



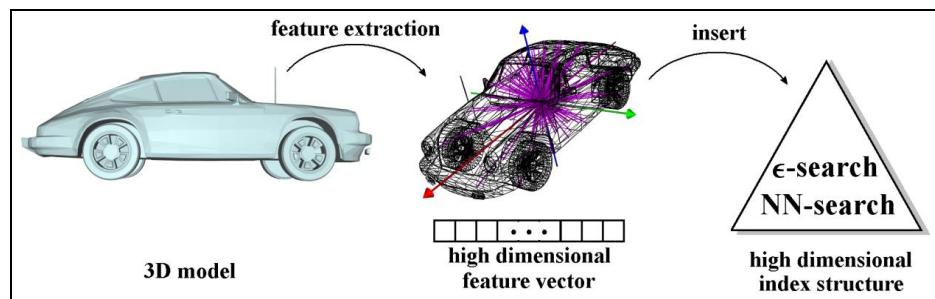
UNIVERSITÄT LEIPZIG
INSTITUT FÜR INFORMATIK



Dejan V. Vranić

3D Model Retrieval

- Ph. D. Dissertation -



Date of submission: December 22, 2003.

Date of defense: June 10, 2004.

Copyright © 2003, Dejan V. Vranić.

All rights reserved. No part of this book may be reproduced or transmitted in any other form or by any means, electronic or mechanical, including photocopy, recording or any information storage and retrieval system, without prior written permission of the author.

To my father, Vukola Vranić, in memoriam.

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbstständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

.....
(Ort, Datum)

.....
(Unterschrift)

Abstract

The topic of the thesis is content-based retrieval of 3D-models by shape-similarity. In our 3D model retrieval system a model, a polygonal mesh, serves as a query and similar objects are retrieved from a collection of 3D-objects. Algorithms proceed first by a normalization step in which models are transformed into a canonical coordinate frame. Second, feature vectors (descriptors) are extracted and compared with those derived from normalized models in the search space. Using a metric in the feature vector space nearest neighbors are computed and ranked. Objects thus retrieved are displayed for inspection, selection, and processing.

Objects represented as polygonal meshes are given in arbitrary orientation, scale, and position in the 3D-space. If the invariance of descriptor with respect to similarity transforms is not provided by the representation of a feature, pose estimation (normalization) is necessary as a step preceding the feature extraction. The pose normalization procedure is a transformation of a 3D-mesh model into a canonical coordinate frame by translating, rotating, scaling, and reflecting (flipping) the original set of vertices. We regard a triangle mesh model as a union of triangles, whence the point set of the model consists of infinitely many points. In contrast to pose normalization techniques based on sums over weighted vertices, we work with sums of integrals over triangles which makes our approach more complete taking into account all points of the model with equal weight.

The main objective of this thesis is construction, analysis, and testing of new techniques for describing 3D-shape of polygonal mesh models. Since a solid formal framework that could be used for defining optimal 3D-shape descriptors does not exist, we develop a variety of descriptors capturing different features of 3D-objects and using different representation methods. We consider a variety of features for characterizing 3D-shape such as extents of a model in certain directions, contours of 2D projections of a model, depth buffer images of a model, artificially defined volumes associated to triangles of a mesh, voxel grids attributed by fractions of the total surface area of a mesh, rendered perspective projections of a model on an enclosing sphere, and layered depth spheres. The used representation techniques include the 1D, 2D, and 3D discrete Fourier transforms, the Fourier transform on a sphere (spherical harmonics), and moments for representing the extent function. We also introduce two approaches for merging appropriate feature vectors, by defining a complex function on a sphere, and by crossbreeding (hybrid descriptors). We present a variety of original feature extraction algorithms and give complete specifications for forming feature vector components for each of presented approaches. A Web-based 3D model retrieval system is implemented and serves as a proof-of-concept.

We compare two techniques for achieving invariance of descriptors with respect to rotation of the polygonal mesh, the Principal Component Analysis (PCA) vs. a property of spherical harmonics. Several tests show that the first approach (PCA)

is better method for attaining rotation invariance of descriptors.

The retrieval performance of our feature vectors is carefully studied and compared to effectiveness of techniques proposed by other authors. We compare 12 different types of 3D-descriptors defined by ourselves to 7 types of descriptors defined by other authors using six ground truth classifications of 3D-models. The results unambiguously show that our best descriptor, a hybrid feature vector, outperforms the state-of-the-art.

Acknowledgments

First of all, I would like to thank to my advisor, Prof. Dr. Dietmar Saupe. This work was supported by an award from the Deutsche Forschungsgemeinschaft (DFG), grant GRK 446/1-98 for the Graduiertenkolleg Wissensrepräsentation (graduate study program on knowledge representation) at the University of Leipzig. I would also like to thank to Prof. Dr. Gerhard Brewka, the chair of the program. The Deutsche Forschungsgemeinschaft also supported my work through the Project SA 449/10-1, within the strategic research initiative "Distributed Processing and Delivery of Digital Documents" (V3D2), SPP 1041.

My experience at the Technical University of Vienna played a decisive role in selecting goals in my life. The person who directly changed my career was Dr. Gordana Popović and I wish to express my gratefulness to her.

During the past four years, I communicated with a lot of people and I want to thank to all of them.

Last but not the least, I thank to my wife Djilija, my daughter Aleksandra, and my mother Živka. My "three funny female persons" mean everything to me and they supported me the most.

Autumn 2003, Konstanz

Dejan V. Vranić

Contents

Introduction	1
1 Multimedia Retrieval	7
1.1 Content-Based Retrieval of Audiovisual Data	7
1.2 MPEG-7 Context	11
1.3 Shape-Similarity Retrieval of 3D Objects	14
1.3.1 Polygonal Mesh Models	14
1.3.2 3D Model Retrieval Algorithm	18
1.3.3 Types of 3D-Shape Features	20
1.3.4 3D-Shape Descriptors Criteria	22
1.4 Similarity Search Metrics	25
1.5 Tools for Evaluation of Retrieval Effectiveness	29
2 Related Research Work	35
2.1 Cords and Moments-Based Descriptors	36
2.2 Descriptor based on Equivalence Classes	39
2.3 Shape Spectrum Descriptor	42
2.4 Topology Matching	45
2.5 Shape Distributions, Reflective Symmetry, and Descriptors Based on Concentric Spheres	50
2.5.1 Shape Distributions	50
2.5.2 Descriptor Based on Binary Voxel Grids	53
2.5.3 Reflective Symmetry Descriptor	56
2.5.4 Descriptor Based on Exponentially Decaying EDT	58
3 Pose Estimation	61
3.1 Problem Description	61
3.2 Principal Component Analysis	63
3.3 Modifications of the PCA	68
3.4 “Continuous” PCA	69
3.5 Evaluation of the Continuous Approach	73

4 3D-Shape Feature Vectors	77
4.1 Ray-Based Feature Vector	78
4.2 Silhouette-Based Feature Vectors	85
4.3 Depth Buffer-Based Feature Vector	91
4.4 Volume-Based Feature Vector	98
4.5 Voxel-Based Approach	107
4.6 Describing 3D-Shape with Functions on a Sphere	118
4.6.1 Spherical Harmonics	119
4.6.2 Ray-Based Approach with Spherical Harmonic Representation	123
4.6.3 Moments-Based Feature Vector	126
4.6.4 Shading-Based Feature Vector	127
4.6.5 Complex Feature Vector	129
4.6.6 Feature Vectors Based on Layered Depth Spheres	131
4.7 Hybrid Descriptors	135
5 Experimental Results	139
5.1 3D Model Datasets	139
5.2 Comparison of 3D-Shape Feature Vectors	143
5.2.1 Ray-Based Feature Vector	145
5.2.2 Silhouette-Based Feature Vectors	148
5.2.3 Depth Buffer-Based Feature Vector	152
5.2.4 Volume-Based Feature Vector	156
5.2.5 Voxel-Based Approach	160
5.2.6 Ray-Based Approach with Spherical Harmonic Representation	165
5.2.7 Moments-Based Feature Vector	168
5.2.8 Shading-Based Feature Vector	170
5.2.9 Complex Feature Vector	172
5.2.10 Feature Vectors Based on Layered Depth Spheres	176
5.2.11 Hybrid Descriptors	182
5.2.12 Other Feature Vectors	185
5.2.13 Global Comparison	192
5.3 Dimension Reduction using the PCA	197
5.4 Summary of Experimental Results	199
6 Conclusion	201
Bibliography	205
Appendix: CCCC	219
Biography	227

Introduction

Since the ubiquitous Internet makes the world an increasingly networked place, effective access to information is becoming more and more important. An object from a database can traditionally be accessed using attached structural data. Other forms of data access include search in collections of textual documents and search in collections of audiovisual data (images, audio sequences, movies, and 3D-objects). Solutions for searching collections of textual documents are reasonably effective. Search in collections of multimedia objects can be performed by using textual annotations or by analyzing content of objects. Content-based search for multimedia objects is more challenging form of accessing audiovisual information. When we started our work in autumn 1999, solutions for content-based search were mostly aimed at retrieving still images, audio sequences, and movies, while only a few techniques for 3D model retrieval were reported. Our goal was to create a variety of 3D-shape descriptors in order to fill in the gap.

Appropriate 3D-shape features are automatically extracted and represented using suitable data structures (e.g., vectors, octrees, graphs). The resulting representation of a feature is regarded as a descriptor. Usually, features are represented by vectors with real-valued components, whence such descriptors are regarded as feature vectors. Descriptors should be defined in such a way that similar 3D-models are attributed feature vectors that are close in the search space. In our 3D model retrieval system a model, a polygonal mesh, serves as a query and similar objects are retrieved from a collection of 3D-objects. Algorithms proceed first by a normalization step (pose estimation) in which models are transformed into a canonical coordinate frame. Second, feature vectors are extracted and compared with those derived from normalized models in the search space. Using a metric in the feature vector space nearest neighbors are computed and ranked. Objects thus retrieved are displayed for inspection, selection, and processing. Shape-similarity retrieval of 3D-objects is not a “toy” problem, because designers of virtual worlds, creators of mechanical parts, scientist studying molecule docking, and authors of copyrighted models need to find 3D-models of particular shapes. Having in mind that recent development of 3D-scanners, visualization techniques, and CAD tools continuously increase the number of available 3D-models, there is a need for content-based 3D model retrieval systems.

The main objective of this thesis is construction, analysis, and testing of new techniques for describing 3D-shape of polygonal mesh models. Since there is no

theory that specifies how to analyze low-level features of polygonal meshes in order to describe the 3D-shape in the optimal way, we developed a variety of descriptors capturing different features of 3D-objects and using different representation methods. The best feature vectors possess high discriminant power. Besides defining 3D-shape descriptors, the retrieval performance of our feature vectors is carefully studied and compared to effectiveness of techniques proposed by other authors. A variety of tools are implemented for efficient experimental analysis and verification of results. A Web-based 3D model retrieval system serves as a proof-of-concept. The results unambiguously show that our best descriptors outperform the state-of-the-art.

This thesis is organized as follows:

- Introduction;
- Chapter 1: Multimedia Retrieval;
- Chapter 2: Related Research Work;
- Chapter 3: Pose Estimation;
- Chapter 4: 3D-Shape Feature Vectors;
- Chapter 5: Experimental Results;
- Chapter 6: Conclusion;
- Bibliography;
- Appendix: CCCC.

The **first chapter** has an introductory character and consists of five sections. The motivation and general concept of multimedia retrieval is presented in section 1.1. An overview of MPEG-7, a standard that provides tools and methods to describe the content of audiovisual data, is given in section 1.2. Section 1.3, which consists of four subsections, focuses on the topic of the thesis, content-based retrieval of 3D-models. Since polygonal mesh is the most common way of representing 3D-objects, the polygonal mesh representation is explained in subsection 1.3.1. A general 3D-model retrieval algorithm is presented in subsection 1.3.2. In contrast to image retrieval, where color and texture can be used for searching similar object, the main challenge in the area of 3D-model retrieval is to describe shape of a 3D-object. Types of features, which are considered for characterizing shape, are discussed in subsection 1.3.3. 3D-shape descriptors should fulfill certain criteria, which are defined and discussed in subsection 1.3.4. Methods for measuring similarity (or dissimilarity) between feature vectors are addressed in section 1.4. Tools that are used in order to evaluate retrieval performance of 3D-shape descriptors are described in section 1.5.

The **second chapter** is dedicated to 3D-shape descriptors proposed by other authors. Since more and more researchers are attracted by the topic, a variety of approaches for describing 3D-shape have recently been reported. Besides, there is a significant work in the area of Computer Vision, which can be regarded as a somewhat similar topic. However, we do not need to infer all the information about 3D-shape from one or more 2D-images, because we deal with objects represented

as polygonal meshes. The criteria for selecting the descriptors that are presented in this chapter are historical reasons and the impact on the area of retrieval of polygonal mesh models. A selection of 9 techniques proposed by five groups of authors is described in details. In section 2.1, cords and moments-based descriptors proposed by Paquet et al. are described. A descriptor based on equivalence classes proposed by Suzuki et al. is presented in section 2.2. Section 2.3 is dedicated to the MPEG-7 shape spectrum descriptor, which is a histogram of curvature indices. An interesting technique, called topology matching, which uses graphs as 3D-shape descriptors, is explained in section 2.4. Section 2.5 is subdivided into 4 parts describing techniques proposed by the Princeton Shape Analysis and Retrieval Group. A concept of shape distributions (subsection 2.5.1) is based on randomized computation of certain geometric properties. A descriptor based on binary voxel grids (subsection 2.5.2) uses an original representation technique that secures invariance of the descriptor with respect to rotations of a 3D-model. The reflective symmetry descriptor (subsection 2.5.3) relies upon the assumption that a measure of similarity between parts of a model laying on the opposite sides of a cutting plane can be used for capturing 3D-shape. A descriptor based on exponentially decaying Euclidean distance transform (subsection 2.5.4) uses almost identical representation as the descriptor based on binary voxel grids. However, the considered feature, a voxel grid attributed by exponentially decaying Euclidean distance transform (EDT), describes the 3D-shape in a more effective way. Based on our evaluation as well as on results presented in the literature, we regard the descriptor based on negatively exponentiated EDT [58] as the state-of-the-art-descriptor.

Objects represented as polygonal meshes are given in arbitrary orientation, scale, and position in the 3D-space \mathbb{R}^3 . 3D-shape descriptors can be defined in such a way that invariance with respect to translation, rotation, scaling, and reflection of a mesh model is provided. Examples of such descriptors are the shape spectrum descriptor (section 2.3), topology matching (section 2.4), and shape distributions (section 2.5.1). If the invariance of descriptor with respect to similarity transforms is not provided by the representation of a feature, pose estimation (normalization) is necessary as a step preceding the feature extraction. The pose normalization procedure is a transformation of a 3D-mesh model into a canonical coordinate frame by translating, rotating, scaling, and reflecting (flipping) the original set of vertices. In the **third chapter**, details about our original pose estimation approach are given. In section 3.1, we describe the problem of finding the canonical coordinates of a mesh. The most prominent tool for solving the problem is the Principal Component Analysis (PCA) [51], also known as the discrete Karhunen-Loeve transform, or the Hotelling transform, which is described in details in section 3.2. Since applying the PCA to the set of vertices of a mesh model can produce undesired normalization results, two modifications of the PCA are given in section 3.3. Both modifications approximate the application of the PCA to the point set of a 3D-object (union of polygons). In section 3.4, we present our original method for analytical computation of various parameters needed to analyze the set of infinitely many points. This computation enables application of the PCA to an infinite point set represented as a union of triangles. We called the approach the *Continuous Principal Component*

Analysis (CPCA). Examples as well as an evaluation of the continuous approach are given in section 3.5.

In the **fourth chapter**, we present our original methods for describing 3D-shape. Since the optimal way of encoding information about 3D-shape is not prescribed, we consider a variety of different features to define shape descriptors (feature vectors). The approaches include: ray-based feature vector in the spatial domain (section 4.1), silhouette-based descriptor (section 4.2), depth buffer-based feature vector (section 4.3), descriptor based on artificial volumes associated to triangles (section 4.4), and descriptor based on voxel grids attributed by fractions of the total surface of a polygonal mesh (section 4.5). Certain features aimed at describing 3D-shape of a polygonal mesh can be considered as samples of a function on a sphere (section 4.6). A suitable tool for representing samples of functions on a sphere is the fast Fourier transform on a sphere. In the frequency (spectral) domain, the samples are represented by spherical harmonic coefficients. In subsection 4.6.1, a brief presentation of spherical harmonics as well as our original approach for forming descriptors with spherical harmonic representation are given. As far as we know, spherical harmonics as a tool for 3D model retrieval are introduced by ourselves in [147]. A set of descriptors with spherical harmonic representation include: ray-based feature vector with spherical harmonic representation (section 4.6.2), descriptor based on rendered perspective projection on an enclosing sphere (section 4.6.4), complex feature vector (section 4.6.5), descriptor based on layered depth spheres (section 4.6.6), and rotation invariant descriptor based on layered depth spheres (section 4.6.6). We also defined a moments-based feature vector (section 4.6.3), to compare different representations of samples of functions on a sphere, spherical harmonics vs. moments. Finally, a concept of hybrid feature vectors is introduced in section 4.7. We follow a typical 3D-model algorithm (pose estimation → feature extraction → similarity search) and most of our feature vectors are extracted in the canonical coordinate frame of a 3D-model. Our techniques are not restricted to closed or orientable polygonal mesh models. In order to be able to verify the results of feature extraction procedure, we usually implement at least two extraction tools for each approach. The forming of feature vector components as well as specifications of feature extraction methods are described in details, in order to provide sufficient information to a reader who wants to implement and test our methods.

A set of 19 implemented types of feature vectors (12 ours and 7 proposed by other authors) is evaluated in the **fifth chapter**. Firstly, in section 5.1, we present available 3D-model collections and classifications, which are used as ground truth for experiments. We use two classifications of our 3D-model collection, which is mostly collected on the Internet (e.g., www.3dcafe.com), two classifications of the MPEG-7 set of 3D-models, and two 3D-model databases (training and test) provided by the Princeton Shape Analysis and Retrieval Group [108]. In section 5.2, we examine different variants and parameter settings of our descriptors, and test a variety of dissimilarity measures. Similar test are performed for descriptors proposed by other authors. After the best version, parameter settings, and dissimilarity measure are determined for each descriptor, a global comparison is performed. The results show

that our hybrid descriptor significantly outperform all other descriptors, including the state-of-the-art descriptor, which is extracted using original tools provided by the authors [108]. Experiments aimed at testing dimension reduction using the Principal Component Analysis (PCA) are presented in section 5.3. A summary of experimental results (section 5.4) concludes the chapter. Some authors object the use of the PCA for orienting a model in the pose estimation step. However, our results show that the best approach relies upon the PCA. Moreover, we modified certain techniques (including our implementation of the state-of-the-art) that avoid the use of the PCA by utilizing a property of spherical harmonics. The modification consists of applying the PCA instead of the property of spherical harmonics. Thus, we compared the competing techniques for achieving rotation invariance, the PCA vs. the property of spherical harmonics, on three types of feature vectors. Each time the result was the same, the descriptors relying upon the PCA outperformed descriptor relying upon the competing approach for attaining rotation invariance.

In the **sixth chapter**, we summarize the contribution, stress the most important results, and suggest directions for future work.

A list of directly used or cited works (**bibliography**), which contains 153 references, is located at the end of the thesis.

Our Web-based 3D model retrieval system, called CCCC, is presented in the **appendix**. Besides serving as a proof-of-concept, the CCCC 3D search engine is useful for obtaining an impression about effectiveness of different descriptors, by inspecting retrieved models as well as by using provided tools for comparing descriptors at three levels (models, classes, and the whole collection).

Our contributions to the area of 3D model retrieval are the following:

- Introduction of spherical harmonics as a tool for 3D model retrieval [147];
- Introduction of the concept of hybrid descriptors (original result in the thesis);
- Introduction of the Continuous Principal Component Analysis (CPCA) as a tool for 3D model retrieval [147];
- Introduction of the ray-based approach as a 3D-model retrieval technique [143, 141];
- Definition and evaluation of different variants of the silhouette-based feature vector (original result in the thesis). We proposed the first variant of the silhouette-based approach in [46, 47];
- Definition and evaluation of different variants of the depth buffer-based feature vector (original result in the thesis). We proposed the first variant of the depth buffer-based approach in [46, 47];
- Definition and evaluation of different variants of the volume-based feature vector (original result in the thesis). We proposed the first variant of the volume-based approach in [46, 47];
- Definition and evaluation of different variants of the voxel-based feature vector (original result in the thesis). We proposed the first variant of the voxel-based approach in [46, 47];

- Introduction of the discrete 3D Fourier transform as a tool for 3D model retrieval [144];
- Definition of the moments-based descriptor [117];
- Consideration of a rendered perspective projection as a feature aimed at describing 3D-shape of polygonal mesh models (shading-based descriptor) [146];
- Introduction of the concept of complex feature vectors [146];
- Introduction of layered depth spheres as a new data structure suitable for characterizing 3D-shape of polygonal meshes (original result in the thesis);
- Comparison of two approaches for achieving invariance with respect to rotations of a polygonal mesh model, the CPCCA vs. a property of spherical harmonics [142];
- Pointing out that 3D-shape descriptors should be robust with respect to outliers. As far as we know, the requirement is mentioned in [144] for the first time;
- Creation of a variety of original and very efficient algorithms for computing certain geometrical properties of polygonal meshes (original result in the thesis);
- Comparison of 19 different types of descriptors using 6 different ground truth classifications (original result in the thesis);
- Implementation of a Web-based retrieval system [140].

Chapter 1

Multimedia Retrieval

In this chapter, we first present the motivation and general concept of multimedia retrieval. Then, we give a brief overview of MPEG-7, a standard that provides tools and methods to describe the content of audiovisual data. Next, we focus on the topic of the thesis, shape-similarity search for 3D-objects. We describe the polygonal-mesh representation of 3D-models, present the algorithm that we use to search in 3D-shape collections, discuss types of features that are used for describing 3D-content, and set criteria for defining 3D-shape descriptors. We also address methods for measuring similarity (or dissimilarity) between feature vectors. Finally, we describe tools that are applied in order to evaluate retrieval performance of 3D-shape descriptors.

1.1 Content-Based Retrieval of Audiovisual Data

The “Space Odyssey”, “Star Trek”, and other science fiction movies that appeared more than three decades ago, besides interesting stories and action scenes with special effects, offered talking computers, which were able to accept a command by recognizing voice and to use an image-based query to retrieve similar images. Three decades ago it was just an interesting idea. Nowadays, the amount of unique information produced in the world is rapidly increasing. Recent studies (like [74]) suggest that this production exceeds 1 exabyte (i.e., 10^{18} bytes) of new information per year, which is roughly 250 megabytes for every human on earth. Magnetic storage is becoming the universal medium for information storage. At the same time, much data are available on-line to a broad range of users. In order to use a document or an audiovisual object, it has to be located first. Therefore, the actual need for efficient data-access has led to the development of different search tools. The role of multimedia is increasingly important in many real-world applications such as e-commerce, communication, education, biomedicine, digital library, and journalism. In this section, we present the general concept of multimedia retrieval and emphasize the importance of content-based search.

An object from a database can traditionally be accessed using attached structural data. Solutions for searching collections of textual documents are very efficient and effective (e.g., *Google* or *AltaVista*). Collections of multimedia objects are mostly searched by using textual annotations, which can be effective when consistency of annotation exists. Since textual annotations are usually manually generated, they are subject to a person who creates them. Thus, the question of consistency arises, when annotations are created by different persons. Besides, only high-level information of audiovisual data can be annotated manually. For instance, an image of a landscape can be described by several words, e.g., “landscape”, “mountains”, “valley”, “river”, etc., while the information about color structure and shapes of objects on the image cannot be described by text, in general. Other drawbacks of retrieval techniques that use manual annotations are time-consuming creation and the fact that not all multimedia data contain attached textual information. The other approach is to analyze appropriate low-level features of a multimedia object (an image, movie, audio sequence, or 3D-model) and describe the content automatically, without interaction of human being. Features that are analyzed depend on a type of the object. In this case, generated descriptions are used for *content-based retrieval* of database objects.

As a contrast to text-based searching techniques in which a string of words serves as a query, content-based retrieval uses a multimedia object as a search key (query). Examples of content based search for multimedia data are:

- *Audio*: humming or whistling can be recorded and used for retrieving similar melodies;
- *Images*: graphs and logos can be retrieved by using a sketch of few lines as a query;
- *Video*: a video clip of goals scored in a football match serves as a key for retrieving video clips of other football matches;
- *3D-model*: one has a 3D-model and wants to retrieve other models similar by shape.

The concept of multimedia retrieval is depicted in figure 1.1. A typical retrieval algorithm possesses three modules: feature extraction, description generation, and search engine.

Feature extraction. The features capture important properties of specific multimedia data. Abstract (high-level) descriptions of an audiovisual object can be either created using certain interactive annotation tools or found as textual data contained in the object itself. Automatic feature extraction from sounds, images, videos, or 3D-models is a more attractive and challenging problem than creation of textual annotations. Consequently, a significant number of techniques for describing multimedia content have been reported. First content-based multimedia retrieval systems were dedicated to images.

A variety of methods for describing content of an *image* can be found in [138, 4, 29, 81, 76, 79, 109, 133, 134, 90, 112, 16, 119, 20, 73, 62, 88, 26, 121, 136, 90, 17]. Considered image features are color histogram, color structure (layout), color moments, color sets, texture homogeneity, texture coarseness, texture regularity, texture con-



Figure 1.1: Concept of multimedia retrieval.

trast, region-shape, contour-shape, edge distribution, etc. Besides general feature such as color, texture, and, shape, domain specific features, e.g., face and finger-print recognition, are considered, as well. Tools that are used to represent features include discrete cosine transform, Fourier analysis, principal component analysis, and wavelets. For shape-based features, automatic segmentation of an image is very important. Image segmentation techniques are based on expectation maximization, Delaunay triangulation, fuzzy entropy, fractals, edge flow, etc.

Features of an *audio* sequence are also studied in depth. Acoustical characteristics, e.g, loudness, pitch, bandwidth, harmonicity, are the most commonly used features for searching audio databases. Some techniques also include periodicity and concentration of energy in a certain area of a sequence. A different kind of problem is retrieval of spoken content or music notes (tone interval), where we encounter

the problem of recognizing words and notes. A general problem in audio analysis is to simply discriminate speech from non-vocal music, silence, or other sounds. Automatic speech recognition deals with keyword spotting, sub-word indexing, and speaker identification. Tools for describing high-level features, timbre and melody, are developed, as well. For more details about audio features and methods, which are used for content-based retrieval, we refer to [82, 32, 149, 70, 71, 31, 66].

A *video* retrieval system can be created by combining audio and image retrieval techniques. Also, there are features specific to videos such as spatial and temporal characteristics. Purely spatial features are color space, luminance, shape, size, texture, and orientation, while object motion and camera operation belong to temporal features. A spatio-temporal feature is, e.g., motion trajectory. Segmentation of videos as well as detection of shots and key frames are important steps of feature extraction procedure. High-level video features contain motion of objects, motion activity (slow or fast), recognition of an important person, recognition of an event, etc. Video retrieval literature is also very rich, e.g., [94, 114, 111, 123, 135, 50, 137].

Several techniques for retrieval of *3D-mesh* models have recently been proposed. All reported features capture the 3D-shape of models (objects). The area of shape similarity search for 3D objects is addressed in section 1.3, while the related work is reviewed in chapter 2. Our original contribution to the topic is presented in chapter 4.

Description generation. According to [85], we refer to a representation of a feature as a *descriptor*. A structure containing an identifier of the object (e.g., a name in local database or a URL) and at least one descriptor is called a *description*. As an example, for an image, we can generate a description consisting of image name, a few keywords (annotations), and representations of color histograms, texture contrast, and contour shape. Content-based retrieval systems usually store descriptions in an internal format, whence other systems cannot use the descriptions without proper transcoding tools and/or specification about the internal description format. One of the objectives of the MPEG-7 standard is to standardize content-based description for various types of audiovisual information. More details about MPEG-7 are given in section 1.2.

Search engine. A search engine for certain type of audiovisual data is an application that accepts a specific input as query (e.g., text or multimedia content), and retrieves objects ranked by the degree of similarity to the query. Descriptions of two “similar” audiovisual object contain descriptors, which should be attributed values that are “close” in a space of descriptors \mathbb{D} . The degree of similarity (or dissimilarity) between two descriptors is computed using a suitable measure $d : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{R}$. Suppose $Q \in \mathbb{D}$ is a descriptor of a query object of certain type (e.g., an image) and $\{D_1, \dots, D_N\} \subset \mathbb{D}$ is a collection of descriptors associated to objects of the same type as the query. For simplicity, the objects in the collection are enumerated (identified) by a set of indices $\{1, \dots, N\}$. If d is a measure of dissimilarity, the object with descriptor D_{m_1} is the *best match* (nearest neighbor) to the query, where

$$d(Q, D_{m_1}) = \min_{1 \leq i \leq N} d(Q, D_i), \quad m_1 \in \{1, \dots, N\}.$$

Finding the best match to a query is a typical pattern recognition problem. In information retrieval, we retrieve $K \leq N$ ranked objects for inspection, selection, and processing. The retrieved objects (matches), which are the most similar to the query object, are ranked so that their descriptors D_{m_1}, \dots, D_{m_K} ($m_1, \dots, m_K \in \{1, \dots, N\}$, $i \neq j \Rightarrow m_i \neq m_j$) satisfy

$$d(Q, D_{m_i}) \leq d(Q, D_{m_{i+1}}), \quad 1 \leq i \leq K-1,$$

where D_{m_i} is a descriptor of the object (match) with identifier m_i , which is regarded as the i -th match. The number of retrieved models, K , depends on the application. A user can specify how many models should be retrieved or a threshold value t can be set to retrieve all objects whose descriptors satisfy $d(Q, D_{m_i}) \leq t$.

The metric d depends on the type of descriptor. If we use textual annotations as descriptors, then $d(Q, D_i)$ can be, e.g., the total number of words that are contained in both descriptors Q and D_i . For multimedia retrieval applications, it is more common that descriptors are stored as vectors with real-valued components and fixed dimensions. Various choices of metric d are addressed in section 1.4.

If the size N of the collection of multimedia objects is relatively small (e.g., $N < 10000$) and the computation of distance d is fast enough (e.g., less than $200\mu s$), then the search can be done sequentially, i.e., we calculate all distances $d(Q, D_i)$ ($1 \leq i \leq N$) and sort them. However, if the order of magnitude of N is higher (e.g., 10^6), then it is necessary to include techniques for accelerating the search [131, 130].

1.2 MPEG-7 Context

The MPEG-7 standard [86, 85], also known as “Multimedia Content Description Interface”, provides a standardized set of tools for describing multimedia content. The standard addresses a broad range of multimedia applications and requirements, and provides a flexible and extensible framework for describing audiovisual data. The goal of the standard is to enable fast and efficient content searching, filtering and identification of various types of audiovisual information, such as music, speech, moving video, still pictures, graphics, and 3D models. Also, information on how objects are combined in scenes is inside the scope of MPEG-7. The MPEG-7 describes several aspects of the content, e.g., low-level features, structure, semantic, models, collections, creation, etc. Moreover, descriptions are independent of the data support, i.e., MPEG-7 descriptions can reference and describe multimedia documents in analogue format (e.g., temporal references to sequences recorded on a VHS tape).

The main elements of the standard are [86, 85, 82, 88, 83, 89, 84]:

- Description tools: descriptors and description schemes;
- Description definition language;
- System tools.

A *descriptor* defines syntax and semantics of the feature representation. A *description scheme* specifies the structure and semantics of the relationships between its

components, which may be both descriptors and description schemes. *Description definition language* [83] is a language to specify descriptors and description schemes, and to allow the extension and modification of existing description schemes. *System tools* are used for encoding descriptions in order to fulfill requirements such as transport and storage efficiency, error resilience, random access, and synchronization between content and descriptions.

The description tools are presented on the basis of the functionality they provide. In practice, they are combined into meaningful sets of description units. An appropriate subset of descriptors and description schemes needs to be selected for each specific application. Description definition language can be used to handle specific needs of the application.

The MPEG-7 does not standardize neither the feature extraction step (analysis) nor the search engine (application), which are parts of the multimedia retrieval chain depicted in figure 1.1. Both areas are left to the creativity and innovation of researchers. Only description generation is standardized, in order to enable interoperability.

The MPEG-7 standard is subdivided into the following parts:

1. *Systems*: tools to encode descriptions;
2. *Description Definition Language*: a language for specifying the standard set of description tools and for defining new description tools;
3. *Visual*: description tools aimed at characterizing visual content (images, movies, and 3D-models);
4. *Audio*: description tools dedicated to audio content;
5. *Multimedia Description Schemes*: generic description tools related to both visual and audio content;
6. *Reference Software*: a software implementation of the standard;
7. *Conformance*: guidelines and procedures for testing conformance of implementations of the standard.
8. *Extraction and Use*: guidelines and examples of the extraction and use of descriptions.

Application domains of the MPEG-7 include: storage and retrieval of audiovisual databases (e.g., image, movie, audio, and 3D-model collections), multimedia directory services (e.g., yellow pages), journalism (e.g., searching speeches, paper clips), tourist information and cultural services, entertainment (e.g., searching a game, karaoke), investigation services (e.g., fingerprint and face recognition), surveillance (e.g., traffic control), e-commerce and tele-shopping (e.g., finding clothes that you like), social services (e.g., dating), etc.

The MPEG-7 has set a wide spectrum of evaluation criteria for description tools and description definition language. For complete specifications we refer to [85]. Requirements for descriptors include:

- *Cross-modality* – to allow queries based on visual descriptions to retrieve audio data and vice versa;
- *Language of text-based descriptions* – to support all natural languages;
- *Linking* – to locate source data in space and in time;
- *Unique identification* – to support a mechanism for uniquely identifying data.
- *Types of features* – to support multimedia descriptions using various types of features;
- *Abstraction levels for multimedia material* – to support hierarchical mechanisms for describing multimedia objects at different levels of abstraction;
- *Application domain* – to cover a broad range of real-world applications;
- *Retrieval effectiveness* – to characterize an important property of an audiovisual object;
- *Scalability* – the size of descriptor is determined only by its definition and not by the size of multimedia content;

To fully exploit the possibilities of MPEG-7 descriptions, automatic extraction of features is very desirable. However, interactive extraction tools can also be of use, in particular when information cannot be deduced from the content, e.g., recording date and conditions, title, author, copyright, coding format, classification, parental rating, links to other relevant material, etc. The higher the level of abstraction, the more difficult automatic extraction is. Information that is present in the content can be categorized in low-level features (e.g., for an image, color, texture, size, position, etc.) and high-level (semantic) features, which are related to a human interpretation of the content and are attached as metadata to the content (e.g., “this is a scene with a guy talking to a girl at a sidewalk and a tram passing by on Augustusplatz in Leipzig”).

The set of visual descriptors consists of 25 techniques (classified in 7 categories) for characterizing various properties of images, movies, and 3D-models. The categorization of visual descriptors, which are part of the standard, is the following:

- *Basic structures*: grid layout, time series (regular and irregular), 2D/3D multiple view, spatial 2D coordinates, and temporal interpolation;
- *Color*: color space, color quantization, dominant color, scalable color, color layout, color structure, and group of frames / group of pixels color;
- *Texture*: homogeneous texture, texture browsing, and edge components histogram
- *Shape*: region shape, contour shape, and shape 3D.
- *Motion*: camera motion, motion trajectory, parametric motion, and motion activity;
- *Localization*: region locator and spatio-temporal locator;
- *Others*: face recognition.

Descriptors that can be engaged for shape similarity 3D-model retrieval are 2D/3D multiple view and shape 3D. The latter technique is described in section 2.3. Descriptor *2D-3D multiple view* analyzes 2D projections of a 3D object seen

from several viewpoints. Each projection is rasterized to an image, which can be described using any 2D visual descriptor, e.g, contour-shape, region-shape, color, or texture. The descriptor allows matching of 3D objects by comparing their views, as well as by comparing pure 2D views (images) to views of 3D objects. The multiple view descriptor can also be associated to a video segment [88, 89].

All description tools are implemented in the part 6 of the standard, reference software, and the latest version of this implementation can be obtained as described in [84]. The reference software, known as the *eXperimentation Model* (XM), creates standard audio descriptions, visual descriptions, and multimedia description schemes. The software also includes a parser and validator for the description definition language. The reference software is normative in the sense that any conforming implementation of the software, taking the same conformant bit streams and using the same output file format, should output the same file. Complying MPEG-7 implementations are not expected to follow the algorithms or the programming techniques used by the reference software. The decoding software is considered normative, whilst the techniques used for extracting descriptors are informative, and the quality and complexity of these extraction tools have not been optimized. The experimentation model serves as a proof-of-concept for description techniques included in the MPEG-7 standard.

An account of a multi-object multi-feature content-based search using MPEG-7 is given in [122], while 3D model retrieval in the context of MPEG-7 is addressed in [102, 152, 145].

1.3 Shape-Similarity Retrieval of 3D Objects

In this section, we address content-based retrieval of 3D-models. Since polygonal mesh is the most common way of representing 3D-objects, we explain the representation in subsection 1.3.1. A general 3D-model retrieval algorithm is presented in subsection 1.3.2. In contrast to image retrieval, where color and texture can be used for searching similar object, the main challenge in the area of 3D-model retrieval is to describe shape of a 3D-object. Types of features, which are considered for characterizing shape, are discussed in subsection 1.3.3. 3D-shape descriptors should fulfill certain criteria, which are defined in subsection 1.3.4.

1.3.1 Polygonal Mesh Models

Polygonal meshes are used for representing boundary surfaces (both inner and outer) of an arbitrary 3D-object. Boundary surface is approximated by a union of 3D polygons, whose vertices are coplanar. If all polygons are triangles, then we have a *triangle mesh*. Because of the simplicity as well as rendering purposes, polygonal mesh models are frequently converted into triangle meshes. For example, a polygon

Ω with k vertices $\mathbf{v}_1, \dots, \mathbf{v}_k$, can be treated as union of $k - 2$ triangles,

$$\Omega = \bigcup_{i=1}^{k-2} \Delta \mathbf{v}_1 \mathbf{v}_{k+1} \mathbf{v}_{k+2}. \quad (1.1)$$

Since certain feature extraction algorithms deal with triangles and not with polygons (e.g., ray-triangle intersection), we iterate through all polygons splitting them into triangles as necessary.

We regard a given triangle mesh as consisting of a set of triangles

$$T = \{T_1, \dots, T_m\}, \quad T_i \subset \mathbb{R}^3, \quad (1.2)$$

given by a set of vertices (geometry)

$$P = \{\mathbf{p}_i \mid \mathbf{p}_i = (x_i, y_i, z_i) \in \mathbb{R}^3, 1 \leq i \leq n\}, \quad (1.3)$$

and a list of indices of three vertices for each triangle (topology)

A_1	B_1	C_1	
	\vdots		
A_m	B_m	C_m	

(1.4)

where $A_i, B_i, C_i \in \{1, \dots, n\}$, $1 \leq i \leq m$. The vertices of the triangle T_i are denoted by \mathbf{p}_{A_i} , \mathbf{p}_{B_i} , and \mathbf{p}_{C_i} . We consider each triangle to be a set of infinitely many points, i.e.,

$$T_i = \{ \mathbf{v} \mid \mathbf{v} = \alpha \mathbf{p}_{A_i} + \beta \mathbf{p}_{B_i} + (1 - \alpha - \beta) \mathbf{p}_{C_i}, \alpha, \beta \in \mathbb{R}, \alpha, \beta \geq 0, \alpha + \beta \leq 1 \}$$

Then,

$$I = \bigcup_{i=1}^m T_i = \bigcup_{i=1}^m \Delta \mathbf{p}_{A_i} \mathbf{p}_{B_i} \mathbf{p}_{C_i} \quad (1.5)$$

is the point set of all triangles, i.e., our given object.

As an example of representing 3D-models by triangle meshes, we give lists of geometry and topology as well as a visualization of an icosahedron ($n = 12$, $m = 20$) in figure 1.2.

In what follows, we denote the surface area of triangle T_i by S_i , while the total area of the mesh is denoted by S ,

$$S_i = \frac{1}{2} |(\mathbf{p}_{C_i} - \mathbf{p}_{A_i}) \times (\mathbf{p}_{B_i} - \mathbf{p}_{A_i})|, \quad S = \sum_{i=1}^m S_i. \quad (1.6)$$

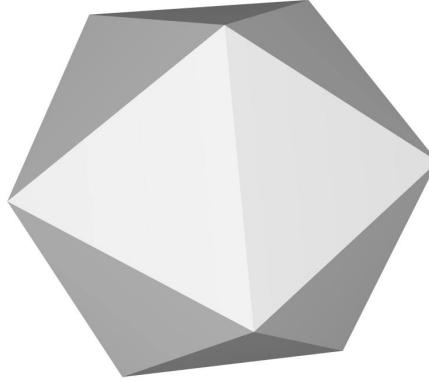
A polygonal mesh model is analogously defined. Instead of triangles (1.2), we have polygons,

$$\Omega = \{\Omega_1, \dots, \Omega_m\}, \quad \Omega_i \subset \mathbb{R}^3.$$

\mathbf{p}_1	1	0	0
\mathbf{p}_2	a	b	0
\mathbf{p}_3	a	c	d
\mathbf{p}_4	a	$-e$	f
\mathbf{p}_5	a	$-e$	$-f$
\mathbf{p}_6	a	c	$-d$
\mathbf{p}_7	-1	0	0
\mathbf{p}_8	$-a$	$-b$	0
\mathbf{p}_9	$-a$	$-c$	$-d$
\mathbf{p}_{10}	$-a$	e	$-f$
\mathbf{p}_{11}	$-a$	e	f
\mathbf{p}_{12}	$-a$	$-c$	d

$a = 1/\sqrt{5}$
 $b = 2/\sqrt{5}$
 $c = (\sqrt{5} - 1)/(2\sqrt{5})$
 $d = \sqrt{5 + \sqrt{5}}/\sqrt{10}$
 $e = \sqrt{3 + \sqrt{5}}/\sqrt{10}$
 $f = \sqrt{5 - \sqrt{5}}/\sqrt{10}$

T_1	1	2	3
T_2	1	3	4
T_3	1	4	5
T_4	1	5	6
T_5	1	6	2
T_6	2	6	10
T_7	3	2	11
T_8	4	3	12
T_9	5	4	8
T_{10}	6	5	9
T_{11}	7	9	8
T_{12}	7	10	9
T_{13}	7	11	10
T_{14}	7	12	11
T_{15}	7	8	12
T_{16}	8	4	12
T_{17}	9	5	8
T_{18}	10	6	9
T_{19}	11	2	10
T_{20}	12	3	11



A visualization of the icosahedron defined by the lists of geometry and topology (left).

Figure 1.2: An icosahedron represented as a triangle mesh model.

The list of indices (1.4) contains an additional field for specifying the number of vertices for each polygon.

Besides geometry and topology, a mesh model may contain additional information about a 3D-object, e.g., list of normals, list of colors, textures, etc. However, the additional information is mostly used for rendering, while geometry and topology are sufficient to represent the 3D-shape.

Let E be the list of edges of all triangles T_1, \dots, T_m (1.2), where the edges of triangle T_i are $\overline{\mathbf{p}_{A_i}\mathbf{p}_{B_i}}$, $\overline{\mathbf{p}_{B_i}\mathbf{p}_{C_i}}$, and $\overline{\mathbf{p}_{C_i}\mathbf{p}_{A_i}}$,

$$E = \{\overline{\mathbf{p}_{A_1}\mathbf{p}_{B_1}}, \overline{\mathbf{p}_{B_1}\mathbf{p}_{C_1}}, \overline{\mathbf{p}_{C_1}\mathbf{p}_{A_1}}, \dots, \overline{\mathbf{p}_{A_m}\mathbf{p}_{B_m}}, \overline{\mathbf{p}_{B_m}\mathbf{p}_{C_m}}, \overline{\mathbf{p}_{C_m}\mathbf{p}_{A_m}}\} \quad (1.7)$$

For certain applications, it is necessary to determine whether a mesh model is closed and orientable, or not. A mesh containing a “hole” is, obviously, not closed (figure 1.3). Closedness of a mesh can formally be defined as follows.

Definition 1.1 *A triangle mesh is closed if for any pair of vertices \mathbf{p}_i and \mathbf{p}_j ($i, j \in \{1, n\}$), the total number of occurrences of edges $\overline{\mathbf{p}_i\mathbf{p}_j}$ and $\overline{\mathbf{p}_j\mathbf{p}_i}$ in E (1.7) is either even or zero.*

The orientation of a triangle is determined by the sequence of its vertices. Let \mathbf{p}_a , \mathbf{p}_b , \mathbf{p}_c be the vertices of a triangle. Although triangles $T_i \equiv \Delta \mathbf{p}_a \mathbf{p}_c \mathbf{p}_b$ and $T'_i \equiv \Delta \mathbf{p}_a \mathbf{p}_b \mathbf{p}_c$ coincide in the 3D-space \mathbb{R}^3 , they have different orientations (figure 1.3). If $T_j \equiv \Delta \mathbf{p}_d \mathbf{p}_c \mathbf{p}_b$, then a mesh containing T_i and T_j is not orientable, because we have two occurrences of the edge $\overline{\mathbf{p}_c\mathbf{p}_b}$ and no occurrence of the edge $\overline{\mathbf{p}_b\mathbf{p}_c}$. If

T'_i is combined with T_j , then the orientability exists, because the edges $\overline{\mathbf{p}_c\mathbf{p}_b}$ and $\overline{\mathbf{p}_b\mathbf{p}_c}$ occur once both.

Definition 1.2 A triangle mesh is orientable if for any pair of vertices \mathbf{p}_i and \mathbf{p}_j ($i, j \in \{1, n\}$), the number of occurrences of the edge $\overline{\mathbf{p}_i\mathbf{p}_j}$ in E (1.7) is equal to the number of occurrences of the edge $\overline{\mathbf{p}_j\mathbf{p}_i}$ in E .

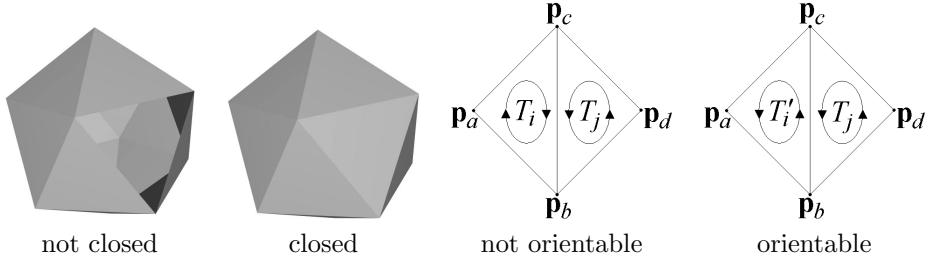


Figure 1.3: Closedness and orientability of triangle mesh models.

Ideally, a triangle mesh should be closed and orientable. However, 3D-models available on the Internet (e.g., www.3dcafe.com) are not necessarily represented by closed and/or orientable meshes. Moreover, there are some other irregularities of the representation (1.3, 1.4) such as

- *Multiple vertices*, ($\exists i, j \in \{1, \dots, n\}$) $\mathbf{p}_i \equiv \mathbf{p}_j \wedge i \neq j$;
- *Floating (non-connected) vertices*, ($\exists i \in \{1, \dots, n\}$) ($\forall j \in \{1, \dots, m\}$) $i \neq A_j \wedge i \neq B_j \wedge i \neq C_j$;
- *Multiple triangles*, ($\exists i, j \in \{1, \dots, m\}$) $A_i = A_j \wedge B_i = B_j \wedge C_i = C_j$;
- *Degenerated triangles*, ($\exists i \in \{1, \dots, m\}$) ($\exists \alpha \in \mathbb{R}$) $\mathbf{p}_{A_i} = \alpha \mathbf{p}_{B_i} + (1 - \alpha) \mathbf{p}_{C_i}$, i.e., \mathbf{p}_{A_i} , \mathbf{p}_{B_i} , and \mathbf{p}_{C_i} are collinear;

Outlying floating vertices may affect dimensions of bounding boxes or spheres. Multiple vertices and triangles have influence on determination of closedness and orientability. In special cases, degenerated triangles are inserted in order to provide orientability of the mesh. Relatively simple algorithms can be used to filter floating and multiple vertices, and to rearrange the topology (1.4). However, filling holes to close a mesh by inserting triangles, or inserting and re-orienting triangles to obtain an orientable mesh, are difficult problems, in general [8, 23, 67]. In our research, we filter multiple vertices and triangles, floating vertices, and degenerated triangles. We neither fill holes nor re-orient triangles.

Mesh models available on the Internet can be found in various 3D file formats. The most popular formats are: VRML (Virtual Reality Modeling Language) [44, 14], DXF (Autodesk Drawing eXchange Format) [5], 3DS (3D Studio file format) [96], OFF (Object File format) [96], OBJ (Wavefront Object files) [96], SMF (Simple Model Format) [37], PLY (Polygon File Format) [132], etc.

1.3.2 3D Model Retrieval Algorithm

A general concept of 3D-model retrieval is summarized in figure 1.4. Each 3D-model is attributed a descriptor (feature vector), which is represented in an appropriate way (e.g., an array), and is used for retrieving K nearest neighbors or all 3D-objects whose descriptors are within a given range (ϵ -search) from the descriptor of a query. Retrieval of 3D-models fits into the multimedia retrieval chain (compare figures 1.4 and 1.1). A 3D-model, in an appropriate 3D file format, serves as a query (search key). Content descriptors are automatically extracted, and used as points in a search space.

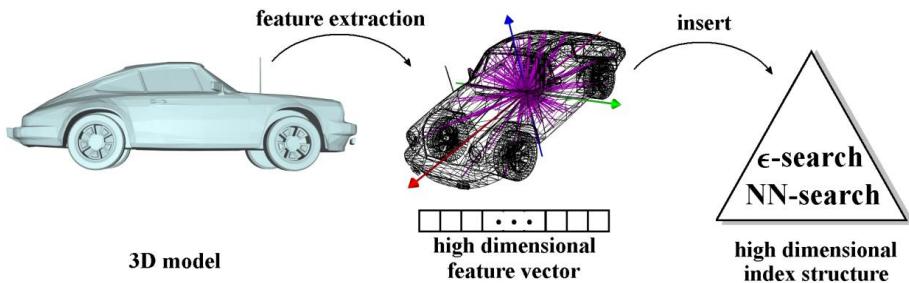


Figure 1.4: Concept of 3D-model retrieval.

The feature extraction step of a 3D-model retrieval algorithm consists of several sub-steps. Firstly, a file storing a 3D-model needs to be *parsed*. As a result, the lists of geometry (1.3) and topology (1.4) are populated. As mentioned in the previous subsection, a mesh model may contain multiple vertices and triangles, floating vertices, and degenerated triangles, whence a *filtering* step is desirable. After the filtering, the 3D-mesh can be processed further.

3D models are given in arbitrary units of measurement and in unpredictable positions and orientations in 3D-space. In certain cases, it is necessary to transform the model into a canonical coordinate frame. We regard to this step as the *pose normalization* (estimation) step. The goal of this procedure is that if one chose a different scale, position, rotation, or orientation of an original model, then the representation in the canonical coordinate frame would still be the same. Moreover, since objects may have different levels-of-detail (e.g., after a mesh simplification to reduce the number of polygons), their normalized representations should be the same as much as possible. The normalization step ensures that models can be retrieved regardless of the choices their authors have made for their mesh representation.

Next, we proceed with the *feature extraction* step. The features capture the 3D shape of the objects. Proposed features range from simple bounding box parameters [105] to complex image-based representations [46, 47]. Usually, the features are stored as vectors with real-valued components and fixed dimension. There is a trade-off between the required storage, computational complexity, and the resulting retrieval performance. The outcome of the extraction procedure are descriptors.

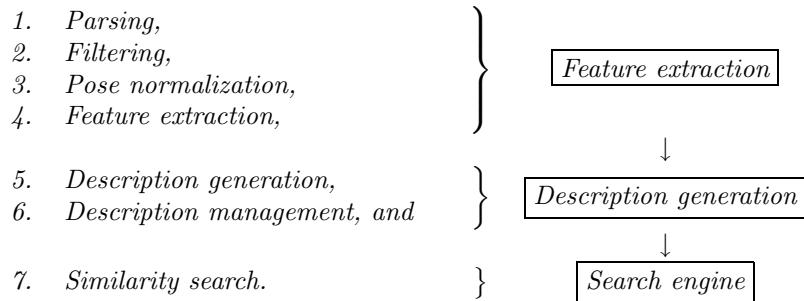
Since features are stored as vectors, and descriptors are regarded as representations of features, we use terms descriptors and feature vectors interchangeably throughout this thesis. A variety of 3D-model feature vectors are presented in chapters 2 and 4.

Extracted descriptors are used in the *description generation* step. Ideally, descriptions should be generated using some standard, e.g., MPEG-7 (section 1.2), in order to provide interoperability between applications. For instance, a search engine might rely upon standardized descriptions only, without engaging feature extraction tools. We use a simple internal format for generating descriptions. A description consists of a model identifier (a string of constant length) and an array of N floating point numbers, where N is the dimension of the feature vector. A module for *description management* organizes and stores descriptions. In our retrieval system, for a given feature type and parameter settings, descriptions of the whole 3D-model collection are stored in a single file. This file starts with a header, containing information about the feature type and parameter settings, followed by descriptions for each model. The descriptions can easily be represented using MPEG-7 description definition language [83], and encoded using standardized MPEG-7 tools.

Finally, an interactive application performs *similarity search* for 3D-objects by processing the stored descriptions. The features are designed so that similar 3D-objects are attributed vectors that are close in feature vector space. Using a suitable metric (see section 1.4) nearest neighbors are computed and ranked. A variable number of objects are thus retrieved by listing the top ranking items.

All the steps, which are present in a typical 3D model retrieval application, are summarized in algorithm 1.1 (compare to figure 1.1).

Algorithm 1.1 *A typical 3D model retrieval algorithm proceeds in the following steps:*



A block-diagram of our 3D-model retrieval system is given in figure 1.5. The system serves as an intermediary between available collections of 3D-objects and users who want to find specific shapes. All available models are processed by applying steps 1-6 of algorithm 1.1. The search engine with a Web-based interface can be used for browsing as well as retrieval. A user can upload a query model, whose description is automatically generated (steps 1-6 of algorithm 1.1), and used for matching procedure. The browsing enables a user to identify a model for querying,

which is reasonably similar to a desired one. Our Web-based retrieval system, called *CCCC* [140], is presented in appendix.

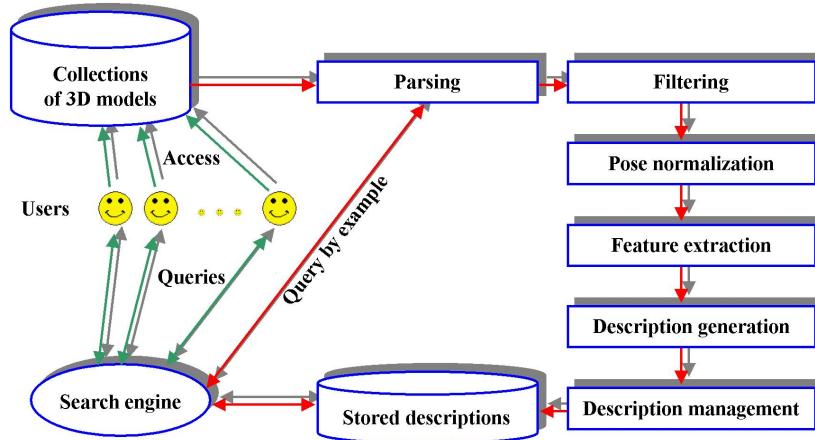


Figure 1.5: The architecture of our 3D model retrieval system.

1.3.3 Types of 3D-Shape Features

As mentioned in section 1.3, in retrieval applications, features of 3D-mesh models are usually analyzed in order to characterize *shape*. Information about normal vectors, colors of triangles or vertices, as well as texture, may also be associated to a polygonal mesh. However, all reported 3D-model retrieval techniques (chapters 2 and 4) consider shape as the crucial property of a 3D-object, used for similarity search.

Considered features can be classified as:

- image-based,
- 3D-geometry-based,
- functions on a sphere,
- statistical, and
- topological.

Image-based features rely upon 2D-projections of a 3D-geometry model. The projections are usually rasterized into rectangular 2D-images (sections 4.2 and 4.3). The number of projections of a model depends on the definition of a feature vector. Rectangular 2D-images of the model, generated from different viewpoints, represent silhouettes, depth-buffers, etc.

3D-geometry-based features rely upon purely 3D quantities or attributes. In section 4.4, artificial volumes are defined, and their distributions are used as 3D-shape features. A volumetric representation of a polygonal mesh model can easily

be defined (sections 2.1, 2.2, 2.5, and 4.5), discretizing the model into a voxel-grid [53]. A voxel attribute can be a binary value, a value representing the “level of presence” of the mesh (e.g., the surface area) in certain region, a value representing the distance to the model’s boundaries, etc. The voxel grid can directly be used as a feature, or it can be processed further. A subset of triangles of the mesh model can be fitted by a parametric 3D-surface, whose features are then analyzed. In the technique presented in section 2.3, a curvature index is the feature of a parametric 3D-surface.

Certain features can be regarded as *functions on a sphere*. A 3D-mesh model can be projected on an enclosing sphere (sections 4.1 and 4.6). A point on the sphere is attributed a value, which can be a distance, surface area, information about shading, curvature index, etc. Also, several concentric spheres can be used for defining functions that represent voxel attributes (sections 2.5) or encode positions of points in the 3D space \mathbb{R}^3 (4.6.6). All functions on concentric spheres are individually processed obtaining a signature for each function. The obtained signatures are then combined to form a compact descriptor of the model.

Image and 3D-geometry-based features as well as features that can be regarded as functions on a sphere have meaningful spatial interpretations. For instance, a voxel attribute is related to a specific region in the 3D-space, a 2D-image pixel attribute is related to the projected part of the model, a curvature index of a parametric 3D-surface is related to a specified group of triangles, and a value of a function on a sphere can also be interpreted. Local features of a polygonal mesh, such as a curvature index associated to a triangle (section 2.3) or the distance of the center of gravity of a triangle to the origin (section 2.1), are summarized by histograms in reported techniques [103, 105, 88, 89]. We stress that local features can be represented in a more suitable way than using histograms, where a meaningful spatial interpretation is lost. However, histogram is the most prominent tool for representing randomized features (e.g., [97, 98]). Features that are derived from several randomly selected points on the mesh, e.g., a distance between two randomly selected points (section 2.5), are purely statistical. We regard these quantities as *statistical features*. Typically, descriptors based on statistical features have inferior retrieval performance comparing to descriptors based on images, 3D-geometry, and functions on a sphere.

Topological features rely upon skeletal structures, such as medial axes [12], medial surfaces, Reeb graph [48], etc. Similarity between skeletal structures of different models is measured by matching connectivity information as well as certain proportions between skeletal primitives (e.g., nodes, branches, etc.). A technique of this type, known as “topology matching”, is presented in section 2.4.

Image-based, 3D-geometry-based, statistical features and features that can be regarded as functions on a sphere are usually represented as real valued vector of fixed dimensions. As an exception, a voxel grid can be represented as an octree (section 4.5). Topological features are represented by graphs.

Most of the feature vectors can be represented in both the spatial and spectral domains. For instance, a sequence of distances, between contour pixels of a silhouette to the origin, can be used as a feature vector in the spatial domain. The same

sequence can be transformed by the discrete Fourier transform (DFT) (4.11), obtaining a sequence of Fourier coefficients whose magnitudes are used as components of a feature vector in the spectral domain. Similar tools for rectangular images, voxel grids, and functions on a sphere are the 2D-DFT (4.26), 3D-DFT (4.48), and the Fourier transform on the sphere (spherical harmonic transform) (4.59). A wavelet transform can be used [105], as well.

As a rule, the spectral domain representation of a feature shows better retrieval performance than the spatial domain representation of the same feature. More details about this fact are given in section 1.4.

1.3.4 3D-Shape Descriptors Criteria

In this section, we give the most important criteria that should be fulfilled by definitions of 3D-shape descriptors. We consider that desirable properties of 3D-shape descriptors are the following:

1. non-restrictiveness to closed or orientable polygonal meshes;
2. non-influence of mesh anomalies (floating vertices, multiple vertices and triangles, and degenerated triangles);
3. invariance with respect to translation, rotation, scaling, and reflection of a 3D-object;
4. robustness with respect to levels-of-detail and different tessellations of a model;
5. robustness with respect to surface noise, outliers, and arbitrary topological degeneracies;
6. multi-resolution feature representation;
7. efficient feature extraction;
8. compact representation;
9. efficient search procedure;
10. shape discrimination.

Mesh models, which are available in the Internet, are not necessarily closed (definition 1.1) or orientable (definition 1.2). For instance, our 3D model collection is mostly collected from www.3dcafe.com, and more than 60% of models are not closed, while more than 30% are not orientable. Restricting a feature extraction technique to closed or orientable meshes leads to the elimination of significant number of available 3D-shapes. Although there are approaches for filling holes in meshes [8, 23, 67], we prefer defining descriptors without imposing any constraint regarding closedness or orientability. Also, mesh anomalies such as floating vertices, multiple vertices and triangles, and degenerated triangles, must not affect extracted feature

vectors. A relatively simple filtering of a mesh can be done, in order to eliminate the anomalies.

Let τ be a concatenation of translations, rotations, reflections, and scaling, and let I (1.5) be a given mesh model. Let \mathbf{f} and \mathbf{f}' be feature vectors of I and $\tau(I)$, respectively. Then, the requirement for invariance of the descriptor with respect to translation, rotation, scaling, and reflection of a 3D-object can be written as $\mathbf{f} = \mathbf{f}'$.

Geometry of 3D-objects is more and more frequently acquired by 3D-scanners. Positions of vertices can precisely be determined, while the connectivity information might not be well formed. The most common problem with meshes generated by 3D-scanners is the presence of holes. If the same object is re-scanned, then the second mesh will have different tessellation. Naturally, the appearances of both 3D-mesh models will be almost identical. Feature vectors should preserve the high similarity that exists between the mesh representations. High-quality 3D-models possess a large number of triangles (> 100000). In some applications, it is useful to have the same 3D-object represented in different levels-of-detail. For instance, for a 3D animation, if an object in a scene is very distant, then it is not necessary to use a mesh representation with 100000 polygons, in order to render the scene. The rendering complexity can be reduced by processing a less detailed polygonal mesh model of the very distant object. In figure 1.6, a model of car is represented in four different levels-of-detail. Feature vectors of the mesh models from figure 1.6 cannot be identical, but they should be “reasonably” close in the search space. Moreover, if the most complex model is used as a query, then the distances between the feature vector of the query and the other three feature vectors should be ordered according to the differences in complexity.

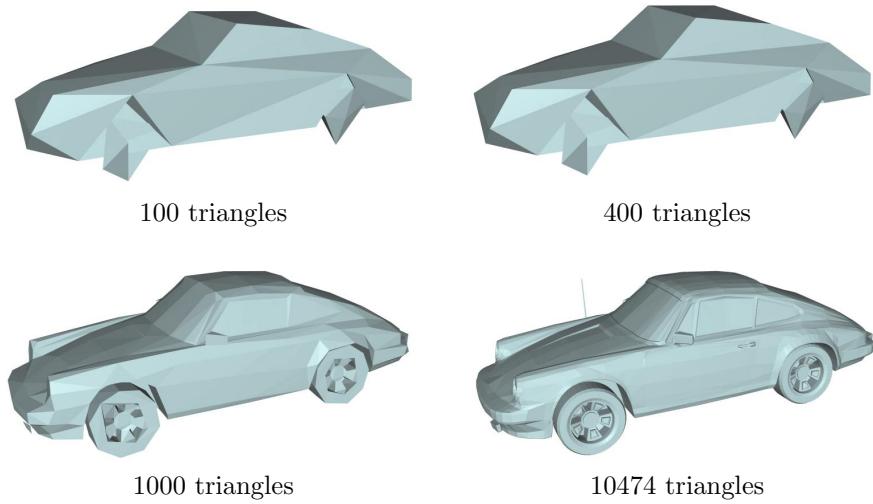


Figure 1.6: Triangle mesh model of a car in four levels-of-detail.

If the vertices \mathbf{p}_i ($i = 1, \dots, n$) of the model I (1.5) are slightly displaced, e.g., $\mathbf{p}_i \leftarrow \mathbf{p}_i + (\epsilon_x, \epsilon_y, \epsilon_z)$ with $\epsilon_x, \epsilon_y, \epsilon_z$ relatively small, then the descriptors of the mesh with displaced vertices and the original one should approximately be the same. Feature vector components should not be sensitive to a “noise” of the 3D-surface. Also, if we add an outlier to a 3D-model, e.g., we have an original model of car and add a long car antenna, then the descriptors of the original model and the model with the outlier should not differ significantly. Hence, an outlier should not affect feature vector components. Topological degeneracies such as small tunnels and handles should not significantly disturb feature vector components.

A technique for describing 3D-shapes should provide a possibility to select the resolution of feature representation. For a descriptor as a real-valued vector, it should be possible to change the dimension of vector. In other words, the resolution of a feature vector should depend on a parameter and a type of the feature.

Let $\mathbf{f}_M = (f_1, \dots, f_M)$ and $\mathbf{f}_N = (f'_1, \dots, f'_N)$ be real-valued feature vectors of a certain type in two different resolutions, with $M < N$. An *embedded multi-resolution feature representation* is provided, if

$$f_i = f'_i, \quad 1 \leq i \leq M. \quad (1.8)$$

This means that the vector \mathbf{f}_N contains all lower-dimensional feature vectors of the same type. Thus, if we search for similar models in the space of feature vectors of the dimension M , there is no need to store \mathbf{f}_M separately, because the first M components of \mathbf{f}_N can be used.

The feature extraction, including filtering and normalization steps, should be efficient. In a 3D model retrieval system, feature vectors of new models (e.g., collected by a Web-crawler application) can be extracted in an idle time. Nevertheless, it is desirable that the extraction time is as small as possible. Moreover, the representation of a feature should be concise. To represent a feature as a real-valued vector of fixed dimension is usually more compact than to use a graph or octree representations. However, the dimension of the vector should be reasonably small. We consider that a feature vector should not have more than 1000 components, and we try to keep the dimension below 500. It is crucial that the indexing of 3D-models by shape-similarity to a given query is fast. A variety of accelerating techniques can be engaged to speed-up the search, by pre-computing certain index structures. The response time of a system to a given query, i.e., the elapsed time between the moment when the query is specified and the moment when similar models are retrieved, should be at most several seconds. If a matching procedure of a pair of descriptors is too time consuming, then the descriptors of that type are not suitable for interactive retrieval applications. The l_1 or l_2 norms are very efficient distance metrics for vectors.

Finally, the most important requirement is the discriminant power of descriptors. By *discriminant power of a descriptor* we refer to the feasibility to distinguish between similar and non-similar objects to the query. When a descriptor possesses high discriminant power, we say that it has good retrieval performance. Occasionally, a descriptor of one type is extracted faster than a descriptor of the other type,

but the retrieval performance of the latter is significantly better. Also, the dimension of a feature vector can be lower than the dimension of the other type of feature vector, but the latter is much more effective. In these trade-offs we consider the retrieval performance to be more important.

1.4 Similarity Search Metrics

Since 3D-shape descriptors are usually represented as N -dimensional vectors with real-valued components, a natural way to compute distances between feature vectors is to use a vector norm.

Let $\mathbf{f}' = (f'_1, \dots, f'_N), \mathbf{f}'' = (f''_1, \dots, f''_N) \in \mathbb{R}^N$ be two feature vectors. The l_p distance between \mathbf{f}' and \mathbf{f}'' is defined by

$$d_p(\mathbf{f}', \mathbf{f}'') = \|\mathbf{f}' - \mathbf{f}''\|_p = \left(\sum_{i=1}^N |f'_i - f''_i|^p \right)^{1/p}, \quad p = 1, 2, \dots \quad (1.9)$$

For $p = 1$, we have the l_1 norm,

$$d_1(\mathbf{f}', \mathbf{f}'') = \|\mathbf{f}' - \mathbf{f}''\|_1 = \sum_{i=1}^N |f'_i - f''_i|, \quad (1.10)$$

while for $p = 2$, we have the l_2 norm of the difference $\mathbf{f}' - \mathbf{f}''$, which is called the Euclidean metric,

$$d_2(\mathbf{f}', \mathbf{f}'') = \|\mathbf{f}' - \mathbf{f}''\|_2 = \sqrt{\sum_{i=1}^N (f'_i - f''_i)^2}. \quad (1.11)$$

When $p \rightarrow \infty$, we have the l -infinity (or l_∞ or l -max) norm,

$$d_\infty(\mathbf{f}', \mathbf{f}'') = \|\mathbf{f}' - \mathbf{f}''\|_\infty = \max_{1 \leq i \leq N} |f'_i - f''_i|. \quad (1.12)$$

The l_p norm is usually ineffective as the distance metric of features in the spatial domain. From the definition of the l_p norm, we observe that the component-wise differences $f'_i - f''_i$ are equally important. According to (1.9), it is assumed that the individual components of the feature vector are independent from each other. However, the values f'_i and f''_i may correspond to a feature related to a specific region Λ_i of the 3D-space \mathbb{R}^3 . For instance, f'_i may be the extent of a model in direction $\mathbf{u}_i \in \mathbb{R}^3$ (see section 4.1). Let Λ_i , Λ_j , and Λ_k be three regions in the 3D-space such that Λ_i is very close (neighbor) to Λ_j and distant to Λ_k . In this case, instead of computing differences only between feature values in the same region, the distribution of the values across the neighboring regions should be taken into account, too. Generally, the retrieval effectiveness of vectors in the spatial domain can be improved by accounting the distribution of regions Λ_i ($i = 1, \dots, n$)

corresponding to vector components f'_i . In other words, it is desirable to account the degree of correlation between vector components.

As an example, suppose that $\mathbf{f}', \mathbf{f}'', \mathbf{f}''' \in \{0, 1\}^{16}$ are three feature vectors corresponding to the images depicted in figure 1.7. Relative positions of the dark fields (pixels), which are attributed a value of 1, are identical for \mathbf{f}' and \mathbf{f}'' . Thus, if \mathbf{f}' is a query object, then \mathbf{f}'' should be ranked above \mathbf{f}''' . However, both the l_1 and l_2 norms give the undesired ranking, since

$$d_1(\mathbf{f}', \mathbf{f}'') = 8 > d_1(\mathbf{f}', \mathbf{f}''') = 6 \quad \text{and} \quad d_2(\mathbf{f}', \mathbf{f}'') = \sqrt{8} > d_2(\mathbf{f}', \mathbf{f}''') = \sqrt{6}. \quad (1.13)$$

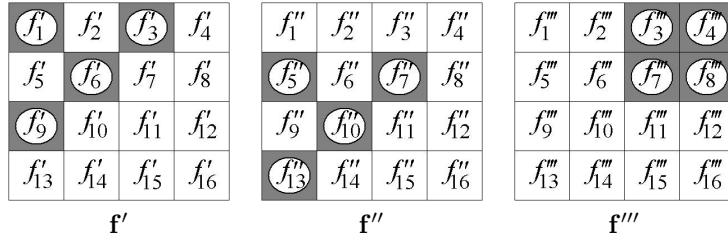


Figure 1.7: Visualizations of three feature vectors (dark fields denote the value of 1, white fields denote the value of 0). According to (1.9), $d_p(\mathbf{f}', \mathbf{f}'') > d_p(\mathbf{f}', \mathbf{f}''') = d_p(\mathbf{f}'', \mathbf{f}''')$, which is in a contradiction to a human perception.

An approach to solve the problem is presented in [2, 43]. A quadratic form distance function is defined in terms of a similarity matrix S of type $N \times N$. The element s_{ij} of the matrix S represents the similarity (correlation) between the components f_i and f_j of the feature vector $\mathbf{f} = (f_1, \dots, f_N)$. The Euclidean distance (l_2) is modified as follows

$$d_2^S(\mathbf{f}', \mathbf{f}'') = \sqrt{(\mathbf{f}' - \mathbf{f}'')^T \cdot S \cdot (\mathbf{f}' - \mathbf{f}'')} = \sqrt{\sum_{i=1}^N \sum_{j=1}^N s_{ij} (f'_i - f''_i)(f'_j - f''_j)}, \quad (1.14)$$

where S is a positive definite matrix.

In order to avoid the constraint regarding the definiteness of matrix S , we modify (1.14), which is proposed in [2], by

$$d_2^S(\mathbf{f}', \mathbf{f}'') = \sqrt{\sum_{i=1}^N \sum_{j=1}^N s_{ij} |f'_i - f''_i| |f'_j - f''_j|}, \quad (1.15)$$

Obviously, the Euclidean distance is a special case of the quadratic form distance (when $S = I_N$, I_N is the identity matrix of order N). Besides introducing similarities between components, it is possible to specify the “importance” of each component, e.g., $S = \text{diag}(\omega_1, \dots, \omega_N)$. Moreover, the elements of matrix S can

be determined by using a relevance feedback [113]. A new level of flexibility introduced by the quadratic form distance causes an increase of computational costs. The computational cost of computing d_2^S is $\mathcal{O}(N^2)$, while in the case of the l_p norm it is $\mathcal{O}(N)$. In order to reduce the cost to $\mathcal{O}(N)$, the matrix S should be sparse, i.e., only a small number of elements should be nonzero.

An example of defining the correlation matrix $S = [s_{ij}]_{N \times N}$, which can be applied for obtaining the expected ranking of features visualized in figure 1.7, is the following

$$s_{i,j} = \begin{cases} e^{-D_N(i,j)}, & D_N(i,j) \leq 2, \\ 0, & D_N(i,j) > 2, \end{cases} \quad \begin{aligned} D_N(i,j) &= (i_x - j_x)^2 + (i_y - j_y)^2, \\ i_x &= i \bmod N, \quad j_x = j \bmod N, \\ i_y &= i \text{ div } N, \quad j_y = j \text{ div } N, \end{aligned} \quad (1.16)$$

where 'mod' and 'div' denote the remainder and the integer quotient.

If we apply (1.15) and (1.16) to compute distances between feature vectors \mathbf{f}' , \mathbf{f}'' , and \mathbf{f}''' ($N = 16$), then we get

$$d_2^S(\mathbf{f}', \mathbf{f}'') = 1.93 < d_2^S(\mathbf{f}', \mathbf{f''}) = 2.75 < d_2^S(\mathbf{f}'', \mathbf{f''}) = 2.88. \quad (1.17)$$

Hence, the quadratic form distance between feature vectors of the first and second images in figure 1.7 is the smallest, which is the result that complies with human perception. Thus, the undesired ranking (1.13) can be avoided by using the quadratic form distance.

In most cases, spatial domain features can be transformed into the spectral (frequency) domain. One of the goals of this transform is to correlate vector components. The transform is also used for providing a mechanism to generate vectors with an embedded multi-resolution representation (1.8). For instance, if we apply the 2D Fast Fourier Transform (2D-FFT), which is defined in section 4.3 by (4.26), to the vectors \mathbf{f}' , \mathbf{f}'' , and \mathbf{f}''' , then we obtain complex Fourier coefficients, whose magnitudes can be used for representing the feature vectors in the spectral domain. For the example in figure 1.7, the transformation from the spatial to the spectral domain is given by (1.18).

Spatial domain	Spectral (frequency) domain	
$\mathbf{f}' = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	\longrightarrow	$\begin{bmatrix} 1 & 1/\sqrt{8} & 1/2 & 1/\sqrt{8} \\ 1/\sqrt{8} & 1/2 & 1/\sqrt{8} & 0 \\ 1/2 & 1/\sqrt{8} & 1 & 1/\sqrt{8} \\ 1/\sqrt{8} & 0 & 1/\sqrt{8} & 1/2 \end{bmatrix} = \hat{\mathbf{f}}'$
$\mathbf{f}'' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$	\longrightarrow	$\begin{bmatrix} 1 & 1/\sqrt{8} & 1/2 & 1/\sqrt{8} \\ 1/\sqrt{8} & 1/2 & 1/\sqrt{8} & 0 \\ 1/2 & 1/\sqrt{8} & 1 & 1/\sqrt{8} \\ 1/\sqrt{8} & 0 & 1/\sqrt{8} & 1/2 \end{bmatrix} = \hat{\mathbf{f}}''$
$\mathbf{f}''' = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	\longrightarrow	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/\sqrt{2} & 1/2 \\ 0 & 1/\sqrt{2} & 1 & 1/\sqrt{2} \\ 0 & 1/2 & 1/\sqrt{2} & 1/2 \end{bmatrix} = \hat{\mathbf{f}}'''$

(1.18)

As a contrast to the spatial domain, the l_p norms are reasonably effective in the spectral domain. Indeed,

$$d_1(\hat{\mathbf{f}}', \hat{\mathbf{f}}'') = 0 > d_1(\hat{\mathbf{f}}', \hat{\mathbf{f}}''') = 3 + 2\sqrt{2}, \text{ and } d_2(\hat{\mathbf{f}}', \hat{\mathbf{f}}'') = 0 > d_2(\hat{\mathbf{f}}', \hat{\mathbf{f}}''') = \sqrt{3}. \quad (1.19)$$

Note that the feature vectors from the spatial domain whose nonzero components have identical relative distributions (\mathbf{f}' and \mathbf{f}'') are transformed into identical vectors in the spectral domain ($\hat{\mathbf{f}}' = \hat{\mathbf{f}}''$).

The computation of the l_p (1.9) norm is less expensive than the computation of the quadratic form distance (1.15). Since the correlation between vector components in the spatial domain is reflected by a spectral domain representation, we consider that the l_p norm is a suitable distance metric for feature vectors in the spectral domain. Besides being efficient, the l_p distance is also effective (see results from section 5.2).

We also tested certain minimizations of the l_1 and l_2 distances, whose definitions are given in a sequel. Generally, if $\mathbf{f}' \in \mathbb{R}^N$ is a feature vector of a query model and $\mathbf{f}'' \in \mathbb{R}^N$ is a feature vector of a matching candidate, then we want to determine a parameter $\alpha \in \mathbb{R}$ so that the distance $d_p^{min}(\mathbf{f}', \mathbf{f}'')$ is minimal.

Definition 1.3 *The minimized l_p (1.9) distance d_p^{min} between vectors $\mathbf{f}' \in \mathbb{R}^N$ and $\mathbf{f}'' \in \mathbb{R}^N$ is defined by*

$$d_p^{min}(\mathbf{f}', \mathbf{f}'') = \min_{\alpha \in \mathbb{R}} d_p(\mathbf{f}', \alpha\mathbf{f}'') = \min_{\alpha \in \mathbb{R}} \|\mathbf{f}' - \alpha\mathbf{f}''\|_p.$$

For $p = 2$, the parameter α is computed using

$$\frac{d(d_2(\mathbf{f}', \alpha\mathbf{f}''))}{d\alpha} = \frac{d\left(\sqrt{\sum_{i=1}^N (f'_i - \alpha f''_i)^2}\right)}{d\alpha} = 0 \Rightarrow \alpha = \frac{\mathbf{f}' \cdot \mathbf{f}''}{\mathbf{f}''^2}. \quad (1.20)$$

For $p = 1$, the computation of the parameter α is more complex, since we want to find

$$\min_{\alpha \in \mathbb{R}} d_1(\mathbf{f}', \alpha\mathbf{f}'') = \min_{\alpha \in \mathbb{R}} \left\{ \sum_{i=1}^N |f'_i - \alpha f''_i| \right\}.$$

The distance $d_1(\mathbf{f}', \alpha\mathbf{f}'')$ is a piecewise linear function of variable α , in the intervals $(-\infty, g_1], [g_1, g_2], \dots, [g_{M-1}, g_M], [g_M, +\infty)$ ($M \leq N$), where

$$g_1 \leq g_2 \leq \dots \leq g_M, \quad \{g_1, \dots, g_M\} = \{f'_i/f''_i \mid f''_i \neq 0, 1 \leq i \leq N\}.$$

Therefore, it holds $\alpha \in \{g_1, \dots, g_M\}$, whence

$$d_1^{min}(\mathbf{f}', \mathbf{f}'') = \min_{1 \leq i \leq M} d_1(\mathbf{f}', g_i \mathbf{f}''). \quad (1.21)$$

Note that the computational complexity of d_1^{min} is $\mathcal{O}(M \cdot N)$, if a brute force is applied. The complexity of our calculation of d_1^{min} , which is given by the pseudocode in figure 1.8, is $\mathcal{O}(N \log N)$.

