76] presented an efficient algorithm, which approximates a Poisson disk distribution in $O(N)$ time and space. Most of the algorithms mentioned above are designed to produce sample points on a plane. Extending these methods to a point set surface may lead to high computational costs.

An efficient sampling strategy to generate sample points with a Poisson disk distribution and blue noise spectral characteristics was presented by Eldar et al. in [89]. Their technique is called *farthest point sampling*, which is explained in detail in the following section. The main idea is to select at each stage of the algorithm a new point among the vertices of the Voronoi diagram that has the farthest distance to the already selected points. This method generates sample points whose distances to each other have a suitable lower-bound (Poisson disk radius) and upper-bound [89].

The farthest point sampling (FPS) method is interesting, since it can be easily extended to manifolds represented by triangular meshes and, in particular, to point clouds [166, 167, 168]. Thus, to sample a set of M points on a point set surface, we adopt this technique, which is described in detail in the following section.

4.6.1.1 Farthest point sampling

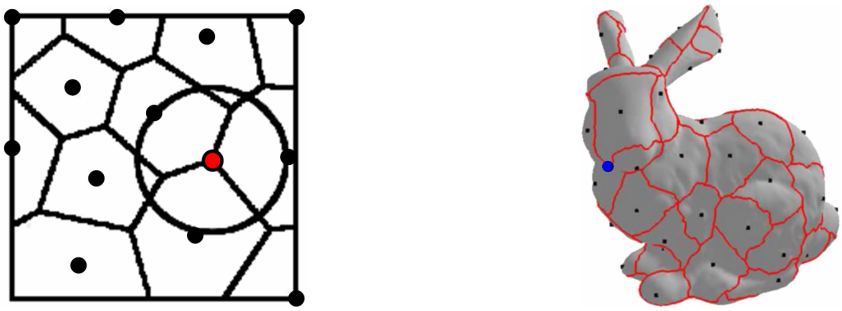
The idea of the farthest point sampling is to iteratively select points from a surface S, such that each new point is the farthest with respect to the already selected points. Eldar et al. [89] showed that a planar Voronoi diagram is intimately related to the farthest point strategy. On a surface a planar Voronoi diagram is extended to a *geodesic Voronoi diagram*.

Let $Q = \{q_i \in S : i = 1, .., M\}$ be a set of M points on a surface S, a *geodesic Voronoi cell* of the point $q_i \in Q$ is the set $GVC(q_i) = \{p \in S : g(p, q_i) \leq g(p, q_j), q_j \in Q,\ i \neq j\}$ of the points $p \in S$ whose geodesic distance $g(p_i, q_i)$ to the point $q_i \in Q$ is less than any other points $q_j \in Q : i \neq j$. The partition of the surfaces S into M geodesic Voronoi cells is called *geodesic Voronoi diagram* (GVD) of the set Q on S.

The *boundary* of $GVC(q_i)$ is definesd as the set $\Omega(GVC(q_i)) = \{p \in S : g(p, q_i) = g(p, q_j),\ q_j \in Q,\ i \neq j\}$. The *vertices* of $GVC(q_i)$ are those points of its boundary $\Omega(GVC(q_i))$, which belong to a boundary of at least two other different GVCs, i.e., $v \in \Omega(GVC(q_i)) \cap \Omega(GVC(q_j)) \cap \Omega(GVC(q_k)), i \neq j \neq k \neq i$. A *boundary segment* of a GVC is a subset of its

boundary, which is bounded by two of its vertices. The boundaries segments of the GVCs define the "edges" of a GVD, and their vertices define the GVD vertices.

If the surface S is bounded the resulting GVD is bounded, and is called bounded geodesic Voronoi diagram (BGVD). Thus, each of its cells (BGVCs) is also bounded, and is defined as the conjunction of the GVC with the domain.



(a) (b)

Figure 4.17: A farthest point visualized with respect to a bounded geodesic Voronoi diagram (BGVD) constructed on a plane and on a surface. (a) On a plane the farthest point (depicted in red) is located at the center of the largest circle, which does not contain any points (black points) generating the BVD. (b) BGVD constructed on a surface of a 3D model. A candidate farthest point is depicted in blue. **[see Color plate D.4]**

Eldar et al. [89] demonstrated that the farthest point p_f from a point set Q, is placed on a vertex of the Voronoi diagram of Q. Thus, on a bounded plane p_f is located at the center of the largest circle, which does not contain any points of the set Q. This property is shown in Figure 4.17a where the black points represent the points of Q and the farthest point is depicted in red. Similarly, on a general surface S the farthest point from Q is placed on a vertex of the BGVD generated by Q.

This property motivated Moenning and Dodgson [166, 167, 168] to conceive a fast marching method to achieve a FPS on a manifold. Their technique is based on incrementally computing the BGVD of the collected points on a point set surface. The geodesic distances used to construct the BGVD are computed with the fast marching method presented by Mémoli and Sapiro in [161]. The FPS algorithm to select M sample points from a surface S consists of the following steps:

1. Start with an initial set of random points $Q_0 : |Q_0| > 1$, and set $i = 0$.

2. Compute the BGVD generated from Q_0 ($BGVD_0$) by simultaneously propagating the geodesic distance function front from each point $q \in Q_0$, till all the surface S is covered by the BGVCs.

3. Let Q_i be the current set of selected points, and $BGVD_i$ the BGVD of the set Q_i. Select the vertex of $BGVD_i$ with the *maximum geodesic distance* to the points of Q_i as the new point q_{i+1}, and add q_{i+1} to Q_i to form the new set Q_{i+1}.

4. Compute $BGVD_{i+i}$ from $BGVD_i$ by only propagating the front of its geodesic distance function outwards of q_{i+1}, and updating the neighboring BGVCs accordingly.

5. Set $i = i + 1$.

6. If ($|Q_i| < M$), go to step 3.

7. Set $Q = Q_i$ as the final set of M points and $BGVD_Q = BGVD_i$.

The complexity of this algorithm is $O(M \log M)$ [167], and the resulting set of sampled points Q has a Poisson disk distribution [89, 167]. In fact, it is guaranteed that two sample points cannot have a geodesic distance smaller that a specific value G_{min} (Poisson disk radius). This value is computed by considering the geodesic distances of the new vertices of the $BGVDs$ that are iteratively generated by adding new points $q_i : i \geq |Q_0|$ to the initial set Q_0. Thus, the vertices of $BGVD_0$ are not considered in this computation. Let $G(v)$ indicates the geodesic distance between a point $v \in BGVD_Q$ and its nearest sample point $q \in Q$,

$$G(v) = \min_{q \in Q} g(v, q), \tag{4.17}$$

$$G_{min} = G(v_{max}) = \max_{v \in BGVD_Q - BGVD_0} G(v), \tag{4.18}$$

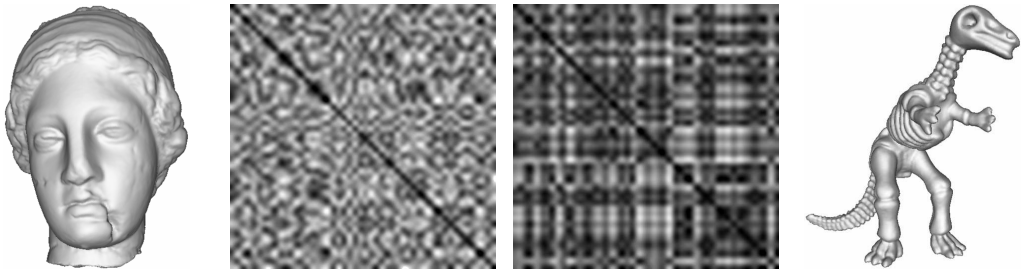
$$g(q_i, q_j) \geq G_{min} \quad \forall\ q_i, q_j \in Q : i \neq j. \tag{4.19}$$

Thus, for each sample point $q_i \in Q$, its geodesic distance $g(q_i, q_j)$ to any other sample point $q_j \in Q$ is at least equal to G_{min}.

We used the technique described above to sample the set Q of M reference points on the point set surface S. In our implementation the geodesic distances are computed with the technique presented in Section 2.3.

4.6.2 Computing a geodesic distance matrix

Given a set Q of M reference points randomly sampled on a point set surface S, we compute the associated geodesic distance matrix (GDM) of size $M \times M$. Each element m_{ij} of the GDM stores the geodesic distance $g(q_i, q_j)$ between the sample points $q_i, q_j \in Q$, respectively associated to the ith row and the jth column. Thus, each point $q_i \in Q$ is associated to the row and column i of the GDM. Obviously, a GDM is symmetric and has all the elements of the main diagonal equal to 0. The geodesic distances are computed with the method described in Section 2.3. The GDM is then normalized with respect to the sum of all the geodesic distances computed. Figure 4.18 shows two point based models with the corresponding normalized GDMs of size 256×256. The matrices are shown in form of matrix images where each of their value is coded by a gray scale color: the value 0.0 is coded as black and the value 1.0 as white.



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

Figure 4.18: Normalized geodesic distance matrices of size 256×256 computed from surfel based models. The element of the matrices are codes as gray scale values, where 0.0 is associated to black and 1.0 to white. The matrix shown in (b) is associated to the model depicted in (a), while the matrix depicted in (c) is associated to the model illustrated in (d).

4.6.3 Comparing geodesic distance matrices

We present an approach to compute the similarity between two point based models, by comparing their corresponding normalized GDMs computed with the method presented in Section 4.6.2.

A GDM has a particular structure. It is symmetric and has all the elements of the main diagonal equal to 0. Moreover, the indices of its rows and columns indicate the order in which the reference points are associated to them (see Section 4.6.2). This order is arbitrary, since the reference points are selected randomly on the point set surface with the FPS strategy (see Section 4.6.1). Given the set $Q_X = \{q_1, q_2, .., q_M\}$ of M reference points ordered according to their selection order, for each $i = 1, .., M$ the point q_i is associated to the row i and the column

i of the GDM M_X. Let $Q_Y = \{q_2, q_1, .., q_M\}$ be the set obtained by Q_X by swapping q_1 with q_2, and M_Y the associated GDM. M_Y is not equal to M_X, although stores the same information. However, if we swap the row 1 with the row 2 of M_Y, and then we swap the column 1 with the column 2, M_Y become equal to M_X. Thus, the problem of comparing two GDMs M_X and M_Y is similar to the problem of finding the best match between the elements of M_X and M_Y among all the permutations of the row and column indices of M_Y. Formally, let M be the number of rows and columns of a GDM, and $\pi : \{1, .., M\} \rightarrow \{1, .., M\}$ be a permutation of the set $\{1, .., M\}$ of indices of its rows and columns, the problem of comparing two GDMs M_X and M_Y can be stated as the problem of finding the permutation π, which minimizes the distance function $d_\pi(M_X, M_Y)$ between the M_X and M_Y, such that

$$d_\pi(M_X, M_Y) = \sum_{1 \leq i < j \leq M} |m_{ij}^{(X)} - m_{\pi(i)\pi(j)}^{(Y)}|, \tag{4.20}$$

where $m_{ij}^{(X)}$ is the element stored in the ith row and jth column of M_X, and $m_{\pi(i)\pi(j)}^{(Y)}$ is the element of M_Y stored in its row $\pi(i)$ and column $\pi(j)$, which are associated by the permutation π to the indices i and j, respectively [162].

Thus, the permutation π establishes the correspondence between the rows and columns of M_X and the rows and columns of M_Y. The dissimilarity value between M_X and M_Y is then given by the following formula:

$$\delta(M_X, M_Y) = \min_{\pi \in \Pi_M} d_\pi(M_X, M_Y), \tag{4.21}$$

where Π_M is the set of all possible permutations of the set $\{1, .., M\}$ of indices of the rows and columns of M_Y. Thus, this approach implies a search on the set Π_M, and needs to be addressed with combinatorial methods. The computation of $\delta(M_X, M_Y)$ is time consuming, since the size of Π_M is $|\Pi_M| = M!$ and usually $M \geq 100$. In fact, in the worst case the computation of $\delta(M_X, M_Y)$ requires $O(M!(M^2 - M)/2)$ arithmetic operations.

To reduce the complexity of this problem Mémoli and Sapiro [162, 163] (see Section 4.5) used a computationally non-optimal heuristic, which restricts the search space by progressively constructing sub-matrices of M_X and M_Y. The sub-matrices of M_X are constructed according to the farthest point sampling strategy, while the sub-matrices of M_Y are built by rearranging the rows and the columns of selected points minimizing point-wise the approximation error of the distance expressed in Equation 4.14.

We approach the problem of computing $\delta(M_X, M_Y)$ by searching the matching between the reference points associated to M_X with those associated to M_Y, which produces the minimal

value of $d_\pi(M_X, M_Y)$. As stated before, each reference point is associated to a row of the GDM, which describes through its geodesic distances the intrinsic geometry and the topology of the surface at that point. Thus, two reference points can be compared by matching their GDMs' rows. To reduce the complexity of this matching, we associate to each row of the GMDs M_X and M_Y a histogram of its elements. Let q_i be the reference point associated to the row i of the GDM M_X, the histogram $h_i^{(X)}$ describes the distribution of the geodesic distances between the point q_i and all the reference points associated to the other rows of M_X. $h_i^{(X)}$ can then be considered as a statistical shape descriptor of the point based surface X at q_i. The generated histograms have a small number of bins, which depends on the number of reference points, e.g., we use 64 bins for rows of 1000 elements. With this approach each GDM ($M \times M$) is described by a set of M histograms of B bins.

The problem of matching two GDMs is then reduced to the problem of finding the best match between two sets of M histograms. We formulate this problem as a bipartite graph matching [137]. A complete undirected bipartite graph $G = (V, E)$ is constructed by taking as set of vertices V the set of histograms $h_i^{(X)}$ and $h_j^{(Y)}, 1 \leq i, j \leq M$, respectively associated to the rows of the matrix M_X and M_Y. Each histogram $h_i^{(X)}$ of the matrix M_X is connected to every histogram $h_j^{(Y)}$ of the matrix M_Y through an edge e_{ij}, which is added to the set of edges E of G (see Figure 4.19). Each edge $e_{ij} \in E$ is then weighted with a distance function $d_h(h_i^{(X)}, h_j^{(Y)})$.

Histograms are usually compared by using various bin-to-bin distance functions, such as: Minkowski L_p distance, χ^2 distance [139], Kullback-Leibler divergence distance [139], and Bhattacharyya distance [24]. These methods assume that the domain of the histograms are already aligned. However, in practice histograms might be misaligned and have different scales, although they represent distributions with the same probability density function (PDF). To overcome this problem, techniques considering cross-bin comparison of histograms (e.g., the Earth Movers distance [202]) were presented in [146, 147, 181, 202]. Following the approach presented in [181] we compare two histograms by comparing piecewise linear functions representing the distribution function of the geodesic distances approximated by the histograms. From each histogram h_i of B bins, we reconstruct a piecewise linear function (PLF) $\vec{f_i}$ with F ($\leq B$) evenly spaced vertices. For each $i = 1, .., M$, $\vec{f_i}$ is stored as a vector of F integers. Let $\vec{f_i}$ and $\vec{f_j}$ be two piecewise linear functions associated to the histogram h_i and h_j, respectively. We then compare h_i with h_j by comparing $\vec{f_i}$ with $\vec{f_j}$. Before comparing $\vec{f_i}$ with $\vec{f_j}$ they are normalized. Osada et al. [181] proposed three normalization methods: (1) align the maximum values of the PLFs, (2)

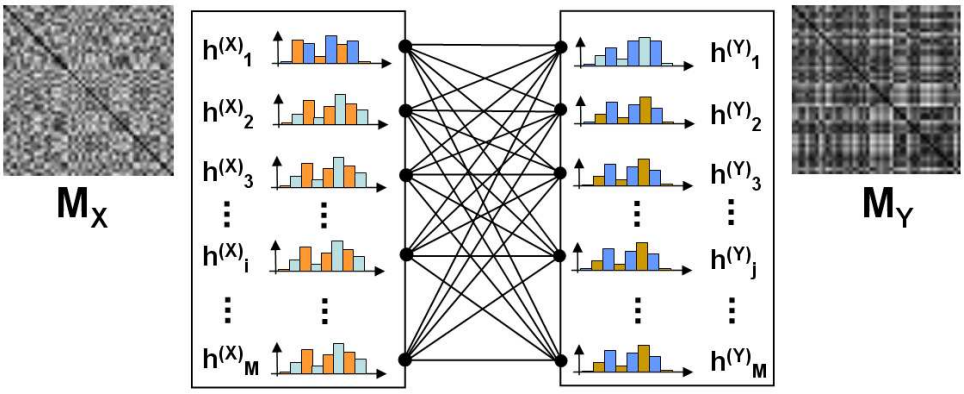


Figure 4.19: Histogram based approach for matching geodesic distance matrices (GDMs). Each row of the GDMs M_X and M_Y of size $M \times M$ is associated to a histogram $h_i^{(X)}$ and $h_j^{(Y)}$, respectively. Each histogram generated from M_X is connected via M edges with all histograms generated from the matrix M_Y. An edge e_{ij} is weighted with the distance $d_h(h_i^{(X)}, h_j^{(Y)})$ between the corresponding histograms $h_i^{(X)}, h_j^{(Y)}$. A matching between M_X and M_Y is that set of edges minimizing a function defined on the edge weights.

align the mean values (or similarly the median) of the PLFs, and (3) search for the scale that produces the minimal dissimilarity measure during each PLF comparison.