$\mathbf{f}' = (f'_1, \dots, f'_N)$, $\mathbf{f}'' = (f''_1, \dots, f''_N)$;
Form an array $\mathbf{m} = \{m_1, \dots, m_M\}$, $m_i = (g_i, n_i, s_i) \in \mathbb{R} \times \mathbb{N} \times \{-1, 1\}$,
 $g_i = f'_i/f''_i$, $(f''_i \neq 0)$, $n_i = i$, $s_i = \text{sign}(f''_i)$;
Sort \mathbf{m} so that $g_i \leq g_{i+1}$ ($1 \leq i < M$) – complexity $\mathcal{O}(M \log M)$;
 $a = \sum_{i=1}^M s_{n_i} f'_{n_i} + \sum_{f''_i=0} |f'_i|$, $b = \sum_{i=1}^M |f''_{n_i}|$;
 $d_1^{min} = a - g_1 b$;
for $i = 2, \dots, M$
 $a \leftarrow a - 2s_{n_i} f'_{n_i}$, $b \leftarrow b - 2|f''_{n_i}|$;
if $(d_1^{min} > a - g_i b) \Rightarrow d_1^{min} = a - g_i b$;

Figure 1.8: Algorithm of complexity $\mathcal{O}(N \log N)$ for computing the minimal l_1 distance.

Our experimental results (section 5.2) show that retrieval performance of some feature vectors in the spatial domain (e.g., the ray-based approach from section 4.1) is better when the minimized l_1 distance (1.21) is used instead of the l_1 (1.10). This also holds for certain feature vectors in the spectral domain.

We consider that the presented distance calculations are reasonably suitable for our application. Nevertheless, we do not exclude that, e.g., modifications of the Hausdorff distance, the Earth Mover’s distance [21], the Bhattacharyya distance [75], or the Kullback-Leibler distance [148] might be more effective for certain feature vectors. Similarity measures and algorithms suitable for content-based search are also addressed in [139, 9, 38, 36].

1.5 Tools for Evaluation of Retrieval Effectiveness

We use common tools from information retrieval theory [7] to evaluate retrieval performance of descriptors. The tools are based on *precision* and *recall* values. The construction precision-recall diagrams as well as computations of the R-precision [7] and the Bull’s Eye Performance [87] values, which we use to compare competing feature vectors, are given in this section.

Let Σ be a collection of 3D-objects. Let $C_1, \dots, C_K \subset \Sigma$ be disjunctive classes of similar objects so that

$$C_i = \{c_1^{(i)}, \dots, c_{n_i}^{(i)}\}, \quad c_k^{(i)} \in \Sigma \quad (1 \leq k \leq n_i), \quad \text{and} \quad i \neq j \Rightarrow C_i \cap C_j = \emptyset. \quad (1.22)$$

Let C and U be the sets of all classified and unclassified objects, respectively,

$$C = \bigcup_{i=1}^K C_i, \quad U = \Sigma \setminus C. \quad (1.23)$$

Models belonging to the same class are regarded as relevant to each other. The categorization of similar (relevant) objects into classes serves as a *ground truth*.

Suppose that $q \in C_i$ is a query model and

$$R = \{s_1, \dots, s_r\} \quad (1.24)$$

is the set of objects retrieved by using a selected descriptor and matching criterion. The object s_1 is regarded as the first (best) match (or nearest neighbor), s_2 is the second match, etc. We consider that relevant models to the query q are in the same class with the query, i.e., the set of relevant objects is $C_i \setminus \{q\}$. The set of retrieved models that are relevant (hits) is denoted by M ,

$$M = (C_i \setminus \{q\}) \cap R = \{h_1, \dots, h_m\}. \quad (1.25)$$

The objective evaluation is based on the *ground truth* defined by (1.22) and (1.23).

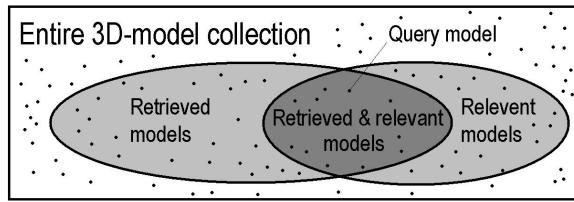


Figure 1.9: For a given query model, a fraction of retrieved models is relevant.

Precision is the proportion of retrieved models that are relevant and recall is the proportion of the relevant models actually retrieved,

$$\text{precision} = \frac{|M|}{|R|} = \frac{m}{r}, \quad \text{recall} = \frac{|M|}{|C_i \setminus \{q\}|} = \frac{m}{n_i - 1}. \quad (1.26)$$

If figure 1.9, four sets of models are illustrated, the entire collection, the set of retrieved models (R), the set of relevant objects ($C_i \setminus \{q\}$), and the set of retrieved relevant objects (M).

The construction of the precision-recall diagrams, which are widely used in section 5.2, is demonstrated by the following example.

Let $C_i \setminus \{q\} = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$, i.e., there are 7 objects relevant to the query q . Firstly, we rank all objects from the collection Σ using the selected descriptor and matching criterion. Suppose that

$$R = \{c_4, r_1, c_2, c_7, r_2, c_1, r_3, r_4, c_6, r_5, c_5, r_6, r_7, r_8, c_3, \dots\}$$

is the ranking of all models from Σ according to the similarity to q , where $r_i \in \Sigma \setminus C_i$, $i = 1, 2, \dots$. Thus, the last relevant model lies at position 15 (match number 15). Next, we construct the table 1.1.

The number of columns in table 1.1 depends on the size of the class of relevant models. In order to average values of precision over classes of different sizes, we estimate precision at standard recall values

$$\text{recall}_k = \frac{k}{G}, \quad 1 \leq k \leq G, \quad G \in \mathbb{N},$$

Match no.	1	3	4	6	9	11	15
Object	c_4	c_2	c_7	c_1	c_6	c_5	c_3
Recall	1/7	2/7	3/7	4/7	5/7	6/7	7/7
Precision	1/1	2/3	3/4	4/6	5/9	6/11	7/15

Table 1.1: An example of computing precision and recall values using the ranking of relevant objects.

by applying linear interpolation to the values from table 1.1. Interpolation results for $G = 5$ and $G = 10$ are shown in table 1.2.

Recall	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
Precision ($G = 5$)	-	0.87	-	0.73	-	0.64	-	0.55	-	0.47
Precision ($G = 10$)	1.00	0.87	0.68	0.73	0.71	0.64	0.57	0.55	0.52	0.47

Table 1.2: Interpolation of precision at standard recall values $\text{recall}_i = i/G$ ($1 \leq i \leq G$) using data from table 1.1, for $G = 5$ and $G = 10$.

The precision-recall diagrams for the values from table 1.1 and the interpolations for $G = 5$, $G = 10$, and $G = 20$ are shown in figure 1.10, where both recall and precision are expressed in percentages. Obviously, the higher the value of G , the better the interpolation. In section 5.2, all precision recall curves possess $G = 20$ standard recall values.

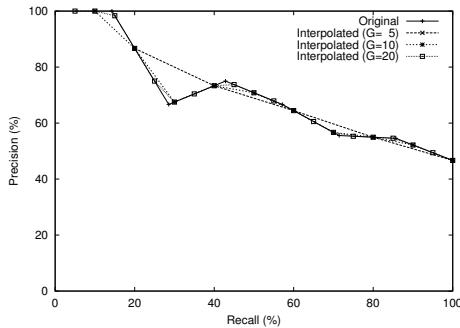


Figure 1.10: Original and three interpolated precision-recall curves.

Values in table 1.1 describe ranking results for a single query. In order to evaluate a descriptor and matching criterion for a selected query set $Q \ni q$, we take each object from the set Q as a query, retrieve models, compute and interpolate precision vs. recall for the single query, and average the results. By examining averaged precision/recall diagrams for different queries (and classes) we estimate the retrieval performance of descriptor for selected settings (e.g., type, parameters, representation, and matching criterion).

We also compute two cumulative parameters, the average precision \bar{p}_{100} over the whole recall range $1/G \leq \text{recall} \leq 1$ and the average precision \bar{p}_{50} over the recall range $1/G \leq \text{recall} \leq 0.5$. We consider these values to be useful for comparing overall performance of different descriptors. For the interpolated values from table 1.2 obtained for $G = 10$, we have

$$\bar{p}_{100} = \frac{1.00 + 0.87 + 0.68 + 0.73 + 0.71 + 0.64 + 0.57 + 0.55 + 0.52 + 0.47}{10} = 0.674 \text{ (or } 67.4\%)$$

and $\bar{p}_{50} = \frac{1.00 + 0.87 + 0.68 + 0.73 + 0.71}{5} = 0.798 \text{ (or } 79.8\%).$

The MPEG-7 has adopted the *Bull's Eye Performance (BEP)* [87] as the evaluation tool for retrieval performance of visual descriptors (section 1.2). For a query $q \in C_i$ (1.22), the *BEP* score is defined as the percentage of all relevant objects that are retrieved among the top $2(n_i - 1)$ ranked objects, where n_i is the size of the class C_i . In other words, the *BEP* is the value of recall when $2(n_i - 1)$ objects are retrieved. Thus, using (1.22), (1.25), and (1.24) we write

$$BEP = \frac{|(C_i \setminus \{q\}) \cap R|}{|C_i \setminus \{q\}|} = \frac{|M|}{|C_i \setminus \{q\}|} = \frac{m}{n_i - 1}, \quad \text{when } r = 2(n_i - 1). \quad (1.27)$$

We also use the R-precision (*RP*) [7], which is the fraction of relevant objects that are retrieved, when the number of retrieved is equal to the number of relevant objects. Using (1.22), (1.25) and (1.24), we have

$$RP = \frac{|M|}{|R|} = \frac{m}{r}, \quad \text{when } r = n_i - 1. \quad (1.28)$$

Note that $r = n_i - 1 \Leftrightarrow \text{Precision} = \text{Recall}$ (1.26).

Some authors [97, 98, 129] refer to the values of *RP* and *BEP* as the “First Tier” and “Second Tier”, respectively.

For the example in table 1.1, since $n_i = 8$, we observe that the number of hits among the first 7 (resp. 14) objects is 4 (resp. 6), whence

$$BEP = 6/7 = 0.857 \text{ (or } 85.7\%) \quad \text{and} \quad RP = 4/7 = 0.571 \text{ (or } 57.1\%).$$

In order to obtain a measure of retrieval performance on a set of queries, the values of *BEP* and R-precision are averaged. We consider that an absolute difference of at least 0.02 (or 2%), between average values of *BEP* (or R-precision) for two competing descriptors, is significant.

As an example, in figure 1.11 two descriptors are compared by averaging precision at standard recall values ($G = 20$) using a given set of queries. Since for all recall values the precision-recall curve of descriptor 1 is clearly above the precision-recall curve of descriptor 2 and all values of \bar{p}_{50} , \bar{p}_{100} , *BEP*, and R-precision are significantly higher for the descriptor 1, we conclude that the descriptor 1 possesses superior retrieval performance with respect to the descriptor 2.

Besides comparing competing descriptors, precision-recall diagrams can be used for estimating the number of retrieved objects in order to find a desired number

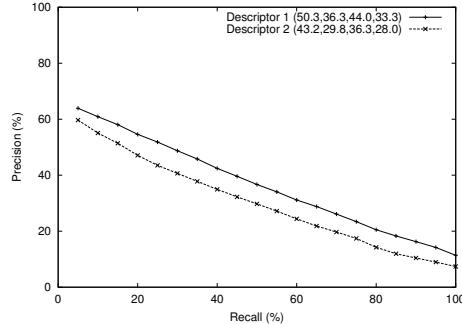


Figure 1.11: Comparison of performance of two descriptors each of which uses specific settings (type, parameters, matching criterion, etc.). The values of \bar{p}_{50} , \bar{p}_{100} , BEP, and R-precision are given in the brackets, respectively.

of relevant items, assuming that the total number of relevant models is known. For instance, suppose we use a model of car as the query and we want to retrieve 10 relevant models from a collection containing $K \gg 10$ models of cars. Then, we calculate the recall value ($10/K$) and find the corresponding value of precision p , using a precision-recall diagram of the engaged descriptor. The precision-recall diagram should be averaged for queries of cars. We estimate that if we retrieve around $10/p$ models, 10 of them should be cars.

An ideal precision-recall curve is constant, having precision=100% for the whole recall range. In practice, precision is well below 100% even at small recall values, and there is a decreasing tendency with the increase of recall values. A precision-recall curve for a single query model (e.g., figure 1.10) appears to be “broken”. An average precision-recall curve (for the whole query set) is usually monotonically non-increasing, i.e., for two points belonging to a curve, (p_1, r_1) and (p_2, r_2) , we have

$$r_1 < r_2 \Rightarrow p_1 \geq p_2.$$

In [7], it is recommended to force this monotonicity by setting $p_2 \leftarrow \min\{p_1, p_2\}$. Nevertheless, in results presented in section 5.2 the monotonicity is not forced.

Chapter 2

Related Research Work

In this chapter, we describe several 3D-shape descriptors proposed by other authors. We focus the related work on the same topic as ours – description of 3D-shape of mesh models. We stress that there is a significant work (e.g., [9, 10, 150, 25, 64, 68, 69, 22], etc.) in the area of Computer Vision, which can be regarded as a somewhat similar topic. However, we do not need to infer all the information about 3D-shape from one or more 2D-images. We deal with 3D-objects represented as polygonal meshes (section 1.3.1), and we use the lists of geometry (1.3) and topology (1.2) to extract descriptors.

As far as we know, the first papers on content-based retrieval of polygonal mesh models appeared in 1997. Therefore, we decided to describe the pioneering work of Paquet [101, 103, 105, 102, 104, 100, 99], cords and moments-based descriptors. Then, we present a descriptor based on equivalence classes proposed by Suzuki [126, 125, 127, 124], followed by the MPEG-7 shape spectrum descriptor [89]. An interesting technique, called topology matching [48], is also presented. Finally, several descriptors have recently been proposed by the Princeton Shape Analysis and Retrieval Group, shape distributions [97, 98], the reflective symmetry descriptor [56, 55], and descriptors based on concentric spheres [57, 36].

The criteria for selecting the descriptors that are presented in this chapter are historical reasons and the impact on the area of retrieval of polygonal mesh models. However, more and more researchers are attracted by the topic. A variety of approaches for describing shape of a 3D mesh model can be found in [2, 72, 153, 151, 95, 27, 28, 19, 77, 11, 110, 65, 129, 93, 91, 92].

In our experiments (section 5.2.12), we tested Paquet’s cords-based and moments-based descriptors, Suzuki’s descriptor based on equivalence classes, MPEG-7 shape spectrum descriptor, and descriptors based on shape distributions and functions on concentric spheres. The program code for feature extraction of the shape spectrum approach is taken from the MPEG-7 *eXperimentation Model* [84]. A descriptor reported in [58] is extracted using binaries (executables) provided by the authors [108]. Implementation of feature extractors for other approaches has been done by ourselves. Certain feature extraction techniques are underspecified [126, 125] or

ambiguous [56, 55]. In order to reduce the probability of missing the correct feature extraction procedure, we implemented several variants of feature extractors, in cases of non-complete descriptor specifications.

2.1 Cords and Moments-Based Descriptors

One of the first works on shape-similarity search for 3D-mesh models was reported in 1997 [101] by Paquet and Rioux. In this section we review three 3D-shape descriptors proposed by these authors, cords-based, moments-based, and wavelet transform-based.

In order to secure rotation invariance of descriptors, a 3D-object, represented as a triangle mesh, is rotated first, by applying the Principal Component Analysis (PCA) [51] to the centers of gravity of triangles (3.15) [103, 105]. More details about the normalization of rotation are given in section 3.3. The normalization of reflection (flipping) is presented in [105]. After orienting the model, i.e., securing invariance with respect to rotation and reflection of a mesh model, the feature extraction follows.

In [103, 105], a *cord* \mathbf{c}_i is defined to be a vector that goes from the center of gravity \mathbf{m}_I of a mesh model to the center of gravity \mathbf{g}_i of a triangle T_i ($i = 1, \dots, m$) (1.2). The center of gravity of the point set I (1.5) is computed by

$$\mathbf{m}_I = (m_x, m_y, m_z) = \frac{1}{S} \sum_{i=1}^m S_i \mathbf{g}_i, \quad \mathbf{g}_i = (g_{x_i}, g_{y_i}, g_{z_i}), \quad (2.1)$$

where S and S_i are defined by (1.6). Thus, the number of cords is equal to the number of triangles of the mesh model. The *cords-based descriptor* consists of three histograms. The first histogram represents the distribution of the angles α_i between the cords and the first principle axis, while the second histogram provides the distribution of the angles β_i between the cords and the second principle axis, where $0 \leq \alpha_i, \beta_i \leq \pi$. The third histogram describes the distribution of the norms of cords $\|\mathbf{c}_i\|$ so that the smallest value is zero and the largest value corresponds to the norm M of the longest cord. In order to provide independence with respect to the number of cords, all three histograms are normalized using the total number of cords m . Therefore, for each triangle we compute

$$\begin{aligned} \mathbf{c}_i &= \mathbf{g}_i - \mathbf{m}_I, \quad \|\mathbf{c}_i\| = \sqrt{(g_{x_i} - m_x)^2 + (g_{y_i} - m_y)^2 + (g_{z_i} - m_z)^2}, \\ M &= \max_{1 \leq i \leq m} \|\mathbf{c}_i\|, \quad \alpha_i = \arccos \left(\frac{g_{x_i} - m_x}{\|\mathbf{c}_i\|} \right), \quad \beta_i = \arccos \left(\frac{g_{y_i} - m_y}{\|\mathbf{c}_i\|} \right). \end{aligned} \quad (2.2)$$

The number of bins in each histogram is equal N , whence the dimension of the cords-based feature vector \mathbf{f} is equal $3N$,

$$\mathbf{f} = (h_1^{(1)}, \dots, h_N^{(1)}, h_1^{(2)}, \dots, h_N^{(2)}, h_1^{(3)}, \dots, h_N^{(3)}). \quad (2.3)$$

The forming of the feature vector \mathbf{f} is described by the following pseudocode

$$\begin{aligned}
 h_i^{(k)} &= 0, \quad i = 1, \dots, N, \quad k = 1, 2, 3; \\
 \text{for } i &= 1, \dots, m \\
 k &= \lceil N \cdot \alpha_i / \pi \rceil; \quad h_k^{(1)} \leftarrow h_k^{(1)} + 1/m; \\
 k &= \lceil N \cdot \beta_i / \pi \rceil; \quad h_k^{(2)} \leftarrow h_k^{(2)} + 1/m; \\
 k &= \lceil N \cdot ||\mathbf{c}_i|| / M \rceil; \quad h_k^{(3)} \leftarrow h_k^{(3)} + 1/m;
 \end{aligned} \tag{2.4}$$

Besides possessing rotation and reflection invariance (normalization), the cords-based feature vector is invariant with respect to translation of a mesh model, because cords are determined relatively to the center of gravity \mathbf{m}_I . Invariance with respect to scaling is provided by binning cord norms $||\mathbf{c}_i||$ so that the highest value correspond to the maximal cord norm M . The average extraction time of this descriptor is extremely low. On a PC with an 1.4 GHz AMD processor running Windows 2000, the cords-based descriptor of a model from the MPEG-7 set (section 5.1) is extracted in less than 9ms, on average. The computational complexity is $\mathcal{O}(m)$. The authors suggested to use $N = 40$ as the best choice for the number of bins.

The discriminant power of the cords-based descriptor is not high, mostly because differing sizes of triangles are not taken into account. Even if the area of triangle T_i is significantly larger than the area of triangle T_j , the corresponding cords \mathbf{c}_i and \mathbf{c}_j are equally important. Different tessellations as well as different levels-of-detail of a 3D-mesh model usually have significantly different descriptors. As a trivial example, if one takes a triangle T_i of a mesh and replaces it with k triangles $\{T_i^{(1)}, \dots, T_i^{(k)}\}$ so that for $a \neq b$, it holds $T_i^{(a)} \cap T_i^{(b)} = \emptyset$ and $T_i = \cup_{j=1}^k T_i^{(j)}$, then we will have absolutely the same appearance of the object, but the corresponding cords-based feature vectors will be different (the number of cords will be $m + k - 1$ instead of m). This example is illustrated in figure 2.1, where T_i is replaced by 27 smaller triangles.

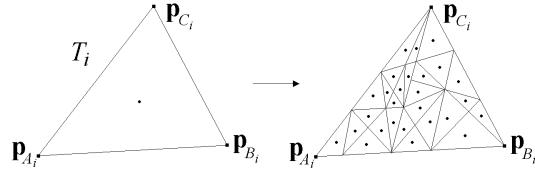


Figure 2.1: Triangle T_i is replaced by 27 smaller triangles, whence the number of cords is increased by 26. The endpoints of cords are denoted by the dots.

Moreover, the cords-based descriptor is sensitive to surface noise and topological degeneracies. Although the authors regard a cord as a "slowly varying normal vector that capture regional characteristics" [105], the overall retrieval performance of this descriptor is not good (see section 5.2.12).

The *feature vector based on statistical moments* is proposed in [105]. After normalizing rotation and reflection of mesh vertices, a 3D-statistical moment M_{qrs} is defined as

$$M_{qrs} = \sum_{i=1}^m S_i (g_{x_i} - m_x)^q (g_{y_i} - m_y)^r (g_{z_i} - m_z)^s, \quad q, r, s \geq 0, \quad (2.5)$$

where S_i (1.6) denotes the surface area of triangle T_i (1.2). The authors did not give any details how to organize the moments M_{qrs} in order to compose a feature vector. Therefore, we tested this approach in the following manner. Firstly, we observe that $M_{000} = \sum_{i=1}^m S_i = S$ (1.6) and $M_{100} = M_{010} = M_{001} = 0$. Indeed, from (2.1) and (2.5), we obtain for M_{100} (similar for M_{010} and M_{001})

$$M_{100} = \sum_{i=1}^m S_i (g_{x_i} - m_x) = \sum_{i=1}^m S_i g_{x_i} - m_x \sum_{i=1}^m S_i = \sum_{i=1}^m S_i g_{x_i} - m_x S = 0.$$

We regard the sum $q + r + s$ as the order of the moment, and form the feature vector using all moments of order 2 to N , i.e., $2 \leq q + r + s \leq N$. The dimension of the moments-based feature vector \mathbf{f} is equal to $(N+1)(N+2)(N+3)/6 - 4$. The vector is formed as described by the algorithm in figure 2.2. Note that an embedded multi-resolution representation is provided (1.8).

```

dim = (N + 1)(N + 2)(N + 3)/6 - 4;
f = (f1, ..., fdim);
i = 1;
for k = 2, ..., N // k-order
    for q = 0, ..., k
        for r = 0, ..., k - q
            s = k - r - s;
            fi ← Mqrs;
            i ← i + 1;

```

Figure 2.2: Algorithm for forming the feature vector based on statistical moments.

The computation of the dimension dim , when the vector is composed according to the algorithm in figure 2.2, is given by

$$dim = \sum_{k=2}^N \sum_{q=0}^k (k - q + 1) = \sum_{k=2}^N \frac{k^2 + 3k + 2}{2} = \frac{(N+1)(N+2)(N+3)}{6} - 4.$$

We set $N = 12$ and obtain the feature vector of dimension 451, thereby all lower-dimensional vectors (for $N = 2, \dots, 11$) are contained in the largest vector. The average extraction time of the feature vector with 451 components is 84ms, for a

model from the MPEG-7 set (section 5.1), on a PC with an 1.4 GHz AMD processor running Windows 2000.

The authors [105] considered the scale of a model as a feature and they did not provide any mean to secure scaling invariance. This is a contradiction to the invariance requirement 3 in section 1.3.4. Consequently, the retrieval performance is very poor (see section 5.2.12). However, even if we fix the scale (section 3.4), the effectiveness is not good enough. Therefore, we conclude that the approach based on statistical moments possesses rather theoretical than practical significance.

We implemented both cords and moments-based descriptors and evaluated their performance (section 5.2.12).

The same authors proposed a *wavelet transform-based descriptor* [105], which is rather underspecified. The main idea is to voxelize a 3D-model and to apply a wavelet transform in each dimension. However, no information about the voxelization is given, i.e., it is not specified what is the region of voxelization as well as how voxel attributes are computed. The authors proposed to use Daubechies-4 (DAU4) wavelets. After performing the wavelet transform, the obtained coefficients are processed further. Firstly, the logarithm of each coefficient is computed, in order to enhance the coefficients corresponding to fine details, whose magnitudes are usually small. Then, the sums of logarithms at each level of resolution are computed and regarded as components of the wavelet transform-based feature vector.

The web-based 3D model retrieval system [99] uses methods presented in [103, 105].

2.2 Descriptor based on Equivalence Classes

In [126, 125], Suzuki et al. proposed a 3D-shape descriptor, which they regard as “rotation invariant shape descriptor”. Originally, the descriptor presented in [126] is not invariant with respect to rotation for an arbitrary angle around an arbitrary axis, but only with respect to rotation of 90 degrees around the coordinate axes. However, the Principal Component Analysis (PCA) [51] is used in [125] for finding a canonical orientation of a model. Since no details about how the PCA is applied to triangle meshes are given, we apply our Continuous PCA (see section 3.4) to rotate an arbitrary triangle mesh model.

After the rotation, a unit cube is formed. The unit cube is not defined neither in [126] nor [125]. Moreover, it is not specified if the unit cube encompasses the whole object. Therefore, we tested three possibilities for defining the unit cube, using cuboid regions that are originally defined for our methods (chapter 4). The tested definitions of the unit cube are the following:

- CBC: the unit cube is the tightest enclosing cube so that the center of the cube coincides with the center of gravity of the mesh model (see definition 4.1);
- EBB: the tightest bounding box is extended to the unit cube so that the centers of both coincide and the length of edge of the cube is equal to the length of the longest edge of the bounding box (see definition 4.3);

- CC2: the center of the unit cube coincides with the center of gravity of a mesh model, while the length of edge is four times the average distance of points on the model's surface to the center of gravity (see definition 4.4).

All cubes, CBC, EBB and CC2, have faces parallel to the coordinate hyper-planes. Note that CBC and EBB encompass the whole object, while there might be some parts of the model outside the CC2. The CC2 is created in order to improve the robustness with respect to outliers (requirement 5 in section 1.3.4).

The unit cube, which is determined by its diagonal points $\mathbf{b}_{min} = (\underline{b}_x, \underline{b}_y, \underline{b}_z) \in \mathbb{R}^3$ and $\mathbf{b}_{max} = (\bar{b}_x, \bar{b}_y, \bar{b}_z) \in \mathbb{R}^3$ ($\underline{b}_x < \bar{b}_x$, $\underline{b}_y < \bar{b}_y$, and $\underline{b}_z < \bar{b}_z$), is subdivided into $(2N + 1)^3$ ($N \in \mathbb{N}$) cubes $\gamma_{ijk} \subset \mathbb{R}^3$ ($-N \leq i, j, k \leq N$). The cube γ_{ijk} is defined by its diagonal points \mathbf{p}_{ijk} and $\mathbf{p}_{ijk} + \mathbf{d}$, where

$$\mathbf{p}_{ijk} = \mathbf{b}_{min} + \mathbf{d} \cdot (i + N, j + N, k + N), \quad \mathbf{d} = (d_x, d_y, d_z) = \frac{\mathbf{b}_{max} - \mathbf{b}_{min}}{2N + 1}.$$

The 3D-space \mathbb{R}^3 is discretized by associating a point $(i, j, k) \in \mathbb{Z}^3$ to the cube γ_{ijk} . The point (i, j, k) is attributed a real value $v_{ijk} \in \mathbb{R}$. Thus, v_{ijk} attributes the cube γ_{ijk} .

Next, a point cloud of a mesh model is created so that the points are uniformly distributed on the surface of the object. Each attribute v_{ijk} is equal to the fraction of points belonging to the cube γ_{ijk} . Hence,

$$\sum_{i=-N}^N \sum_{j=-N}^N \sum_{k=-N}^N v_{ijk} = 1.$$

The authors defined *equivalence classes* in \mathbb{Z}^3 . Points $\mathbf{x} = (x_i, x_j, x_k) \in \mathbb{Z}^3$ and $\mathbf{y} = (y_i, y_j, y_k) \in \mathbb{Z}^3$ belong to the same equivalence class if and only if there is a finite concatenation ρ of rotations ρ_x , ρ_y , and ρ_z (2.6) of 90 degrees around the coordinate axes such that $\mathbf{y} = \rho(\mathbf{x})$.

$$\rho_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \quad \rho_y = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \rho_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.6)$$

Hence, if it is possible to transform \mathbf{x} into \mathbf{y} by applying a finite sequence of rotations of 90 degrees around the coordinate axes, then \mathbf{x} and \mathbf{y} are in the same equivalence class.

The equivalence classes are initially used to secure invariance with respect to rotations of 90 degrees around the coordinate axes [126]. The invariance of the feature vector was attained using a well known general principle. Any feature vector can be made invariant with respect to a finite group of transformations of space by summing or averaging feature vectors computed from all possible transformations of an object.

For the set of points $\{(i, j, k) \in \mathbb{Z}^3 \mid -N \leq i, j, k \leq N\}$, the total number of

equivalence classes is calculated by

$$\begin{aligned} \dim &= \sum_{i=0}^N \sum_{j=0}^{N-i} \sum_{k=0}^{N-i-j} 1 + \sum_{i=3}^N \sum_{j=3}^{N-i} \sum_{k=3}^{N-i-j} 1 \\ &= \frac{(N+3)(N+2)(N+1) + N(N-1)(N-2)}{6}. \end{aligned} \quad (2.7)$$

The dimension of the feature vector, which is proposed in [126, 125], is equal to the total number of equivalence classes, \dim . The value of the component f_i ($i \in \{1, \dots, \dim\}$) of the feature vector $\mathbf{f} = (f_1, \dots, f_{\dim})$ is associated to an equivalence class and is equal to the sum of all attributes v_{ijk} of the points belonging to the equivalence class. We refer to the approach presented in [126, 125] as *descriptor based on equivalence classes*.

As an example, for $N = 1$ we have a $3 \times 3 \times 3$ grid with $\dim = 4$ equivalence classes (figure 2.3). The components of the feature vector $\mathbf{f} = (f_1, f_2, f_3, f_4)$ are computed by summing up the attributes v_{ijk} in corresponding classes.

$$\begin{aligned} f_1 &= v_{0,0,0}, \\ f_2 &= v_{-1,0,0} + v_{0,-1,0} + v_{0,0,-1} + v_{1,0,0} + v_{0,1,0} + v_{0,0,1}, \\ f_3 &= v_{-1,-1,0} + v_{-1,1,0} + v_{1,-1,0} + v_{1,1,0} + v_{-1,0,-1} + v_{-1,0,1} + v_{1,0,-1} + v_{0,1,1} \\ &\quad + v_{0,-1,-1} + v_{0,-1,1} + v_{0,1,-1} + v_{0,1,1}, \\ f_4 &= v_{-1,-1,-1} + v_{-1,-1,1} + v_{-1,1,-1} + v_{-1,1,1} + v_{1,-1,-1} + v_{1,-1,1} + v_{1,1,-1} + v_{1,1,1}. \end{aligned}$$

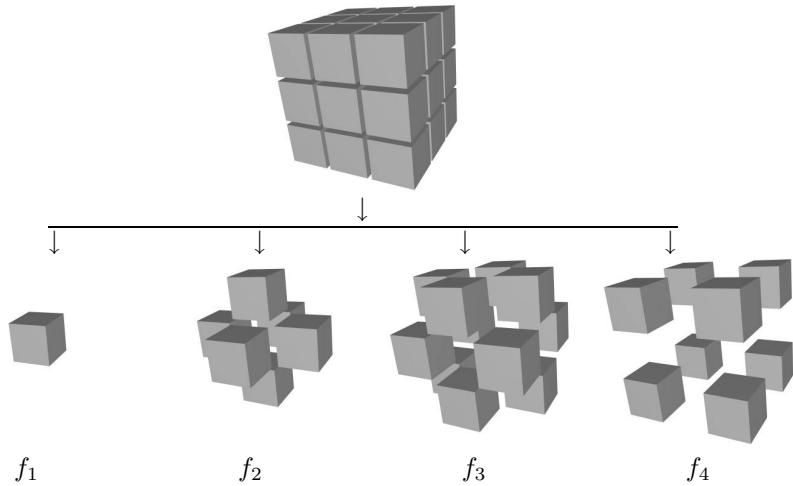


Figure 2.3: Equivalence classes of the $3 \times 3 \times 3$ grid ($N = 1$).

The average extraction times, for various dimensions of the feature vector based on equivalence classes, are given in table 2.1. The descriptors are extracted using our implementation of the approach. The results are obtained on a PC with an

1.4 GHz AMD processor running Windows 2000, using the MPEG-7 set of models (section 5.1).

N	1	2	3	4	5	6	7	8	9	10	11
dim	4	10	21	39	66	104	155	221	304	406	529
Time [ms]	17	26	34	45	53	63	74	86	100	117	135

Table 2.1: Average extraction times (in milliseconds) of the descriptor based on equivalence classes.

According to our evaluation (section 5.2.12), the best choice of the unit cube is EBB (definition 4.3). The feature vector based on equivalence classes of dimension 406 ($N = 10$, i.e., $21 \times 21 \times 21$ grid) possesses better overall retrieval performance than vectors of lower dimensions. However, the vector of dimension 155 is just slightly less effective, whence if there is a strong requirement for a compact feature representation, then we recommend $dim = 155$. The descriptor based on equivalence classes significantly outperforms the cords and moments-based descriptors from section 2.1.

2.3 Shape Spectrum Descriptor

The MPEG-7 shape spectrum descriptor [88, 89] is extracted *without any normalization* of polygonal mesh model. This descriptor can be interpreted as a histogram of curvature indices. A curvature index is associated to each triangle of a mesh and is obtained by local parametric surface fitting around a triangle.

In this approach, mesh models are supposed to be regular enough, i.e., without multiple edges, floating (isolated) faces or vertices, or any other topological degeneracies. If such degeneracies occur, a preliminary filtering step of the 3D mesh model is highly recommended before applying the feature extraction. There is also a requirement for orientability of a triangle mesh (see definition 1.2).

The curvature index $\Gamma_j \in [0, 1]$ (2.8), associated to the triangle T_j , $1 \leq j \leq m$, of a mesh model (1.2), is defined using the two principal curvatures k'_j and k''_j .

$$\Gamma_j = \frac{1}{2} - \frac{1}{\pi} \arctan \frac{k'_j + k''_j}{k'_j - k''_j}, \quad k'_j \geq k''_j. \quad (2.8)$$

Before computing Γ_j , a checking for singularities is performed.

Two triangles T_i and T_j are considered to be adjacent if they share a common vertex. Let Λ_j be the set of indices of all triangles adjacent to T_j and σ_j be the sum of surface areas given by

$$\Lambda_j = \{i \mid T_i \text{ adjacent to } T_j\}, \quad \sigma_j = S_j + \sum_{i \in \Lambda_j} S_i. \quad (2.9)$$

If the number of triangles adjacent to T_j (the cardinality of the set Λ_j) is less than 5, then the area σ_j is regarded as an area of singular surface.

If no singularity is stated, then the computation of the curvature index Γ_j proceeds in the following steps:

1. Computation of the mean normal vector of the triangle T_j ;
2. Local parametric surface fitting around T_j ;
3. Calculation of the principal curvatures k'_j and k''_j .
4. Checking for planar surfaces.

The mean normal vector $\bar{\mathbf{n}}_j$ (2.10) of the triangle T_j is defined as the weighted average of normal vectors of adjacent triangles to T_j ,

$$\bar{\mathbf{n}}_j = \frac{\mathbf{s}}{\|\mathbf{s}\|}, \quad \mathbf{s} = \sum_{i \in \Lambda_j} S_i \mathbf{n}_i, \quad (2.10)$$

where S_i and \mathbf{n}_i are the surface area and normal vector of triangle T_i , while Λ_j is given by (2.9). Obviously, inconsistent orientations of triangles lead to wrong values of the mean normal vectors. Hence, the triangle mesh supposed to be orientable (definition 1.2) in order to enable the correct estimation of the mean normal vector.

The authors stress [88] that adjacent triangles of the second order (adjacent to adjacent) and higher order (further recursion) can be considered. In that case, the computational complexity increases proportionally to the size of the set Λ_j . However, retrieval performance will not be improved [88].

Local fitting of a parametric 3D-surface is performed through the centers of gravity of the triangle T_j and the adjacent triangles T_i ($i \in A_j$). The parametric surface is quadratic, expressed as a second order polynomial,

$$z = f(x, y) = a_0x^2 + a_1y^2 + a_2xy + a_3x + a_4y + a_5, \quad a_0, \dots, a_5 \in \mathbb{R}. \quad (2.11)$$

The equation 2.11 can be expressed as a scalar product of vectors,

$$f(x, y) = \mathbf{a} \cdot \mathbf{b}(x, y), \quad \mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5), \quad \mathbf{b}(x, y) = (x^2, y^2, xy, x, y, 1). \quad (2.12)$$

A new Cartesian coordinate system is defined so that its origin coincides with the center of gravity \mathbf{g}_j of triangle T_j and the mean normal vector $\bar{\mathbf{n}}_j$ (2.10) is taken as the z -axis. In the rest of this section, coordinates (x, y, z) are assumed to be expressed in the new frame.

Let $G = \{\mathbf{g}_i = (g_{xi}, g_{yi}, g_{zi}) \mid i \in \Lambda_j\}$ be the set of centers of gravity of the triangles adjacent to T_j . A linear regression procedure is applied in order to determine the vector \mathbf{a} (2.12). The optimal fit of the quadratic surface (2.12) through the points \mathbf{g}_i ($i \in \Lambda_j$), which minimizes the mean square error, is secured if the vector \mathbf{a} is computed by

$$\mathbf{a} = \frac{\sum_{i \in \Lambda_j} g_{zi} \mathbf{b}(g_{xi}, g_{yi})}{\sum_{i \in \Lambda_j} \mathbf{b}(g_{xi}, g_{yi}) \cdot \mathbf{b}(g_{xi}, g_{yi})}. \quad (2.13)$$

After determining the parametric surface approximation, the principle curvatures k'_j and k''_j are computed using the shape operator (also known as the Weingarten map or the second fundamental tensor) [148]. Namely, k'_j and k''_j are eigenvalues of the shape operator W defined by

$$W = I^{-1}II, \quad I = \begin{bmatrix} 1 + f_x^2 & f_x f_y \\ f_x f_y & 1 + f_y^2 \end{bmatrix}, \quad II = \frac{1}{\sqrt{1 + f_x^2 + f_y^2}} \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}, \quad (2.14)$$

where I and II denote the first and the second fundamental differential form [148]. We use the standard Monge notation for partial derivatives, i.e.,

$$f_x = \frac{\partial f}{\partial x}, \quad f_y = \frac{\partial f}{\partial y}, \quad f_{xx} = \frac{\partial^2 f}{\partial x^2}, \quad f_{yy} = \frac{\partial^2 f}{\partial y^2}, \quad \text{and} \quad f_{xy} = \frac{\partial^2 f}{\partial x \partial y}.$$

Hence, by combining (2.12), (2.13), and (2.14), we compute k'_j and k''_j . Finally, a surface is declared as planar if the following condition is fulfilled

$$\sigma_j \sqrt{(k'_j)^2 + (k''_j)^2} < t,$$

where t is a given threshold (recommended $0.1 \leq t \leq 0.4$) and σ_j is the total area of local surfaces defined by (2.9).

After all triangles T_j of the mesh are processed, the shape spectrum descriptor \mathbf{f} is composed from a histogram of curvature indices Γ_j (2.8) and two additional components describing the proportion of planar and singular surfaces $f(x, y)$ (2.11). If N is the number of histogram bins, then the dimension of the feature vector \mathbf{f} is $\dim = N + 2$. The exact algorithm for populating feature vector components is given by the following pseudocode.

```

dim = N + 2; // N - the number of histogram bins
f = (h1, ..., hN, planar, singular);
Initialize: h1 = ... = hN = planar = singular = 0;
for j = 1, ..., m // m - the number of triangles (1.2)
    Find  $\Lambda_j$  and compute  $\sigma_j$  (2.9);
    if  $\sigma_j$  is a singular surface
        singular  $\leftarrow$  singular +  $\sigma_j/S$ ; // S - the total surface area of the mesh
    else
        Compute  $k'_j$ ,  $k''_j$ , and  $\Gamma_j$ ;
        if  $(\sigma_j \sqrt{(k'_j)^2 + (k''_j)^2} < t)$  // e.g.,  $t = 0.25$ 
            planar  $\leftarrow$  planar +  $\sigma_j/S$ ;
        else
            i =  $\lceil \Gamma_j N \rceil$ ; //  $\lceil \cdot \rceil$  is the ceiling function
            hi  $\leftarrow$  hi +  $\sigma_j/S$ ;

```

The MPEG-7 eXperimentation model [84] includes a C++ source code that we use for generating the shape spectrum descriptor. On average, a descriptor is

extracted for 1.92s on a PC with an 1.4 GHz AMD processor running Windows 2000, using the models from the MPEG-7 collection (section 5.1). The extraction time does not depend on the dimension of the feature vector, but on the complexity of models. We also tested the extraction when adjacent triangles of the second order are used for forming the set Λ_j (2.9). In this case, the average extraction time amounts 5.37s. However, our tests confirm the claim from [88] that the retrieval performance is not better if adjacent triangles of the second order are considered.

The descriptor is particularly effective in the case of articulated modifications of models (e.g., different poses of a model of human), while its major drawback is a high sensitivity with respect to the levels of detail or different tessellations of an object. Also, there is a requirement for meshes to be orientable. An inconsistent orientation of triangles of a mesh deteriorates the discriminant power of the shape spectrum descriptor.

In [88], it is stated that a pre-processing step consisting of a mesh subdivision algorithm (up to 2 subdivision levels) could strongly increase the accuracy of the curvature estimation. However, our experience suggests that the increase of the accuracy of the curvature estimation will not lead to the increase of retrieval performance, because the feature vector will still be sensitive to different tessellations and inconsistent orientation. Our evaluation results (section 5.2.12) show that the shape spectrum descriptor is generally the most inferior descriptor, which is tested in this thesis.

2.4 Topology Matching

An interesting and sophisticated technique, called topology matching, is presented in [48]. The technique relies upon matching graph representations of 3D-objects. A novel graph structure, *multiresolutional Reeb graph* (MRG), is introduced and used for representing 3D-mesh models. The graph representation of a mesh model is regarded as a descriptor. The topology matching technique can be subdivided into two phases:

1. Generation of graph representation (descriptor), and
2. Computation of similarity between two graphs.

The method does not require any pose normalization step, because the invariance with respect to similarity transforms is provided by the definition of descriptor. No restrictions regarding closedness or orientability of the mesh models are imposed.

The main idea of the approach is to represent topology information by the MRG. The MRG is generated using a suitable function $\mu(\mathbf{u})$, which approximates integral of geodesic distances between a point $\mathbf{u} \in I$ on the surface of 3D-model to all other points of the point set I (1.5), defined by

$$\mu(\mathbf{u}) = \int_{\mathbf{p} \in I} g(\mathbf{u}, \mathbf{p}) dv, \quad (2.15)$$

where $g(\mathbf{u}, \mathbf{p})$ denotes the geodesic distance between the point \mathbf{u} and the point \mathbf{p} on the surface of model I . Obviously, the value of $\mu(\mathbf{u})$ depends on the scale of the object. To avoid the scale dependency, the function μ is normalized as

$$\mu_n(\mathbf{u}) = \frac{\mu(\mathbf{u}) - \min_{\mathbf{p}} \mu(\mathbf{p})}{\max_{\mathbf{p}} \mu(\mathbf{p})}. \quad (2.16)$$

In order to balance the trade-off between the computational costs and the quality of the approximation of μ , certain pre-processing steps, aimed at reorganizing topology and geometry of the mesh, are performed. The feature extraction algorithm can be summarized into the following steps:

- Resampling of a triangle mesh model;
- Generation of short-cut edges;
- Selection of base vertices;
- Computation of geodesic distances;
- Construction of the multiresolutional Reeb graph;
- Representation of the multiresolutional Reeb graph.

Since a value of the function μ_n is assigned to each vertex of the mesh, the distribution of the vertices should be fine enough and as uniform as possible. Therefore, it is usually necessary to resample the triangles (figure 2.4) until all edge lengths are less than a threshold τ . The authors do not specify how to determine the threshold.

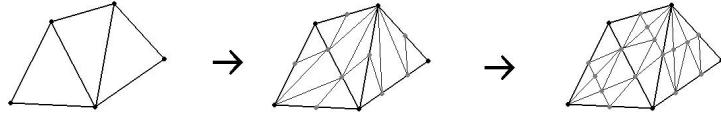


Figure 2.4: Resampling until all edge lengths are less than a threshold τ .

The computation of the geodesic distance between two vertices is approximated using the set of all edges E (1.7) of the mesh model. Since the triangle inequality may deteriorate the quality of approximation, the authors recommend to add short-cut edges to the set E . The generation of the short-cut edges is depicted in figure 2.5. A triangle and three neighboring triangles are unfolded first, i.e., each neighboring triangle is rotated around the common edge with the triangle in the middle so that all triangles are coplanar. A diagonal of the unfolded polygon is added to the set of edges E , if it is completely inside the polygon. In figure 2.5, the line connecting two vertices denoted by squares is not considered as a short-cut edge, because it is not inside the unfolded polygon. The lengths of the generated edges are the Euclidean distances between the corresponding vertices of the unfolded polygon. Thus, a short-cut edge is weighted by the geodesic distance between its points.

After the resampling and generation of short-cut edges, the set of vertices V along with the set of edges E are treated as a weighted graph (V, E) . A weight

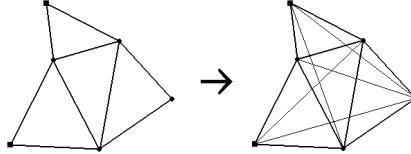


Figure 2.5: Generation of short-cut edges.

associated to an edge is the geodesic distance between the end points of the edge. Therefore, the problem of finding the geodesic distance between two vertices of the mesh can be interpreted as the problem of finding the shortest path from a point in the graph (the source) to a destination. An efficient algorithm, proposed by Dijkstra [24], can be used for computing simultaneously shortest paths from a fixed point (node) \mathbf{b}_i to all other nodes of the graph (see the algorithm in figure 2.6).

```

 $\mathbf{b}_i$  – is the base vertex (source node);
 $V = \{\mathbf{v}_1, \dots, \mathbf{v}_m\}$  – is the set of vertices;
 $LIST$  – is an ordered binary tree;
Initialize  $g(\mathbf{b}_i, \mathbf{v}_j) = +\infty$ , for all vertices  $\mathbf{v}_j \in V$ ;
Set  $g(\mathbf{b}_i, \mathbf{b}_i) = 0$  and insert  $\mathbf{b}_i$  to  $LIST$ ;
while ( $LIST$  is not empty)
    Take  $\mathbf{v} \in LIST$  so that  $g(\mathbf{b}_i, \mathbf{v}) = \min\{g(\mathbf{b}_i, \mathbf{v}_j) \mid \mathbf{v}_j \in LIST\}$ ;
    Remove  $\mathbf{v}$  from  $LIST$ ;
    for each vertex  $\mathbf{v}_a$  adjacent to  $\mathbf{v}$ 
        if  $g(\mathbf{b}_i, \mathbf{v}_a) > g(\mathbf{b}_i, \mathbf{v}) + weight(\mathbf{v}, \mathbf{v}_a)$ ;
             $g(\mathbf{b}_i, \mathbf{v}_a) = g(\mathbf{b}_i, \mathbf{v}) + weight(\mathbf{v}, \mathbf{v}_a)$ ;
            Insert  $\mathbf{v}_a$  to  $LIST$ ; (*)
```

Figure 2.6: Dijkstra's algorithm for finding the shortest path from a base vertex (source node) \mathbf{b}_i to all other nodes (vertices) $\mathbf{v}_1, \dots, \mathbf{v}_m$.

The set of base vertices $B = \{\mathbf{b}_1, \dots, \mathbf{b}_N\}$, which are scattered almost equally on the surface of the model, is selected using a modification of the Dijkstra's algorithm. Firstly, an arbitrary vertex is selected as \mathbf{b}_1 and taken as the base point of the Dijkstra's algorithm. The original Dijkstra's algorithm (figure 2.6) is modified as follows:

- Step (*) is changed to:
 - if $g(\mathbf{b}_i, \mathbf{v}_a) < t_r$, then insert \mathbf{v}_a to $LIST$;
 - where t_r is a threshold.
- If the $LIST$ is empty, then an arbitrary unvisited vertex is selected as a new base vertex \mathbf{b}_{i+1} (\mathbf{b}_i is the current base vertex), inserted in $LIST$, set $g(\mathbf{b}_{i+1}, \mathbf{b}_{i+1}) = 0$, and the while loop is repeated.

The number of base vertices, N , depends on the threshold t_r . It is recommended to set $t_r = \sqrt{0.005S}$, whereby the average number of base vertices N is around 150. We recall that S denotes the surface area of the mesh model (1.6).

After the set B is selected, the original Dijkstra's algorithm is applied for each base vertex $\mathbf{b}_i \in B$ (N times in total). As a result, all the values $g(\mathbf{b}_i, \mathbf{v}_j)$, $\mathbf{b}_i \in B$, $\mathbf{v}_j \in V$, are determined. The integral (2.15) is approximated by

$$\mu(\mathbf{v}_j) = \sum_{i=1}^N g(\mathbf{b}_i, \mathbf{v}_j) \cdot \sigma_i, \quad (2.17)$$

where σ_i is the sum of areas of faces composed of vertices whose distances from \mathbf{b}_i are less than t_r . It holds, $\sum_{i=1}^N \sigma_i = S$. The specification how to select areas in order to compute the value of σ_i is not given. In particular, we consider that the handling of special cases is important. The special cases include:

- Vertices of triangle T_j are closest to different base vertices;
- A vertex \mathbf{v} satisfies $g(\mathbf{b}_i, \mathbf{v}) < t_r$ and $g(\mathbf{b}_j, \mathbf{v}) < t_r$.

Typically, for $t_r = \sqrt{0.005S}$ around $N = 150$ base vertices are selected, and the approximation (2.17) achieves sufficient accuracy.

After computing values of the function μ_n (2.16) at all vertices $\mathbf{v}_1, \dots, \mathbf{v}_m$, the multiresolutional Reeb graph is generated. Construction of the MRG is depicted using the example in figure 2.7. For simplicity, a height function is used as the function μ_n on a 2D-triangle mesh. In figure 2.7(a), an example 2D-triangle mesh is visualized and four ranges of values of the function μ_n (briefly, μ_n -ranges) are marked. If a triangle is spread across two or more μ_n -ranges, then a subdivision is necessary (figure 2.7(b)). For instance, if \mathbf{v}_1 , \mathbf{v}_2 , and \mathbf{v}_3 are vertices of a triangle and $0.25 < \mu_n(\mathbf{v}_1) < 0.50 < \mu_n(\mathbf{v}_2) < 0.75$, then a new vertex \mathbf{v} is inserted so that

$$\mathbf{v} = \frac{(\mu_n(\mathbf{v}_2) - \mu_n(\mathbf{v}))\mathbf{v}_2 + (\mu_n(\mathbf{v}) - \mu_n(\mathbf{v}_1))\mathbf{v}_1}{\mu(\mathbf{v}_2) - \mu(\mathbf{v}_1)},$$

where $\mu_n(\mathbf{v}) = 0.50$. The triangle $\Delta\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$ is subdivided into triangles $\Delta\mathbf{v}_1\mathbf{v}\mathbf{v}_3$ and $\Delta\mathbf{v}\mathbf{v}_2\mathbf{v}_3$. This process is repeated until each edge lies in one μ_n -range. Then, the nodes of the MRG are identified (bold polygonal lines in figure 2.7(c)). The nodes and edges of the MRG at the finest resolution (level 2) are visualized in figure 2.7(d), while the other two resolutions are shown in figure 2.7(e-f). Note that there is a parent-child relationship between nodes at different levels, e.g., n_6 is parent of n_0 , n_1 , and n_2 , etc.

Finally, certain attributes of all nodes from the MRG are computed and stored together with both the connectivity and parental information. The stored representation of the MRG is regarded as a 3D-shape descriptor. As a first approach, the authors suggested to use a pair of values $(a(n_i), l(n_i))$ to attribute the node n_i . The value of $a(n_i)$ is the quotient of the area of triangles related to n_i and the total area S , while $l(n_i)$ is the ratio of the length $len(n_i)$ of the node n_i to the sum of lengths $\sum_j len(n_j)$, where n_j are the nodes at the finest level. The length of a node is defined in [48]. The recommended number of μ_n -ranges is 64, i.e., an MRG with 7 resolution levels ($2^6 = 64$) is tested in [48].

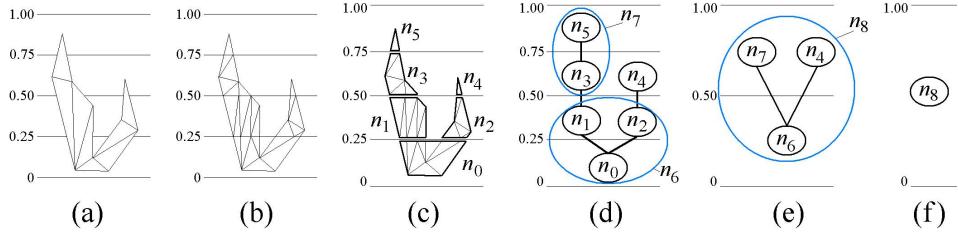


Figure 2.7: Construction of a multiresolutional Reeb graph using four ranges of function values. Visualizations include: (a) original mesh, (b) subdivision of triangles spread over two or more ranges, (c) identification of nodes, (d) the finest level of the MRG – level 2, (e) MRG level 1, (e) the coarsest level of the MRG – level 0.

An original similarity measure between two descriptors (MRGs with attributes) is used in the search procedure. The matching procedure, which is based on a coarse-to-fine strategy, starts with finding pairs of nodes whose similarity can be computed. The similarity computation starts at the coarsest level. A rule is that nodes are matching candidates only if they have the same μ_n -range. An additional condition for matching candidates, which are not at the finest level, is that their parents are matched. There are other rules concerning propagation of matched nodes along branches. For more details, we refer to [48].

One of the drawbacks of the topology matching approach is the sensitivity of MRG construction at boundaries of μ_n -ranges. The problem is depicted in figure 2.8. Two almost identical objects have different MRGs, because of small differences at the boundaries of μ_n -ranges. The authors proposed to allow simultaneous matching of sets of nodes, if all the nodes from a set are adjacent to the same node. For example, the node n'_1 can match n_1 and n_2 simultaneously, because n_1 and n_2 are adjacent to the same node n_0 . In this case, the node n'_1 have three matching candidates n_1 , n_2 , and $n_1 + n_2$, where the sign + denotes the addition of attributes.

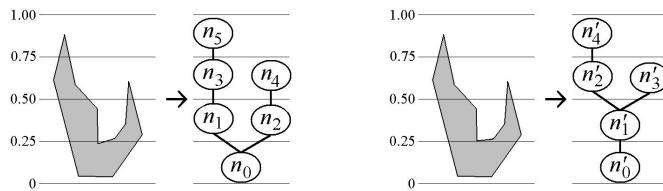


Figure 2.8: Sensitivity at boundaries of μ_n -ranges. Small differences may significantly disturb the MRGs.

In the results presented in [48], the average feature extraction time, i.e., the average time for constructing a MRG, is around 15s, for a triangle mesh of 10000

vertices. The results are obtained on a PC with an 400 MHz Pentium II processor running Linux. For $64 \mu_n$ -ranges, the average number of nodes in the MRG is around 300. The average matching between two MRGs is approximately 0.05s, which the authors regard as a quick procedure. We consider that the extraction procedure takes reasonable amount of time, because in a retrieval system features can be extracted at idle time. However, we disagree with the authors regarding the quickness of the matching procedure. For a given query, if the matching procedure has to be performed 1000 times, then the retrieval system responds after 50 seconds. We consider this amount of time to be unacceptable for interactive applications.

We also feel that the effectiveness of the topology matching breaks down when models are complex (geometrically and topologically). We expect more frequent occurrences of the problem depicted in figure 2.8 as well as unexpected matching pairs during the similarity calculation procedure. Our assumptions comply with remarks presented in [11].

2.5 Shape Distributions, Reflective Symmetry, and Descriptors Based on Concentric Spheres

In this section, we describe four 3D-shape descriptors introduced in [97, 98, 36, 57, 55, 56, 58]. The first descriptor is based on a technique called shape distributions and is essentially a statistical approach [97, 98]. A descriptor based on binary voxel grids [36, 57], in which the fast Fourier transform is applied to functions on concentric spheres, utilizes a property of spherical harmonics (see property 4.1 in section 4.6.1). We stress that only the use of the property of spherical harmonics is introduced in [36, 57] in order to secure rotation invariance, while spherical harmonics as a tool for 3D-model retrieval are introduced in [147]. Reflective symmetry descriptor [55, 56] relies on the computation of a symmetry measure using a negatively exponentiated Euclidean distance transform. By replacing the binary voxel grid with the negatively exponentiated Euclidean distance field and using the approach presented in [36, 57], a more powerful descriptor is created in [58]. A common characteristic of all four descriptors is that the Principal Component Analysis (PCA) (section 3.4) is not used for securing rotation invariance. Two approaches for achieving rotation invariance of 3D-shape descriptors, the PCA vs. the property of spherical harmonics, are compared in section 5.2.13. We believe that it is better to use the Continuous PCA (section 3.4) than the property of spherical harmonics to accomplish rotation invariance.

2.5.1 Shape Distributions

The shape distributions approach, which is presented in [97, 98], does not require any normalization of a 3D-mesh model. The main idea is to randomly select points on the surface of a 3D-object, compute certain geometric property, and create a histogram of obtained values. Thus, the approach is purely statistical.

A point \mathbf{v} on the surface of a 3D-mesh, $\mathbf{v} \in I$ (1.5), is randomly selected using the following procedure. A vector $\sigma = (\sigma_1, \dots, \sigma_m)$ of cumulative surface areas is created, where the elements σ_j are defined by

$$\sigma_j = \sum_{i=1}^j S_i, \quad (S_i \text{ is the surface area of triangle } T_i). \quad (2.18)$$

Then, a random value $\rho \in [0, S]$ (1.6) is generated to select the triangle T_j by performing a binary search on the array σ . Namely, the index j of the selected triangle is defined as

$$j = \min\{i \mid \rho \leq \sigma_i\}.$$

Note that the probability of selecting the triangle T_j is proportional to its area S_j .

The selection of a uniform random point $\mathbf{v} \in T_j$ with respect to surface area is done by generating two random numbers $r_1 \in [0, 1]$ and $r_2 \in [0, 1]$ and computing \mathbf{v} by

$$\mathbf{v} = (1 - \sqrt{r_1})\mathbf{p}_{A_j} + \sqrt{r_1}(1 - r_2)\mathbf{p}_{B_j} + \sqrt{r_1}r_2\mathbf{p}_{C_j}. \quad (2.19)$$

The following geometric properties (functions), which are based on random points on the surface of a 3D-object, are considered:

- **A3** – the angle between three random points;
- **D1** – the distance between the center of gravity of the model (3.11) and one random point;
- **D2** – the distance between two random points;
- **D3** – the square root of the area of the triangle determined by three random points;
- **D4** – the cube root of the volume of the tetrahedron determined by four random points.

A chosen function is randomly sampled N times. The sample values $\{s_1, \dots, s_N\}$ are used to construct a histogram \mathbf{h} of $B \in \mathbb{N}$ bins, defined by

$$\mathbf{h} = (h_1, \dots, h_B), \quad h_i = \begin{cases} \left| \left\{ s_j \mid s_j \in \left[(i-1)\frac{3\bar{s}}{B}, i\frac{3\bar{s}}{B} \right) \right\} \right|, & 1 \leq i \leq B-1, \\ \left| \left\{ s_j \mid s_j \in \left[(B-1)\frac{3\bar{s}}{B}, +\infty \right) \right\} \right|, & i = B, \end{cases} \quad (2.20)$$

where $|\{\dots\}|$ denotes the cardinality of the underlying set, and $\bar{s} = (s_1 + \dots + s_N)/N$ is the mean value of samples $s_i \geq 0$. Thus, the value of h_i is the count of samples belonging to a given range of values. Note that scaling invariance is provided by defining the ranges of values using the mean value \bar{s} .

A piecewise linear function g with $\dim \leq B$ equally spaced vertices is constructed from the histogram \mathbf{h} . There are several possibilities to determine the ordinate values of the vertices g_i , $1 \leq i \leq \dim$. The approach used by the authors [35] is the following

$$g_i = h_k, \quad k = \left\lceil i \frac{B}{\dim} \right\rceil, \quad (2.21)$$

where $\lceil \cdot \rceil$ is the ceiling function. The components of the feature vector based on shape distributions, $\mathbf{f} = (f_1, \dots, f_{dim})$, are defined by

$$f_i = \left\lceil \frac{10^6}{\Sigma} g_i \right\rceil, \quad \Sigma = g_1 + \dots + g_{dim} \quad (2.22)$$

where $\lfloor \cdot \rfloor$ denotes the rounded value. Thus, the dimension of the feature vector \mathbf{f} is determined by the number of vertices of the piecewise linear function g . The factor $10^6/\Sigma$ normalizes the values of f_i so that the vector components do not depend on the number of samples N .

The authors suggest that the best choice of the geometric function is D2, while $N = 1024^2$ samples, and $dim = 64$ are recommended parameter settings. The number of bins B is tested for values 1024 and 64 ($B = dim$). The feature extraction procedure for the case $B = dim$, when the D2 function is used, is summarized in the algorithm shown in figure 2.9. Note that $B = dim \Rightarrow \Sigma = N$.

```

N = 10242; // the number of samples
dim = 64; // dimension of the feature vector
f = (f1, ..., fdim); // the feature vector
Initialization: fi = 0 (i = 1, ..., dim);
s̄ ← 0;
for i = 1, ..., N
    Select random points v1 ∈ I and v2 ∈ I (2.19);
    si = ||v1 - v2||; // Euclidean distance between 3D points
    s̄ ← s̄ + si;
    s̄ ← s̄/N;
for i = 1, ..., N // form histogram
    j = min{⌈sidim/(3s̄)⌉, dim}; fj ← fj + 1;
    for i = 1, ..., dim // normalize values
        fi ← ⌈(106/N)fi⌉;

```

Figure 2.9: Feature extraction algorithm of the shape distribution descriptor, for the case $B = dim$, when the D2 function is used.

The average extraction time of the shape distribution descriptor is around 1.12s, when the algorithm from figure 2.9 is used for generating descriptors of the models from the MPEG-7 collection (5.1), on a PC with an 1.4 GHz AMD processor running Windows 2000.

This approach satisfies all requirements listed in section 1.3.4. In spite of satisfying the desirable requirements, the discriminant power of the shape distribution descriptor is poor (see section 5.2.12 as well as [36]). In general, purely statistical shape feature vectors do not have sufficient discriminant power, because components of the vector are not related to a specific spatial region. The results and examples given in [97, 98] are based on a very small collection of only 133 3D-models. Some examples show a good discrimination between shape distributions

for cubes, spheres, cylinders, and other geometric primitives. However, the situation is totally different when dealing with complex 3D-meshes, i.e., the retrieval performance of descriptors based on shape distributions is not good enough.

2.5.2 Descriptor Based on Binary Voxel Grids

A 3D-shape descriptor based on a binary voxel grid is proposed in [36, 57]. The descriptor is defined by sampling functions on concentric spheres. The sample values are determined using the binary voxel grid. The sample points of functions on concentric spheres are chosen so that the fast Fourier transform on the sphere (see section 4.6.1) can be applied. In the normalization step, a 3D-model is translated and scaled only, while the invariance with respect to rotation and reflection is secured by relying on a property of spherical harmonics (see property 4.1). Spherical harmonics as a tool for 3D-model retrieval are introduced in [147]. More details about the generation of the descriptor are given in a sequel.

Firstly, a model I (1.5) is translated so that the center of gravity \mathbf{m}_I (3.11) lies at the coordinate origin and the model is scaled by dividing by the average distance d_{avg} (3.25) of a point on the surface of the model to the center of gravity. Hence, a set I' is formed,

$$I' = \{\mathbf{v}' \mid \mathbf{v}' = (\mathbf{v} - \mathbf{m}_I)/d_{avg}, \mathbf{v} \in I\}. \quad (2.23)$$

Next, the cubic region $-2 \leq x, y, z \leq 2$ in the 3D-space \mathbb{R}^3 is rasterized into a $2R \times 2R \times 2R$ voxel grid in the discrete space \mathbb{Z}^3 . A voxel cell $\eta_{abc} \subset \mathbb{R}^3$ ($a, b, c \in \{1, \dots, 2R\}$) corresponds to the cuboid region

$$\eta_{abc} = \left\{ (x, y, z) \mid a = \left\lfloor \frac{x+2}{2} R \right\rfloor + 1, b = \left\lfloor \frac{y+2}{2} R \right\rfloor + 1, c = \left\lfloor \frac{z+2}{2} R \right\rfloor + 1 \right\}. \quad (2.24)$$

The voxel cell η_{abc} is attributed a binary value v_{abc} , defined by

$$v_{abc} = \begin{cases} 0, & I' \cap \eta_{abc} = \emptyset, \\ 1, & I' \cap \eta_{abc} \neq \emptyset. \end{cases} \quad (2.25)$$

Note that some points of the point set I' lay outside the region of voxelization. Those points are ignored as outliers. The binary voxel grid of a model of an airplane is shown in figure 2.10.

Then, a function $f_r(\theta, \varphi)$ ($\theta \in [0, \pi]$, $\varphi \in [0, 2\pi]$) on the sphere Ω_r with the center at the origin and radius r , is defined using (2.24) and (2.25)

$$f_r(\theta, \varphi) = v_{abc}, \quad \text{whereby } \eta_{abc} \ni (r \sin \theta \cos \varphi, r \cos \theta, r \sin \theta \sin \varphi). \quad (2.26)$$

Note that $f_r(\theta, \varphi)$ is a binary function [36].

Totally R functions on concentric spheres $\Omega_1, \dots, \Omega_R$ with radii $r = 1, \dots, R$ are sampled at $4B^2$ sample points $\mathbf{u}'_{ab} = (x_{ab}, z_{ab}, y_{ab})$ ($0 \leq a, b \leq 2B - 1$), where x_{ab} , y_{ab} , and z_{ab} are defined by (4.63). In [57], it is recommended to set $R = 32$.

In our implementation of this approach, we use $B = 64$, i.e., we have 16384 sample values $f_{r,a,b}$ for each function $f_r(\theta, \varphi)$. The samples obtained using the binary voxel grid are visualized on the right side of figure 2.10. In this visualization, neighboring samples whose values are equal 1 are treated as vertices of triangles. Samples whose values are 0 are not visualized. Functions on successive concentric spheres are differently colored.

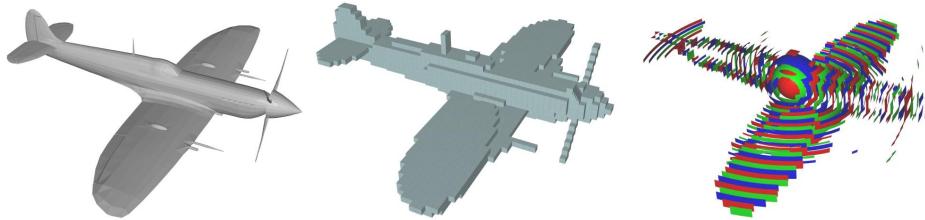


Figure 2.10: Approach based on a binary voxel grid: a 3D model (left) is represented by a binary voxel grid (middle), which is used to define binary functions on concentric spheres (right).

For each function f_r (2.26), the fast Fourier transform on the sphere Ω_r (see section 4.6.1) is applied to the sample values $f_{r,a,b} \in \{0, 1\}$. As a result, we obtain B^2 complex coefficients $\hat{f}_{r,l,m} \in \mathbb{C}$ ($0 \leq |m| \leq l \leq B$). A signature \mathbf{s}_r of the function f_r is defined using the property 4.1, in order to attain invariance of the signature with respect to rotation of the underlying object (see remark 4.1). The signature is generated using the first L bands of spherical harmonics, i.e.,

$$\mathbf{s}_r = (||f_{r,0}||, \dots, ||f_{r,L-1}||), \quad \text{where } ||f_{r,l}|| = \sqrt{\sum_{m=-l}^l |\hat{f}_{r,l,m}|^2}. \quad (2.27)$$

The feature vector \mathbf{f} based on a binary voxel grid is formed by concatenating the signatures of spherical functions,

$$\mathbf{f} = (\mathbf{s}_1 | \mathbf{s}_2 | \dots | \mathbf{s}_R) = (||f_{1,0}||, \dots, ||f_{1,L-1}||, \dots, ||f_{R,0}||, \dots, ||f_{R,L-1}||). \quad (2.28)$$

Thus, the dimension of the feature vector is $\dim = R \cdot L$. In [57], it is recommended to set $R = 32$ and $L = 16$, i.e., $\dim = 512$. Hence, a rough representation of a 3D model by a $64 \times 64 \times 64$ binary grid is used for defining 32 functions on concentric spheres, while 16 bands of spherical harmonics are used in forming the feature vector.

In [36], the authors state that the motive for using the binary voxel grid is to achieve a better robustness with respect to variances of the polygonal surface. However, we feel that many fine details are lost in a rough binary voxel grid. We expect that a better choice is to have many samples of a model (i.e., of spherical functions), and let the Fourier transform reduce the variance (high-frequency noise),

than filtering the noise by voxelizing the model. The results presented in [142], where the descriptor based on a binary voxel grid is compared to a descriptor formed using a ray-casting technique (see section 4.6.6), comply with our assumptions.

The discriminant power of the descriptor based on a binary voxel grid is limited because of the following reasons:

- If the resolution of voxelization R is low, than the representation of the model by a binary voxel grid is too rough (coarse);
- If we increase R , then the functions on concentric spheres, corresponding to similar models whose scaling factors are slightly different, can be mismatched.

The problem of mismatched binary functions is illustrated in figure 2.11. For simplicity, a 2D case is depicted, i.e., instead of voxels we have square fields and instead of functions on spheres we have functions on circles. At the resolution R , functions on the circle (analogous to (2.26)) of both objects are identical. However, at the resolution $2R$ the functions are almost complementary. The depicted example represents an extreme situation, which rarely happens. Nevertheless, a similar mismatching is present, because the method relies upon the center of gravity of a 3D-mesh model (center of concentric spheres) and the average distance of a point on the surface to the center of gravity (scaling factor). We anticipate that similar objects may have slightly different scaling factors so that the function $f_r^{(a)}$ of the model I_a matches $f_{r\pm 1}^{(b)}$ of the model I_b .

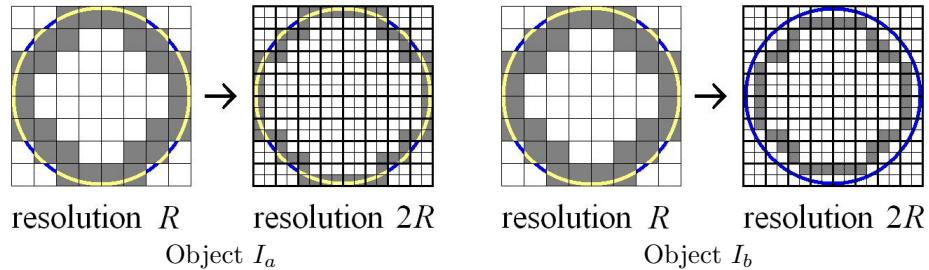


Figure 2.11: At the resolution R , the functions on the circle are identical for both objects, while at the resolution $2R$, the functions are almost complementary. Darker parts of the circles denote the value of 0, while brighter parts denote the value of 1.

We object to the choice to use binary functions f_r (2.26) for characterizing binary voxel grids, because the number of voxels whose attributes are used to determine the sample values of f_r increases with the increase of r . Thus, only 8 voxel attributes determine all sample values of f_1 , while 10496 voxels (according to our implementation) determine samples of f_{32} . Therefore, we believe that it is meaningful to specify the “importance” of the function f_r . A possible way to modify (2.26) is just to multiply the voxel attribute by r ,

$$f_r(\theta, \varphi) = r \cdot v_{abc}, \quad \text{whereby } \eta_{abc} \ni (r \sin \theta \cos \varphi, r \cos \theta, r \sin \theta \sin \varphi). \quad (2.29)$$

The experimental results (section 5.2.12) show that (2.29) is significantly better choice of defining f_r than (2.26).

2.5.3 Reflective Symmetry Descriptor

A descriptor based on calculation of a symmetry measure, with respect to a plane cutting a 3D-model, is presented in [55, 56]. The model is cut by many planes passing through the center of gravity of the model. For each plane, a symmetry between parts of the model, being on the opposite sides of the plane, is estimated. The symmetry estimation relies upon a suitable representation of the object by a voxel grid, whose voxels are attributed values computed by a negatively exponentiated Euclidean distance transform.

Hence, a 3D-model is voxelized first. The voxelization starts with generating the binary voxel grid $V = \{v_{abc} \mid 1 \leq a, b, c \leq 2R, v_{abc} \in \{0, 1\}\}$ (2.25), which is described in the previous subsection. The Euclidean distance transform is applied to the 3D-array V , and new attributes v'_{abc} satisfy the condition

$$v'_{abc} = \min \{(a - x)^2 + (b - y)^2 + (c - z)^2 \mid v_{xyz} = 1, 1 \leq x, y, z \leq 2R\}, \quad (2.30)$$

Thus, the value of the attribute v'_{abc} represents the square of the shortest distance to a voxel whose attribute is 1. This can also be interpreted as an approximate squared distance from a point $\mathbf{x} \in \eta_{abc}$ (2.24) to the nearest point on the model I' (1.5). The Euclidean distance transform is computed using the algorithm proposed by Saito and Toriwaki [115], whose computational complexity is $\mathcal{O}(N^3)$ for an $N \times N \times N$ voxel grid. Finally, the attributes v''_{abc} are computed using a negatively exponentiated distance transform,

$$v''_{abc} = \exp \left(\frac{-v'_{abc}}{r_I^2} \right), \quad (2.31)$$

where r_I is the average distance from a point on the surface of a model I to the center of gravity of the mesh model (3.11). Having in mind the normalization of translation and scale (2.23), we observe that $r_I = 1$ for the point set I' .

A point \mathbf{n}_i on a sphere of radius 1, with the center at the origin, is considered as the normal vector of the plane containing the origin. Let π_i be the plane containing the origin and having \mathbf{n}_i as the normal vector. The plane π_i cuts the underlying model into two parts. An approach for measuring the symmetry distance, between parts of the model lying on the opposite sides of the cutting plane, is presented in [55, 56]. The method relies on the voxelized model, whereby the attributes defined by (2.31) are used. The problem of computing the symmetry measure of a voxel grid is transformed into a problem of computing the similarity measure of a collection of functions defined on concentric spheres. The computational complexity of the algorithm is $\mathcal{O}(N^4 \log_2 N)$, for an $N \times N \times N$ voxel grid. Since the method relies on reflection of the model with respect to π_i , the authors called the technique *reflective symmetry descriptor*.

Let $\sigma_i \in \mathbb{R}$ be the symmetry measure between parts of a model lying on the opposite sides of π_i . Since all parts of the 3D-model are used to computes the

value of σ_i , the reflective symmetry distance σ_i is a global feature. The reflective symmetry feature vector \mathbf{f} is formed by treating σ_i as components, i.e.,

$$\mathbf{f} = (\sigma_1, \dots, \sigma_{dim}). \quad (2.32)$$

The authors do not specify how the points (i.e., normal vectors) \mathbf{n}_i on the unit sphere are chosen. Also, the vector dimension dim is not suggested. We assume that the normal vectors are uniformly distributed across the unit sphere.

According to [56], for an $64 \times 64 \times 64$ voxel grid ($R = 32$), the voxelization (2.25) takes between 0.2 and 3.5 seconds, the computation of the exponentially decaying distances (2.31) takes 1.2 seconds, and the estimation of reflective symmetry distances takes 5 seconds. Hence, the feature extraction lasts between 6.4 and 9.7 seconds. The results are obtained on an 800 MHz Athlon processor with 512 MB of RAM.

The reflective symmetry descriptor is invariant with respect to translation, scaling, and reflection, while the invariance with respect to rotation is not provided. The evaluation results in [55, 56] are based on a 3D-model collection of manually oriented objects. In practice, such a 3D-shape descriptor, which is highly sensitive even to rotation of a mesh model for a very small angle around a coordinate axis, cannot be useful. Hence, the importance of the approach presented in [55, 56] is mainly theoretical.

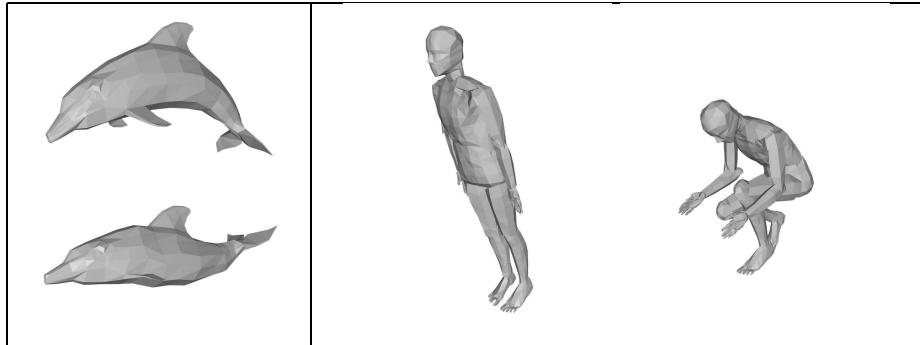


Figure 2.12: Models that are considered similar may have significantly different components of the reflective symmetry feature vector.

The authors stress that because of the global nature of the reflective symmetry distance, a large difference in the values for a single component of the feature vector (2.32) “provides a provable indication that two models are significantly different”. However, we disagree with this statement because of two reasons. First, since rotation invariance is not provided, models should be perfectly aligned (oriented). Obviously, if a model is rotated for an angle around an arbitrary axis, then the reflective symmetry descriptor will be significantly different. Second, a counter-example to the claim is depicted in figure 2.12. Namely, we have models that are

considered relevant (similar) to each other, which are articulately modified. Since there is no perfect alignment for these models, a number of components of the corresponding reflective symmetry feature vectors of similar models will always be significantly different.

Limitations of retrieval effectiveness of the reflective symmetry descriptor are also commented in [56]. Instead of 3D-model retrieval, registration is considered as another possible usage of the presented technique.

2.5.4 Descriptor Based on Exponentially Decaying EDT

A very effective 3D-shape descriptor, which we call descriptor based on exponentially decaying Euclidean distance transform (EDT), is defined in [58] by merging the techniques presented in subsections 2.5.2 and 2.5.3. Instead of using a binary voxel grid to define functions on concentric spheres by (2.26), the following modification is used [54],

$$f_r(\theta, \varphi) = r \cdot \exp\left(\frac{-v'_{abc}}{16}\right), \quad \text{whereby } \eta_{abc} \ni (r \sin \theta \cos \varphi, r \cos \theta, r \sin \theta \sin \varphi). \quad (2.33)$$

where v'_{abc} is defined by (2.30), and $1 \leq r \leq R$. The rest of the extraction procedure presented in subsection 2.5.2 is applied. Each function f_r is sampled and the Fourier transform on the sphere is applied to the sample values. The first L bands of the obtained complex coefficients are used to form the signature \mathbf{s}_r (2.27) of the function f_r .

In [58], the obtained signatures \mathbf{s}_r ($1 \leq r \leq R$) are post-processed further, by substituting the constant and the second order harmonic norms, $\|f_{r,0}\|$ and $\|f_{r,2}\|$, with three real values $c_1, c_2, c_3 \in \mathbb{R}$. The motivation for this post-processing step is the minimization of the L_2 difference between the quadratic components of two functions on a sphere. The sum ψ_r of the constant ($f_{r,0}$) and the quadratic ($f_{r,2}$) components of a function on sphere can be expressed as

$$\begin{aligned} \psi_r(x, y, z) &= f_{r,0}(x, y, z) + f_{r,2}(x, y, z) \\ &= a_0 + a_1 x^2 + a_2 y^2 + a_3 z^2 + a_4 xy + a_5 xz + a_6 yz \\ &= [x \ y \ z] \begin{bmatrix} m_1 & m_4 & m_5 \\ m_4 & m_2 & m_6 \\ m_5 & m_6 & m_3 \end{bmatrix} [x \ y \ z]^T, \end{aligned} \quad (2.34)$$

where $x^2 + y^2 + z^2 = 1$ and

$$m_1 = \psi_r(1, 0, 0), \quad m_4 = \psi_r(1/\sqrt{2}, 1/\sqrt{2}, 0) - \frac{m_1 + m_2}{2},$$

$$m_2 = \psi_r(0, 1, 0), \quad m_5 = \psi_r(1/\sqrt{2}, 0, 1/\sqrt{2}) - \frac{m_1 + m_3}{2},$$

$$m_3 = \psi_r(0, 0, 1), \quad m_6 = \psi_r(0, 1/\sqrt{2}, 1/\sqrt{2}) - \frac{m_2 + m_3}{2}.$$

Then, the symmetric matrix M is regarded as a covariance matrix (see equation (3.4)) and the singular value decomposition is performed (see section 3.2). After the transformation into a new coordinate frame, whose axes are denoted by X , Y , and Z , we obtain

$$\psi_r(X, Y, Z) = b_1 X^2 + b_2 Y^2 + b_3 Z^2 = (b_1, b_2, b_3).$$

Any function on a sphere spanned by X^2 , Y^2 , and Z^2 can be represented using an orthonormal bases $E = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. The suggested choice of E can be expressed in the basis $\{X^2, Y^2, Z^2\}$ as follows

$$\begin{aligned} \mathbf{e}_1 &= (\alpha + \beta, \alpha, \alpha) = (\alpha + \beta)X^2 + \alpha Y^2 + \alpha Z^2, & \alpha &= \sqrt{\frac{1}{3}} - \sqrt{\frac{5}{6}}, \\ \mathbf{e}_2 &= (\alpha, \alpha + \beta, \alpha) = \alpha X^2 + (\alpha + \beta)Y^2 + \alpha Z^2, & \text{where} \\ \mathbf{e}_3 &= (\alpha, \alpha, \alpha + \beta) = \alpha X^2 + \alpha Y^2 + (\alpha + \beta)Z^2, & \beta &= \sqrt{\frac{15}{2}}. \end{aligned}$$

Finally,

$$[c_1 \ c_2 \ c_3]^T = [\mathbf{e}_1 \ | \ \mathbf{e}_2 \ | \ \mathbf{e}_3]^{-1} [b_1 \ b_2 \ b_3]^T \quad (2.35)$$

and the new signature is formed by

$$\mathbf{s}'_r = (c_1, c_2, c_3, \|f_{r,1}\|, \|f_{r,3}\|, \|f_{r,4}\| \dots, \|f_{r,L-1}\|). \quad (2.36)$$

The feature vector \mathbf{f} is obtained by concatenating signatures \mathbf{s}'_r , and is normalized by dividing by ω so that $\|\mathbf{f}\| = 1$,

$$\mathbf{f} = (\mathbf{s}'_1 | \mathbf{s}'_2 | \dots | \mathbf{s}'_R) / \omega \Rightarrow \dim(\mathbf{f}) = R(L+1). \quad (2.37)$$

Thus, the feature vector is normalized to the Euclidean unit length. The suggested parameter settings are $R = 32$ and $L = 16$, whence $\dim = 544$.

The feature vector based on the exponentially decaying EDT significantly outperforms (see section 5.2.12) the descriptor based on a binary voxel grid (subsection 2.5.2). Note that in both cases, almost the same concept is used to represent features. The only difference is the post-processing, which actually does not significantly improve retrieval effectiveness, but is regarded as a valuable theoretical result.

The significantly better retrieval performance of the approach based on the exponentially decaying EDT, comparing to the approach based on a binary voxel grid, can easily be explained. Namely, if a voxel attribute describes how far an arbitrary point is from the model, then the problem of mismatched functions on a sphere (see figure 2.11) is almost eliminated. The extracted feature, the voxel grid with attributes defined by (2.33) represented in the described manner, satisfies all the requirements on 3D-shape descriptors, which are listed in section 1.3.4. In general, the descriptor based on the exponentially decaying EDT significantly outperforms all other descriptors presented in this chapter.

Chapter 3

Pose Estimation

Objects represented as polygonal meshes are given in arbitrary orientation, scale, and position in the 3D-space \mathbb{R}^3 . 3D-shape descriptors can be defined in such a way that invariance with respect to translation, rotation, scaling, and reflection of a mesh model is provided. Examples of such descriptors are the shape spectrum descriptor (section 2.3), topology matching (section 2.4), and shape distributions (section 2.5.1). If the invariance of descriptor with respect to similarity transforms is not provided by the representation of a feature, pose estimation (normalization) is necessary as a step preceding the feature extraction (see section 1.3.2). The pose normalization procedure is a transformation of a 3D-mesh model I (1.5) into a canonical coordinate frame by translating, rotating, scaling, and reflecting (flipping) the original set of vertices (1.3).

In section 3.1, we describe the problem of finding the canonical coordinates of a mesh. The most prominent tool for solving the problem is the Principal Component Analysis (PCA) [51], also known as the discrete Karhunen-Loeve transform, or the Hotelling transform, which is described in details in section 3.2. Since applying the PCA to the set of vertices (1.3) of a mesh model can produce undesired normalization results, two modifications of the PCA are given in section 3.3. Both modifications approximate the application of the PCA to the point set I (1.5) of an object. In section 3.4, we present our original method for analytical computation of various parameters needed to analyze the set of infinitely many points. This computation enables application of the PCA to an infinite point set represented as a union of triangles. We called the approach the *Continuous Principal Component Analysis* (CPCA). Examples as well as an evaluation of the continuous approach are given in section 3.5.

3.1 Problem Description

Polygonal mesh models can be generated using a variety of techniques, e.g., 3D designers use CAD software, optical devices of 3D scanners determine positions of vertices and capture the connectivity information, range images are processed in

order to obtain 3D-mesh representations, etc. Consequently, 3D-models are given in arbitrary units, position, and orientation. The majority of our 3D-shape feature vectors (chapter 4) depend on the absolute position of triangles of the underlying object (1.2). In order to fulfill the invariance criteria 3 from section 1.3.4, these features need to be extracted in a canonical coordinate frame, i.e., a coordinate system determined by the relative position of triangles of the object. An example of pose estimation (normalization) is depicted in figure 3.1. Three 3D-models of cars are shown in the original position, orientation, and scale (a, b, and c), while the corresponding appearances in the canonical frame are displayed in the second row (d, e, and f). The models are viewed from the positive side of the z -axis while the positive side of the x -axis is turned to the right.

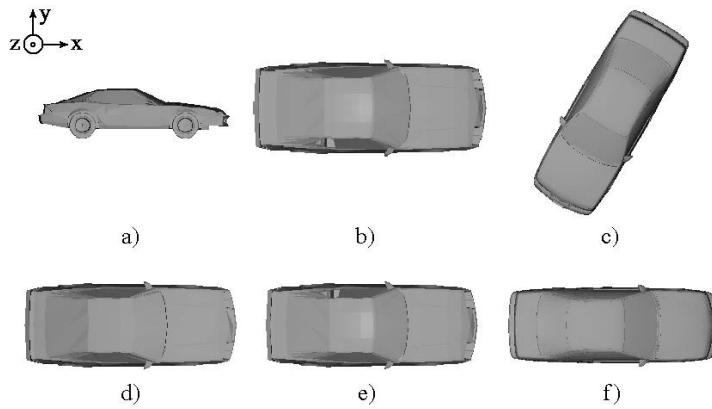


Figure 3.1: Models of cars are initially given in arbitrary units, position, and orientation (a,b, and c). The outcome of the pose estimation procedure is the canonical positioning of each model (d, e, and f).

To find the *canonical coordinate frame* of a given 3D-object we derive an affine map $\tau : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ in such a way that for an arbitrary concatenation σ of translations, rotations, reflections, and scaling the desired invariance property of τ holds, namely

$$\tau(I) = \tau(\sigma(I)), \quad (3.1)$$

where the point set I is given by (1.5). We have set $\sigma(I) := \{\sigma(\mathbf{v}) | \mathbf{v} \in I\}$ and similarly for τ .

There have been several approaches for estimating the pose of a 3D mesh model [52, 103, 105, 143, 147, 91], the most prominent one being the *Principal Component Analysis* (PCA) that produces an affine transformation of the space \mathbb{R}^3 . The basics as well as certain extensions of the PCA are given in the next sections.

Pose estimation of a 3D-mesh model based on the Extended Gaussian Images (EGIs) is one of the first approaches reported in the literature. An EGI defines a function on a unit sphere, by using normal vectors of faces of the mesh. The method

is sensitive to polygon tessellations of a 3D-shape, noise, and face orientation. More details about the technique can be found in [52].

In [91], the pose estimation is also based on the PCA wherein the underlying 3D-model is supposed to be a solid. However, 3D-models are not guaranteed to consist of closed surfaces bounding one or more solids, and it would be a difficult and questionable undertaking to enforce objects to be solids by stitching up surfaces with boundaries. Therefore, the approach is suitable only for a small class of 3D-models. The other significant drawback lies in the fact that the procedure is time-consuming.

We consider that the pose normalization should be both efficient and effective. Also, meshes in different levels-of-detail representing the same real-world object, e.g., a car represented by a mesh of 100 polygons and by a mesh of 10000 polygons, should be reasonably aligned in the canonical frame. Then, different tessellations of an object must not significantly affect the canonical coordinates. Next, if an arbitrary number of vertices are added to the set of vertices (1.3) and a triangulation of a mesh is done so that the point set I (1.5) is unchanged, then the canonical coordinates should remain the same. Moreover, the normalization should be robust with respect to outliers, noise, and arbitrary topological degeneracies of a model. Finally, the pose estimation should not be restricted to closed or orientable mesh models.

3.2 Principal Component Analysis

The Principal Component Analysis (PCA) [41, 51, 106], is widely used in signal processing, statistics (data analysis), compression, and neural computing. In some application areas, the PCA is also called the (discrete) Karhunen-Loeve transform, or the Hotelling transform. The following presentation of the original PCA analysis is adopted from [41]. The PCA is based on the statistical representation of a random variable. Suppose we have a finite set of data vectors V , where

$$V = \{ \mathbf{v} \mid \mathbf{v} = (v_1, v_2, \dots, v_n) = [v_1, v_2, \dots, v_n]^T \in \mathbb{R}^n, n \in \mathbb{N} \}. \quad (3.2)$$

Let \mathbf{m}_V be the mean of the set V

$$\mathbf{m}_V = E\{V\} = \frac{1}{|V|} \sum_{\mathbf{v} \in V} \mathbf{v}, \quad (3.3)$$

where $|V|$ denotes the number of elements of the set V (i.e., the cardinal number). The associated *covariance matrix* of the same data set is given by

$$C_V = [c_{ij}]_{n \times n} = E\{(\mathbf{v} - \mathbf{m}_V)(\mathbf{v} - \mathbf{m}_V)^T\} = \frac{1}{|V|} \sum_{\mathbf{v} \in V} (\mathbf{v} - \mathbf{m}_V)(\mathbf{v} - \mathbf{m}_V)^T. \quad (3.4)$$

Matrix C_V is, by definition, symmetric real matrix with non-negative elements. Elements c_{ij} ($i \neq j$) represent the covariances between the components v_i and v_j (3.2). If two components v_i and v_j of the data are uncorrelated, then their

covariance is zero ($c_{ij} = c_{ji} = 0$). The element c_{ii} represents the variance of the component v_i , which indicates the spread of the component values around its mean value. Eigenvalues and eigenvectors of the covariance matrix are used to form an orthonormal basis of the space \mathbb{R}^n . We recall that the eigenvectors \mathbf{e}_i ($\|\mathbf{e}_i\| = 1$) and the corresponding eigenvalues λ_i are the solutions of equations

$$C_V \mathbf{e}_i = \lambda_i \mathbf{e}_i \quad (i = 1, \dots, n). \quad (3.5)$$

We stress that in the case of a symmetric non-negative matrix all eigenvalues are non-negative real numbers. By ordering the eigenvectors according to the order of descending eigenvalues, we obtain an orthonormal basis with the first eigenvector coinciding with the direction of largest variance of the set V (3.2). Directions of largest variance are usually regarded as directions in which the original data set possesses the most significant amounts of *energy*.

Let A be a matrix consisting of ordered eigenvectors of the covariance matrix as the row vectors. The ordered eigenvectors can be seen as basis of a new coordinate frame with the origin placed at the point \mathbf{m}_V . We regard the new coordinate system as the *PCA coordinate system (frame)*. A data vector $\mathbf{v} \in V$ from the original system is transformed into the vector \mathbf{p} in the PCA frame,

$$\mathbf{p} = A(\mathbf{v} - \mathbf{m}_V). \quad (3.6)$$

In the PCA frame data are uncorrelated, i.e., the non-diagonal elements of the covariance matrix are equal to zero.

Before explaining how we engage the PCA for 3D-model retrieval purposes, we present applications of the PCA in data compression and image processing. These applications are motivation for a set of experiments aimed at reducing dimensionality of 3D-shape feature vectors, without significant loss in retrieval effectiveness (section 5.3).

Data can be compressed using the PCA in the following manner. The original vector \mathbf{v} , which is projected on the coordinate axes of the PCA frame (3.6), can be reconstructed by applying an affine map to the projection \mathbf{p} given by,

$$\mathbf{v} = A^T \mathbf{p} + \mathbf{m}_V, \quad (3.7)$$

where we used the property of an orthogonal matrix $A^{-1} = A^T$ (A^T denotes the transpose of matrix A). If we do not use all the eigenvectors of the covariance matrix, the data can be represented in a lower dimensional space, whose dimension is determined by the number of used eigenvectors, i.e., basis vectors of the orthonormal basis.

Let A_k be the matrix consisting of the first k (ordered) eigenvectors as the row vectors. By substituting A with A_k in equation (3.6), we obtain

$$\mathbf{p} = A_k(\mathbf{v} - \mathbf{m}_V) \quad \text{and} \quad \hat{\mathbf{v}} = A_k^T \mathbf{p} + \mathbf{m}_V. \quad (3.8)$$

Hence, we project the original data vector from an n -dimensional linear metric space \mathbb{R}^n on a new k -dimensional vector space \mathbb{R}^k , whose orthonormal basis consists of

the first k eigenvectors of the covariance matrix. Then, we perform a kind of reverse transform similar to (3.7). However, we cannot reconstruct the original data vector, i.e., $\hat{\mathbf{v}} \neq \mathbf{v}$, because the matrix A_k of the type $k \times n$ possesses the following properties: $A \cdot A^T = I_k$, but $A^T \cdot A \neq I_n$, where I_n denotes the identity matrix of type $n \times n$. We regard $\hat{\mathbf{v}}$ as an approximation of \mathbf{v} , which is represented in a lower-dimensional space. If we choose $k \ll n$, then the original data is *compressed* by a factor of k/n . It can be proven [51] that the described linear dimension reduction technique is optimal in the mean-square sense. In other words, the mean-square error between the original data \mathbf{v} and the reconstructed data $\hat{\mathbf{v}}$ (3.8) obtained by using a given number of eigenvectors is minimized.

Data compression using the PCA possesses the following useful properties:

- The computational costs of the subsequent processing steps are reduced;
- The presence of noise in original data will be reduced, because the directions of largest spreads (the first components) are more robust to noise than directions of lowest variance;
- By setting $k = 3$ (or $k = 2$), a high-dimensional space is projected so that data can be visualized.

The values of sorted eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ carry a useful information. Namely, the value of λ_i is proportional to the variance (energy) along the direction determined by the eigenvector \mathbf{e}_i . For applications in which a varying amount of energy of the original data should be preserved, we simply fix the number of used eigenvectors. Alternatively, the total amount of energy carried by the first k eigenvectors can be used to determine the dimensionality. For instance,

$$k = \max \left\{ j \mid \sum_{i=1}^j \lambda_i \leq t \sum_{i=1}^n \lambda_i \right\}, \quad (3.9)$$

where $t \in [0, 1]$ is a threshold. In this case, the total amount of energy (information) is approximately consistent with a varying dimensionality k . Both alternatives are applied to concatenated 3D-shape feature vectors and tested in section 5.3.

Thus, dealing with a lossy compression gained by the PCA introduces a trade-off between the reduction of vector dimension (we want to simplify the representation as much as possible) and the loss of information (we want to preserve as much as possible of the original information content). The PCA offers convenient mechanisms (fixed k vs. fixed t) to control this trade-off.

Properties of the PCA can be depicted using an application in image processing. Suppose that we have a color image of dimensions $M \times N$. Each pixel is represented by a triplet of red, green, and blue (RGB) component values. We consider that each image consists of three bands, i.e., three grayscale images each of which represents pixel values of the corresponding color. If we want to generate a single grayscale image so that the most details are shown, then we apply the PCA to the set of 3D points, which are obtained by treating color triplets of pixels as points in \mathbb{R}^3 . An example RGB image and the outcome of the PCA are shown in figure 3.2. Pixels of the given image are represented as points in a 3D space, where x , y , and z

axes represent values of red, green, and blue components, respectively. The first eigenvector (\mathbf{P}_1) having the largest eigenvalue points to the direction of largest variance (spread) whereas the second (\mathbf{P}_2) and the third (\mathbf{P}_3) eigenvectors are orthogonal to the first one.

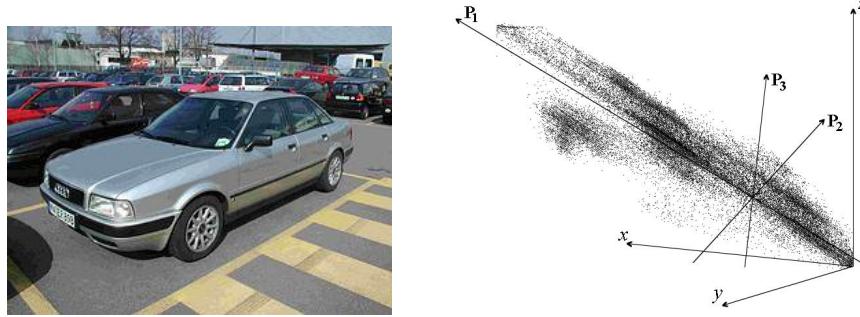


Figure 3.2: The analyzed image (left) and the pixels of the image in the color space (right). Axes x , y , and z represent values of red, green, and blue components, respectively. The PCA coordinate axes are denoted by \mathbf{P}_1 , \mathbf{P}_2 , and \mathbf{P}_3 .

In this example the first eigenvalue corresponding to the first eigenvector is $\lambda_1 = 9211.79$ while the other eigenvalues are $\lambda_2 = 437.38$ and $\lambda_3 = 74.29$. Hence, the first eigenvector contains almost all the energy. This means that the data could be well approximated with a one-dimensional representation. The result is depicted in figure 3.3. In the first row, red, green, and blue bands of the original image (figure 3.2) are represented by grayscale images. The first image in the second row (\mathbf{P}_1) is created by normalizing the first coordinate of each point in the PCA frame. In this example, we deal with 8-bit gray scale and the pixel values are normalized so that the lowest value of the first PCA coordinate is mapped to 0, the highest to 255, while the values in between are proportionally set. The last two images in figure 3.3 are obtained analogously. Because of the largest variance, the maximum contrast is contained in image 3.2d.

The example of finding largest spreads of a point set (right-hand side in figure 3.2) suggests an analogous application to the point set I (1.5) of a mesh model. In our 3D model retrieval algorithm we find the canonical coordinate frame of a 3D object (1.5) by applying the following sequence of transformations:

- *Translation* of the set I (1.5) moving its center of mass to the origin of the coordinate system;
- *Rotation* is applied so that the largest spread of the transformed points (the variance) is along the x -axis. Then a rotation around the x -axis is carried out so that the maximal spread in the yz -plane occurs along the y -axis (PCA);
- *Scaling* to a certain unit size; and
- *Reflection* with respect to xy coordinate plane is performed if the sum of certain moments is negative. Reflections with respect to yz and zx planes are fixed

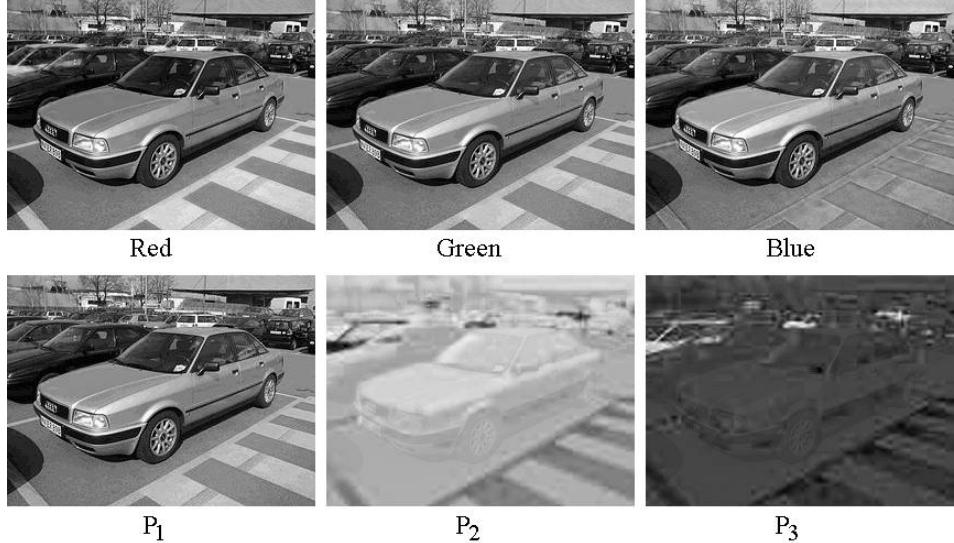


Figure 3.3: The grayscale images formed from the red, green, and blue band are shown in the first row, while the grayscale images in the second row are formed by normalizing coordinates in the PCA frame.

analogously.

Let S_i and \mathbf{g}_i be the area of the triangle T_i (1.2) and the center of gravity, respectively. For simplicity of notation we may assume that the triangles of a 3D-model intersect only on subsets of measure zero so that we may write the overall surface in the model as (compare (1.6))

$$S := S_1 + \dots + S_m = \iint_I ds. \quad (3.10)$$

Then, the center of gravity of a model, \mathbf{m}_I , is calculated by

$$\mathbf{m}_I = \frac{1}{S} \sum_{i=1}^m S_i \mathbf{g}_i \quad (3.11)$$

and the *translation invariance* is secured by forming the point set

$$I_1 := I - \mathbf{m}_I = \{\mathbf{p}' \mid \mathbf{p}' = \mathbf{p} - \mathbf{m}_I, \mathbf{p} \in I\}. \quad (3.12)$$

Having in mind that sizes of triangles of a mesh model (1.2) significantly differ, the PCA should not be applied to the set of vertices (1.3). If the differing sizes of triangles are not taken into account, then we may obtain widely varying normalized coordinate frames for models that are identical except for finer triangle resolution in some parts of the model. This fact induced modifications of the standard PCA, which are described in the following section.

3.3 Modifications of the PCA

We addressed the problem of differing sizes of triangles in [143], where we introduced appropriately chosen vertex weights w_k ($k = 1, \dots, n$). Differing sizes of triangles are taken into account by applying the PCA to the set of weighted vertices. The weights are used to approximate the covariance matrix C_I of the point set I (1.5). Using the notation introduced in section 1.3.1, the equation (3.11) can be written in the following way

$$\mathbf{m}_I = \frac{1}{S} \sum_{i=1}^m S_i \mathbf{g}_i = \frac{1}{n} \sum_{i=1}^m \frac{n S_i}{S} \frac{\mathbf{p}_{A_i} + \mathbf{p}_{B_i} + \mathbf{p}_{C_i}}{3} = \frac{1}{n} \sum_{i=1}^n \frac{n S'_i}{3S} \mathbf{p}_i = \frac{1}{n} \sum_{i=1}^n w_k \mathbf{p}_i,$$

where S'_i is the sum of surfaces of all triangles that have \mathbf{p}_i as a vertex.

Hence, the weights w_k are defined by $w_k = \frac{n S'_k}{3S}$. Obviously, $\sum_{k=1}^n w_k = n$.

The covariance matrix (3.4) of the set I is approximated as follows

$$C_I \approx \frac{1}{n} \sum_{i=1}^n w_i (\mathbf{p}_i - \mathbf{m}_I)(\mathbf{p}_i - \mathbf{m}_I)^T. \quad (3.13)$$

The rest of the PCA procedure remains the same, i.e., after forming the rotation matrix A we transform the set of vertices P (1.3)

$$\mathbf{p}'_i = A(\mathbf{p}_i - \mathbf{m}_I), \quad i = 1, \dots, n. \quad (3.14)$$

In this way, translation and rotation invariance, as well as robustness with respect to levels of detail and different tessellations of a mesh model, are achieved.

Paquet et al. [105] used centers of gravity of triangles to form the input for the PCA. Each center of gravity is multiplied by the area of surface of the corresponding triangle, applying the PCA to the obtained set of vectors. Thus, the matrix C_I is approximated by

$$C_I \approx \frac{1}{m} \sum_{i=1}^m S_i (\mathbf{g}_i - \mathbf{m}_I)(\mathbf{g}_i - \mathbf{m}_I)^T. \quad (3.15)$$

Both approaches (3.13) and (3.15) represent reasonable solutions to the problem. However, the matrix C_I is only approximated, i.e., not all the points of the set I are treated in the same way. Thus, the principal directions cannot be computed exactly and, for instance, the orientation of a symmetrical object can deviate from the desired one (see figure 3.5).

In order to secure reflection (flipping) invariance, we compute values f_x , f_y , and f_z . The value of f_x (similar f_y and f_z) is defined by

$$f_x = \frac{1}{S} \sum_{i=1}^m \text{sign}(x'_{A_i} + x'_{B_i} + x'_{C_i}) S_i \left(\frac{x'_{A_i} + x'_{B_i} + x'_{C_i}}{3} \right)^2, \quad (3.16)$$

where $\mathbf{p}'_i = (x'_i, y'_i, z'_i)$ (3.14). Then, we form a diagonal matrix

$$F = \text{diag}(\text{sign}(f_x), \text{sign}(f_y), \text{sign}(f_z)).$$

We calculate a scale factor s by

$$s = \frac{1}{S} \sum_{i=1}^m S_i \left\| \frac{\mathbf{p}'_{A_i} + \mathbf{p}'_{B_i} + \mathbf{p}'_{C_i}}{3} \right\|, \quad (3.17)$$

where $A_i, B_i, C_i \in \{1, \dots, n\}$ are given by the list of topology (1.4).

Finally, the point $\mathbf{p}''_i = s^{-1} F \mathbf{p}'_i$ (3.14) is invariant with respect to translation, rotation, reflection, and scaling of the original mesh model point \mathbf{p}_i .

Our method of applying the PCA, so that *all* of the (infinitely many) points of a mesh object I (1.5) are equally relevant for the transformation [147], is presented in the following section.

3.4 “Continuous” PCA

We regard a triangle mesh model as a union of all triangles. The continuous point set I (1.5) of the model consists of infinitely many points. In this section, we describe our approach to compute the covariance matrix of the point set I by a suitable integration. Moreover, we present a general concept for computing an integral of a function on the model’s surface.

We consider a function $f : \mathbb{T} \mapsto \mathbb{M}$, where \mathbb{T} is the set of all triangles in \mathbb{R}^3 , and \mathbb{M} can be a set of scalars (e.g., \mathbb{R}), vector space (e.g., \mathbb{R}^3), or space of matrices (e.g., matrices of the type 3×3) depending on the function f . We treat a triangle T_i (1.2) of the mesh I as an element of \mathbb{T} , i.e., $T_i \in \mathbb{T}$. We recall that a triangle T_i is determined by its vertices \mathbf{p}_{A_i} , \mathbf{p}_{B_i} , and \mathbf{p}_{C_i} . For a given affine map τ , the transformed triangle $\tau(T_i)$ is determined by vertices $\tau(\mathbf{p}_{A_i})$, $\tau(\mathbf{p}_{B_i})$, and $\tau(\mathbf{p}_{C_i})$. The surface area of T_i is denoted by S_i (1.6).

By representing a point \mathbf{v} of the triangle $T_i \in \mathbb{T}$ by barycentric coordinates we define an operator ℓ_f of the function f on the set \mathbb{T} ,

$$\ell_f(T_i) = \iint_{\mathbf{v} \in T_i} f(\mathbf{v}) ds = 2S_i \int_0^1 d\alpha \int_0^{1-\alpha} f(\alpha \mathbf{p}_{A_i} + \beta \mathbf{p}_{B_i} + (1 - \alpha - \beta) \mathbf{p}_{C_i}) d\beta. \quad (3.18)$$

Then,

$$\ell_f(I) = \sum_{i=1}^m \ell_f(T_i) = \iint_{\mathbf{v} \in I} f(\mathbf{v}) ds. \quad (3.19)$$

Equations (3.18) and (3.19) enable calculation of different parameters (values) of the set I by selecting the appropriate function f . For instance, $f = f_1(\mathbf{v}) = 1$ leads to the calculation of the surface area of the 3D mesh model (3.10), while for $f = f_2(\mathbf{v}) = \mathbf{v}$ we have (3.11)

$$\mathbf{m}_I = \frac{1}{S} \sum_{i=1}^m \iint_{\mathbf{v} \in T_i} \mathbf{v} ds = \frac{1}{S} \sum_{i=1}^m S_i \frac{\mathbf{p}_{A_i} + \mathbf{p}_{B_i} + \mathbf{p}_{C_i}}{3} = \frac{1}{S} \sum_{i=1}^m S_i \mathbf{g}_i.$$

The covariance matrix C_I can be *computed* by setting

$$f = f_3(\mathbf{v}) = (\mathbf{v} - \mathbf{m}_I) \cdot (\mathbf{v} - \mathbf{m}_I)^T. \quad (3.20)$$

We used this approach for the first time in [145], while we presented more details in [147]. By combining (3.18) and (3.20), the formula for calculating C_I can be written as

$$\begin{aligned} C_I &= \frac{1}{S} \iint_{\mathbf{v} \in I} (\mathbf{v} - \mathbf{m}_I) \cdot (\mathbf{v} - \mathbf{m}_I)^T ds \\ &= \frac{1}{12S} \sum_{i=1}^m (f_3(\mathbf{p}_{A_i}) + f_3(\mathbf{p}_{B_i}) + f_3(\mathbf{p}_{C_i}) + 9f_3(\mathbf{g}_i)) S_i. \end{aligned} \quad (3.21)$$

After the *exact* computation of C_I (in section 3.3 the covariance matrix is just approximated), the remaining part of the PCA follows the standard procedure. Since the matrix C_I is a symmetric real matrix its eigenvalues are real and the eigenvectors orthogonal. We calculate the eigenvalues of C_I , sort them in decreasing order, compute the corresponding eigenvectors and scale them to the Euclidean unit length. We form the rotation matrix A , which has the scaled eigenvectors as rows. We regard the described approach of applying the PCA to the whole point set I as the *Continuous PCA* (CPCA).

A new point set I_2 is obtained by *translating* (3.12) and *rotating* the set I using the vector \mathbf{m}_I (3.11) and matrix A ,

$$I_2 := A \cdot (I - \mathbf{m}_I) = \{\mathbf{v} \mid \mathbf{v} = A \cdot (\mathbf{u} - \mathbf{m}_I), \mathbf{u} \in I\}. \quad (3.22)$$

To ensure the *reflection invariance* we multiply points in I_2 by a diagonal matrix $F = \text{diag}(\text{sign}(f_x), \text{sign}(f_y), \text{sign}(f_z))$, where f_x is computed by

$$f_x = \frac{1}{S} \iint_{\mathbf{v}' \in I_2} \text{sign}(v'_x) |v'_x|^p ds, \quad p = 2, 3, \dots \quad (f_y, f_z \text{ similar}), \quad (3.23)$$

and $\mathbf{v}' = (v'_x, v'_y, v'_z) \in I_2$. Thus, we set $f(\mathbf{v}') = \text{sign}(v'_x) |v'_x|^p$ in (3.18).

We performed tests for $p = 2$ and $p = 3$ and concluded that better results were obtained for $p = 2$. For $p = 2$, the value of f_x is analytically computed by

$$\begin{aligned} f_x &= \frac{1}{6S} \sum_{i=1}^m F_i^x S_i, \\ F_i^x &= \begin{cases} J_i^x, & x'_{A_i}, x'_{B_i}, x'_{C_i} \geq 0 \\ J_i^x - 2L_i^x, & x'_{A_i} < 0, x'_{B_i}, x'_{C_i} \geq 0 \\ -J_i^x + 2L_i^x, & x'_{A_i} \geq 0, x'_{B_i}, x'_{C_i} < 0 \\ -J_i^x, & x'_{A_i}, x'_{B_i}, x'_{C_i} < 0 \end{cases} \end{aligned} \quad (3.24)$$

$$\begin{aligned} J_i^x &= (x'_{A_i})^2 + (x'_{B_i})^2 + (x'_{C_i})^2 + x'_{A_i} x'_{B_i} + x'_{A_i} x'_{C_i} + x'_{B_i} x'_{C_i}, \\ L_i^x &= \frac{(x'_{A_i})^4}{(x'_{B_i} - x'_{A_i})(x'_{C_i} - x'_{A_i})}, \end{aligned}$$

where $A(\mathbf{p}_i - \mathbf{m}_I) = (x'_i, y'_i, z'_i) \in I_2$. Note that the conditions $x'_{A_i} < 0, x'_{B_i}, x'_{C_i} \geq 0$ are interpreted as “two x -coordinates of triangle vertices are non-negative and one is negative”, while $x'_{A_i} \geq 0, x'_{B_i}, x'_{C_i} < 0$ is the opposite. This means that if we originally have, for example, $x'_{B_i} < 0, x'_{A_i}, x'_{C_i} \geq 0$, then we exchange the values of x'_{A_i} and x'_{B_i} and the condition $x'_{A_i} < 0, x'_{B_i}, x'_{C_i} \geq 0$ is fulfilled. The exchange of values corresponds to the renaming (reordering) of vertices.

Scaling invariance is achieved by scaling the set I_2 by the inverse of a scaling factor s . We have four approaches for calculating the scaling factor s . As the first approach, we take

$$s = d_{avg}, \quad (3.25)$$

where d_{avg} is the average distance of a point \mathbf{p} on the surface of a model I to the center of gravity of the model (i.e., the new coordinate origin). By setting $f(\mathbf{v}') = \|\mathbf{v}'\| = \sqrt{\mathbf{v}' \cdot \mathbf{v}'}$ in (3.18), $\mathbf{v}' = (v'_x, v'_y, v'_z) \in I_2$ (3.22), we obtain,

$$\begin{aligned} d_{avg} &= \frac{1}{S} \iint_{\mathbf{v}' \in I_2} \sqrt{\mathbf{v}' \cdot \mathbf{v}'} ds \\ &= \frac{2}{S} \sum_{i=1}^m S_i \int_0^1 d\alpha \int_0^{1-\alpha} \sqrt{(\alpha \mathbf{p}'_{A_i} + \beta \mathbf{p}'_{B_i} + (1-\alpha-\beta) \mathbf{p}'_{C_i})^2} d\beta, \end{aligned} \quad (3.26)$$

where $\mathbf{p}'_{A_i} = (x'_{A_i}, y'_{A_i}, z'_{A_i}) = A(\mathbf{p}_{A_i} - \mathbf{m}_I)$, and similarly for \mathbf{p}'_{B_i} and \mathbf{p}'_{C_i} .

Since the computation of the integrals in (3.26) is too expensive, we approximate the value of d_{avg} by sampling the surface of the mesh uniformly. The following pseudocode describes our algorithm for approximating d_{avg} :

```

p_min = 64000; // - the minimal number of samples
d_avg = 0; // - the approximation of average distance
for j = 1, ..., m // m - the number of triangles (1.2)
    p_j = ⌈ √(p_min * S_j / S) ⌉; // see (1.6) and (4.38), ⌈ · ⌉ is a ceiling function
    d_AB = (p'_B_j - p'_A_j) / p_j;
    d_AC = (p'_C_j - p'_A_j) / p_j;
    d_g = (d_AB - d_AC) / 3;
    δ = S_j / p_j^2;
    for x = 0, ..., p_j - 2
        for y = 0, ..., x
            g = p'_A_j + (x - y)d_AB + yd_AC + d_g;
            d_avg ← d_avg + δ(||g|| + ||g + d_g||);
    for y = 0, ..., p_j - 1
        g = p'_A_j + (p_j - 1 - y)d_AB + yd_AC + d_g;
        d_avg ← d_avg + δ||g||;
    d_avg ← d_avg / S;

```

This approximation is based on a subdivision of a triangle $T_j \equiv \Delta \mathbf{p}_{A_j} \mathbf{p}_{B_j} \mathbf{p}_{C_j}$ into p_j^2 coincident triangles (see figure 4.18). It is also possible to approximate the

average distance of a point on the surface from the center of gravity by randomly selecting $N >> 0$ points $\mathbf{v}_i \in I$, $1 \leq i \leq N$, using (2.19), and computing $d_{avg} \approx \sum_{i=1}^N \|\mathbf{v}_i\|/N$. The approximation obtained by using the presented pseudocode provides a better trade-off between accuracy and efficiency than the approximation by random sampling.

As the second method, we calculate a *continuous scaling factor*

$$s = \sqrt{\frac{s_x^2 + s_y^2 + s_z^2}{3}}, \quad (3.27)$$

where s_x , s_y , and s_z denote the average distances of points $\mathbf{v} \in I_2$ (3.22) from the yz -, xz -, and xy -coordinate hyperplanes, respectively, i.e., by setting $f(\mathbf{v}') = |v'_x|$ in (3.18), we have

$$s_x = \frac{1}{S} \iint_{\mathbf{v}'=(v'_x, v'_y, v'_z) \in I_2} |v'_x| ds \quad \text{and likewise for } s_y, s_z. \quad (3.28)$$

Analytically, s_x is computed by

$$s_x = \frac{1}{3S} \sum_{i=1}^m M_i^x S_i,$$

$$M_i^x = \begin{cases} |x'_{A_i} + x'_{B_i} + x'_{C_i}|, & x'_{A_i} x'_{B_i}, x'_{B_i} x'_{C_i}, x'_{A_i} x'_{C_i} \geq 0 \\ |x'_{A_i} + x'_{B_i} + x'_{C_i}| - 2K_i^x, & x'_{A_i} x'_{B_i}, x'_{A_i} x'_{C_i} \leq 0, x'_{B_i} x'_{C_i} \geq 0 \\ K_i^x = \frac{x_{A_i}^3}{(x'_{B_i} - x'_{A_i})(x'_{C_i} - x'_{A_i})}, & \end{cases} \quad (3.29)$$

where $A(\mathbf{p}_i - \mathbf{m}_I) = (x'_i, y'_i, z'_i) \in I_2$.

The third choice for the scaling factor s is to set

$$s = s_x, \quad (3.30)$$

i.e., to make the average distance (3.28) from the yz -coordinate hyperplane constant, equal to 1.

Finally, as the fourth option, we take

$$s = \sqrt{\lambda_1}, \quad (3.31)$$

where λ_1 is the largest eigenvalue of the covariance matrix C_I (3.21). Thus, the energy of the first principal component is averaged.

Our experiments (see section 5.2) suggest that the first approach, $s = d_{avg}$, is the best choice.

Putting all the above together, the affine map τ (3.1), defined by

$$\tau(\mathbf{v}) = s^{-1} \cdot F \cdot A \cdot (\mathbf{v} - \mathbf{m}_I) \quad (3.32)$$

is applied to all points of the original object I (1.5). In practice, it suffices to transform only the set of vertices P (1.3).

3.5 Evaluation of the Continuous Approach

In contrast to the usual application of the PCA, we work with sums of integrals over triangles (3.19) in place of sums over vertices which makes our approach more complete taking into account all points of the model I (1.5) with equal weight. The calculation of the integrals is only slightly more expensive. The time needed to calculate parameters and transform an object into the canonical frame (normalization time) linearly depends on the complexity of the object. Figure 3.4 comprises scatter plots of normalization time vs. complexity of 3D-mesh models, using the CPCPA (section 3.4), when (3.27) is used to fix the scale, and the two modifications of the PCA (section 3.3). We selected the number of triangles as a parameter that denotes the complexity of a 3D-model (x -axis) and measured normalization times (in milliseconds) for each of the three cases. The tests were carried on a computer running Windows 2000 Professional, with 1GB RAM and an 1.4 GHz AMD processor. The most complex model consists of 215473 triangles and the continuous normalization step of this model takes 589ms. Each scatter plot is approximated by a line $y = \alpha x + \beta$ representing the optimal approximation in the mean-square sense. The values of α are the following: in the case of the CPCPA, $k = 0.00277$; the modification with the weights associated to centroids (3.15) $k = 0.00189$; and the modification with the weights associated to vertices (3.13) $k = 0.00146$. This means that the time needed for normalization step when we use the CPCPA is approximately twice the time needed for normalization when the modification proposed in [143] is used. The method based on weights associated to vertices (3.13) is faster than then the approach based on weights assigned to centroids (3.15), because a triangle mesh model possesses more triangles than vertices, on average (see section 5.1).

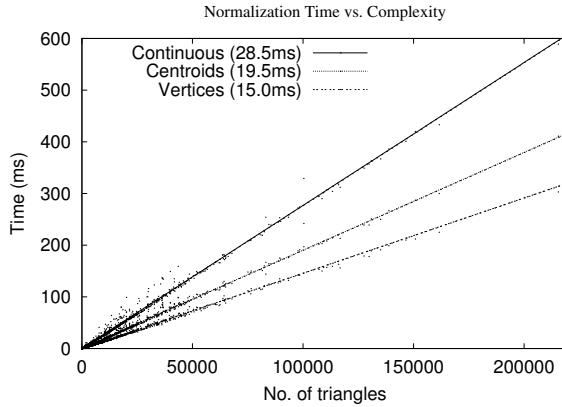


Figure 3.4: Scatter plots of normalization time vs. complexity of 3D-mesh models, using the CPCPA and modifications with weights associated to centroids and vertices. Average normalization times are given in the brackets

The average normalization time of the continuous approach is 28.5ms, while the normalization based on weights assigned to vertices is completed in 15.0ms. We consider the average difference of 13.5ms to be slight.

To demonstrate differences between the three presented normalization methods we give examples in figure 3.5. As expected, the CPCPA is the most stable. However, in figure 3.5c the canonical frame obtained by using the modification with weights associated to vertices produces better positioning of the model, because of the method's inability to treat equally all the points of the set I (1.5). The better performance of the discrete approach than the continuous method is just a spacial case. A different tessellation of the model in figure 3.5c may cause varying principal directions \mathbf{P}'_1 , \mathbf{P}'_2 , and \mathbf{P}'_3 . To explore the stability of all three approaches, we inserted $k \cdot m$ ($k \in \{1, 3, 10, 30, 100\}$) random vertices in triangles T_i ($1 \leq i \leq m$), and performed a re-triangulation. For instance, if $\mathbf{p} \in T_i$ is inserted in the list of vertices V (1.3), then the triangle $T_i = \Delta \mathbf{p}_{A_i} \mathbf{p}_{B_i} \mathbf{p}_{C_i}$ is erased from the list of triangles T (1.2) and the new triangles $\Delta \mathbf{p}_{A_i} \mathbf{p}_{B_i} \mathbf{p}$, $\Delta \mathbf{p}_{B_i} \mathbf{p}_{C_i} \mathbf{p}$, and $\Delta \mathbf{p}_{C_i} \mathbf{p}_{A_i} \mathbf{p}$ are added to T . As the number ($k \cdot m$) of inserted vertices increases, \mathbf{P}'_1 and \mathbf{P}''_1 converge to \mathbf{P}_1 . The continuous principal axis \mathbf{P}_1 remains invariant with respect to the number of inserted vertices.

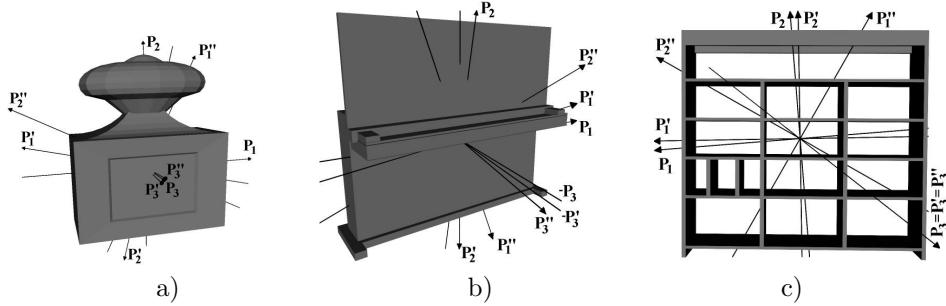


Figure 3.5: Examples of pose estimation using three presented approaches. The CPCPA axes are denoted by \mathbf{P}_1 , \mathbf{P}_2 , and \mathbf{P}_3 , the axes obtained by using the modification with weights associated to vertices are denoted by \mathbf{P}'_1 , \mathbf{P}'_2 , and \mathbf{P}'_3 , and the axes obtained by using the modification with weights associated to centroids are denoted by \mathbf{P}''_1 , \mathbf{P}''_2 , and \mathbf{P}''_3 .

Our normalization method is very efficient and rather effective for many categories of 3D-objects. However, it is not perfect. Examples of pose estimation using the continuous approach for categories of cups, cars, and models of humans are depicted in figure 3.6. Models of cups in the first row have different rotations, reflections, and assignments of principle axes. In a way, the category of cups is sub-classified by our normalization step. However, models of cars in the second row are almost ideally transformed. The outcome of our normalization step for models of humans is also satisfying, regardless the presence of outliers (third model in the last row). Our approach based on the CPCPA is suitable for categories of 3D-objects such as cars, bottles, humans, missiles, swords, ships, glasses, dogs, horses,

etc. Conversely, categories of models of chairs, cups, airplanes, trees, etc., are not consistently aligned in the canonical coordinates, whence these categories are sub-classified. Moreover, examples given in figure 3.6 motivate us to improve the scaling factor (third model in the last row) as well as to fix the reflection invariance in a more stable way (last model in the second row).

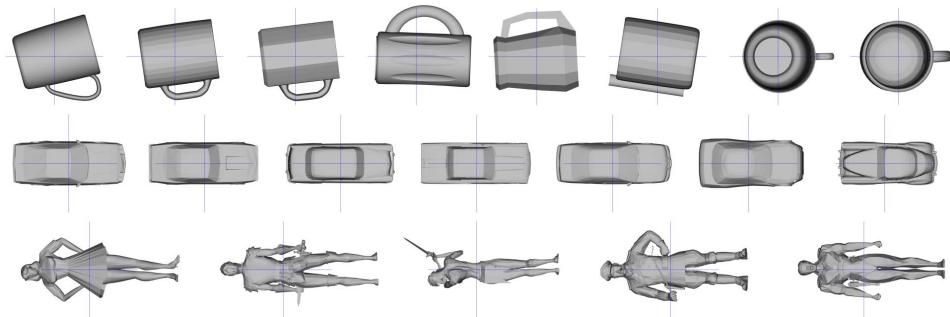


Figure 3.6: Examples of pose estimation using the continuous approach. All models are visualized from positive sides of the z -axis, while the x -axis travels to the right-hand side.

Since our pose normalization step does not align all models in an ideal way, a natural question is:

Should we eliminate 3D-shape descriptors that require the use of the PCA, and focus only those in which the invariance with respect to similarity transforms is provided by the definition of descriptor?

Shape descriptors that are presented in sections 2.3, 2.4, and 2.5 do not require the use of the PCA. We recall that the topology matching (section 2.4) relies on a graph representation, which is invariant with respect to rotation of a mesh model. The MPEG-7 shape spectrum descriptor (section 2.3) is based on local features (curvature indices). Shape distributions (subsection 2.5.1) are extracted from relative features. Both the shape spectrum and shape distributions are represented in a form of histogram. Finally, in section 2.5.2 a technique that uses certain mathematical properties to secure rotation invariance is presented. As a contrast to all these approaches, our feature vectors (chapter 4) heavily rely upon the pose normalization step, because we mostly consider *absolute features*. In section 5.2.13, we compared approaches that use the CPCPA vs. approaches that avoid the PCA, and the results show that descriptors that use the PCA outperform the others.

Explanations for these results are the following:

- For categories of models that are suitable for the PCA, good object alignments make absolute feature effective;
- When a class of models is not uniformly oriented by the CPCPA, then it is usually subdivided into classes of consistently oriented models. Absolute features are effective on a subclass of consistently oriented models;

- Certain descriptors based on absolute features possess specific invariance properties, which compensate different orientations in specific cases. For example, a feature vector may be inherently invariant with respect to rotations around the z -axis (see remark 4.2). Suppose that the z -axis is correctly determined for a pair of similar objects (e.g., airplanes, cups, desks), but the models are differently rotated around the z -axis. Then, the resulting feature vectors will not be affected by the non-consistent alignment. A proof-of-concept is demonstrated in figure 3.7 (relevant models that are differently rotated around the z -axis in the canonical frame are still among the top matches);
- Local and relative features represented by histograms, as well as features obtained by averaging (or summing up), form descriptors whose elements (vector components) are not related to specific parts of a model. Usually, statistical features are inferior to the absolute features, which can be used to reconstruct the model (i.e., vector components are related to specific parts of the model);

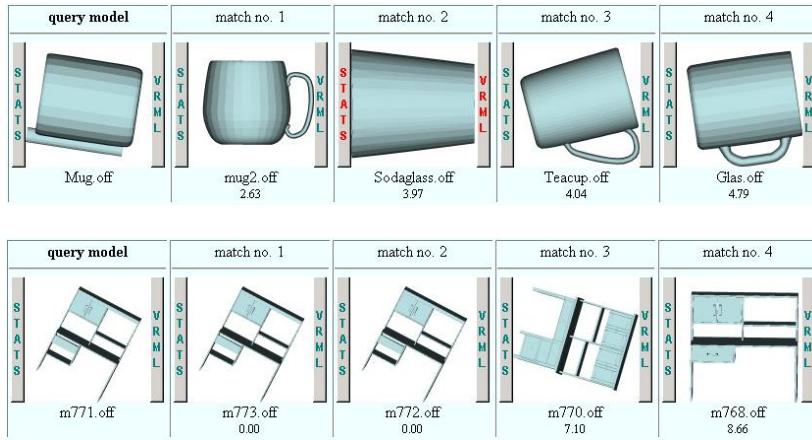


Figure 3.7: Shape-similarity search for a cup and desk using a descriptor that is inherently invariant with respect to rotations around the z -axis. In this case, the task of the CPCAs is to fix the z -axis correctly. The models are visualized from the positive side of the z -axis in the canonical coordinate frame, while the x -axis travel to the right.

Since no reported technique that avoids the use of the PCA shows better performance (see section 5.2) than our best methods relying upon the CPCAs, we consider that the use of the PCA is justified. Moreover, we expect that further improvements of the normalization step will result in increased retrieval performance of methods relying upon pose estimation.

Chapter 4

3D-Shape Feature Vectors

In this chapter, we present our original methods for describing 3D-shape, which are listed in table 4.1. Since the optimal way of encoding information about 3D-shape is not prescribed, we consider a variety of different features to define shape descriptors (feature vectors). The approaches range from considering 2D rectangular images (SIL,DBD) and images on a sphere (RAY,RSH,SSH,CSH) to exploiting 3D-features (VOL) and volumetric data (VOX). Besides, we define moments (MOM) and a novel data structure – layered depth sphere (LDS,RID), which are used for describing shape. Finally, a concept of hybrid descriptors (HYB), obtained by crossbreeding complementary feature vectors, is introduced.

Approach	Section	Abbreviation
Ray-based (icosahedron)	4.1	RAY
Silhouette-based	4.2	SIL
Depth buffer-based	4.3	DBD
Volume-based	4.4	VOL
Voxel-based	4.5	VOX
Ray-based with spherical harmonic representation	4.6.2	RSH
Moments-based	4.6.3	MOM
Shading-based	4.6.4	SSH
Complex	4.6.5	CSH
Layered depth sphere-based	4.6.6	LDS
Rotation invariant	4.6.6	RID
Hybrid	4.7	HYB

Table 4.1: Our 3D-shape descriptors.

We follow the general 3D-model retrieval algorithm 1.1, and most of our feature vectors are extracted in the canonical coordinate frame of a 3D-model. In other words, feature extraction follows after the complete normalization step (see section 3.4). The only exception is the rotation invariant feature vector based on layered

depth spheres (RID), where we use certain properties to provide rotation invariance of the descriptor without orienting the model using the CPCA.

We stress that our techniques are *not restricted to closed and orientable polygonal meshes* (see definitions 1.1 and 1.2). Since we consider a 3D-model to be the point set I (1.5), we do not introduce a constraint that the polygonal mesh need to be closed or orientable, in order to extract any of descriptors.

We describe in details all our feature vectors, in order to provide sufficient information to a reader who wants to implement and test our methods. For each feature vector, we give the average feature extraction time, which does not include times needed for loading and normalizing a 3D-mesh model. Retrieval performance of the feature vectors, which are presented in this chapter, is addressed in section 5.2.

4.1 Ray-Based Feature Vector

Information about 3D-shape of a model can be obtained by measuring the extent of the object in given directions, i.e., along defined rays. In this section, a 3D-shape descriptor that we call *ray-based feature vector in the spatial domain* [143, 141] is presented. The feature extraction is performed in the canonical frame by probing a 3D-mesh model along selected directional (unit) vectors. Suppose we have a given set of N unit vectors \mathbf{u}_i , ($\|\mathbf{u}_i\| = 1$, $i = 1, \dots, N$). Then, we intersect the polygonal mesh with the *ray* emanating from the coordinate origin and traveling in the direction \mathbf{u}_i . The distance r_i to the farthest intersection point is taken as the i -th component of the feature vector. If there is no intersection, then we take $r_i = 0$. The dimension of the vector is equal to the number of samples, N .

Let S^2 be a sphere of radius 1 with the center at the origin. The directional unit vectors \mathbf{u}_i can be interpreted as points on the sphere S^2 .

We define a function on the sphere $r(\mathbf{u})$ ($\mathbf{u} \in S^2$)

$$\begin{aligned} r &: S^2 \rightarrow [0, +\infty) \\ r(\mathbf{u}) &= \max\{ r \geq 0 \mid r\mathbf{u} \in I \cup \{O\} \} \end{aligned} \quad (4.1)$$

where I is given by (1.5). The function $r(\mathbf{u})$ measures the extent of the object from the origin O in the directions given by $\mathbf{u}_1, \dots, \mathbf{u}_N$. The ray-based feature vector \mathbf{f} of a model is composed as

$$\mathbf{f} = (r_1, \dots, r_N), \quad r_i = r(\mathbf{u}_i). \quad (4.2)$$

In [143] the vertices of a dodecahedron, with the center at the coordinate origin, are taken as directional vectors ($N = 20$). The corresponding feature vector possesses the fixed number of components. In order to provide a multi-resolution representation of the feature, which is a desirable property of descriptors (section 1.3.4, property 6), we engaged [141] dodecahedron's dual solid – the icosahedron (defined in figure 1.2). Each triangle of the icosahedron is subdivided into k^2 ($k = 1, 2, \dots$) coincident triangles (see figure 4.1).

A similar subdivision could be done using the dodecahedron, e.g., to subdivide each pentagon into five coincident triangles, which could recursively be subdivided

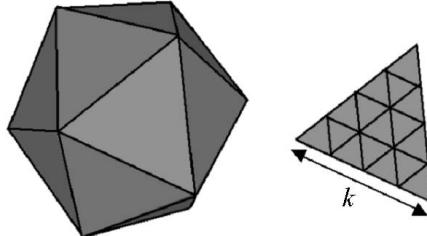


Figure 4.1: Icosahedron with subdivision.

further. Since the subdivision of dodecahedron is less uniform, we decided to use the subdivision of icosahedron. Hence, after the subdivision we obtain $20k^2$ triangles with totally $10k^2 + 2$ distinctive vertices \mathbf{v}_i ($i = 1, \dots, 10k^2 + 2$). Each vertex \mathbf{v}_i is projected on the sphere S^2 by finding its unit vector $\mathbf{u}_i = \mathbf{v}_i / \|\mathbf{v}_i\|$. The unit vectors \mathbf{u}_i are used as directional vectors. Thus, the changeable dimension of the feature vector is provided, i.e., $N = 10k^2 + 2$ ($k = 1, 2, \dots$).

Our feature extraction procedure is very efficient. The ray-triangle intersection problem has been well-studied [6, 80, 40]. In figure 4.2, a ray \mathbf{u} , cast from the coordinate origin O , intersects the plane of triangle ABC at the point P . Using the fact that P is collinear with \mathbf{u} , $P = d \cdot \mathbf{u}$, and representing P by barycentric coordinates, $P = \alpha A + \beta B + (1 - \alpha - \beta)C$, we obtain the system of three equations, with unknown quantities d , α , and β . If $d > 0$ and if α and β satisfy the conditions

$$\alpha \geq 0, \beta \geq 0, \text{ and } \alpha + \beta \leq 1, \quad (4.3)$$

then the point of intersection P belongs to the triangle. Instead of solving the system of equations using Cramer's rule directly and testing the conditions (4.3), we created a faster method aimed at reducing the total number of arithmetic operations. If \mathbf{n} is the normal vector of the plane containing the triangle ABC and point P , then it holds $\mathbf{n} \cdot P = \mathbf{n} \cdot A$. Therefore, we compute the value of d directly, $d = \mathbf{n} \cdot A / \mathbf{n} \cdot \mathbf{u}$. If $d > 0$, then we determine if $P = d \cdot \mathbf{u}$ lies inside the triangle ABC by checking the conditions

$$\begin{aligned} ((A - P) \times (B - P)) \cdot \mathbf{n} &\geq 0, \\ ((B - P) \times (C - P)) \cdot \mathbf{n} &\geq 0, \text{ and} \\ ((C - P) \times (A - P)) \cdot \mathbf{n} &\geq 0, \end{aligned}$$

which are similar to (4.3). Since we measure the extent r of a polygonal mesh in the direction \mathbf{u} , the value of r is initialized to zero, $r = 0$. If a triangle is intersected by the ray travelling along \mathbf{u} , then we check if the computed extent d is greater than the current one, $r = \max\{r, d\}$. Our ray-triangle intersection method is described by the pseudocode in figure 4.3, as algorithm 1. The algorithm deals also with special cases, i.e., it examines if the triangle is degenerated (surface area equal to zero) as well as if the triangle plane is parallel to \mathbf{u} .

Möller and Trumbore [80] proposed a method that is believed to be the fastest ray-triangle intersection routine for triangles, which do not have precomputed plane

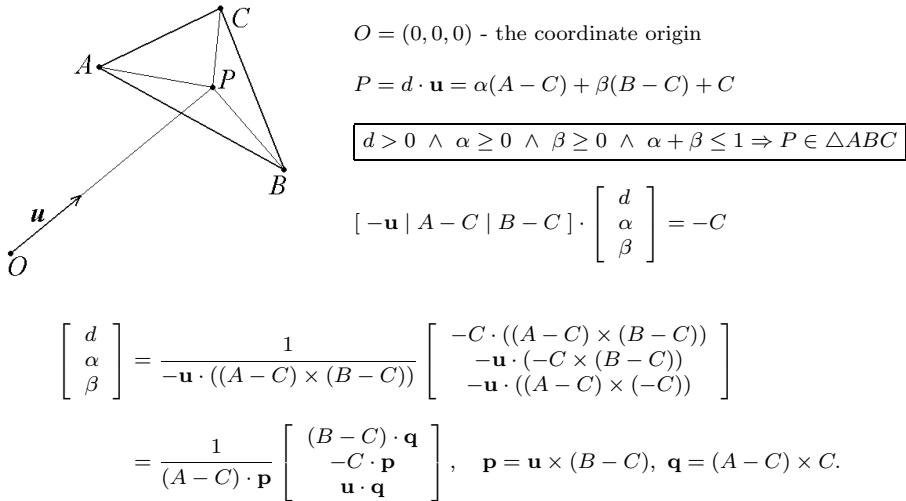


Figure 4.2: Ray-triangle intersection.

equations. Their method is depicted in figure 4.2. Instead of solving the system of equations directly by Cramer's rule, the number of arithmetic operations is reduced by appropriate reordering of computations. Besides, the statistical analysis presented in [80] suggests that it is better to test first if the point P belongs to the triangle ABC (4.3), and then to calculate the extent d . In order to reduce the number of operations further, we slightly modified the algorithm given in [80] by eliminating two divisions. The modified method is described by the pseudocode in figure 4.3, as algorithm 2. Obviously, the algorithm 1 performs more arithmetic operations than the algorithm 2. Note that if the normal vector of the triangle is precomputed, the average operation count of algorithm 1 drops.

The ray-based feature vector of a 3D-model with m triangles can be extracted using a brute force method, by intersecting $10k^2 + 2$ rays with all m triangles. The brute force approach ("exhaustive search") was used in [141]. In order to accelerate the feature extraction procedure, we created an algorithm, which is more suitable for our application than previously presented methods [39, 13, 3, 40, 1]. There is a variety of accelerated ray-casting techniques, which use various data structures including octrees, bounding volume hierarchies, spatial partitions, and uniform grids. For complex 3D-scenes with a lot of objects, a binary space partitioning tree (or BSP Tree) [34] is a frequently used data structure that organizes objects within a space. A common problem of techniques that use 3D-space subdivision [39] is that a ray which misses everything must still be checked against contents of each region or voxel it intersects. The idea of space subdivision was extended in [3] by including ray direction. The space of all rays is adaptively subdivided into equivalence classes E_1, \dots, E_l . Candidate object sets C_1, \dots, C_l are constructed in such a way that

```

// r - the current extent in the direction u
// ε = 10-10 (a small positive value)

h = (B - A) × (C - A);
// Is the triangle degenerated?
if ||h|| > ε // NON degenerated
    // Calculate the normal unit vector
    n = h / ||h||;
    // Is u orthogonal to n?
    if |n · u| > ε // NON orthogonal
        d = n · A / n · u;
        if (d > r)
            P = d · u;
        // Is P inside △ABC ?
        if ( ((A - P) × (B - P)) · n ≥ 0 ∧
            ((B - P) × (C - P)) · n ≥ 0 ∧
            ((C - P) × (A - P)) · n ≥ 0 )
            r = d;

```

Algorithm 1.

```

d = 0, CA = A - C, CB = B - C;
p = u × CB;
det = CA · p;
if det > ε
    αs = -C · p; // α = αs/det
    if αs ≥ 0 ∧ αs ≤ det
        q = CA × C;
        βs = u · q; // β = βs/det
        if βs ≥ 0 ∧ αs + βs ≤ det
            d = (CB · q)/det;
    else if det < -ε
        αs = -C · p; // α = αs/det
        if αs ≤ 0 ∧ αs ≥ det
            q = CA × C;
            βs = u · q; // β = βs/det
            if βs ≤ 0 ∧ αs + βs ≥ det
                d = (CB · q)/det;
    if d > r
        r = d;

```

Algorithm 2.

Figure 4.3: Ray-triangle intersection algorithms.

C_i contains all objects that can be intersected by a ray from E_i . A method for ray tracing triangular meshes presented in [1] requires preprocessing and a complex data structure is attached to each triangle. The most significant drawback of the approach [1] is the exhaustive search. Another problem of the method is a requirement that all triangles of mesh models must be correctly oriented, which is usually not the case if the models are retrieved from the Internet.

Our technique for ray-casting triangular meshes does not need any preprocessing step for triangles and there are no restrictions regarding orientations of triangles. The number of ray-triangle intersections is significantly reduced compared to the brute force method. Therefore, we consider that our technique is very suitable for extracting the ray-based feature vector. The key idea of the method is finding a set of rays that are candidates to intersect a triangle under consideration. Firstly, all directional unit vectors

$$\mathbf{u}_i = \mathbf{u}_i(\varphi_i, \theta_i) = (\cos \varphi_i \sin \theta_i, \sin \varphi_i \sin \theta_i, \cos \theta_i), \quad i = 1, \dots, N, \quad (4.4)$$

$$-\pi < \varphi_i \leq \pi, \quad 0 < \theta_i \leq \pi,$$

are sorted in the non-decreasing order of φ_i and θ_i . More precisely,

$$\mathbf{u}_i(\varphi_i, \theta_i) < \mathbf{u}_j(\varphi_j, \theta_j) \iff \varphi_i < \varphi_j \vee (\varphi_i = \varphi_j \wedge \theta_i < \theta_j).$$

We stress that the sorting of directional vector is not performed for each 3D-model, but only once. The quicksort algorithm sorts an array of n elements with $\mathcal{O}(n \log n)$ comparisons on average. In the worst case, the complexity is $\mathcal{O}(n^2)$. However, we

use the *intrasort* algorithm, which is a part of the SGI Standard Template Library (STL) [42]. Intrasort is very similar to quicksort, and is at least as fast as quicksort on average. A characteristic that makes the intrasort more prominent is that the worst case complexity is equal to the average case complexity, $\mathcal{O}(n \log n)$.

As a contrast to all previously mentioned techniques, our algorithm efficiently determines a set of rays which are candidates for intersection with a given triangle. We simply calculate the range of angles φ_i and θ_i of directional vectors, which could intersect the triangle. For a triangle T , we analytically determine φ_{min} , φ_{max} , θ_{min} , and θ_{max} , where

$$\begin{aligned}\varphi_{min} &= \min \Phi, & \varphi_{max} &= \max \Phi, & \Phi &= \{ \varphi \mid \mathbf{p}(\theta, \varphi, \rho) \in T \}, \\ \theta_{min} &= \min \Theta, & \theta_{max} &= \max \Theta, & \Theta &= \{ \theta \mid \mathbf{p}(\theta, \varphi, \rho) \in T \}, \\ \mathbf{p}(\theta, \varphi, \rho) &= \rho(\cos \varphi \sin \theta, \sin \varphi \sin \theta, \cos \theta).\end{aligned}$$

Then, a necessary condition that the ray $\mathbf{u}_i(\varphi_i, \theta_i)$ can intersect the triangle T is:

$$\varphi_{min} \leq \varphi_i \leq \varphi_{max} \quad \wedge \quad \theta_{min} \leq \theta_i \leq \theta_{max}. \quad (4.5)$$

We efficiently determine (using binary search) the subset C_T of sorted directional vectors \mathbf{u}_i (4.4), which satisfy (4.5). We apply the ray-triangle intersection algorithm to each directional unit vector from C_T , whence the number of ray-triangle intersections is significantly reduced. The reduction is depicted in figure 4.4, where the ratio of the total number of performed ray-triangle intersections using our approach and using the brute force is shown, for various numbers of rays, $N = 10^2k+2$, $k = 1, \dots, 8$. For $k \geq 2$, our approach performs less than 1% of the number of ray-triangle intersections performed by the exhaustive search. The results are obtained by extracting features from the collections of 3D-models described in section 5.1.

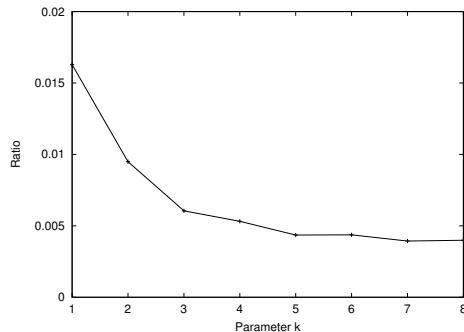


Figure 4.4: Ratio of the number of ray-triangle intersections using our approach and using the brute force vs. parameter k . The number of rays is equal to $10k^2 + 2$.

The speed-up of our approach over the brute force method for ray-casting triangular meshes is shown in table 4.2, where the average extraction times for various

dimensions of the ray-based feature vector are obtained using the official MPEG-7 collection (section 5.1). The results are obtained on a PC with an 1.4 GHz AMD processor running Windows 2000. The brute force method is used with the ray-triangle intersection algorithm 1, while our method for ray-casting triangular meshes is tested with both algorithm 1 and algorithm 2. We observe that the most efficient way of extracting the ray-based descriptor is to combine our ray-casting approach with the algorithm 2.

Dimension	12	42	92	162	252	362	492	642
Brute force (ms)	11.8	42.6	91.8	161.8	252.1	361.9	491.9	641.5
Algorithm 1 (ms)	16.0	18.2	19.2	21.8	22.6	25.3	26.8	30.4
Algorithm 2 (ms)	14.6	16.0	17.0	19.6	20.4	23.2	24.7	28.1

Table 4.2: Average extraction times for various dimensions of the ray-based feature vector using the brute force method, our approach combined with the algorithm 1, and our approach combined with the algorithm 2 (figure 4.3).

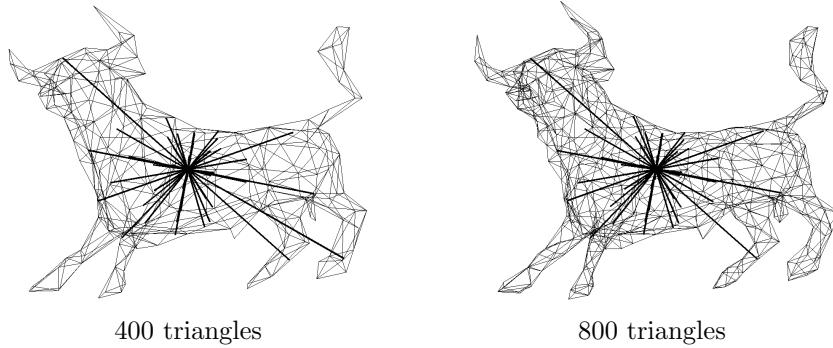
In order to visualize the feature vector and to depict characteristics of the ray-based descriptor, we give two examples in figure 4.5. Examples of feature extraction for vectors of dimension $N = 42$ ($k = 2$) are shown, where wire-frame representations of triangular meshes are visualized. The model on the left side possesses 400 triangles and is obtained by simplifying the model with 800 triangles using *QSlim 1.0* simplification software [37]. For each model, rays are emanated from the origin (center of mass) in given directions, and the furthest points of intersections with the underlying mesh-model are found, i.e., the function r (4.1) is sampled at points \mathbf{u}_i (4.4).

The ray-based 3D-shape descriptor in the spatial domain is suitable for retrieving some categories of 3D models (e.g., missiles, cars, swords, etc.). However, we noticed certain properties of the feature vector that aggravate its retrieval performance:

- Low-dimensional vectors do not capture sufficient information about the object;
- The l_p metric (1.9) is not effective in the spatial domain;
- Very similar 3D-models can have large differences between specific feature vector components.

Indeed, if we take a small number of samples (e.g., $k = 2$, $N = 42$), then not all the parts of a model are captured. For instance, in figure 4.5 the front legs of models are missed in both examples. However, if we increase the number of samples (i.e., the dimension of feature vector), then, in spite of capturing more information about the model, the retrieval performance usually decreases or stagnate (see section 5.2.1). As mentioned in section 1.4, the reason for the decrease of retrieval performance with the increase of vector dimension (the number of samples) is the ineffectiveness of the l_p (1.9) metric in the spatial domain.

Another problem is depicted in figure 4.5. Since the same model is represented in two levels of detail, the triangle meshes differ slightly. Therefore, the corresponding components of their feature vectors should approximately be the same. However, the back left leg of the bull is missed by all rays in one case (right), while it is intersected



$$\begin{aligned}
 \mathbf{f}_{400} = & (2.33, 1.15, 1.12, 0.84, 0.97, 0.67, 1.80, 0.67, 0.62, 1.09, 1.23, 1.63, 2.62, 1.39, \\
 & \mathbf{3.00}, 2.16, 1.10, 1.22, 1.02, 1.19, 1.26, 1.10, 0.96, 1.14, 0.79, 0.89, 0.76, 2.84, \\
 & 1.18, 1.75, 1.67, 1.65, 0.60, 1.07, 0.66, 0.56, 0.93, 0.61, 0.56, 1.29, 0.93, 1.83), \\
 \mathbf{f}_{800} = & (2.30, 1.16, 1.14, 0.83, 0.90, 0.64, 1.77, 0.69, 0.62, 1.09, 1.23, 1.65, 2.57, 1.43, \\
 & \mathbf{1.20}, 2.19, 1.15, 1.22, 1.04, 1.17, 1.25, 1.08, 0.97, 1.15, 0.76, 0.88, 0.72, 2.80, \\
 & 1.09, 1.73, 1.62, 1.64, 0.61, 1.10, 0.65, 0.53, 0.86, 0.60, 0.54, 1.29, 0.85, 1.82), \\
 (|\mathbf{f}_{400}^1 - \mathbf{f}_{800}^1|, \dots, |\mathbf{f}_{400}^{42} - \mathbf{f}_{800}^{42}|) = & \\
 & (0.03, 0.01, 0.02, 0.01, 0.07, 0.03, 0.03, 0.02, 0.00, 0.00, 0.00, 0.00, 0.02, 0.05, 0.04, \\
 & \mathbf{1.80}, 0.03, 0.05, 0.00, 0.02, 0.02, 0.01, 0.02, 0.01, 0.01, 0.03, 0.01, 0.04, 0.04, \\
 & 0.09, 0.02, 0.05, 0.01, 0.01, 0.03, 0.01, 0.03, 0.07, 0.01, 0.02, 0.00, 0.08, 0.05).
 \end{aligned}$$

Figure 4.5: A problem of the ray-based shape descriptor in the spatial domain. Ray-based feature vectors ($N = 42$) of models with 400 and 800 triangles are visualized. The feature vector instances of both models as well as componentwise differences are given. Regardless of high similarity between the models, vector components may significantly differ (bold values).

with one of the rays in the other case (left). This implies a large difference between the corresponding components of the feature vectors as shown in figure 4.5, where the component corresponding to the ray traveling in the direction of the back left leg is in bold.

Hence, it is desirable to increase significantly the number of samples of a model, but still to characterize its shape with feature vectors of reasonable dimensions. Also, large differences between components of feature vectors of very similar models should be avoided. To achieve these goals as well as to strengthen the discriminant power of the descriptor, we use a different representation of the ray-based feature. For instance, spherical wavelets [118] or spherical harmonics [45, 78] can be engaged. Spherical wavelets presented in [118] are based on the geodesic sphere construction starting with an icosahedron using the same subdivision as in figure 4.1. The ray-based feature with spherical harmonic representation is described in section 4.6.2.

4.2 Silhouette-Based Feature Vectors

Content-based image retrieval methods have intensively been studied in recent years [64, 15, 16, 109, 61, 62, 38, 63]. Various techniques have been explored, e.g., global color histograms [109], modifications of global histograms taking into account the structure of an image [89], edge histograms [89], automatic segmentation by color [15, 16], etc. However, none of these methods can be extended and applied to retrieve 3D-mesh models by shape. An approach to capture shape characteristics of objects in 2D-images is to analyze silhouettes (contours). According to [64], a definition of a silhouette might be:

The word silhouette indicates the region of a 2D-image of an object Ω , which contains the projections of the visible points of Ω .

A silhouette can also be defined as an outline of a solid object. We regard a contour as a collection of boundary points of a silhouette.

In [38] a contour is approximated by a polygonal line whose vertices are either points of extreme curvature or points which are added to refine the polygonal approximation. The obtained polygonal lines can be used for both global and local similarity search. For global matching of two contours all polygonal vertices are taken into account, while subsets of polygonal vertices obtained by an adequate processing step are used for finding local similarity between two contours. An example of local similarity search is matching of human limbs at different body poses. A dual approach to contour matching is based on medial axes of the silhouette boundary, e.g., “the chain of circles” presented in [18]. Both methods concern a typical computer vision problem where all the information is contained in a single 2D image of a 3D object. The image is usually taken from an arbitrary viewpoint. The techniques can be very effective in certain special cases, e.g., matching contours of a dog and a horse, which are regarded as similar, on images taken from different (but suitably chosen) viewpoints [38]. The disadvantage of both approaches is sensitivity to small perturbations of contour points, which represent “noise” in the polygonal approximation as well as medial axes. Besides, a matching of two images takes a few seconds, whence the techniques cannot be applied to searching collections with thousands of images interactively.

Silhouettes of 2D-shapes have a natural arc length parameterization. Since the parameterization cannot be extended to 3D surfaces of arbitrary genus, we define a 3D-shape descriptor based on 2D silhouettes. In our approach, a 3D-object in the CPCA frame (section 3.4) is projected perpendicularly on the coordinate hyperplanes, in order to generate three monochrome images as depicted in figure 4.6. The silhouette images are formed using a *canonical bounding cube* of 3D-model.

Definition 4.1 *The canonical bounding cube (CBC) of a 3D-mesh model, which is transformed into a canonical coordinate frame, is a cube whose set of vertices is given by $\{(x, y, z) \mid x, y, z \in \{-a_{max}, a_{max}\}\}$, with*

$$a_{max} = \max \left\{ \max_{1 \leq i \leq n} |x_i|, \max_{1 \leq i \leq n} |y_i|, \max_{1 \leq i \leq n} |z_i| \right\}, \quad (4.6)$$

where x_i , y_i , and z_i denote vertex coordinates (1.3) in the canonical frame.

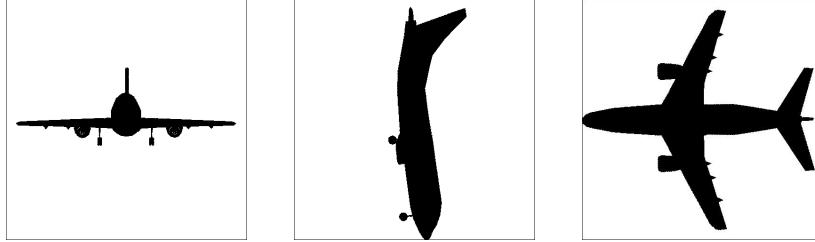


Figure 4.6: Silhouette images of an airplane model obtained by projecting the model on the coordinate hyper-planes yOz , zOy , and xOy , respectively.