The first two alignment approaches can be evaluated analytically, and the method (2) resulted in being more effective than the method (1). Once two histograms h_i, h_j are aligned, their distance $d_h(h_i, h_j)$ is computed by using the L_1 norm between $\vec{f_i}$ and $\vec{f_j}$, i.e.,

$$d_h(h_i, h_j) = d(\vec{f_i}, \vec{f_j}) = |\vec{f_i} - \vec{f_j}|. \tag{4.22}$$

The method (3) involves a minimization procedure. In fact, the distance function $d_{h_{min}}(h_i, h_j)$ is computed with the following formula:

$$d_{h_{min}}(h_i, h_j) = \min_s d(\vec{f_i}(x), s\vec{f_j}(sx)), \tag{4.23}$$

where s is a scale factor. As in [181], we perform the search over s by first scaling $\vec{f_i}$ and $\vec{f_j}$ with respect to their mean value. Then, $d(\vec{f_i}(x), s\vec{f_j}(sx))$ is evaluated for values of s in 100 equally spaced intervals.

The method (3) provided the most effective solution among the three methods mentioned above. However, it is not as efficient as the method (2). Thus, in case a fast comparison of many GDMs is needed, like for querying a database of point based models, the method (2) is more suitable than the methods (3) and (1). Nevertheless, since in practice our histograms

are small and the geodesic distances of the GDMs are normalized, we experimentally observed that the best compromise between efficiency and effectiveness can be obtained by using the χ^2 distance between histograms, which is defined as follows:

$$d_{h_{\chi^2}}(h_i, h_j) = \frac{1}{2}\sum_{k=1}^{B}\frac{(h_i[k] - h_j[k])^2}{h_i[k] + h_j[k]}, \tag{4.24}$$

where B is the number of bins of the histograms h_i and h_j, and $h_i[k]$, $h_j[k]$ are the values stored in the kth bin of h_i and h_j, respectively. Bins equal to zero, $(h_i[k] + h_j[k] = 0)$ are not compared.

A *matching* in G is a subset $E_M \subset E$ of edges of E, such that no two edges in E_M share a common vertex of V, i.e., a vertex coming from the matrix M_X, can be connected to only one vertex coming from the matrix M_Y. A *maximum cardinality matching* is a matching with the maximum number of edges. A *minimum weight matching* is a matching, which minimizes the sum of the weights of the edges in the matching.

We aim at finding the *minimum weight perfect matching*, which is the minimum weight matching with cardinality $|E_M| = |V|/2$. This problem can be solved in running time $O(|V|^2 \log |V|)$ with optimal implementations of the Edmonds' blossom algorithm as in [59]. However, Schwartz et al. [213] presented an algorithm achieving a running time of $O(|V|^2)$ under the assumption that the edge weights are integers and uniformly distributed. We use the implementation of the Edmonds' blossom algorithm kindly provided by Cook [59].

Once the minimum perfect matching E_M between the histograms is found we have a bijection $\pi^* : \{1, .., M\} \rightarrow \{1, .., M\}$ between the histograms computed from M_X and those computed from M_Y. Therefore, the dissimilarity between the two GDMs M_X and M_Y is computed with the function $\delta_h(M_X, M_Y)$, which is calculated as follows:

$$\delta_h(M_X, M_Y) = \sum_{i=1}^{M} d_{h_{\chi^2}}(h_i, h_{\pi^*(i)}), \tag{4.25}$$

In order to save storage space, we store as shape descriptor the histograms (or the PLFs) associated to the rows of a GDM, i.e., $M \cdot B$ integers, instead of storing a GDM of M^2 real values.

4.6.4 Results

We evaluate the effectiveness of our matching method on a collection of 24 surfel based models subdivided in 6 classes of isometric objects. Each class contains different models of the same

object, which have been deformed with a bending deformation (see Appendix A). These models were obtained by surfelizing, with the technique presented in Section 4.3, a collection of triangle meshes kindly provided by Prof. Kimmel and his group at the Technion, Haifa, Israel [196]. These models were also used to evaluate bending invariant shape descriptors in [85, 87, 162]. Figure 4.20 shows the entire collection of surfel based models arranged by rows according to their classes. Each row identifies a class, which is associated to a specific color. Objects belonging to the same class have the same color. Each class contains four different models of the same object, which has been deformed with bending deformations.

We test our geodesic based matching method by performing several queries on this collection of models. Figure 4.21a shows the result of a query considering a model of a human body as query object. The retrieved objects are ranked according to their dissimilarity values to the query object. The rank and the dissimilarity value of each object are shown as header of its icon. The icons representing the objects have different layouts: blue (query object - top left), green (relevant objects), and red (non-relevant objects). We recall that, a relevant object is an object belonging to the same class of the query object. Our matching technique retrieves in the first positions of the resulting ranked objects all the models of the same class. This means that our method is able to recognize objects of the same class despite of bending deformations. The same query is performed using our image based shape descriptor presented in Section 4.4. We use depth-buffer images of size 256×256 and FVs of dimension 438. Figure 4.21b shows the result of this query. Notice that in the first positions of the ranking do not appear models belonging to the same class of the query object. The first relevant model is placed in the 12th position.

A query for each object of the collection is also performed. The dissimilarity values of the query objects with respect to all other objects of the collection are computed with Equation 4.25. Then, they are stored in a dissimilarity matrix, which is visualized as distance matrix image (see Section 4.1.2) in Figure 4.22a. White pixels indicate the maximum dissimilarities, while black pixels zero dissimilarities. Models of the same class are grouped along each row, so that they are associated to contiguous matrix elements (contiguous pixels). Notice that the matrix depicted in Figure 4.22a presents blocks of darker pixels along the main diagonal. Each of these blocks is as large as the related class size. This indicates that every object matches the objects within its class better than those in other classes. Thus, our method is able to successfully identify the class of all the objects of the collection. However, there are two cases where our method does not retrieve a relevant object as object of the same class of the query object. This

fact is shown in Figure 4.21, where the two relevant models of dogs are retrieved at position 8 (Figure 4.21a) and 6 (Figure 4.21b) instead of at the first four positions. The non-optimal results of these two queries slightly drops the R-precision measured with our method to the value 97.2%, which is a bit smaller than the maximal value (100%).

The same experiment is executed by using our image based shape descriptor. Figure 4.22b shows the resulting dissimilarity matrix image. Notice the this image is not characterized by blocks of darker pixels along the main diagonal as the matrix visualized in Figure 4.22a, but rather by a large block of dark pixels grouping the first five classes together. This means that the image based descriptor was not able to discriminate the different classes of the collection. The R-precision measured by using this method is 40.3%.

All the experiments reported above using our geodesic based method are performed by selecting 256 reference points with the FPS method, which generate GDMs of size $M \times M$ and set of M histograms with B bins, where $M = 256$ and $B = 16$. The histogram are compared with the function defined by Equation 4.23. Those values of M and B represent the best tradeoff between effectiveness and execution time. However, the precision of the method strongly depends on the number of reference points selected on the point set surface of an object. A small number of reference points implies a high probability to neglect important topological/geometrical structures of the object in the comparison. Suitable parameters to achieve a high precision comparison are $M = 1000$ and $B = 64$.

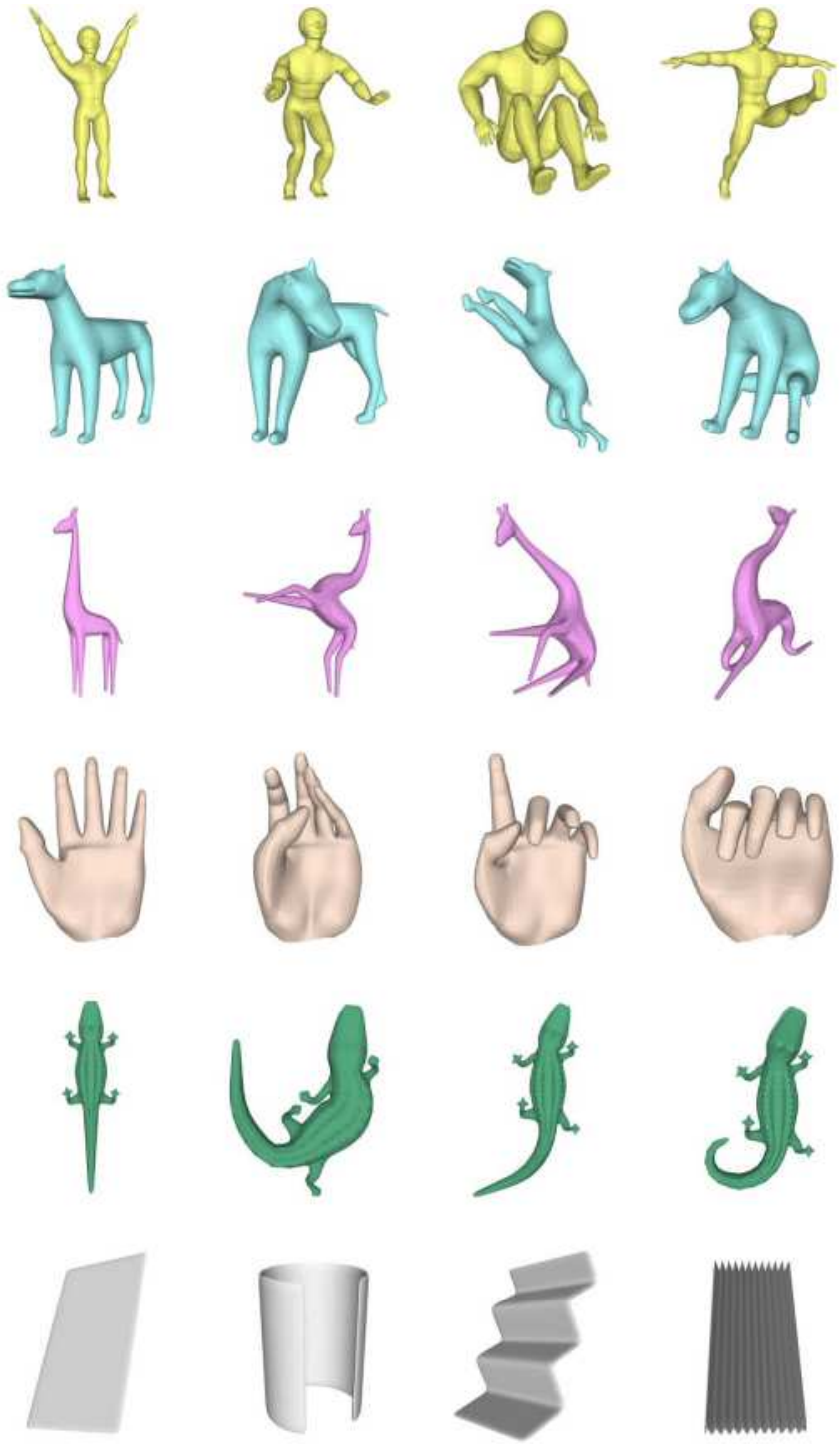
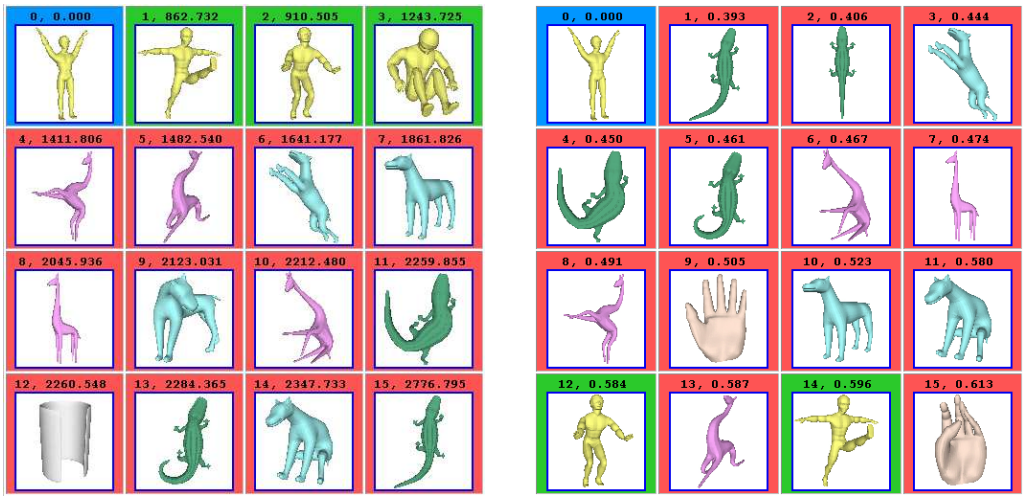
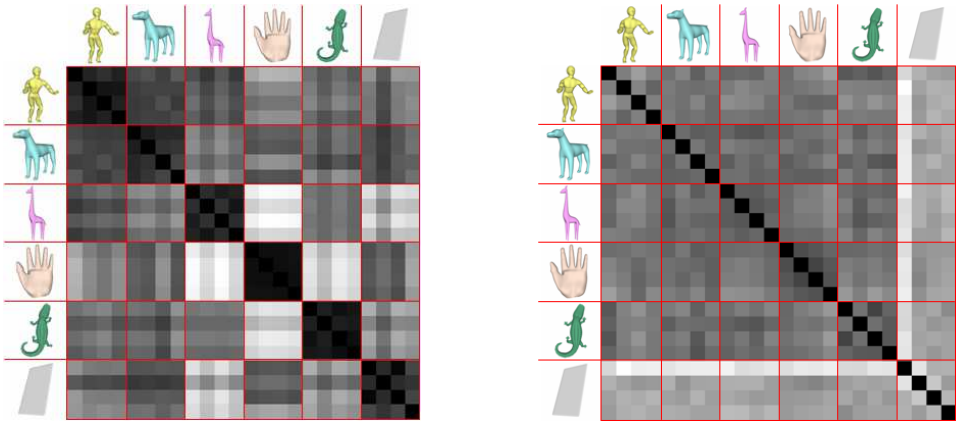


Figure 4.20: Collection of surfel based models partitioned in classes of deformed objects. The models are arranged by rows. Each row identifies a class, and is associated to a specific color. Each class contains different models of the same object, which has been deformed with bending deformations. [**see Color plate D.7**]



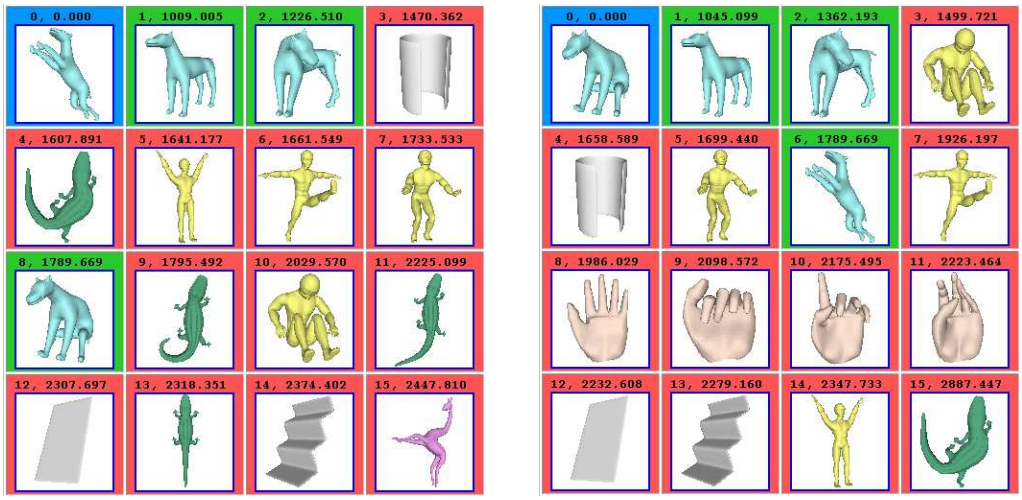
(a) (b)

Figure 4.21: Result of a query using the geodesic based (a) and the image based (b) shape descriptors on the collection of deformed surfel based models illustrated in Figure 4.20. The retrieved models are ranked according to their dissimilarity values to the query model (shown at position 0). The rank and the dissimilarity value of each model are shown as header of its icon. The icons have different layouts: blue (query model - top left), green (relevant models), and red (non-relevant models). [**see Color plate D.5**]



(a) (b)

Figure 4.22: Dissimilarity matrices images computed using the geodesic based (a) and the image based (b) shape descriptors on the collection of deformed surfel based models illustrated in Figure 4.20. White pixels indicate the maximum dissimilarities, while black pixels dissimilarities equal to 0. Models of the same class are grouped along each row.