Perpendicular projections of the CBC on the coordinate hyper-planes determine regions that are rasterized into images of dimensions $N \times N$. The coordinate origin O always lies at the center of the image. An example of rasterization as well as generation of a silhouette image of a triangle are depicted in figure 4.7. Suppose that triangle vertices are given by Cartesian coordinates so that $A = (a_x, a_y, a_z)$, $B = (b_x, b_y, b_z)$, and $C = (c_x, c_y, c_z)$. If the silhouette is generated on the yOz hyper-plane, then the vertex coordinates of the projected triangle are (a_y, a_z) , (b_y, b_z) , and (c_y, c_z) . The silhouette image of the triangle is obtained by filling the interior of the projected triangle. The union of silhouette images of all triangles of a mesh-model defines the silhouette image of the 3D-model.

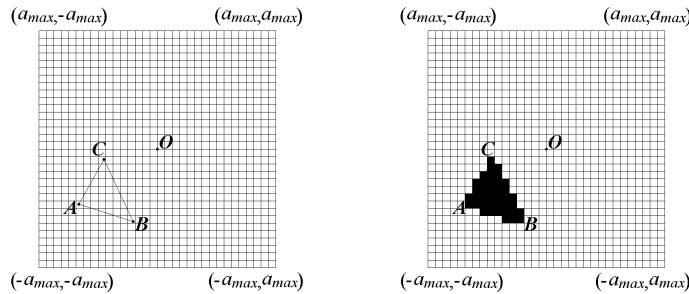


Figure 4.7: Generating a silhouette of a triangle.

The next step is finding the outer contour. We consider the image as an $N \times N$ matrix $C = [c_{ij}]$, where the elements c_{ij} ($i, j = 1, \dots, N$) denote pixel attributes of a monochrome image. Attribute 1 denotes background pixels (white) and attribute 0 denotes silhouette pixels (black). For fixed i and j , pixel c_{ij} is a contour point if its attribute is 0 and at least one of its four closest neighbors (east, south, west, and north pixels) belongs to the background. A contour Γ is made up of a cyclic sequence of contour points,

$$\Gamma = [c_{i_0j_0}, \dots, c_{i_{L-1}j_{L-1}}], \quad L \in \mathbb{N},$$

where L denotes the length of the contour, which depends on the silhouette image. Moreover, it holds

$$(i_p \neq i_q \vee j_p \neq j_q) \quad \wedge \quad \max\{|i_p - i_q|, |j_p - j_q|\} = 1,$$

where indices p and q can take values

$$q = (p + 1) \bmod L, \quad p = 0, \dots, L - 1.$$

For simplicity of notation, we associate a two-dimensional vector \mathbf{c}_p to the contour point (pixel) $c_{i_p j_p}$ at position (i_p, j_p) . We can also represent the contour Γ using the arc length parameterization,

$$\Gamma : \{0, 1, 2, \dots, L - 1\} \rightarrow \mathbb{R}^2, \quad \Gamma(p) \equiv (i_p, j_p) = \mathbf{c}_p. \quad (4.7)$$

In the case that there is more than one contour in the silhouette image, which may occur either if the object possesses holes or is composed from disjoint parts, we process the longest contour (with the largest L).

Since the contour length L is non constant, we select K points to process further, where K is fixed. The selected points are used for forming a sequence S_K defined by

$$S_K = \{ \mathbf{s}_0, \dots, \mathbf{s}_{K-1} \}, \quad \mathbf{s}_i \in \left\{ \frac{a_{\max}}{N} (\mathbf{c}_1 - O), \dots, \frac{a_{\max}}{N} (\mathbf{c}_L - O) \right\} \cup \{(0, 0)\}, \quad (4.8)$$

where $O = (N/2, N/2)$ is the center of the image and a_{\max} is defined by (4.6). The multiplication of contour points by a_{\max}/N is necessary in order to provide invariance with respect to outliers. We tested the following two methods for selecting points from the set $\Gamma \cup \{O\}$, in order to generate the sequence S_K (4.8):

- a) *Adjacent selected points have constant distance along the contour.*

The points \mathbf{s}_i (4.8) are determined by

$$\mathbf{s}_i = \frac{a_{\max}}{N} \left(\Gamma \left(\left\lfloor \frac{iL}{K} \right\rfloor \right) - O \right), \quad (4.9)$$

where $\lfloor t \rfloor \in \mathbb{N}$ denotes the greatest integer that is not greater than $t \in \mathbb{R}$.

- b) *Polar angles of adjacent selected points differ by a constant.*

Let $(\rho(\mathbf{c}_p), \varphi(\mathbf{c}_p))$, $\rho(\mathbf{c}_p) \geq 0$, $-\pi < \varphi(\mathbf{c}_p) \leq \pi$, be coordinates of the contour point $\mathbf{c}_p = (i_p, j_p)$ in the polar coordinate system with the origin O , i.e.,

$$\mathbf{c}_p = (i_p, j_p) = O + \rho(\mathbf{c}_p) (\cos \varphi(\mathbf{c}_p), \sin \varphi(\mathbf{c}_p)).$$

The points \mathbf{s}_i (4.8) are determined by

$$\mathbf{s}_i = \begin{cases} (0, 0), & \Psi_i = \emptyset, \\ \frac{a_{\max}}{N} (\mathbf{c}_p - O), & \rho(\mathbf{c}_p) = \max \Psi_i, \end{cases} \quad \Psi_i = \left\{ \rho(\mathbf{c}_q) \mid \varphi(\mathbf{c}_q) \approx \frac{2i\pi}{K} \right\}. \quad (4.10)$$

Basically, we apply a similar approach as in the case of the ray-based feature vector, but in the two-dimensional space. We intersect the contour with the ray emanating from the origin O and traveling in the direction $(\cos(2i\pi/K), \sin(2i\pi/K))$. If the intersection exists, then the furthest intersection point of contour is selected to form \mathbf{s}_i . Otherwise we set $\mathbf{s}_i = (0, 0)$.

Note that the coordinates of \mathbf{s}_i ($0 \leq i \leq K - 1$) are given in the coordinate system with the origin O .

Both approaches are visualized in figure 4.8. Each silhouette is approximated by a polygonal line formed by the points, which are used for forming the sequence S_K (4.8). As it can be seen, polygonal approximations significantly deviate from the contour, when points are selected by (4.10). Nevertheless, experimental results (see section 5.2.2) suggest that selecting the contour points that have uniform polar angles (4.10) is the better choice. The results are expected because method (4.9) is more sensitive to local deformations of contours (noise).

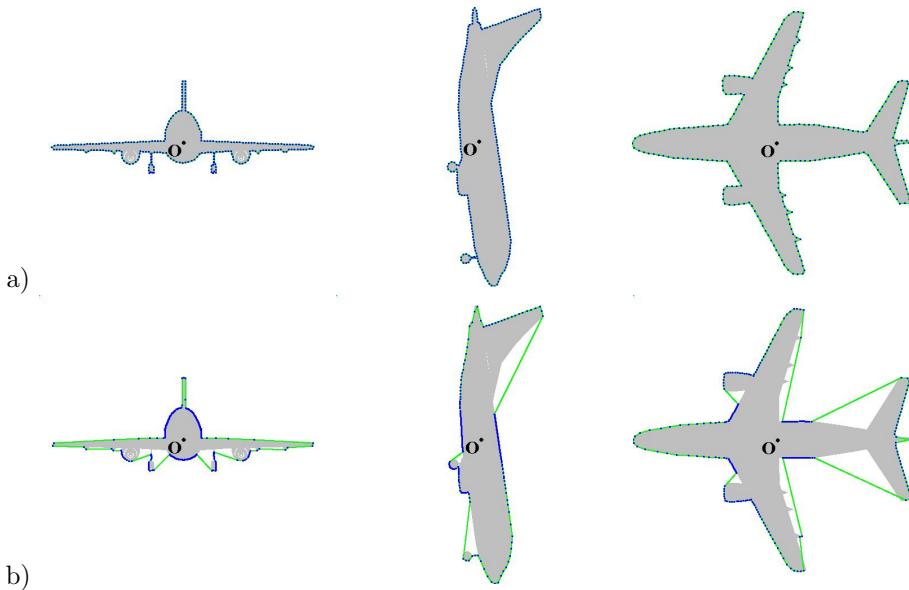


Figure 4.8: Extraction of the silhouette-based shape descriptor: a) adjacent selected points have constant distance along the contour (4.9), b) polar angles of adjacent selected points differ by a constant (4.10).

Points $\mathbf{s}_i = \rho_i(\cos \varphi_i, \sin \varphi_i)$ (4.8) represent the contour as a feature in the spatial domain. There are several possibilities to process the information contained in them. Two sets of contour points can be matched in the spatial domain using an appropriate similarity measure. Some authors [95] use values ρ_i to form a histogram for each contour engaging the l_1 distance (1.10) to calculate dissimilarity

between two histograms. However, experimental results presented in [95] show a very poor performance of this approach. Intuitively, this is expected, because by binning distances of contour points to the origin we loose information about their relative positions. We recall that the l_1 norm is usually not effective distance metric for features represented in the spatial domain (see section 1.4). Our choice is to represent the feature in the spectral domain using the discrete Fourier transform (DFT). The transform of a discrete sequence of complex values $\{f_0, f_1, \dots, f_{K-1}\}$ is given by

$$\hat{f}_p = \frac{1}{\sqrt{K}} \sum_{i=0}^{K-1} f_i e^{-j \frac{2\pi}{K} i \cdot p}, \quad p = 0, \dots, K-1, \quad (4.11)$$

where j is the imaginary unit and $\hat{f}_p \in \mathbb{C}$ are Fourier coefficients. If K is a power of 2, then a fast Fourier transform (FFT) algorithm [148] can be used. The complexity $\mathcal{O}(K^2)$ of the DFT can significantly be reduced with the FFT algorithm ($\mathcal{O}(K \log_2 K)$). We tested the following three possibilities for defining the values f_i using the selected contour points \mathbf{s}_i :

1. Consider \mathbf{s}_i as points in the complex plane,

$$f_i = \rho_i \cdot e^{j\varphi_i}; \quad (4.12)$$

2. Transform only distances of \mathbf{s}_i from the origin,

$$f_i = \rho_i; \quad (4.13)$$

3. Transform x -coordinates and y -coordinates of \mathbf{s}_i separately, i.e.,

$$\text{first we transform } f_i = \rho_i \cos \varphi_i \text{ and then } f_i = \rho_i \sin \varphi_i. \quad (4.14)$$

After the evaluation (section 5.2.2), we decided to use the second approach (4.13). The absolute values of the obtained coefficients, $|\hat{f}_p|$ (4.11), are used to form the *silhouette-based* feature vector. Namely, for each of the three contours, we take the first k absolute values of coefficients as components of the vector, whence the dimension of the vector is equal to $3k$. More precisely, if $\{\hat{f}_0^{(1)}, \dots, \hat{f}_{K-1}^{(1)}\}$, $\{\hat{f}_0^{(2)}, \dots, \hat{f}_{K-1}^{(2)}\}$, and $\{\hat{f}_0^{(3)}, \dots, \hat{f}_{K-1}^{(3)}\}$ are the coefficients obtained from three silhouettes, then the components of the corresponding feature vector \mathbf{f} ($\dim(\mathbf{f}) = 3k$) are defined in the following way

$$\mathbf{f} = \left(|\hat{f}_0^{(1)}|, |\hat{f}_0^{(2)}|, |\hat{f}_0^{(3)}|, |\hat{f}_1^{(1)}|, |\hat{f}_1^{(2)}|, |\hat{f}_1^{(3)}|, \dots, |\hat{f}_{k-1}^{(1)}|, |\hat{f}_{k-1}^{(2)}|, |\hat{f}_{k-1}^{(3)}| \right). \quad (4.15)$$

By forming vector components according to (4.15), an *embedded multi-resolution representation* (1.8) of the feature is provided, i.e., the vector of dimension $3k$ contains all vectors of lower dimensions. Note that, from (4.11), we have

$$\{f_0, \dots, f_{K-1}\} \in \mathbb{R} \implies \hat{f}_p = \overline{\hat{f}_{K-p}}, \quad p = 1, \dots, K/2 - 1, \quad (4.16)$$

i.e., the coefficients \hat{f}_p and \hat{f}_{K-p} are complex conjugate, whence their absolute values are equal, $|\hat{f}_p| = |\hat{f}_{K-p}|$. Thus, we propose to choose the values of k and K so that $k \leq K/2$.

We stress an interesting property of the DFT, which makes the magnitudes of obtained coefficients (approximately) invariant with respect to rotation of the underlying silhouette image. Suppose that $\{f_0, \dots, f_{K-1}\}$ is the original sequence of samples, and $\{g_0, \dots, g_{K-1}\}$ is a sequence of samples after rotating the original silhouette image for an arbitrary angle ϕ . Approximately, the original samples are shifted so that

$$g_{i'} \approx f_i, \quad 0 \leq i, i' \leq K-1, \quad (4.17)$$

where

$$i' = (i - t) \bmod K, \quad t = \left\lceil \frac{K\phi}{2\pi} \right\rceil \in \mathbb{N}, \quad (t \text{ is a rounded value}).$$

Then, the absolute values of Fourier coefficients \hat{f}_p , obtained by transforming the original sequence, are approximately equal to the absolute values of coefficients \hat{g}_p , obtained by transforming the shifted sequence. The higher the value of K , the better the approximation. Indeed, using (4.11) and (4.17), we have

$$\begin{aligned} \hat{f}_p &= \frac{1}{\sqrt{K}} \sum_{i=0}^{K-1} f_i e^{-j2\pi ip/K} \\ &= e^{-j2\pi tp/K} \frac{1}{\sqrt{K}} \sum_{i=0}^{K-1} f_i e^{-j2\pi(i-t)p/K} \quad \Rightarrow \quad |\hat{f}_p| \approx |\hat{g}_p|. \quad (4.18) \\ &\approx e^{-j2\pi tp/K} \frac{1}{\sqrt{K}} \sum_{i=0}^{K-1} g_i e^{-j2\pi ip/K} \\ &= e^{-j2\pi tp/K} \cdot \hat{g}_p \end{aligned}$$

As an example, the rotated silhouette images, which are shown in figure 4.9, will be characterized by approximately the same magnitudes of Fourier coefficients. We stress that it can also be proven that the magnitudes of obtained coefficients are invariant with respect to reflections of a mesh around the coordinate hyper-planes. Therefore, it is not necessary to reflect a mesh model by multiplying its vertices with the matrix F (3.23), during the normalization procedure (section 3.4).

We tested the presented approach for $N, K \in \{128, 256, 512, 1024\}$ (16 different settings), and we suggest to take $N = 256$ and $K = 256$ (see section 5.2.2). In our implementation, we set $k = 100$, i.e., the largest dimension of the vector is 300, embedding the vectors of dimensions 297, 294, ..., 3. With these settings, the average feature extraction time for models from the MPEG-7 collection (section 5.1), which have already been positioned in the canonical coordinate frame, amounts 33.4 ms (on a PC with an 1.4 GHz AMD processor running Windows 2000).

The silhouette-based 3D-shape feature vector fulfills the requirements described in section 1.3.4. We stated that, if it is not possible to determine a single unique contour of the silhouette image (for 3D-models with holes or disjoint parts), only

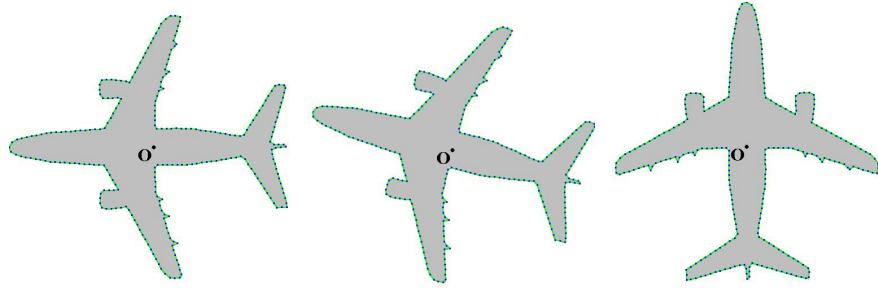


Figure 4.9: If a silhouette image is rotated, the magnitudes of Fourier coefficients remain the same 4.18.

the longest contour is processed. Therefore, certain parts of 3D-objects might not be described. We stress that method (4.10) can successfully be applied to models whose silhouettes comprises more than one contour. Instead of a single contour, the union of all contours is processed, while the presented procedure remains the same.

4.3 Depth Buffer-Based Feature Vector

When we create silhouette images of 3D-objects all the information about shape is contained in contour points, because each interior point of the silhouette has the same attribute. Therefore, we considered other approaches for creating 2D images from 3D-objects in order to capture 3D-shape characteristics. We define another feature vector, which is obtained from six depth-buffer (or Z-buffer) images. Depth-buffers are formed using the faces of an appropriate cuboid region. We use the depth-buffer algorithm which is described in [30]. Each depth-buffer is associated to one face of the cuboid region, belonging the front clipping plane, while the back clipping plane contains the parallel face. Initially, all values of the depth-buffer are set to 0, representing the value at the back clipping plane. The largest value that can be stored in the depth-buffer is 1 and represents the value at the front clipping plane. The exact way of forming the depth buffer images is given in a sequel.

Let ρ be a cuboid region, whose faces are used for defining depth buffer images,

$$\rho = \{ (x, y, z) \mid x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}, z_{min} \leq z \leq z_{max} \}. \quad (4.19)$$

Let the face, determined by vertices

$$(x_{min}, y_{min}, z_{min}), (x_{min}, y_{min}, z_{max}), (x_{min}, y_{max}, z_{max}), \text{ and } (x_{min}, y_{max}, z_{min}), \quad (4.20)$$

belong to the front clipping plane. The depth-buffer image, which corresponds to the face (4.20), is formed in the following way. Firstly, the face is subdivided (rasterized) into $N \times N$ coincident rectangles (or squares, if ρ is a cube). Each rectangle (square) corresponds to a pixel of a gray scale image of dimensions $N \times N$.

Let v_{ab} be the attribute of the pixel at position (a, b) , $0 \leq a, b \leq N - 1$. The value of v_{ab} is associated to the rectangle (square) ν_{ab} determined by

$$\nu_{ab} = \{(x_{min}, y, z) \mid y_a \leq y < y_a + d_y, z_b \leq z < z_b + d_z\}, \quad (4.21)$$

where

$$\begin{aligned} y_a &= y_{min} + a \cdot d_y, & d_y &= (y_{max} - y_{min})/N, \\ z_b &= z_{min} + b \cdot d_z, & d_z &= (z_{max} - z_{min})/N. \end{aligned}$$

All pixel attributes are set to an initial value ($v_{ab} = 0$). A point $P = (x_P, y_P, z_P) \in I$ of a 3D-mesh model I (1.5), which lies inside the region ρ ($P \in \rho$), is orthogonally projected on the face (4.20). The projection P' is determined by coordinates (x_{min}, y_P, z_P) . If $P' \in \nu_{ab}$, then the attribute v_{ab} is updated,

$$v_{ab} \leftarrow \max \left\{ v_{ab}, \frac{x_{max} - x_p}{x_{max} - x_{min}} \right\}. \quad (4.22)$$

Note that $v_{ab} \in [0, 1]$. An example of the depth buffer image of a single triangle is depicted in figure 4.10. Black pixels correspond to the value of 0, white pixels are attributed the value of 1. Since the pixel corresponding to the point A is almost white, the point A is the closest to the front clipping plane ($x = x_{min}$). The depth buffer image of a single triangle is formed by computing the attributes corresponding to the vertices of the triangle, and by filling the interior points. The depth-buffer image of the whole 3D-mesh model is obtained by taking maximal values v_{ab} ($0 \leq a, b \leq N - 1$) of depth-buffer attributes of all triangles. The remaining five depth buffer images are analogously generated, by substituting the front clipping plane (4.20).

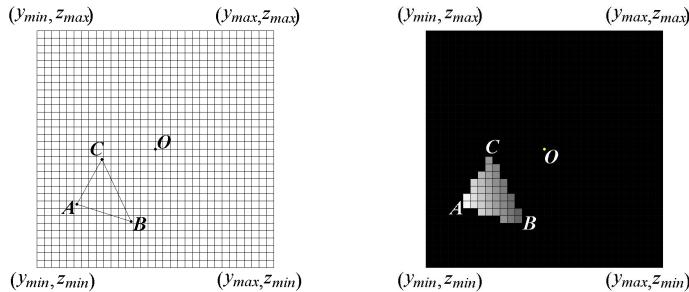


Figure 4.10: Generation of a depth-buffer image of a triangle (right), using the face determined by (4.20).

The choice of the cuboid region ρ (4.19) is an interesting problem. Note that only the part of a 3D-mesh model being inside ρ is processed, while a subset of I (1.5), which is outside ρ , is ignored. As a first choice, we take the canonical bounding cube (CBC) (see definition 4.1) as the region ρ . We also use faces of two other cubes, *extended bounding box* and *canonical cube*, to form the depth buffer images. Firstly, we define the *bounding box* of a 3D-model.

Definition 4.2 *The bounding box (BB) of a model I (1.5) is the tightest box, with the faces parallel to the coordinate hyper-planes, which encloses the whole model. The set of vertices of the bounding box is defined by*

$$\{ (x, y, z) \mid x \in \{x_1, x_2\}, y \in \{y_1, y_2\}, z \in \{z_1, z_2\} \},$$

$$\begin{aligned} x_1 &= \min_{1 \leq i \leq n} x_i, & y_1 &= \min_{1 \leq i \leq n} y_i, & z_1 &= \min_{1 \leq i \leq n} z_i, \\ x_2 &= \max_{1 \leq i \leq n} x_i, & y_2 &= \max_{1 \leq i \leq n} y_i, & z_2 &= \max_{1 \leq i \leq n} z_i, \end{aligned} \quad (4.23)$$

where (x_i, y_i, z_i) are the vertices of the model (1.3).

It is desirable that the region ρ (4.19) is a cube, in order to keep the correct aspect ratio of depth buffer images. Hence, the bounding box is not suitable for forming depth buffer images, because its faces are rectangular, in general. Therefore, we extend the bounding box to a cube, whose center coincide with the center of bounding box.

Definition 4.3 *The extended bounding box (EBB) of a 3D-model I (1.5) is defined by the set of vertices*

$$\begin{aligned} \{ (x, y, z) \mid x \in \{c_x - w, c_x + w\}, y \in \{c_y - w, c_y + w\}, z \in \{c_z - w, c_z + w\} \}, \\ w = \max \left\{ \frac{x_2 - x_1}{2}, \frac{y_2 - y_1}{2}, \frac{z_2 - z_1}{2} \right\}, \\ c_x = \frac{x_1 + x_2}{2}, \quad c_y = \frac{y_1 + y_2}{2}, \quad c_z = \frac{z_1 + z_2}{2}, \end{aligned} \quad (4.24)$$

where x_1, x_2, y_1, y_2, z_1 , and z_2 are given by 4.23.

Using the CBC (definition 4.1) or EBB (definition 4.3), for forming depth buffer images, leads to potential problems with outliers. Obviously, an outlier affects the dimensions of both CBC and EBB, whence the depth buffer images are affected by the outlier, as well. In order to reduce the influence of outliers, we define the canonical cube of a 3D-model.

Definition 4.4 *The canonical cube (CC) of a 3D-model I (1.5) is a cube in the canonical coordinate frame, defined by the set of vertices*

$$\{ (x, y, z) \mid x, y, z \in \{-w, w\} \}, \quad w \in \mathbb{R}. \quad (4.25)$$

Thus, the length of edges of the CC is fixed (constant), $2w$, the center of the CC lies at the origin of the canonical coordinate frame, and the faces are parallel to the coordinate hyper-planes. Note that both the CBC and EBB are specific for each 3D-model, while the CC is constant, i.e., equal for all models. Using the cube of constant size in the canonical frame, for forming the depth-buffers, lessens the impact of outliers on extracted feature vectors. A difficult problem is the selection

of the constant value w . If we choose w to be large enough so that all models lie inside the cube, then the depth-buffer images are mostly black, i.e., models occupy only a small area in the middle of each image. We expect that discriminant power of descriptors, relying upon depth buffer images that are mostly black, is poor. On the other hand, if w is too small, then a significant part of a model may lie outside the canonical cube, whence the most of model is clipped. Consequently, the feature vector cannot capture information about all parts of the model. An experimental analysis for $w = 2$, $w = 4$, $w = 8$, and $w = 16$ is presented in section 5.2.3.

The depth-buffer images of dimensions $N \times N$ pixels can directly be used as a feature in the spatial domain. An example is shown in figure 4.11, where the underlying model is the same as in figure 4.8. The six depth-buffers are visualized in the first row as the grayscale images. The attributes of all six $N \times N$ images can be taken as components of a feature vector. The dimension of the corresponding feature vector is $6N^2$. As mentioned in section (1.4), the l_1 or l_2 norms are not suitable for calculating distances between feature vectors in the spatial domain. In order to correlate components of vectors in the spatial domain, a suitable distance metric need to be defined. Another solution is to correlate spatial features by transforming them into the spectral domain. Therefore, we use the two-dimensional discrete Fourier transform (2D-DFT) of depth-buffers to represent the feature in the spectral domain. Briefly, for a two-dimensional sequence of complex numbers $f_{ab} \in \mathbb{C}$, $a = 0, \dots, M - 1$, $b = 0, \dots, N - 1$, the Fourier coefficients $\hat{f}_{pq} \in \mathbb{C}$ are calculated by

$$\hat{f}_{pq} = \frac{1}{\sqrt{MN}} \sum_{a=0}^{M-1} \sum_{b=0}^{N-1} f_{ab} e^{-j2\pi(pa/M + qb/N)}, \quad (4.26)$$

where j is the imaginary unit and $p = 0, \dots, M - 1$, $q = 0, \dots, N - 1$. If we apply the formula (4.26) directly, then the computational complexity of the 2D-DFT is $\mathcal{O}(M^2N^2)$. Because of the separability, we can reduce the DFT from a 2-dimensional operation to two 1-dimensional operations. Separability can be expressed by re-writing (4.26),

$$\hat{f}_{pq} = \frac{1}{\sqrt{M}} \sum_{a=0}^{M-1} \underbrace{\left(\frac{1}{\sqrt{N}} \sum_{b=0}^{N-1} f_{ab} e^{-j2\pi qb/N} \right)}_{\text{1D-DFT (4.11)}} e^{-j2\pi pa/M},$$

Hence, we should not apply the formula (4.26) directly, but rather two consecutive 1D-DFTs (4.11). First we compute the DFT of the columns of an image and then follow up with the DFT of the rows (or vice versa). Therefore, the complexity is reduced to $\mathcal{O}(MN(M + N))$. If M and N are powers of two, then the FFT can be applied speeding up the computation additionally. The computational complexity of the 2D-FFT (when two consecutive FFTs are applied) is of order $\mathcal{O}(MN \log(MN))$. The achieved speed up of computation using the 2D-FFT over the approaches when two consecutive DFTs are performed, and when the formula (4.26) is directly applied, is depicted in table 4.3.

M	N	$\frac{\mathcal{O}(MN(M+N))}{\mathcal{O}(MN \log(MN))}$	$\frac{\mathcal{O}(M^2N^2)}{\mathcal{O}(MN \log(MN))}$
64	64	10.7	341.3
128	128	18.3	1170.3
256	256	32.0	4096.0
512	512	56.9	14563.6
1024	1024	102.4	52428.8

Table 4.3: Approximate speed up for various image dimensions when two FFTs are used instead of other two approaches for computing the 2D-DFT.

In our case, the attributes $v_{ab} \in [0, 1]$ (4.22) of a square image of type $N \times N$ ($M = N$) serve as the input for the 2D-FFT. We stress that N should be a power of 2, in order to apply the fast Fourier transform. Before the 2D-FFT (4.26) is applied, we set

$$f_{ab} = v_{a'b'}, \quad (4.27)$$

where $a = (a' + N/2) \bmod N$, $b = (b' + N/2) \bmod N$, and $0 \leq a', b' \leq N - 1$. Practically, the new array f_{ab} is obtained by cyclically shifting the original one v_{ab} so that $f_{00} = v_{N/2,N/2}$. After applying the 2D-FFT, the obtained array of complex coefficients \hat{f}_{pq} is also shifted so that $\hat{v}_{N/2,N/2} = \hat{f}_{00}$. The new array \hat{v}_{pq} ($0 \leq p, q \leq N - 1$) represents the feature in the spectral (frequency) domain. The visualizations of arrays \hat{v}_{pq} by the grayscale images in the second row of figure 4.11, each of which corresponds to the depth-buffer image above, is done by taking $\min\{1, |\hat{v}_{pq}| \}$ as pixel attributes (grayscale values). Thus, Fourier coefficients with lower-frequencies are depicted by pixels that are located in the middle of the image.

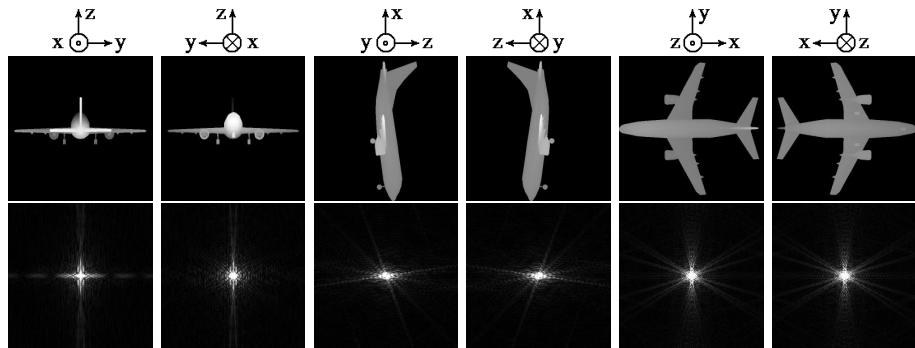


Figure 4.11: Extraction of the depth buffer-based shape descriptor. In the first row, the depth-buffer images are formed using the canonical bounding cube (definition 4.1). In the second row, each depth-buffer is transformed using the 2D-FFT (4.26).

Since the input sequence of the 2D-DFT consists of real numbers v_{ab} , the obtained coefficients \hat{v}_{pq} possess the symmetry property, similar to (4.16), which fol-

lows from (4.26),

$$v_{ab} \in \mathbb{R} \Rightarrow \hat{v}_{pq} = \overline{\hat{v}_{p'q'}}, \quad (p + p') \bmod N = 0, \quad (q + q') \bmod N = 0, \quad (4.28)$$

where $\overline{\hat{v}_{p'q'}}$ is the complex conjugate of $\hat{v}_{p'q'}$, $0 \leq a, b, p, q, p', q' \leq N - 1$. The distribution of symmetrical Fourier coefficients of an $N \times N$ image is shown in figure 4.12. The coefficients marked by gray color can be calculated using the symmetry property.

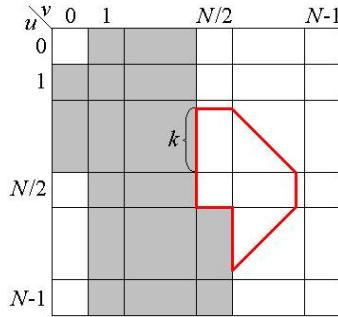


Figure 4.12: Coefficients marked by gray color can be calculated using the symmetry 4.28 of the 2D-DFT. The coefficients that are included in the feature vectors are located inside the denoted boundaries.

The depth-buffer feature vector is represented in the spectral domain as follows. The absolute values of coefficients \hat{v}_{pq} corresponding to the white area in figure 4.12, whose indices satisfy the inequality

$$|p - N/2| + |q - N/2| \leq k \leq N/2, \quad (k \in \mathbb{N}), \quad (4.29)$$

are taken as components of the feature vector. The coefficients are located inside the boundaries depicted in figure 4.12. The total number of coefficients inside the marked area is $k^2 + k + 1$, whence the dimension of the vector obtained using the six depth-buffers is $6(k^2 + k + 1)$. For $k = 8$, the vector possesses 438 components. Note that the vectors with 342, 258, 186, 126, 78, and 42 components (i.e., for $k = 7, 6, 5, 4, 3, 2$) can easily be embedded in the vector of dimension 438.

Let the six depth-buffer images be uniquely indexed by $i \in \{1, \dots, 6\}$, and let $\hat{v}_{pq}^{(i)}$ ($0 \leq p, q \leq N - 1$) be the Fourier coefficients of the depth-buffer image indexed by i . As an example, the components of the feature vector \mathbf{f} of dimension 78 ($k = 3$)

are defined in the following way

$$\mathbf{f} = (|\hat{v}_{c,c}^{(1)}|, \dots, |\hat{v}_{c,c}^{(6)}|, |\hat{v}_{c,c-1}^{(1)}|, \dots, |\hat{v}_{c,c-1}^{(6)}|, |\hat{v}_{c+1,c}^{(1)}|, \dots, |\hat{v}_{c+1,c}^{(6)}|, \\ |\hat{v}_{c,c-2}^{(1)}|, \dots, |\hat{v}_{c,c-2}^{(6)}|, |\hat{v}_{c+1,c-1}^{(1)}|, \dots, |\hat{v}_{c+1,c-1}^{(6)}|, |\hat{v}_{c+2,c}^{(1)}|, \dots, |\hat{v}_{c+2,c}^{(6)}|, \\ |\hat{v}_{c+1,c+1}^{(1)}|, \dots, |\hat{v}_{c+1,c+1}^{(6)}|, |\hat{v}_{c,c-3}^{(1)}|, \dots, |\hat{v}_{c,c-3}^{(6)}|, |\hat{v}_{c+1,c-2}^{(1)}|, \dots, |\hat{v}_{c+1,c-2}^{(6)}|, \\ |\hat{v}_{c+2,c-1}^{(1)}|, \dots, |\hat{v}_{c+2,c-1}^{(6)}|, |\hat{v}_{c+3,c}^{(1)}|, \dots, |\hat{v}_{c+3,c}^{(6)}|, |\hat{v}_{c+2,c+1}^{(1)}|, \dots, |\hat{v}_{c+2,c+1}^{(6)}|, \\ |\hat{v}_{c+1,c+2}^{(1)}|, \dots, |\hat{v}_{c+1,c+2}^{(6)}|), \quad (4.30)$$

where $c = N/2$. Thus, the *embedded multi-resolution representation* (1.8) is provided, as it is the case with the silhouette-based descriptor (section 4.2).

We set the parameter N to 64, 128, 256, and 512, whereby the average feature extraction times of the depth-buffer descriptor, for models from the MPEG-7 collection (section 5.1), are 41.6 ms, 85.6 ms, 262.8 ms, and 1048.1 ms, respectively (on a PC with an 1.4 GHz AMD processor running Windows 2000). Since N is a power of 2, the 2D-FFT can be used producing a fast feature extraction. According to the evaluation results (section 5.2.3), we recommend to set $N = 256$ and to use depth buffer-based vectors of dimension 438.

The choice of the region ρ (4.19), CBC (definition 4.1), EBB (definition 4.3), or CC (definition 4.4), is an interesting problem. As stated earlier, the negative influence of *outliers* is present when the CBC or EBB are used. Thus, the requirement 5 from section 1.3.4 is fulfilled only if the CC is used. To demonstrate the problem, we give three retrieval examples in figure 4.13, where the query model possesses an outlier. The query model is obtained by adding a long antenna to the original model of car. When the CBC and EBB are used for forming depth buffer images, the top ranked models are not relevant to the query. However, if CC with $w = 2$ is used, robustness with respect to outliers is achieved, and all top ranked models are relevant. Note that the first match is the original model, which has been used to create the query object. We stress that the displayed thumbnail images in figure 4.13 serve just to visualize the models, i.e., they do not correspond to depth buffer images. Models in figure 4.13 are visualized from the positive side of the z -axis in the canonical coordinate frame, while the x -axis travels to the right. The distance of a viewpoint is selected so that the corresponding thumbnail image captures the whole 3D-object, whence the canonical scale is not depicted in figure 4.13. The l_1 distances between the query and the top ranked models are given, as well.

Regardless of the demonstrated sensitivity with respect to outliers, the results (section 5.2.3) suggest that it is better to use the EBB, than the CC, for any value of w . We assume that these results are caused by non-significant presence of outliers in the 3D-mesh models from collections, which are described in section 5.1. If outliers are more frequent, then we recommend to use the CC for generating depth buffer images.

Feature vector: Depth Buffer (EBB), dimension: 438, used distance: l_1				
query model	match no. 1	match no. 2	match no. 3	match no. 4
				
82porsch_outlier	A3dcell3 289.55	Phone3d 344.81	Flowers 383.04	Woman1a 386.54

Feature vector: Depth Buffer (CBC), dimension: 438, used distance: l_1				
query model	match no. 1	match no. 2	match no. 3	match no. 4
				
82porsch_outlier	A3dcell1 258.59	Tank 270.92	T80 311.18	Clayvase 319.47

Feature vector: Depth Buffer (CC, w=2), dimension: 438, used distance: l_1				
query model	match no. 1	match no. 2	match no. 3	match no. 4
				
82porsch_outlier	82porsch 27.88	Porsche 152.39	Porsche 163.73	Jaguar 165.13

Figure 4.13: When the CBC (definition 4.1) and EBB (definition 4.3) are used for forming depth buffer images, the descriptor is sensitive with respect to outliers. In the case when the CC (definition 4.4) is used, robustness with respect to outliers is achieved.

4.4 Volume-Based Feature Vector

As mentioned in section 4.1 (see figure 4.5), if the ray-based approach is used in the spatial domain, very similar 3D-models may have feature vectors whose components significantly differ. We recall that the problem is caused by relying upon a one-dimensional feature – extent along a given direction. The problem is also depicted in figure 4.14a). The triangles T and T' , which belong to two very similar objects, are positioned at slightly different locations in the canonical coordinate frame. The ray, emanated from the origin O in the direction \mathbf{u} , intersects only the triangle T at the point \mathbf{p} . If the components of a feature vector are formed using distances from intersection points to the origin, as it is the case with the ray-based approach, the values of the components corresponding to the directional vector \mathbf{u} significantly

differ.

A similar problem may occur when calculating surface area inside certain regions. For instance, let the plane δ in figure 4.14b) separate two regions and let the triangles T and T' be almost parallel to δ , but located on the opposite sides of the plane. If we have a feature vector in the spatial domain, defined by accumulating the areas of triangles inside the regions and attributing the obtained values to corresponding vector components, then the areas of T and T' are accumulated in different components of the feature vectors. As a result, we can have a gap between vector components of very similar models.

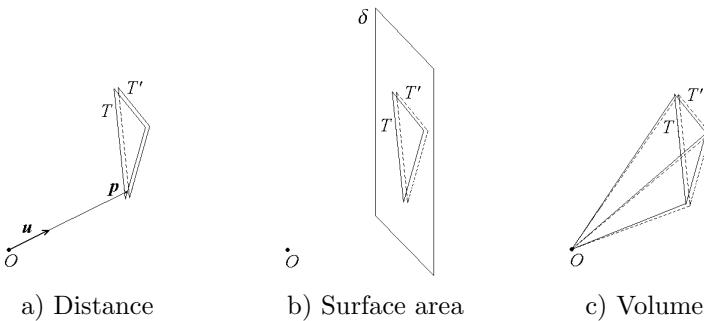


Figure 4.14: Large differences of sample values occur mostly when intersecting rays with triangles (a) and rarely when calculating the surface area inside certain regions (b). The differences cannot be large when relying upon volumes (c).

Hence, in the spatial domain, the gaps between corresponding sample values of similar 3D-objects are the most frequent when calculating features based on one-dimensional properties (e.g., ray casting), and are potentially present when computing features based on two-dimensional characteristics (e.g., surface area). This fact motivates us to define a feature relying upon a three-dimensional characteristic, volume. Since a 3D-mesh model is not necessarily a solid object, we define artificial volumes as shown in figure 4.14c). Namely, each triangle of the mesh is considered to be the base of a pyramid having the top vertex at the coordinate origin. We calculate the volume of each pyramid taking into account the orientation of the base, i.e., the volume can be negative. The signed volume V_{T_j} , which is associated to the triangle T_j (1.2) with the vertices \mathbf{p}_{A_j} , \mathbf{p}_{B_j} , and \mathbf{p}_{C_j} ($j = 1, \dots, m$), is calculated by

$$V_{T_j} = \mathbf{p}_{A_j} \cdot ((\mathbf{p}_{B_j} - \mathbf{p}_{A_j}) \times (\mathbf{p}_{C_j} - \mathbf{p}_{A_j})) \quad (4.31)$$

The reason to determine the sign of pyramid's volume is depicted in figure 4.15. Triangles $\Delta \mathbf{p}_1 \mathbf{p}_2 \mathbf{p}_3$ and $\Delta \mathbf{p}_2 \mathbf{p}_4 \mathbf{p}_3$ belong to a mesh model. For simplicity of illustration, the points O , \mathbf{p}_4 , and \mathbf{p}_1 are taken to be co-linear. The volume limited by the triangles, which is "inside" the model, is depicted on the right side of the figure. The value of volume is computed by summing appropriate scalar triple products of three vectors being the signed values of volumes.

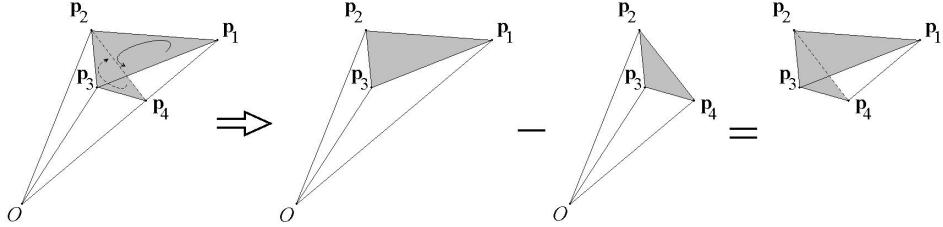


Figure 4.15: Calculation of volume.

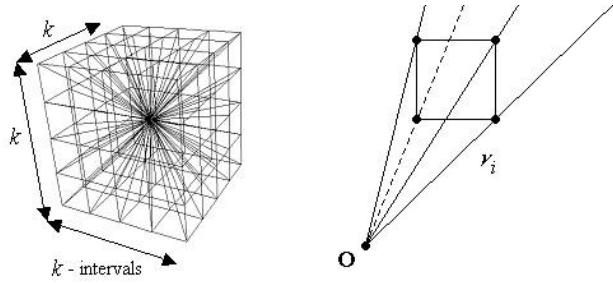
In order to form a feature vector from the calculated volumes, the whole 3D space (in the CPCPA frame) is subdivided into N disjunctive regions ν_1, \dots, ν_N . Our *volume based feature vector in spatial domain*

$$\mathbf{f} = (f_1, \dots, f_N) \quad (4.32)$$

is formed by calculating distribution of V_{T_j} ($j = 1, \dots, m$) across the regions so that the vector component f_i is equal to the total (artificially defined) volume belonging to the region ν_i . It holds,

$$V_{total} = \sum_{i=1}^N f_i = \sum_{j=1}^m V_{T_j}. \quad (4.33)$$

We subdivide the 3D-space into $N = 6k^2$ disjunctive regions by using the canonical cube with $w = 1$ (see definition 4.4). We regard the CC with $w = 1$ as the unit cube. Each face of the unit cube is subdivided into k^2 coincident squares, each of which represents the base of a pyramid with the top vertex at the origin, as depicted in figure 4.16. By scaling each pyramid by a large factor $w \rightarrow \infty$, the whole 3D-space is subdivided into $6k^2$ pyramid-like regions.

Figure 4.16: Subdivision of the 3D-space into $6k^2$ regions ν_i ($i = 1, \dots, 6k^2$).

Analytical definition of the region ν_i ($i = 1, \dots, 6k^2$) can be formulated as follows. Let $c \in \{0, 1, 2, 3, 4, 5\}$ denote the faces of the unit cube, so that faces lying

in the planes $x = 1$, $x = -1$, $y = 1$, $y = -1$, $z = 1$, and $z = -1$, are indexed by 0, 1, 2, 3, 4, and 5, respectively. We compute c by

$$c = \left\lfloor \frac{i-1}{k^2} \right\rfloor. \quad (4.34)$$

Let $s_1, s_2, s_3 \in \{1, 2, 3\}$ denote the coordinate axes, so that the axes x , y , and z are indexed by 1, 2, and 3, respectively. We compute

$$s_1 = \lfloor c/2 \rfloor + 1, \quad s_2 = s_1 \bmod 3 + 1, \quad s_3 = s_2 \bmod 3 + 1. \quad (4.35)$$

We define two auxiliary variables $a, b \in \{0, \dots, k-1\}$ by

$$a = \left\lfloor \frac{i - c \cdot k^2}{k} \right\rfloor, \quad b = i - c \cdot k^2 - a \cdot k. \quad (4.36)$$

Finally, we define the region ν_i as

$$\nu_i = \left\{ \mathbf{p} \mid \mathbf{p} = (p_1, p_2, p_3) \in \mathbb{R}^3 \wedge (1 - 2(c \bmod 2))p_{s_1} > 0 \wedge 2\frac{a}{k} - 1 \leq \frac{p_{s_2}}{|p_{s_1}|} < 2\frac{a+1}{k} - 1 \wedge 2\frac{b}{k} - 1 \leq \frac{p_{s_3}}{|p_{s_1}|} < 2\frac{b+1}{k} - 1 \right\}. \quad (4.37)$$

We use two algorithms for finding the distribution of V_{T_j} ($j = 1, \dots, m$) across the regions ν_i ($i = 1, \dots, 6k^2$): approximative and analytical.

Our *approximative algorithm* is based on partitioning each volume V_{T_j} into p_j^2 ($p_j \in \mathbb{N}$) partitions (pyramids) of equal volume, by subdividing the triangle T_j , as depicted in figure 4.17. Having in mind that areas of triangles of a mesh model significantly differ, we use an adaptive partitioning of V_{T_j} in which the number of partitions depends on the area of T_j . The parameter p_j that fixes the number of partitions of V_{T_j} is defined by

$$p_j = \left\lceil \sqrt{p_{min} \frac{S_j}{S}} \right\rceil, \quad (4.38)$$

where S_j is the surface area of triangle T_j , S is the surface area of the whole mesh (3.10), p_{min} is a parameter used for setting the fineness of approximation, and $[t] \in \mathbb{Z}$ ($t \in \mathbb{R}$) denotes the ceiling function ($[t]$ is the first integer greater than t). The parameter p_{min} is used to specify the lower bound of the total number of partitions p_{total} of all volumes V_{T_j} ($i = j, \dots, m$), i.e.,

$$p_{total} = \sum_{j=1}^m p_j^2 > \sum_{j=1}^m p_{min} \frac{S_j}{S} = p_{min} \frac{\sum_{j=1}^m S_j}{S} = p_{min}. \quad (4.39)$$

The whole volume $\delta = V_{T_j}/p_j^2$ of a pyramid-like partition is accumulated in a single component of the feature vector. The component is determined using the center of

gravity of the base of the pyramid. Centers of gravity are marked with bold dots in figure 4.18, and are denoted by the variable G in the algorithm given by the pseudocode in figure 4.18. For each partition, we determine the region containing the center of gravity of the base triangle (function 'vector_index'), and accumulate the whole volume δ of partition in the corresponding component of the feature vector. Note that we first check if the whole triangle is located in a single region ν_i (4.37). In that case, there is no need for subdivision.

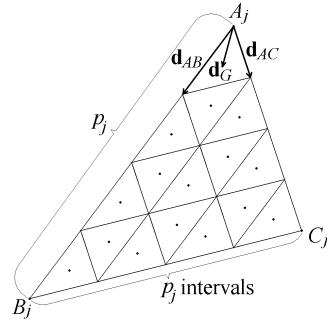


Figure 4.17: Subdivision of a triangle.

```

pos = vector_index(A_j, k);
if pos = vector_index(B_j, k)  $\wedge$  pos = vector_index(C_j, k)
    fpos  $\leftarrow$  fpos + V_{T_j};
else // There is a need for subdivision
    dAB = (Bj - Aj)/pj;
    dAC = (Cj - Aj)/pj;
    dG = (dAB + dAC)/3;
    δ = V_{T_j}/pj2;
    for x = 0, ..., pj - 2
        for y = 0, ..., x
            G = Aj + (x - y)dAB + ydAC + dG;
            pos = vector_index(G, k);
            fpos  $\leftarrow$  fpos + δ;
            pos = vector_index(G + dG, k);
            fpos  $\leftarrow$  fpos + δ;
        for y = 0, ..., pj - 1
            G = Aj + (pj - 1 - y)dAB + ydAC + dG;
            pos = vector_index(G, k);
            fpos  $\leftarrow$  fpos + δ;

```

Figure 4.18: Approximation of volume distribution.

We recall that the parameter k fixes the dimension of the feature vector, which is equal to the number of regions subdividing the 3D-space. The regions ν_i (4.37)

are indexed with an integer from the range $[1, 6k^2]$, representing the position of the vector component. The function 'vector_index', used for determining the region containing a point $A = (a_1, a_2, a_3)$, is described by the following pseudocode:

```
// Returned value: the position of the vector component (integer) from the range [1, 6k2]
// Argument A = (a1, a2, a3) ∈ ℝ3 represents a 3D-point
// Argument k ∈ ℕ is the subdivision parameter
int vector_index(A, k)
    s1, s2, s3, a, b, c ∈ ℤ;
    s1 = 1;
    if |a1| < |a2| ⇒ s1 = 2;
    if |as1| < |a3| ⇒ s1 = 3;
    s2 = s1 mod 3 + 1;
    s3 = s2 mod 3 + 1;
    a = ⌊k(as2/|as1| + 1)/2⌋;
    b = ⌊k(as3/|as1| + 1)/2⌋;
    if as1 < 0 ⇒ c = 2s1 - 1;
    else c = 2s1 - 2;
    vector_index = c · k2 + a · k + b + 1;
```

where c , s_1 , s_2 , s_3 , a , and b are defined by (4.34), (4.35), and (4.36). Hence, we determine the region where the point A is located by finding a central projection of A on the unit cube. The *central projection* from the center O onto the unit cube projects any point A onto a corresponding point A' belonging to the *closest* face of the unit cube such that O , A and A' are collinear. The projection is obtained by intersecting the unit cube with the ray emanated from the origin O and traveling in the direction A . We recall that each face of the unit cube is rasterized into a $k \times k$ grid. Then, we find the grid location (a, b) containing the projection of the point. Finally, the index of the vector component is computed $(c \cdot k^2 + a \cdot k + b + 1)$.

Since the whole volume δ of a partition is not necessarily contained in a single region of the 3D-space, an error is introduced by accumulating the whole volume in a single vector component. By choosing larger values of p_{min} , we obtain a finer approximation, whence the introduced error is smaller. However, there is a trade-off between the approximation error and the computational complexity of our algorithm. With the increase of p_{min} , the approximation errors of vector components decrease, but the computation time rises.

We also implemented another feature extraction algorithm that analytically computes components of the feature vector. With this algorithm, the distribution of the volume V_{T_j} over the regions ν_i (4.37) is exactly calculated. Briefly, our *analytical algorithm* consists of the following steps:

1. We find the central projection of each point $P \in T_j$ on the unit cube.
2. If the face c (4.34) of the unit cube contains projected points, then the set of all (infinitely many) projected points form a polygon on the face c . We determine the vertices of the polygon obtaining the polygonal line.
3. The polygonal line is used to determine a set of regions $\Omega = \{\nu_{c_1}, \dots, \nu_{c_q}\}$ that contain parts of the volume V_{T_j} .

4. For each region $\nu \in \Omega$, we intersect the pyramid, formed by T_j and the origin O , with the region ν , and compute the portion of V_{T_j} being inside ν .

In practice, finding the vertices of projected polygons (steps 1 and 2) is realized as follows:

- A list L of points, which are used to determine the set Ω of regions containing parts of the volume V_{T_j} , is initialized, so that L consists of the triangle vertices, i.e., $L = \{A_j, B_j, C_j\}$.
- The vertices of the unit cube are centrally projected on the plane of triangle T_j . The projections that are inside the triangle are added to the list L .
- Triangle edges A_jB_j , B_jC_j , and C_jA_j are intersected with all planes containing two adjacent vertices of the unit cube and the origin O . Let P_1 and P_2 be two adjacent vertices of the unit cube, and let the plane ρ be determined by P_1 , P_2 , and O . Let the line containing A_j and B_j intersect the plane ρ at the point P_{AB} . We can write

$$P_{AB} = tA_j + (1 - t)B_j, \quad t \in \mathbb{R}.$$

If $t \in [0, 1]$, then P_{AB} lies on the edge A_jB_j , and the intersection between ρ and the edge A_jB_j exists. If $t \notin [0, 1]$, then there is no intersection between ρ and the edge A_jB_j , because P_{AB} is not between A_j and B_j . The example in figure 4.19 depicts the plane ρ , where intersections P_{AB} and P_{BC} exist, while there is no intersection between the edge C_jA_j and the plane ρ . The intersection points lying inside the area marked in figure 4.19 are added to the list L . The marked area is bounded by rays cast from the origin O and traveling in the directions P_1 and P_2 . Conditions, which are used to check if a point of the plane ρ belongs to the marked area, are also given in figure 4.19.

- All points from the list L are centrally projected on the unit cube.

Determining the set Ω of regions containing V_{T_j} (step 3) is done by analyzing the projected polygonal lines for each face of the unit cube. The set of regions $\Theta = \{\nu_{p_1}, \dots, \nu_{p_r}\}$ (4.37) containing the polygonal line is determined. Finally, the set $\Omega \supset \Theta$ consists of all regions bounded by Θ . By finding the set Ω , we minimize the number of volume intersections (step 4).

A visualization of the extracted feature for $k = 4$ ($\dim = 96$) is shown in figure 4.20. Volumes of the depicted 96 pyramids, each of which belongs to a different region ν_i , are equal to the values of feature vector components corresponding to specific regions.

Feature extraction times of the volume-based feature vector of various dimensions using both the approximative and the analytical approach are given in table 4.4. The results are obtained on a PC with an 1.4 GHz AMD processor running Windows 2000. We used the MPEG-7 collection (section 5.1) to measure efficiency of our extraction algorithms. The computational complexity of the approximative algorithm depends on both the fineness of the approximation (parameter p_{min}) and the vector dimension (lower-dimensional vectors require less subdivisions), while the analytical algorithm depends on the vector dimension (parameter k). Our experimental results (5.2.4) show that the approximative algorithm for $p_{min} = 64000$

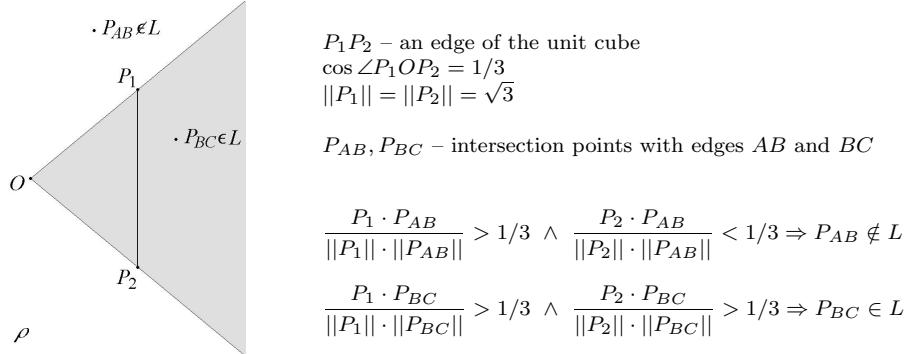


Figure 4.19: Checking if the intersection points lie in the marked area of the plane.

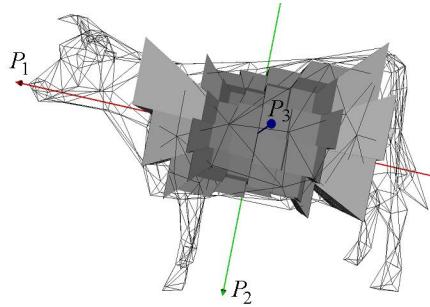


Figure 4.20: A visualization of the volume-based feature vector.

extracts feature vectors of nearly the same retrieval performance as feature vectors extracted by the analytical algorithm. Hence, in practice we use the approximative algorithm for a fast feature extraction, while the analytical method is used only for verifying results in a testing phase.

The volume-based feature vector possesses many desirable properties (section 1.3.4), e.g., translation, rotation, scale, and reflection invariance, robustness with respect to level-of-details, different tessellations, and outliers, and multi-resolution feature representation (adjustable dimensionality). Also, in the case of consistent orientation of polygons, vector components of very similar objects cannot significantly differ (figure 4.14). However, the constraint that all the models should consistently be oriented is too stringent. Many 3D-mesh models, which are available on the Internet, are non-orientable polygonal meshes (see also [49]). Therefore, as long as we rely on the orientation of polygons of a mesh model, we should not extract the volume-based descriptor without filtering (re-orienting) the mesh model. Since we do not deal with complex filtering of models, we slightly modify the method

Algorithm	Approximative						Analytical
	32	64	128	256	512	1024	
p_{min} [in 1000s]	20.2	26.7	39.2	63.8	112.5	208.9	39.3
Average p_{total} [in 1000s]	45.3	80.7	149.0	284.5	549.6	1075.9	—
$dim = 24 (k = 2)$	20.8	27.7	41.0	67.6	120.0	223.4	44.8
$dim = 54 (k = 3)$	21.6	29.5	44.4	74.1	132.6	247.9	54.4
$dim = 96 (k = 4)$	22.0	30.1	45.7	76.5	137.2	256.9	60.7
$dim = 150 (k = 5)$	22.5	31.1	47.3	80.1	144.0	270.3	70.4
$dim = 216 (k = 6)$	22.8	31.5	48.3	81.7	147.3	276.3	77.7
$dim = 294 (k = 7)$	23.0	32.1	49.5	83.8	151.5	284.5	86.8
$dim = 384 (k = 8)$	23.1	32.3	50.0	84.9	153.4	288.4	95.4
$dim = 486 (k = 9)$	—	—	—	—	—	—	—

Table 4.4: Feature extraction times (in milliseconds) for various dimensions of the volume-based feature vector, using both the approximative and analytical algorithm.

in order to avoid the orientation constraint. Instead of V_{T_j} (4.31), which can be negative, we take the absolute value $|V_{T_j}|$, in order to eliminate the problem of non-consistent orientation of polygons. All other aspects of the presented technique remain the same. Note that by taking non-negative values of volumes $|V_{T_j}|$, we do not calculate volume as depicted in figure 4.15, but we accumulate all non-negative volumes being inside regions ν_i (4.37).

We also introduce an additional modification by normalizing the feature vector \mathbf{f} (4.32) so that $\|\mathbf{f}\|_1 = 1$ (1.10). This normalization makes the computed distribution of volumes V_{T_j} or $|V_{T_j}|$ independent of scale of the object. In this case, no scaling during the normalization step (section 3.4) is necessary. Thus, there are four variants of the volume-based feature vector in the spatial domain, which are given in table 4.5.

Variant	Signed V_{T_j} (4.31)	Scaled ($\ \mathbf{f}\ _1 = 1$)
V1	yes	no
V2	no	no
V3	yes	yes
V4	no	yes

Table 4.5: Four variants of the voxel-based feature vector in the spatial domain.

Note that spectral representations of all variants of the feature vector \mathbf{f} (4.32) can easily be obtained by applying the 2D-DFT (4.26) to 6 2D-arrays $F^{(0)}, \dots, F^{(5)}$, defined using \mathbf{f} ,

$$F^{(i)} = [f_{ab}^{(i)}]_{k \times k}, \quad f_{ab}^{(i)} = f_{ck^2+ka+b+1}, \quad 0 \leq a, b \leq k-1, \quad c = 0, \dots, 5.$$

The arrays $F^{(i)}$ of volume values are transformed into arrays $\hat{F}^{(i)} = [\hat{f}_{pq}^{(i)}]_{k \times k}$ of complex Fourier coefficients. Next, we apply exactly the same approach as it is the case with the depth-buffer feature vector in the spectral domain. The magnitudes of the obtained coefficients $|\hat{f}_{pq}^{(i)}|$, whose indices satisfy the following inequality, which

is similar to (4.29),

$$|p - k/2| + |q - k/2| \leq l \leq k/2, \quad (l \in \mathbb{N}),$$

are taken as components of the feature vector (for more details see section 4.3). In practice, we extract this feature vector using the approximative algorithm with $k = 128$. We recall that the dimension of the feature vector in the frequency domain is $6(l^2 + l + 1)$, where $l < k/2$ ($l \in \mathbb{N}$) is the highest order of harmonics that are used to form the vector. Also, an embedded multi-resolution representation (1.8) is provided, by forming the vector according to (4.30).

We compared all variants of the volume-based method in both spatial and spectral domains (see section 5.2.4). Because of the fact that 3D-models are not necessarily orientable, the variants V2 and V4 (table 4.5) outperform the variants V1 and V3. Retrieval effectiveness can slightly be improved by representing the volume-based feature in the spectral domain. Nevertheless, the overall retrieval performance of the volume-based descriptor is inferior to the silhouette-based and depth buffer-based descriptors presented in this chapter, regardless of a high stability of the volume-based approach with respect to variations of the polygonal surface (high-frequency noise of the surface), different tessellations, as well as levels-of-detail. After evaluation (section 5.2.4), we suggest to use the variant V4 of the volume-based descriptor in the spectral domain of dimension 438.

4.5 Voxel-Based Approach

Polygonal mesh is the most common way of representing 3D-objects, and all the models that we analyze are represented as polygonal meshes (1.5). However, in order to define a new 3D-shape descriptor, we represent models as volumetric data by voxelizing polygonal meshes. We define a specific volumetric representation of a mesh model, which can be either directly used as a feature vector in the spatial domain or further processed to generate a descriptor in the spectral domain [144]. In what follows, we mostly adopt the terminology given in [53].

Voxelization transforms the continuous 3D-space \mathbb{R}^3 , which contains models represented as polygonal meshes, into the discrete 3D voxel space \mathbb{Z}^3 . The voxelization proceeds in three steps: discretization, sampling, and storing. Discretization yields the cellular subdivision of the continuous 3D-space into *voxels* (volume elements). Generally, a cuboid region $\rho \subset \mathbb{R}^3$ (4.19), is discretized into an $N \times M \times P$ grid of voxels, which are defined as follows.

Definition 4.5 A voxel $\mu_{abc} \subset \rho$ (4.19) is a continuous cuboid region of dimensions $d_x \times d_y \times d_z$ contained by a 3D discrete point $(a, b, c) \in \mathbb{Z}^3$ such that

$$\mu_{abc} = \left\{ (x, y, z) \mid (x, y, z) \in \mathbb{R}^3, a \leq \frac{x - x_{min}}{d_x} < a + 1, \right. \\ \left. b \leq \frac{y - y_{min}}{d_y} < b + 1, c \leq \frac{z - z_{min}}{d_z} < c + 1 \right\}, \quad (4.40)$$

where

$$d_x = \frac{x_{max} - x_{min}}{M}, \quad d_y = \frac{y_{max} - y_{min}}{N}, \quad d_z = \frac{z_{max} - z_{min}}{P}, \quad (4.41)$$

and

$$a = 0, \dots, M - 1, \quad b = 0, \dots, N - 1, \quad c = 0, \dots, P - 1.$$

After the sampling, which is application specific, the voxel μ_{abc} is attributed a value $v_{abc} \in \mathbb{K}$ depending on positions of the polygons of a 3D-mesh model. In other words, v_{abc} depends on the point set I (1.5) of the model. Usually, v_{abc} is a scalar value, and we deal either with a binary voxel grid ($\mathbb{K} = \{0, 1\}$) or real voxel grid ($\mathbb{K} \equiv \mathbb{R}$). In the binary case, the voxel value v_{abc} is defined by

$$v_{abc} = \begin{cases} 0 \text{ (or } 1\text{),} & I \cap \mu_{abc} = \emptyset \\ 1 \text{ (or } 0\text{),} & I \cap \mu_{abc} \neq \emptyset \end{cases} \quad (4.42)$$

Thus, if there is a point $\mathbf{p} \in I$ laying inside μ_{abc} , then we set $v_{abc} = 1$, otherwise $v_{abc} = 0$, or vice versa. There are several choices to attribute a voxel with a real value. For instance, the voxel attribute v_{abc} might be a color value. Also, the voxel grid might be used for storing 3D distance fields [33, 56].

Voxel values are usually stored in a 3D-array $[v_{abc}]_{M \times N \times P}$. If the 3D-array is mostly populated with zeros, the space needed for storing a voxel grid can significantly be reduced by using an octree structure [116].

A voxelization method is defined by specifying the region ρ (4.19), the dimensions of voxel grid M , N , and P (or voxel dimensions d_x , d_y , and d_z), and the exact calculation of voxel attributes. In what follows, we always use cubic voxel grids of type $N \times N \times N$ ($M = N = P$). We use several definitions of the cuboid region ρ (4.19), which is voxelized:

1. ρ is the canonical cube (CC) (definition 4.4);
2. ρ is the canonical bounding cube (CBC) of a model (definition 4.1);
3. ρ is the bounding box (BB) of a model (definition 4.2);
4. ρ is the extended bounding box (EBB) of a model (definition 4.3).

In our approach, a 3D-mesh model I (1.5) is voxelized so that the voxel attribute v_{abc} is equal to the fraction of the total surface area S of the mesh, which is inside the region μ_{abc} (4.40), i.e,

$$v_{abc} = \frac{\text{area}\{\mu_{abc} \cap I\}}{S}, \quad 0 \leq a, b, c \leq N - 1. \quad (4.43)$$

We created both approximative and analytical algorithm for computing the voxel attributes v_{abc} (4.43).

The *approximative* algorithm is given in figure 4.21, and is based on the same idea as the algorithm described in figure 4.18. Each triangle T_j of a mesh model is

subdivided (figure 4.17) into p_j^2 coincident triangles each of which has the surface area equal to $\delta = S_j/p_j^2$, where S_j is the area of T_j . If all vertices of the triangle T_j lie in the same cuboid region μ_{abc} (4.40), then we set $p_j = 1$, otherwise we use (4.38) to determine the value of p_j . For each newly obtained triangle, the center of gravity G is computed, and the voxel $\mu_{abc} \ni G$ is determined. Finally, the attribute v_{abc} is incremented by δ . As mentioned in section 4.4, the quality of approximation is set by the parameter p_{min} (4.38). For clarity, several lines are separated as the procedure *inc_att* in the pseudocode in figure 4.21. In our implementation, we do not have procedure calls as well as the computation of centers of gravity is done without multiplications.

```

set all voxel attributes  $v_{abc}$  to 0;
for each triangle  $T_j$  (with vertices  $A_j$ ,  $B_j$ , and  $C_j$ ) do
    //  $S$  and  $S_j$  are defined by (1.6)
    //  $p_j$  is the parameter given by (4.38)
    //  $d_x$ ,  $d_y$ , and  $d_z$  are defined by (4.41)
    if  $A_j$ ,  $B_j$ , and  $C_j$  belong to the same voxel  $\Rightarrow p_j = 1$ ;
     $\mathbf{d}_{AB} = (B_j - A_j)/p_j$ ;
     $\mathbf{d}_{AC} = (C_j - A_j)/p_j$ ;
     $\mathbf{d}_G = (\mathbf{d}_{AB} + \mathbf{d}_{AC})/3$ ;
     $\delta = S_j/(p_j^2 S)$ ;
    for  $x = 0, \dots, p_j - 2$ 
        for  $y = 0, \dots, x$ 
            inc_att( $A_j + (x - y)\mathbf{d}_{AB} + y\mathbf{d}_{AC} + \mathbf{d}_G, \delta$ );
            inc_att( $A_j + (x - y)\mathbf{d}_{AB} + y\mathbf{d}_{AC} + 2\mathbf{d}_G, \delta$ );
    for  $y = 0, \dots, p_j - 1$ 
        inc_att( $A_j + (p_j - 1 - y)\mathbf{d}_{AB} + y\mathbf{d}_{AC} + \mathbf{d}_G, \delta$ );

    // The following procedure determines indices  $(a, b, c)$  of the voxel region
    // that contains the point  $G = (g_x, g_y, g_z)$ , and increments  $v_{abc}$  by  $\delta$ 
    inc_att( $G, \delta$ )
     $a = \lfloor (g_x - x_{min})/d_x \rfloor$ ,  $b = \lfloor (g_y - y_{min})/d_y \rfloor$ ,  $c = \lfloor (g_z - z_{min})/d_z \rfloor$ ;
    //  $x_{min}$ ,  $y_{min}$ , and  $z_{min}$  are given in (4.19)
    //  $d_x$ ,  $d_y$ , and  $d_z$  are defined by (4.41)
     $v_{abc} \leftarrow v_{abc} + \delta$ ;

```

Figure 4.21: Approximation of voxel attributes.

The *analytical* algorithm for calculating the attributes v_{abc} (4.43), for a given mesh model I (1.5), also starts with initializing voxel values to 0. Next, for each triangle T_j (1.2) of the mesh, we determine a set of voxels V_j intersected by T_j , using a very efficient algorithm presented in [53]. We recall that the triangle T_j is defined by its vertices A_j , B_j , and C_j , and $1 \leq j \leq m$ (1.2). Then, for each $\mu_{abc} \in V_j$, we compute the area δ_{abc} of the triangle T_j , which is inside μ_{abc} (4.40). The value of $\delta_{abc} = \text{area}\{\mu_{abc} \cap T_j\}$ is computed by performing the following steps:

1. *Check if the triangle vertices A_j , B_j , and C_j lay inside μ_{abc} . The vertices being inside μ_{abc} are inserted in a list L , which is initially empty.*

The point $A_j = (a_{x_j}, a_{y_j}, a_{z_j})$ (similarly B_j and C_j) is processed as follows:

$$\begin{aligned} a \leq \frac{a_{x_j} - x_{min}}{d_x} &< a + 1 \\ \wedge \\ A_j \in \mu_{abc} \iff b \leq \frac{a_{y_j} - y_{min}}{d_y} &< b + 1 , \quad A_j \in \mu_{abc} \implies A_j \in L. \\ \wedge \\ c \leq \frac{a_{z_j} - z_{min}}{d_z} &< c + 1 \end{aligned}$$

2. If $L = \{A_j, B_j, C_j\}$, then $\delta_{abc} = S_j$ (S_j is the surface area of T_j). Otherwise, the algorithm proceeds with the next step.
3. Check if the edges $\overline{A_jB_j}$, $\overline{B_jC_j}$, and $\overline{C_jA_j}$ intersect the boundaries of μ_{abc} . The intersection points are added to the list L .

The cuboid region μ_{abc} is bounded by 6 rectangles (or squares) each of which is intersected with the three edges. Suppose that the edge $\overline{A_jB_j}$ is intersected with the rectangle R_{abc}^{x-} , which is defined by

$$R_{abc}^{x-} = \left\{ (x_{min} + a d_x, y, z) \mid b \leq \frac{y - y_{min}}{d_y} < b + 1, c \leq \frac{z - z_{min}}{d_z} < c + 1 \right\},$$

where $d_x, d_y, d_z, x_{min}, y_{min}$, and z_{min} are given by (4.41) and (4.19). Firstly, we find the intersection point $\mathbf{p} = (p_x, p_y, p_z)$ of the line A_jB_j and the plane $\pi_{abc}^{x-} \supset R_{abc}^{x-}$. Since the plane $\pi_{abc}^{x-} \ni \mathbf{p}$ is determined by equation $x = x_{min} + a d_x$, we have $p_x = x_{min} + a d_x$. The point \mathbf{p} also lies on the line A_jB_j , i.e., $\mathbf{p} = \alpha(A_j - B_j) + B_j$. We directly compute the value of α by

$$p_x = \alpha(a_{x_j} - b_{x_j}) + b_{x_j} \implies \alpha = \frac{x_{min} + a d_x - b_{x_j}}{a_{x_j} - b_{x_j}}.$$

If $0 \leq \alpha \leq 1$, then $\mathbf{p} \in \overline{A_jB_j}$. Finally, we compute p_y and p_z (α is known), and check if $\mathbf{p} \in R_{abc}^{x-}$, i.e.,

$$b \leq \frac{p_y - y_{min}}{d_y} < b + 1 \quad \wedge \quad c \leq \frac{p_z - z_{min}}{d_z} < c + 1$$

If $\mathbf{p} \in \overline{A_jB_j}$ and $\mathbf{p} \in R_{abc}^{x-}$, we insert \mathbf{p} into the list L .

4. Check if voxel edges intersect the triangle T_j , and add the intersection points to the list L .

Let $\overline{\mathbf{p}_0\mathbf{p}_1}$ be one of 12 voxel edges, e.g.,

$$\begin{aligned} \mathbf{p}_0 &= (x_{min} + ad_x, y_{min} + bd_y, z_{min} + cd_z), \\ \mathbf{p}_1 &= (x_{min} + ad_x, y_{min} + bd_y, z_{min} + (c+1)d_z). \end{aligned}$$

Let \mathbf{p} be the intersection point of the line $\mathbf{p}_0\mathbf{p}_1$ and the plane of the triangle T_j . We have

$$\mathbf{p} = (x_{min} + ad_x, y_{min} + bd_y, p_z) = \alpha(A_j - C_j) + \beta(B_j - C_j) + C_j,$$

whence we form a system of two equations with unknowns α and β ,

$$\begin{aligned}\alpha(a_{x_j} - c_{x_j}) + \beta(b_{x_j} - c_{x_j}) &= x_{min} + ad_x - c_{x_j}, \\ \alpha(a_{y_j} - c_{y_j}) + \beta(b_{y_j} - c_{y_j}) &= y_{min} + bd_y - c_{y_j}.\end{aligned}\quad (4.44)$$

After solving the system, we check if $\mathbf{p} \in T_j$ ($\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta \leq 1$), compute p_z , and check if $\mathbf{p} \in \overline{\mathbf{p}_0\mathbf{p}_1}$ ($c \leq (p_z - z_{min})/d_z < c + 1$).

If $\mathbf{p} \in T_j$ and $\mathbf{p} \in \overline{\mathbf{p}_0\mathbf{p}_1}$, we insert \mathbf{p} into the list L .

Note that we could also use the ray-triangle intersection algorithms (figures 4.2 and 4.3). However, the ray-triangle intersection algorithm for a general case is less effective, because x and y coordinates of the points \mathbf{p} , \mathbf{p}_0 , and \mathbf{p}_1 are the same. Therefore, this special case is better handled by the presented approach (4.44).

5. The list $L = \{\mathbf{t}_1, \dots, \mathbf{t}_l\}$ contains coplanar points. If the size of the list is greater than 3 ($l > 3$), then the list is sorted so that $\mathbf{t}_1 \dots \mathbf{t}_l$ form a convex polygon.

The list is sorted using the following procedure. Firstly, a new 2D-coordinate system is set. The origin \mathbf{m} , the abscissa \mathbf{x} , and the ordinate \mathbf{y} are computed by

$$\mathbf{m} = \frac{1}{l} \sum_{i=1}^l \mathbf{t}_i, \quad \mathbf{x} = \frac{\mathbf{t}' - \mathbf{m}}{\|\mathbf{t}' - \mathbf{m}\|}, \quad \mathbf{y} = \frac{\mathbf{x} \times (\mathbf{t}'' - \mathbf{m})}{\|\mathbf{x} \times (\mathbf{t}'' - \mathbf{m})\|} \times \mathbf{x},$$

where $\mathbf{t}', \mathbf{t}'' \in L$ are chosen so that $\|\mathbf{t}' - \mathbf{m}\|$ and the angle $\angle(\mathbf{x}, \mathbf{t}'' - \mathbf{m})$ are maximal, i.e.,

$$\|\mathbf{t}' - \mathbf{m}\| = \max_{1 \leq i \leq l} \{\|\mathbf{t}_i - \mathbf{m}\|\}, \quad \frac{|\mathbf{x} \cdot (\mathbf{t}^* - \mathbf{m})|}{\|\mathbf{t}^* - \mathbf{m}\|} = \min_{1 \leq i \leq l} \left\{ \frac{|\mathbf{x} \cdot (\mathbf{t}_i - \mathbf{m})|}{\|\mathbf{t}_i - \mathbf{m}\|} \right\}.$$

The points \mathbf{t}_i are represented in the new coordinate system

$$\mathbf{t}_i = (x_i, y_i) \equiv \mathbf{m} + x_i \mathbf{x} + y_i \mathbf{y}, \quad x_i = (\mathbf{t}_i - \mathbf{m}) \cdot \mathbf{x}, \quad y_i = (\mathbf{t}_i - \mathbf{m}) \cdot \mathbf{y}.$$

The points \mathbf{t}_i are reordered in the non-decreasing order of the polar coordinate $\varphi_i = \arctan(y_i/x_i)$.

6. The surface area δ_{abc} of the polygon $\mathbf{t}_1 \dots \mathbf{t}_l$ is calculated by summing up areas of $l - 2$ triangles, $\Delta \mathbf{t}_1 \mathbf{t}_2 \mathbf{t}_3, \dots, \Delta \mathbf{t}_1 \mathbf{t}_{l-1} \mathbf{t}_l$. The voxel attribute v_{abc} is incremented by δ_{abc}/S (1.6).

In the case when the CBC, BB, and EBB are used, a 3D-model I is completely located inside the region ρ . If the CC is used as the cuboid region ρ (4.19), the

underlying model might be clipped, when the parameter w is small. If we want to enclose the whole 3D-model into the CC, then we take an empirically determined value $w = 32$. For such a large value of w , the model is relatively small comparing to the region ρ . In order to obtain a decent shape representation of the model, we recommend $N \geq 32w$. Such a voxel grid has more than 10^9 voxels, but only a very small fraction of them has non-zero values. In this case, an octree is a very suitable structure for an effective storing of voxel attributes. As it is the case with the depth-buffer method (section 4.3), the CC is used in order to make the voxel representation more robust with respect to outliers. Namely, if we add an outlier to a model, then the CBC, BB, and EBB can significantly change, resulting in a totally different voxel representation (figure 4.22), while the CC will remain approximately the same, resulting in very slight changes of voxel attributes. We recall that the CC with $w = 2$ is used for generating depth buffer images (section 4.3), producing good shape descriptors.

Note that when the BB is used, voxelized models are deformed, i.e., the aspect ratio of the model is not preserved by the voxel grid. Nevertheless, our tests (section 5.2.5) show that engaging the BB to form the voxel-based feature vector in the spatial domain is a good choice.

Usually, we set $N \in \{3, 4, 5, 6, 7, 8\}$, and treat the obtained voxel grids as feature vectors of dimensions N^3 . The experiments (section 5.2.5) show that we get a better retrieval performance for odd values of N . This can be explained by the fact that the coordinate hyper-planes of the CPCA frame (section 3.4) intersect the middle parts of voxel regions, when N is an odd number. We recall that a model in the canonical frame is positioned so that the largest spreads of its point set I (1.5) coincide with the coordinate axes. When N is even, a number of voxel regions is bounded by the coordinate hyper-planes. For small values of N , the information about distribution of the point set I is captured in a more useful way if no voxels are bounded by the coordinate hyper-planes.

If a voxel grid is stored in an octree structure, we set $N = 2^r$, where $r \in \mathbb{N}$ is the maximal depth (level) of the octree. We regard the parameter r as the *resolution of voxelization*. Note that all octrees with the depth less than r are included, whence an embedded multi-resolution representation is provided. The attributes stored in an octree can be regarded as a descriptor in the spatial domain. A specific characteristic of this descriptor is the non-constant size. We created an efficient algorithm for traversing two octree descriptors in order to compute the l_p distance (1.9). Nevertheless, the time needed to compute the l_1 distance between two octrees of depth 6, formed using the CBC, is approximately 21ms on a PC with an 1.4 GHz AMD processor running Windows 2000. For comparison, this time is 300 times greater the time needed to compute the l_1 distance between two vectors of dimensions 400. Hence, some important requirements for 3D-shape descriptors (section 1.3.4), such as compact representation and efficient search, are not fulfilled. We consider that an octree is not appropriate representation of a feature. However, octrees can be engaged for processing voxel grids of higher resolutions. Then, we benefit by using such an efficient structure for storing a voxel grid.

As it is the case with other descriptors in the spatial domain, the increase of the

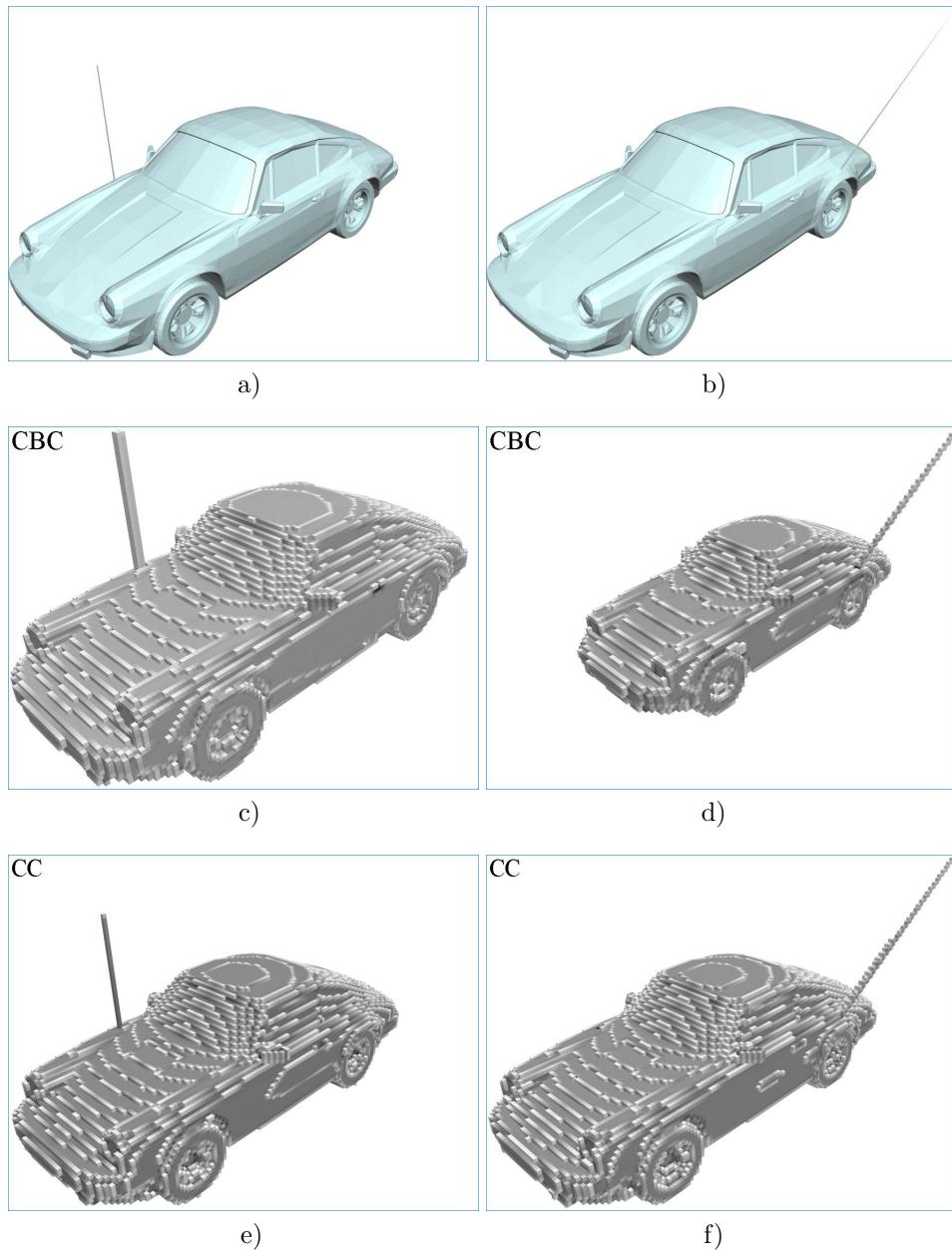


Figure 4.22: The impact of outliers on voxel representations. The only difference between the original models a) and b) are positions of the antennas. The models are voxelized at the same resolution and visualized from the same viewpoint using the CBC (c,d) and CC (e,f).

vector dimensionality (resolution of voxelization) leads to a lower effectiveness of the l_p norms (1.9). A simple example, shown in figure 4.23, illustrates the trade-off between the resolution of voxelization (level of details or the depth of the octree) and the effectiveness of the l_p distance. Only three voxels v_{abc} , w_{abc} , and u_{abc} of different feature vectors \mathbf{v} , \mathbf{w} , and \mathbf{u} are considered. Note that the voxels lay at the same position in the \mathbb{Z}^3 space. Let $v_{abc} \approx w_{abc} \approx \delta$ and $u_{abc} = 0$ at the resolution r . At the resolution $r + 1$, each voxel is subdivided into 8 new voxels, $v_{abc}^{(1)}, \dots, v_{abc}^{(8)}$ (similarly for w_{abc} and u_{abc}). Let $v_{abc}^{(1)} \approx v_{abc}^{(4)} \approx v_{abc}^{(5)} \approx v_{abc}^{(8)} \approx w_{abc}^{(2)} \approx w_{abc}^{(3)} \approx w_{abc}^{(6)} \approx w_{abc}^{(7)} \approx \delta/4$, while the remaining attributes are equal 0 at the resolution $r + 1$. If the l_p norm (1.9) is applied for calculating the dissimilarity between the voxel attributes at the position (a, b, c) and resolution r , then we have

$$d_{vw} = \|v_{abc} - w_{abc}\|_p = 0, \quad d_{vu} = \|v_{abc} - u_{abc}\|_p = \delta \Rightarrow d_{vw} < d_{vu}. \quad (4.45)$$

However, at the resolution $r + 1$ the dissimilarity measure gives the opposite result,

$$d_{vw} = \left(\sum_{t=1}^8 |v_{abc}^{(t)} - w_{abc}^{(t)}|^p \right)^{1/p} > \left(\sum_{t=1}^8 |v_{abc}^{(t)} - u_{abc}^{(t)}|^p \right)^{1/p} = d_{vu}. \quad (4.46)$$

Hence, from (4.45) it follows $d_{vw} < d_{vu}$, while (4.46) gives the opposite conclusion, $d_{vw} > d_{vu}$. This situation may happen when corresponding parts of two models (e.g., doors of car models) are just slightly displaced in the CPCPA frame, and the displacement is registered at the resolution $r + 1$. From a user's aspect, voxels v_{abc} and w_{abc} are identical at the resolution r and very distinctive from u_{abc} . This impression should be somehow preserved at the resolution $r + 1$. Obviously, it can not be done with the l_p norm.

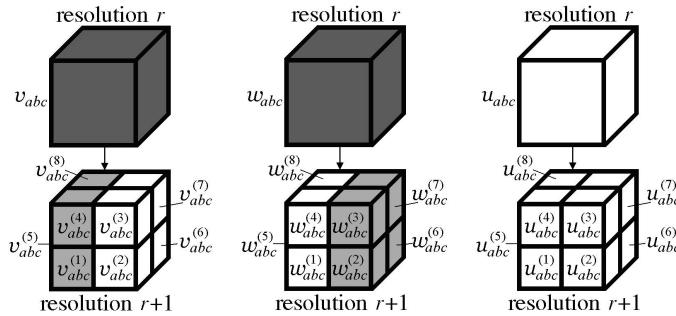


Figure 4.23: The increase of the resolution of voxelization reduces the effectiveness of the l_p norm in the spatial domain.

The information contained in a voxel grid can be processed further, in order to obtain both correlated information and more compact representation of voxel attributes as a feature. In [144], we applied the 3D Discrete Fourier Transform (3D-DFT) to obtain a spectral domain feature vector. Briefly, a 3D-array of complex

numbers $F = [f_{abc}]_{M \times N \times P}$ ($f_{abc} \in \mathbb{C}$) is transformed into another 3D-array $\hat{F} = [\hat{f}_{pqs}]_{M \times N \times P}$ ($\hat{f}_{pqs} \in \mathbb{C}$) by

$$\hat{f}_{pqs} = \frac{1}{\sqrt{MNP}} \sum_{a=0}^{M-1} \sum_{b=0}^{N-1} \sum_{c=0}^{P-1} f_{abc} e^{-2\pi j(ap/M+bq/N+cs/P)}, \quad (4.47)$$

where j is the imaginary unit and $0 \leq p \leq M - 1$, $0 \leq q \leq N - 1$, $0 \leq s \leq P - 1$.

Since we apply the 3D-DFT to an $N \times N \times N$ voxel grid with real-valued attributes v_{abc} , we shift the indices (i.e., the space \mathbb{Z}^3) so that (a, b, c) is translated into $(a - N/2, b - N/2, c - N/2)$. We introduce the following abbreviation

$$v'_{a-N/2, b-N/2, c-N/2} \equiv v_{abc}.$$

Thus, the origin $(0, 0, 0)$ is shifted to $(N/2, N/2, N/2)$. Therefore, we adjust the formula (4.47),

$$\hat{f}_{pqs} = \frac{1}{\sqrt{N^3}} \sum_{a=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{b=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{c=-\frac{N}{2}}^{\frac{N}{2}-1} v'_{abc} e^{-2\pi j(ap+bq+cs)/N}, \quad (4.48)$$

where $-N/2 \leq p, q, s \leq N/2 - 1$. Usually, we use octrees to store voxel attributes, which are used as the input for the 3D-DFT.

To form a shape descriptor, we apply a similar approach as in the previous sections. We take magnitudes $|\hat{f}_{pqs}|$ of low-frequency coefficients as components of the feature vector. Since the 3D-DFT input is a real-valued 3D-array, the symmetry is present among obtained coefficients (compare (4.16) and (4.28)),

$$\begin{aligned} v'_{abc} \in \mathbb{R} \implies \hat{f}_{pqs} &= \overline{\hat{f}_{p'q's'}} \implies |\hat{f}_{pqs}| = |\hat{f}_{p'q's'}|, \\ (p + p') \bmod N &= 0, \quad (q + q') \bmod N = 0, \quad (s + s') \bmod N = 0. \end{aligned} \quad (4.49)$$

The feature vector is formed from all non-symmetrical coefficients \hat{f}_{pqs} so that

$$1 \leq |p| + |q| + |s| \leq k \leq N/2. \quad (4.50)$$

The coefficient \hat{f}_{000} is not included, because it is always equal to 1, when the CBC, BB, or EBB are used. However, if the CC is engaged, then $\hat{f}_{000} \leq 1$, because a model may be cropped. Since we are interested in distribution of voxel attributes as a feature, we normalize \hat{f}_{pqs} by dividing by $|\hat{f}_{000}|$. By suitably choosing the component values, an embedded multi-resolution feature representation (1.8) is provided. The exact way of forming the feature vector $\mathbf{f} = (f_1, \dots, f_{dim})$ is described by the pseudocode in figure 4.24.

For a fixed value of h (the outer loop in the pseudocode in figure 4.24), the total number of taken coefficients $d(h)$ can be computed by

$$d(h) = 3 + 6(h - 1) + 2(h - 1)(h - 2) = 2h^2 + 1. \quad (4.51)$$

```

i = 1;
scale = 1;
if  $|\hat{f}_{0,0,0}| > 0 \Rightarrow scale = |\hat{f}_{0,0,0}|$ ;
for  $h = 1, \dots, k$ 
     $f_i = |\hat{f}_{h,0,0}|/scale, i \leftarrow i + 1$ ;
     $f_i = |\hat{f}_{0,h,0}|/scale, i \leftarrow i + 1$ ;
     $f_i = |\hat{f}_{0,0,h}|/scale, i \leftarrow i + 1$ ;
    for  $x = 1, \dots, h - 1$ 
         $f_i = |\hat{f}_{0,x,h-x}|/scale, i \leftarrow i + 1$ ;
         $f_i = |\hat{f}_{0,x,x-h}|/scale, i \leftarrow i + 1$ ;
         $f_i = |\hat{f}_{x,0,h-x}|/scale, i \leftarrow i + 1$ ;
         $f_i = |\hat{f}_{x,0,x-h}|/scale, i \leftarrow i + 1$ ;
         $f_i = |\hat{f}_{x,h-x,0}|/scale, i \leftarrow i + 1$ ;
         $f_i = |\hat{f}_{x,x-h,0}|/scale, i \leftarrow i + 1$ ;
    for  $x = 1, \dots, h - 2$ 
        for  $y = 1, \dots, h - 1 - x$ 
             $f_i = |\hat{f}_{x,y,h-x-y}|/scale, i \leftarrow i + 1$ ;
             $f_i = |\hat{f}_{x,y,x+y-h}|/scale, i \leftarrow i + 1$ ;
             $f_i = |\hat{f}_{x,-y,h-x-y}|/scale, i \leftarrow i + 1$ ;
             $f_i = |\hat{f}_{x,-y,x+y-h}|/scale, i \leftarrow i + 1$ ;

```

Figure 4.24: The forming of the voxel-based feature vector $\mathbf{f} = (f_1, \dots, f_{dim})$, in the spectral domain.

The dimension dim of the feature vector, i.e., the total number of all taken coefficients, is determined by

$$dim = \sum_{h=1}^k d(h) = k(2k^2 + 3k + 4)/3. \quad (4.52)$$

We recommend to set $k = 8$ so that the vectors of dimensions 12, 31, 64, 115, 188, and 287 ($k = 2, \dots, 7$) are embedded in the feature vector of 416 components.

The efficiency of the analytical and approximative algorithms are compared using the results from table 4.6. The results are obtained on a PC with an 1.4 GHz AMD processor running Windows 2000. The MPEG-7 collection (section 5.1) is used for measuring times and statistics. The voxel-based descriptor in the spatial domain is extracted for $N = 2, \dots, 8$, producing the vectors of dimensions 8, 27, 64, 125, 216, 343, and 512. The approximative extraction algorithm (figure 4.21) is tested for $p_{min} \in \{16000, 32000, 64000, 96000\}$. For each value of p_{min} , the average number of samples p_{total} (4.39) is also given. Since we set $p_j = 1$, when the whole triangle attributes a single voxel, the total number p_{inc_att} of executions of the procedure *inc_att* (figure 4.21) is significantly lower than p_{total} , for small N . The computational complexity linearly depends on p_{inc_att} . Since there is a certain amount of processing time, which does not depend on p_{inc_att} , but on the number of triangles m of the mesh, the computational complexity can be written

as $\mathcal{O}(c \cdot m + p_{inc_att})$, where c is a constant. As an example of interpreting the results from table 4.6, we read that, on average, the voxel-based feature vector of dimension 343 is extracted in 12.5ms, when the approximative algorithm is used with $p_{min} = 32000$, while the average p_{inc_att} is equal to 34180. For $p_{min} = 32000$, the average value of p_{total} is 45311, which means that $45311 - 34180 = 11131$ executions of the procedure *inc_att* are saved (on average) by checking if the whole triangle attributes a single voxel.

Algorithm	Approximative				Analytical
p_{min} (4.38)	16000	32000	64000	96000	-
Average p_{total} (4.39)	27179	45311	80734	114814	-
Voxel grid $N = 2$ ($dim = 8$) [ms]	9.7	11.3	14.4	17.3	17.5
Average p_{inc_att}	18312	27257	44716	61899	-
Voxel grid $N = 3$ ($dim = 27$) [ms]	9.7	11.1	14.1	16.9	15.5
Average p_{inc_att}	17954	26498	43314	59837	-
Voxel grid $N = 4$ ($dim = 64$) [ms]	10.1	11.9	15.6	19.1	22.6
Average p_{inc_att}	20195	30935	51890	72419	-
Voxel grid $N = 5$ ($dim = 125$) [ms]	10.1	11.9	15.7	19.3	22.8
Average p_{inc_att}	20349	31228	52567	73387	-
Voxel grid $N = 6$ ($dim = 216$) [ms]	10.3	12.5	16.7	20.8	29.6
Average p_{inc_att}	21875	34196	58339	81914	-
Voxel grid $N = 7$ ($dim = 343$) [ms]	10.3	12.5	16.7	20.8	30.0
Average p_{inc_att}	21858	34180	58274	81781	-
Voxel grid $N = 8$ ($dim = 512$) [ms]	11.6	12.8	17.4	21.8	36.0
Average p_{inc_att}	22905	36263	62308	87655	-
Avg. no. of octree nodes (depth=6)	8643	9733	10432	10733	11561
Oct6 (octree with depth=6) [ms]	28	36	49	61	523
3D-DFT $dim = 416$ (with Oct6) [ms]	153	178	192	199	224
Total time (Oct6 + 3D-DFT) [ms]	181	214	241	260	747
Avg. no. of octree nodes (depth=7)	21967	28389	34609	37665	47391
Oct7 (octree with depth=7) [ms]	53	68	89	110	2273
3D-DFT $dim = 416$ (with Oct7) [ms]	339	470	604	671	887
Total time (Oct7 + 3D-DFT) [ms]	392	538	693	781	3160

Table 4.6: Average feature extraction times of the approximative and analytical algorithms for generating voxel-based feature vectors in the spatial domain, octree voxel grids, and spectral representations using the 3D-DFT. In all cases, the CBC is used. The number of executions of the procedure *inc_att* (figure 4.21) is denoted by p_{inc_att} .

The octree-based representation is tested with two maximal depths, 6 and 7. An octree with the depth r stores a $2^r \times 2^r \times 2^r$ voxel grid. Thus, we voxelized each model into $64 \times 64 \times 64$ and $128 \times 128 \times 128$ grids using four different settings of the approximative algorithm and the analytical algorithm. The average number of octree nodes increases with the quality of approximation. For these dimensions of grids, $p_{total} \approx p_{inc_att}$. The given extraction times of the octree-based representations take into account memory allocations of octree nodes. The complexity of the 3D-DFT applied to an octree is directly proportional to the number of nodes in the octree. The total extraction time of the voxel-based descriptor in the spectral do-

main is obtained by adding times needed for generating the octree and performing the 3D-DFT. The extraction times in table 4.6 are obtained when the CBC is used.

We observe that the approximative method is significantly faster than the analytical algorithm, in particular for larger values of N . Our experiments (section 5.2.5) show that feature vectors extracted by the approximative method for $p_{min} = 32000$ possess almost the same retrieval effectiveness as vectors obtained by the analytical approach. We tested all presented choices for fixing the region of voxelization. In the spatial domain, we suggest to use the BB for voxelization and dimension 343. In the spectral domain, we recommend the vector of dimension 417 obtained by applying the 3D-DFT to an octree of depth 7, which represents a voxel grid formed using the CC with $w = 2$ (definition 4.4). The scale should be fixed by the average distance (3.25), if the CC is used. Note that, if the voxel attributes are defined by (4.43) and CBC, BB, or EBB are used, there is no need to fix the scale of an object during the normalization step (section 3.4).

4.6 Describing 3D-Shape with Functions on a Sphere

Certain features aimed at describing 3D-shape [147, 117, 146, 142] can be considered as samples of a function on a sphere S^2 . The sphere $S^2 \subset \mathbb{R}^3$ is a sphere of an arbitrary radius with the center at the origin. For example, for a (normalized) model I (1.5) define a function f on the sphere S^2 ,

$$\begin{aligned} f &: S^2 \rightarrow \mathbb{K} \\ \mathbf{u} &\mapsto f(\mathbf{u}), \end{aligned} \quad (4.53)$$

where $\mathbb{K} = \{0, 1\}$, $\mathbb{K} \equiv \mathbb{R}$ or $\mathbb{K} \equiv \mathbb{C}$. This function $f(\mathbf{u})$ measures a property of the object in the directions given by

$$\mathbf{u} = \mathbf{u}(\theta, \varphi) = (\cos \varphi \sin \theta, \sin \varphi \sin \theta, \cos \theta) \in S^2, \quad 0 \leq \theta \leq \pi, \quad 0 \leq \varphi < 2\pi. \quad (4.54)$$

Hence, f depends only on angular coordinates θ and φ . The angle θ is measured down from the z -axis, while φ is measured counterclockwise off the x -axis in the plane xOy .

We can take a number of samples $f(\mathbf{u})$ as components of a feature vector in the spatial domain. However, such a descriptor is sensitive to small perturbations of the model, because a property along given direction is measured (see examples in figures 4.5 and 4.14a).

In [147], we introduced a concept that improves the robustness of the descriptor by sampling the spherical function $f(\mathbf{u})$ at many points but characterizing the map by just a few parameters, using either spherical harmonics [45, 128, 78] or moments [117].

In this section, we give a brief introduction to the Spherical Fast Fourier Transform (SFFT) and describe our general approach for generating 3D-shape descriptors based on a function on the sphere S^2 . Then, the ray-based approach (section 4.1) is used to generate 3D-shape descriptors based on spherical harmonic representation

and to define our moments-based feature vector. Next, we present several 3D-shape descriptors that are based on various definitions of the function $f(\mathbf{u})$. We also explore the idea to define functions on concentric spheres and to use a property of spherical harmonics to obtain a rotation invariant descriptor, where the invariance is achieved not using the PCA, but by the definition of descriptor.

4.6.1 Spherical Harmonics

The need for numerical computation of Fourier series of functions on the sphere S^2 is present in many areas of applied science [45, 60], e.g, astronomy, image understanding, physics, biology, chemistry, etc. As far as we know, spherical harmonics are introduced as a tool for 3D model retrieval in [147]. In what follows, we summarize some basic facts from the theory of spherical harmonic adopting the terminology and results given in [45]. We also present our approach of generating 3D-shape feature vectors using spherical harmonics.

Let $L^2(S^2)$ be the Hilbert space [148] of square integrable (complex) functions on the sphere S^2 . In this case, the inner product of two functions $f, g \in L^2(S^2)$ is given by

$$\langle f, g \rangle = \int_0^\pi \left[\int_0^{2\pi} f(\theta, \varphi) \overline{g(\theta, \varphi)} \right] \sin \theta d\theta. \quad (4.55)$$

The Fourier transform on the sphere uses the spherical harmonic basis functions $Y_l^m(\theta, \varphi)$ to represent any spherical function $f \in L^2(S^2)$ as

$$f = \sum_{l \geq 0} \sum_{|m| \leq l} \hat{f}_{l,m} Y_l^m, \quad (4.56)$$

where $\hat{f}_{l,m}$ denotes the (l, m) -Fourier coefficient. Hence, spherical harmonics provide an orthonormal basis for $L^2(S^2)$. The (l, m) -spherical harmonic Y_l^m ($l \geq 0, |m| \leq l$) is a harmonic homogeneous polynomial of degree l , which has a factorization

$$Y_l^m(\theta, \varphi) = k_{l,m} P_l^m(\cos \theta) e^{jm\varphi}, \quad (4.57)$$

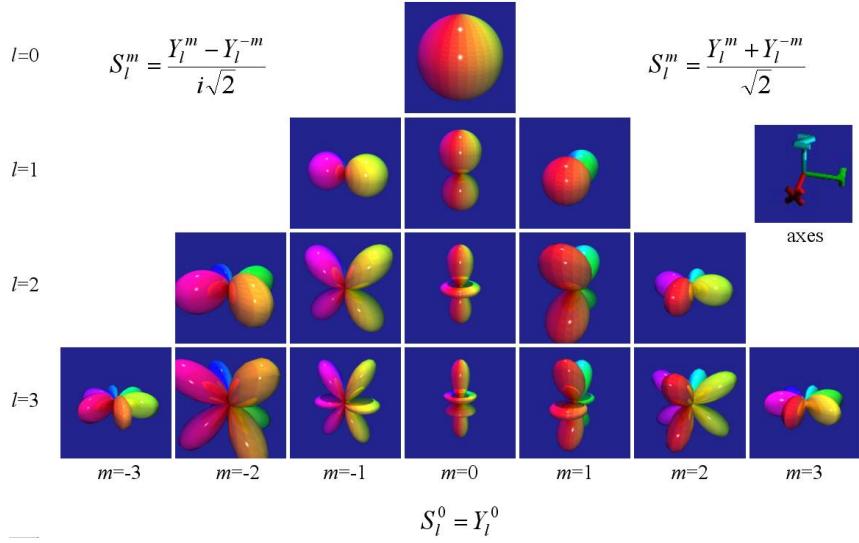
where j is the imaginary unit, $k_{l,m}$ is a normalization constant, and P_l^m is the associated Legendre polynomial of degree l and order m that satisfies the following characteristic three-term recurrent relation

$$(l - m + 1)P_{l+1}^m(t) - (2l + 1)tP_l^m(t) + (l + m)P_{l-1}^m(t) = 0. \quad (4.58)$$

A visualization of real combinations S_l^m of spherical harmonics Y_l^m , for $|m| \leq l \leq 3$, is shown in figure 4.25.

By using (4.57) to separate the longitudinal coordinate φ and latitudinal θ , the computation of the spherical harmonic coefficients can be performed as a common Fourier transform followed by a projection on the corresponding Legendre functions,

$$\hat{f}_{l,m} = \langle f, Y_l^m \rangle = k_{l,k} \underbrace{\int_0^\pi \left[\int_0^{2\pi} f(\theta, \varphi) e^{-jm\varphi} d\varphi \right]}_{\text{1D-FT}} P_l^m(\cos \theta) \sin \theta d\theta. \quad (4.59)$$



Courtesy of Quantum Chemistry Group, University of Oviedo

Figure 4.25: Spherical harmonic basis functions: a visualization of real combinations S_l^m ($l = 0, 1, 2, 3$, $-m \leq m \leq l$). The surfaces have been colored according to the value of the φ spherical coordinate of the points.

A very important property of spherical harmonics, which can be used for defining rotation invariant 3D-shape descriptors, is the following:

Property 4.1 *A subspace of $L^2(S^2)$ of dimension $2l + 1$, which is spanned by the harmonics Y_l^m ($-l \leq m \leq l$) of degree l , is invariant with respect to rotation of the sphere S^2 .*

The SFFT method proposed in [45] and the corresponding software *SpharmonicKit* [59] deal with *band-limited* functions.

Definition 4.6 *A function $f \in L^2(S^2)$ is band-limited with band-width or bandlimit $B \leq 0$ iff $\hat{f}_{l,m} = 0$ for all $l \geq B$.*

In the case of band-limited functions, the integrals (4.59) can be reduced to finite weighted sums of a sampled vector obtained from the integrand. We give the sampling theorem which is stated in [45] as theorem 1.

Theorem 4.1 *[Sampling Theorem] Let $f \in L^2(S^2)$ have bandwidth B . Then, for each $|m| \leq l < B$,*

$$\hat{f}_{l,m} = \frac{\sqrt{2\pi}}{2B} \sum_{a=0}^{2B-1} \sum_{b=0}^{2B-1} c_a^{(B)} f(\theta_a, \varphi_b) e^{-jm\varphi_b} P_l^m(\cos \theta_a), \quad (4.60)$$

where the sample points are chosen from the equiangular grid:

$$\theta_a = \frac{(2a+1)\pi}{4B}, \quad \varphi_b = \frac{2b\pi}{2B}, \quad (4.61)$$

and the weights $c_a^{(B)}$ play a role analogous to the $\sin \theta$ factor in the integrals (4.59).

Property 4.2 Let $f \in L^2(S^2)$ be a real-valued function, i.e., $f : S^2 \rightarrow \mathbb{R}$. Let the SFFT coefficients $\hat{f}_{l,m}$ be obtained by applying the sampling theorem (4.1). Then, the following symmetry between the coefficients exists

$$\hat{f}_{l,m} = (-1)^m \overline{\hat{f}_{l,-m}} \implies |\hat{f}_{l,m}| = |\hat{f}_{l,-m}|, \quad (4.62)$$

i.e., $\hat{f}_{l,m}$ and $(-1)^m \hat{f}_{l,-m}$ are conjugate complex numbers.

The complex Fourier coefficients $\hat{f}(l, m)$ can efficiently be computed by the spherical FFT (SFFT) algorithm (B is a power of 2) presented in [45], applied to samples taken at points \mathbf{u}_{ab} (4.54) defined using (4.61)

$$\begin{aligned} \mathbf{u}_{ab} &= \mathbf{u}(\theta_a, \varphi_b) = (x_{ab}, y_{ab}, z_{ab}) = (\cos \varphi_b \sin \theta_a, \sin \varphi_b \sin \theta_a, \cos \theta_a), \\ \theta_a &= \frac{(2a+1)\pi}{4B}, \quad \varphi_b = \frac{2b\pi}{2B}, \quad 0 \leq a, b \leq 2B-1, \quad B \text{ is a power of 2. } \end{aligned} \quad (4.63)$$

Hence, we need $4B^2$ samples (complex values) as the input for the SFFT, but we get B^2 complex coefficients $\hat{f}_{l,m}$ ($|m| \leq l < B$). In the case of the band-limited functions, we can reconstruct the original sample values by applying the inverse SFFT to the coefficients. If the function f is not band-limited and the forward SFFT is performed, then we cannot recover the original samples by performing the inverse transform of the obtained coefficients. However, the recovered samples can be regarded as an approximation to the original values. The quality of approximation is better for larger values of B .

As a general principle, our algorithm for defining 3D-shape descriptors based on a function on the sphere S^2 proceeds as follows.

Algorithm 4.1 A 3D-shape feature vector based on a function on the sphere S^2 is extracted by the following steps:

1. Define a function $f(\mathbf{u})$ on the S^2 ;
 2. Sample f at $4B^2$ points \mathbf{u}_{ab} defined by (4.63), $B \in \{32, 64, 128, 256, 512\}$;
 3. Perform the SFFT to obtain B^2 complex coefficients $\hat{f}_{l,m}$ ($|m| \leq l < B$);
 4. Find the magnitudes $|\hat{f}_{l,m}|$ of the obtained coefficients;

5. If f is a complex-valued function, then take the magnitudes $|\hat{f}_{l,m}|$ from the first k rows of coefficients ($|m| \leq l < k$) as components of the feature vector \mathbf{f} ;

$$\begin{aligned} \mathbf{f} = (|\hat{f}_{0,0}|, |\hat{f}_{1,-1}|, |\hat{f}_{1,0}|, |\hat{f}_{1,1}|, \dots, |\hat{f}_{k-1,-k+1}|, \dots, |\hat{f}_{k-1,0}|, \dots, |\hat{f}_{k-1,k-1}|) \\ \implies \dim(\mathbf{f}) = k^2, \end{aligned} \quad (4.65)$$

6. If f is a real-valued function, then the symmetry (4.62) exists (property 4.2). The feature vector \mathbf{f} is composed by taking the magnitudes $|\hat{f}_{l,m}|$ with positive values of the index m and half of the magnitudes $|\hat{f}_{l,0}|$ from the first k rows of the obtained coefficients ($l < k$);

$$\begin{aligned} \mathbf{f} = (|\hat{f}_{0,0}|/2, |\hat{f}_{1,0}|/2, |\hat{f}_{1,1}|, \dots, |\hat{f}_{k-1,0}|/2, \dots, |\hat{f}_{k-1,k-1}|) \\ \implies \dim(\mathbf{f}) = k(k+1)/2. \end{aligned} \quad (4.66)$$

Note that a feature vector formed either using (4.65) or (4.66) contains all feature vectors of the same type of smaller dimension, thereby providing an embedded multi-resolution representation (1.8) for 3D shape feature vectors.

Remark 4.1 The property 4.1 can be used to achieve rotation invariance of feature vectors without normalizing the orientation (rotation) of a 3D-mesh model I (1.5). Our normalization technique is described in section 3.4. After applying the first 4 steps of the algorithm 4.1, we can form the feature vector \mathbf{f} so that

$$\mathbf{f} = (||f_0||, \dots, ||f_{dim-1}||), \quad ||f_l|| = \sqrt{\sum_{m=-l}^l |\hat{f}_{l,m}|^2}, \quad dim \leq B. \quad (4.67)$$

If I_{rot} is the point set of a 3D-model obtained by rotating the points I around an arbitrary axis for an arbitrary angle, and $\mathbf{f}_{rot} = (||f'_0||, \dots, ||f'_{dim-1}||)$ is the feature vector of I_{rot} , extracted without the normalization step using (4.67), then it holds $||f_l|| \approx ||f'_l||$.

Remark 4.2 Suppose that I (1.5) is a given point set of a mesh model and I_z is the point set obtained by rotating the set I around the z -axis for an arbitrary angle. Let $\hat{f}_{l,m}$ and $\hat{f}'_{l,m}$ ($|m| \leq l \leq B$) be complex Fourier coefficients obtained after the step 3 of the algorithm 4.1, associated to I and I_z . If the number of samples is large enough ($B >> 0$) and the sample points are defined by (4.63), then the magnitudes of the obtained coefficients are approximately the same. Thus, a feature vector formed by the algorithm 4.1 is invariant with respect to rotations of a model around the z -axis. The property directly follows from the equation (4.60).

Remark 4.3 The users of the *SpharmaticKit* (version 2.5 and earlier) should correct the normalization problem that exists. Namely, after the SFFT, the obtained coefficients $\hat{f}_{l,m}$ should be scaled as follows

$$\hat{f}_{l,m} \leftarrow 2\sqrt{\pi}\hat{f}_{l,m}, \quad \hat{f}_{l,0} \leftarrow \sqrt{2\pi}\hat{f}_{l,0}, \quad 1 \leq |m| \leq l, \quad 0 \leq l < B.$$

The inverse scaling should be performed before the inverse transform. Note that without the scaling, the property 4.1 is not valid. We informed the authors of the *SpharmaticKit* about the problem, and the following versions will probably include the correction.

4.6.2 Ray-Based Approach with Spherical Harmonic Representation

The function $r(\mathbf{u})$ on the sphere S^2 defined by (4.1) is used to form the *ray-based feature vector in the spectral domain* [147]. We recall that r measures the extent of a model in the direction \mathbf{u}_{ab} (4.63). After sampling the function r at $4B^2$ points and performing the forward SFFT, we obtain B^2 complex coefficients (theorem 4.1). One may use the spherical harmonic coefficients to reconstruct an approximation of the underlying object at different levels (figure 4.26). In the first row of figure 4.26, the extent function of the original model is sampled at 1024^2 points and all 512^2 obtained coefficients are used to reconstruct the sampled model. In the second row, the number of samples is 128^2 and when the model is reconstructed using all 64^2 coefficients some artifacts are visible. We recall that the function r on the sphere S^2 is not necessarily band-limited (see definition 4.6), thus, we cannot precisely reconstruct the original samples. We observe that, for the larger number of samples, the reconstruction introduces very slight artifacts. Nevertheless, if we use both sets of coefficients (512^2 and 64^2) to reconstruct the samples using a significantly smaller number of spherical harmonics, then the reconstructed sample values are almost identical. In figure 4.26, the first 4^2 , 8^2 , 12^2 , 16^2 , 20^2 , and 24^2 harmonics are used to reconstruct the sampled function.

Since r (4.1) is a real-valued function, the symmetry in the rows of the coefficients exists (see property 4.2). Therefore, the feature vector is formed using (4.66). An example output of the absolute values of the spherical Fourier coefficients (up to $l = 4$) is given in table 4.7. The feature vector is composed from the coefficients with $m \geq 0$.

$l \setminus m$	-4	-3	-2	-1	0	1	2	3	4
0					4.448				
1				0.139	0.754	0.139			
2			0.589	0.111	1.202	0.111	0.589		
3		0.051	0.041	0.056	0.057	0.056	0.041	0.051	
4	0.060	0.053	0.369	0.004	0.554	0.004	0.369	0.053	0.060

Table 4.7: An example output of magnitudes $|\hat{r}_{l,m}|$ of spherical harmonic coefficients, when function r (4.1) is used to sample the original model from figure 4.26.

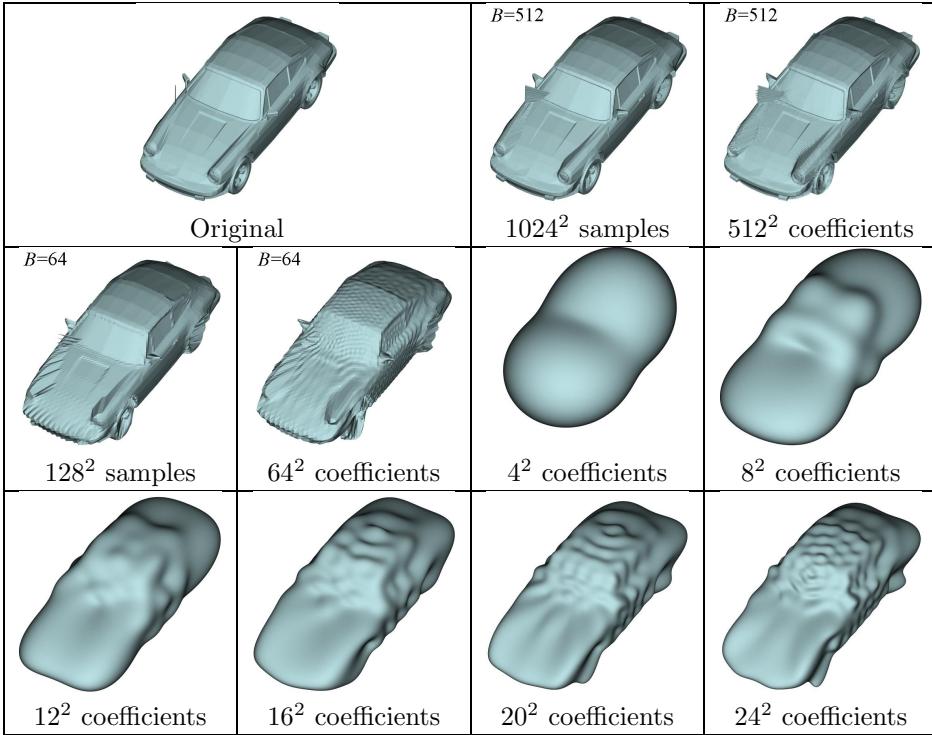


Figure 4.26: Multi-resolution representation of the function $r(\mathbf{u})$ (4.1) used to derive feature vectors from Fourier coefficients for spherical harmonics.

We recommend to take the magnitudes $|\hat{f}_{l,m}|$ from the first $k \geq 19$ rows of coefficients as components of the feature vector. We usually extract the descriptor for $k = 29$ ($0 \leq m \leq l < 29$), whence the maximal dimension of the vector is 435 and all lower-dimensional vectors are embedded. We apply the ray-triangle intersection algorithm presented in section 4.1 (algorithm 2 in figure 4.3) to sample the function r at $4B^2$ points, and use the program code [59] to perform the SFFT. We tested our feature extraction algorithm for $B \in \{32, 64, 128, 256, 512\}$ and the extraction times are shown in table 4.8. The results are obtained on a PC with an 1.4 GHz AMD processor running Windows 2000, using the MPEG-7 collection of 3D-models (section 5.1). In section 5.2.6, we compare how the number of samples affects retrieval performance. The effectiveness between tested numbers of samples is not significantly different, for $B \geq 64$. The vectors of dimensions from the range $91 \leq \dim \leq 190$ (4.66) outperform vectors of dimensions outside the range.

As mentioned in section 4.1, spherical harmonics can be used to eliminate significant component-wise differences (gaps) between feature vectors of very similar models. We recall that the ray-based descriptor is used also in the spatial domain (section 4.1). The extent function r (4.1) is sampled at *only a few* points, and the

Band-limit (B)	32	64	128	256	512
No. of samples ($4B^2$)	4096	16384	65536	262144	1048576
No. of coefficients (B^2)	1024	4096	16384	65536	262144
Extraction time [ms]	28.0	68.6	266.8	1311.2	7816.0

Table 4.8: Extraction times of the ray-based feature vector with spherical harmonic representation, for various numbers of sample points. In each case, $\dim = 435$.

$$\begin{aligned} \mathbf{f}_{400} &= (1.22, 0.13, 0.15, 0.25, 0.03, 0.29, 0.02, 0.01, 0.01, 0.15, 0.06, 0.03, 0.07, 0.03, 0.04, \\ &\quad 0.00, 0.03, 0.01, 0.07, 0.01, 0.10, 0.02, 0.03, 0.03, 0.00, 0.02, 0.03, 0.03, 0.00, 0.01, \\ &\quad 0.02, 0.03, 0.01, 0.02, 0.01, 0.02, 0.01, 0.02, 0.01, 0.02, 0.02, 0.02, 0.03, 0.03, 0.03), \\ \mathbf{f}_{800} &= (1.21, 0.13, 0.15, 0.25, 0.02, 0.29, 0.02, 0.01, 0.02, 0.15, 0.06, 0.02, 0.07, 0.03, 0.04, \\ &\quad 0.00, 0.03, 0.01, 0.07, 0.00, 0.10, 0.02, 0.02, 0.03, 0.00, 0.02, 0.03, 0.04, 0.00, 0.01, \\ &\quad 0.02, 0.03, 0.01, 0.02, 0.01, 0.03, 0.01, 0.02, 0.01, 0.02, 0.02, 0.02, 0.03, 0.03, 0.02), \\ (\|\mathbf{f}_{400}^1 - \mathbf{f}_{800}^1\|, \dots, \|\mathbf{f}_{400}^{45} - \mathbf{f}_{800}^{45}\|) &= \\ &\quad (0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00, \\ &\quad 0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, \\ &\quad 0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, \\ &\quad 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.01). \end{aligned}$$

Figure 4.27: The ray-based feature vectors ($dim = 45$) in the spectral domain of the models in figure 4.5. The maximal difference between component is equal 0.01.

sample values are directly used as components of the feature vectors. One of the problems of such a descriptor is depicted in figure 4.5, where a ray does not hit the bottom-left part of the model on the right-hand side. As a result, there is a gap between the corresponding vector components. When we sample the function r at *many* points (4.63), we also have many gaps between sample values for models in figure 4.5. However, when we apply the SFFT to the samples and form the vector in the spectral domain, then the component-wise differences between the similar models from figure 4.5 are negligible (see figure 4.27). The SFFT filters differences caused by local variances of models' surfaces.

Therefore, the spectral representation of the ray-based feature vector eliminates drawbacks that are present in the spatial domain (see section 4.1):

- Many samples capture sufficient information about the underlying object;
 - The l_1 norm is a reasonably effective distance measure in the spectral domain, because the values of samples are correlated;
 - Different levels of detail as well as different tessellations of a model introduce a noise among samples. Feature vectors in the spatial domain are not affected by the noise, which is filtered by the Fourier transform.

4.6.3 Moments-Based Feature Vector

An alternative to the representation of a spherical function by spherical harmonics is given by moments [117]. To be consistent we sample the spherical function $r(\mathbf{u})$ (4.1) at the same sample points \mathbf{u}_{ab} (4.63) as for the representation by spherical harmonics. As moments we define

$$M^{m_1, m_2, m_3} = \sum_{a=0}^{2B-1} \sum_{a=0}^{2B-1} r(\mathbf{u}_{ab}) \Delta s_a x_{ab}^{m_1} y_{ab}^{m_2} z_{ab}^{m_3}, \quad m_1, m_2, m_3 = 0, 1, 2, \dots, \quad (4.68)$$

where the factor Δs_a represents the surface area on the sphere corresponding to the sample point \mathbf{u}_{ab} (4.63) and compensates for the non-uniform sampling,

$$\Delta s_a = \frac{2\pi}{B} \left(\cos\left(\theta_a - \frac{\pi}{2B}\right) - \cos\left(\theta_a + \frac{\pi}{2B}\right) \right). \quad (4.69)$$

To compose the feature vector, we ignore $M^{0,0,0}$, and use the values of moments M^{m_1, m_2, m_3} so that $1 \leq m_1 + m_2 + m_3 \leq k$. For a fixed k , the forming of the moments-based feature vector $\mathbf{f} = (f_1, \dots, f_{dim})$ is described by the following pseudocode,

```
i = 0;
for m = 1, ..., k
    for m1 = 0, ..., m
        for m2 = 0, ..., m - m1
            fi = M^{m1, m2, m - m1 - m2};
            i ← i + 1;
```

whence the embedded multi-resolution representation is also provided (1.8). The calculation of moments $M^{m_1, m_2, m - m_1 - m_2}$ (4.68) is implemented so that the operation count is minimized. The dependency of the vector dimension dim on the parameter k is given by

$$dim = \frac{(k+1)(k+2)(k+3)}{6} - 1. \quad (4.70)$$

Usually, we set $k = 11$.

In order to measure extraction times of various dimensions of the vector, we computed feature vectors for $k = 2, \dots, 11$ using three different numbers of samples, 64^2 , 128^2 , and 256^2 . The average feature extraction times, which are shown in table 4.9, are obtained on a PC with an 1.4 GHz AMD processor running Windows 2000, and using the 3D-mesh models from the MPEG-7 set (section 5.1). We recall that the vector of dimension 363 ($k = 11$) contains all lower-dimensional vectors (for $k = 2, \dots, 10$).

Hence, the feature extraction consists of two steps: sampling and computing moments. For a given complexity of a 3D-model (number of triangles), the sampling time t_s is proportional to the number of samples, $t_s = \mathcal{O}(B^2)$, while the time t_m needed for computation of moments can be expressed as $t_m = \mathcal{O}(dim \cdot B^2)$. We can

k dim	2 9	3 19	4 34	5 55	6 83	7 119	8 164	9 219	10 285	11 363
$B = 32$	27.1	27.9	28.9	30.6	32.6	35.2	38.5	42.0	46.3	51.3
$B = 64$	59.6	62.7	67.4	74.1	82.5	92.9	106.4	121.0	138.5	158.4
$B = 128$	184.2	197.5	214.7	240.1	272.2	311.8	365.4	423.6	490.9	570.0

Table 4.9: Average extraction times (in milliseconds) of the moments-based feature vectors for various dimensions and $B \in \{32, 64, 128\}$.

write the total time complexity as $\mathcal{O}((c_f + dim)B^2)$, where c_f is a constant. The dimensionality vs. time complexity diagrams for three different numbers of samples ($4B^2$) are shown in figure 4.28.

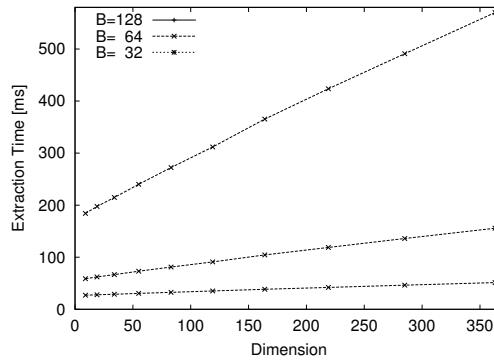


Figure 4.28: Dimensionality vs. extraction times for $B \in \{32, 64, 128\}$.

The results (see section 5.2.7) show that $dim \geq 363$ is the best choice of dimension for the moments-based feature vector. Hence, in practice, we extract only one feature vector of dimension $dim > 363$, embedding all lower-dimensional vectors.

4.6.4 Shading-Based Feature Vector

As another possibility to define a function on the sphere (4.53), which can be used for describing 3D-shape, we consider a rendered perspective projection of the object on an enclosing sphere [146]. We define a function $s(\mathbf{u})$ on the sphere S^2 relying upon the definition of $r(\mathbf{u})$ (4.1)

$$s : S^2 \rightarrow [0, 1] \\ s(\mathbf{u}) = \begin{cases} 0, & \text{if } r(\mathbf{u}) = 0 \\ |\mathbf{u} \cdot \mathbf{n}(\mathbf{u})|, & \text{if } r(\mathbf{u}) \neq 0 \end{cases}, \quad \mathbf{u} \in S^2, \quad (4.71)$$

where $\mathbf{n}(\mathbf{u})$ is the normal unit vector of the triangle containing the point $r(\mathbf{u})\mathbf{u}$ ($r(\mathbf{u}) \neq 0$). The computation of values of the function $s(\mathbf{u})$ is illustrated in figure

4.29. The triangle $\triangle ABC$ contains the furthest point of intersection along the directional unit vector \mathbf{u} . Since $\|\mathbf{u}\| = \|\mathbf{n}(\mathbf{u})\| = 1$, the value $\mathbf{u} \cdot \mathbf{n}(\mathbf{u}) = \cos \angle(\mathbf{u}, \mathbf{n}(\mathbf{u}))$ is regarded as information about flat shading [30]. Therefore, we call this descriptor *shading-based*.

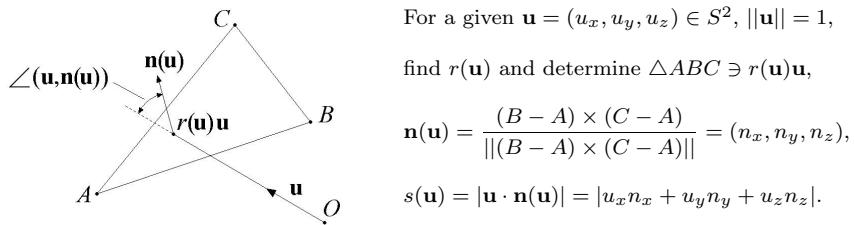


Figure 4.29: Calculation of $s(\mathbf{u})$ (4.71) when $r(\mathbf{u}) \neq 0$.

The feature extraction procedure of the shading-based vector follows the algorithm 4.1. In order to sample the function s (4.71) at $4B^2$ points \mathbf{u}_{ab} (4.63), we sample the function r (4.1) at these points, populating an additional array of $4B^2$ integers by storing the ordinal numbers of triangles containing points $r(\mathbf{u}_{ab})\mathbf{u}_{ab}$. Then, we compute the normal vectors of triangles whose indices are stored in the array, and calculate the sample values according to (4.71). The point $r(\mathbf{u}_{ab})\mathbf{u}_{ab}$ is the furthest point of the mesh model I (1.5) in the direction \mathbf{u}_{ab} from the origin O . However, $r(\mathbf{u}_{ab})\mathbf{u}_{ab}$ is the closest point of the model I to an enclosing sphere along the ray emanated from the origin and traveling in the direction \mathbf{u}_{ab} . Therefore, we regard this technique as a *rendered perspective projection of the object on an enclosing sphere*. Note that the rendered perspective projection is independent of the scale of the underlying 3D-mesh model. Therefore, it is not necessary to scale the model during the normalization step (section 3.4). After the sampling, we apply the SFFT, and form the vector according to (4.66). The feature extraction time of this vector is slightly greater than the extraction time of the ray-based feature vector in the spectral domain (see table 4.8). We suggest to set $B = 64$ and use $\dim = 91$, i.e., $k = 13$ in (4.66). Evaluation results for the shading-based descriptor with spherical harmonic representation are given in section 5.2.8.

The shading-based feature vector possesses an embedded multi-resolution representation (1.8). A drawback of the shading-based approach is sensitivity with respect to levels of detail and different tessellations of a model. Since we rely upon normal vectors, if we have a model in different levels of detail, then sample values (4.71) of a very simple model (small number of triangles) significantly differ from the samples of a very complex model (large number of triangles). This difference is reflected in vector components. Therefore, the requirement 4 in section 1.3.4 is not fulfilled.

As an example, we consider the models of cow in figure 4.30. The original model consists of 5804 triangles, while the other models are obtained by simplifying, using

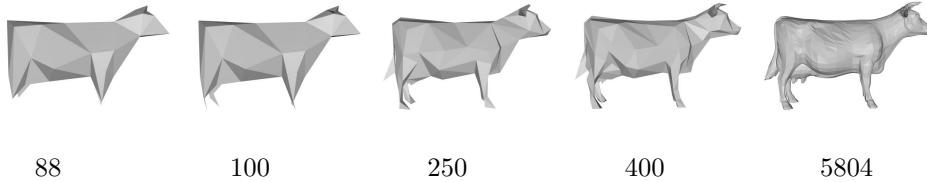


Figure 4.30: A model of cow in 5 different levels of detail. The numbers of triangles are given below corresponding models.

QSlim 1.0 software [37]. All 5 models are included in our 3D model collection (see section 5.1). Suppose the shading-based descriptor is used and the l_1 metric is selected as the distance measure to rank the models. If the model with 88 triangles is taken as a query, then the models with 100, 250, 400, and 5804 triangles are retrieved as 1st, 2nd, 4th, and 23rd match, respectively. If the model with 5804 triangles serves as a query, then the models with 400, 250, 100, and 88 triangles are at positions 1, 5, 123, and 143. Note that in both examples the retrieved models are order according to the level of detail (complexity). All the other feature vectors, which are presented in this chapter so far, do not suffer from the described problem. In other words, if we use any other feature vector and any of the five cow-models as a query, then the first four matches are the remaining cow-models.

The problem of non-robustness with respect to levels of detail and tessellations reduces discriminant power of the shading based descriptor. However, the functions $r(\mathbf{u})$ and $s(\mathbf{u})$ can be combined to define a 3D-shape descriptor, which possesses better retrieval performance than the ray-based and shading-based descriptors. The combined descriptor is presented in a sequel.

4.6.5 Complex Feature Vector

We presented two approaches (sections 4.6.2 and 4.6.4) of defining functions on the sphere S^2 (4.53), which are used for generating shape descriptors according to the algorithm 4.1. In both cases, we have a real-valued function and use (4.66) to compose vector components. In [146], we merged two features, the extent measure r (4.1) and information about shading s (4.71), by defining a complex-valued function $z_\alpha(\mathbf{u})$,

$$z_\alpha(\mathbf{u}) = \frac{1}{\alpha} r(\mathbf{u}) + j s(\mathbf{u}), \quad \mathbf{u} \in S^2, \quad z_\alpha(\mathbf{u}) \in \mathbb{C}, \quad (4.72)$$

where α is a parameter used for weighting the importance of functions r and s , and j is the imaginary unit. For $\alpha \rightarrow 0$, the extent-based feature prevails, while for $\alpha \rightarrow \infty$ the shading-based feature is predominant.

We apply the algorithm 4.1 to $z_\alpha(\mathbf{u})$. Sample values of the function $z_\alpha(\mathbf{u}_{ab})$ (4.63) are simply obtained by combining samples of $r(\mathbf{u}_{ab})$ and $s(\mathbf{u}_{ab})$. Then, the SFFT is applied to the complex-valued samples. Since $z_\alpha(\mathbf{u}) \in \mathbb{C}$, we use (4.65) to form the feature vector from the first k rows of spherical harmonic coefficients $\hat{f}_{l,m}$

(4.64). We recall that the vector of dimension k^2 embeds all lower-dimensional vectors. An example output of the absolute values of the spherical Fourier coefficients (up to $l = 4$) is given in table 4.10. The feature vector is composed from all coefficients, because the symmetry (see property 4.2 and table 4.7) between coefficients does not exist. As far as we know, this is the first attempt to describe a 3D-shape by using a complex function. Therefore, we refer to the descriptor as *complex*.

$l \setminus m$	-4	-3	-2	-1	0	1	2	3	4
0					4.041				
1				0.093	0.554	0.095			
2			0.451	0.073	0.859	0.076	0.450		
3		0.041	0.030	0.052	0.189	0.053	0.031	0.039	
4	0.063	0.038	0.272	0.026	0.399	0.027	0.272	0.040	0.062

Table 4.10: An example output of magnitudes $\hat{r}_{l,m}$ of spherical harmonic coefficients, when function $z_{0.8}$ (4.72) is used to sample the original model from figure 4.26.

The average extraction time of the complex feature vector is almost identical to the average extraction time of the shading-based descriptor (section 4.6.4), i.e., it is slightly greater than the average extraction time of the ray-based feature vector in the spectral domain (see table 4.8).

We tested (section 5.2.9) the complex feature vector of maximal dimension 484 ($k = 22$) for $B \in \{32, 64, 128, 256\}$, while the parameter α (4.72) took values 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0, 4.0, and 6.0. We recommend the following settings: $B = 64$, $\alpha = 0.4$, $\dim = 196$.

Thus, the descriptor obtained by combining two features (extent and shading) shows better retrieval performance than both the ray-based and the shading-based descriptors. Intuitively, we expect that such a combination of two features result in a descriptor, whose discriminant power is higher than the discriminant power of the shading-based, and lower than discriminant power of the ray-based descriptor in the spectral domain. However, it appears that the weaker descriptor, the shading-based, can be used to improve retrieval performance of the better descriptor, the ray-based. We explain this result by a kind of “orthogonality” (complementarity) that exists between combined features. In other words, some 3D-shape characteristics, which are not captured by one of descriptors, are recorded by the other, whence the combination gives better results. This result motivates us to create other definitions of complex functions as well as to try to merge several descriptors in an appropriate way (section 4.7).

There are other possibilities to define a complex function (4.72) on the sphere S^2 in order to combine different features by using the presented concept. For instance, the curvature index (2.8) can be used in the same way as the information about the shading (4.71). Let $c(\mathbf{u})$ be a real-valued function on the sphere S^2

$$c(\mathbf{u}) = \begin{cases} 0, & \text{if } r(\mathbf{u}) = 0 \\ \Gamma_j, & \text{if } r(\mathbf{u}) \neq 0 \end{cases}, \quad \mathbf{u} \in S^2, \quad (4.73)$$

where Γ_j is the curvature index (2.8) of triangle $T_j \ni r(\mathbf{u})\mathbf{u}$, $1 \leq j \leq m$ (1.2). Then, we form another complex function on the sphere S^2 in the same way as (4.72),

$$z'_\alpha(\mathbf{u}) = \frac{1}{\alpha} r(\mathbf{u}) + j c(\mathbf{u}), \quad \mathbf{u} \in S^2, \quad z'_\alpha(\mathbf{u}) \in \mathbb{C}, \quad (4.74)$$

and perform the previously described extraction procedure. However, we find that using (4.72) produces better descriptors than (4.74).

4.6.6 Feature Vectors Based on Layered Depth Spheres

The ray-based descriptor (section 4.1) is defined using a single function (4.1) on the sphere S^2 so that the most distant points of a mesh model I (1.5), measured from the origin in certain radial directions (4.54), determine values of the function. Hence, a number of outlying points of the model I is directly used for computing the feature vector, while all other points are used only during the normalization step. We expect that information about the interior structure of a 3D object can enhance the performance of descriptor that relies upon the extent function (4.1), which describes the distribution of outer points. By *interior structure* of a model, we refer to polygons whose points are not used for determining the function values, e.g., polygons closer to the origin or polygons bounded by other polygons.

In order to collect more data about a 3D-mesh object, than it is the case with the function (4.1), we introduce *Layered Depth Spheres* (LDSs). The concept of LDS is analogous to the concept of layered depth images (LDIs) [120], which are used as a rendering tool for 3D-scenes. An arrangement of three orthogonal LDIs is called a layered depth cube (LDC) [107] and is also used for rendering. As a contrast to both LDI and LDC, we use LDSs for describing 3D-shapes. The structure of an LDS is summarized by the conceptual representation given in figure 4.31.

To generate an LDS, we cast rays from the origin traveling in the directions \mathbf{u}_{ab} (4.63), $0 \leq a, b \leq 2B - 1$. For fixed values of a and b and a given model I , we find all intersection points $\{\mathbf{t}_{a,b,0}, \dots, \mathbf{t}_{a,b,n_{ab}-1}\}$ with I . It holds

$$\mathbf{t}_{a,b,i} = |\mathbf{t}_{a,b,i}| \mathbf{u}_{ab}, \quad \mathbf{t}_{a,b,i} \in I \cup \{O\}, \quad 0 \leq i \leq n_{ab} - 1,$$

where $|\mathbf{t}_{a,b,i}|$ denotes the distance of the point $\mathbf{t}_{a,b,i}$ from the origin. Then, we sort the values $|\mathbf{t}_{a,b,i}|$ in the non-increasing order, $|\mathbf{t}_{a,b,i-1}| \leq |\mathbf{t}_{a,b,i}|$ ($1 \leq i \leq n_{ab} - 1$). Hence, along the direction \mathbf{u}_{ab} there are $n_{ab} \geq 0$ points of intersection between the ray and the model I . A structure that we call *layered depth ray* (figure 4.31) consists of the longitudinal index a , latitudinal index b , number n_{ab} of points $\mathbf{t}_{a,b,i}$ intersecting the ray, and the sorted array of values $|\mathbf{t}_{a,b,i}|$. Finally, we define an LDS to be a structure containing a 2D-array of layered depth rays as well as the longitudinal and latitudinal dimensions, which are both equal $2B$ (4.63) in our implementation. The example in figure 4.31 depicts the concept of LDS. All points $\mathbf{t}_{a,b,i}$ of the model of human that are used to form the LDS are marked either by dots (points that are not furthest along rays) or by small squares (furthest points along rays). We recall that only points marked with small squares are taken into

```

LayeredDepthRay {
    LatitudinalIndex: integer
    LongitudinalIndex: integer
    NoOfValues: integer
    Values[0..NoOfValues]:
        sorted array of real numbers
}

LayeredDepthSphere {
    LatitudinalDim: integer
    LongitudinalDim: integer
    Rays[0..LatitudinalDim,0..LongitudinalDim]:
        array of LayeredDepthRay objects
}

```

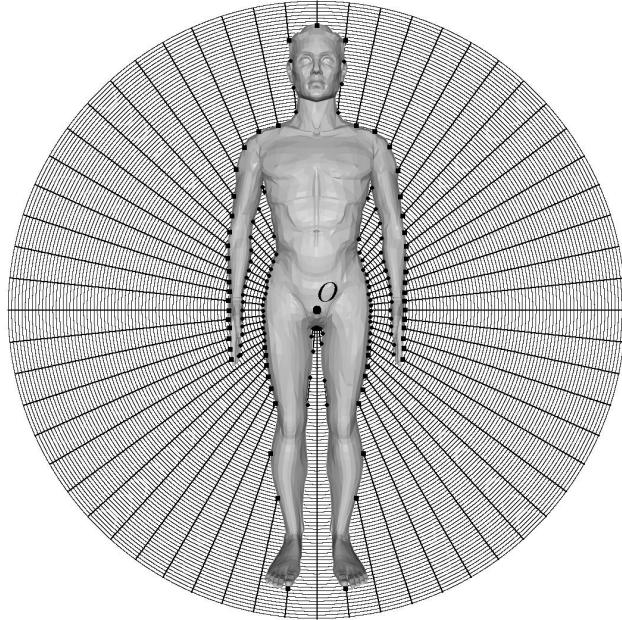


Figure 4.31: Concept of the Layered Depth Sphere with an example in the 2D-space.

account when forming the ray-based feature vector (section 4.6.2), by sampling the function r (4.1).

We define a function r_k ($k \in \{1, \dots, R\}$) on the sphere S^2 by restricting codomain of the function r given by (4.1),

$$\begin{aligned} r_k &: S^2 \rightarrow [g_k, g_{k+1}] \cup \{0\} \\ r_k(\mathbf{u}) &= \max\{ \{0\} \cup \{r_k \mid g_k \leq r_k < g_{k+1}, \mathbf{r}_k \mathbf{u} \in I\} \}. \end{aligned} \quad (4.75)$$

The lower bound g_k and the upper bound g_{k+1} restrict function values.

In our feature extraction procedure, we consider R concentric spheres S_1, \dots, S_R to define the bounds g_k of functions r_k ($k = 1, \dots, R$). The sphere S_k has the center at the origin and the radius equal to $k \cdot M/R$, where M is an empirically determined constant value. The bounds g_k of function values are given by

$$g_k = \begin{cases} 0, & k = 1 \\ \left(k - \frac{1}{2}\right) \frac{M}{R}, & 2 \leq k \leq R \end{cases}, \quad g_{R+1} \rightarrow +\infty. \quad (4.76)$$

We use the LDS structure to form sample values of R functions r_1, \dots, r_R on the sphere S^2 ,

$$r_{k,a,b} = r_k(\mathbf{u}_{ab}) = \max \left\{ \{0\} \cup \left\{ |\mathbf{t}_{a,b,i}| \mid g_k \leq |\mathbf{t}_{a,b,i}| < g_{k+1}, 0 \leq i \leq n_{ab} - 1 \right\} \right\}, \quad (4.77)$$

where \mathbf{u}_{ab} is a sample point given by (4.63) and $r_{k,a,b}$ is the sample value of the function r_k at the point \mathbf{u}_{ab} .

An example of sampling using (4.77) is depicted in figure 4.32. In this example, which is given in the 2D-space, there are three functions r_1 , r_2 , and r_3 ($R = 3$), defined using spheres S_1 , S_2 , and S_3 . There are three rays cast in the directions \mathbf{u}_1 , \mathbf{u}_2 , and \mathbf{u}_3 . The ray $r\mathbf{u}_1$ intersects the underlying model of a plane at the point $\mathbf{t}_{1,0}$, while the intersection points with rays $r\mathbf{u}_2$ and $r\mathbf{u}_3$ are $\mathbf{t}_{2,0}$, $\mathbf{t}_{2,1}$, and $\mathbf{t}_{2,2}$, and $\mathbf{t}_{3,0}$, $\mathbf{t}_{3,1}$, and $\mathbf{t}_{3,2}$, respectively. The sample values are shown on the right-hand side of figure 4.32.

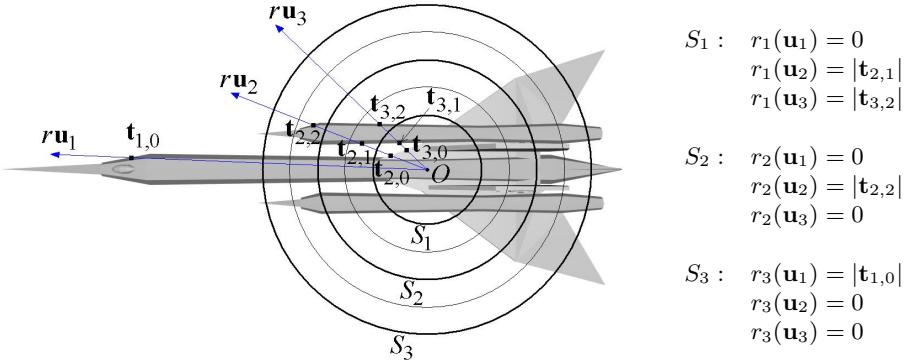


Figure 4.32: An example of sampling functions r_1 , r_2 , and r_3 , in the 2D-space.

For each function r_k , we apply the SFFT to the samples $r_{k,a,b}$ ($0 \leq a, b \leq 2B-1$). As the result, we obtain spherical harmonic coefficients $\hat{r}_{k,l,m}$ ($0 \leq l \leq B-1$, $-l \leq m \leq l$). Having in mind that $r_{k,a,b} \in \mathbb{R}$, we use (4.66) to form a *signature* of the function r_k from the first L rows (bands) of spherical harmonic coefficients. Finally, we collect the signatures of all functions r_1, \dots, r_R to form a *feature vector based on LDS*,

$$\mathbf{f} = (|\hat{r}_{1,0,0}|/2, \dots, |\hat{r}_{R,0,0}|/2, |\hat{r}_{1,1,0}|/2, \dots, |\hat{r}_{R,1,0}|/2, |\hat{r}_{1,1,1}|, \dots, |\hat{r}_{R,1,1}|, \dots, |\hat{r}_{1,L-1,0}|/2, \dots, |\hat{r}_{R,L-1,0}|/2, \dots, |\hat{r}_{1,L-1,L-1}|, \dots, |\hat{r}_{R,L-1,L-1}|). \quad (4.78)$$

The dimension of this feature vector is $\dim(\mathbf{f}) = R \cdot L(L+1)/2$. For a fixed value of R , an embedded multi-resolution representation is provided (1.8).

Regarding the implementation of the feature extraction procedure, we use the same ray-casting method that is used for extracting the ray-based feature vector

(section 4.6.2, algorithm 2 in figure 4.3). Distances to the origin of all found intersection points are stored in $4B^2$ sorted arrays, which are implemented using the STL [42]. Extraction times for various numbers of functions are given in table 4.11. The number of functions (R) takes values between 1 and 10, while the number of bands (L) is adjusted so that the vector dimension does not exceed the value of 550. The number of samples is 16384 ($B = 64$) and the constant M is set to 4. The results are obtained on a PC with an 1.4 GHz AMD processor running Windows 2000, extracting features of the 3D-mesh models from the MPEG-7 collection (section 5.1). Note that for $R = 1$ we obtain the feature vector presented in section 4.6.2. In table 4.8, the extraction time is equal to 68.6ms, while the same feature vector is extracted for 105.7ms (table 4.11), when the LDS is used. The difference of ~ 37 ms is the time needed for managing the LDS structure.

R	1	2	3	4	5	6	7	8	9	10
L	25	22	18	15	14	13	12	11	10	9
dim	325	506	513	480	525	546	546	528	495	450
Time	105.7	118.4	131.1	134.1	158.0	164.1	181.5	188.1	204.7	212.0

Table 4.11: Average extraction times (in milliseconds) of feature vectors based on LDS for various numbers of functions on the sphere S^2 . The number of functions is denoted by R , the number of bands of spherical harmonics used to form a feature vector is L . In all cases, $M = 4$ and $B = 64$.

Thus, we encode the spatial distribution of the point set I (1.5) by considering regions of the 3D-space, which are between concentric spheres. The presented feature vector based on LDS follows the standard algorithm for extracting descriptors with spherical harmonic representation, i.e., after the normalization step, the signature of the function r_k is extracted using algorithm 4.1. However, we can use property 4.1 to form a rotation invariant feature vector without normalizing the orientation of a 3D-model. We recall that the canonical orientation of a model is determined by using the CPCCA (see section 3.4). An alternative is to fix only translation and scaling during the normalization step, and after finding the spherical harmonic coefficients $\hat{r}_{k,l,m}$ ($1 \leq k \leq R$, $|m| \leq l \leq B - 1$), instead of using (4.78), we form a *rotation invariant feature vector based on LDS* according to 4.67 (see remark 4.1),

$$\mathbf{f} = (\|f_{1,0}\|, \dots, \|f_{R,0}\|, \dots, \|f_{1,L-1}\|, \dots, \|f_{R,L-1}\|), \quad (4.79)$$

where

$$\|f_{k,l}\| = \sqrt{\sum_{m=-l}^l |\hat{r}_{k,l,m}|^2}.$$

The dimension of this feature vector is $dim(\mathbf{f}) = R \cdot L$. Note that the ordering of feature vector components provides an embedded multi-resolution representation, for a fixed value of R . The average extraction times (for $1 \leq R \leq 10$) of the rotation invariant feature vector (4.79) are almost identical to the times in table 4.11.

For the descriptor formed by (4.78), we suggest to set $M = 4$ (or $M = 6$), $R = 8$, $L \geq 9$, and $\dim = 360$. As the settings for the approach (4.79), we suggest $M = 6$, $R = 8$, $L \geq 16$, and $\dim = 128$. By comparing retrieval performance of (4.78) and (4.79), we actually compare the two approaches for accomplishing rotation invariance of feature vectors, PCA vs. property 4.1. At the same time, we compare two different representations of the feature based on the LDS, (4.78) vs. (4.79). In section 5.2.10 (see also [142]), our results suggest that it is better to use our full normalization step (section 3.4), i.e., to form descriptors using (4.78).

4.7 Hybrid Descriptors

In this section, we present a concept of hybrid descriptors. Our experience suggests that no single descriptor is absolutely the best. If we manually classify 3D-models by shape-similarity, then for different categories different descriptors are the most suitable. For instance, the MPEG-7 shape spectrum descriptor (section 2.3) is the most suitable for the category of models of humans (see section 5.2.13), while its general retrieval performance is inferior to other descriptors. Thus, a natural idea is to try to merge descriptors whose performance is generally satisfying. We regard a descriptor obtained by merging (i.e., crossbreeding) two or more feature vectors (parents) as *hybrid*. Naturally, the goal of crossbreeding is to create a descriptor that possesses better retrieval performance than all parental feature vectors. Our general concept of crossbreeding is given in a sequel.

Let $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \dots, \mathbf{f}^{(k)}$ ($k \in \mathbb{N}$) be crossbreeding feature vectors of dimensions $\dim_1, \dim_2, \dots, \dim_k$, respectively,

$$\mathbf{f}^{(i)} = \left(f_1^{(i)}, f_2^{(i)}, \dots, f_{\dim_i}^{(i)} \right), \quad 1 \leq i \leq k \in \mathbb{N}.$$

A hybrid feature vector \mathbf{h} of dimension $\dim = \dim_1 + \dim_2 + \dots + \dim_k$ is defined by crossbreeding $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \dots, \mathbf{f}^{(k)}$,

$$\mathbf{h} = \left(\omega_1 f_1^{(1)}, \dots, \omega_1 f_{\dim_1}^{(1)}, \omega_2 f_1^{(2)}, \dots, \omega_2 f_{\dim_2}^{(2)}, \dots, \omega_k f_1^{(k)}, \dots, \omega_k f_{\dim_k}^{(k)} \right), \quad (4.80)$$

where ω_i is a weighting factor associated to the elements of the feature vector $\mathbf{f}^{(i)}$. The value of ω_i specifies the “importance” of the vector $\mathbf{f}^{(i)}$. The vector dimension $\dim = \sum_i \dim_i$ should be kept in a reasonable limit (e.g., $\dim < 1000$).

The question is how to select good candidates for crossbreeding. We can apply an exhaustive approach, i.e., to try all possible combinations of available feature vectors (types, variants, representations, and dimensions) with a variety of different combinations of values for the weighting factors. As the opposite approach to the exhaustive method, we use our study of techniques for describing 3D-shape to anticipate how retrieval performance can be improved. We consider that feature vectors, which possess an embedded multi-resolution representation (1.8), are suitable for crossbreeding. We recall that an embedded multi-resolution representation is usually provided by transforming a feature representation from the spatial domain into the spectral domain, using an appropriate transform, e.g., 1D-FFT (4.11) in section

4.2, 2D-FFT (4.26) in sections 4.3 and 4.4, 3D-FFT (4.48) in section 4.5, or the FFT on a sphere (see section 4.6). First components of feature vectors in the spectral domain correspond to the low-frequency Fourier coefficients, which carry the most important information. By increasing the dimension of a feature vector with an embedded multi-resolution representation, we actually add information about fine details. Thus, the fact that low-dimensional feature vectors contain useful information, as well as the requirement to keep the dimension of a hybrid feature vector inside a reasonable limit, motivate us to try crossbreeding of feature vectors with an embedded multi-resolution representation.

We also consider that crossbreeding candidates should describe features that are complementary (“orthogonal”). For instance, the ray-based feature (section 4.6.2) gives information about the extent of an object from the center of gravity along a *radial* direction, while the depth buffer-based approach (section 4.3) describes how distant is the object from a face of a cuboid by measuring the distance along a direction that is *perpendicular* to the face of the cuboid.

Instead of testing different choices of the weights ω_i and dimensions dim_i , we define the weighting factors by

$$\omega_i = \frac{dim_i}{|f_1^{(i)}| + |f_2^{(i)}| + \dots + |f_{dim_i}^{(i)}|} \Rightarrow \left\| \omega_i \mathbf{f}^{(i)} \right\|_1 = dim_i, \quad 1 \leq i \leq k, \quad (4.81)$$

so that “importance” of the vector $\mathbf{f}^{(i)}$ is specified by its dimension dim_i . Note that from (4.80) and (4.81), it follows $\|\mathbf{h}\|_1 = dim$ (1.10).

As a first choice, we combine only two descriptors ($k = 2$), the depth buffer-based feature vector of dimension 258, where the extended bounding box (EBB) is used (section 4.3), and the ray-based vector of dimension 190 with spherical harmonic representation (section 4.6.2). Hence, the obtained hybrid feature vector possesses 448 components. The reason for choosing the larger dimension for the depth buffer-based descriptor is to weight the depth buffer-based method as the more important than the ray-based approach. Since depth buffer-based descriptors possess better retrieval performance than the ray-based feature vectors (see sections 5.2.3 and 5.2.6), the hybrid obtained by crossbreeding should inherit more information from the superior parent.

An example of improving performance of original descriptors by crossbreeding is shown in figure 4.33. The ray-based, depth buffer-based, and hybrid descriptors are used to rank models by shape-similarity to the query model of bunny. There are only three more models of bunnies, which are considered relevant to the query, in a collection of 907 models. Only the hybrid approach retrieves a relevant object as the first match, which is ranked at positions 6 (ray-based) and 13 (depth buffer) when other approaches are engaged. The models of bushes, which are matches 1 and 4 in the second row, are ranked as matches 59 and 120, when the hybrid descriptor is used. Therefore, not only that relevant objects are ranked as higher matches, but some non-relevant objects, which have a high rank by a parent descriptor, are ranked significantly lower by the hybrid vector. Besides, top matches ranked by the hybrid descriptor are more coherent than top matches ranked by a parent descriptor.

More details about the given example can be seen by visiting the Web-based search engine *CCCC* presented in appendix.

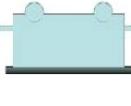
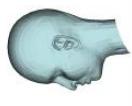
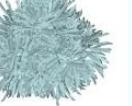
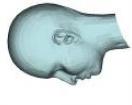
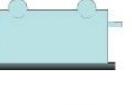
query model	match no. 1	match no. 2	match no. 3	match no. 4
				
m110.off	m1584.off 3.05	m33.off 3.26	m1805 off 3.27	m392.off 3.29
Ray-based feature vector with spherical harmonic representation ($dim = 190$)				
query model	match no. 1	match no. 2	match no. 3	match no. 4
				
m110.off	m961.off 447.39	m370.off 480.42	m333.off 494.20	m966.off 516.02
Depth buffer-based feature vector with 2D-FFT representation ($dim = 258$)				
query model	match no. 1	match no. 2	match no. 3	match no. 4
				
m110.off	m111.off 159.35	m370.off 159.62	m1584.off 163.86	m366.off 165.43
Hybrid feature vector ($dim = 448$)				

Figure 4.33: An example of improving performance of original descriptors by cross-breeding. The l_1 distances between the query model and the matches are displayed.

The example given in figure 4.33 does not represent a special case. The evaluation, which is presented in section 5.2.13, confirms that top ranked models retrieved by the hybrid descriptor are more relevant, whence the retrieval effectiveness is increased. In our experiments (section 5.2.11), we tested crossbreeding of the following four descriptors:

1. Depth buffer-based descriptor (section 4.3) based on the EBB (definition 4.3);
2. Silhouette-based descriptors (section 4.2) with equiangular sampling (4.10);
3. Ray-based descriptor with spherical harmonic representation (section 4.6.2);
4. Voxel-based descriptor in the spectral domain (section 4.5) based on the CC (definition 4.4) with $w = 2$.

We performed crossbreeding of two, three, and all four descriptors. We suggest to use the hybrid obtained by crossbreeding the depth buffer-based feature vector of dimension 186, the silhouette-based feature vector of dimension 150, and the ray-based feature vector of dimension 136. The resulting hybrid feature vector, formed using (4.80) and (4.81), possesses 472 components. Having in mind that we used heuristics (i.e., we expected that complementary descriptors with an embedded multi-resolution representation are good candidates for crossbreeding), rather than an exhaustive approach for determining the best crossbreeding combination (number of descriptors k , types, and dimensions), we assume that the chosen hybrid vector of dimension 472 is not optimal. Thus, the exhaustive approach might result in even better hybrid. Nevertheless, the experimental results (section 5.2.13) show that the hybrid feature vector of dimension 472, obtained by crossbreeding the depth buffer-based descriptor, the silhouette-based descriptor, and the ray-based feature vector with spherical harmonic representation, significantly outperforms the state of the art.

Chapter 5

Experimental Results

In this chapter, we evaluate retrieval performance of 3D-shape descriptors presented in chapters 2 and 4. Firstly, we describe 3D-model collections that are used for experiments (section 5.1). Then (section 5.2), we compare retrieval performance, using tools presented in section 1.5, for different normalization parameters, dimensions, and types of feature vectors as well as different similarity metrics defined in section 1.4. The use of the PCA as a data compression tool, which is described in section 3.2, is engaged for reducing dimensionality of combinations of feature vectors (section 5.3). Finally, all the results are summarized and the best choices for describing 3D-shape are recommended (section 5.4).