(a) (b)

Figure 4.23: Result of two queries performed using the geodesic based shape descriptor on the collection of deformed surfel based models illustrated in Figure 4.20. The retrieved models are ranked according to their dissimilarity values to the query model (shown at position 0). The rank and the dissimilarity value of each model are shown as header of its icon. The icons have different layouts: blue (query model - top left), green (relevant models), and red (non-relevant models). These are the only two queries where a relevant object is not retrieved as object of the same class of the query object, as shown by the models ranked at position 8 (a) and 6 (b) instead of at the first four positions. [**see Color plate D.6**]

Chapter 5

Conclusions

Following is a summary of the contributions and the most important results of this thesis as well as proposals for future work.

In this thesis we presented a framework for analysis and processing of point set surfaces based on geodesics.

Considering the lack of topology information in point based representations, the problem of computing exact geodesics on point set surfaces is challenging and relevant for many applications. In Chapter 2 we presented two techniques to compute approximations of geodesics on point set surfaces.

The first technique, presented in Section 2.3, computes fast approximations of geodesics on point clouds. It relies on a graph representation of the surface using an extended sphere of influence graph (eSIG). Approximations of geodesics are computed as shortest paths on the eSIG using Dijkstra-like algorithms. The main contribution of this technique is to have proposed an extended version of sphere of influence graph optimized for surfel based representations, which allows to efficiently and effectively approximate a large quantity of geodesics in a short time. This technique resulted in being an efficient method for geometry and topology processing of surfel set surfaces. Examples of its application to textured geometry compression and topology matching of point set surfaces are presented in Section 3.2 and Section 4.2, respectively. This method to compute approximations of geodesics is fast and efficient, while at the same time it is sensitive to sampling. Its precision depends strongly on the quality of the sampling of the underlying surface. Irregular and non-dense sampling induces approximation errors, while regular sampling induces metrication errors.

The second technique, presented in Section 2.4, computes accurate approximations of geodesics, and is robust with respect to noise and outliers. It is a novel technique, which computes

piecewise linear approximations of geodesics on point set surfaces by minimizing an energy function defined for piecewise linear paths. The function considers path length, closeness to the surface for the nodes of the piecewise linear path and for the intermediate line segments. In order to avoid local minima, our approach needs to start from a good initial approximation of a geodesic, which is computed with the fast technique mentioned above (see also Section 2.3).

A study of experimental convergence of our method was provided for both noisy surfaces and surfaces without noise. Our experiments showed that our method is robust with respect to noise and outliers. We provided a comparison of our method with the Hofer and Pottmann method (here called HP method) [111]. In general, on noisy surfaces our method proved to be more precise and reliable than the HP method.

Although our technique was designed for surfels, it was implemented also for MLS surfaces. The precision achieved by our method using MLS surfaces is similar to that using surfel based surfaces. Optimizations of our method for MLS surfaces will be considered in future work. However, our technique is general, in the sense that it can be applied to other surface representations, where the distance between a 3D point and the surface can be estimated, e.g., triangle meshes, RBF surfaces, etc.

Our method is suitable to compute good approximations of geodesics directly on noisy surfaces with outliers, i.e., the typical output of 3D scanning systems. It also presents multi-resolution features. In fact, it can be applied iteratively on initial paths at different resolutions, in order to improve the precision of the method and its execution time, especially for long geodesics. However, the time complexity of our method is still too high, and therefore not yet feasible for the computing of large numbers of geodesics, especially on large data sets. Future work will be oriented towards developing the multi-resolution characteristics of our method by considering multi-resolution curves representations, e.g., curves based on wavelets or hierarchical B-splines.

Geodesics being naturally related to the intrinsic geometry and topology of their underlying surface may lead to effective characterizations of surfaces. Moreover, they are invariant with respect to isometric transformations of surfaces, like bending deformations (see Appendix A). In this thesis we demonstrated the importance of geodesics for geometrical/topological processing and analysis of point set surfaces, especially for compressing textured point set surfaces of genus-0 (in Chapter 3) and for geometry and topology matching of point based models (in Chapter 4).

In Chapter 3 we described a surface compression scheme designed for genus-0 surfaces represented by surfel sets. This compression scheme, presented by Darom et al. in [66, 67], handles various surface attributes, notably geometry and texture, within a unified framework. Geometry, texture and other surface attributes are incorporated in a unified manner. The input surfaces are represented by surfels. Each surfel, with its attribute vector, is mapped onto a sphere in the sense of geodesic distance preservation. The resulting spherical vector-valued function is then resampled on a regular spherical grid. Its components are decorrelated by the Karhunen-Loève transform, represented by spherical wavelets and encoded using the zerotree algorithm.

This compression method demonstrated better performance than other methods in term of induced geometry error at very low bit-rates. However, its major contribution is a novel dense spherical mapping algorithm. Unlike previous approaches [86], there is no need to down-sample the surface or interpolate the solution. The introduction of rectangular (rather than square) geodesic distance matrices in MDS-type surface flattening dramatically lowers the memory needed for a given resolution and accuracy, and turns accurate spherical mapping of large surfaces into a computationally-feasible task. Following Bankirer [14], this approach is expected to have an impact on algorithms for surface flattening onto the plane [99, 265].

The described compression scheme necessitated a geodesic path finder for surfel based surfaces (Section 3.2.1). In fact, in the spherical parameterization method (Section 3.2) the geodesic distances between each surfel and a set of key-surfels are preserved in order to improve the resolution and eliminating the need for interpolation that complicates and slows down existing surface unfolding methods. These geodesic distances are computed by using the fast technique presented in Section 2.3, since for our spherical parameterization method it is not crucial to have exact geodesic distances. Furthermore, the method of Section 2.3 provided a good balance between performance and efficiency. For higher precision, one possibility would be to use the technique presented in Section 2.4, but its execution time is not feasible for compressing large size point set surfaces. In fact, such an approach must be achievable to compute many geodesics in a short time. Another possibility would be to apply only few iterations of the HP method [111] modified for surfels, in order to produce at least a slight improvement in the precision of our geodesic estimates. However, in this scenario the tradeoff between geodesics accuracy, execution time and distortion of the spherical parametrization plays a fundamental role, which will be investigated in future work.

The tradeoff between geometry and texture bit-allocation could also be improved by using a perceived visual quality criterion introduced in [19]. The described compression scheme is also ready to incorporate this further development.

In Chapter 4 we presented a retrieval system for point based models. In this system an object represented with a point based representation is submitted as a query, and a set of similar objects are retrieved from a collection or a database of point based models. The presented system includes a surfelization technique for converting triangular mesh models into corresponding surfel based representations, and two techniques for matching point based models.

The presented surfelization technique (Section 4.3) is able to convert large databases of triangular mesh models into corresponding databases of surfel based models in an acceptable period of time. Our technique was designed to convert benchmark databases of triangular models, which are commonly used in the 3D model retrieval community for testing, evaluating and comparing the retrieval effectiveness of several shape matching methods. For this reason our surfelization method was conceived to preserve the geometric features of the original triangular models. With our technique we converted two databases of triangular mesh models (the CCCC database [17] and the Princeton shape benchmark database [219]) into databases of surfel based models, which mainly have the same visual and geometrical characteristics as the original databases. The main contribution of our technique is the preservation of the visual and geometrical similarity of the original models. This fact allowed us to obtain surfel based models of high resolution, and to compare the performance of image based matching methods for triangular mesh model with the same methods developed for surfel based models.

However, our surfelization technique does not consider the texture properties of a triangle mesh, since the currently available retrieval benchmark databases consist of non-textured models. Indeed, shape matching algorithms do not consider texture information of an object, but rather information about its geometry, topology, and in few cases its reflectance. Although textures are not relevant for shape analysis and retrieval, they are fundamental for realistic visualization. Thus, an interesting future development of our technique would be to modify the geometry sampling strategy in order to also preserve the texture quality of the final surfel based models.

In Section 4.4 we presented a technique for matching and retrieving surfel based models, which is based on depth-buffer images. This technique consists in generating a shape descriptor from depth-buffer images rendered by orthogonally projecting a surfel based model onto the six faces of a suitable bounding cube. This bounding cube is constructed after a pose estimation of

a surfel based model by using PCA. The resulting six depth-buffer images are then processed with a 2D-FFT. The magnitudes of low-frequency Fourier coefficients of the resulting six Fourier spectra are then stored in a feature vector, which represents the final shape descriptor of the surfel based model.

Our depth-buffer image based shape descriptor possesses an embedded multi-resolution representation, is invariant with respect to similarity transforms and robust with respect to outliers. This approach resulted to be effective in comparing the geometry of several classes of objects represented with surfel based representations. In fact, it obtained approximately the same good retrieval results as the depth-buffer image based method presented in [249] for triangular meshes, which demonstrated the best average retrieval effectiveness in [40]. However, it is well known that the effectiveness of a shape matching method depends on the class of models on which it is applied, and on the application [40, 219, 249]. Therefore, further improvements of our method and implementations of alternative shape descriptors for point based models can lead to a better matching effectiveness for specific classes of objects of particular interest, such as: human bodies, human faces, molecular models, CAD articulated models, etc.

In Section 4.6 we presented a matching method for point based models that is particularly suitable to discriminate class of objects, which are deformed through isometric transformations, like bending deformations (Appendix A). Such classes of objects are typical of several applications, such as: face recognition and mining of articulated objects in mechanical industries and of surfaces generated from medical images.

Our technique uses geodesic distances between point pairs randomly selected on a point set surface to match the geometry and the topology of objects represented with point based representations. A set of random points evenly distributed on a point set surface is selected using the farthest point sampling method. Then, a normalized geodesic distance matrix is computed by storing in each element the geodesic distance between the two corresponding random points. The dissimilarity of two point based models is then computed by comparing the associated distance matrices by using a histogram based approach. A histogram of geodesic distances is generated for each row of the matrix. Similar histograms in the two matrices are matched using the Edmonds' blossom algorithm for computing a matching in a bipartite graph. The matched histograms are then compared to evaluate the dissimilarity of the two objects. This histogram based approach for comparing geodesic distance matrices represents the main contribution of our technique. Although, our approach is still not enough efficient for retrieval applications on large databases, it offers a direction to improve the performances and the effectiveness of

topology matching of point based models. The main limit of our method is the time required for GDMs alignment, which implies a time consuming minimization procedure. Future work will be oriented both to reduce the number of histograms to be aligned and to bypass the alignment procedure, which would allow to directly produce a unique and compact feature vector representation.

Our geodesic based method demonstrated a good retrieval effectiveness for classes of deformed models and isometric objects. Moreover, unlike the image based method presented in Section 4.4, it does not require any pose normalization step, since the invariance with respect to similarity transforms is already provided by the definition of the shape descriptor. Indeed, the geodesic distances between selected points on the object surface do not vary after similarity transforms.

Geodesics carry other topological/geometrical information, which may be used for topology matching, e.g., the geometric features of the geodesic curves. Future work will be focused in this direction, by also considering the Morse theory as a tool for selecting the end points of the geodesics.

Appendix A

Isometric transformation

Definition 1 (Metric). A *metric* (also called *distance function*) on a set X is a non-negative definite function $d : O^2 \rightarrow \mathbb{R}^+ \cup \{0\}$ with the following properties:

1. *Identity*: $\forall x \in X, d(x, x) = 0$.
2. *Positivity*: $\forall x \neq y \in X, d(x, y) > 0$.
3. *Symmetry*: $\forall x, y \in X, d(x, y) = d(y, x)$.
4. *Triangle inequality*: $\forall x, y, z \in X, d(x, z) \leq d(x, y) + d(y, z)$.

Definition 2 (Metric space). A set X is a *metric space* if for every points $x, y \in X$ there exists a metric $d_X : O^2 \rightarrow \mathbb{R}^+ \cup \{0\}$ on X, which gives the distance between them [244]. For clarity, a metric space if specified together with its metric as (X, d_X).

Definition 3 (Isometric transformation). An *isometric transformation* or *isometry* is a bijective function $\phi : X \rightarrow Y$ between two metric spaces (X, d_X) and (Y, d_Y) such that $d_X(x_1, x_2) = d_Y(\phi(x_1), \phi(x_2))$, for all $x_1, x_2 \in X$. The two metric spaces (X, d_X) and (Y, d_Y) are then called *isometric metric spaces*. Isometric transformations are length preserving transformations, meaning that they do not generate shearing, stretching or shrinking on the surface of an object during deformation. In some cases, deformations of objects can be approximated by isometric transformations, e.g., moving components of articulated objects and varying face expressions.

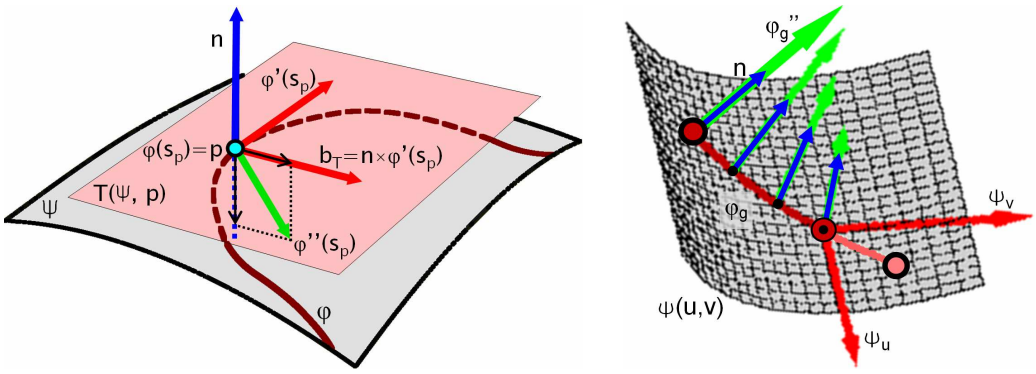
We call **bending deformations** those isometric transformations, which deform an object by bending its surface.