5.1 3D Model Datasets

The evaluation tools, which are defined in section 1.5, heavily rely upon the classification (1.22) of 3D-models, which is regarded as the ground truth. Unfortunately, a uniquely defined ground truth does not exist, because shape similarity is subjective. If all tests are done using a single categorization of 3D objects, then the obtained results depend on the specific 3D-model collection and the criterion of categorization. Since the topic of the thesis is retrieval of 3D-mesh models by shape similarity, the criterion of the categorization should be the 3D-shape of models, rather than semantics. In order to make our evaluation as general as possible, we use six different classifications of 3D-models (1.22) in our experiments.

Our 3D-model collection consists of 1841 mesh models in the OFF file format [96]. The models are mostly collected in the Internet (e.g., www.3dcafe.com, www.viewpoint.com, etc.). Some of them are generated by using *QSlim* [37] or modified by inserting or displacing vertices. The first classification (O1) of our collection consists of only five categories of models. The categories are 26 bottle models, 20 missiles, 63 airplanes, 33 cars, and 28 swords. In the first classification, 170 models are classified, while the rest 1671 models are left unclassified. The second classification (O2) of the same 3D data set has been performed by our colleagues, without our influence. In the second classification, 473 models are classified into 55

categories, 1362 models are left unclassified, while 6 models are considered as not suitable and are removed from the collection. The smallest class contains only 2 models, while the largest consists of 56 objects. The second classification is strictly shape-based, e.g., limousines and convertible cars are not in the same category as well as commercial airplanes, biplanes, and fighters are separated. A 3D-model from our collection possesses 5653 vertices and 10304 triangles, on average.

We also use the official MPEG-7 test set [87]. The MPEG-7 collection (M1) consists of 227 models in VRML 2.0 format [14], which are classified into 15 categories. The names of the categories are: aerodynamic (35 models), balloon (7), building (10), car (17), elm (9), finger (30), fourlimb (31), letter a (10), letter b (10), letter c (10), letter d (10), letter e (10), missile (10), soma (7), and tree (21). There are no unclassified object in the original MPEG-7 classification. We consider that the MPEG-7 test set has three major drawbacks, small size, improper selection of models, and non-consistent classification. The total number of models (227) is too small for a reliable evaluation of descriptors. The classes “finger”, “letter a”, “letter b”, “letter c”, “letter d”, and “letter e” contain totally 80 models (35% of the whole set), which are almost identical to other models in the same category. As a result, many of 3D-shape descriptors will have ideal precision-recall curves for these 6 categories (precision=100% for the whole recall range). On the other side, the category “fourlimb” contains models of humans, alligators, cows, dogs, horses, reptiles, etc., while the category “aerodynamic” contains airplanes, helicopters, dolphins, and sharks. In order to have more consistent and shape-based categorization, we re-classified (M2) the original MPEG-7 test set into 20 categories, so that 222 models are classified, while 5 are left unclassified. For a 3D mesh model from the MPEG-7 collection, the average number of vertices is 6638, while the average number of triangles is 9046.

Two 3D model collections are provided by the Princeton Shape Analysis and Retrieval Group [108]. The sets are called Training Database (TR) and Test Database. Both sets consist of 907 3D-objects and all of them are classified. The training set is subdivided into 90 classes, while the test set consists of 92 categories. In both sets, the smallest category has 4 models, while the largest category possesses 50 meshes. Both categorizations are reasonably consistent and are mostly shape-based. Nevertheless, a number of categories are formed using semantics (e.g., the categories “one story home” and “fantasy animal”), rather than shape similarity.

The complexity of 3D-models inside each of the four collections is depicted by histograms shown in figure 5.1. We take the number of vertices and triangles as parameters denoting the complexity of a 3D-mesh. The histogram values are normalized by the total number of models in the corresponding collection. We observe that for our collection, as well as for the training and the test databases, approximately 1/3 of all models possess between 1000 and 5000 triangles.

Basic information about all six classifications of 3D-models, which are used in experiments, are shown in table 5.1. We observe that the most vertices (206067) possesses a model from the MPEG-7 set, while a model from the test database is formed by tessellating 316498 triangles. The fraction of orientable models (see definition 1.2) ranges from 64.6% (classification TR) to 100% (M1 and M2). The

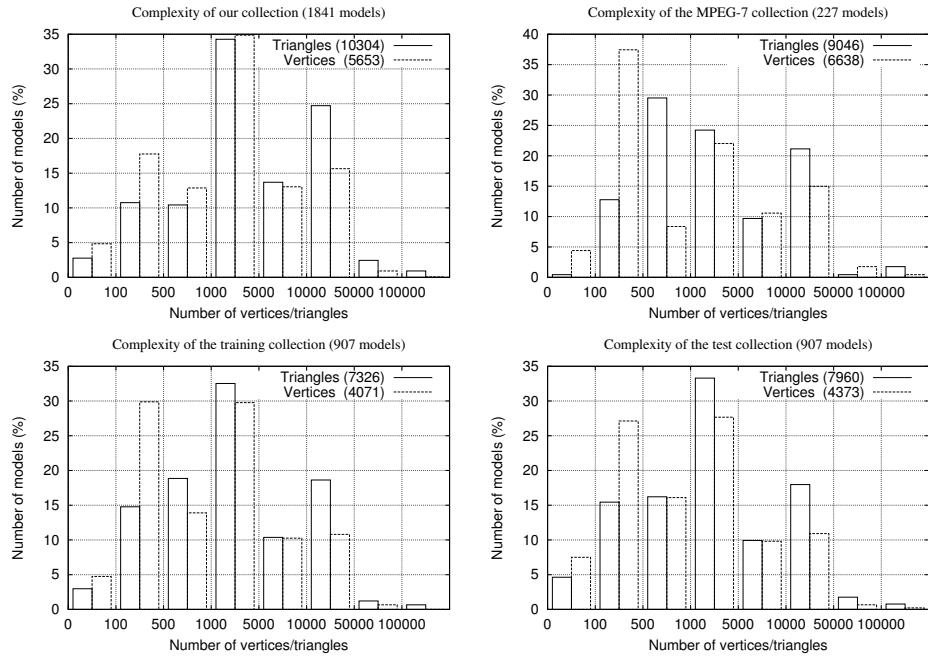


Figure 5.1: Complexity of four 3D-model collections, which are used in experiments. The average numbers of vertices and triangles are given in brackets.

fraction of closed models (see definition 1.1) ranges from 26.6% (classification TR) to 40.1% (M1 and M2). If a 3D shape description technique is restricted only to closed and orientable models, then 70% of the models from our collection, and more than 3/4 of the models from the training and test databases, are not suitable. Therefore, a 3D-shape descriptor should not have any restrictions regarding closedness and orientability. We recall that one version of our volume-based descriptor (section 4.4) relies on orientation of triangles (4.31), whence the retrieval performance of the method is low. We stress that all other descriptors, which are presented in chapter 4, do not depend on orientation of triangles.

We regard the classifications O2, TR, and TS as more relevant than O1, M1, and M2.

Usually, for a given classification, we average results (precision-recall curves, \bar{p}_{50} , \bar{p}_{100} , RP , and BEP defined in section 1.5) over all classified models, in order to compare general retrieval performance of 3D-shape descriptors. Thus, all classified models are used as queries. However, a descriptor of the best overall effectiveness is not necessarily the best descriptor for a specific class of objects. Also, the RP (1.28) and BEP (1.27) can give contradictory results, e.g., when two descriptors are compared and the one having higher RP possesses lower BEP . Another problem is the fact that we use six different classifications. If we compare two descriptors

Classification	O1	O2	M1	M2	TR	TS
No. of models	1841	1835	227	227	907	907
No. of classes	5	55	15	20	90	92
No. of classified models	170	473	227	222	907	907
No. of unclassified models	1671	1362	0	5	0	0
Average class size	34.0	8.6	15.1	11.1	10.1	9.9
Largest class size	63	56	35	30	50	50
Smallest class size	20	2	7	3	4	4
Average no. of vertices	5653	5653	6638	6638	4071	4373
Maximal no. of vertices	107155	107155	206067	206067	92583	160940
Minimal no. of vertices	4	4	33	33	28	10
Average no. of triangles	10304	10304	9046	9046	7326	7960
Maximal no. of triangles	215473	215473	219283	219283	185092	316498
Minimal no. of triangles	4	4	57	57	30	16
Closed models [%]	31.6	31.6	40.1	40.1	26.6	27.4
Orientable models [%]	68.1	68.1	100.0	100.0	64.6	66.4
Closed and orientable [%]	30.0	30.0	40.1	40.1	22.8	24.6

Table 5.1: Basic information about 3D-model classifications: O1 (the first classification of our collection), O2 (the second classification of our collection), M1 (the original classification of the MPEG-7 test set), M2 (our re-classification of the MPEG-7 test set), TR (the Princeton training set), and TS (the Princeton test set)

on two different classifications, we might get contradictory results, as well. The question is how to select reliable method to declare the retrieval performance of one feature vector as better than the other. We propose a concept to compare retrieval performance of descriptors on multiple classifications, which is given in a sequel.

Let \mathbf{f}' and \mathbf{f}'' be two competing descriptors (feature vectors), let (r'_i, p'_i) and (r''_i, p''_i) ($1 \leq i \leq G$, $G \in \mathbb{N}$) be vertices of corresponding precision-recall curves (e.g., see table 1.2), which are averaged over all classified models, and \bar{p}'_{50} , \bar{p}''_{50} , \bar{p}'_{100} , \bar{p}''_{100} , RP' , RP'' , BEP' , and BEP'' be the parameters, which are defined in section 1.5, computed for a fixed classification. The following criteria are used to distinguish the better approach:

1. If, for all six classifications, we have

$$\begin{aligned} p'_i &\geq p''_i \quad (1 \leq i \leq G), \\ \bar{p}'_{50} &> \bar{p}''_{50}, \quad \bar{p}'_{100} > \bar{p}''_{100}, \quad RP' > RP'', \quad BEP' > BEP'', \end{aligned} \quad (5.1)$$

then the descriptor \mathbf{f}' is unambiguously better than \mathbf{f}'' .

2. If the condition 5.1 is not fulfilled for all six classifications, but only for O2, TR, and TS, then we also declare the descriptor \mathbf{f}' as more effective than \mathbf{f}'' .
3. If, for at least the classifications O2, TR, and TS, we have

$$\begin{aligned} p'_i &\geq p''_i \quad (1 \leq i \leq G/2), \quad \bar{p}'_{50} > \bar{p}''_{50}, \\ \bar{p}'_{100} &> \bar{p}''_{100} - \delta, \quad RP' > RP'' - \delta, \quad BEP' > BEP'' - \delta, \end{aligned} \quad (5.2)$$

where δ is small (e.g., $\delta = 2\%$), then the descriptor \mathbf{f}' is better.

4. If the previous criteria are not satisfied, then it is difficult to distinguish which descriptor is better, and we declare the performance of competing descriptors as approximately equal (or similar).

5.2 Comparison of 3D-Shape Feature Vectors

In this section, we evaluate retrieval performance of descriptors presented in chapter 4 as well as the most of descriptors presented in chapter 2. The used evaluation tools, precision-recall (1.26), \bar{p}_{50} , \bar{p}_{100} , RP (1.28), and BEP (1.27), are defined in section 1.5. Descriptors are tested for various distance metrics (section 1.4) and different normalization methods (chapter 3). As described in section 5.1, we use six different classifications to compare the performance of the feature vectors. In what follows, we use the abbreviations from table 4.1 for our methods, while the abbreviations for implemented methods from chapter 2 are given in table 5.2.

Approach	Abbreviation
Cords-based descriptor	PCD
Moments-based descriptor	PMD
Descriptor based on equivalence classes	DEC
MPEG-7 shape spectrum descriptor	SSD
Descriptor based on “shape distributions”	SDD
Descriptor based on binary voxel grids	BVG
Descriptor based on exponentially decaying EDT	EDT

Table 5.2: Abbreviations for the descriptors from chapter 2.

Abbreviations for the six classifications of 3D-models, which are presented in section 5.1, are given in table 5.3

Classification	Abbreviation 1	Abbreviation 2
The first classification of our collection	Our DB1	O1
The second classification of our collection	Our DB2	O2
The original MPEG-7 set	MPEG-7 DB1	M1
Our modification of the MPEG-7 set	MPEG-7 DB2	M2
Princeton training database	Train DB1	TR
Princeton test database	Test DB1	TS

Table 5.3: Abbreviations for the classifications of 3D-models (section 5.1).

Abbreviations, which are used for denoting different scaling factors (section 3.4) and different distance calculations (section 1.4), are given in tables 5.4 and 5.5. Regarding the application of dissimilarity measures, we apply all distances presented in section 1.4 directly. However, we also test the application of the l_1 distance to

re-scaled feature vectors. More precisely, each feature vector is scaled by a factor ω defined by (4.81). The motivation for the re-scaling is to test the behavior of the feature vectors when the l_1 norm of the vector \mathbf{f} is fixed, $\|\mathbf{f}\|_1 = \text{const}$. We do not test a similar approach for the l_2 norm ($\|\mathbf{f}\|_2 = \text{const}$), because it can be proven that the ranking of feature vectors will be the same as when the d_2^{\min} (minL2) is used.

Scaling factor	Abbreviation
Average distance (3.25)	A
Continuous scaling factor (3.27)	C
Average distance along the x -axis (3.30)	X
Square root of the largest eigenvalue (3.31)	L
No scaling (scaling factor is equal to 1)	N

Table 5.4: Abbreviations for the scaling factors from section 3.4.

Distance calculation	Abbreviation
The l_1 norm (1.10)	L1
The l_2 norm (1.11)	L2
The l_{\max} norm (1.12)	Lmax
The minimized l_1 distance, d_1^{\min} (1.21)	minL1
The minimized l_2 distance, d_2^{\min} (1.20)	minL2
The quadratic form distance d_2^S (1.15)	QFD
The l_1 is applied after re-scaling each feature vector by (4.81)	L1, scaled

Table 5.5: Abbreviations for the distance metrics from section 1.4.

Only a fraction of all generated precision-recall diagrams is shown. The strategy of presenting the evaluation is the following. For each descriptor, we first present precision-recall curves for different dimensions of feature vectors, with appropriate parameter settings, and try to infer the best resolution (dimension) of the vector. Then, we demonstrate that certain settings are the best choice, by comparing different variants of the approach. Finally, we compare different distance calculations. When the best representative (dimension and parameter settings) of a technique is selected, for all our approaches (chapter 4), we compare the best representatives to each other. Afterwards, we compare our best descriptors with the state of the art (chapter 2). In most cases, we give diagrams for four classifications of 3D-models, O1, O2, TR, and TS, while the results obtained using the classifications of MPEG-7 models, M1 and M2, are only occasionally presented.

We also stress that:

- In precision-recall diagrams, the dimension of a feature vector as well as the values of \bar{p}_{50} , \bar{p}_{100} , BEP, and RP (section 1.5) are respectively given in brackets;
- If not explicitly stated, the normalization of a mesh model is done by our con-

tinuous approach (section 3.4), where the average distance (3.25), from a point on the surface to the center of gravity, is taken as the scale factor.

As an example, the caption “RAY-A, Our DB1, L1” of a precision recall diagram is sufficient to specify the type (the ray-based feature vector in the spatial domain), the scaling (A), the used classification (our original), and the distance calculation (l_1).

5.2.1 Ray-Based Feature Vector

In order to determine the best choice of vector dimension, we test the *ray-based feature vector in the spatial domain* (section 4.1) in the following resolutions (dimensions): 12, 42, 92, 162, 252, 362, 492, and 642. We recall that the dimension of the vector is given by the formula $\dim = 10k^2 + 2$, thus we took $k = 1, \dots, 8$. Average precision/recall diagrams of four classifications, O1, O2, TR, and TS (table 5.3), for the selected dimensions of the ray-based feature vector, are shown in figure 5.2. For $\dim \geq 42$, the retrieval performance of the descriptor is similar regardless of the dimension. Nevertheless, we consider that the ray-based feature vector of dimension 162 ($k = 4$) slightly outperforms vectors of other dimensions.

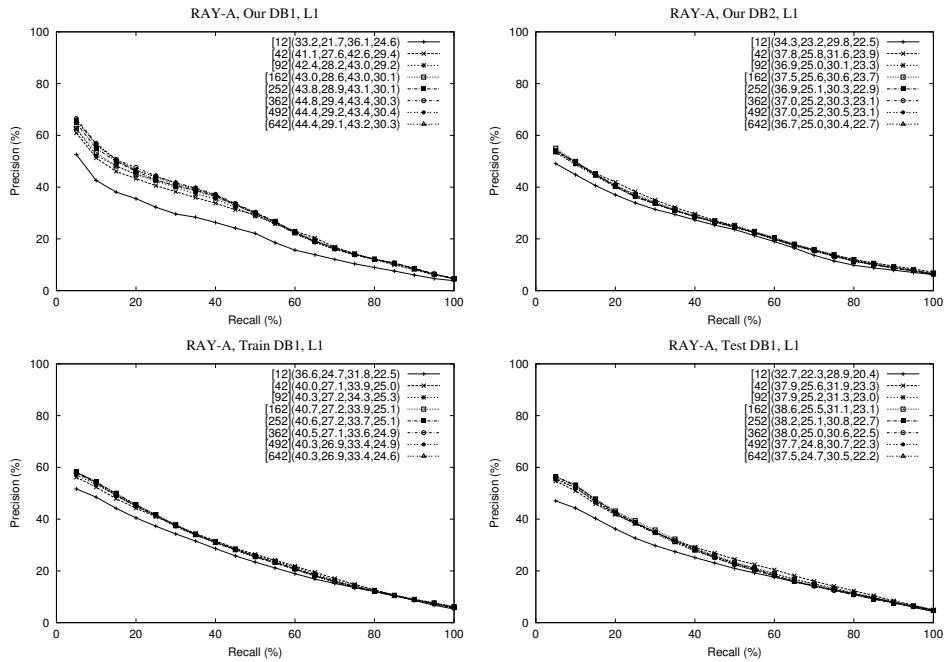


Figure 5.2: Average precision/recall diagrams of four model classifications for various dimensions of the ray-based feature vector (section 4.1).

As dissimilarity measures, we examine the distances discussed in section 1.4, i.e., the l_1 , l_2 , and l_∞ norms, as well as the minimized l_1 and l_2 distances, d_1^{min} and d_2^{min} (definition 1.3). We also test a version of the quadratic form distance d_2^S (1.15), so that the elements of the matrix S are defined by

$$s_{ij} = \begin{cases} \exp(-2kD(i,j)), & k^2 D(i,j) < 4.8, \\ 0, & k^2 D(i,j) \geq 4.8, \end{cases} \quad D(i,j) = (\mathbf{u}_i - \mathbf{u}_j)^2, \quad (5.3)$$

where the directional unit vectors \mathbf{u}_i and \mathbf{u}_j ($1 \leq i, j \leq 10k^2 + 2$) are given by (4.4). In figure 5.3, we compare the l_1 , l_2 , d_1^{min} , and d_2^S (5.3). We observe that d_2^S gives slightly better results than the l_1 distance. The minimized l_1 distance d_1^{min} is more effective than the l_1 norm only when our reclassified collection is used. In all cases, the l_2 norm is the most inferior distance metric.

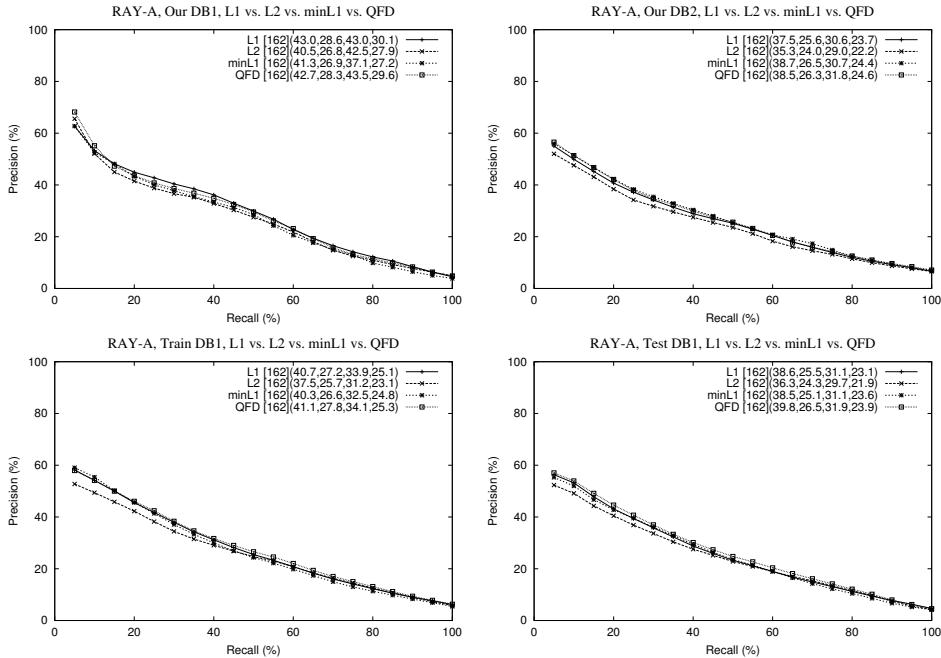


Figure 5.3: Average precision/recall diagrams of the ray-based feature vector ($dim = 162$) for different distance metrics (table 5.5).

On average, if the feature vector of dimension 162 is used, the ranking according to the similarity with a query, i.e., computation and sorting of distances between the query and all other models in the collection, is approximately two times faster when the l_1 or l_2 metrics are used instead of d_1^{min} and d_2^S . Relative computational costs for calculating distances between vectors of different dimensions, without sorting,

are given in table 5.6. We read that, e.g., for dimension 162, the computation of the quadratic form distance d_2^S , defined by (1.15) and (5.3), is 2.24 times more expensive than the computation of the l_1 norm, which is 2.04 times (on average) faster than the calculation of the minimized l_1 distance, d_1^{min} .

Dimension	12	42	92	162	252	362	492	642
l_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
l_2	1.11	1.06	1.06	1.06	1.07	1.08	1.08	1.08
d_1^{min}	1.56	1.82	1.88	2.04	2.10	2.17	2.21	2.29
d_2^S (5.3)	1.56	2.00	2.00	2.24	2.33	2.32	2.39	4.08

Table 5.6: Average cost factors for calculating distances between ray-based feature vectors, for different dimensions. Results are normalized by the average computational time for the l_1 distance.

In order to demonstrate that the scaling by (3.25) is the best choice, we display the precision-recall diagrams on the left side of figure 5.4. The results are obtained using the l_1 norm as distance metric. On the right-hand side of figure 5.4, the d_1^{min} is used as distance metric for feature vectors, which are extracted using different scale factors as well as without scaling. We observe that all five curves are exactly the same. Hence, if the d_1^{min} is used for measuring dissimilarity between ray-based descriptors, we do not need to scale 3D-mesh models during the normalization step.

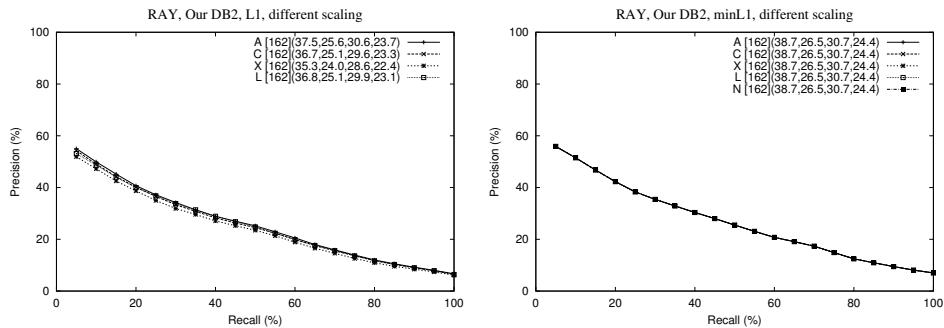


Figure 5.4: Average precision/recall diagrams of our reclassified collection, when the ray-based feature vectors are extracted using different scale normalization techniques (table 5.4). Used distances are l_1 and d_1^{min} .

The minimization of the l_1 distance (similar for the minimization of l_2) can be interpreted as an indirect rescaling of 3D-objects. We recall that the minimization of the l_1 distance $d_1^{min}(\mathbf{f}_q, \mathbf{f}_c)$ (1.21), between the feature vector \mathbf{f}_q of a query model and the vector \mathbf{f}_c of a matching candidate, is a computation of the optimal scale factor α for \mathbf{f}_c , so that $\|\mathbf{f}_q - \alpha\mathbf{f}_c\|_1$ (1.10) is minimal. Since the function r (4.1) measures the

extent of a model, along a given directional unit vector \mathbf{u} , if the candidate model is rescaled by α , then the ray-based feature vector \mathbf{f}_c is also rescaled by the same factor (α). In this case, the l_1 distance between the query and candidate vectors is equal to the minimized l_1 distance d_1^{\min} . Therefore, the results depicted on the right-hand side of figure 5.4 are expected, because the $d_1^{\min}(\mathbf{f}_q, \mathbf{f}_c)$ distances, when different initial scaling factors are used, are proportional.

We recommend to use the ray-based feature vector in the spatial domain of dimension 162, to fix the scale by (3.25), and to use the d_2^S , defined by (1.15) and (5.3), as dissimilarity measure.

5.2.2 Silhouette-Based Feature Vectors

In this section, we compare different variants of the *silhouette-based feature vectors* (section 4.2). If not explicitly stated, all silhouette-based descriptors are extracted from silhouette images of dimensions 256×256 ($N = 256$), the number of sample points is equal to 256 ($K = 256$), the sample points are formed using (4.10), and the sample values are defined using (4.13).

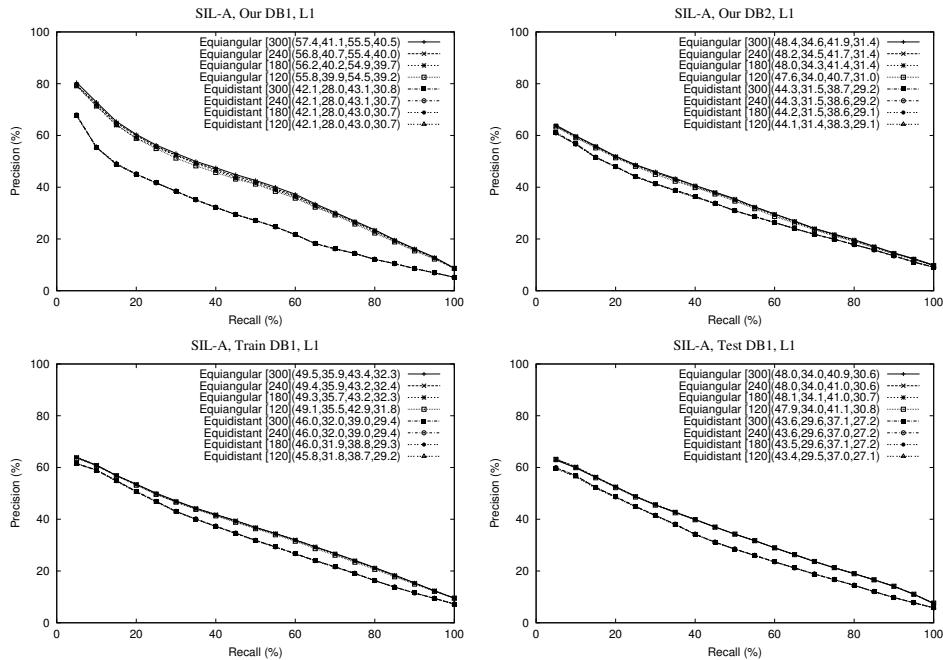


Figure 5.5: Average precision/recall diagrams of four model classifications, using the silhouette-based feature vectors with different approaches for selecting contour points: equiangular (4.10) and equidistant (4.9).

We compared the performance of the silhouette-based feature vectors of various dimensions (30, 60, 90, 120, 150, 180, 210, 240, 270, and 300), for both approaches of selecting contour points, depicted in figure 4.8. In figure 5.5, we present a part of our experimental results. Four model classifications, O1, O2, TR, and TS (table 5.3), are used for comparing vectors of dimensions 120, 180, 240, and 300, as well as the two proposed sampling methods, (4.10) and (4.9). The shown results suggest that the better approach is to sample equiangular points on the contour (figure 4.8b), than to have adjacent samples with equal arc distances (figure 4.8a). We consider that the silhouette-based feature vector of dimension 300 slightly outperforms the competing vectors.

In another set of experiments, we examined the influence of different normalization parameters on retrieval effectiveness. On the left-hand side of figure 5.6, average precision-recall diagrams, for the silhouette-based feature vectors of dimension 300, are shown. The feature vectors are extracted after using different scale factors, the average distance (3.25), the continuous scale factor (3.27), the average distance along the x -coordinate (3.30), and the square root of the largest eigenvalue of the corresponding covariance matrix (3.31). The results show that the best choice of fixing the scale is to use the average distance (3.25), while the continuous scale factor is better than the remaining two approaches.

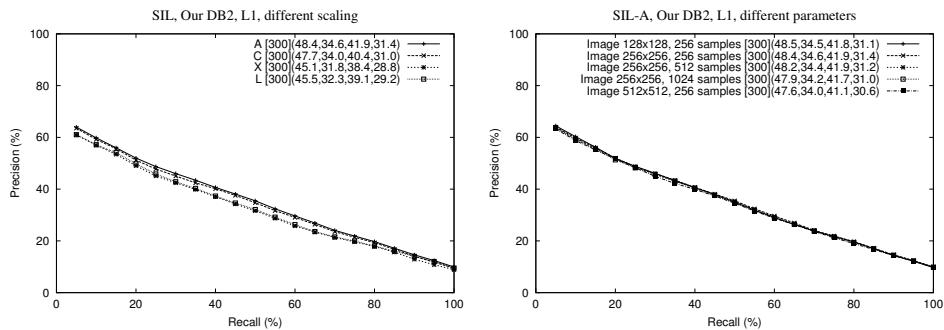


Figure 5.6: Average precision/recall diagrams of our reclassified collection, for the silhouette-based feature vectors. On the left-hand side, different scale normalization techniques are used (table 5.4). On the right-hand side, different parameter settings, image dimensions and number of sample points, are tested.

On the right-hand side of figure 5.6, we tested different parameter setting for the silhouette-based feature vector of dimension 300, which is extracted using (3.25) as the scale factor. We examined the following dimensions of silhouette images, 128×128 , 256×256 , and 512×512 , and the following number of equiangular sample points, 256, 512, 1024. Although the precision-recall curves look almost the same, we consider that the descriptor extracted from 256 sample points of silhouette images of dimensions 256×256 is slightly better than others. Note that

only results obtained using our reclassified 3D-model collection are shown in figure 5.6. Nevertheless, all the conclusions, which are inferred using our collection, are also supported by the results obtained using the training and test databases. Other precision-recall diagrams are omitted to save space.

In figure 5.7, we explore the influence of distance metric (table 5.5) on the retrieval performance of the silhouette based feature vector of dimension 300. Average results of four classifications, O1, O2, TR, and TS (table 5.3), are shown. We observe that the l_1 distance applied to re-scaled feature vectors (“L1, scaled”) is the most effective for ranking the models. The d_1^{min} (minL1) dissimilarity measure is more effective than the l_1 norm directly applied. Both minimized distances, d_1^{min} and d_2^{min} , are more effective than the original distances, l_1 and l_2 . Finally, the l_{max} norm is not appropriate distance metric. Having in mind the definition of the descriptor, the inefficiency of the l_{max} norm can be anticipated.

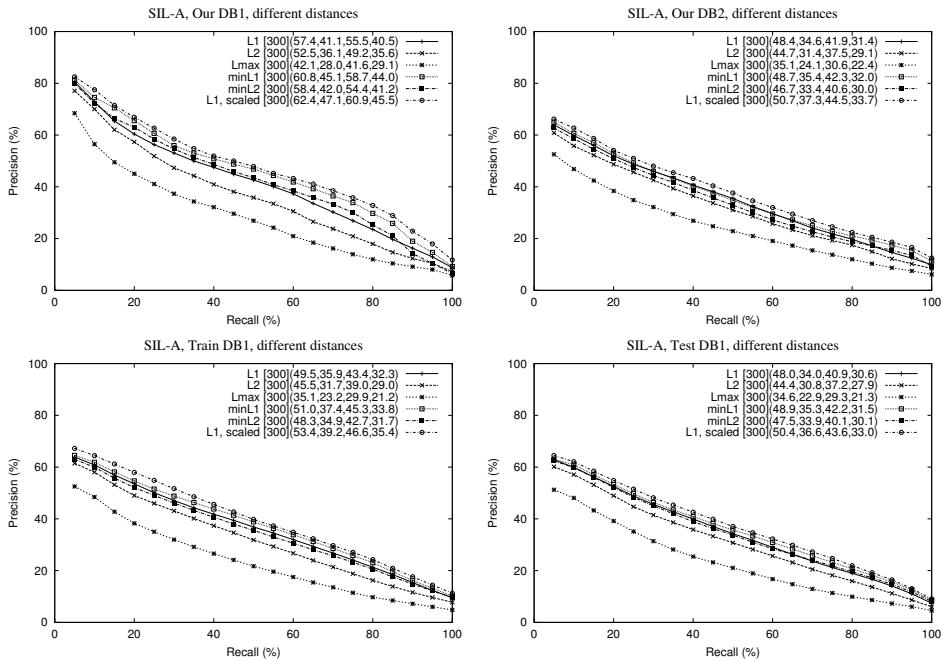


Figure 5.7: Average precision/recall diagrams of four model classifications, when different distance calculations from section 1.4 (table 5.5) are applied to the silhouette-based feature vector.

The average computational costs for different distance metrics are compared in table 5.7 (compare to table 5.6). All average costs are normalized by the average time needed to compute the l_1 distance. The computational costs for ranking using the l_1 distance are approximately two times lower than the costs for ranking by d_1^{min} .

Distance	l_1	l_2	l_{max}	d_1^{min}	d_2^{min}	l_1 , scaled
Relative cost factor	1.0	1.1	1.0	2.3	1.2	1.0

Table 5.7: Average cost factors (normalized by the average computational time for the l_1 distance) for calculating distances (table 5.5) between silhouette-based feature vectors of dimension 300.

If we apply the minimized distance d_1^{min} to the silhouette-based feature vectors, extracted using different scale factors (table 5.4), then we expect to obtain coinciding precision-recall curves, because of the indirect rescaling of 3D-meshes (see remarks about figure 5.4). Similar results are expected for the d_2^{min} measure as well as for the l_1 distance applied after re-scaling each feature vector (L1 scaled). The anticipated results are verified by the diagrams shown in figure 5.8. Thus, if “L1 scaled”, d_1^{min} , or d_2^{min} are used, no scaling is necessary.

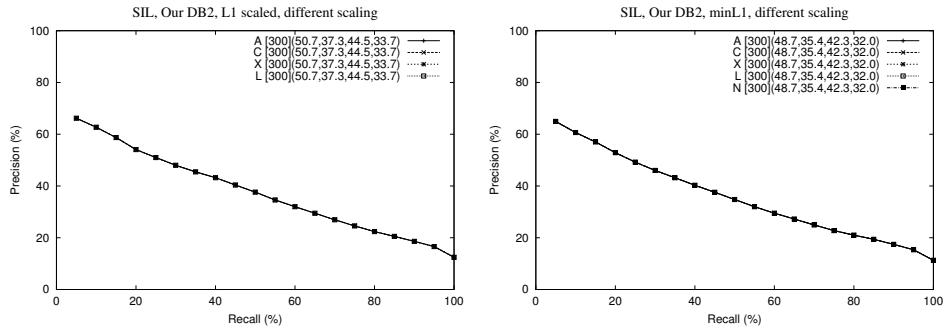


Figure 5.8: Average precision/recall diagrams of our reclassified collection, when the silhouette-based feature vectors are extracted using different scaling factors (table 5.4). Dissimilarities are calculated by d_1^{min} and d_2^{min} (table 5.5).

Finally, we test three different choices for defining sample values, which are used as the input for the discrete fast Fourier transform (4.11). The three different definitions of sample values are given by (4.12), (4.13), and (4.14). In figure 5.9, results for classifications O1, O2, TR, and TS (table 5.3) are shown. We observe that, for each classification, the average precision-recall curves of three sampling methods are very close to each other. Nevertheless, there are slight differences, typically less than 0.5%, between precision-recall curves. Unfortunately, these differences are contradictory for different classifications. For instance, the method 2 (4.13) is the best, when the training database is used, while the method 1 (4.12) is the best for the test database. Therefore, we consider that any of three possibilities can be engaged giving approximately the same retrieval results. We decided to use method 2, in which the values of samples are defined by (4.13).

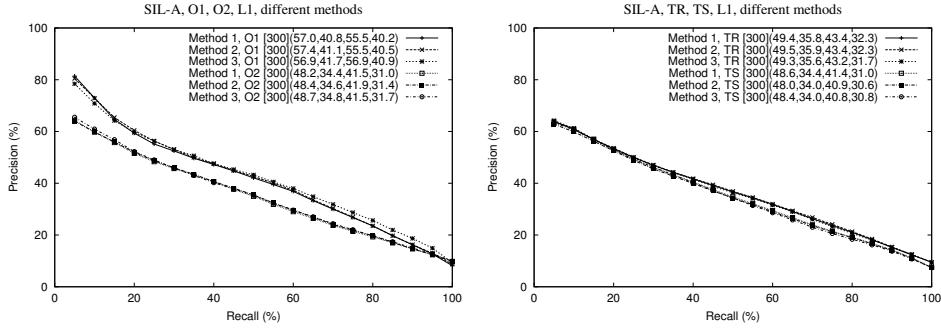


Figure 5.9: Average precision/recall diagrams of four classifications, when the silhouette-based feature vectors are extracted using different definitions of sample values: Method 1 (4.12), Method 2 (4.13), and Method 3 (4.14).

We recommend to use the silhouette-based feature vector of dimension 300, which is extracted without scaling a 3D-model in the normalization step, and to apply the l_1 norm to re-scaled (normalized) feature vectors. We suggest the following settings: dimensions of silhouette images – 256×256 , the number of sample points – 256, the sampling technique – equiangular (4.10), and the definition of sample values given by (4.13).

5.2.3 Depth Buffer-Based Feature Vector

We explored all variants of the *depth buffer-based feature vector* (section 4.3). Firstly, we test a version of the depth buffer-based descriptor, which is robust with respect to outliers. More precisely, we use the faces of the canonical cube (CC) (definition 4.4) to form the depth buffers, where we have set $w = 2$. We recall that the robustness of the descriptor relying upon the CC is demonstrated in figure 4.13. The normalization has been done by our continuous approach (section 3.4), where (3.25) is used to fix scale. In what follows, the 2D-FFT (4.26) is applied to depth buffer images of dimensions 256×256 , if not explicitly stated otherwise. In figure 5.10, we give precision-recall curves for different dimensions of the feature vector. The tested dimensions are 18, 42, 78, 126, 186, 258, 342, and 438. Note that only the feature vector of dimension 438 ($k = 8$ in (4.29)) is extracted embedding all lower dimensional vectors. Results for classifications O1, O2, TR, and TS (table 5.3) are depicted. We conclude that the best choice of dimension is 438.

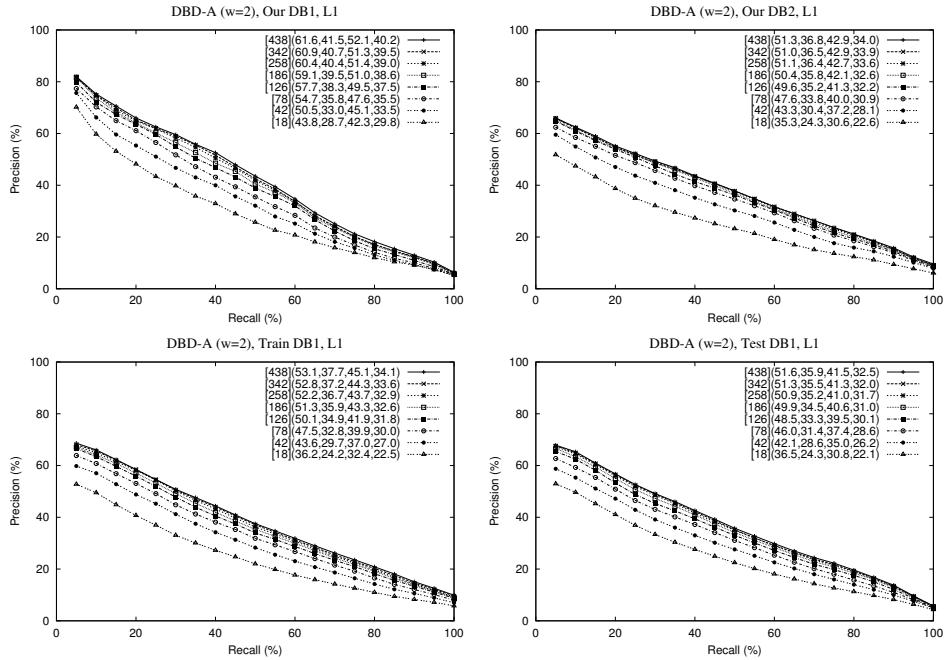


Figure 5.10: Average precision/recall diagrams of four model classifications, for various dimensions of the depth buffer-based feature vectors. Depth buffer images are formed using the canonical cube, defined by (4.25), with $w = 2$.

A selection of results, aimed at determining the optimal choice of distance metric, is displayed in figure 5.11. The four classifications, O1, O2, TR, and TS, are used. The best approach for ranking the models is to apply the l_1 norm after the rescaling (L1 scaled). As a contrast to results shown in figure 5.7, the distances l_2 and d_2^{min} are not inferior to l_1 and d_1^{min} . However, the retrieval performance is increased if the corresponding distance is minimized. We stress that the relative costs for computing dissimilarity between two feature vectors using the d_2^{min} (1.20) are approximately 1.2 times higher than the costs for computing the l_1 distance (see table 5.7). The d_1^{min} and d_2^{min} are approximately equally effective. We consider that the l_1 distance is slightly more effective than the l_2 norm. Finally, the l_{max} norm is not suitable and should not be used.

Next, we examine the use of different cubes for generating depth buffer images, which are described in section 4.3. Two versions of the descriptor, which are not robust with respect to outliers, are formed using the canonical bounding cube (CBC), defined by (4.6), and using the extended bounding box (EBB), defined by (4.24). We tested four versions of the approach, when the canonical cube (CC), defined by (4.25), is used (for $w \in \{2, 4, 8, 16\}$). In all cases, the scale is fixed by the average

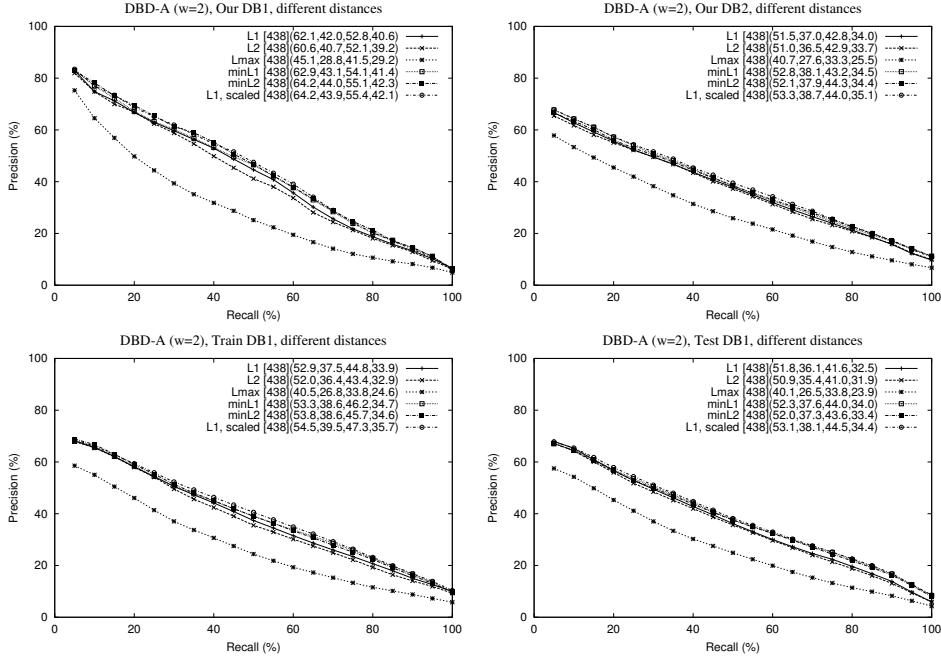


Figure 5.11: Average precision/recall diagrams of four model classifications, when different distance metrics (table 5.5) are applied to the depth buffer-based feature vector. Depth buffer images are formed using the canonical cube, defined by (4.25), with $w = 2$.

distance (3.25). Note that there is no need for scaling during the normalization step, when the CBC and EBB are used. The results for our reclassified collection are shown in figure 5.12. The l_1 and d_1^{\min} distances are used for ranking. We observe that the best version of the depth buffer approach is obtained, when the EBB is used. As expected, the retrieval performance of the descriptor, which relies upon the CC, deteriorates with the increase of w . We recall that, when the cube defined by (4.25) is used, only the part of a mesh inside the cube is processed. The points of the model I (1.5), which are outside the cube, are effectively ignored as outliers. By setting $w = 2$, we practically crop certain number of models. If we increase the value of w , the number of ignored points (parts of a mesh) decreases. However, the increase of w deteriorates the retrieval performance of the descriptor, because the object occupies a smaller part of the corresponding depth buffer image. In other words, for a large value of w , the representation of a 3D-model by depth buffer images is too coarse, whence shape characteristics of the underlying 3D-model cannot be captured in an appropriate way. Thus, the best choice is to set $w = 2$. The performance of the descriptor relying upon the CBC is weaker than performance of descriptors based on the EBB or CC with $w = 2$. Regardless of the obtained

results, we suggest to use the descriptor, which is robust with respect to outliers (requirement 5 from section 1.3.4), relying upon the CC with $w = 2$. The reason for giving advantage to the CC is depicted by the example in figure 4.13.

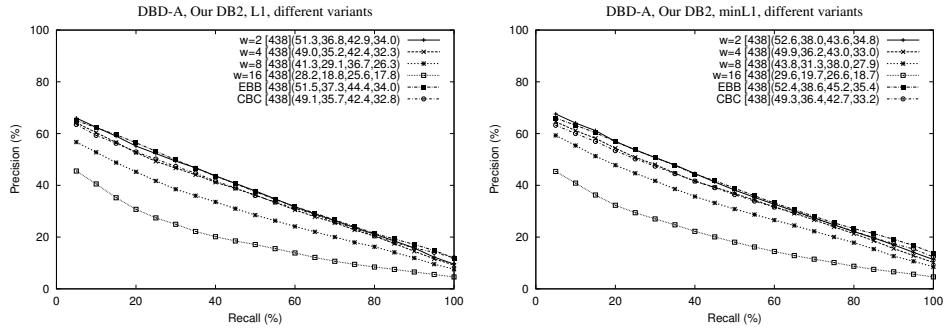


Figure 5.12: Average precision/recall diagrams of our reclassified collection, for different parameter settings of the depth buffer-based feature vectors, which are described in section 4.3. Used distance metrics are l_1 and d_1^{min} (table 5.5).

In figure 5.13, we test different scaling factors (table 5.4) in the normalization procedure. After the normalization, the feature vectors are extracted from 256×256 depth buffer images, using the canonical cube defined by (4.25) with $w = 2$. The l_1 and d_1^{min} distances are used for ranking. Obviously, the average distance (3.25) is the best choice of the scaling factor.

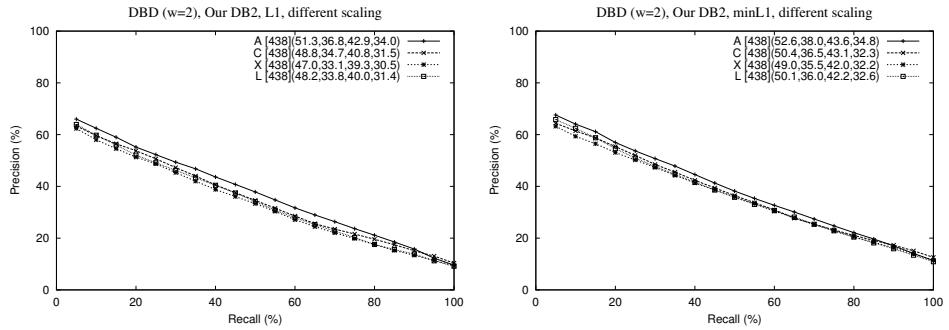


Figure 5.13: Average precision/recall diagrams of our reclassified collection, when the depth buffer-based feature vectors are extracted using different scale factors (table 5.4). Used distance metrics are l_1 and d_1^{min} (table 5.5). Depth buffer images are formed using the canonical cube, defined by (4.25), with $w = 2$.

In figure 5.14, we test the influence of dimensions of the depth buffer images on the retrieval performance. We tested images of types 64×64 , 128×128 , 256×256 , and 512×512 , and no significant difference in retrieval performance is present. However, we consider that image dimensions 256×256 are slightly better than other choices. We have also performed tests, which are analogous to the ones from figures 5.12, 5.13, and 5.14, using the training (TR) and test (TS) databases. We stress that the obtained results support conclusions, which are inferred using our reclassified collection.

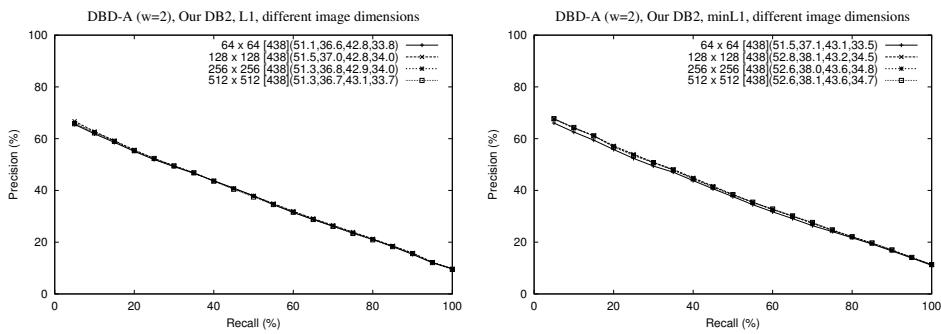


Figure 5.14: Average precision/recall diagrams of our reclassified collection, when the feature vectors are extracted from depth buffer images of dimensions 64×64 , 128×128 , 256×256 , and 512×512 . Used distance measures are l_1 and d_1^{min} . Depth buffer images are formed using the canonical cube, defined by (4.25), with $w = 2$.

We recommend to use the depth buffer-based feature vector of dimension 438, extracted in the canonical coordinate frame after scaling a model by (3.25), and using the CC (definition 4.4) with $w = 2$ to form depth buffer images of dimensions 256×256 . We propose to rank feature vectors by applying the l_1 distance after the re-scaling (4.81).

5.2.4 Volume-Based Feature Vector

The *volume-based descriptors* (section 4.4) have been tested in all four variants (table 4.5) in both spatial and spectral domains. Firstly, we determine the best choice of dimension of the variant V4 in the spatial domain, when non-negative volumes $|V_{T_j}|$ (4.31) are used and vectors \mathbf{f} are scaled so that $\|\mathbf{f}\|_1 = 1$. In figure 5.15, we show results obtained using different dimensions of the feature vector on classifications O1, O2, TR, and TS (table 5.3). We recall that the dimension dim depends on a parameter k , so that $dim = 6k^2$. Also, there is no need for scaling during the normalization step (section 3.4), when the variant V4 is used. The results suggest that the feature vector of dimension 294 ($k = 7$) outperforms the competing descriptors of dimensions 54, 96, 150, 216, and 384. The feature vector of

dimension 486 performs very similar to the feature vector of dimension 294. In the case of almost identical performance of two descriptors, we give advantage to the lower-dimensional feature vector. We also observe that the feature vectors whose dimensions are fixed by an odd value of the parameter k perform better than the vectors whose dimension is fixed by an even value of the parameter k . This observation can be explained by the fact that, for odd values of k , after the subdivision of the 3D-space (figure 4.16), the coordinate hyper-planes are not boundaries of the obtained regions ν_i (4.37), rather they are located in the middle of a number of regions ν_i .

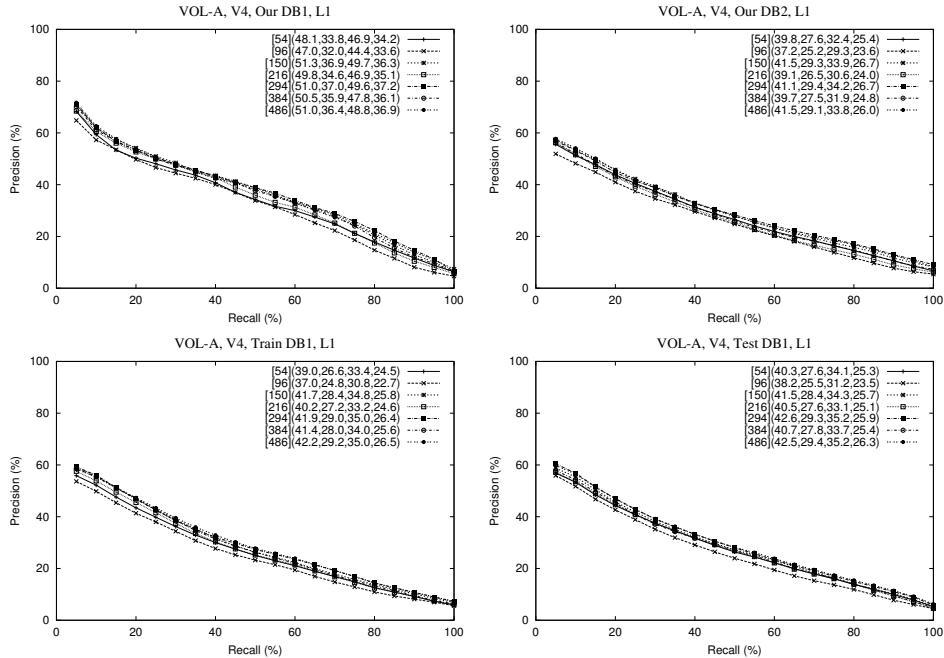


Figure 5.15: Average precision/recall diagrams of four model classifications, for various dimensions of the variant V4 (table 4.5) of the volume-based feature vector in the spatial domain.

The four variants (table 4.5) of the volume-based approach in the spatial domain are evaluated in figure 5.16. We recall that the variants V1 and V3 rely upon the orientation of triangles, because the signed values of the volume V_{T_j} (4.31) are used, while the variants V2 and V4 deal with the non-negative volumes $|V_{T_j}|$. Also, the variants V1 and V2 are extracted after fixing the scale of a 3D-mesh model by the average distance (3.25), while the feature vector \mathbf{f} is “post normalized” so that $\|\mathbf{f}\|_1 = 1$, when the variants V3 and V4 are used. The results obtained using our reclassified collection (O2) and the test database (TS) are shown in figure 4.5.

Results obtained for other classifications (O1, TR, M1, and M2) comply with the presented results. According to the shown precision-recall diagrams, the variant V4 is the best, while the variant V2 is better than the rest two. The variant V3 outperforms the variant V1. Thus, if we rely upon orientation of triangles and upon a canonical scale, the feature vector shows extremely poor retrieval performance (V1). By normalizing the values of vector components, we obtain a better descriptor (V3). However, by having non-negative volumes without post normalization, the retrieval performance is even better (V2). The best solution is to have both non-negative volumes and post normalized vector components (V4).

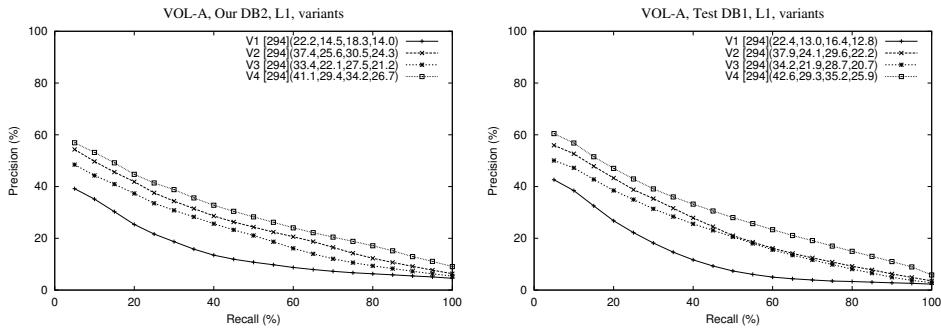


Figure 5.16: Average precision/recall diagrams of our reclassified model collection and the test database, for all four variants (table 4.5) of the volume-based feature vector in the spatial domain.

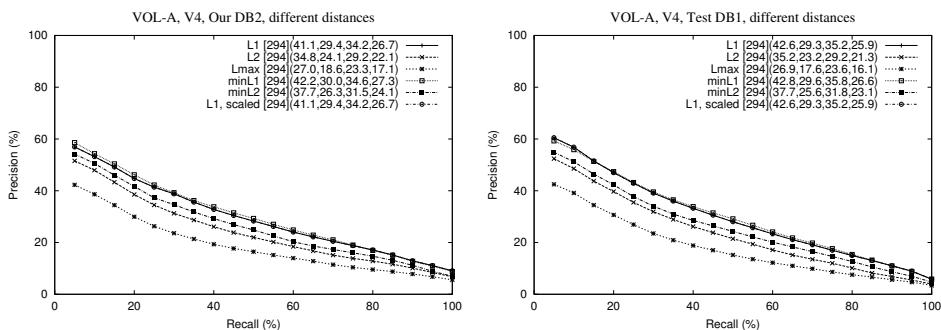


Figure 5.17: Average precision/recall diagrams of our reclassified model collection and the test database, obtained by applying different distance measures (table 5.5) to the variant V4 of the volume-based feature vector in the spatial domain.

In order to explore the influence of the distance calculation on the retrieval performance of the variant V4 of the volume-based descriptor, we present the precision-recall diagrams of our reclassified collection and the test database (figure 5.17). The obtained results suggest that the d_1^{min} (1.21) is slightly more effective than the l_1 norm (1.10). Note that the L1 and “L1 scaled” give exactly the same rankings, because the sum of components of the volume-based feature vector is equal to 1, i.e., the l_1 norm is already constant. The d_2^{min} is more effective than the l_2 distance, but both of them are inferior to the l_1 norm. As expected, the results show that the use of the l_{max} distance should be avoided. The results shown in figure 5.17 are obtained using classifications O2 and TS. Results for the classifications O1 and TR (table 5.3) are similar.

Next, we want to examine if we gain in retrieval effectiveness by representing the volume-based feature in the spectral domain. In figure 5.18, we display selected precision-recall curves of the classifications O2 and TS (table 5.3), for the V4 vector of dimension 294 in the spatial domain and the same variant of the volume-based feature represented in the spectral domain (see section 4.4). For the feature vector in the spectral domain, we tested dimensions 186, 258, 342, and 438. When the l_1 norm is applied after the re-scaling (4.81), the volume-based feature vector of dimension 438 in the spatial domain is superior to the competing descriptors. If the l_1 norm is directly used (i.e., without the re-scaling), then the spatial domain representation is better. Hence, we conclude that in the case of the volume-based approach, we gain in retrieval effectiveness by representing the feature in the spectral domain only if the obtained vector are re-scaled before ranking using the l_1 distance. Results for the classifications O1, TR, and TS support this conclusion.

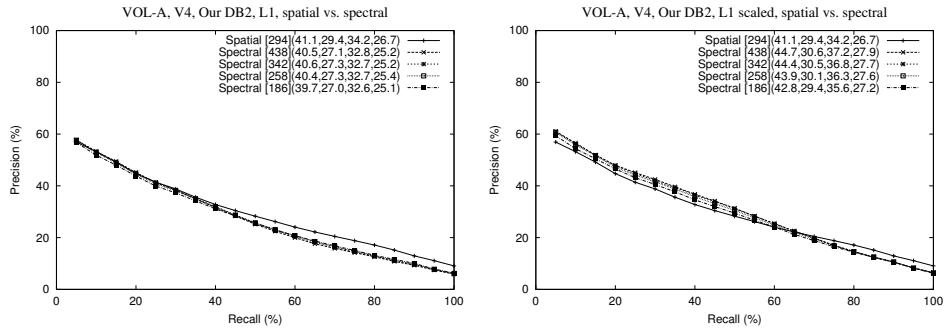


Figure 5.18: Average precision/recall diagrams of our reclassified model collection, for the variant V4 of the volume-based feature vector in both spatial and spectral domains. The l_1 norm is applied directly as well as after the re-scaling (table 5.5).

As stated in section 4.4, we created two algorithms for generating volume-based descriptors, the analytical and the approximative. The approximative approach (see algorithm in figure 4.18), is tested for different values of the param-

eter p_{min} (4.38), which specifies the fineness of approximation. We took $p_{min} \in \{32000, 64000, 128000, 256000\}$. In figure 5.19, all displayed precision-recall curves, which are obtained for our reclassified collection and the training database, almost coincide. Therefore, even for $p_{min} = 32000$, the approximative algorithm generates descriptors whose performance is equal to the performance of descriptors extracted using the analytical approach, which is more time consuming (see table 4.4). We suggest to set $p_{min} = 64000$.

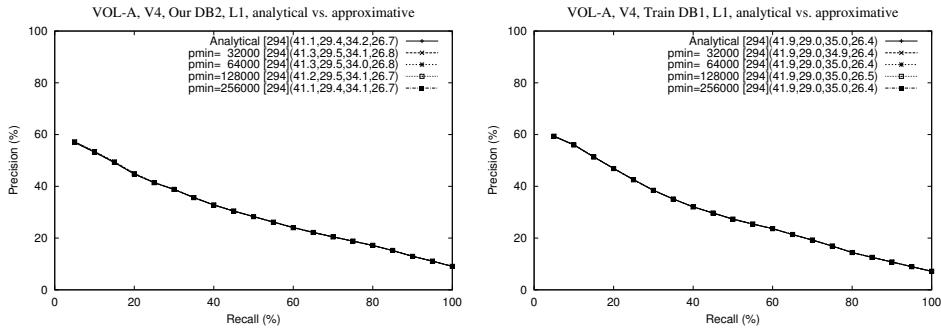


Figure 5.19: Average precision/recall diagrams of our reclassified model collection and the training database, for the variant V4 of the volume-based feature vector in the spatial domain, extracted using the analytical (section 4.4) and the approximative algorithms (figure 4.18).

We recommend to use the variant V4 (table 4.5) of the volume-based feature vector of dimension 438 in the spectral domain, to generate descriptors using the approximative algorithm with $p_{min} = 64000$, and to engage the l_1 distance to rank feature vectors that have previously been re-scaled (L1 scaled).

5.2.5 Voxel-Based Approach

We tested all variants as well as all representations of the *voxel-based descriptor* (section 4.5). We recall that the voxel-based feature can be represented in both the spatial and spectral domains. Also, we use the CC (definition 4.4), CBC (definition 4.1), BB (definition 4.2), and EBB (definition 4.3) to voxelize a 3D-model. Firstly, we show a part of experimental results aimed at determining the optimal dimension of the feature vectors in both spatial and spectral domains. The results for four classifications are shown in figure 5.20. The region of voxelization ρ (4.19) is defined to be the CC with $w = 2$. In the spatial domain, the dimension dim depends on the parameter k as $dim = k^3$. The vector of 343 components ($k = 7$) shows the best performance, while the vectors whose dimensions are fixed by an odd value of the parameter k outperform the vectors whose dimensions are fixed by an even value of k . In the spectral domain, the feature vector of dimension 416 (4.52) outperforms the others. The spectral domain representation is based on $128 \times 128 \times 128$ voxel grids.

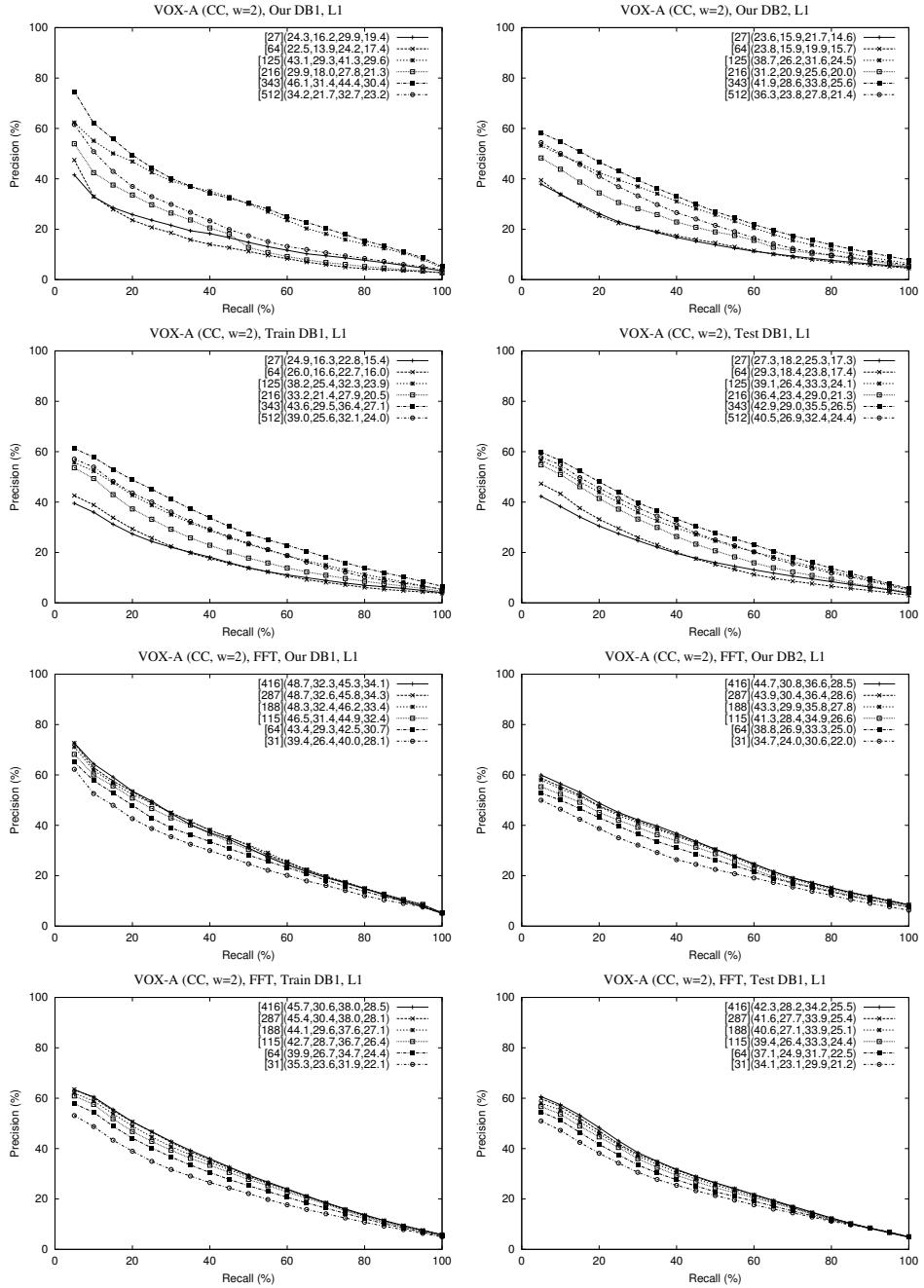


Figure 5.20: Average precision/recall diagrams of four model classifications, for various dimensions of the voxel-based feature vectors in the spatial and spectral (FFT) domains, which are extracted using the CC with $w = 2$ (definition 4.4).

The influence of the choice of the region of voxelization on the retrieval performance is depicted in figure 5.21. We tested the CBC, BB, EBB, as well as the CC with $w = 2$. Our reclassified collection and the test database are used for a series of experiments with feature vectors in the spatial and spectral domains. In the spatial domain, the BB is the best choice of the region of voxelization. Thus, the feature vector that utilizes deformed representations of 3D-models (the aspect ratio is not preserved when using the BB) is better than the feature vector that is robust with respect to outliers (using the CC with $w = 2$). In the spectral domain, the descriptor relying upon the CC with $w = 2$ is the best. We observe that the performance of the best descriptor in the spatial domain is better than the performance of the best descriptor in the spectral domain.

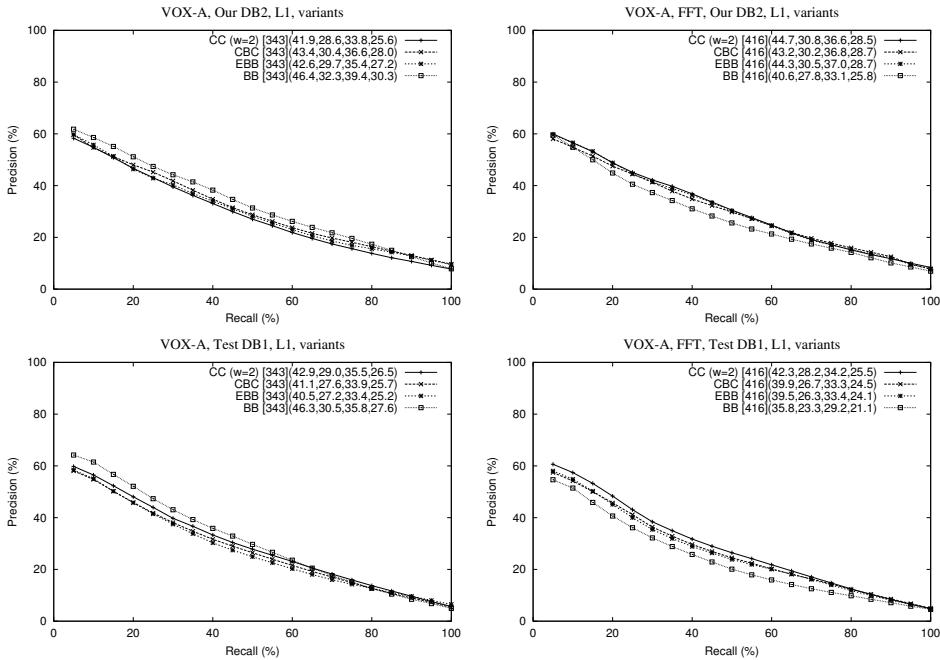


Figure 5.21: Average precision/recall diagrams of our reclassified collection and the test database, for the voxel-based feature vectors extracted using various choices of the region of voxelization, in the spatial and spectral (FFT) domains.

Retrieval performance depends on the applied dissimilarity measure (table 5.5) for ranking descriptors of 3D-objects. In figure 5.22, results for the classifications O2 and TS (table 5.3) are displayed. The best feature vectors in the spatial (BB, $dim = 343$) and spectral domains (CC with $w = 2$, $dim = 416$) are used for the experiments. In the spatial domain, the l_1 distance slightly outperforms the d_1^{min} ,

which is clearly better than the d_2^{min} . Since the l_1 norm of the voxel-based feature vector in the spatial domain, which is extracted relying upon the BB, is equal to 1, the l_1 distance gives always the same ranking as the “L1, scaled”. The l_2 norm shows significantly lower retrieval performance than the d_2^{min} . The l_{max} is absolutely not suitable distance metric. In the spectral domain, the d_2^{min} is slightly better than the d_1^{min} , “L1, scaled”, and l_2 . The l_1 norm outperforms only the l_{max} distance.

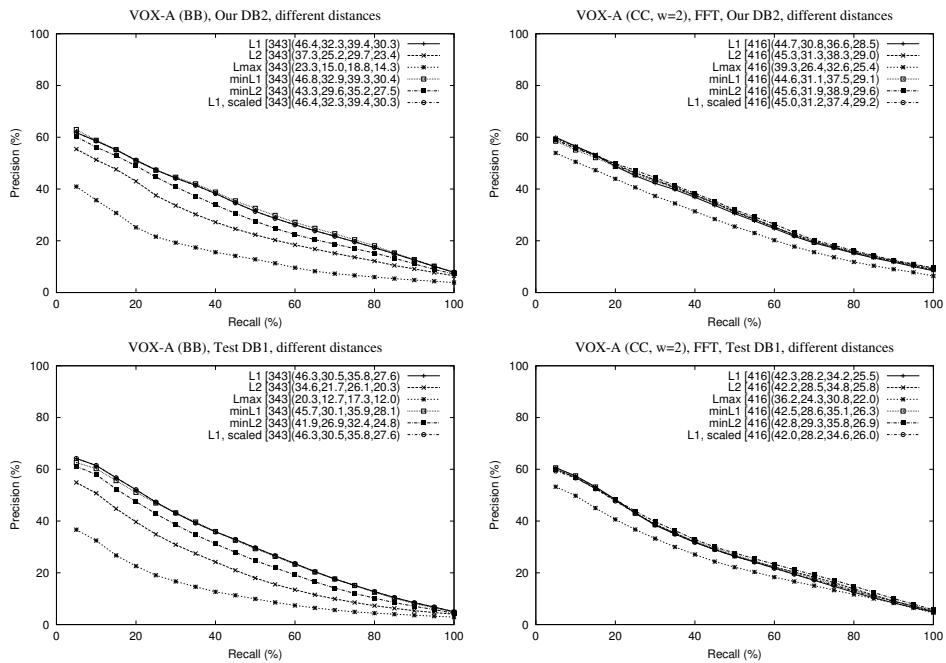


Figure 5.22: Average precision/recall diagrams of our reclassified collection and the test database, applying different distance metrics (table 5.5) to the voxel-based feature vectors in the spatial (BB) and spectral (CC with $w = 2$) domains.

The results shown in figure 5.23 demonstrate that the best choice of the parameter w , which fixes the dimensions of the CC (definition 4.4), is $w = 2$. Also, the scaling factor (3.25) is the best choice. These conclusions are inferred from the precision-recall diagrams on the left-hand side in figure 5.23. We compare descriptors extracted after normalizing a 3D-model by (3.25) and using the CC with $w \in \{2, 4, 8\}$. We observe that the retrieval performance deteriorates with the increase of w . Also, we compare descriptors extracted after normalizing the scale of a 3D-object by different methods (table 5.4) and using the CC with $w = 2$. As expected, the average distance (3.25) outperforms other options. On the right-hand side in figure 5.23, we examine the influence of the resolution of voxelization (di-

mensions of the voxel grids) on retrieval effectiveness. We test voxel grids of types $32 \times 32 \times 32$, $64 \times 64 \times 64$, and $128 \times 128 \times 128$, i.e., the octrees of depths 5, 6, and 7. The largest voxel grid shows the best performance

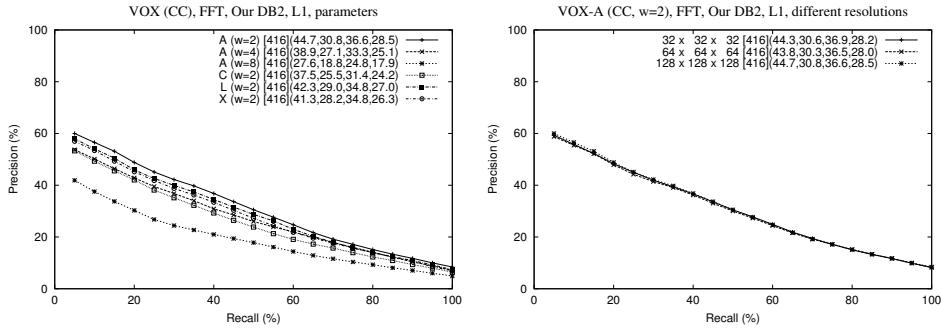


Figure 5.23: Average precision/recall diagrams of our reclassified collection, for the voxel-based feature vectors in the spectral domain. On the left-hand side, the CC is tested for $w \in \{2, 4, 8\}$ and for different scaling factors (table 5.4). On the right-hand side, different resolutions (dimensions) of voxel grids are used.

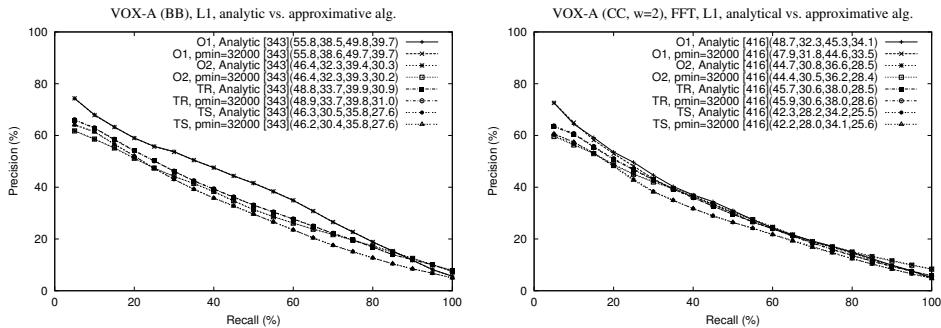


Figure 5.24: Average precision/recall diagrams of four model classifications, for the voxel-based feature vectors in the spatial (BB) and spectral domains (CC with $w = 2$). The feature vectors are extracted using the analytical algorithm (section 4.5) and the approximative algorithm (figure 4.21) with $p_{min} = 32000$.

The algorithms for generating voxel grids, the analytical (section 4.5) and the approximative (figure 4.21), are compared in figure 5.24. We recall that the analytical algorithm exactly computes the distribution of the point set I (1.5) of a 3D-mesh model across the region of voxelization ρ (4.19), while the approximative method uses the parameter p_{min} to fix the fineness of approximation. The results show that for the classifications O1, O2, TR, and TS (table 5.3), the descriptors

extracted using the approximative algorithm with $p_{min} = 32000$ perform almost identical as when the analytical method is used, in both spatial (BB, $dim = 343$) and spectral domains (CC with $w = 2$, $dim = 416$). We recall (see table 4.6) that the average generation time of the feature vector of dimension 343 in the spatial domain amounts 12.5ms, when the approximative algorithm with $p_{min} = 32000$ is used, and 30ms, when attributes of voxel grids are analytically computed. In the spectral domain, we use octrees of depth 7 ($128 \times 128 \times 128$ voxel grid), and apply the 3D-DFT. The approximative algorithm with $p_{min} = 32000$ generates a descriptor in 538ms, on average, while the analytical approach takes 3160ms to produce a feature vector. We stress that results for the training and test databases (table 5.3) comply with the presented results obtained for our reclassified collection.

The most effective variant of the voxel-based approach is the feature vector of dimension 343 in the spatial domain, which is extracted relying upon the BB. If the robustness with respect to outliers is a requirement that should be fulfilled, then we recommend the spectral representation of the voxel-based feature of dimension 416, which is obtained by applying the 3D-DFT to a $128 \times 128 \times 128$ voxel grid and using the CC with $w = 2$ as the region of voxelization. Before forming the voxel grid, each model should be normalized (section 3.4). We recommend the average distance (3.25) as the scaling factor. In order to gain in efficiency, we recommend to use the approximative algorithm (figure 4.21) with $p_{min} = 32000$. The recommended distance metrics are the l_1 (1.10) in the spatial domain and the d_2^{min} (1.20) in the spectral domain.