Appendix B

Color plates of Chapter 1 and 2

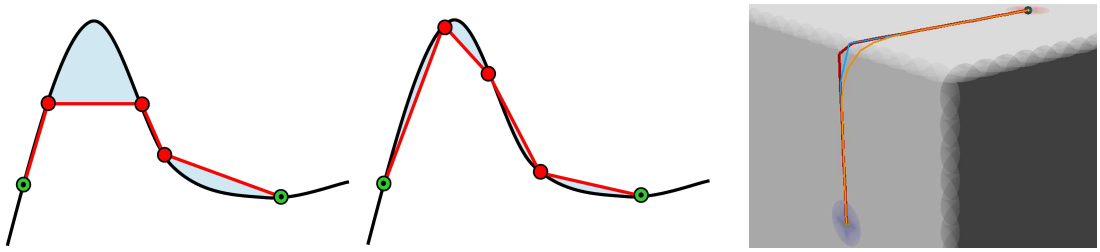
(a) (b)

Figure B.1: Point based representations of the *Butterfly* model by Arius Inc. [10]: (a) point cloud; (b) surfels (splats). **[see Figure 1.1 on page 1]**



(a) (b)

Figure B.2: Geometric properties of a curve and a geodesic. (a) A parameterized curve φ on a surface Ψ. The second derivative vector $\varphi''(s_p)$ (green vector) of φ at the point $p = \varphi(s_p) \in \Psi$ is shown with respect to other geometric properties, such as: the tangent plane $T(\Psi, p)$ of Ψ at p (pink plane); the normal vector n of Ψ at p (blue vector); the tangent vector $\varphi'(s_p)$ of φ at p (red vector 1); the vector $b_T = \varphi'(s_p) \times n$ (red vector 2); the components of the vector $\varphi''(s_p)$ on the direction of the vectors $\varphi'(s_p)$ and b_T (black vectors). (b) A geodesic curve (φ_g) on a parametric surface ($\psi(u, v)$) visualized as a trajectory of a particle. Along the trajectory of the particle, particle's acceleration vectors (φ_g'', in green) are parallel to the surface normals (n, in blue). The vectors spanning the tangent plane of $\psi(u, v)$ at the current particle position are shown as ψ_u and ψ_v (red vectors). [**see Figure 2.2 on page 11**]



(a) (b) (c)

Figure B.3: (a)(b) 2D view of two piecewise approximations of a geodesic between two points. The highlighted area represents a measure of the path proximity to the underlying surface (black curve). (c) Approximations of geodesic paths computed on a surfel based model with sharp edges: the initial path computed as shortest path on the graph using Dijkstra's algorithm (in light-blue); the path computed with our method using the $U_S(P)$ term with $\beta = 2.0$ (in dark-red) and with $\beta = 0.0$ (in orange). [**see Figure 2.7 on page 25**]

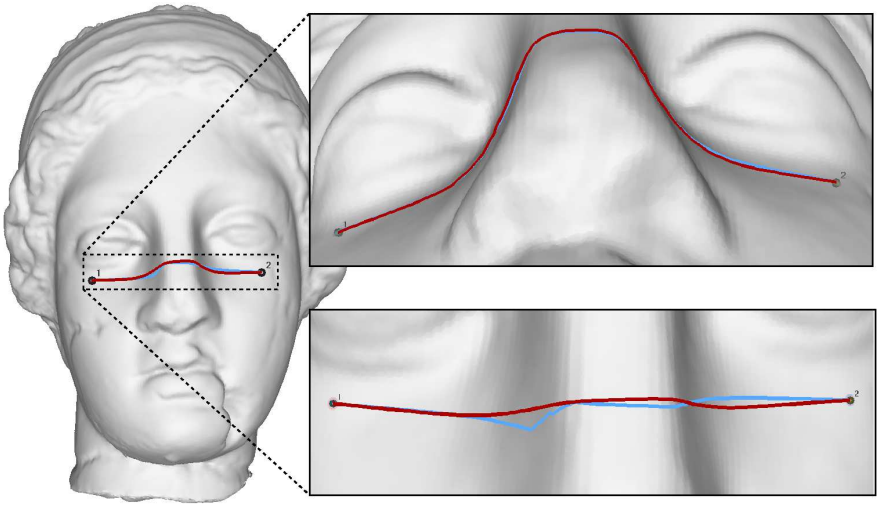


Figure B.4: Approximations of geodesic path computed with our method (dark-red) with respect to the initial path (light-blue) computed by applying Dijkstra's algorithm on the eSIG (Section 2.4.2). Our path properly fits the underling surface (top-right) and is shorter than the initial path (bottom-right). [**see Figure 2.8 on page 27**]

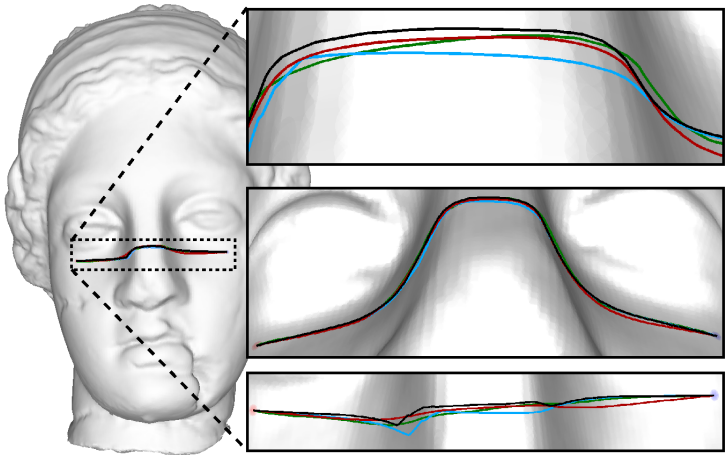


Figure B.5: Geodesic paths computed with different methods: initial path computed as shortest path on the graph with Dijkstra's algorithm (light-blue), path computed with our method on surfels (dark-red) and on MLS surface (dark-green), and path obtained with the HP method (black). [**see Figure 2.10 on page 35**]

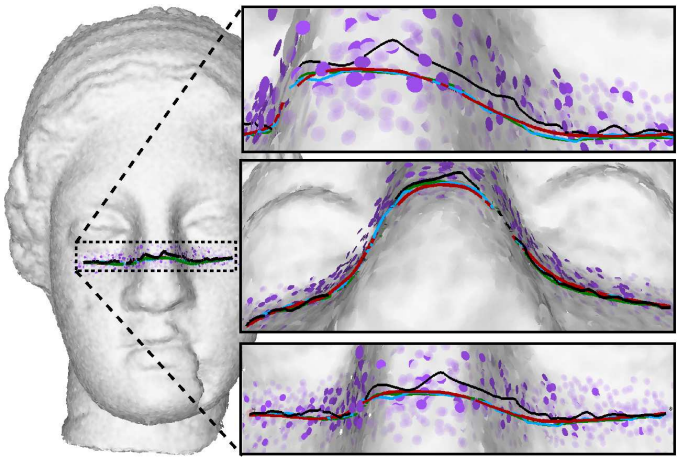
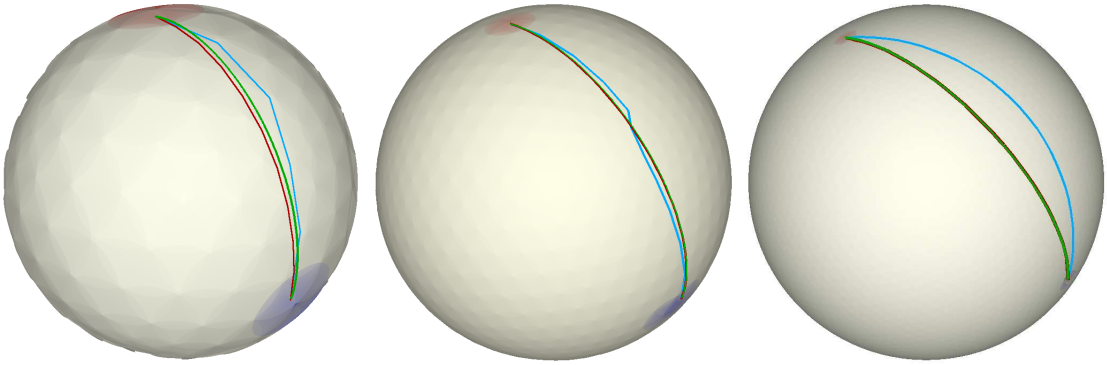
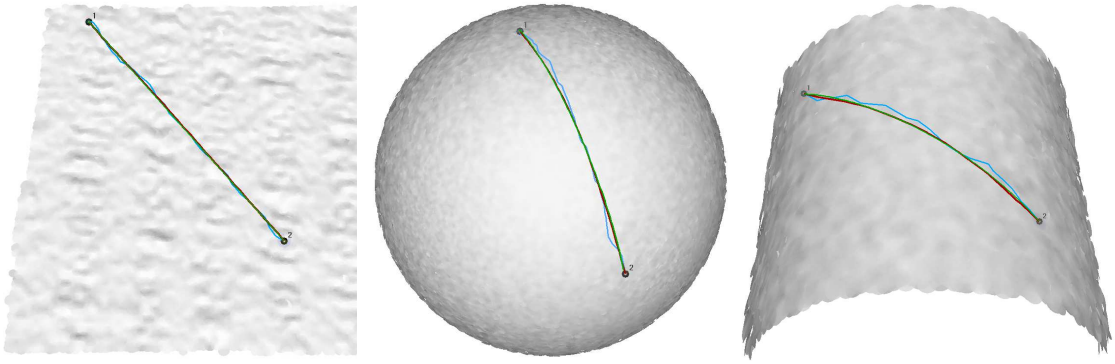


Figure B.6: Geodesic paths computed on a noisy surfel based model with outliers. Noise of level 100% and outliers (surfels in violet) have been added to the surfel based model of the Igea [188]. Outliers are generated by adding new noisy surfels with noise level 200% to the noisy model. Paths: initial path computed as shortest path on the graph with Dijkstra's algorithm (light-blue), path computed with our method on surfels (dark-red) and on MLS surface (dark-green), and path obtained with the HP method (black). [**see Figure 2.11 on page 35**]



(a) (b) (c)

Figure B.7: (a) Result of our method on surfel based models of unit spheres obtained from different subdivision levels using the Loop's scheme (3-(a), 4-(b), 5-(c)). Exact geodesic path computed analytically (green), Dijkstra's shortest path computed on the graph (light-blue), and geodesic path approximation computed with our method (dark-red). **[see Figure 2.12 on page 36]**



(a) (b) (c)

Figure B.8: Comparing the exact geodesic path (green) with the geodesic path approximation computed with our method (dark-red), and the initial solution computed on the graph with Dijkstra's algorithm (light-blue) on different noisy surfaces: a plane (a) a sphere (b), and a cylinder (c). **[see Figure 2.13 on page 36]**

Appendix C

Color plates of Chapter 3

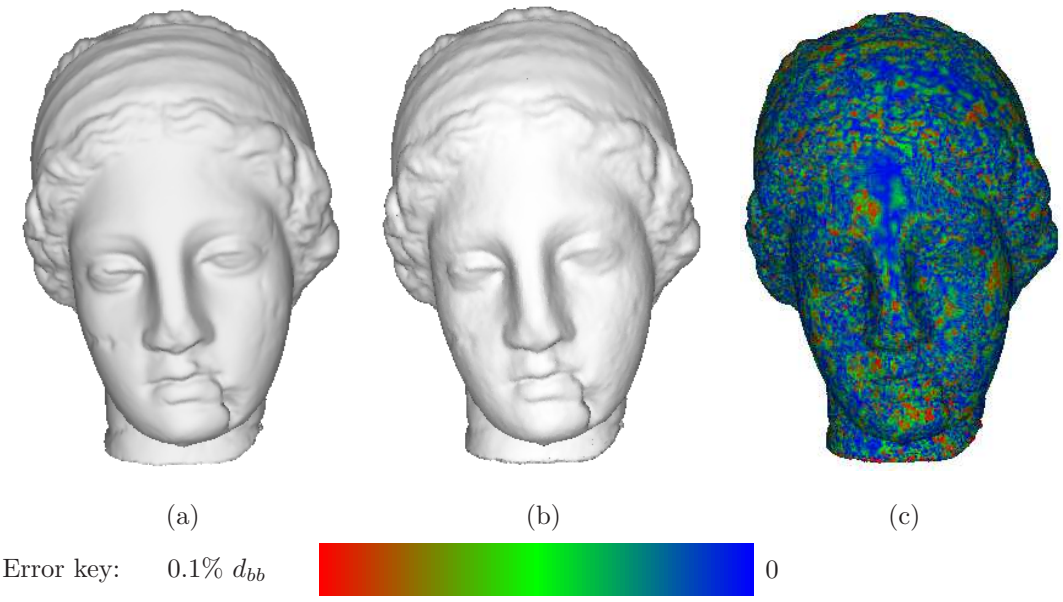


Figure C.1: Example of error measurement: (a) original *Igea* model [188]. (b) Compressed model. (c) Error mapped on the compressed model, total $E_{RMS} = 3.01 \cdot 10^{-4}\ d_{bb}$. **[see Figure 3.7 on page 55]**

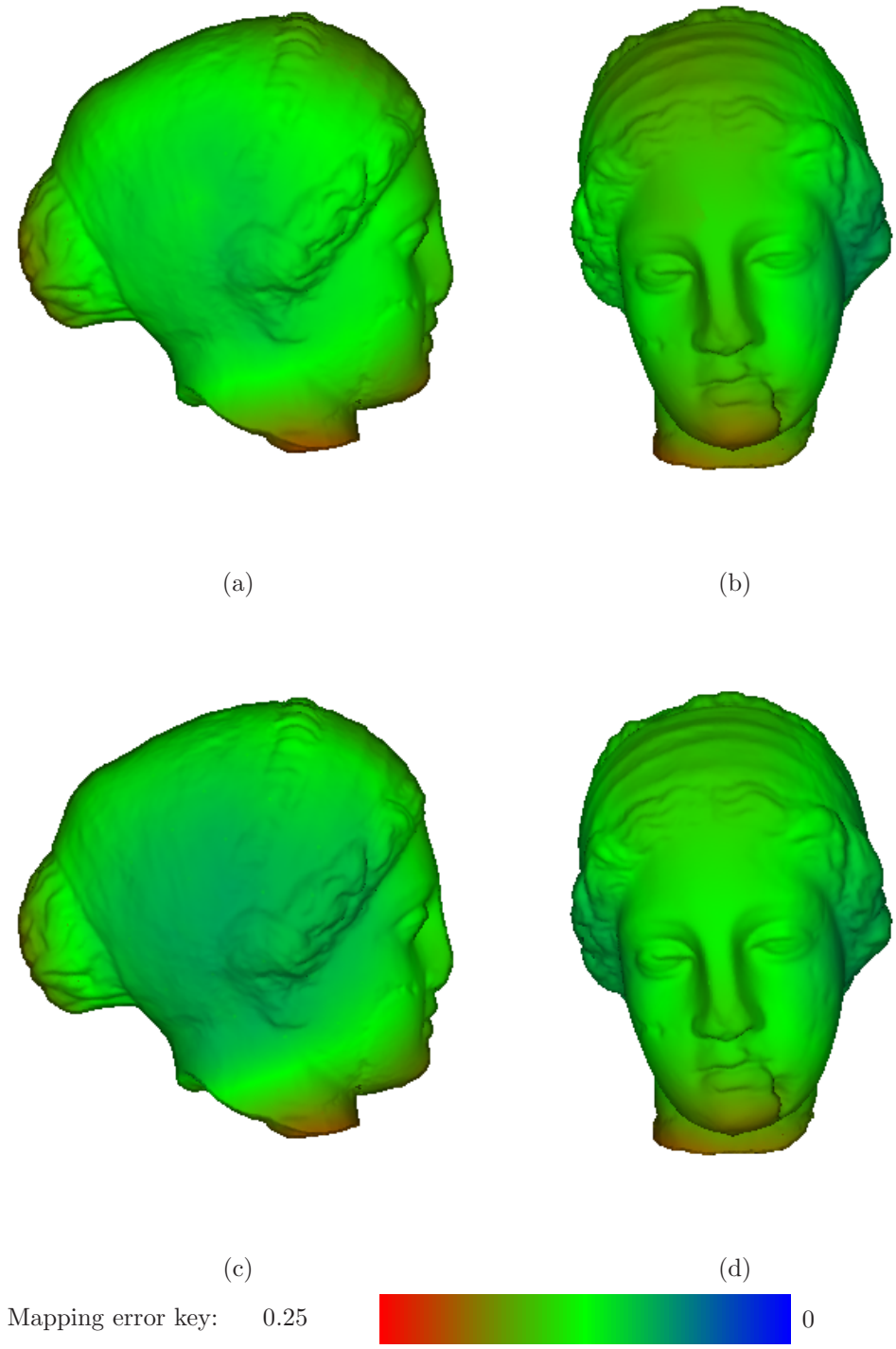


Figure C.2: Dense spherical MDS: The mapping error of each surfel in the Igea model [188] is color-coded. (a), (b): Using 10 reference surfels. (c), (d): Using 150 reference surfels. **[see Figure 3.1 on page 46]**

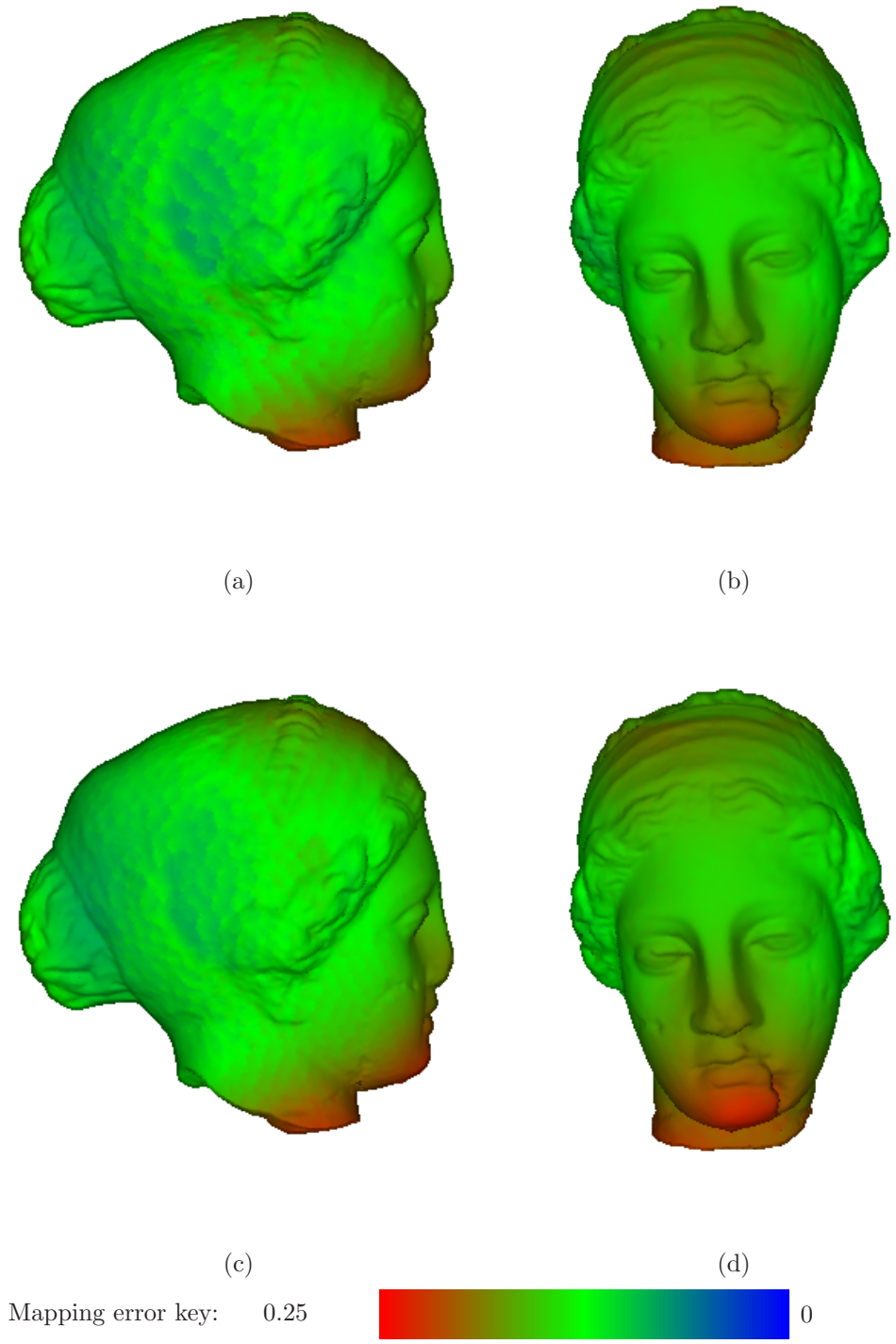


Figure C.3: Constrained spherical MDS [86] followed by interpolation. The mapping error of each surfel in the Igea model [188] is color-coded. (a), (b): Using 1000 reference surfels. (c), (d): Using 2000 reference surfels. **[see Figure 3.2 on page 47]**

(a) (b)

Figure C.4: Compression of a textured Igea model [188] (134345 surfels). (a) Original textured model. (b) Compression to 3.56 bits/surfel. [**see Figure 3.4 on page 50**]

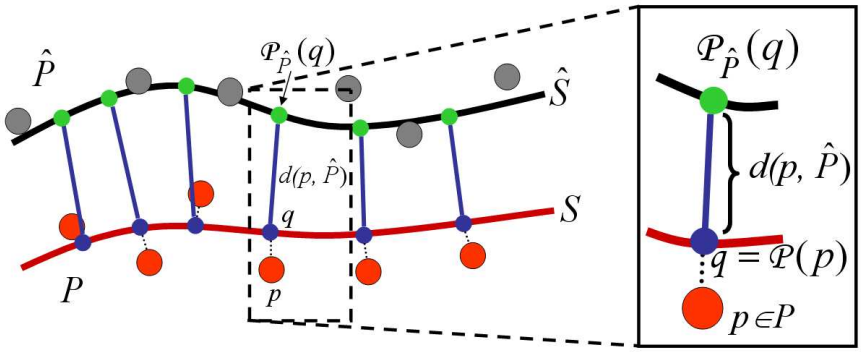


Figure C.5: Measuring the distance between two surfaces S (red curve) and $\hat{S}$ (black curve) represented by two point sets P (red dots) and $\hat{P}$ (grey dots), respectively. Each point $p \in P$ is projected to the point $q = \mathcal{P}_P(p) \in S$ (blue dot). The point $q \in S$ is then projected onto the surface $\hat{S}$, resulting in the point $\mathcal{P}_{\hat{P}}(q) \in \hat{S}$ (green dot). The distance $d(p, \hat{P})$ is the Euclidean distance between these two points, $\left\| q - \mathcal{P}_{\hat{P}}(q) \right\|$. [**see Figure 3.6 on page 52**]

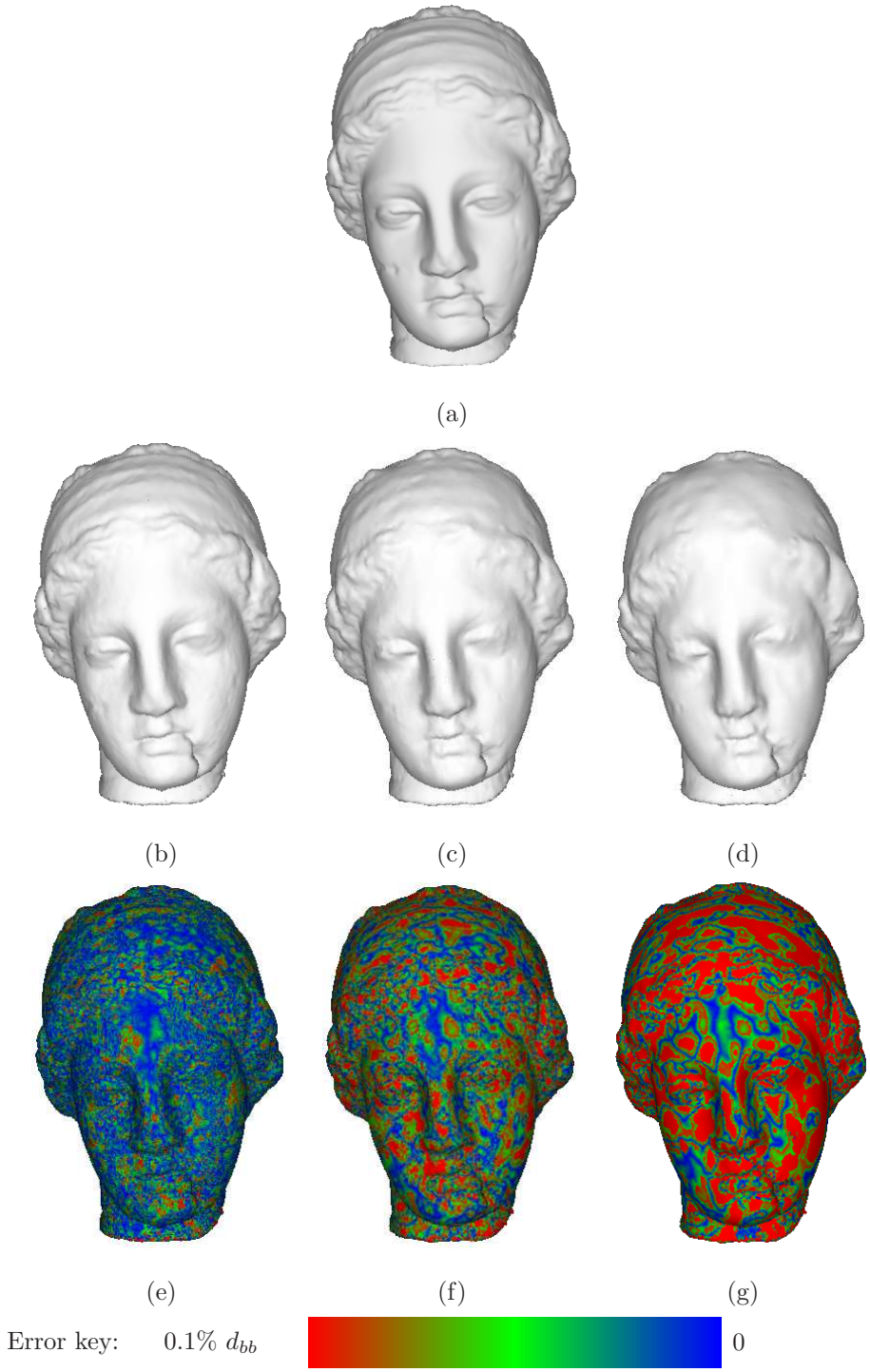


Figure C.6: Compression of the *Igea* model (geometry only). (a) Original model. (b) Compressed to 5.17 bits/surfel. (c) Compressed to 1.22 bits/surfel. (d) Compressed to 0.42 bits/surfel. (e) The point-wise reconstruction error in (b). (f) Error in (c). (g) Error in (d). **[see Figure 3.9 on page 56]**

(a) (b) (c)

Figure C.7: Compression of the *Igea* 3D model [188] with texture. (a) Original model, 134345 surfels, technically 120 bits/surfel. (b) Compression to 3.56 bits/surfel. (c) Compression to 0.89 bits/surfel. **[see Figure 3.10 on page 57]**

(a) (b) (c)

Figure C.8: Compression of the textured *Trilobite* model (3D model courtesy of Arius3D Inc. [10]). (a) Original model, 726027 surfels, technically 120 bits/surfel. (b) Compression to 6.3 bits/surfel. (c) Compression to 3.77 bits/surfel. **[see Figure 3.11 on page 57]**

Appendix D

Color plates of Chapter 4

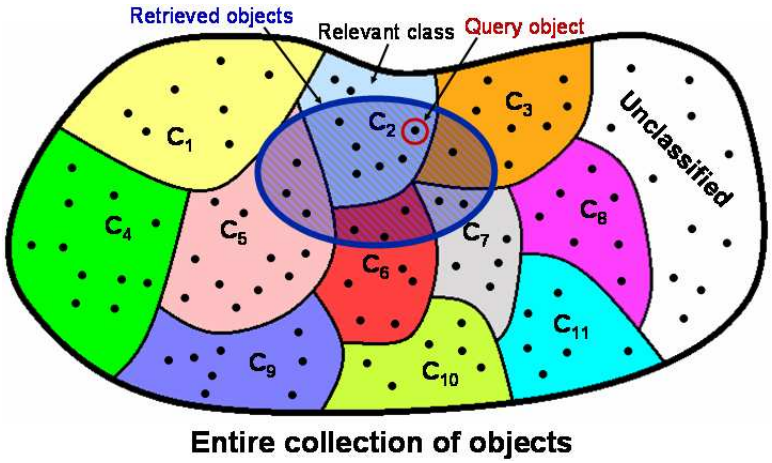
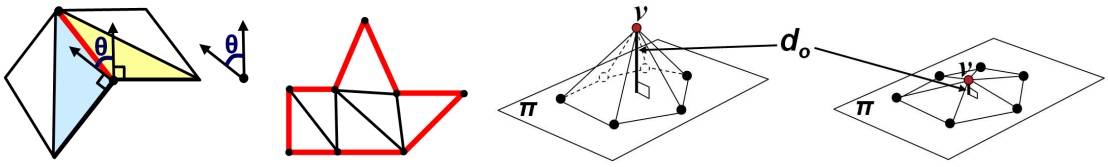


Figure D.1: Principle of a precision-recall diagram. Given a query object, usually only a portion of the retrieved objects is relevant for the query, i.e., those objects belonging to the same class of the query object (C_2). [**see Figure 4.2 on page 66**]



(a) (b) (c)

Figure D.2: Geometric features to preserve. Edges to preserve (in red): crease edges (a) and boundary edges (b). (c) Non-flat (left) and flat (right) triangular surface. [**see Figure 4.4 on page 79**]

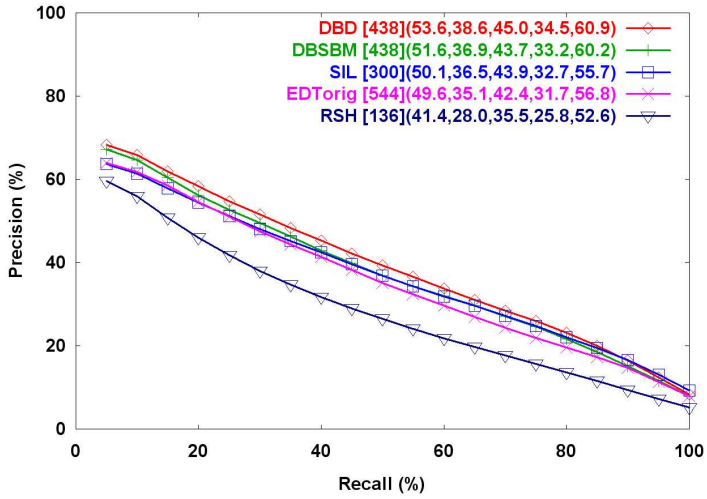
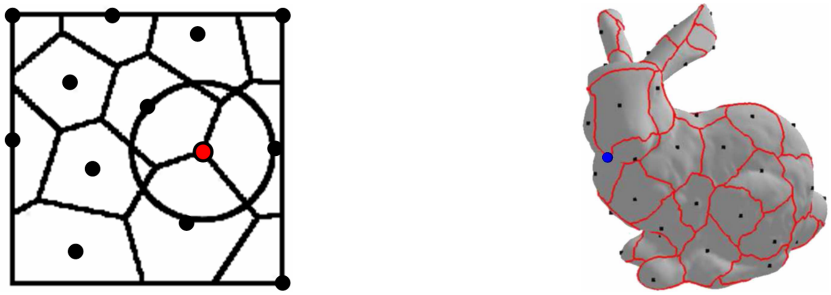
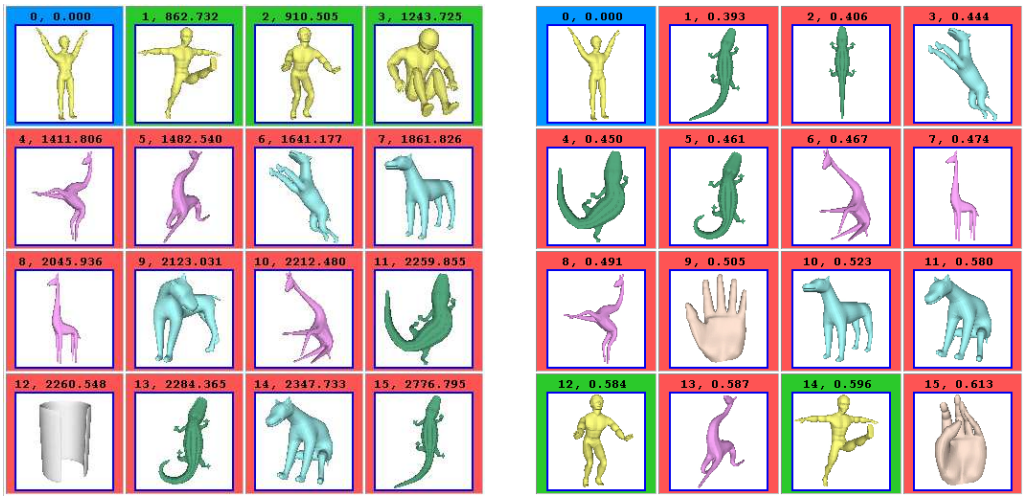