5.2.6 Ray-Based Approach with Spherical Harmonic Representation

The spectral representation of the ray-based feature, which we regard as the *ray-based feature vector with spherical harmonic representation* (section 4.6.2), is evaluated in this subsection. We recall that the descriptor is formed using the first k rows of Fourier coefficients (algorithm 4.1), obtaining the feature vector of dimension $dim = k(k + 1)/2$ (4.66). In order to determine a good choice of dimension, we generated descriptors for $k = 29$ ($dim = 435$) embedding all feature vectors whose dimensions are fixed by $k < 29$. Our numerous test show that the ray-based feature vectors with the spherical harmonic representation whose dimensions are fixed by $13 \leq k \leq 19$ have similar retrieval effectiveness, while the vectors obtained for $k < 13$ or $k > 19$ have lower performance. To demonstrate our results, we selected precision-recall curves of four classifications, O1, O2, TR, and TS (table 5.3), in figure 5.25. We notice that the vectors of dimensions 91, 136, 171, and 190 perform very similar, while the vectors of dimensions 253 and 435 show that the performance decreases with the increase of dimension. Since it is difficult to select the best choice of vector dimension, we consider that any of the dimensions 91 ($k = 13$), 105, 120, 136, 153, 171, and 190 ($k = 19$) is acceptable. In what follows, we show results of the ray-based descriptor in the spatial domain of dimension 136.

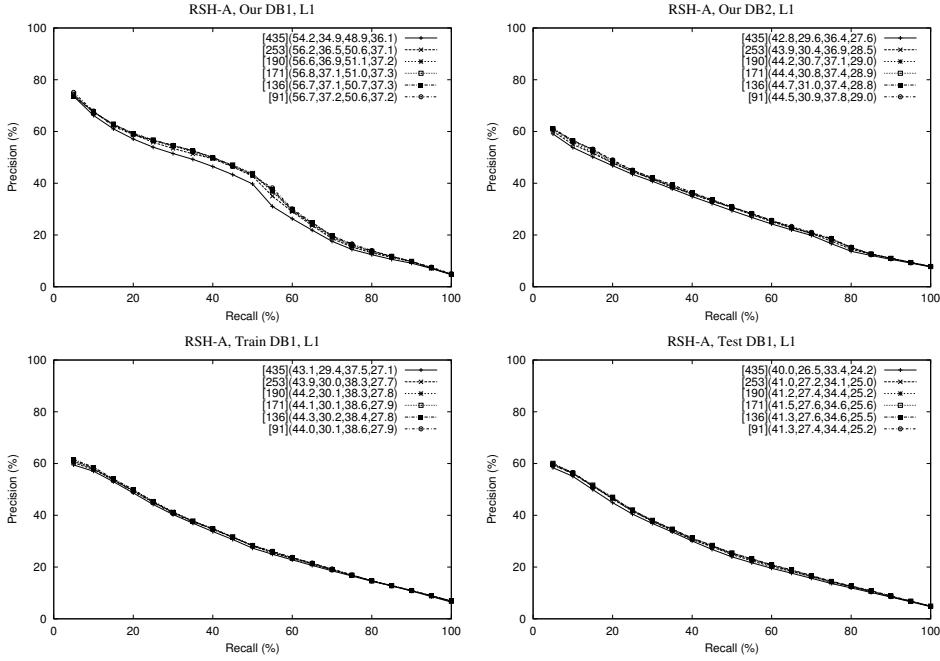


Figure 5.25: Average precision/recall diagrams of four model classifications, for various dimensions of the ray-based feature vector with spherical harmonic representation.

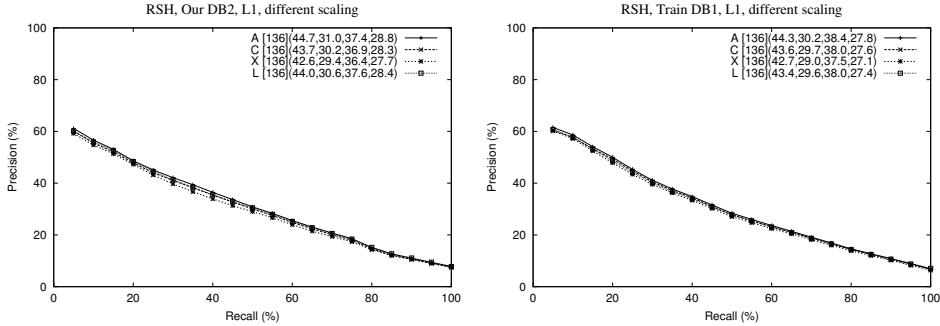


Figure 5.26: Average precision/recall diagrams of our reclassified collection and the training database, when the ray-based feature vectors with spherical harmonic representation are extracted using different scaling factors (table 5.4).

Next, we examine the impact of the scaling factor (table 5.4) on retrieval performance of the ray-based descriptor with spherical harmonic representation. In figure 5.26, the precision-recall diagrams of classifications O2 and TR are depicted. We observe that the average distance (3.25) is the best choice of scaling factor.

We recall that the more samples (4.63) of the extent function r (4.1) are taken the better the approximation of the underlying 3D-object (see figure 4.26). The samples serve as the input for the Fourier transform on the sphere (section 4.6.1). Therefore, we want to test the influence of the number of samples on the performance of the ray-based descriptor in the spectral domain. In figure 5.27, we display results obtained for our reclassified collection and the test database. The tested numbers of samples are 64^2 ($B = 32$), 128^2 , 256^2 , and 512^2 . Having in mind the average feature extraction times (table 4.8) and the given precision-recall diagrams, we suggest to use $B = 64$, i.e., to have 128^2 sample values of the function r .

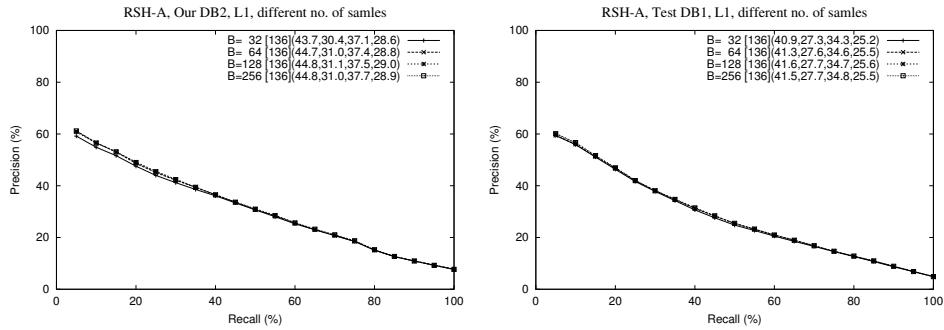


Figure 5.27: Average precision/recall diagrams of our reclassified collection and the test database, when the ray-based feature vectors with spherical harmonic representation are extracted using different number of samples, $4B^2$.

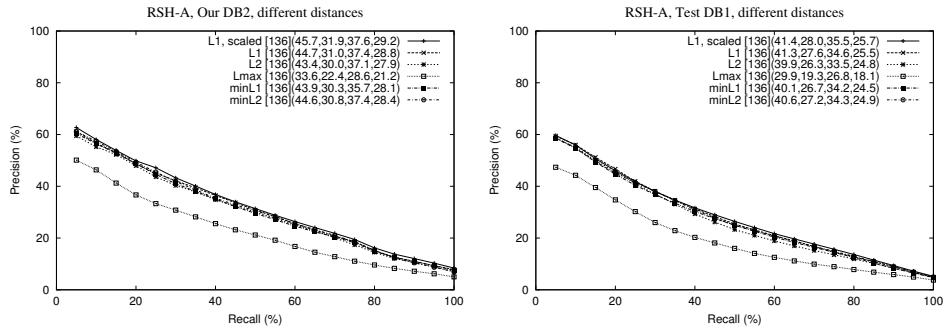


Figure 5.28: Average precision/recall diagrams of our reclassified collection and the test database, obtained by applying different distance metrics (table 5.5) to the ray-based feature vector with spherical harmonic representation.

The influence of the used distance metric (table 5.5) on the performance of the ray-based descriptors in the spectral domain is illustrated in figure 5.28. We observe that the best performance is obtained by applying the l_1 distance after

the re-scaling so that the l_1 norm of each feature vector is constant (4.81). The differences in precision-recall values obtained applying the l_1 , d_2^{min} , d_1^{min} , and l_2 are not significant. The only exception is the l_{min} distance, which shows the worst performance. Hence, we suggest to use the “L1, scaled” technique, whence it is not necessary to fix the scale invariance in the normalization step (section 3.4), because each feature vector is re-scaled. A similar example is shown in figure 5.8.

We recommend to sample the extent function r (4.1) at 128^2 points defined by (4.63), before applying the Fourier transform on the sphere (section 4.6.1). Acceptable dimensions of the ray-based feature vectors with spherical harmonic representation are 91, 105, 120, 136, 153, 171, and 190. We recommend to normalize the l_1 length of each feature vector and to use the l_1 (1.10) distance for ranking.

5.2.7 Moments-Based Feature Vector

The *moments-based descriptor* (section 4.6.3) is also evaluated for different dimensionality, scale factor, number of samples, and distance calculations. We recall that the dimension of the feature vector based on moments defined by (4.68) is fixed by a parameter k according to (4.70). We tested the following vector dimensions 9, 19, 34, 55, 83, 119, 164, 219, 285, and 363 (i.e., $k = 2, \dots, 11$).

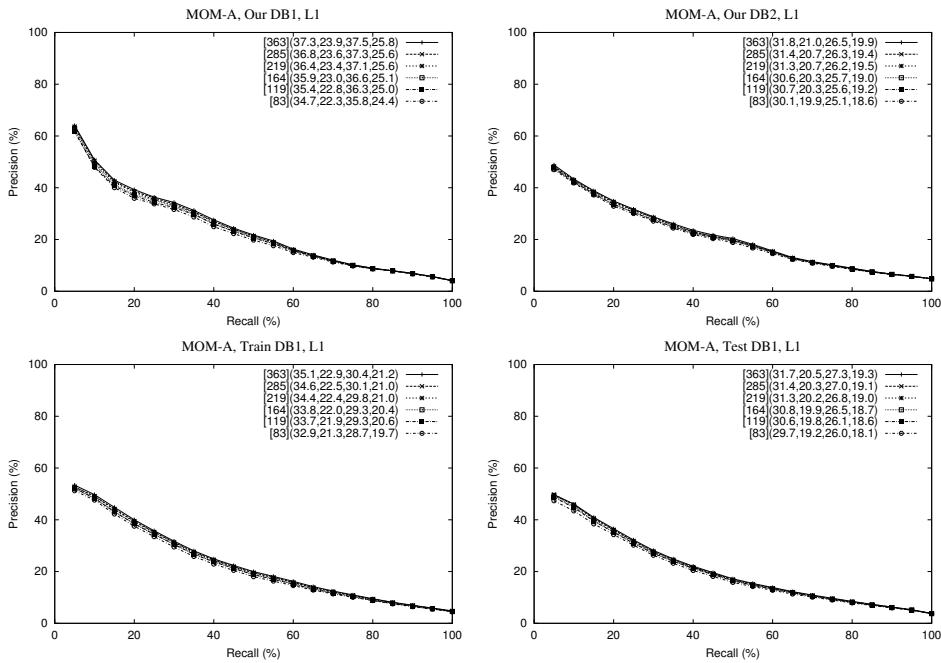


Figure 5.29: Average precision/recall diagrams of four model classifications, for various dimensions of the moments-based feature vector.

The precision-recall curves in figure 5.29, which are selected from all generated diagrams, demonstrate that $\dim = 363$ ($k = 11$) is the best choice of dimension.

The influence of the four scaling factors (table 5.4), which are presented in section 3.4, on the retrieval performance of the moments-based descriptor is depicted in figure 5.30. As it is the case with all approaches tested in this section, the shown precision-recall diagrams as well as the other evaluation parameters, \bar{p}_{100} , \bar{p}_{50} , BEP , and RP (section 1.5), unambiguously show that the average distance (3.25) is the best choice of the scaling factor. The results obtained for our original classification as well as for the test database lead to the same conclusion.

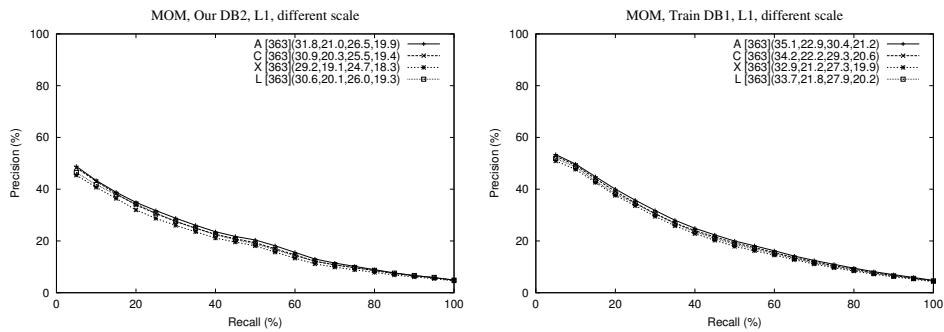


Figure 5.30: Average precision/recall diagrams of our reclassified collection and the training database, when the moments-based feature vectors are extracted using different scaling factors (table 5.4).

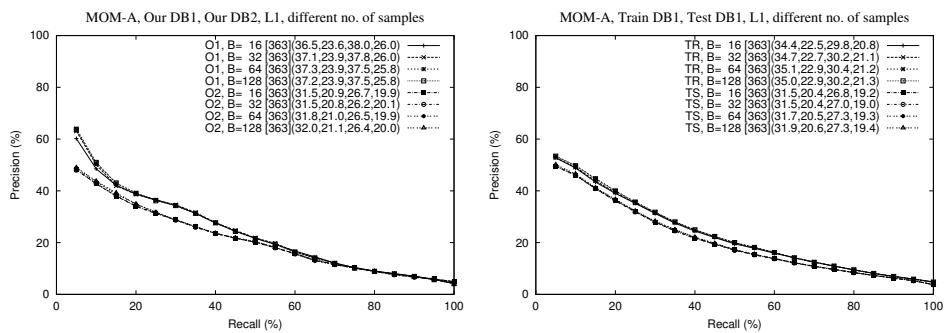


Figure 5.31: Average precision/recall diagrams of four classifications, when the moments-based feature vectors are extracted using different number of samples, $4B^2$.

The influence of the number of samples, $4B^2$, on retrieval effectiveness of the feature vector defined using moments (4.68) is illustrated in figure 5.31. Similarly as it is the case with the ray-based descriptor in the spatial domain (figure 5.27),

we observe that $B = 64$ is the best choice of the number of samples. The conclusion is based on both the shown performance and the average extraction times given in table 4.9.

The effectiveness of distance measures (table 5.5) used for ranking moments-based feature vectors is depicted in figure 5.32. The l_1 distance applied to normalized (re-scaled) feature vectors is slightly more effective than the d_1^{min} . The minimized l_1 distance d_1^{min} outperforms other dissimilarity measures. The experiments on our original classification and on the training database also confirm that the d_1^{min} is the best choice of distance calculation. Having in mind the definition of moments (4.68), if the “L1, scaled” technique is applied, there is no need for fixing the scale of a 3D-mo deli during the normalization procedure (section 3.4). A similar example is shown in figure 5.8.

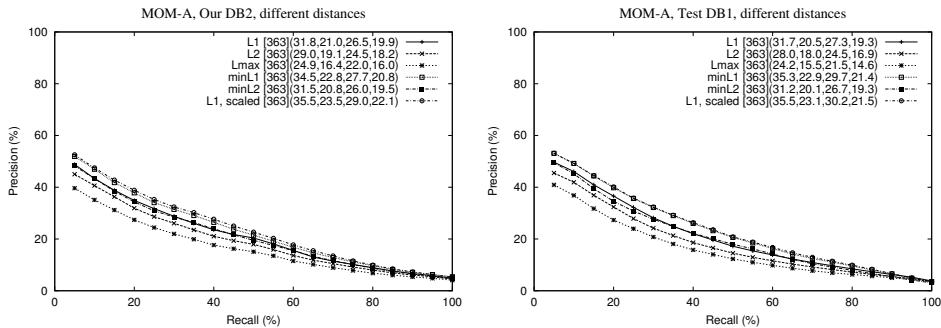


Figure 5.32: Average precision/recall diagrams of our reclassified collection and the test database, obtained by applying different distance metrics (table 5.5) to the moments-based feature vector.

Thus, we recommend to use the following settings for the moments-based descriptor: number of samples 128^2 ($B = 64$), dimension $dim = 363$, and to normalize the l_1 length of feature vector before applying the l_1 distance (L1 scaled).

5.2.8 Shading-Based Feature Vector

In this subsection, we evaluate the retrieval performance of the *shading-based descriptor* (section 4.6.4). We recall that the used feature is a rendered perspective projection of a 3D-object on an enclosing sphere (4.71). The vector is represented in the spectral domain, using spherical harmonics (algorithm 4.1), with the approach (4.66). We recall that the shading-based descriptor is invariant with respect to scaling by the definition. However, it is sensitive with respect to different tessellations and levels-of-detail of a mesh model. The dimension of the vector depends on a parameter k as $dim = k(k+1)/2$ (4.66). We tested all dimensions for $k = 6, \dots, 29$. An excerpt from our results is shown in figure 5.33. We declare $dim = 91$ ($k = 13$) as the optimal choice of dimension for the shading-based method.

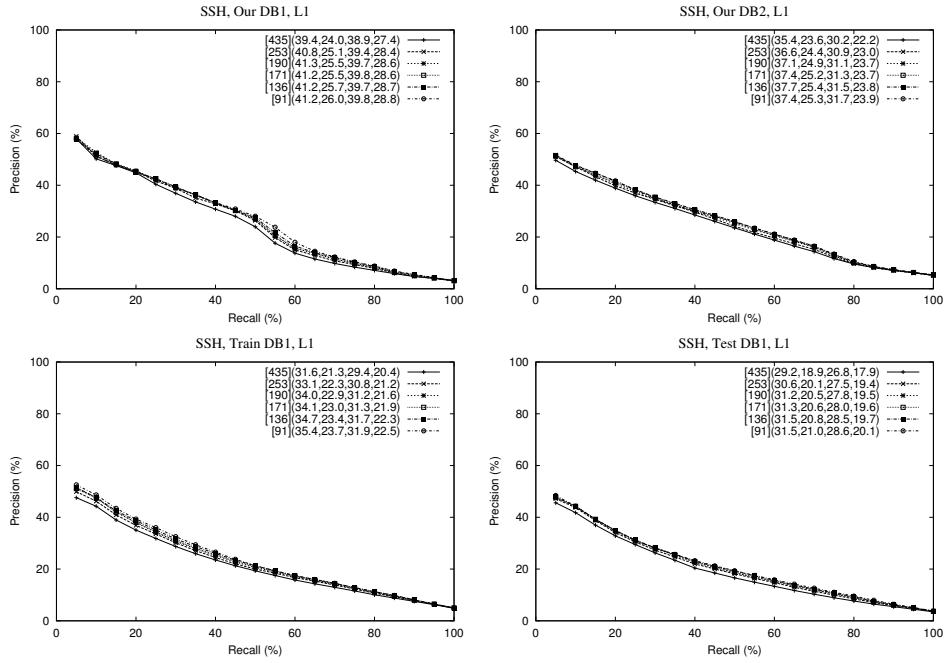


Figure 5.33: Average precision/recall diagrams of four model classifications, for various dimensions of the shading-based feature vector.

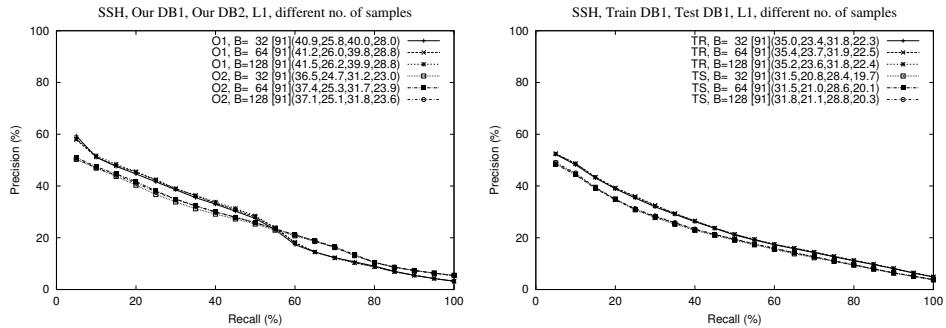


Figure 5.34: Average precision/recall diagrams of four classifications, when the shading-based feature vectors are extracted using different number of samples, $4B^2$.

In figure 5.34, we examine the dependence of retrieval performance on the number of samples ($4B^2$) of the function $s(\mathbf{u})$ defined by (4.71). There is not a significant difference between the shown precision-recall diagrams. Nevertheless, we consider $B = 64$ as the best choice.

The l_2 norm of the difference between two shading-based feature vectors outperforms all tested dissimilarity measures. The conclusion is based on results from a set of experiments on all available model classifications, applying different metrics (table 5.5) to the shading-based feature vector. Precision/recall diagrams of our reclassified collection and the test database, obtained by applying different dissimilarity measures to the shading-based descriptor, are illustrated in figure 5.35.

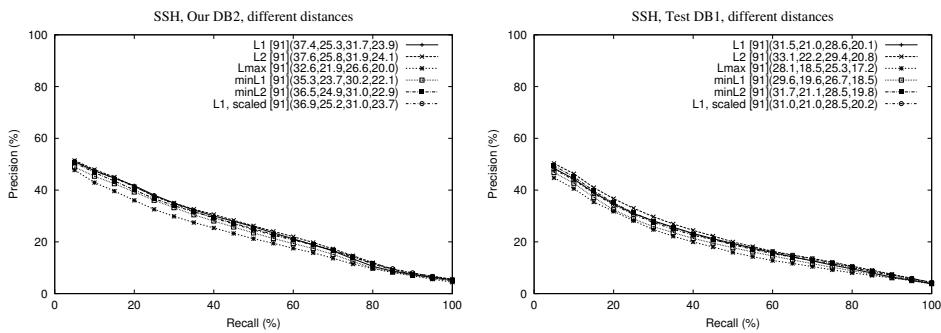


Figure 5.35: Average precision/recall diagrams of our reclassified collection and the test database, obtained by applying different distance metrics (table 5.5) to the shading-based feature vector.

We recommend the following settings for the shading-based descriptor: the number of samples 128^2 ($B = 64$), the dimension $\dim = 91$, and the l_2 distance for ranking the feature vectors.

5.2.9 Complex Feature Vector

The *complex feature vector* (section 4.6.5) is defined using the function $z_\alpha(\mathbf{u})$ given by (4.72). We recall that the extent function r (4.1) multiplied by α^{-1} forms the real part, while the shading function s (4.71) represents the imaginary part of the function z_α . The factor α balances the ‘importance’ of functions r and s . Thus, besides performing experiments in order to determine the best dimension, scaling factor, and distance metric, we also have a task to suggest an acceptable value of the parameter α . Firstly, we set $\alpha = 1$, i.e., the functions r and s are equally important, and try to find the best choice of vector dimension. In figure 5.36, precision-recall curves of four classifications are displayed. We observe that it is not possible to state that one value of vector dimension is absolutely better than all others. Therefore, we suggest $\dim = 196$ as an acceptable choice of vector dimension.

Next, we test different values of the weighting parameter α , in order to examine its influence on retrieval effectiveness of the complex descriptor. For large values of α , the shading-based descriptor is dominant, while the ray-based feature prevails when $\alpha \rightarrow 0$. We tested $\alpha \in \{0.1, 0.2, 0.3, \dots, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0\}$.

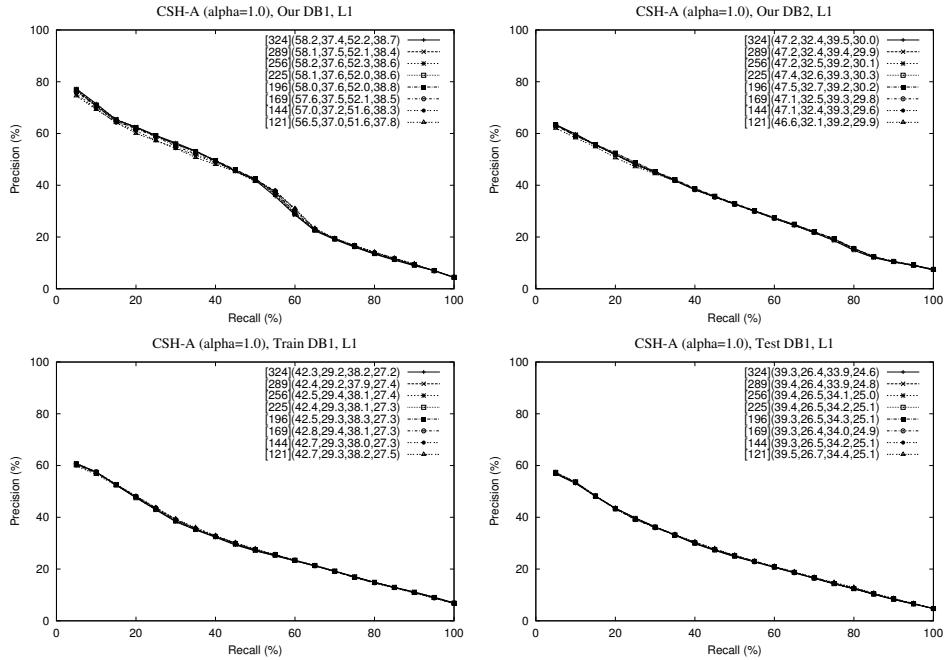


Figure 5.36: Average precision/recall diagrams of four model classifications, for various dimensions of the complex feature vector with $\alpha = 1.0$.

Precision-recall diagrams of four model classifications, obtained using complex descriptors extracted for various values of the parameter α , are displayed in figure 5.37. In all four diagrams, we observe that retrieval performance deteriorates for $\alpha > 1$. We notice that for the classifications O1 and O2 (table 5.3) the optimal value of α is 0.6, while $\alpha = 0.4$ is the best choice of the parameter value for the classifications TR and TS. Hence, we consider both values to be acceptable. In what follows, we show results obtained for $\alpha = 0.4$. By comparing results given in figures 5.25, 5.33, and 5.37, we observe that the ray-based descriptor in the spectral domain is superior to the shading-based descriptor. Also, for $\alpha \in \{0.4, 0.6\}$, the complex feature vector slightly outperforms the ray-based descriptor for all four classifications. Thus, the shading-based approach serves to refine the ray-based feature. An acceptable choice of the value of α is expected to be less than 1, because the better method (ray-based) should be rated as more important than the inferior approach (shading-based).

Different approaches for fixing the scale of a 3D-mesh model are tested, as well. The retrieval performance of the complex feature vector varies depending on the used scaling factor. The sensitivity to scaling is inherited from the ray-based approach, while the shading-based feature is invariant to scaling by definition.

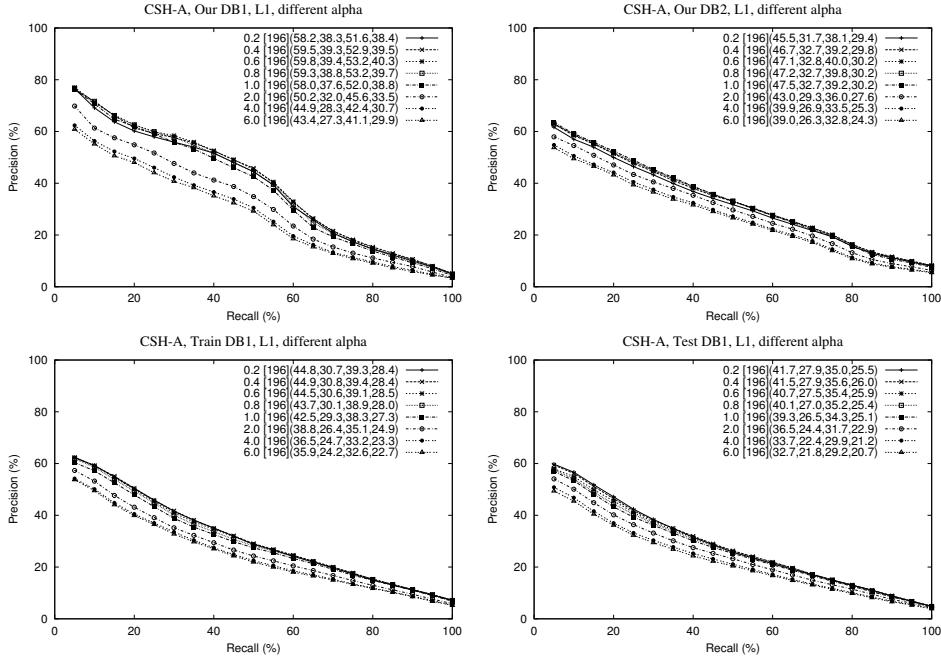


Figure 5.37: Average precision/recall diagrams of four model classifications, when the complex feature vectors are extracted using different values of the parameter α .

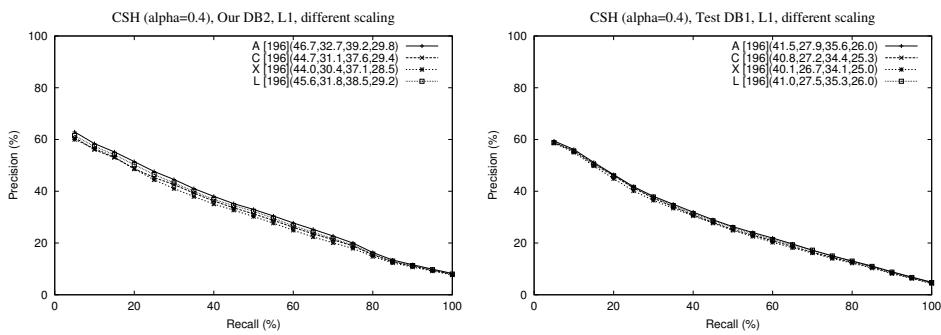


Figure 5.38: Average precision/recall diagrams of our reclassified collection and the test database, when the complex feature vectors are extracted using different scaling factors (table 5.4).

The results presented in figure 5.38 suggest that the scaling factor defined as the average distance (3.25) is the best choice.

The influence of dissimilarity measure on retrieval effectiveness of the complex descriptor is depicted by precision-recall diagrams in figure 5.39. The l_1 norm slightly outperforms the “L1, scaled” approach. Evidently, other dissimilarity measures are inferior to the top two.

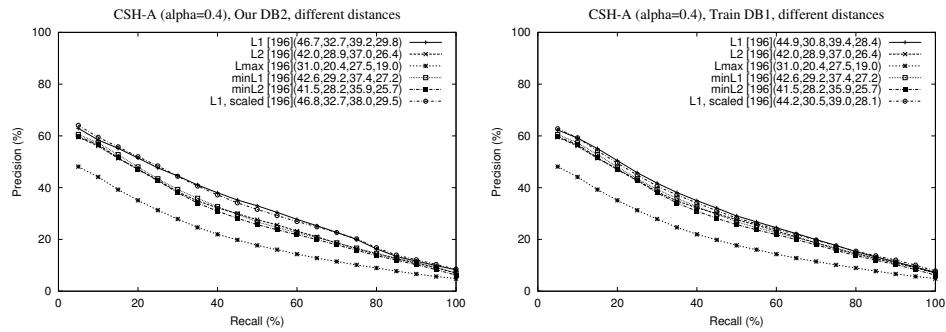


Figure 5.39: Average precision/recall diagrams of our reclassified collection and the training database, obtained by applying different distance metrics (table 5.5) to the complex feature vector.

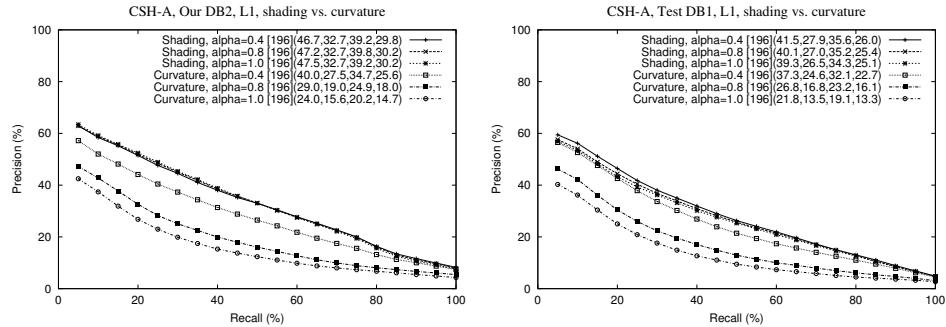


Figure 5.40: Average precision/recall diagrams of our reclassified collection and the test database, obtained by using the feature vector that combines the extent function and shading (4.72) and the feature vector that combines the extent function and curvature (4.74).

We also tested the complex feature vector defined using the function z'_α (4.74), which combines the extent function r (4.1) and the function c (4.73) based on the curvature index (2.8). A summary of obtained results is presented in figure 5.40.

The impact of the shading-based feature on retrieval performance of the resulting complex descriptor is compared to the impact of the curvature-based feature. The results show that the shading function s (4.71) is significantly better choice of the imaginary part than the curvature-based function c . Having in mind that the curvature index (2.8) heavily rely upon the orientation of triangles of the mesh and is sensitive to different tessellations and levels-of-detail of the model, the results in figure 5.40 are expected.

We recommend to use the complex feature vector of dimension 196, which is extracted after normalizing a model using the CPCPA approach (section 3.4), where the scale is fixed by the average distance (3.25). The recommended values of the parameter α are 0.4 and 0.6 (4.72). The l_1 is the recommended distance metric for ranking the feature vectors. We also suggest to sample the function z (4.72) at 128^2 points ($B = 64$). The results demonstrating the influence of the number of samples on retrieval efficiency are omitted to save the space.

5.2.10 Feature Vectors Based on Layered Depth Spheres

Two descriptors based on layered depth spheres (LDSs), which are introduced in section 4.6.6, describe the same feature in different manners. Both descriptors are formed using R functions on concentric spheres (4.75), which are sampled using LDS structures. The *feature vector based on LDS* is formed by (4.78), relying upon the algorithm 4.1. The same feature can be represented using the property of spherical harmonics (property 4.1) to secure rotation invariance (see remark 4.1) without applying the PCA in the normalization step (chapter 3). The descriptor formed by (4.79) is regarded as the *rotation invariant feature vector based on LDS* (RID). We recall that dimensions of both feature vectors are fixed by two parameters, the number R of functions on concentric spheres and the number L of used bands of spherical harmonics. The dimension of the LDS descriptor is $\dim = R \cdot L(L+1)/2$, while the dimension of the RID feature vector is $\dim = R \cdot L$. Also, the radius M of the largest sphere is a parameter affecting retrieval performance of descriptors. Note that by comparing LDS vs. RID descriptors, we compare different representations of the same feature, but also different approaches for achieving rotation invariance, the CPCPA (section 3.4) vs. the property of spherical harmonics (remark 4.1).

The influence of dimension of the LDS feature vector on retrieval efficiency is examined by testing various vector dimensions for different settings for the parameters R and M . In figure 5.41, precision-recall diagrams of four model classifications are displayed, for various dimensions of the feature vector based on LDSs, with $R = 8$, and $M = 4$. For the given settings, we observe that the retrieval performance of the feature vectors of dimensions 224, 288, 360, 440, and 528 is very similar. After studying the trade-off between the retrieval effectiveness and efficiency, we suggest $\dim = 360$ as the best choice of dimension of the LDS feature vector for the given parameter settings. Thus, the feature vector should be formed using the first 9 bands of spherical harmonics ($\dim = 360 \wedge R = 8 \implies L = 9$). In what follows, we try to determine the optimal values of the parameters M and R .

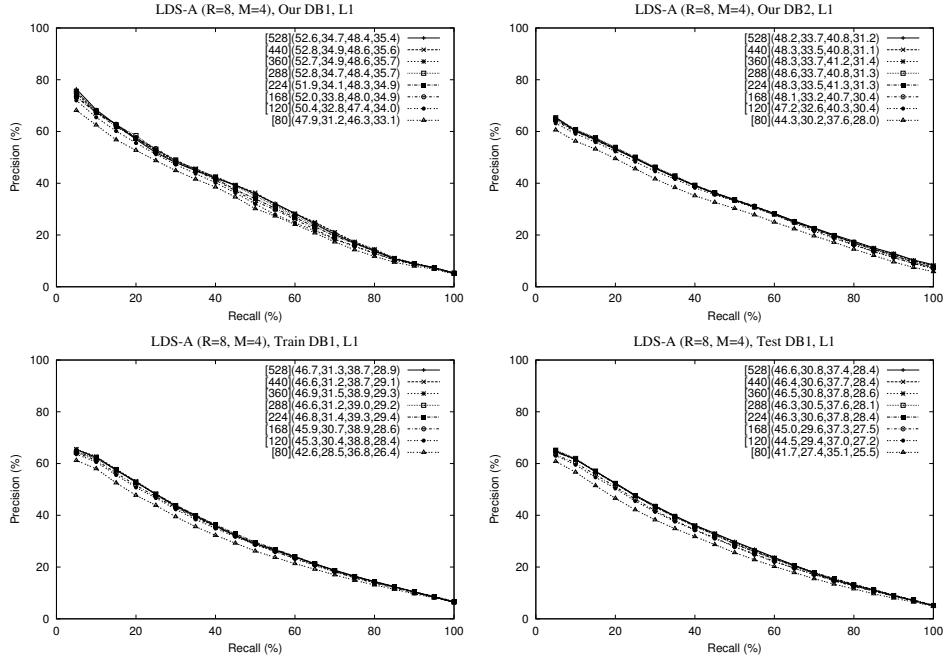


Figure 5.41: Average precision/recall diagrams of four model classifications, for various dimensions of the feature vector based on LDSs, with $R = 8$ and $M = 4$.

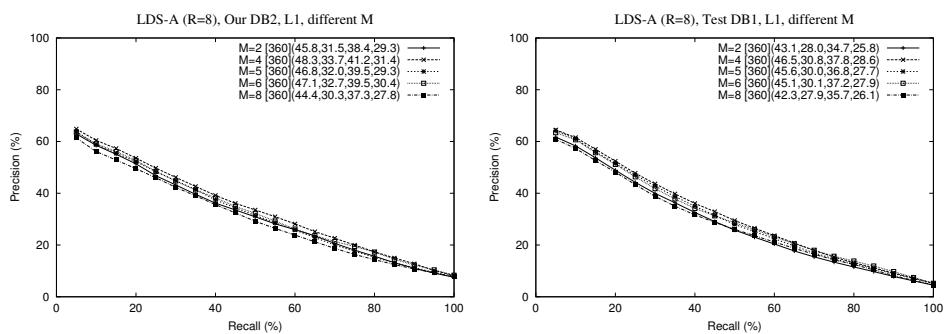


Figure 5.42: Average precision/recall diagrams of our reclassified collection and the test database, when LDS feature vectors are extracted using different values of the parameter M (radius of the largest sphere), with $R = 8$.

Effectiveness of LDS descriptors, which are extracted using different values of the largest radius M , for the fixed value of the number of functions on a sphere ($R = 8$), is depicted in figure 5.42. The tested values of the parameter M are 2, 4, 5, 6, and 8. The precision-recall curves obtained for $M = 4$ lay above all the other curves. Thus, we declare $M = 4$ as an acceptable choice of the parameter value. The results shown in figure 5.42 are obtained using our reclassified collection and the test database. The experiments on our original collection and the training database suggest that $M = 6$ is also an acceptable choice of the parameter value.

Next, the influence of the number of functions on a sphere on the retrieval performance of the feature vector based on LDSs is demonstrated in figure 5.43. The value of the largest radius is fixed, $M = 4$. In separate tests, we determined the best dimensions of feature vectors extracted using 4, 12, and 16 functions on a sphere. The best precision-recall curves for $R = 4$, $R = 8$, $R = 12$, and $R = 16$ are displayed. By comparing the diagrams, we conclude that the curve obtained for $R = 8$ is superior to the others. Our reclassified collection and the training database are used for deriving the conclusion, which is also supported by experiments performed on our original classification as well as the test database.

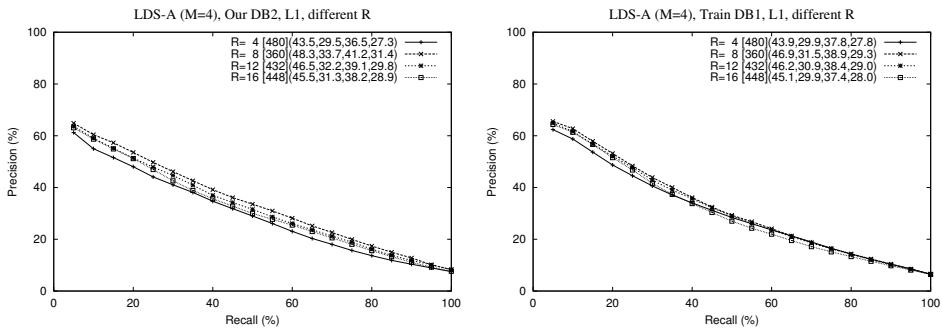


Figure 5.43: Average precision/recall diagrams of our reclassified collection and the training database, when LDS feature vectors are extracted using different values of the parameter R (number of functions on a sphere), with $M = 4$.

Thus, we state that the acceptable parameter settings for the LDS feature vector are: $R = 8$, $M = 4$, and $\text{dim} = 360$ ($L = 9$). Different scale normalization factors (table 5.4) affect the retrieval efficiency. In figure 5.44, we show the precision-recall diagrams of our reclassified collection and the test database, when the LDS feature vectors are extracted using different scaling factors. The results confirm that the average distance (3.25) is the best choice of normalizing the scale of a 3D-mesh model.

We stress that all feature vectors that are analyzed in this subsection are extracted using 128^2 samples of functions on a sphere, because the results shown in figures 5.27, 5.31, and 5.34 suggest that having 128^2 samples is a good choice.

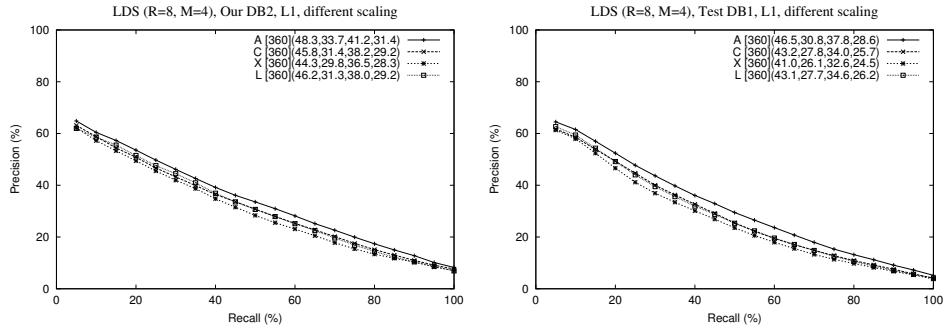


Figure 5.44: Average precision/recall diagrams of our reclassified collection and the test database, when the LDS feature vectors are extracted using different scaling factors (table 5.4), with $R = 8$ and $M = 4$.

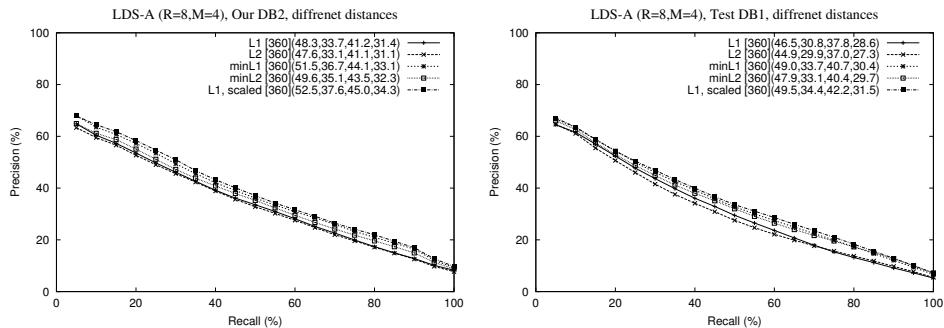


Figure 5.45: Average precision/recall diagrams of our reclassified collection and the test database, obtained by applying different distance calculations (table 5.5) to the LDS feature vector with $R = 8$ and $M = 4$.

In a set of experiments aimed at determining the best choice of distance calculation, we use the dissimilarity measures listed in table 5.5. The results obtained for the collections O1, O2, TR, and TS (table 5.3) are consistent. The precision-recall diagrams in figure 5.45 suggest that the most effective ranking is achieved by scaling each feature vector by the factor ω so that the l_1 norm of each vector is equal to the vector dimension (4.81) followed by application of the l_1 distance (L1, scaled). We notice that the d_1^{min} outperforms the remaining dissimilarity measures from table 5.5. We recall that the minimized l_1 distance d_1^{min} also re-scales feature vectors (see comments related to figure 5.4). Thus, in the case of the LDS feature vector, re-scaling by normalizing the l_1 norm of each feature vector is a better choice than re-scaling by minimizing the l_1 norm. Note that the re-scaling does not make the descriptors invariant to scaling without normalizing the initial scale of the model, because the value of the largest radius M relies upon the canonical scale.

We recommend the following settings for the LDS descriptor: the number of functions on a sphere $R = 8$, the number of samples 128^2 , the largest radius $M = 4$ (or $M = 6$), the vector dimension $dim = 360$, and the average distance (3.25) as the scaling factor in the normalization step. We recommend to scale each feature vector \mathbf{f} by the factor ω (4.81) so that $\|\mathbf{f}\|_1 = dim$, and to apply the l_1 distance for ranking the scaled feature vectors.

We also thoroughly analyzed the rotation invariant feature vector based on LDSs. To demonstrate the obtained results, we first show the precision-recall diagrams of our reclassified collection and the training database, for various dimensions of the rotation invariant feature vector based on LDSs, with $R = 8$ and $M = 6$. By analyzing the results in figure 5.46, we state that $dim = 512$ ($L = 64$) is the best choice of vector dimension, for the given settings. The results for our original 3D-model classification and the test database lead to the same conclusion.

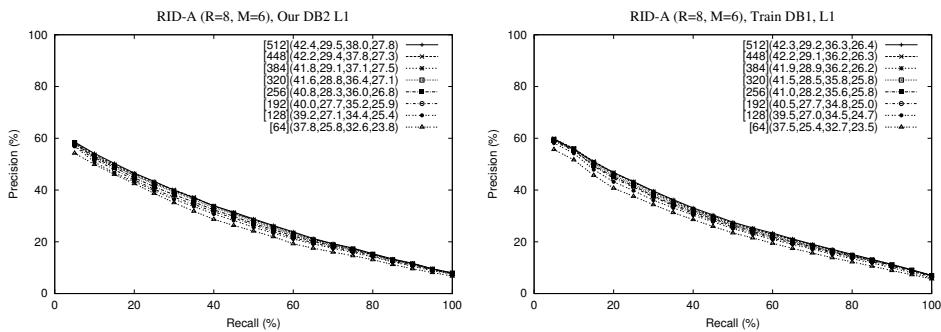


Figure 5.46: Average precision/recall diagrams of our reclassified collection and the training database, for various dimensions of the rotation invariant feature vector based on LDSs, with $R = 8$ and $M = 6$.

In order to show that $R = 8$ and $M = 6$ are acceptable values of the parameters, we display the results in figure 5.47, obtained for our reclassified collection and the training database. The following values of the parameters are tested: $R \in \{4, 8, 12\}$ and $M \in \{2, 4, 6, 8\}$. We notice that the setting $R = 8$ and $M = 6$ is evidently the best for the training database. For our reclassified collection, the setting $R = 8$ and $M = 6$ is the best for recall values greater than 25%, while the setting $R = 8$ and $M = 4$ is the best for recall values less than 25%. Nevertheless, we consider that $R = 8$ and $M = 6$ is the best overall setting. The conclusion is supported by the results obtained for our original classification and the test database. In separate tests, we verified that the average distance (3.25) is the best choice of the scaling factor in the normalization step (section 3.4).

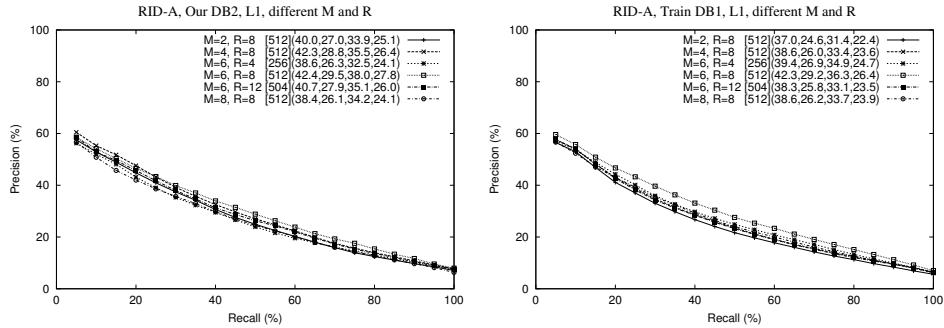


Figure 5.47: Average precision/recall diagrams of our reclassified collection and the training database, when the rotation invariant LDS feature vectors are extracted using different values of the parameters M and R .

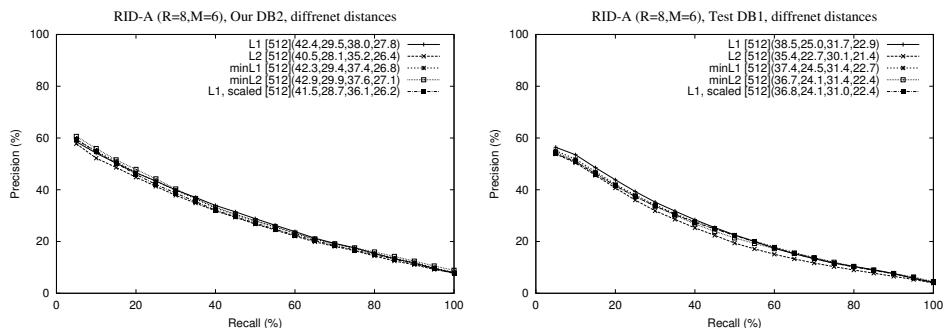


Figure 5.48: Average precision/recall diagrams of our reclassified collection and the test database, obtained by applying different distance calculations (table 5.5) to the rotation invariant LDS feature vector with $R = 8$ and $M = 6$.

We performed tests, which are similar to the ones shown in figure 5.45, in order to study the influence of distance measure on retrieval performance of the rotation invariant LDS descriptor. In figure 5.48, the precision-recall diagrams of our reclassified collection and the test database suggest that the l_1 distance outperforms all competing dissimilarity techniques, including the application of the l_1 norm after re-scaling each feature vector. We recall that the “L1-scaled” technique performs the best in the case of the LDS feature vector.

We recommend the following settings for rotation invariant version of the LDS descriptor: the number of functions on a sphere $R = 8$, the number of samples 128^2 , the largest radius $M = 6$, the vector dimension $dim = 512$, the average distance (3.25) as the scaling factor in the normalization step, and the l_1 norm as distance metric.

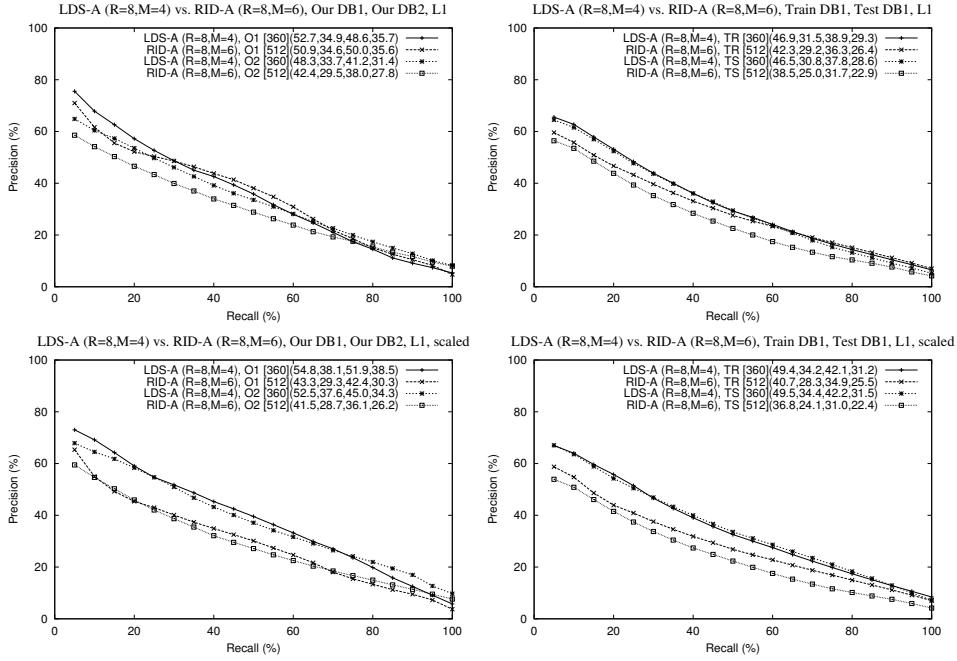


Figure 5.49: Average precision/recall diagrams of four model classifications, obtained by using the best choices of both variants of the descriptors based on LDSs. Used distance metrics: L1 and “L1, scaled” (table 5.5).

Finally, we compare the best variants of two approaches for representing the same feature, the LDS descriptor (LDS) vs. the rotation invariant version (RID). The obtained results for four model classifications are shown in figure 5.49. We compared approaches using the l_1 norm directly and after the re-scaling of feature vectors by ω (4.81). Evidently, for the classifications O2, TR, and TS (table 5.3), the approach (LDS) that uses our complete normalization step (section 3.4) significantly outperforms the approach (RID) based on avoiding the CPCCA and using the property 4.1 of spherical harmonics to secure rotation invariance of the descriptor. Therefore, we conclude that the CPCCA should not be avoided, regardless of certain weaknesses (section 3.5), because the descriptor relying upon the CPCCA possesses higher discriminant power than the descriptor relying upon the competing technique for achieving rotation invariance.

5.2.11 Hybrid Descriptors

The concept of *hybrid descriptors* (section 4.7), relies on crossbreeding of complementary (“orthogonal”) feature vectors. Since descriptors with an embedded multi-resolution representation (1.8) carry the most important information about represented features in the first components of corresponding vectors, we consider

such descriptors as good candidates for crossbreeding. Moreover, a mechanism to keep the dimension of resulting hybrid feature vector inside reasonable limits (e.g., $\dim < 500$) is provided. The goal of crossbreeding is to generate a descriptor that outperforms all parental feature vectors.

In order to evaluate the concept of hybrid feature vectors, we tested crossbreeding of the following four descriptors (candidates):

- (C1) Depth buffer-based descriptor (section 4.3) based on the EBB (definition 4.3), extracted from depth buffer images of dimensions 128×128 ;
- (C2) Silhouette-based descriptor (section 4.2) with equiangular sampling (4.10), extracted from silhouette images of dimensions 256×256 using 256 sample points, where sample values are defined by (4.13);
- (C3) Ray-based descriptor with spherical harmonic representation (section 4.6.2), obtained by sampling the extent function (4.1) at 128^2 points (4.63);
- (C4) Voxel-based descriptor in the spectral domain (section 4.5) based on a $128 \times 128 \times 128$ voxel grid, which is formed using the CC (definition 4.4) with $w = 2$ as the region of voxelization. Each model is scaled using the average distance (3.25) as the scaling factor.

We performed crossbreeding of two, three, and all four candidates. We recall that the average extraction times for the given settings of candidate descriptors C1, C2, C3, and C4 are 85ms, 33ms, 68ms, and 538ms, respectively.

In figure 5.50, we give results for four 3D-model classifications, O1, O2, TR, and TS (see table 5.3). We recall that the abbreviations for types of our descriptors are given in table 4.1. The l_1 norm is the best choice of distance metrics (results for other distance measures are omitted). On the left-hand side, the precision-recall diagrams obtained using hybrids of two descriptors are displayed. We observe that the descriptor denoted by “DBD258*RSH190”, which is obtained by crossbreeding the candidate feature vector C1 of dimension 258 and the candidate feature vector C3 of dimension 190, outperforms other descriptors obtained by crossbreeding two candidates. Thus, the dimension of the obtained hybrid feature vector is equal to 448. On the right-hand side, we compare the hybrid of all four candidates to the hybrids of three candidate descriptors. To illustrate the introduced improvement of the overall retrieval performance, the precision-recall curve of the candidate descriptor C1 of dimension 438 (DBD438) is shown, as well. By comparing precision-recall curves as well as the given evaluation parameters for each classification, we observe that we benefit if we create a hybrid of three or four descriptors. The hybrid of all four descriptors of dimension 475, denoted by “DBD186*SIL120*RSH105*VOX064”, and the hybrid feature vector of dimension 472, denoted by “DBD186*SIL150*RSH136”, possess very similar retrieval effectiveness and outperform all competing hybrid descriptors. The hybrid descriptor DBD186*SIL150*RSH136 is obtained by crossbreeding the candidate feature vector C1 of dimension 186, the candidate vector C2 of dimension 150, and the candidate descriptor C3 of dimension 136.

Experimental Results

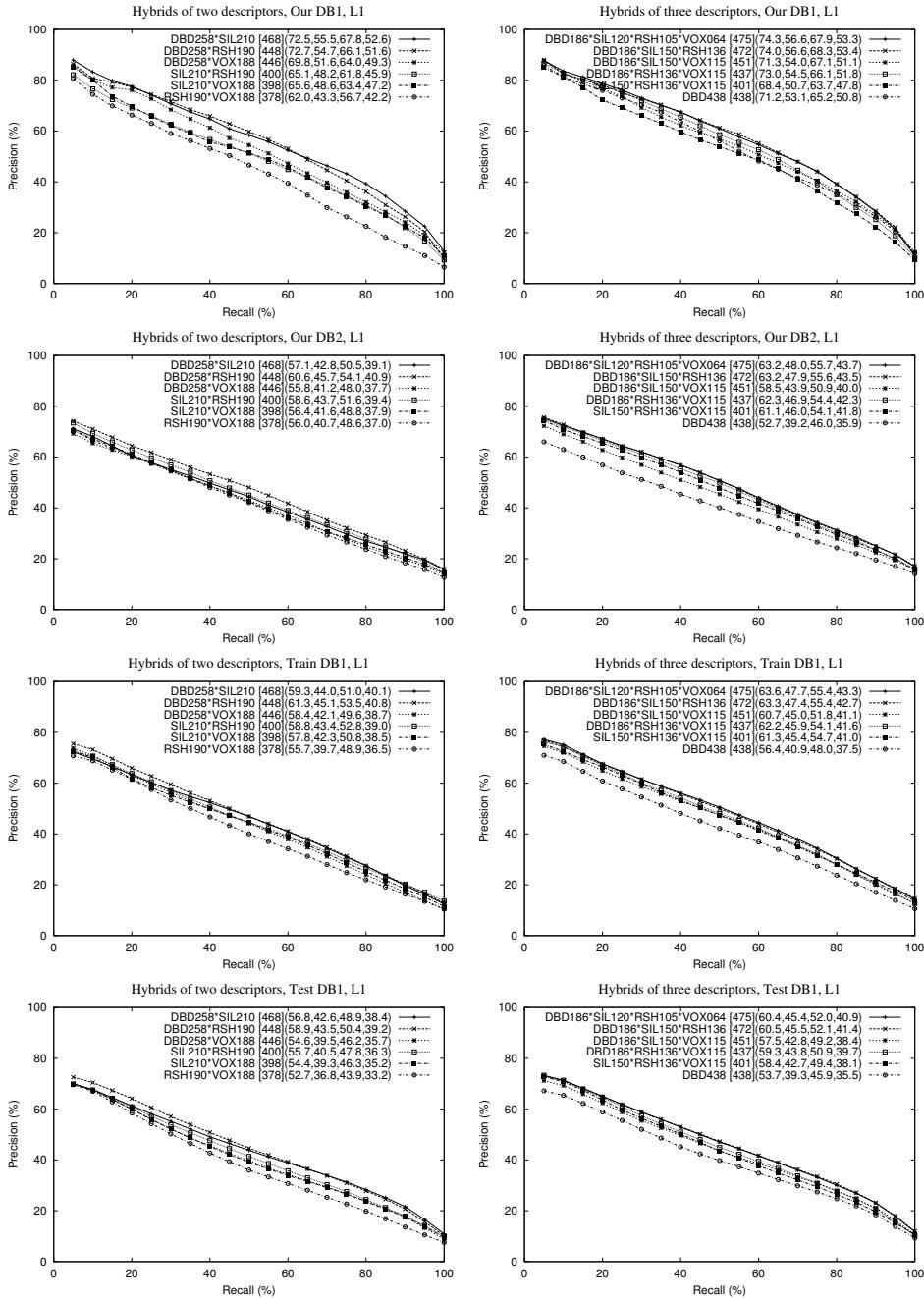


Figure 5.50: Average precision/recall diagrams of four model classifications, obtained by using various hybrid descriptors.

Since the hybrid of four descriptors is extracted in approximately 724ms, while the average extraction time of the hybrid DBD186*SIL150*RSH136 is 186ms, we recommend to use the latter, i.e., the best hybrid of three candidate descriptors. Note that the recommended hybrid is not robust with respect to outliers, because the EBB is used for extracting the depth buffer-based descriptor (C1). As demonstrated in figure 4.13, the robustness can be achieved by substituting the EBB with the CC (definition 4.4).

5.2.12 Other Feature Vectors

In this section, we evaluate descriptors that are presented in chapter 2. The abbreviations that are used for the state-of-the-art descriptors are given in table 5.2.

Firstly, we test the performance of the *cords-based descriptor* and the *moments-based feature vector* presented in section 2.1. A fraction of our experimental results is shown in figure 5.51. We infer that the best dimension of the cords-based descriptor is 120, while the d_1^{min} (1.21) is the best choice of dissimilarity measure.

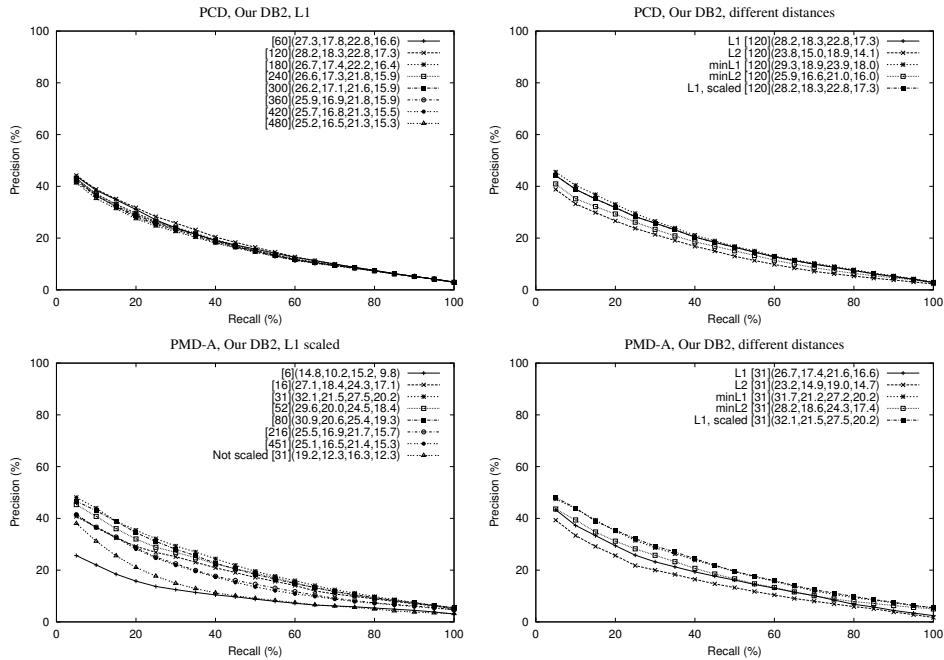


Figure 5.51: Average precision/recall diagrams of our reclassified collection, obtained by using the cords-based (first row) and moments-based descriptors (second row), presented in section 2.1. Feature vectors of different dimensions (left) and different distance measures from table 5.5 (right) are tested for both approaches.

The *descriptor based on statistical moments* (2.5) shows best performance when the vector of 31 components is first re-scaled by (4.81) and the l_1 norm is used for ranking (L1 scaled). The authors consider the original scale of an object as a feature, whence the scale normalization is not suggested in [105]. Conversely, we believe that all models should be scaled by the average distance (3.25), before extracting the moments-based descriptor. In the bottom-left diagrams in figure 5.51, the precision-recall curve of the moments-based descriptor of dimension 31, which is extracted without normalizing the scale of a 3D-model, is shown (denoted by “Not scaled”). Obviously, the retrieval performance is better if the scale normalization is performed. Experimental results for the training and test databases support the conclusions for both cords and moments-based descriptors.

We recommend the following settings for the *descriptor based on equivalence classes* (section 2.2): the EBB (definition 4.3) should be taken as the unit cube, the vector dimension should be 406, and the minimized l_1 norm d_1^{\min} (1.21) should be used for ranking. A part of experimental results, depicted in figure 5.52, illustrates that the suggested settings are the best choice.

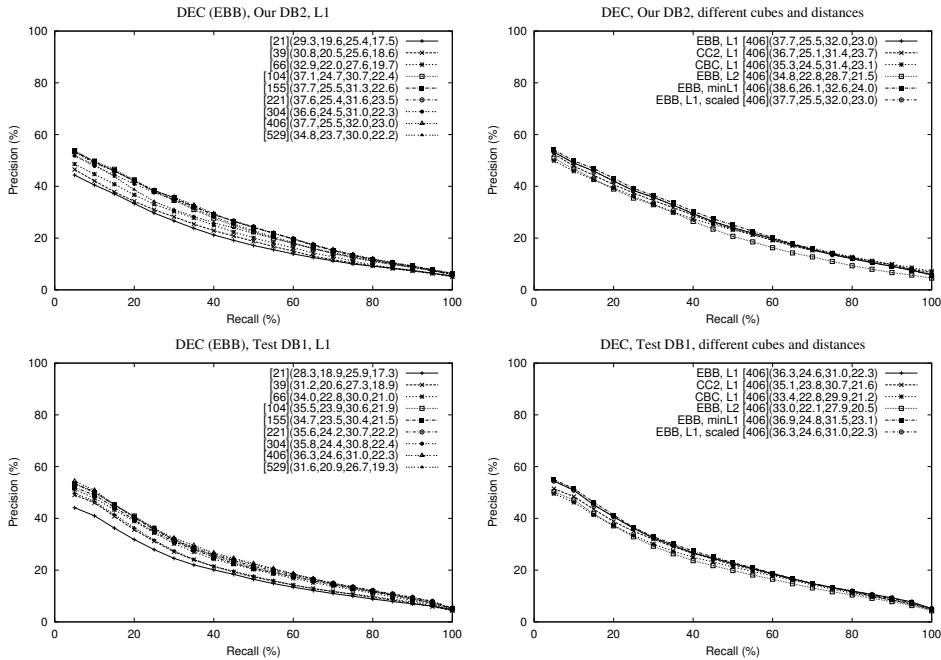


Figure 5.52: Average precision/recall diagrams of our reclassified collection and the test database, obtained by using the descriptor based on equivalence classes (section 2.2). Feature vectors of different dimensions (left) as well as different selections of the unit cube and distance measures from table 5.5 (right) are tested.

The MPEG-7 *shape spectrum descriptor* (section 2.3) is evaluated using the implementation provided inside the eXperimentation Model [84]. We tested different dimensions of the shape spectrum feature vector, extracted using the adjacent triangles or the first ($n=1$) and the second order ($n=2$). As it can be seen in figure 5.53, the overall retrieval performance of the shape spectrum descriptor is poor. The results suggest that the feature vector of dimension 452 slightly outperforms descriptors of other dimensions. Our results confirm the statement from [88] that the retrieval performance is not better if adjacent triangles of the second order are considered. The l_1 norm is the best choice of distance measure for this descriptor.

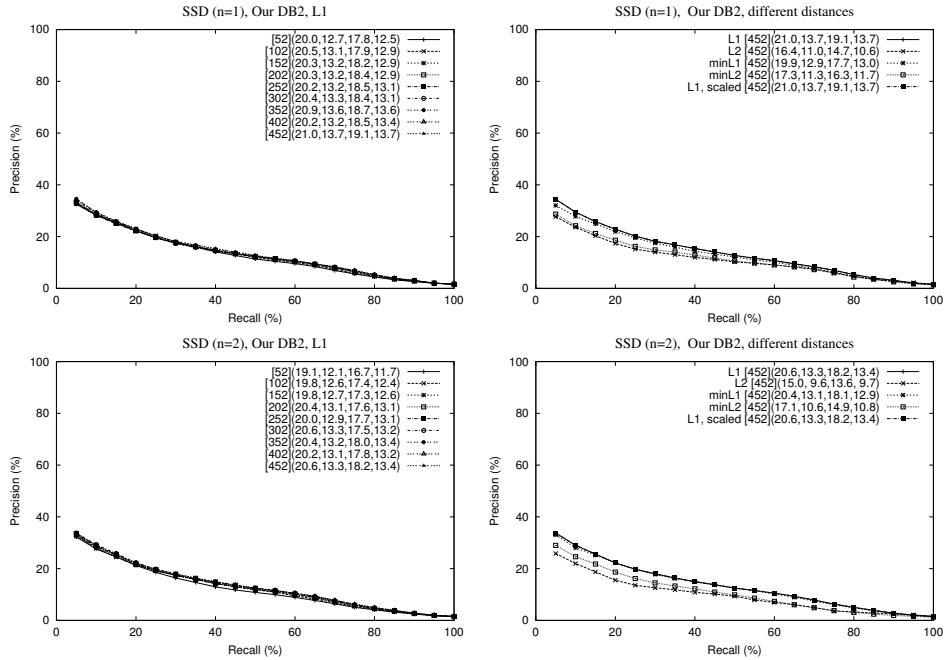


Figure 5.53: Average precision/recall diagrams of our reclassified collection, obtained by using the shape spectrum descriptor (section 2.3). Feature vectors of different dimensions (left) and different distance measures from table 5.5 (right) are tested, when the adjacent triangles of the first ($n=1$) and the second order ($n=2$) are used for computing curvature indices (2.8).

The version D2 of the *descriptor based on shape distributions* (section 2.5.1) is tested for different dimensions, while the number of samples N is fixed to 4192^2 (compare to the algorithm in figure 2.9). The results shown in figure 5.54, suggest that the best choices of vector dimension and distance metric are $dim = 64$ and l_1 (1.10). We stress that the authors suggested the same dimension in [97, 98].

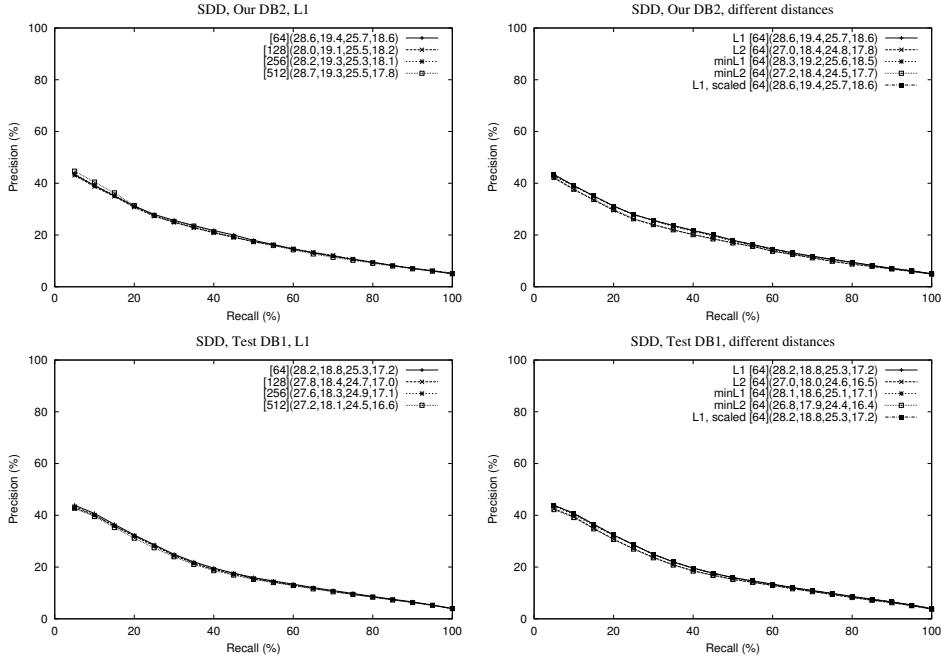


Figure 5.54: Average precision/recall diagrams of our reclassified collection and the test database, obtained by using descriptors based on shape distributions (D2) (section 2.5.1). Feature vectors of different dimensions (left) and different distance measures from table 5.5 (right) are tested. The number of samples is $N = 4192^2$.

We compare three variants of the *descriptor based on binary voxel grids* (section 2.5.2). The first variant (B1) is extracted using the original definition of binary functions (2.26) given in [36]. The second variant (Br) relies upon the redefinition of functions on a sphere given by (2.29). Both variants are extracted without applying the CPCCA (section 3.4) to the original mesh model, i.e., without finding a canonical orientation. We recall that only translation and scale are normalized, while the property 4.1 of spherical harmonics is used to form inherently rotation invariant feature vectors (2.27). We implemented the third variant (Br, our approach) of the descriptor based on binary voxel grids so that each 3D-model is normalized first, then the binary voxel grid is generated, the functions defined by (2.29) are sampled, and the spherical Fourier transform (section 4.6.1) is applied to each function. Instead of forming the signatures by (2.27), we used the algorithm 4.1, i.e., (4.78). The dimension of the first and the second version of the descriptor based on binary voxel grids is $\dim = RL$, where R is the number of functions on concentric spheres and L is the number of bands of spherical harmonics that are used to compute the signatures. In [57, 36], the recommended settings are $R = 32$

and $L = 16$, i.e., $\dim = 512$. The dimension of the third version of feature vector is $\dim = R \cdot L(L + 1)/2$. Since the number of functions on concentric spheres is fixed to $R = 32$ [57, 36], we set $L = 6$ so that the dimension of the third variant of the descriptor is $\dim = 672$. Experimental results obtained for the classifications O1, O2, TR, and TS (table 5.3), aimed at comparing the three variants of the descriptor based on binary voxel grids, are displayed in figure 5.55. The results for all four classifications are consistent. The variant “B1” is evidently inferior to the variant “Br”, whence the 3D-shape is represented in a more appropriate manner if the definition (2.29) is used instead of (2.26). The variant “Br, our approach” is evidently superior to the variant “Br”. Therefore, if the binary voxel grid is considered as a feature that describes shape of a 3D-model, than the representation relying upon the property 4.1 of spherical harmonics and avoiding the use of the CPCPA is inferior to our method that uses the CPCPA and (4.78) to form the vector.

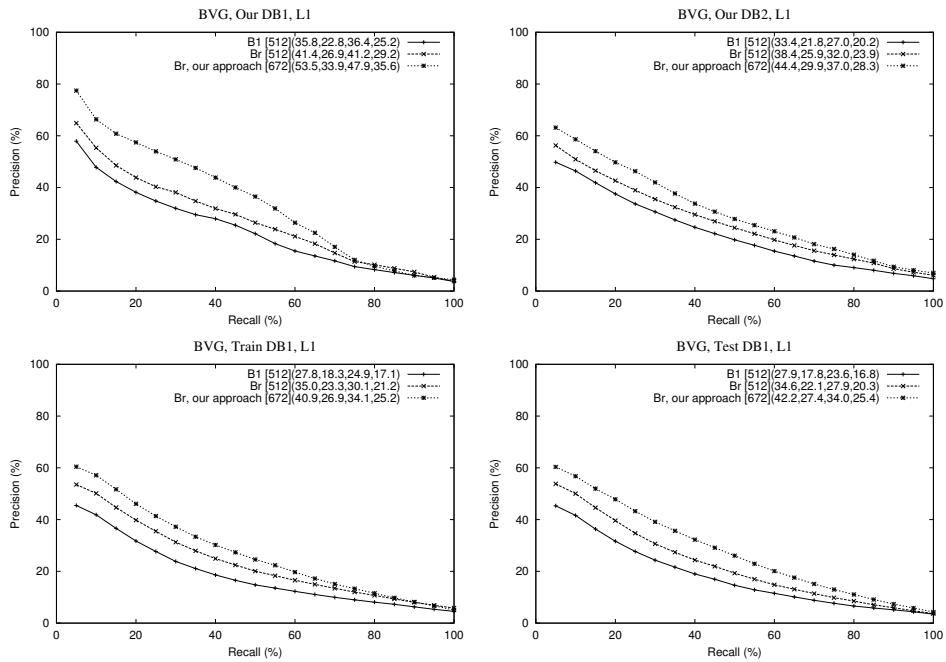


Figure 5.55: Average precision/recall diagrams of four model classifications, obtained by using the descriptor based on binary voxel grids (section 2.5.2). The descriptor relying on binary functions on a sphere defined by (2.26) is denoted by “B1”, while the feature vector extracted using the redefinition (2.29) is denoted by “Br”. The descriptor formed by using (2.29) and by applying the algorithm 4.1 (i.e., the CPCPA and (4.78)) are used instead of (2.28)) is denoted by “Br, our approach”.

Thus, the conclusion inferred using the results in figure 5.49 is supported by the results given in figure 5.55, i.e., the CPCa should not be avoided, regardless of certain weaknesses (section 3.5), because the descriptor relying upon the CPCa possesses higher discriminant power than the descriptor relying upon the competing technique for achieving rotation invariance.

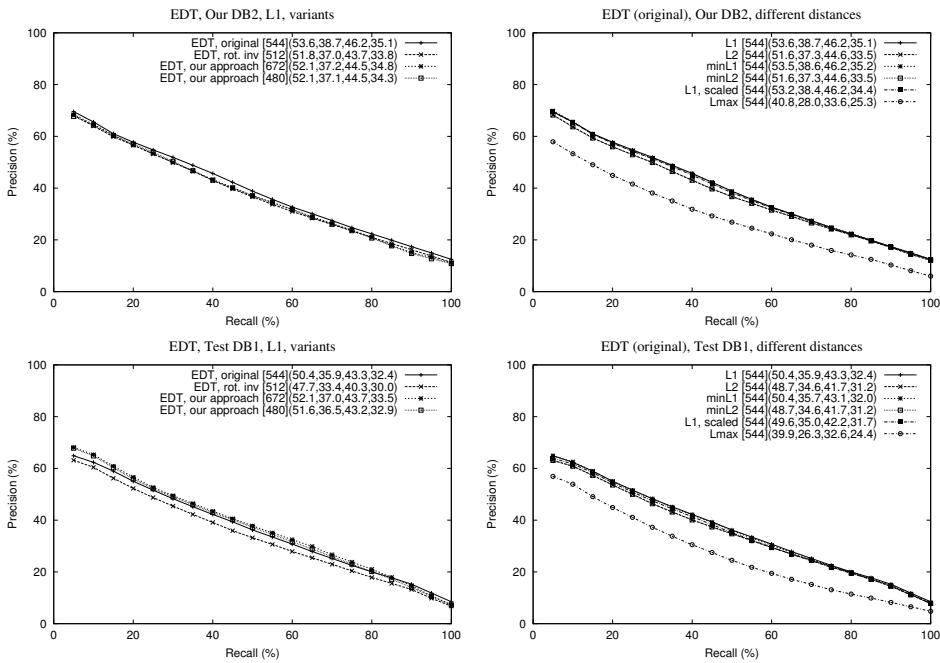


Figure 5.56: Average precision/recall diagrams of our reclassified collection and the test database, obtained by using descriptors based on exponentially decaying EDT (section 2.5.4). On the left-hand side, the descriptors extracted by the executables provided by the authors are denoted by “EDT, original”, our implementation of the same approach is denoted by “EDT, rot. inv”, while “EDT, our approach” denotes the application of the algorithm 4.1 (i.e., we use the CPCa and (4.78) instead of (2.37)). On the right-hand side, the descriptors extracted using the executables provided by the authors are tested for different distance measures from table 5.5.