Figure D.3: Precision-recall diagram computed for different feature vectors (DBD, EDTorig, SIL, RSH) of the original triangles meshes with respect the proposed feature vector extracted from the corresponding surfel based models (DBSBM). Dimension of feature vectors are given in brackets. Between round parentheses different retrieval effectiveness measures defined in Section 4.1.2 are reported, i.e., ($\bar{P}_{50}$, $\bar{P}_{100}$, BEP, RP, NN). The same measure are clearly reported in Table 4.2. [**see Figure 4.13 on page 91**]



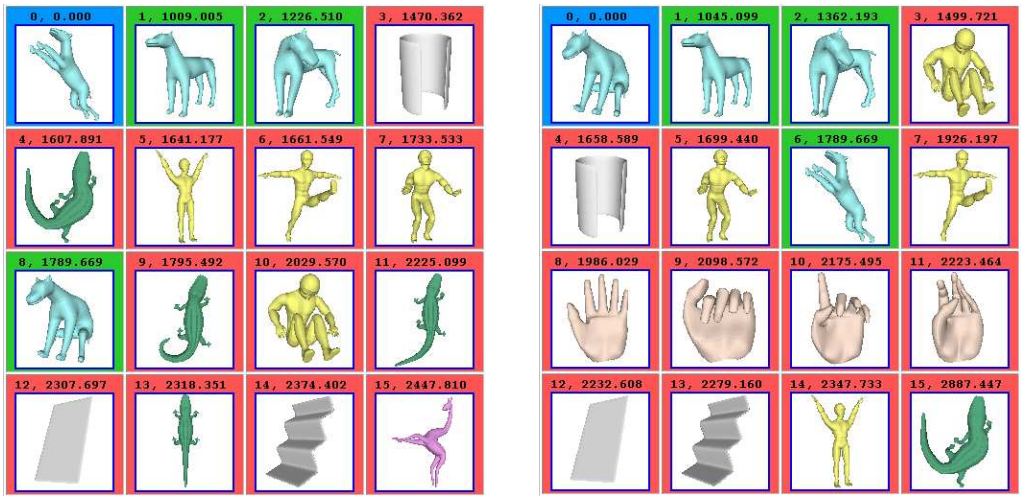
(a) (b)

Figure D.4: A farthest point visualized with respect to a bounded geodesic Voronoi diagram (BGVD) constructed on a plane and on a surface. (a) On a plane the farthest point (depicted in red) is located at the center of the largest circle, which does not contain any points (black points) generating the BVD. (b) BGVD constructed on a surface of a 3D model. A candidate farthest point is depicted in blue. [**see Figure 4.17 on page 107**]



(a) (b)

Figure D.5: Result of a query using the geodesic based (a) and the image based (b) shape descriptors on the collection of deformed surfel based models illustrated in Figure 4.20. The retrieved models are ranked according to their dissimilarity values to the query model (shown at position 0). The rank and the dissimilarity value of each model are shown as header of its icon. The icons have different layouts: blue (query model - top left), green (relevant models), and red (non-relevant models). [**see Figure 4.21 on page 117**]



(a) (b)

Figure D.6: Result of two queries performed using the geodesic based shape descriptor on the collection of deformed surfel based models illustrated in Figure 4.20. The retrieved models are ranked according to their dissimilarity values to the query model (shown at position 0). The rank and the dissimilarity value of each model are shown as header of its icon. The icons have different layouts: blue (query model - top left), green (relevant models), and red (non-relevant models). These are the only two queries where a relevant object is not retrieved as object of the same class of the query object, as shown by the models ranked at position 8 (a) and 6 (b) instead of at the first four positions. [**see Figure 4.23 on page 118**]

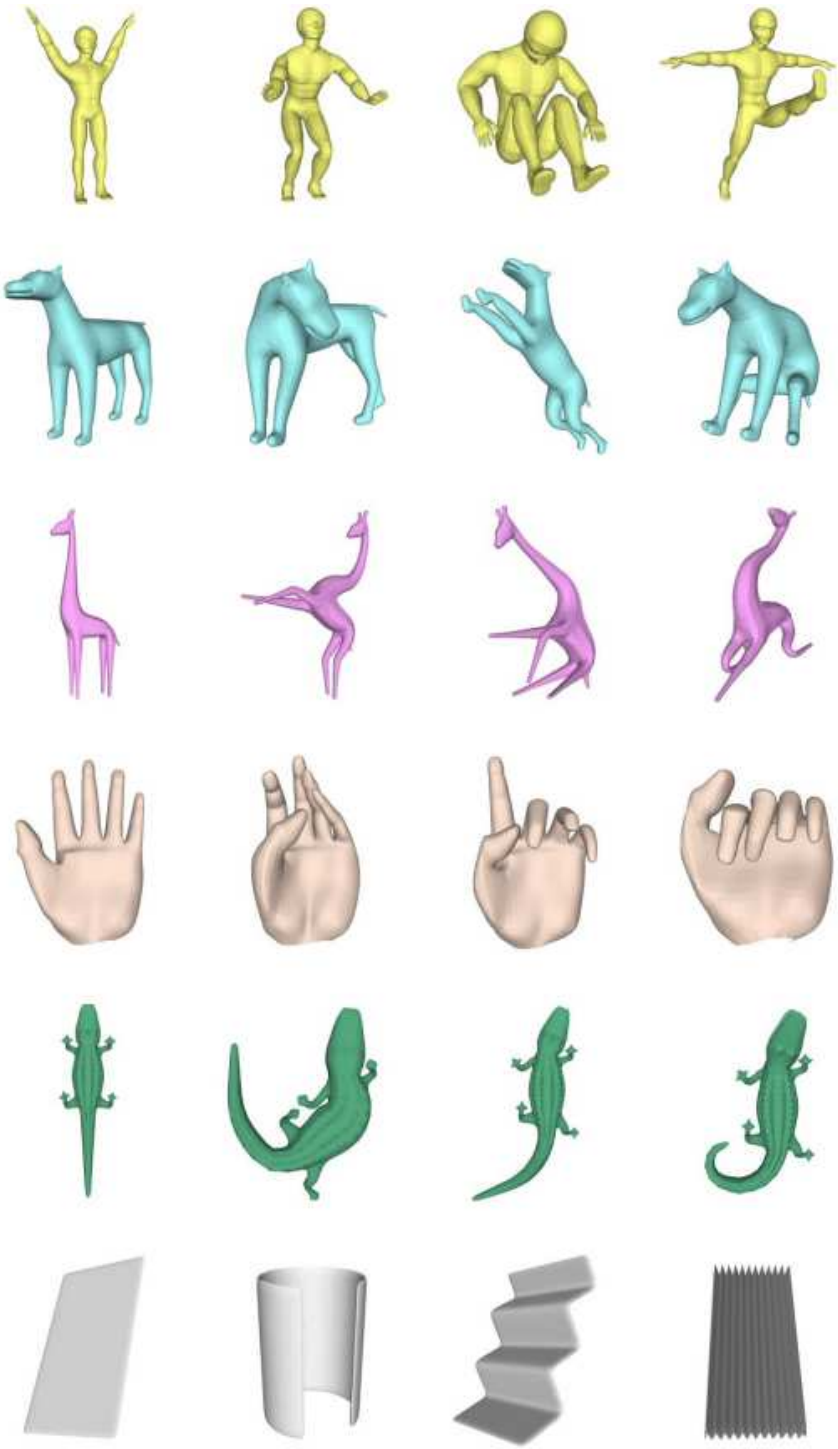


Figure D.7: Collection of surfel based models partitioned in classes of deformed objects. The models are arranged by rows. Each row identifies a class, and is associated to a specific color. Each class contains different models of the same object, which has been deformed with bending deformations. [**see Figure 4.20 on page 116**]

References

[1] 3D-DOCTOR: Vector-Based 3D Medical Modeling and Imaging Software. http://www.ablesw.com/3d-doctor/index.html. 39

[2] Bart Adams, Richard Keiser, Mark Pauly, Leonidas J. Guibas, Markus Gross, and Philip Dutr. Efficient raytracing of deforming point-sampled surfaces. *Computer Graphics Forum*, 24(3), 2005. 3

[3] A. Adamson and M. Alexa. On normals and projection operators for surfaces defined by point sets. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 149–156, 2004. 15, 20, 30

[4] Anders Adamson and Marc Alexa. Ray tracing point set surfaces. In *International Conference on Shape Modeling and Applications*, page 272, Los Alamitos, CA, USA, 2003. IEEE Computer Society. 3

[5] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):3–15, 2003. 2, 3, 15, 23, 29, 40, 52, 77

[6] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 347–354, New York, NY, USA, 2002. ACM Press. 102

[7] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 213–222, New York, NY, USA, 2000. ACM Press. 22

[8] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Trans. on Graphics*, 23(3):264–270, 2004. 2, 15, 23, 30

[9] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *SSD '99: Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 207–226, London, UK, 1999. Springer-Verlag. 69

[10] Arius3D Inc. http://www.arius3d.com. 1, 57, 127, 138

[11] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999. 65

[12] Chandrajit L. Bajaj and Daniel Schikore. Topology preserving data simplification with error bounds. *Computers & Graphics*, 22(1):3–12, 1998. 93

[13] S.H. Baloch, H. Krim, I. Kogan, and D. Zenkov. Rotation invariant topology coding of 2d and 3d objects using morse theory. In *Proceedings of the IEEE International Conference on Image Processing 2005*, pages 796–799, 2005. 93, 94

[14] C. Bankirer and N. Kiryati. Efficient surface flattening. Technical report, Department of Electrical Engineering-Systems Faculty of Engineering, Tel Aviv University,Tel Aviv, ISRAEL, 2005. 43, 121

[15] Stephen T. Barnard and Martin A. Fischler. Computational stereo. *ACM Comput. Surv.*, 14(4):553–572, 1982. 39

[16] Vic Barnett and Toby Lewis. *Outliers in Statistical Data*. John Wiley and Sons Ltd, 3 edition, February 1994. 18

[17] CCCC: Content based classification of 3D-models by capturing spatial characteristics. http://merkur01.inf.uni-konstanz.de/CCCC/. 60, 75, 80, 122

[18] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 28:23652385, 1998. 70, 96

[19] Anup Basu, Irene Cheng, and Tao Wang. Balanced incomplete designs for 3D perceptual quality estimation. In *Proceedings of the IEEE International Conference on Image Processing*, September 2005. 122

[20] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38, New York, NY, USA, 2003. ACM Press. 19

[21] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002. 96

[22] Dmitriy Bespalov, Ali Shokoufandeh, William C. Regli, and Wei Sun. Scale-space representation of 3d models and topological matching. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 208–215, New York, NY, USA, 2003. ACM Press. 63, 64, 95

[23] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *ICDT '99: Proceeding of the 7th International Conference on Database Theory*, pages 217–235, London, UK, 1999. Springer-Verlag. 65

[24] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematics Society*, 35:99–110, 1943. 111

[25] Silvia Biasotti. Topological techniques for shape understanding. In *Central European Seminar on Computer Graphics CESCG 2001*, pages 163–172, Bratislava, Slovakia 2001, 2001. 94

[26] Silvia Biasotti, Simone Marini, Michela Mortara, and Giuseppe Patanè. An overview on properties and efficacy of topological skeletons in shape modelling. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 245, Washington, DC, USA, 2003. IEEE Computer Society. 69, 95

[27] Silvia Biasotti, Simone Marini, Michela Mortara, Giuseppe Patanè, Michela Spagnuolo, and Bianca Falcidieno. 3d shape matching through topological structures. In *DGCI*, pages 194–203, 2003. 94

[28] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001. 65

[29] Jan Böhm and Maria Pateraki. From point samples to surfaces - on meshing and alternatives. In *ISPRS Commission V Symposium Image Engineering and Vision Metrology*, pages 25–27, September 2006. 20

[30] I. Borg and P. Groenen. *Modern Multidimensional Scaling*. Springer Series in Statistics. Springer, 1997. 41, 43, 100

[31] Mario Botsch, Michael Spernat, and Leif Kobbelt. Phong splatting. In *EG Symp. on Point-Based Graphics*, pages 25–32, 2004. 3

[32] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A multi-resolution data structure for two-dimensional morse-smale functions. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 19, Washington, DC, USA, 2003. IEEE Computer Society. 93

[33] Peer-Timo Bremer and John C. Hart. A sampling theorem for mls surfaces. In *Proc. EG Symp. on Geometry Processing*, 2005. 2, 15, 23

[34] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Three-dimensional face recognition. *Int. J. Comput. Vision*, 64(1):5–30, 2005. 96

[35] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Efficient computation of isometry-invariant distances between surfaces. *SIAM J. Sci. Comput.*, 28(5):1812–1836, 2006. 97, 98

[36] A.M. Bronstein, M.M. Bronstein, and R. Kimmel. Expression-invariant face recognition via spherical embedding. In *Proceedings of the IEEE International Conference on Image Processing*, pages 756–759, 2005. 40, 41

[37] M.Z. Brown and G.D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1007, August 2003. 39

[38] B. Bustos, O. Deussen, S. Hiller, and D. Keim. A graphics hardware accelerated algorithm for nearest neighbor search. In *Proc. International Conference on Computational Science (ICCS'06) Part IV, LNCS 3994*, pages 196–199. Springer, 2006. 63, 65

[39] Benjamin Bustos, Daniel Keim, Dietmar Saupe, Tobias Schreck, and Dejan Vranić. Automatic selection and combination of descriptors for effective 3d similarity search. In *ISMSE '04: Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04)*, pages 514–521, Washington, DC, USA, 2004. IEEE Computer Society. 71

[40] Benjamin Bustos, Daniel Keim, Dietmar Saupe, Tobias Schreck, and Dejan Vranić. An experimental comparison of feature-based 3d retrieval methods. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pages 215–222, Washington, DC, USA, 2004. IEEE Computer Society. 63, 64, 70, 71, 72, 83, 123

[41] Benjamin Bustos, Daniel Keim, and Tobias Schreck. A pivot-based index structure for combination of feature vectors. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1180–1184, New York, NY, USA, 2005. ACM Press. 63, 65

[42] A. Cachia, Jean-Francois Mangin, Denis Rivière, Dimitri Papadopoulos-Orfanos, Isabelle Bloch, and Jean Régis. Gyral parcellation of the cortical surface using geodesic voronoi diagrams. In *MICCAI '02: Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part I*, pages 427–434, London, UK, 2002. Springer-Verlag. 4

[43] R. Campbell and P. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166–210, 2001. 1, 60, 73, 95

[44] Tom Capizzi. *Inspired 3D Modeling and Texture Mapping.* Course Technology Ptr; 1 edition, August 2002. 1, 39

[45] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11(2):149–187, 2005. 64, 68, 70, 73

[46] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01*, pages 67–76, 2001. 2, 15, 23

[47] Edwin Earl Catmull. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, University of Utah, Salt Lake City, December 1974. 3

[48] F. Cazals, J. Giesen, M. Pauly, and A. Zomorodian. Conformal alpha shapes. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 55–61, 2005. 2

[49] Philippe Chanzy, Luc Devroye, and Carlos Zamora-Cura. Analysis of range search for random k-d trees. *Acta Informatica*, 37(4-5):355–383, 2000. 26

[50] Ding-Yun Chen and Ming Ouhyoung. A 3d object retrieval system based on multi-resolution reeb graph. In *Proc. of Computer Graphics Workshop*, Tainan, Taiwan, June 2002. 94

[51] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003. Eurographics 2003 Conference Proceedings. 69, 70

[52] Jindong Chen and Yijie Han. Shortest paths on a polyhedron. *Int. J. Comput. Geom. Appl.*, 6(2):127–144, 1996. 15

[53] Vincent Cicirello and William C. Regli. Machining feature-based comparisons of mechanical parts. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 176, Washington, DC, USA, 2001. IEEE Computer Society. 95

[54] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998. 40, 51

[55] Liviu Coconu and Hans-Christian Hege. Hardware-accelerated point-based rendering of complex scenes. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 43–52, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. 77

[56] Jonathan D. Cohen, Daniel G. Aliaga, and Weiqiang Zhang. Hybrid simplification: combining multi-resolution polygon and point rendering. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 37–44, Washington, DC, USA, 2001. IEEE Computer Society. 78, 102

[57] A. Collins, A. Zomorodian, G. Carlsson, and L. Guibas. A barcode shape descriptor for curve point cloud data. *Computers and Graphics*, 28:881–894, 2004. 64, 68, 73

[58] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986. 104

[59] William Cook and Andre Rohe. Computing minimum-weight perfect matchings. *INFORMS J. on Computing*, 11(2):138–148, 1999. 113

[60] Nicu D. Cornea, M. Fatih Demirci, Deborah Silver, Ali Shokoufandeh, Sven J. Dickinson, and Paul B. Kantor. 3d object retrieval using many-to-many matching of curve skeletons. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05)*, pages 368–373, Washington, DC, USA, 2005. IEEE Computer Society. 95, 96

[61] Nicu D. Cornea, Deborah Silver, and P. Min. Curve-skeleton properties, applications and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 2006. 96

[62] Jonathan R. Corney, Heather J. Rea, Doug E. R. Clark, John Pritchard, Michael Breaks, and Roddy MacLeod. Coarse filters for shape matching. *IEEE Computer Graphics and Applications*, 22(3):65–74, 2002. 69

[63] T.F. Cox and M.A.A. Cox. Multidimensional scaling on a sphere. *Communications in Statistics: Theory and Methods*, 20:2943–2953, 1991. 40, 41, 42, 44

[64] Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 495–498, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press. 77

[65] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, New York, NY, USA, 1996. ACM Press. 100

[66] T. Darom, M.R. Ruggeri, D. Saupe, and N. Kiryati. Processing of textured surfaces represented as surfel sets: Representation, compression and geodesic paths. In *Proceedings of the IEEE International Conference on Image Processing*, pages 605–608, 2005. 4, 5, 39, 41, 121

[67] T. Darom, M.R. Ruggeri, D. Saupe, and N. Kiryati. Compression of textured surfaces represented as surfel sets. *EURASIP journal on Image Communication*, 21(9):770–786, October 2006. 4, 5, 39, 121

[68] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry-and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1996. ACM Press. 1, 39

[69] T. K. Dey, H. Edelsbrunner, and S. Guha. Computational topology. *In Advances in Discrete and Computational Geometry (Contemporary mathematics 223)*, pages 109–143, 1999. 92, 93

[70] T. K. Dey and J. Sun. An adaptive mls surface for reconstruction with guarantees. In *Proc. EG Symp. on Geometry Processing*, pages 43–352, 2005. 2, 15, 23, 30

[71] Tamal K. Dey, Joachim Giesen, and Samrat Goswami. Shape segmentation and matching with flow discretization. In M. Smid Eds. F. Dehne, J.-R. Sack, editor, *Proc. Workshop Algorithms Data Strucutres (WADS 03), LNCS 2748*, pages 25–36, 2003. 70

[72] Tamal K. Dey, Joachim Giesen, and Samrat Goswami. Shape segmentation and matching from noisy point clouds. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 193–199, Zürich, Switzerland, June 2004. 68, 93

[73] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 330–339, New York, NY, USA, 2004. ACM Press. 2

[74] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 330–339, 2004. 22

[75] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976. 10, 11, 31

[76] Daniel Dunbar and Greg Humphreys. A spatial data structure for fast poisson-disk sample generation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 503–508, New York, NY, USA, 2006. ACM Press. 106

[77] G. Dutton. Locational properties of quaternary triangular meshes. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 901–910, July 1990. 49

[78] Rex A. Dwyer. The expected size of the sphere-of-influence graph. *Comput. Geom. Theory Appl.*, 5(3):155–164, 1995. 21

[79] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. on Graphics*, 9(2):160–169, April 1990. 50

[80] David H. Eberly. *3D game engine design: a practical approach to real-time computer graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000. 1

[81] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers; 2nd edition, September 1998. 1, 39

[82] Sabry El-Hakim. A practical approach to creating precise and detailed 3d models from single and multiple views. In *Proceedings of the XIX ISPRS Congress*, pages 202–209, Amsterdam, Holland, July 2000. IEEE Computer Society. 39

[83] Mohamed EL-MEHALAWI and R. Allen MILLER. A database system of mechanical components based on geometric and topological similarity. part ii: indexing, retrieval, matching and similarity assessment. *Journal of Computer-Aided Design*, 35(1):95–105, January 2003. 69, 95

[84] A. Elad, Y. Keller, and R. Kimmel. Texture mapping via spherical multi-dimensional scaling. In *Proc. of Scale Space*, pages 443–455, 2005. 40, 41

[85] A. Elad and R. Kimmel. Bending invariant representations for surfaces. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 168–174, 2001. 4, 9, 96, 100, 114

[86] A. Elad and R. Kimmel. Spherical flattening of the cortex surface. In R. Malladi, editor, *Geometric Methods in Biomedical Image Processing*. Springer, 2002. 41, 42, 43, 44, 45, 47, 48, 121, 135

[87] Asi Elad and Ron Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003. 64, 96, 100, 102, 114

[88] Michael Elad, Ayellet Tal, and Sigal Ar. Content based retrieval of vrml objects: an iterative and interactive approach. In *Proceedings of the sixth Eurographics workshop on Multimedia 2001*, pages 107–118, New York, NY, USA, 2002. Springer-Verlag New York, Inc. 96

[89] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, September 1997. 43, 104, 106, 107, 108

[90] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993. 1, 39, 100

[91] S. Fleishman, D. Cohen-Or, M. Alexa, and C. T.Silva. Progressive point set surfaces. *ACM Trans. on Graphics*, 22(4):997–1011, October 2003. 40, 53

[92] Lille France: 3D Retrieval FOX-MIIRE research group. http://www-rech.enic.fr:8080/3Dretrieval/. 60, 61, 75

[93] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15:1691–1704, 1980. 43, 49

[94] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Trans. Graph.*, 22(1):83–105, 2003. 60, 61, 64, 69, 90

[95] Andrew V. Goldberg. Shortest path algorithms: Engineering aspects. In *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*, pages 502–513, London, UK, 2001. Springer-Verlag. 21, 45

[96] Rafael C. Gonzalez and Paul Wintz. *Digital image processing (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987. 87

[97] Markus Gross. Getting to the point...? *IEEE Comput. Graph. Appl.*, 26(5):96–99, 2006. 3, 40

[98] J. P. Grossman and William J. Dally. Point sample rendering. In *Rendering Techniques 98*, pages 181–192, July 1998. 3, 76, 77, 102

[99] R. Grossmann, N. Kiryati, and R. Kimmel. Computational surface flattening: A voxel-based approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:433–441, 2002. 4, 9, 42, 121

[100] MPEG-7 Requirements Group. Overview of the mpeg-7 standard. Technical Report V.8. ISO/IEC N4980, MPEG-7, Klangenfurt, July 2002, Klangenfurt, July 2002. 67

[101] MPEG-7 Video Group. Description of core experiments for motion and shape. Technical Report ISO/IEC N3397, MPEG-7, Klangenfurt, July 2002, Geneva, June 2000. 67

[102] X. Gu, S.J. Gortler, and H. Hoppe. Geometry images. In *Proceedings of SIGGRAPH*, pages 355–361, 2002. 40, 48

[103] A. Ben Hamza and Hamid Krim. *Discrete Geometry for Computer Imagery*, volume 2886, chapter Geodesic Object Representation and Recognition, pages 378–387. Springer Berlin / Heidelberg, 2003. 63, 96, 99

[104] Sariel Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. In *Symposium on Computational Geometry*, pages 359–365, 1997. 4, 9

[105] John C. Hart. Computational topology for shape modeling. In *SMI '99: Proceedings of the International Conference on Shape Modeling and Applications*, pages 36–45, Washington, DC, USA, 1999. IEEE Computer Society. 92

[106] M. Heczko, D. Keim, D. Saupe, and D. V. Vranić. Verfahren zur Ähnlichkeitssuche auf 3d-objekten. *Datenbank-Spektrum Zeitschrift für Datenbanktechnologie*, 2(2):54–63, February 2002. 70, 83, 90

[107] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM Press. 63, 64, 69, 93, 94, 96, 99

[108] Stefan Hiller, Oliver Deussen, and Alexander Keller. Tiled blue noise samples. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 265–272. Aka GmbH, 2001. 106

[109] Stefan Hiller, Heino Hellwig, and Oliver Deussen. Beyond stippling - methods for distributing objects on the plane. *Computer Graphics Forum*, 22(3):515–522, 2003. 104

[110] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004. 19

[111] Michael Hofer and Helmut Pottmann. Energy-minimizing splines in manifolds. *ACM Trans. on Graphics*, 23(3):284–293, 2004. 4, 9, 16, 17, 30, 44, 120, 121

[112] H. Hoppe and E. Praun. Shape compression using spherical geometry images. In N. Dodgson, M. Floater, and M. Sabin, editors, *Advances in Multiresolution for Geometry Modelling*, pages 27–46. Springer-Verlag, 2005. 42

[113] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26, New York, NY, USA, 1993. ACM Press. 102

[114] Berthod Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686, December 1984. 71

[115] Jianghai Hu, Maria Prandini, Karl Henrik Johansson, and Shankar Sastry. Hybrid geodesics as optimal solutions to the collision-free motion planning problem. In *Hybrid Systems: Computation and Control : 4th International Workshop*, pages 305–318, 2001. 4, 9

[116] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. *Proceedings of SIGGRAPH 99*, pages 409–416, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California. 61

[117] Greece: 3D Search Informatics & Telematics Institiute. http://3d-search.iti.gr/3DSearch. 60, 75

[118] 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 Press. 69, 95

[119] J.W. Jaromczyk and G.T. Toussaint. Relative neighborhood graphs and their relatives. *P-IEEE*, 80:1502–1517, 1992. 18, 21

[120] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, 1999. 102

[121] Thouis R. Jones. Efficient generation of poisson-disk sampling patterns. *journal of graphics tools*, 11(2):27–36, 2006. 106

[122] M. Kageura and K. Shimada. Finding the shortest path on a polyhedral surface and its application to quality assurance of electric components. *Journal of Mechanical Design*, 126(6):1017–1026, November 2004. 15

[123] Takashi Kanai and Hiromasa Suzuki. Approximate shortest path on a polyhedral surface and its applications. *Computer-Aided Design*, 33(11):801–811, 2001. 15

[124] S. B. Kang and K. Ikeuchi. 3-d object pose determination using complex egi. Technical Report CMU-RI-TR-90-18, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, October 1990. 71

[125] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. on Graphics*, pages 954–961, 2003. 4, 9

[126] 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. 69

[127] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of SIGGRAPH*, pages 271–278, 2000. 48, 51

[128] R. Kimmel and N. Kiryati. Finding shortest paths on surfaces by fast global approximation and precise local refinement. *Int. Journal of Pattern Recogn. and Artif. Intellig.*, 10:643–656, 1996. 15

[129] Danil Kirsanov. *Minimal Discrete Curves and Surfaces.* PhD thesis, Applied Math., Harvard University, 2004. 16

[130] N. Kiryati and G. Székely. Estimating shortest paths and minimal distances on digitized three dimensional surfaces. *Pattern Recognition*, 26:1623–1637, 1993. 15

[131] Jan Klein and Gabriel Zachmann. Point cloud surfaces using geometric proximity graphs. *Computers and Graphics*, 28(6):839–850, December 2004. 16, 18, 30

[132] Jan Klein and Gabriel Zachmann. Proximity graphs for defining surfaces over point clouds. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 131–138, 2004. 18, 19

[133] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004. 3, 15, 20, 40

[134] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98*, pages 105–114, 1998. 4, 9

[135] Konstanz 3D Model Repository. http://www.inf.uni-konstanz.de/cgip/projects/surfac/. 21, 37, 38

[136] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive wang tiles for real-time blue noise. *ACM Transactions on Graphics*, 25(3 (Proc. SIGGRAPH 2006)):to appear, 2006. 104

[137] Bernhard Korte and Jens Vygen, editors. *Combinatorial Optimization, Theory and Algorithms.* Springer-Verlag, 3rd edition, 2000. 111

[138] M. Körtgen, G.-J. Park, M. Novotni, and R. Klein. 3d shape matching with 3d shape contexts. In *The 7th Central European Seminar on Computer Graphics*, April 2003. 69

[139] Solomon Kullback. *Information theory and statistics.* Dover Publications, 1968. 111

[140] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006. 22, 25

[141] Sungyeol Lee, Joonhee Han, and Haeyoung Lee. Straightest paths on meshes by cutting planes. In Myung-Soo Kim and Kenji Shimada, editors, *Geometric Modeling and Processing - GMP 2006, 4th International Conference, Pittsburgh, PA, USA, July 26-28, 2006, Proceedings*, volume 4077 of *Lecture Notes in Computer Science*, pages 609–615. Springer, 2006. 16

[142] Sylvain Lefebvre and Hugues Hoppe. Perfect spatial hashing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 579–588, New York, NY, USA, 2006. ACM Press. 2

[143] M. Levoy and T. Whitted. The use of points as display primitives. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 1985. 3

[144] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 1, 39, 100

[145] Haibin Ling and David W. Jacobs. Using the inner-distance for classification of articulated shapes. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 719–726, Washington, DC, USA, 2005. IEEE Computer Society. 96

[146] Haibin Ling and K. Okada. Diffusion distance for histogram comparison. In *CVPR'06: IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 111

[147] Haibin Ling and K. Okada. Emd-l1: An efficient and robust algorithm for comparing histogram-based descriptors. In *ECCV06: Proceedings of the 9th European Conference on Computer Vision*, 2006. 63, 111

[148] Dani Lischinski and Ari Rappoport. Image-based rendering for non-diffuse synthetic scenes. In *EG Workshop on Rendering Techniques*, pages 301–314, 1998. 76

[149] J. Löffler. Content-based retrieval of 3d models in distributed web databases by visual shape information. In *IV '00: Proceedings of the International Conference on Information Visualisation*, page 82, Washington, DC, USA, 2000. IEEE Computer Society. 69

[150] Charles Loop. *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, CM.S. Mathematics, University of Utah, 1987. 31, 32

[151] W.E. Lorenson and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21:163–169, 1987. 2, 39

[152] Macopt: Conjugate Gradient optimizer. http://wol.ra.phy.cam.ac.uk/mackay/c/macopt.html. 26

[153] Diego Macrini, Ali Shokoufandeh, Sven Dickinson, Kaleem Siddiqi, and Steven Zucker. View-based 3-d object recognition using shock graphs. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3*, page 30024, Washington, DC, USA, 2002. IEEE Computer Society. 69

[154] Songrit Maneewongvatana and David M. Mount. An empirical study of a new approach to nearest neighbor searching. In *ALENEX '01: Revised Papers from the Third International Workshop on Algorithm Engineering and Experimentation*, pages 172–187, London, UK, 2001. Springer-Verlag. 2

[155] do Carmo Manfredo Perdigao. *Riemannian geometry*. Birkhuser, 1993. 13

[156] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. *ACM Trans. Graph.*, 21(3):427–437, 2002. 1

[157] Michael McCool and Eugene Fiume. Hierarchical poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface '92*, pages 94–105, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. 106

[158] David McWherter, Mitchell Peabody, William C. Regli, and Ali Shokoufandeh. An approach to indexing databases of graphs. Technical Report DU-MCS-01-03, Drexel University, Computer Science Department, June 2001. 69

[159] Boris Mederos, Nina Amenta, Luiz Vehlo, and Luiz Henrique de Figueiredo. Surface reconstruction from noisy point clouds. In *Proc. EG Symp. on Geometry Processing*, pages 53–62, 2005. 2, 22

[160] Barbara J. Meier. Painterly rendering for animation. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 477–484, New York, NY, USA, 1996. ACM Press. 102, 104

[161] F. Mémoli and G. Sapiro. Distance functions and geodesics on points clouds. In *IEEE Workshop on Variational Geometric and Level Set Methods in Computer Vision*, 2003. 16, 18, 34, 107

[162] Facundo Mémoli and Guillermo Sapiro. Comparing point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 32–40, New York, NY, USA, 2004. ACM Press. 68, 96, 97, 98, 101, 110, 114

[163] Facundo Mémoli and Guillermo Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Found. Comput. Math.*, 5(3):313–347, 2005. 96, 97, 98, 110

[164] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH Comput. Graph.*, 25(4):157–164, 1991. 104

[165] C. Moenning and N. A. Dodgson. Intrinsic point cloud simplification. In *GraphiCon*, 2004. 4, 9

[166] Carsten Moenning and Neil A. Dodgson. Fast marching farthest point sampling. In *EUROGRAPHICS*, Granada, Spain, September 2003. 76, 106, 107

[167] Carsten Moenning and Neil A. Dodgson. Fast marching farthest point sampling. Technical Report UCAM-CL-TR-562, University of Cambridge, Computer Laboratory, April 2003. 76, 106, 107, 108

[168] Carsten Moenning and Neil A. Dodgson. Fast marching farthest point sampling for point clouds and implicit surfaces. Technical Report 565, University of Cambridge, Computer Laboratory, April 2003. 106, 107

[169] Saied Moezzi, Arun Katkere, Don Y. Kuramura, and Ramesh Jain. Reality modeling and visualization from multiple video sequences. *IEEE Comput. Graph. Appl.*, 16(6):58–63, 1996. 1, 39

[170] J. Mundy, C. Huang, J. Liu, W. Hoffman, D. Forsyth, C. Rothwell, A. Zisserman, S. Utcke, and O. Bournez. MORSE: A 3D object recognition system based on geometric invariants. In *ARPA Image Understanding Workshop*, pages 1393–1402, 1994. 96

[171] L. R. Nackman. Two–dimensional critical point configuration graphs. *IEEE Trans. on PAMI*, 6(4):442–450, July 1984. 94

[172] P. J. Narayanan, Peter W. Rander, and Takeo Kanade. Constructing virtual worlds using dense stereo. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 3, Washington, DC, USA, 1998. IEEE Computer Society. 1, 39

[173] Japan: Ogden VI rxw (A Search Engine for 3D Models) National Institute of Multimedia Education. http://ship.nime.ac.jp/ 60, 61, 75

[174] Diego Nehab and Philip Shilane. Stratified point sampling of 3D models. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 49–56, Zürich, Switzerland, June 2004. 102, 103, 104

[175] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Trans. Graph.*, 23(3):613–622, 2004. 93

[176] M. Novotni and R. Klein. Computing geodesic distances on triangular meshes. In *WSCG'2002*, pages 8431–8435, 2002. 16

[177] T. Ochotta and D. Saupe. Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 103–112, Zürich, Switzerland, June 2004. 40, 51, 52, 53

[178] Ryutarou Ohbuchi, Tomo Otagiri, Masatoshi Ibato, and Tsuyoshi Takei. Shape-similarity search of three-dimensional models using parameterized statistics. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 265, Washington, DC, USA, 2002. IEEE Computer Society. 69, 72

[179] Ryutarou Ohbuchi and Tsuyoshi Takei. Shape-similarity comparison of 3d models using alpha shapes. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 293, Washington, DC, USA, 2003. IEEE Computer Society. 72, 102

[180] Y. Ohtake, A. Belyaev, and H.-P. Seidel. 3d scattered data approximation with adaptive compactly supported radial basis functions. In *Shape Modeling International 2004*, pages 31–39, 2004. 2, 15, 23

[181] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, 2002. 63, 69, 72, 102, 111, 112

[182] D. Page, A. Koschan, M. Abidi, and J. Overholt. Ridge-valley path planning for 3d terrains. In *ICRA '06: Proc. IEEE International Conference on Robotics and Automation*, pages 119–124, Orlando, USA, May 2006. 4

[183] E. Paquet, A. Murching, T. Naveen, A. Tabatabai, and M. Rioux. Description of shape information for 2-d and 3-d objects. *Signal Processing: Image Communication*, 16(1):103–122, September 2000. 69

[184] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society. 40, 51, 73, 92

[185] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003. 3, 73, 77

[186] G. Peyr and L. Cohen. Surface segmentation using geodesic centroidal tesselation. In *3DPVT 2004*, 2004. 4, 9

[187] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Siggraph 2000, Computer Graphics Proceedings*, pages 335–342. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000. 3, 14, 15, 20, 73, 76

[188] PointShop3D Repository. http://graphics.ethz.ch/pointshop3d/download.html. 21, 26, 35, 37, 38, 46, 47, 50, 53, 55, 57, 58, 130, 133, 134, 135, 136, 138

[189] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics*, pages 135–150. Springer Verlag, Heidelberg, 1998. 15

[190] H. Pottmann and M. Hofer. A variational approach to spline curves on surfaces. *Comput. Aided Geom. Design*, 22(7):693–709, 2005. 24

[191] Emmanuel Prados, Fabio Camilli, and Olivier Faugeras. A unifying and rigorous shape from shading method adapted to realistic data and applications. *Journal of Mathematical Imaging and Vision*, 25(3):307–328, 2006. 39

[192] Emmanuel Prados, Stefano Soatto, Christophe Lenglet, Jean-Philippe Pons, Nicolas Wotawa, Rachid Deriche, and Olivier Faugeras. Control theory and fast marching techniques for brain connectivity mapping. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1076–1083, Washington, DC, USA, 2006. IEEE Computer Society. 4

[193] E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Trans. on Graphics*, 22:340–349, 2003. 42

[194] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing, 2nd edition*. Cambridge University Press, 1992. 25, 26

[195] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing, 2nd edition*. Cambridge University Press, February 2002. 43

[196] Prof. Dr. Ron Kimmel Home page, Computer Science Department, Technion - Israel Institute of Technology, Haifa, Israel. http://www.cs.technion.ac.il/ ron/. 114

[197] Kimmel R. and Sethian J. A. Computing geodesics on manifolds. In *Proc. of National Academy of Sciences*, volume 95, pages 8431–8435, 1998. 16, 22, 96, 98

[198] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Trans. on Graphics*, 2(2):91–108, July 1983. 3

[199] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space ewa splatting: A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum*, 21(3), September 2002. 3

[200] Patrick Reuter, Ireneusz Tobor, Christophe Schlick, and Sbastien Dedieu. Point-based modelling and rendering using radial basis functions. In *ACM Graphite 2003*, pages 111–118, 2003. 2, 15, 23

[201] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. *Modeling in Computer Graphics*, pages 455–465., 1993. 73

[202] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000. 63, 111

[203] Mauro R. Ruggeri, Tal Darom, Dietmar Saupe, and Nahum Kiryati. Approximating geodesics on point set surfaces. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 85–93, Boston, Massachusetts, USA, July 2006. 4

[204] Mauro R. Ruggeri, Dejan V. Vranić, and Dietmar Saupe. Shape similarity search for surfel-based models. In *International Conference on Computer Vision and Graphics 2004 (ICCVG04)*, Computer Vision and Graphics, pages 131–140, Warsaw, Poland, September 2004. Springer. 5, 69

[205] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 438–446, New York, NY, USA, 2002. ACM Press. 100

[206] Szymon Rusinkiewicz and Marc Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *ACM SIGGRAPH 2000*, pages 343–352, 2000. 73, 74, 77

[207] F. Sadlo, T. Weyrich, R. Peikert, and M. Gross. A practical structured light acquisition system for point-based geometry and texture. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 359–365, 2005. 39

[208] J. Salvi, J. Pagés, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, April 2004. 39

[209] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proc. EG Symp. on Geometry Processing*, pages 146–155, 2003. 4, 9

[210] Philip Schneider and David H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, September 2002. 10

[211] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *Proceedings of SIGGRAPH*, pages 161–172, 1995. 48, 49, 50

[212] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 65–70, New York, NY, USA, 1992. ACM Press. 79

[213] Justus Schwartz, Angelika Steger, and Andreas Weißl. Fast algorithms for weighted bipartite matching. In Sotiris E. Nikoletseas, editor, *Experimental and Efficient Algorithms, 4th International Workshop, WEA 2005*, volume 3503 of *Lecture Notes in Computer Science*, pages 476–487, Santorini Island, Greece, May 2005. Springer. 113

[214] Thomas B. Sebastian and Benjamin B. Kimia. Curves vs. skeletons in object recognition. *Signal Process.*, 85(2):247–263, 2005. 95

[215] James Albert Sethian. *Level set methods: Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science.* Number 3 in Cambridge monographs on applied and computational mathematics. Cambridge University Press, Cambridge, U.K., 1996. 218 pages. 16

[216] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, New York, NY, USA, 1998. ACM Press. 76

[217] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993. 50

[218] Gregory C. Sharp, Sang W. Lee, and David K. Wehe. Icp registration using invariant features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):90–102, 2002. 100

[219] P. Shilane, M. Kazhdan, P. Min, and T. Funkhouser. The princeton shape benchmark. In *SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04)*, pages 167–178, Washington, DC, USA, 2004. IEEE Computer Society. 60, 64, 70, 71, 72, 74, 75, 80, 81, 82, 83, 84, 89, 122, 123

[220] Yoshihisa Shinagawa, Tosiyasu L. Kunii, and Yannick L. Kergosien. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 1991. 93

[221] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. *Int. J. Comput. Vision*, 35(1):13–32, 1999. 95

[222] Oren Sifri, A. Sheffer, and C. Gotsman. Geodesic-based surface remeshing. In *Proc. of 12th International Meshing Roundtable*, pages 189–199, 2003. 4, 9, 16

[223] Hermilo Snchez-Cruz and Ernesto Bribiesca. A method of optimum transformation of 3d objects used as a measure of shape dissimilarity. *Image and Vision Computing*, 21(12):1027–1036, November 2003. 70

[224] P. Soille, J. Vogt, and R. Colombo. Carving and adaptive drainage enforcement of grid digital elevation models. *Water Resources Research*, 39:10–1, December 2003. 4

[225] Marc Stamminger and George Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 151–162, London, UK, 2001. Springer-Verlag. 78

[226] Gilbert Strang. *Introduction to Linear Algebra.* Wellesly-Cambridge Press, 3rd edition, 2003. 64

[227] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *SMI '03: Proceedings of the Shape Modeling International 2003*, pages 130–142. IEEE Computer Society, 2003. 64, 95

[228] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. on Graphics*, 24(3):553–560, 2005. 16, 22

[229] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998. 48, 50

[230] National Taiwan University: 3D Model Retrieval System. http://3d.csie.ntu.edu.tw/ dynamic. 60, 61, 75

[231] Kwok-Leung Tam, Rynson W.H. Lau, and Chong-Wah Ngo. Deformable object model matching by topological and geometric similarity. *Proceedings of ICPR 2004*, pages 910–913, August 2004. 95

[232] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04)*, pages 145–156, Washington, DC, USA, 2004. IEEE Computer Society. 62, 64, 69, 102

[233] Johan W. H. Tangelder and Remco C. Veltkamp. Polyhedral model retrieval using weighted point sets. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 119, Washington, DC, USA, 2003. IEEE Computer Society. 70, 73, 102

[234] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000. 64, 97, 100

[235] The QSplat Multiresolution Point Rendering System Repository. http://graphics.stanford.edu/software/qsplat/models/. 21, 37, 38

[236] The Stanford 3D Scanning Repository. http://www.graphics.stanford.edu/data/3Dscanrep/. 21

[237] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. Invariant High-Level Reeb Graphs of 3D Polygonal Meshes. In *3rd IEEE International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'06)*, Chapel Hill, North Carolina, USA, 2006. 63, 94

[238] Ireneusz Tobor, Christophe Schlick, and Laurent Grisoni. Rendering by surfels. In *Proceedings of the 10th International Conference on Computer Graphics and Vision GraphiCon*, August 2000. 77

[239] C. Touma and C. Gotsman. Triangle mesh compression. In *Proceedings Graphics Interface*, pages 26–34, 1998. 48

[240] Tony Tung and Francis Schmitt. Augmented reeb graphs for content-based retrieval of 3d mesh models. In *International Conference on Shape Modeling and Applications (SMI 2004)*, pages 157–166. IEEE Computer Society, 2004. 63, 94

[241] Greg Turk. Re-tiling polygonal surfaces. *SIGGRAPH Comput. Graph.*, 26(2):55–64, 1992. 77, 78, 79, 102

[242] Robert A. Ulichney. Dithering with blue noise. *Proceedings of IEEE*, 76(1):56–79, January 1988. 104

[243] N. Vajramushti, I. A. Kakadiaris, T. Theoharis, and G. Papaioannou. Efficient 3d object retrieval using depth images. In *MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 189–196, New York, NY, USA, 2004. ACM Press. 69

[244] V. A. Vassiliev. *Introduction to Topology*. American Mathematical Society, 2001. 125

[245] Remco C. Veltkamp and Michiel Hagedoorn. State of the art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999. 95

[246] D. V. Vranić and D. Saupe. 3d model retrieval. In B. Falcidieno, editor, *Proceedings of the Spring Conference on Computer Graphics and its Applications (SCCG2000)*, pages 89–93, Budmerice, Slovakia, May 2000. 60

[247] D. V. Vranić, D. Saupe, and J. Richter. Tools for 3d-object retrieval: Karhunen-loève transform and spherical harmonics. In J.-L. Dugelay and K. Rose, editors, *Proceedings of the IEEE 2001 Workshop Multimedia Signal Processing*, pages 293–298, Cannes, France, 2001. 69, 90

[248] Dejan V. Vranić. An improvement of rotation invariant 3d shape descriptor based on functions on concentric spheres. In *Proceedings of the IEEE International Conference on Image Processing*, pages 757–760, September 2003. 69

[249] Dejan V. Vranić. *3D model retrieval*. PhD thesis, University of Leipzig, 2004. 64, 67, 70, 83, 84, 87, 88, 89, 90, 123

[250] Michael Wand. *Point-Based Multi-Resolution Rendering*. PhD thesis, Department of computer science and cognitive science, University of Tübingen, 2004. 78, 102

[251] Michael Wand, Matthias Fischer, Ingmar Peter, Friedhelm Meyer auf der Heide, and Wolfgang Strasser. The randomized z-buffer algorithm: interactive rendering of highly complex scenes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 361–370, New York, NY, USA, 2001. ACM Press. 78, 102

[252] Y. Wang, B. S. Peterson, and L. H. Staib. Shape-based 3d surface correspondence using geodesics and local geometry. In *CVPR'00: Proceedings of the 2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 644–651, Hilton Head Island, SC, USA, June 2000. 4, 9

[253] Yongmei Wang, Bradley S. Peterson, and Lawrence H. Staib. 3d brain surface matching based on geodesics and local geometry. *Comput. Vis. Image Underst.*, 89(2-3):252–271, 2003. 4

[254] M. Waschbüsch, M. Gross, F. Eberhard, E. Lamboray, and S. Würmlin. Progressive compression of point-sampled models. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, pages 95–102, Zürich, Switzerland, June 2004. 40, 53

[255] Roger Weber, Hans-J. Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. 65

[256] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277, New York, NY, USA, 1994. ACM Press. 77

[257] Jianhua Wu and Leif Kobbelt. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum*, 23(3):643–652, 2004. 3, 15, 20

[258] Yijun Xiao, Naoufel Werghi, and Paul Siebert. A topological approach for segmenting human body shape. In *ICIAP '03: Proceedings of the 12th International Conference on Image Analysis and Processing*, page 82, Washington, DC, USA, 2003. IEEE Computer Society. 94

[259] T. Zaharia and F. J. Preteux. 3D-shape-based retrieval within the MPEG-7 framework. In Edward R. Dougherty and Jaakko T. Astola, editors, *Proc. SPIE Nonlinear Image Processing and Pattern Analysis XII*, volume 4304, pages 133–145, May 2001. 69, 71

[260] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An interface for sketching 3D scenes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, pages 163–170. Addison Wesley, 1996. 61

[261] S. Zelinka and M. Garland. Similarity-based surface modelling using geodesic fans. In *Proc. EG Symp. on Geometry Processing*, pages 209–218, 2004. 4, 9

[262] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 64, 65

[263] C. Zhang and T. Chen. Efficient feature extraction for 2d/3d objects in mesh representation. In *Proceedings of the IEEE International Conference on Image Processing*, pages 935–938, 2001. 69

[264] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999. 39

[265] G. Zigelman, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via via multi-dimensional scaling. *IEEE Trans. on Visualization and Computer Graphics*, 8:198–207, 2002. 4, 9, 42, 121

[266] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, 2005. 69

[267] M. Zwicker and C. Gotsman. Meshing point clouds using spherical parameterization. In *Proc. of EG/IEEE Symposium on Point-Based Graphics*, Zürich, Switzerland, June 2004. 42

[268] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d: an interactive system for point-based surface editing. *ACM Trans. on Graphics*, 21(3):322–329, 2002. 3, 29, 30, 53, 77, 92

[269] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 3

[270] Matthias Zwicker, Jussi Räsänen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. Perspective accurate splatting. In *GI '04: Proc. Graphics interface 2004*, pages 247–254, 2004. 3