The performance of the *descriptor based on the exponentially decaying Euclidean distance transform* (EDT), which is described in section 2.5.4, is evaluated using executables (binaries) that are provided by the authors [108]. When the first $L = 16$ bands of spherical harmonics of $R = 32$ functions on concentric spheres are used to form the feature vector according to (2.37), the dimension of the resulting descriptor is $\dim = R(L + 1) = 544$. In figure 5.56, the precision-recall curves and the corresponding evaluation parameters (in brackets) obtained by analyzing descriptors

generated by using the provided executables are denoted by “EDT, original”.

We also implemented the same method using the functions on a sphere defined by (2.33), without the postprocessing (2.36), but simply using (2.28) and normalizing the extracted feature vector to the Euclidean unit length. Our implementation of the method is denoted by “EDT, rot. inv”. Without the postprocessing, the dimension of the feature vector is $\dim = RL = 512$.

Similarly to figure 5.55, we modify the descriptor based on exponentially decaying EDT, which is inherently invariant with respect to rotations of a 3D-object. The modification consists of applying the CPCPA in the normalization step and forming the feature vector according to (4.78). Thus, the property 4.1 of spherical harmonics is not used to form inherently rotation invariant feature vectors, but the invariance is secured by the CPCPA. In figure 5.56, the modification of the approach based on exponentially decaying EDT is denoted by “EDT, our approach”. We recall that the dimension of the modified feature vector is computed by $\dim = R \cdot L(L + 1)/2$. We tested dimensions 480 ($L = 5$) and 672 ($L = 6$) of the modified descriptor.

The results depicted in figure 5.56 show that our implementation “EDT, rot. inv” possesses lower retrieval performance than the original implementation. We assume that this difference is not caused by the postprocessing step that is originally used, rather by certain parameter settings that are not explicitly stated in [58]. However, we notice that “EDT, our approach” clearly outperforms “EDT, rot. inv” (left-hand side in figure 5.56). Hence, our third comparison of approaches for achieving rotation invariance, the CPCPA vs. the property 4.1, gives the same conclusion as the previous two tests (see figures 5.49 and 5.55). Namely, the descriptor relying upon the CPCPA (EDT, our approach) possesses higher discriminant power than the descriptor relying upon the competing technique (EDT, rot. inv). We assume that if the original implementation (EDT, original) is modified in the same way, the conclusion will still be the same. We also notice that “EDT, our approach” outperforms “EDT, original”, when the test database is used.

On the right-hand side in figure 5.56, we examine the influence of the used dissimilarity measure (table 5.5) on retrieval effectiveness of the original implementation of the descriptor based on exponentially decaying EDT. We observe that the l_1 norm is the best choice of distance metric. Note that the postprocessing described in section 2.5.4 is aimed at minimizing the l_2 distance of functions on a sphere consisting of the constant and the quadratic component (2.34). However, our experimental results show that the l_2 norm does not outperform the l_1 distance as a tool for ranking the descriptors based on exponentially decaying EDT.

By comparing results given in figures 5.55 and 5.56, we observe that the discriminant power of the descriptor relying upon the EDT is significantly higher than the discriminant power of the descriptor relying upon the binary voxel grid. The binary voxel grid as a feature is inferior to the voxel grid attributed by exponentially decaying EDT. By defining functions on concentric spheres using a binary voxel grid, function-mismatching problems depicted in figure 2.11 reduce the retrieval effectiveness of the extracted descriptor. Conversely, if a voxel attribute (EDT) describes how far an arbitrary point is from the model, then the problem of mismatched functions on a sphere is almost eliminated.

5.2.13 Global Comparison

The global comparison of the best versions of all 3D-shape descriptors, which are tested in sections 5.2.1–5.2.12, is performed in two stages, the group stage and the final stage. Firstly, we subdivided almost all descriptors listed in table 4.1 into two groups. The only descriptor from table 4.1 that is omitted in both groups is the RID, because the direct comparison to the LDS (figure 5.49) showed the inferiority of the RID descriptor. The methods listed in table 5.2 form the third group of competing methods. The BVG descriptor is omitted from the third group, because its inferiority to the EDT descriptor is stated in section 5.2.12. In the final stage, the best descriptors from each group are compared to the best hybrid descriptor (section 5.2.11). Most of the precision-recall curves, which are displayed in this subsection, can be found in the previous subsections.

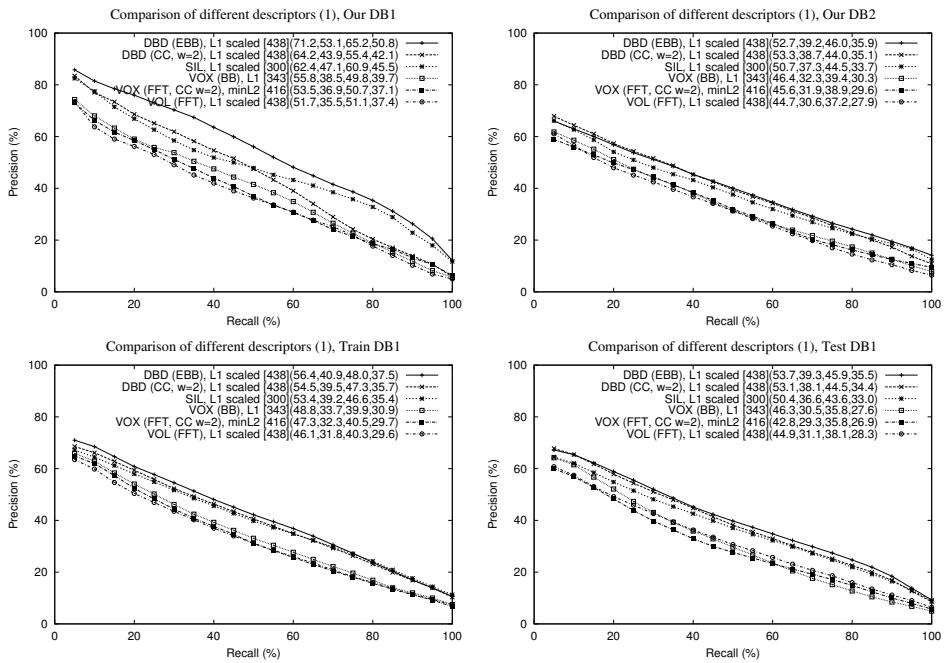


Figure 5.57: Average precision/recall diagrams of four model classifications (table 5.3) for the best variants of descriptors (table 4.1) DBD (section 5.2.3), SIL (section 5.2.2), VOX (section 5.2.5), and VOL (section 5.2.4).

The first group of competing descriptors is made up of the best versions of the depth buffer-based descriptors (section 5.2.3) extracted using the EBB (definition 4.3) and the CC (definition 4.4) with $w = 2$, the silhouette-based descriptor (section 5.2.2), the voxel-based descriptors (section 5.2.5) in the spatial domain extracted

using the BB (definition 4.2), the voxel-based descriptor in the spectral domain extracted using the CC with $w = 2$, and the volume-based descriptor in the spectral domain (section 5.2.4). The recommended dimension and dissimilarity measure (table 5.5) of each descriptor are used for creating the precision-recall diagrams in figure 5.57. We conclude that the depth buffer-based descriptor extracted using the EBB outperforms the competing feature vectors. As mentioned in sections 4.3 and 5.2.3 and demonstrated by the example in figure 4.13, an outlier can significantly affect components of the vector created by relying upon the EBB, while the descriptor extracted using the CC is more robust with respect to outliers. We also notice that the silhouette-based descriptor outperforms the remaining three feature vectors.

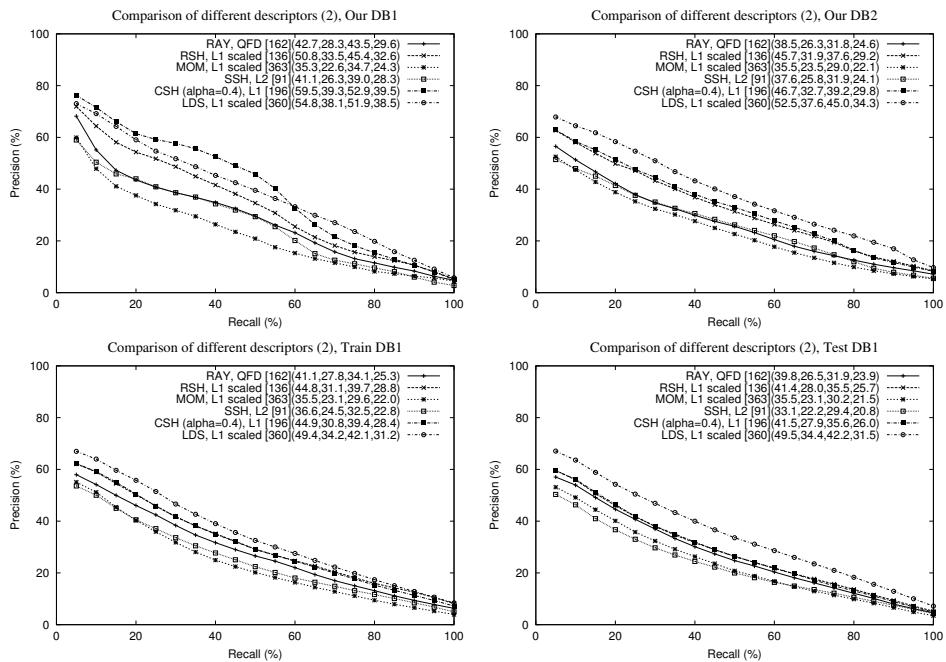


Figure 5.58: Average precision/recall diagrams of four model classifications (table 5.3) for the best variants of descriptors (table 4.1) RAY (section 5.2.1), RSH (section 5.2.6), MOM (section 5.2.7), SSH (section 5.2.8), CSH (section 5.2.9), and LDS (section 5.2.10).

The second group of competing descriptors consists of the ray-based descriptor in the spatial domain (section 5.2.1), the ray-based feature vector in the spectral domain (section 5.2.6), the moments-based descriptor (section 5.2.7), the shading-based descriptor (section 5.2.8), the complex-based descriptor (section 5.2.9) extracted using the function (4.72) with $\alpha = 0.4$, and the descriptor based on

LDSs (section 5.2.10) defined by (4.78). For each descriptor, the recommended dimension and distance measure (table 5.5) are used. The results in figure 5.58 suggest that the feature vector based on LDSs outperforms the competing descriptors from the second group. The complex descriptor outperforms the remaining four feature vectors. The shading-based descriptor outperforms only the moments-based descriptor. Finally, we observe that we benefit if the ray-based feature is represented by spherical harmonics, in the spectral domain. Thus, besides filtering the surface noise, which is captured by measuring the extent in the spatial domain, providing an embedded multi-resolution feature representation (1.8), and improving robustness with respect to outliers (see figures 4.5 and 4.27), the representation in the spectral domain possesses higher effectiveness than the representation of the ray-based feature in the spatial domain.

The descriptors proposed by other authors (chapter 2) are compared in figure 5.59. Evidently, the descriptor presented in [58] that is based on exponentially decaying EDT (section 2.5.4) significantly outperforms all other approaches. Therefore, we regard the EDT descriptor as the state-of-the-art descriptor. We stress that we use the executables (binaries) provided by the authors [108] to extract the EDT descriptor.

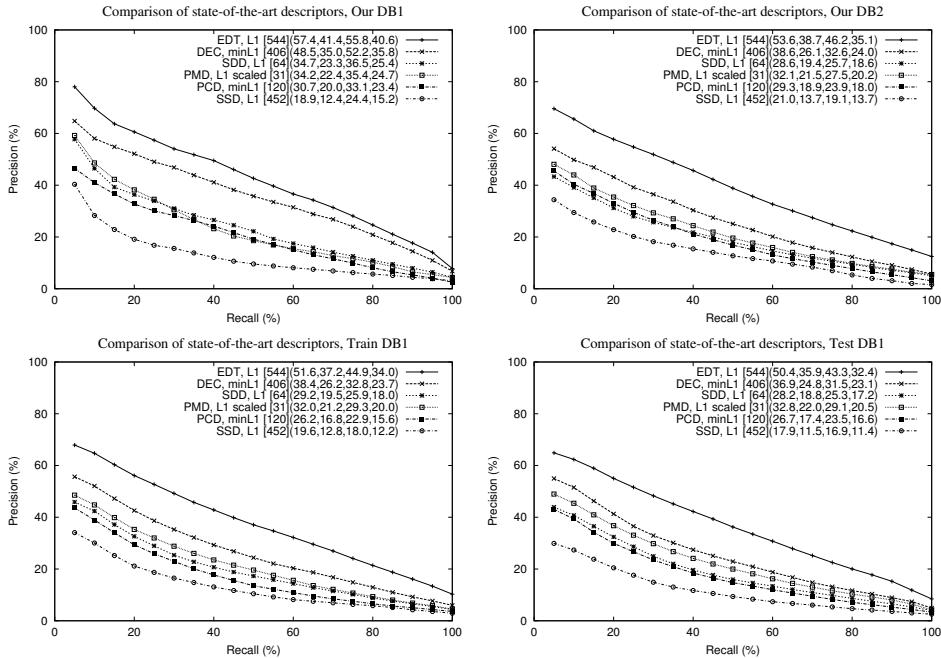


Figure 5.59: Average precision/recall diagrams of four model classifications (table 5.3) for the best variants of descriptors (table 5.2) EDT, DEC, SDD, PMD, PCD, and SSD (section 5.2.12).

In our final comparison, we collected the best precision-recall curves from figures 5.57, 5.58, 5.59, and 5.50 in figure 5.60. The recommended hybrid descriptor (DBD186*SIL150*RSH136) of dimension 472 significantly outperforms all other descriptors. We recall that the hybrid is obtained by crossbreeding the depth buffer-based descriptor of dimension 186 (4.30) extracted using the EBB (definition 4.3), the silhouette-based feature vector of dimension 150 (4.15) extracted using the equiangular sampling (4.10) and the definition of sample values given by (4.13), and the ray-based descriptor of dimension 136 with spherical harmonic representation (4.66).

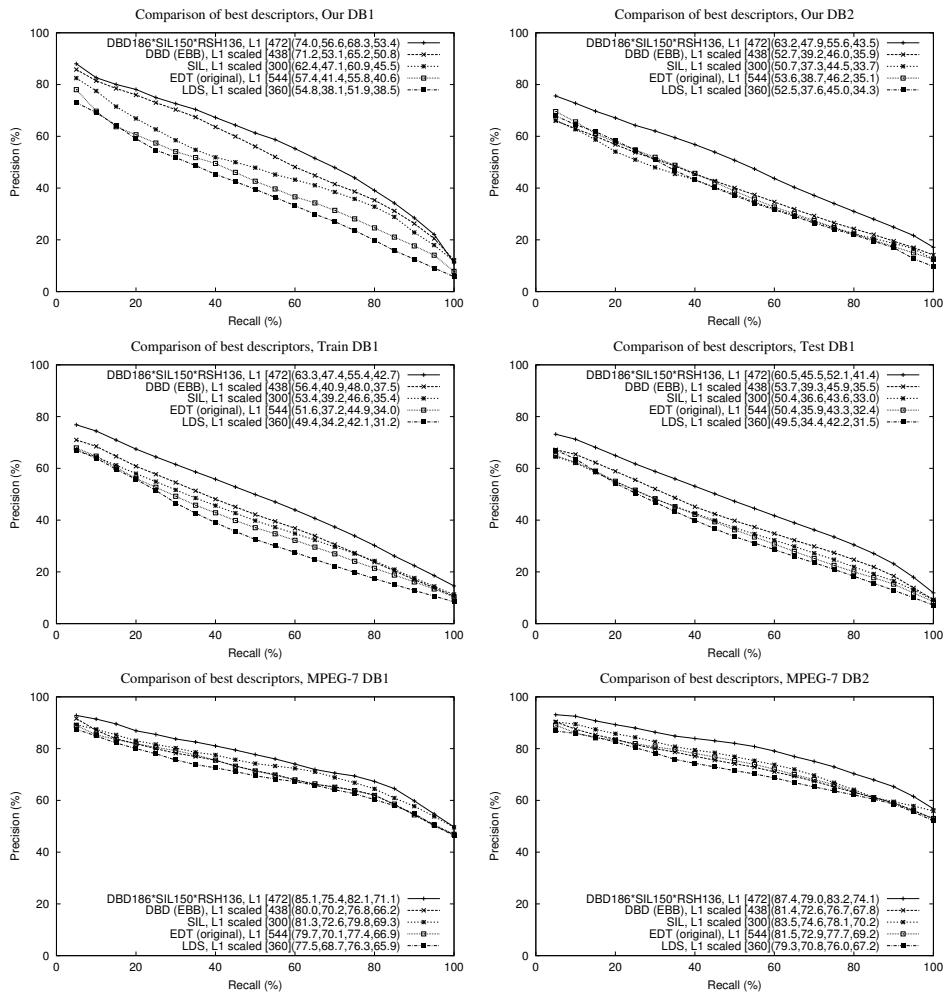


Figure 5.60: Average precision/recall diagrams of six model classifications (table 5.3), obtained by using the best descriptors in figures 5.50, 5.57, 5.58, and 5.59.

In what follows, we will regard the hybrid descriptor DBD186*SIL150*RSH136 as the DSR472 descriptor. Note that the DSR472 fulfills the criterion (5.1). Of all descriptors compared in figure 5.60, the descriptor based on LDSs shows the lowest retrieval performance. We consider that the depth buffer approach slightly outperforms the state-of-the-art descriptor (EDT, original). The descriptor based on exponentially decaying EDT possesses higher discriminant power than the silhouette-based feature vector.

We also computed the percentage of retrieving a relevant model as the best match (nearest neighbor), when each classified model is used as a query (see table 5.1). The results shown in table 5.8 confirm the superiority of the DSR472 descriptor. The original implementation of the descriptor based on exponentially decaying EDT retrieves more non-relevant best matches than the depth buffer-based feature vector, on average.

Classification	O1	O2	TR	TS	M1	M2
DSR472	85.3%	70.1%	70.6%	66.4%	91.6%	91.0%
DBD (EBB)	81.8%	60.0%	63.1%	60.0%	90.3%	87.8%
SIL	80.6%	60.4%	59.6%	55.6%	87.2%	87.4%
EDT (original)	71.8%	62.7%	60.3%	57.8%	85.5%	85.6%
LDS	66.5%	60.8%	58.3%	60.1%	84.6%	81.1%

Table 5.8: Percentage of retrieving a relevant model as the best match to a query (nearest neighbor) for all six collections (table 5.3), using the descriptors from figure 5.60. The DSR472 descriptor (i.e., the hybrid DBD186*SIL150*RSH136) shows the best performance for all available collections.

We consider that our evaluation is complete, because we estimate that the approaches presented in [2, 72, 153, 151, 95, 27, 28, 19, 77, 11, 65, 129, 93, 91, 92] as well as the topology matching technique [48] (section 2.4) are inferior to the state-of-the-art descriptor, which is described in section 2.5.4. Our estimation is partially based on results that are presented in the literature. For instance, the results presented in [36] demonstrate that the descriptor based on exponentially decaying EDT significantly outperforms descriptors proposed in [2] and [28]. The techniques proposed in [19, 77, 11] are suitable for CAD 3D-models. The results presented in [72, 153, 151, 95, 27, 65, 129, 93] also suggest the inferiority to the EDT descriptor [58]. The topology matching technique is presented in [48], but no decent evaluation has been published yet. Thus, we assume that the sensitivity of the method, which is depicted in figure 2.8, deteriorates the overall effectiveness. We consider that methods presented in [91, 92] have rather theoretical than practical value. Finally, we used six different model classifications to evaluate the retrieval performance of descriptors. Since the results are consistent for all available ground-truth classifications, we believe that the best 3D-shape descriptor has been found.

We stress that the DSR472 is the best descriptor in general. However, for certain categories of 3D-models the DSR472 might not be the most suitable descriptor. An extreme example is shown on the left-hand side in figure 5.61. Typically the most inferior descriptor (see figure 5.59), the shape spectrum feature vector, is compared

to the best descriptors (figure 5.60), but only for the category of models of humans. The category belongs to our reclassified collection and consists of 56 models of humans in different poses (some of the models are articulately modified). Evidently, generally the poorest descriptor (SSD) outperforms the best descriptors. The results are expected, because the SSD is the only descriptor robust with respect to articular modifications of 3D-models, when the level-of-detail and tessellation are not significantly changed. We recall that the SSD is not robust to different tessellations and levels-of-details. For the category of models of fighter jets (the training database), the most suitable descriptor is the depth buffer-based. Nevertheless, we consider that the global results (not category-wise) suggest which descriptor is the best for the majority of categories.

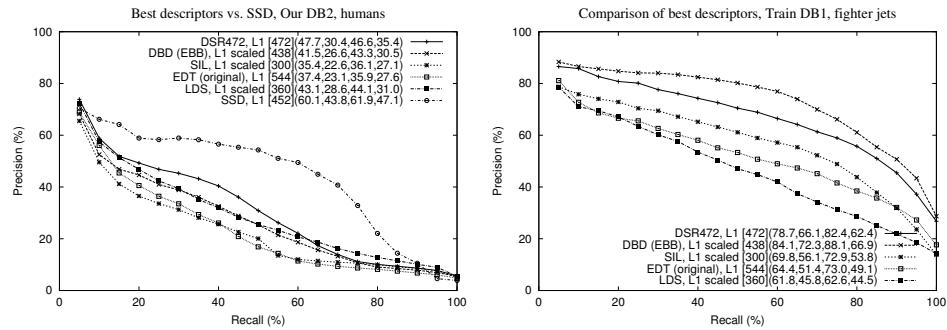


Figure 5.61: Average precision/recall diagrams of categories “human” from our reclassified collection and “fighter jet” from the training database, obtained by using the best overall descriptors. For the category of human models, the shape spectrum descriptor (sections 2.3) is compared to the best descriptors (left).

5.3 Dimension Reduction using the PCA

In this section, we demonstrate that the dimension of a feature vector can be reduced using the PCA (section 3.2) so that the loss in retrieval effectiveness is usually not significant. Moreover, in certain cases, the retrieval performance can be improved by applying the standard PCA to a set of feature vectors.

We performed the PCA analysis of the four best descriptors (section 5.2.13), the hybrid descriptor DSR472, the depth buffer-based descriptor (DBD), the silhouette-based feature vector (SIL), and the descriptor based on exponentially decaying Euclidean distance transform (EDT). The EDT descriptor is extracted using the original tools provided by the authors [108]. The analysis is performed on the descriptors associated to our reclassified collection of 1835 3D-models (see section 5.1). Thus, there are four sets (3.2) of 1835 feature vectors of dimensions 472 (DSR472), 438 (DBD), 300 (SIL), and 544 (EDT). We recall that the compression (dimension reduction) is achieved by multiplying the original n -dimensional vectors

in each set by the corresponding matrix A_k (3.8), obtaining k -dimensional vectors ($k < n$). The value of the parameter k can be fixed or selected so that a fixed amount of “energy” (variance) of the initial data is preserved after the transformation. We tested the second approach by setting $t = 0.95$ in equation (3.9), i.e., we want to determine the value of the parameter k so that at least 95% of energy of the original data vectors is preserved. The following values of the parameter k are obtained: $k = 35$ for the set of DSR472 descriptors, $k = 25$ for the set of depth buffer-based feature vectors, $k = 12$ for the set of silhouette-based feature vectors, and $k = 32$ for the set of descriptors based on exponentially decaying EDT.

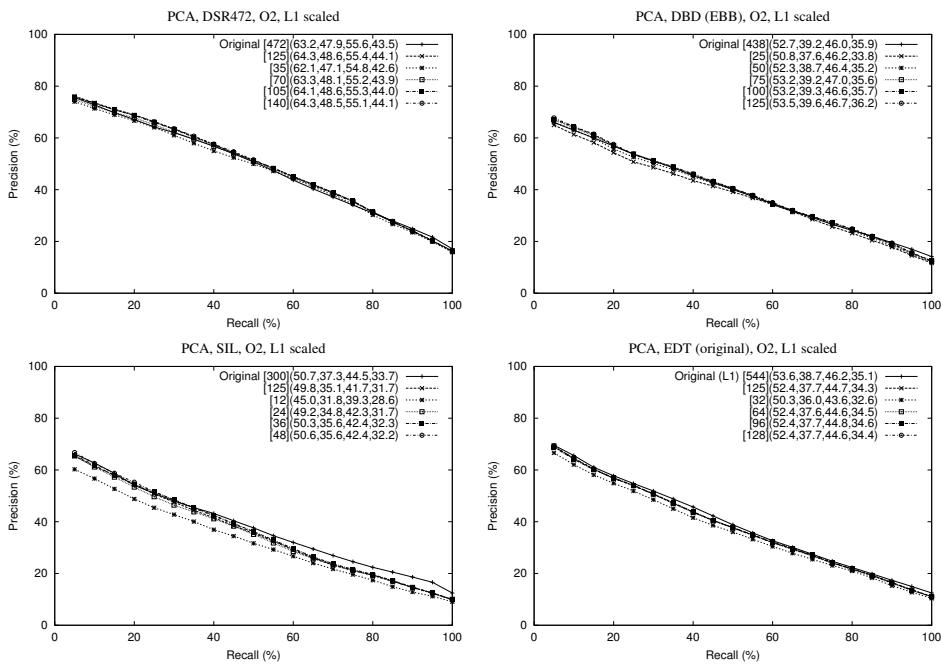


Figure 5.62: Average precision/recall diagrams of our reclassified collection, obtained by using the original descriptors and compressed feature vectors of different dimensions generated by applying the standard PCA (section 3.2) to the corresponding sets of original descriptors.

In figure 5.62, we compare the retrieval performance of the original descriptors vs. the compressed descriptors. Dimensions of compressed feature vectors are $\dim = k$, $\dim = 2k$, $\dim = 3k$, $\dim = 4k$, and $\dim = 125$, where the parameter k is selected using (3.9) with $t = 0.95$. Separate tests, which are omitted, suggested that the l_1 distance applied to normalized feature vectors (L1 scaled) is the best choice of distance measure (table 5.5) for compressed feature vectors. The results confirm that the standard PCA can be used for reducing dimension of feature vectors without significant loss in retrieval effectiveness. The used linear dimension

reduction technique is optimal in the mean-square sense [51]. We observe that the retrieval performances of the hybrid DSR472 and the depth buffer-based descriptors are even increased by compressing the original vectors. The explanation for the increased retrieval effectiveness lies in the fact that the presence of noise in original data is also reduced. Note that the feature vector of 35 components generated by compressing the DSR472 descriptor outperforms both the depth buffer-based and the EDT descriptors.

The compression by applying the PCA analysis to sets of feature vectors has two goals: to preserve the retrieval performance and to reduce computational costs of subsequent processing steps. The feasibility of the first goal is demonstrated in figure 5.62. Obviously, the ranking of low-dimensional vectors using the dissimilarity measures from table (5.5) is faster than the ranking of high-dimensional feature vectors. The average ranking times (distance computations and sorting) of our reclassified collection (1835 models) when the l_1 norm (1.10) is used for computing distances between feature vectors of dimensions 150, 300, and 500 are 50ms, 100ms, and 155ms, respectively. For the set of 1835 vectors, the computation of the 472×472 covariance matrix (3.4) took 8.4 seconds, while the computation and sorting of the eigenvalues and eigenvectors (3.5), i.e., the computation on the transformation matrix, lasted 40.0 seconds. The results are obtained on a computer running Windows 2000 Professional, with 1GB RAM and an 1.4 GHz AMD processor.

The application of the PCA to feature vectors will be investigated further. For instance, we plan to apply the PCA to weighted concatenations of feature vectors. Also, an addition of a large number of similar objects to a collection of 3D-models may significantly change the principal axes of the corresponding set of feature vectors. Therefore, we expect that certain modifications of the standard PCA will be necessary in order to provide robustness with respect to the variance of 3D-shapes in a given collection. Finally, if a user upload a new model to our Web-based retrieval system CCCC (see appendix), the feature vectors are automatically extracted. Assuming that a single added feature vector cannot significantly change the computed transformation parameters, we can use the previously calculated mean vector (3.3) and transformation matrix A_k to reduce the vector dimension, so that a user receives a prompt response from the system. Then, the mean vector (3.3) and the transformation matrix can be updated (re-computed) in an idle time. However, the robustness of the computed transformation parameters to an enlargement of the underlying set of data vectors (descriptors) by a single data vector should be examined.

5.4 Summary of Experimental Results

In section 5.2, we analyzed retrieval effectiveness of 19 types of 3D-shape descriptors (see tables 4.1 and 5.2). Most of the descriptors have different variants. Various parameter settings affect the retrieval performance of descriptors. We used six 3D-model classifications (section 5.1) to evaluate retrieval effectiveness of each type of

descriptor, for various variants and parameter settings, using different dissimilarity measures (table 5.5). A selection of more than 3000 generated precision-recall diagrams is shown in section 5.2.

We consider that the most important state-of-the-art descriptors are included in our tests (section 5.2.13). The results show that the DSR472 hybrid descriptor is unambiguously the best 3D-shape descriptor, in general. However, the version of the DSR472 feature vector that is tested in sections 5.2.11 and 5.2.13 relies upon a parental descriptor that is not robust with respect to outliers. Namely, the depth buffer-based descriptor extracted using the extended bounding box (definition 4.3) is sensitive to outliers, whence a similar level of sensitivity is inherited by the hybrid. Conversely, if the DSR472 descriptor is obtained by crossbreeding the depth buffer-based feature vector extracted relying upon the canonical cube (definition 4.4) with $w = 2$, the silhouette-based descriptor, and the ray-based feature vector in the spectral domain, then the resulting hybrid is robust with respect to outliers, because all parental feature vectors are not sensitive to outliers.

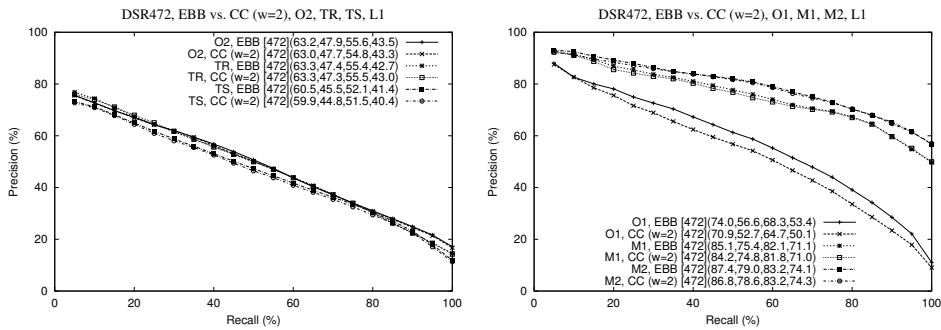


Figure 5.63: Average precision/recall diagrams of six model classifications (table 5.3), obtained by using two variants of the DSR472 hybrid, the variant that is sensitive to outliers (DBD is extracted using the EBB (definition 4.3)) and the variant that is robust with respect to outliers (DBD is extracted using the CC (definition 4.4) with $w = 2$).

In figure 5.63, we compare the variant of the DSR472 descriptor that is sensitive to outliers (DSR472, EBB) to the variant that is robust with respect to outliers (DSR472, CC $w=2$), using all available ground truth classifications (table 5.3). The results show that the variant that is sensitive to outliers slightly outperforms the robust variant of the DSR472 descriptor. Having in mind the example in figure 4.13, we consider that it is important to satisfy the requirement 5 from section 1.3.4. Therefore, we recommend to use the DSR472 hybrid descriptor, which is robust with respect to outliers, as the best choice of all tested approaches (descriptor types, variants, and parameter settings). The instances of the DSR472 feature vector should be ranked using the l_1 distance (1.10).

Chapter 6

Conclusion

The topic of the thesis is characterization of global shape of a 3D-mesh model by defining appropriate 3D-shape descriptors (feature vectors). The feature vectors are used for retrieval of 3D models by shape-similarity. In this chapter, we summarize the contribution, stress the most important results, and suggest directions for future work.

In chapter 1, we describe the architecture of our 3D model retrieval system (figure 1.5), discuss types of 3D-shape features, and set criteria for 3D-shape descriptors. Most of the requirements are also addressed in the MPEG-7 committees [85]. However, certain requirements are specific for 3D-mesh models, e.g., robustness with respect to different levels-of-detail, tessellations, and surface noise of a mesh model. As far as we know, the requirement for robustness with respect to outliers is mentioned in [144] for the first time. Afterwards, other authors modified their approaches in order to fulfill the requirement. For instance, the descriptor proposed in [57] relies upon a bounding sphere, whence the corresponding descriptor is sensitive to outliers. The same authors modified the approach in order to provide the invariance to outliers [36]. We also formally define the properties of closedness and orientability of polygonal meshes. We modified the quadratic form distance, which is used in [2] for retrieving 3D-models, so that searching using certain types of descriptors is more effective. As far as we know, the use of the minimized l_1 and l_2 norms as tools for ranking feature vectors has not been reported in the literature yet. Thus, our optimized algorithm for computing the minimized l_1 distance between two n -dimensional vectors with real-valued components (figure 1.8) represents an original result.

In chapter 2, we describe several techniques proposed by other authors. Our contribution to the method based on equivalence classes [126, 125] is the exploration of different parameter settings that are not addressed in the original papers. We consider that the descriptor proposed in [58] represent the most effective 3D-shape feature vector proposed by other authors.

In chapter 3, we present our *Continuous Principal Components Analysis* (CPCA) for normalizing the pose of a 3D-mesh model. By applying the PCA to a set

of vertices of a 3D-mesh, differing sizes of triangles are not taken into account. Modifications of the PCA are introduced in [143] and [105], in order to account different sizes of triangles by suitable weighting factors. We regard a triangle mesh model as a union of triangles, whence the point set of the model consists of infinitely many points. In contrast to the usual application of the PCA, we work with sums of integrals over triangles (3.19) in place of sums over (weighted) vertices which makes our approach more complete taking into account all points of the model with equal weight. The formulas for computing all necessary parameters for the normalization of translation, rotation, scale, and reflection, using the continuous approach, are given in section 3.4.

In chapter 4, we present our original 3D-shape descriptors, which are defined using various features and representation techniques. We considered a variety of features for characterizing 3D-shape such as

- Extents of a model in certain directions;
- Contours of 2D projections of a model;
- Depth buffer images of a model;
- Artificially defined volumes associated to triangles of a mesh;
- Voxel grids attributed by fractions of the total surface area of a mesh;
- Rendered perspective projections of a model on an enclosing sphere;
- Layered depth spheres.

As far as we know, the layered depth spheres represent an original concept, which is introduced in this thesis.

The used representation techniques include:

- Fourier transforms: 1D, 2D, 3D;
- Fourier transform on a sphere;
- Moments for representing the extent function (original definition).

We stress that spherical harmonics (the fast Fourier transform on a sphere) are introduced as a tool for 3D model retrieval by ourselves in [147].

We introduced two approaches for merging appropriate feature vectors, by defining a complex function on a sphere, and by crossbreeding. The concept of complex descriptors is proposed in [146], while the concept of hybrid feature vectors is introduced in this thesis. We also present a variety of original feature extraction algorithms and give complete specifications for forming feature vector components for each of presented approaches.

In chapter 5, we evaluate 19 types of 3D-shape descriptors (see tables 4.1 and 5.2) using six different classifications (ground truths) of 3D-models (section 5.1). Most of the 3D-model classifications are not formed by ourselves, whence we consider that our evaluation is not subjective. As tools for comparing competing descriptors, we use precision-recall diagrams, the R-precision (first tier), the Bull's eye performance (second tier), and two other parameters that are presented in section 1.5. As far as we know, no reported results in the literature include this number of tested techniques and this number different classifications. Since the state-of-the-art descriptors [36, 58] are also tested, we believe that our evaluation is competent.

The global comparison (section 5.2.13) of the best versions of all 19 descriptors unambiguously suggest that a version of our hybrid descriptor significantly outperforms all competing descriptors. The best descriptor, which we called “DSR472”, is formed by crossbreeding the following three descriptors:

1. Depth buffer-based descriptor of dimension 186 (section 4.3), extracted from depth buffer images of dimensions 128×128 , which are formed using the canonical cube (definition 4.3) with $w = 2$;
2. Silhouette-based descriptor of dimension 150 (section 4.2), extracted from silhouette images of dimensions 256×256 using 256 equiangular sample points (4.10), where sample values are defined by (4.13);
3. Ray-based descriptor of dimension 136, with spherical harmonic representation (section 4.6.2), obtained by sampling the extent function (4.1) at 128^2 points (4.63).

The DSR472 descriptor possesses all desirable properties specified in section 1.3.4 such as robustness with respect to levels-of-detail, different tessellations, surface noise, outliers, arbitrary topological degeneracies. The feature extraction and the search procedure are efficient, as well. The most important, discriminant power of the DSR472 descriptor is significantly higher than discriminant power of competing descriptors. We recommend to use the l_1 norm for computing distances between DSR472 feature vectors.

In section 5.3, we verified that the standard PCA analysis can be applied to a set of n -dimensional feature vectors of real-valued components to reduce dimensionality of the feature vectors without significant loss in retrieval effectiveness. The reduction increases retrieval efficiency.

The invariance with respect to translation is achieved using the center of gravity of a model, while we recommend to scale the model by the average distance d_{avg} (3.25) of a point on the surface to the center of gravity of a model. In section 3.4, we give an original algorithm for approximating the value of d_{avg} . We also provide a means for fixing reflections around the coordinate hyper-planes (3.23). To attain rotation invariance, we use the CPCCA. Several authors (e.g., [36, 91]) object the use of the CPCCA, because it is not an ideal tool for fixing the orientation of a 3D-model. Statements such as “all descriptors relying upon the PCA show poor retrieval performance” are based on wrong intuitive assumptions and are not supported by experimental results. The descriptor defined in [36] is inherently invariant with respect to rotation, i.e., it is not necessary to apply the PCA in the normalization step. However, we consider that the overall performance is the most important parameter of quality of a descriptor. We compared our DSR472 descriptor to the descriptor proposed in [58], which is extracted using the original tools provided by the authors [108], and the results show that our approach is superior. Moreover, the DSR472 descriptor shows better performance than the descriptor based on exponentially decaying Euclidean distance transform (EDT) even for certain categories of models that are not well aligned in the canonical coordinate frame (e.g., categories “desk with hutch”, “TV”, “spider”, “handgun”, “desk lamp”, “piano”, “palm”, “tree”, etc. from the training database [108]). The rotation invariance of

the EDT descriptor is attained by applying a property of spherical harmonics (see property 4.1), i.e., by summing up squares of magnitudes of spherical harmonic coefficients in the same band. Hence, there is a trade-off, information contained in individual coefficients is sacrificed in order to achieve rotation invariance. Instead of using the property of spherical harmonics, we modified the approach presented in [57, 36, 58] by applied the CPCCA before the extraction and taking the magnitudes of spherical harmonics as components of the modified feature vector. We used three opportunities (figures 5.49, 5.55), and 5.56) to test two approaches for achieving the rotation invariance, the CPCCA vs. the property of spherical harmonics. The same conclusion is inferred after all three experiments: the descriptor relying upon the CPCCA possesses higher discriminant power than the descriptor relying upon the competing technique.

We invite you to visit our Web-based 3D model retrieval system, CCCC (see appendix), and test both descriptors for different categories of models. Tools (executables) for generating some of our 3D-shape feature vectors are available for download at the CCCC 3D-model search engine.

We recommend the DSR472 descriptor as currently the best 3D-shape descriptor, on average. However, this descriptor is not guaranteed to be the best for each category (class) of query objects (see figure 5.61). Therefore, we consider that introduction of a “query processor” would be meaningful. The query processor will have a task to estimate the most suitable descriptor for a given query. The estimation could be based on a measure of coherence of retrieved objects. The hybrid DSR472 is not necessarily the best possible hybrid descriptor, because it is created using heuristics, rather than using an exhaustive approach to find the best combination of parental descriptors. We consider that further improvements of the normalization step can be done by selecting a subset of all triangles of a mesh as the input for the CPCCA. Possible selection criterion might be the visibility of a triangle from viewpoints on an enclosing sphere or cube. The creation of a query processor, the finding of the best hybrid, and further modifications of the normalization step are left for future work. The CCCC search engine will also be improved further. We plan to provide a means to obtain a feedback from a user. The feedback might be used for rating classifications (ground truths) of 3D-models as well as for finding solutions for estimating the best descriptor for a given query model. Definition of new 3D-shape descriptors, which will outperform currently the best approaches, will remain the most challenging task.

Bibliography

- [1] J. Amanatides and K. Choi, “Ray Tracing Triangular Meshes,” in *Proc. of the Eighth Western Computer Graphics Symposium*, 1997, pp. 43–52.
- [2] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl, “3D Shape Histograms for Similarity Search and Classification in Spatial Databases,” in *Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99), Lecture Notes in Computer Science*, R. H. Güting, D. Papadias, and F. H. Lochovsky, Eds., Hong Kong, China, July 1999, vol. 1651, pp. 207–226, Springer Verlag.
- [3] J. Arvo and D. Kirk, “Fast ray tracing by ray classification,” in *Proc. SIGGRAPH 1987*, Anaheim, CA, July 1987, pp. 55–64, ACM SIGGRAPH.
- [4] J. Ashley, M. Flickner, J. L. Hafner, D. Lee, W. Niblack, and D. Petković, “The Query By Image Content (QBIC) System,” in *Proc. 1995 ACM SIGMOD*, M. J. Carey and D. A. Schneider, Eds. 1995, p. 475, ACM Press.
- [5] Autodesk, “DXF Reference,” <http://www.autodesk.com/techpubs/autocad/dxf/>.
- [6] D. Badouel, “An Efficient Ray-Polygon Intersection,” in *In Graphics Gems*, A. S. Glassner, Ed. 1990, pp. 390–393, Academic Press.
- [7] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, 1999.
- [8] G. Barequet and M. Sharir, “Filling Gaps in the Boundary of a Polyhedron,” *Computer-Aided Geometric Design*, **12**(2):207–229, March 1995.
- [9] R. Basri and D. Weinshall, “Distance Metric between 3D Models and 2D Images for Recognition and Classification,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **18**(4):465–470, 1996.
- [10] P. J. Besl and N. D. McKay, “A method for registration of 3D shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(2):239–256, February 1992.

- [11] D. Bespalov, A. Shokoufandeh, W. C. Regli, and W. Sun, “Scale-space representation of 3D models and topological matching,” in *Proc. of the eighth ACM symposium on Solid modeling and applications*, Seattle, Washington, USA, June 2003, pp. 208–215, ACM Press.
- [12] A. Bonnassie, F. Peyrin, and D. Attali, “Shape Description of Three-Dimensional Images Based on Medial Axis,” in *Proc. 2001 IEEE International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 2001, pp. 931–934.
- [13] P. Bunyk, A. E. Kaufman, and C. T. Silva, “Simple, Fast, and Robust Ray Casting of Irregular Grids,” in *Proc. Scientific Visualization Conference Dagstuhl ’97*, Dagstuhl, Germany, June 1997, pp. 30–36.
- [14] R. Carey, G. Bell, and C. Marrin, “Virtual Reality Modeling Language (VRML97),” <http://www.vrml.org/Specifications/VRML97>.
- [15] C. Carson, S. Belongie, H. Greenspan, and J. Malik, “Color- and Texture-Based Image Segmentation Using EM and Its Application to Image Querying and Classification,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2002, (to appear).
- [16] C. C. Carson, *Region-based image querying and classification*, Ph.D. thesis, University of California, Berkeley, 1999.
- [17] A. Celentano and S. Sabbadin, “Multiple Features Indexing in Image Retrieval Systems,” in *Proc. European Workshop on Content-Based Multimedia Indexing (CBMI 99)*, Toulouse, France, October 1999.
- [18] J.-M. Chung and N. Ohnishi, “Matching and recognition of planar shaped using medial axis properties,” in *First International Workshop on Multi-media Intelligent Storage and Retrieval Management (MISRM’99)*, Orlando, Florida, October 1999.
- [19] V. Cicirello and W. C. Regli, “Machining Feature-based Comparisons of Mechanical Parts,” in *Proc. SMI 2001*, Genova, Italy, May 2001, pp. 176–185.
- [20] L. Cinque, S. Levialdi, K. A. Olsen, and A. Pellican, “Color-Based Image Retrieval Using Spatial-Chromatic Histograms,” in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*. IEEE Computer Society, 1999, vol. II, pp. 969–973.
- [21] S. Cohen and L. Guibas, “The Earth Mover’s Distance under Transformation Sets,” in *Proc. of the 7th IEEE International Conference on Computer Vision (ICCV 1999) - Volume 2*, Corfu, Greece, September 1999, pp. 1076–1083, IEEE Computer Society.

- [22] F. Cutzu and M. J. Tarr, “The representation of three-dimensional object similarity in human vision,” in *SPIE Proceedings from Electronic Imaging: Human Vision and Electronic Imaging II*, San Jose, CA, February 1997, vol. 3016, pp. 460–471.
- [23] J. Davis, S. R. Marschner, M. Garr, and M. Levoy, “Filling holes in complex surfaces using volumetric diffusion,” in *Proc. the 1st International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'02)*, Padua, Italy, June 2002, pp. 428–438, IEEE Computer Society.
- [24] E. W. Dijkstra, “A Note on Two Problems in Connection with Graphs,” *Numerische Mathematik*, **1**:269–271, 1955.
- [25] S. Edelman, “Representation of Similarity in 3D Object Discrimination,” *Neural Computation*, **7**:407–422, 1995.
- [26] E. A. El-Kwae and M. R. Kabuka, “Efficient Content-Based Indexing of Large Image Databases,” *ACM Transactions on Information Systems*, **18**(2):171–210, April 2000.
- [27] M. Elad, A. Tal, and S. Ar, “Directed Search in a 3D Objects Database Using SVM,” Tech. Rep. HPL-2000-20R1, HP Laboratories Israel, August 2000.
- [28] M. Elad, A. Tal, and S. Ar, “Content Based Retrieval of VRML Objects - An Iterative and Interactive Approach,” in *Proc. of the Sixth Eurographics Workshop in Multimedia*, Manchester, UK, September 2001, pp. 97–108.
- [29] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petković, and W. Equitz, “Efficient and Effective Querying by Image Content,” *Journal of Intelligent Information Systems*, **3**(3-4):231–262, 1994.
- [30] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley, 1995.
- [31] J. Foote, “An Overview of Audio Information Retrieval,” *Multimedia Systems*, **7**(1):2–10, 1998.
- [32] J. T. Foote, “Content-based retrieval of music and audio,” in *Multimedia Storage and Archiving Systems II, Proceedings of SPIE*, 1997, vol. 3229, pp. 138–147.
- [33] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, “Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics,” in *Proc. SIGGRAPH 2000*, New Orleans, Louisiana, July 2000, pp. 249–254, ACM SIGGRAPH.
- [34] H. Fuchs, Z. M. Kedem, and B. F. Naylor, “On visible surface generation by a priori tree structures,” in *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, 1980, pp. 124–133.

- [35] T. Funkhouser, Personal Communication.
- [36] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs, “A Search Engine for 3D Models,” *ACM Transactions on Graphics*, **22**(1):83–105, January 2003.
- [37] M. Garland and P. S. Heckbert, “QSlim Simplification Software,” <http://www.cs.cmu.edu/afs/cs/user/garland/www/quadratics/qslim10.html>.
- [38] Y. Gdalyahu and D. Weinshall, “Flexible Syntactic Matching of Curves and its Application to Automatic Hierarchical Classification of Silhouettes,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **21**(12):1312–1328, 1999.
- [39] A. S. Glassner, “Space Subdivision for Fast Ray Tracing,” *IEEE Computer Graphics and Applications*, **4**(10):15–22, 1984.
- [40] A. S. Glassner, *An Introduction to Ray Tracing*, Academic Press, August 1989.
- [41] R. C. Gonzalez and R. E. Woods, *Digital image processing*, Addison Wesley, 2001.
- [42] S. Graphics, “Standard Template Library Programmer’s Guide,” <http://www.sgi.com/tech/stl/>.
- [43] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, “Efficient Color Histogram Indexing for Quadratic Form Distance Functions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(7):729–736, July 1995.
- [44] J. Hartman and J. Wernecke, *VRML 2.0 Handbook, The: Building Moving Worlds on the Web*, Addison Wesley, 1996.
- [45] D. Healy, D. Rockmore, P. Kostelec, and S. Moore, “FFTs for the 2-Sphere - Improvements and Variations,” Tech. Rep. TR2002-419, Department of Computer Science, Dartmouth College, Hanover, NH, 2002.
- [46] M. Heczko, D. A. Keim, D. Saupe, and D. V. Vranić, “Verfahren zur Ähnlichkeitssuche auf 3D-Objekten,” in *Proc. BTW 2001*, A. Heuer, F. Leymann, and D. Priebe, Eds., Oldenburg, Germany, March 2001, pp. 384–401, Informatik Aktuell, Springer Verlag.
- [47] M. Heczko, D. A. Keim, D. Saupe, and D. V. Vranić, “Verfahren zur Ähnlichkeitssuche auf 3D-Objekten,” *Datenbank-Spektrum Zeitschrift für Datenbanktechnologie*, **2**(2):54–63, February 2002.
- [48] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, “Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes,” in *Proc. SIGGRAPH 2001*, Los Angeles, CA, August 2001, pp. 203–212, ACM SIGGRAPH.

- [49] P. Hoogvorst, “Core Experiments on 3D Shape: Filtering of the 3D Mesh VRML Models,” ISO/IEC M6101, MPEG-7, Geneva, June 2000.
- [50] M. Irani, H. Sawhney, R. Kumar, and P. Anandan, “Interactive Content-Based Video Indexing and Browsing,” in *Proc. IEEE 1997 Workshop Multimedia Signal Processing (MMSP1997)*, Princeton, NJ, June 1997, pp. 313–318.
- [51] I. T. Jolliffe, *Principal component analysis*, Springer-Verlag, 1986.
- [52] S. B. Kang and K. Ikeuchi, “3-D Object Pose Determination Using Complex EGI,” Tech. Rep. CMU-RI-TR-90-18, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, October 1990.
- [53] A. E. Kaufman and E. Shimony, “3D scan-conversion algorithms for voxel-based graphics,” in *Proc. of the 1986 workshop on Interactive 3D graphics*, Chapel Hill, NC, 1987, pp. 45–75, ACM Press.
- [54] M. Kazhdan, Personal Communication.
- [55] M. Kazhdan, B. Chazelle, D. Dobkin, A. Finkelstein, and T. Funkhouser, “A Reflective Symmetry Descriptor,” in *Proc. European Conference on Computer Vision 2002 (ECCV 2002)*, Copenhagen, Denmark, May 2002, pp. 642–656.
- [56] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser, and S. Rusinkiewicz, “A Reflective Symmetry Descriptor for 3D Models,” *Algorithmica*, 2003, (to appear).
- [57] M. Kazhdan and T. Funkhouser, “Harmonic 3D Shape Matching,” in *SIGGRAPH 2002 Technical Sketches*, San Antonio, TX, July 2002, p. 191, ACM SIGGRAPH.
- [58] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, “Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors,” in *Proc. the Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP 2003)*, Aachen, Germany, June 2003, pp. 156–164, Eurographics Association.
- [59] P. Kostelec, “SpharmKit,” <http://www.cs.dartmouth.edu/~geelong/sphere/>.
- [60] P. Kostelec, D. Maslen, D. Rockmore, and D. Healy, “Computational Harmonic Analysis for Tensor Fields on the Two-Sphere,” *J. Computational Physics*, **162**:514–535, 2000.
- [61] L. J. Latecki and R. Lakämper, “Shape Similarity Measure Based on Correspondence of Visual Parts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(10):1185–1190, October 2000.

- [62] L. J. Latecki and R. Lakämper, “Application of Planar Shape Comparison to Object Retrieval in Image Databases,” *Pattern Recognition*, **35**(1):15–29, January 2002.
- [63] L. J. Latecki, R. Lakämper, and U. Eckhardt, “Shape Descriptors for Non-rigid Shapes with a Single Closed Contour,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2000)*, Hilton Head Island, South Carolina, June 2000.
- [64] A. Laurentini, “How far 3d shapes can be understood from 2d silhouettes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(2):188–195, February 1995.
- [65] G. Leifman, S. Katz, A. Tal, and R. Meir, “Signatures of 3D Models for Retrieval,” in *Proc. of the Fourth Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, Tel-Aviv, Israel, February 2003, pp. 159–163.
- [66] S. Z. Li, “Content-Based Audio Classification and Retrieval Using the Nearest Feature Line Method,” *IEEE Trans. on Speech and Audio Processing*, **8**(5):619–625, 2000.
- [67] P. Liepa, “Filling holes in meshes,” in *Proc. the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Aachen, Germany, June 2003, pp. 200–205, Eurographics Association.
- [68] Y. Liu and F. Dellaert, “A Classification Based Similarity Metric for 3D Image Retrieval,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR’98)*, Santa Barbara, CA, June 1998, pp. 800–805.
- [69] Y. Liu and F. Dellaert, “Classification-driven medical image retrieval,” in *Proc. DAPAR Image Understanding Workshop (IUW’98)*, November 1998, pp. 1261–1267.
- [70] Z. Liu, J. Huang, Y. Wang, and T. Chen, “Audio Feature Extraction and Analysis for Scene Classification,” in *Proc. IEEE 1997 Workshop Multimedia Signal Processing (MMSP1997)*, Princeton, NJ, June 1997, pp. 343–348.
- [71] Z. Liu, Y. Wang, and T. Chen, “Audio Feature Extraction and Analysis for Scene Classification,” *Journal of VLSI Signal Processing*, pp. 61–79, 1998.
- [72] J. Löfller, “Content-based Retrieval of 3D Models in Distributed Web Databases by Visual Shape Information,” in *Proc. IEEE International Conference on Information Visualization (IV2000)*, London, UK, July 2000, pp. 82–87.
- [73] G. Lu and A. Sajjanhar, “Region-based Shape Representation and Similarity Measure Suitable for Content-based Image Retrieval,” in *Multimedia Systems*, P. V. Rangan, Ed. 1999, vol. 7(2), pp. 165–174, ACM/Springer.

- [74] P. Lyman, H. R. Varian, J. Dunn, A. Strygin, and K. Swearingen, "How Much Information?," Univ. California at Berkeley, School of Information Management and Systems, <http://www.sims.berkeley.edu/how-much-info/>.
- [75] B. Mak and E. Barnard, "Phone Clustering using the Bhattacharyya Distance," in *Proc. of the Fourth International Conference on Spoken Language Processing (ICSLP 96) - Volume 4*, Philadelphia, PA, October 1996, pp. 2005–2008.
- [76] J. Malik, C. Carson, and S. Belongie, "Region-Based Image Retrieval," in *Proc. DAGM'99*, W. Förstner, J. Buhmann, A. Faber, and P. Faber, Eds., Bonn, Germany, September 1999, pp. 152–154, Springer Verlag.
- [77] D. McWherter, M. Peabody, W. C. Regli, and A. Shokoufandeh, "Transformation Invariant Shape Similarity Comparison of Solid Models," in *Proc. ASME Design Engineering Technical Conf., 6th Design for Manufacturing Conf. (DETC 2001/DFM-21191)*, Pittsburgh, PA, September 2001.
- [78] M. J. Mohlenkamp, "A Fast Transform for Spherical Harmonics," *The Journal of Fourier Analysis and Applications*, **5**(2/3):159–184, 1999.
- [79] A. Mojsilović and B. Rogowitz, "Capturing Image Semantics with Low-Level Descriptors," in *Proc. 2001 IEEE International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 2001, pp. 18–21.
- [80] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *ACM Journal of Graphics Tools*, **2**(1):21–28, 1997.
- [81] G. Mori, S. Belongie, and J. Malik, "Shape contexts enable efficient retrieval of similar shapes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001)*, December 2001, vol. 1, pp. 723–730.
- [82] MPEG-7 Audio Group, "Information Technology – Multimedia Content Description Interface – Part 4: Audio," ISO/IEC FCD 15938-4 / N4004, MPEG-7, Singapore, March 2001.
- [83] MPEG-7 DDL Group, "Information technology – Multimedia Content Description Interface – Part 2: Description Definition Language," ISO/IEC FCD 15938-2 / N4002, MPEG-7, Singapore, March 2001.
- [84] MPEG-7 Implementation Studies Group, "Information Technology – Multimedia Content Description Interface – part 6: Reference Software," ISO/IEC FCD 15938-6 / N4006, MPEG-7, Singapore, March 2001.
- [85] MPEG-7 Requirements Group, "MPEG-7 Requirements Document," V.15. ISO/IEC N4713, MPEG-7, Sydney, July 2001.
- [86] MPEG-7 Requirements Group, "Overview of the MPEG-7 Standard," V.8. ISO/IEC N4980, MPEG-7, Klagenfurt, July 2002.

- [87] MPEG-7 Video Group, “Description of Core Experiments for Motion and Shape,” ISO/IEC N3397, MPEG-7, Geneva, June 2000.
- [88] MPEG-7 Video Group, “Information Technology – Multimedia Content Description Interface – Part 3: Visual,” ISO/IEC FCD 15938–3 / N4062, MPEG-7, Singapore, March 2001.
- [89] MPEG-7 Video Group, “MPEG-7 Visual Part of eXperimentation Model,” V.9. ISO/IEC N3914, MPEG-7, Pisa, January 2001.
- [90] S. Newsam, B. Sumengen, and B. S. Manjunath, “Category-Based Image Retrieval,” in *Proc. 2001 IEEE International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 2001, pp. 596–599.
- [91] M. Novotni and R. Klein, “A Geometric Approach to 3D Object Comparison,” in *Proc. SMI 2001*, Genova, Italy, May 2001, pp. 167–175.
- [92] M. Novotni and R. Klein, “3D Zernike Descriptors for Content Based Shape Retrieval,” in *Proc. the eighth ACM symposium on Solid modeling and applications*, Seattle, Washington, USA, June 2003, pp. 216–225, ACM Press.
- [93] R. Ohbuchi, M. Nakazawa, and T. Takei, “Retrieving 3D Shapes Based On Their Appearance,” in *Proc. 5th ACM SIGMM Workshop on Multimedia Information Retrieval (MIR 2003)*, Berkeley, California, November 2003, pp. 39–46.
- [94] J.-R. Ohm, F. Bunjamin, W. Liebsch, B. Makai, K. Müller, A. Smolic, and D. Zier, “A multi-feature Description Scheme for image and video database retrieval,” in *Proc. IEEE Multimedia Signal Processing Workshop*, Copenhagen, Denmark, September 1999, IEEE Computer Society.
- [95] Y. Okada, “3D Model Database System by Hand Sketch Query,” in *Proc. 2002 IEEE International Conference on Multimedia (ICME 2002)*, Lausanne, Switzerland, August 2002, pp. 889–892.
- [96] P. Oliver, “Wotsit’s Format,” <http://www.wotsit.org/>.
- [97] R. Osada, T. Funkhouser, B. Chazelle, , and D. Dobkin, “Matching 3D Models with Shape Distributions,” in *Proc. SMI 2001*, Genova, Italy, May 2001, pp. 154–166.
- [98] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Shape Distributions,” *ACM Transactions on Graphics*, **21**(4):807–832, October 2002.
- [99] E. Paquet, “Nefertiti - Solutions for Multimedia Databases,” <http://www.cleopatra.nrc.ca/>.
- [100] E. Paquet, S. El-Hakim, A. Beraldin, and S. Peters, “The Virtual Museum: Virtualisation of Real Historical Environments and Artifacts and Three-Dimensional Shape-based Searching,” in *Proc. International Symposium on*

Virtual Reality and Augmented Architecture (VAA01), Dublin, Ireland, June 2001, pp. 183–193.

- [101] E. Paquet and M. Rioux, “Nefertiti: a Query by Content Software for Three-Dimensional Databases Management,” in *Proc. IEEE 3-D Digital Imaging and Modelling*, Ottawa, Canada, May 1997, pp. 345–352.
- [102] E. Paquet and M. Rioux, “The MPEG-7 Standard and the Content-based Management of Three-dimensional Data: A Case Study,” in *Proc. IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, June 1999, pp. 375–380.
- [103] E. Paquet and M. Rioux, “Nefertiti: a Query by Content System for Three-Dimensional Model and Image Databases Management,” *Image and Vision Computing*, **17**:157–166, 1999.
- [104] E. Paquet and M. Rioux, “Influence of pose on 3-D shape classification: Part II,” in *Proc. Digital Human Modeling for Design and Engineering Conference*, Arlington, VI, June 2001.
- [105] E. Paquet, M. Rioux, A. Murching, T. Naveen, and A. Tabatabai, “Description of Shape Information for 2-D and 3-D Objects,” *Signal Processing: Image Communication*, **16**(1-2):103–122, 2000.
- [106] M. Petrou and P. Bosdogianni, *Image Processing: The Fundamentals*, John Wiley & Sons, 1999.
- [107] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, “Surfels: Surface Elements as Rendering Primitives,” in *Proc. SIGGRAPH 2000*, New Orleans, Louisiana, July 2000, pp. 335–342, ACM SIGGRAPH.
- [108] Princeton Shape Retrieval and Analysis Group, “3D Model Search Engine,” <http://shape.cs.princeton.edu/>.
- [109] S. Ravela and R. Manmatha, “On computing global similarity in images,” in *Proc. IEEE Workshop on Applications of Computer Vision (WACV98)*, Princeton, NJ, 1998, pp. 82–87, IEEE Computer Society.
- [110] W. C. Regli, S. Lombeyda, and V. Cicirello, “National Design Repository,” Drexel University, <http://www.designrepository.org>.
- [111] V. Roth, “Content-Bbased Retrieval From Digital Video,” *Image and Vision Computing*, **17**(7):531–540, 1999.
- [112] Y. Rui, *Efficient Indexing, Browsing and Retrieval of Image/Video Content*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1998.

- [113] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, “Relevance Feedback: A Power Tool in Interactive Content-Based Image Retrieval,” *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Segmentation, Description, and Retrieval of Video Content*, **8**(5):644–655, September 1998.
- [114] L. A. A. Safadi and J. R. Getta, “Semantic Modeling for Video Content-Based Retrieval Systems,” in *Proceedings of the IEEE Australasian Computer Science Conference*, Canberra, Australia, January 2000, pp. 2–9.
- [115] T. Saito and J. ichiro Toriwaki, “New Algorithms for Euclidean Distance Transformation of an n-Dimensional Digitized Picture with Applications,” *Pattern Recognition*, **27**(11):1551–1565, 1994.
- [116] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.
- [117] D. Saupe and D. V. Vranić, “3D Model Retrieval with Spherical Harmonics and Moments,” in *Proc. DAGM 2001*, B. Radig and S. Floryczyk, Eds., Munich, Germany, September 2001, pp. 392–397, Springer Verlag.
- [118] P. Schröder and W. Sweldens, “Spherical Wavelets: Efficiently Representing Functions on the Sphere,” in *Proc. SIGGRAPH 95*, Los Angeles, CA, August 1995, pp. 161–172, ACM SIGGRAPH.
- [119] S. Servetto and K. R. T. Huang, “A Successively Refinable Wavelet-Based Representation for Content-Based Image Retrieval,” in *Proc. IEEE 1997 Workshop Multimedia Signal Processing (MMSP1997)*, Princeton, NJ, June 1997, pp. 325–330.
- [120] J. W. Shade, S. J. Gortler, L.-W. He, and R. Szeliski, “Layered Depth Images,” in *Proc. SIGGRAPH 1998*, Orlando, FL, July 1998, pp. 231–242, ACM SIGGRAPH.
- [121] S. Sigelkow, M. Schael, and H. Burkhardt, “SIMBA - Search IMages By Appearance,” in *Proc. DAGM 2001*, B. Radig and S. Floryczyk, Eds., Munich, Germany, September 2001, pp. 9–16, Springer Verlag, <http://simba.informatik.uni-freiburg.de/>.
- [122] J. R. Smith, Y.-C. Chang, and C.-S. Li, “Multi-Object Multi-Feature Content-Based Search using MPEG-7,” in *Proc. 2001 IEEE International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 2001, pp. 584–587.
- [123] S. Smoliar and H. Zhang, “Content-Based Video Indexing and Retrieval,” *IEEE Multimedia Magazine*, **1**(2):62–72, 1994.
- [124] M. T. Suzuki, “MMD Project, National Institute of Multimedia Education, Japan,” <http://www.nime.ac.jp/~motofumi/Ogden/>.

- [125] M. T. Suzuki, "A Web-based Retrieval System for 3D Polygonal Models," in *Proc. Joint 9th IFSA World Congress and 20th NAFIPS International Conference (IFSA/NAFIPS2001)*, Vancouver, Canada, July 2001, pp. 2271–2276.
- [126] M. T. Suzuki, T. Kato, and N. Otsu, "A Similarity Retrieval of 3D Polygonal Models Using Rotation Invariant Shape Descriptors," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC2000)*, Nashville, Tennessee, October 2000, pp. 2946–2952.
- [127] M. T. Suzuki and Y. Y. Sugimoto, "Personalized Retrieval from Material Color Databases for 3D Polygonal Models," in *Proc. 6th International Conference on Virtual Systems and Multimedia (VSMM2000)*, Gifu, Japan, Oktober 2000, pp. 577–584, IOS Press.
- [128] P. N. Swarztrauber and W. F. Spotz, "Generalized discrete spherical harmonic transforms," *Journal of Computational Physics*, **159**:213–230, 2000.
- [129] J. W. Tangelder and R. C. Veltkamp, "Polyhedral model retrieval using weighted point sets," Tech. Rep. UU-CS-2002-019, Utrecht University, the Netherlands, 1999.
- [130] Y. Tao, D. Papadias, and Q. Shen, "Continuous Nearest Neighbor Search," in *Proc. 28th Very Large Data Bases Conference (VLDB 2002)*, Hong Kong, August 2002, pp. 287–298, ACM SIGGRAPH.
- [131] P. Tsaparas, "Nearest Neighbor Search in Multidimensional Spaces," Tech. Rep. 319-02, Department of Computer Science, University of Toronto, Toronto, Canada, June 1999, Qualifying Depth Oral Report.
- [132] G. Turk, "The PLY File Format," http://www.cc.gatech.edu/projects/large_models/ply.html.
- [133] University of California, Berkeley, Digital Library Project, "Image Retrieval by Image Content," <http://galaxy.cs.berkeley.edu/photos/blobworld/>.
- [134] University of Massachusetts, Center for Intelligent Information Retrieval, "Image Retrieval Demo," <http://cowarie.cs.umass.edu/~demo/Demo.html>.
- [135] T. Ushijima, K. Hirobe, K. Sakai, L. Sun, and T. Watanabe, "STRIKE: A Movie Retrieval System Based on Event-Action Model," Tech. Rep. IPSJ SIGNotes DataBase System No.115 - 011, Department of Information Engineering, Graduate School of Engineering, Nagoya University, Nagoya, Japan, 2001.
- [136] N. Vasconcelos and M. Kunt, "Content-Based Retrieval from Image Databases: Current Solutions and Future Directions," in *Proc. 2001 IEEE*

- International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 2001, pp. 6–9.
- [137] N. Vasconcelos and A. Lippman, “Content-Based Pre-Indexed Video,” in *IEEE International Conference on Image Processing (ICIP 1997)*, Santa Barbara, CA, October 1997, pp. 542–545.
 - [138] R. C. Veltkamp and M. Hagedoorn, “State-of-the-art in shape matching,” Tech. Rep. UU-CS-2001-03, Utrecht University, the Netherlands, 1999.
 - [139] R. C. Veltkamp and M. Hagedoorn, “Shape Matching: Similarity Measures and Algorithms,” Tech. Rep. UU-CS-1999-27, Utrecht University, the Netherlands, 2001.
 - [140] D. V. Vranić, “CCCC: Content-based Classification of 3D-models by Capturing spatial Characteristics,” <http://www.informatik.uni-leipzig.de/~vranic/CCCC/>.
 - [141] D. V. Vranić, “An improvement of Ray-Based Shape Descriptor,” in *Proceedings of the 8. Leipziger Informatik-Tage (LIT'2M)*, W. Wittig and S. Paul, Eds., Leipzig, Germany, September 2000, pp. 55–58, HTWK Leipzig.
 - [142] D. V. Vranić, “An Improvement of Rotation Invariant 3D-Shape Descriptor Based on Functions on Concentric Spheres,” in *IEEE International Conference on Image Processing (ICIP 2003), Volume III*, Barcelona, Spain, September 2003, 757–760.
 - [143] D. V. Vranić and D. Saupe, “3D Model Retrieval,” in *Proc. Spring Conference on Computer Graphics and its Applications (SCCG2000)*, B. Falcidieno, Ed., Budmerice Manor, Slovakia, May 2000, pp. 89–93, Comenius University.
 - [144] D. V. Vranić and D. Saupe, “3D Shape Descriptor Based on 3D Fourier Transform,” in *Proc. EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001)*, K. Fazekas, Ed., Budapest, Hungary, September 2001, pp. 271–274, Scientific Association of Infocommunications - HTE.
 - [145] D. V. Vranić and D. Saupe, “A Feature Vector Approach for Retrieval of 3D Objects in the Context of MPEG-7,” in *Proc. International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging (ICAV3D 2001)*, V. Giagourta and M. Strintzis, Eds., Mykonos, Greece, May 2001, pp. 37–40.
 - [146] D. V. Vranić and D. Saupe, “Description of 3D-Shape Using a Complex Function on the Sphere,” in *Proc. 2002 IEEE International Conference on Multimedia (ICME 2002)*, Lausanne, Switzerland, August 2002, pp. 177–180.
 - [147] D. V. Vranić, D. Saupe, and J. Richter, “Tools for 3D-Object Retrieval: Karhunen-Loeve Transform and Spherical Harmonics,” in *Proc. IEEE 2001*

- Workshop Multimedia Signal Processing (MMSP2001)*, J.-L. Dugelay and K. Rose, Eds., Cannes, France, October 2001, pp. 293–298.
- [148] E. Weisstein, “Wolrd of Mathematics,” <http://mathworld.wolfram.com/>.
 - [149] E. Wold, T. Blum, D. Keislar, and J. Wheaton, “Content-Based Classification, Search and Retrieval of Audio,” *IEEE Multimedia Magazine*, **3**(3):27–36, 1996.
 - [150] J. H. Yi and D. M. Chelberg, “Model-Based 3D Object Recognition Using Bayesian Indexing,” *Computer Vision and Image Understanding*, **69**(1):87–105, January 1998.
 - [151] T. Zaharia and F. Prêteux, “Hough transform-based 3D mesh retrieval,” in *Proceedings SPIE Conference 4476 on Vision Geometry X*, San Diego, CA, August 2001, pp. 175–185.
 - [152] T. Zaharia and F. Prêteux, “Three-dimensional shape-based retrieval within the MPEG-7 framework,” in *Proceedings SPIE Conference 4304 on Nonlinear Image Processing and Pattern Analysis XII*, San Jose, CA, January 2001, pp. 133–145.
 - [153] C. Zhang and T. Chen, “Efficient Feature Extraction for 2D/3D Objects in Mesh Representation,” in *Proc. 2001 IEEE International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 2001, pp. 935–938.

Appendix: CCCC

Our Web-based retrieval system, called *CCCC* [140], serves as a proof-of-concept for our implemented methods and tools for content-based search for 3D-mesh models. The CCCC is also useful for obtaining a subjective impression about effectiveness of different descriptors. In this appendix, we give an overview of the on-line system, which is currently located at the following address:

[http://merkur01.inf.uni-konstanz.de/CCCC/.](http://merkur01.inf.uni-konstanz.de/CCCC/)

We assume that the URL of the CCCC 3D search engine will be changed in the future. However, we expect that it will be relatively easy to locate a new address using conventional search engines (e.g., *google.com*) or by visiting the original site [140]. For the implementation, we used C++, Perl, and JavaScript. The starting screen of the CCCC is shown in figure 6.1.

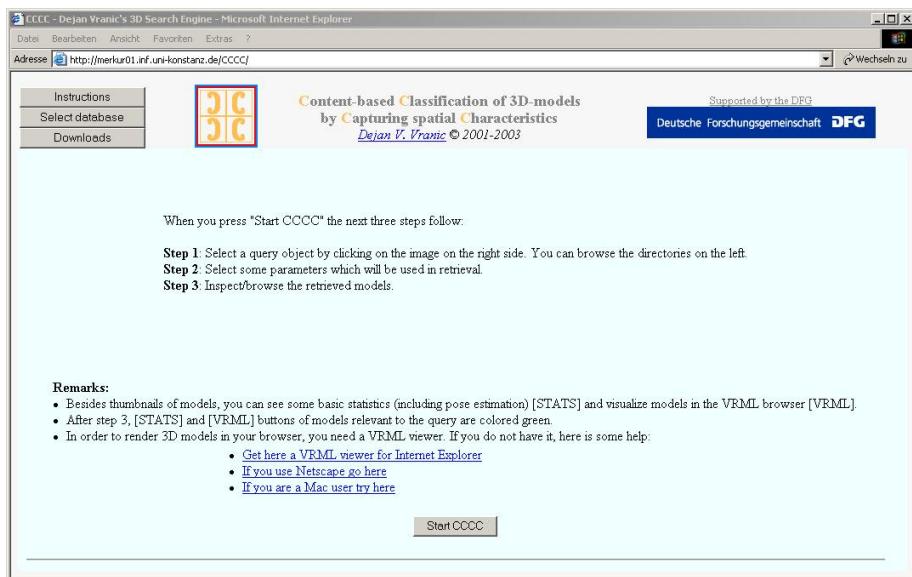


Figure 6.1: The starting screen.

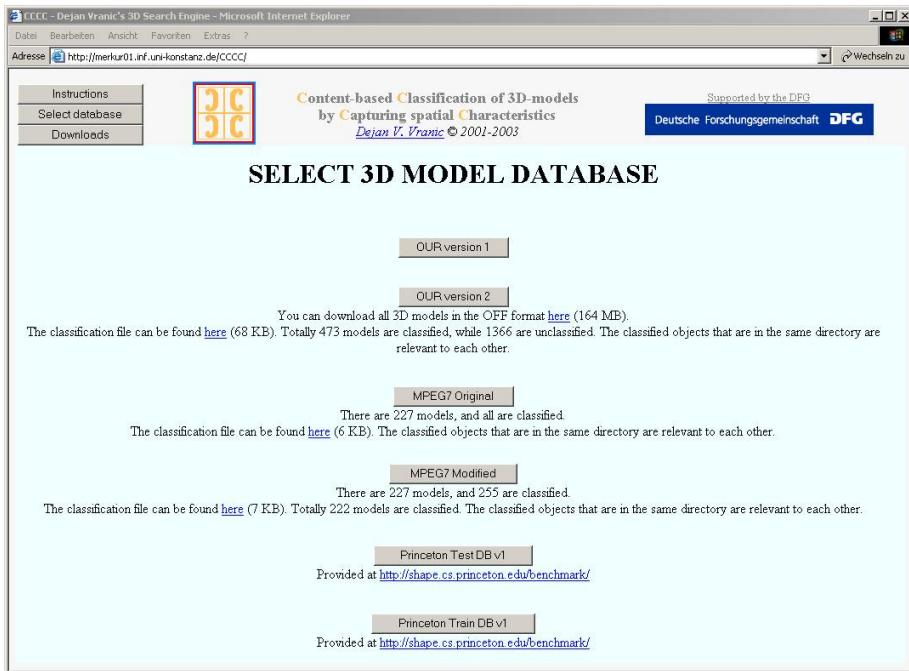


Figure 6.2: Selection of a 3D-model classification.

A user can select a classification of a 3D-model collection (figure 6.2). The classification serves as a ground truth. Currently, the six classifications that are described in section 5.1 are available for selection. Moreover, all the models that we collected on the Internet (mostly on www.3dcafe.com) are available for download. Information about the classification is attached as well. We stress that only models from the selected classification are considered in the subsequent steps. Since there is no classification that can be regarded as a unique ground truth, our goal is to gather as many as possible different categorizations of 3D-models. If an approach outperforms others for almost all categorizations, then we are more confident to declare which method is the best method. In the future, an interface will be created so that a user can alter the used classification. We consider that a new module in our system, which will process data obtained by the feedback, can be useful for ranking the classifications. For instance, in our reclassified collection we have a single category of models of humans, while both the training and the test databases contain two categories of models of humans, classified by poses ("human arms out" and "human walking"). We expect that users' feedback will help us determine if the models of humans with arms out should be considered relevant to the humans walking. Feedback from users might also be used for forming weighted concatenations of feature vectors, in order to improve the retrieval performance further.

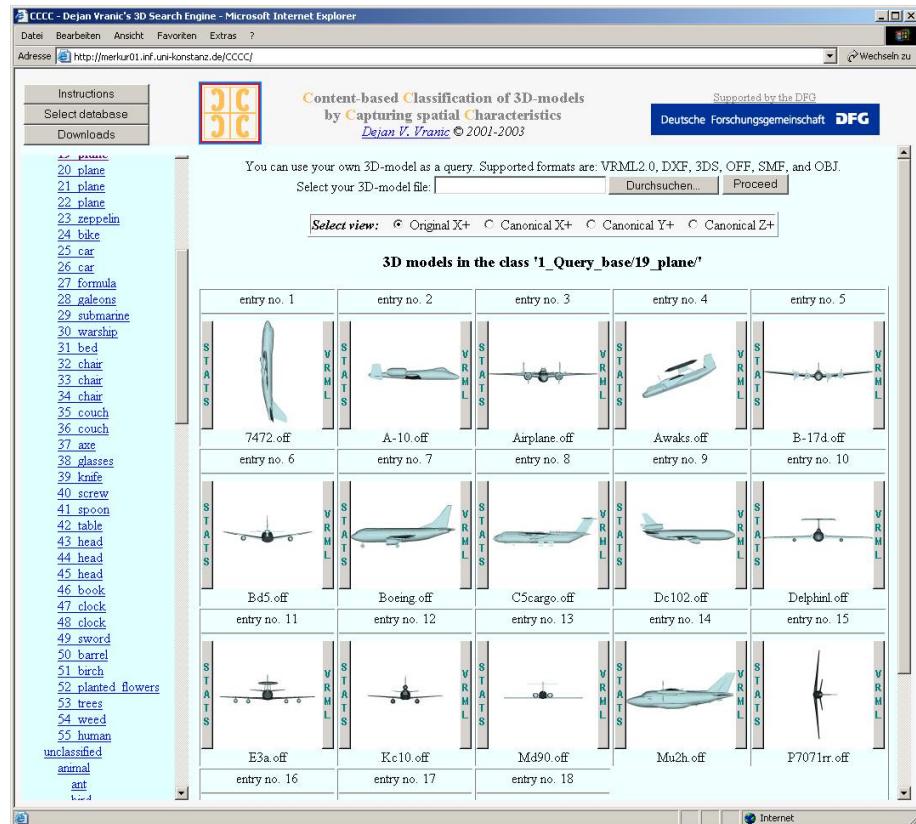


Figure 6.3: Browsing the selected 3D-model classification.

Browsing models from the selected classification is enabled. On the right-hand side of the screen (figure 6.3), a classification hierarchy is displayed. The models from the selected class are depicted using 2D thumbnail images. Initially, each model is visualized in the original orientation, so that the viewpoint is on the positive side of the x -axis, while the y -axis travels to the right. In order to obtain more information about a model, basic statistical data about the triangle mesh (e.g., number of vertices, number of faces, information about the closedness and orientability of the model, normalization parameters from section 3.4, etc.) can be displayed in a separate window (button "STATS"). Also, a model can be rendered in a VRML-viewer, if an appropriate plug-in is installed (button "VRML").

As a very useful option, we provided a possibility to visualize the orientations of the models in the canonical coordinate frame. A user can select one of four possible views: from the positive side of the x -axis of the original coordinate frame (before applying the CPCA from section 3.4) as well as from the positive side of each axis

in the canonical coordinate frame. In figure 6.4, all the thumbnail images visualize models in the canonical coordinate frame, viewed from the positive side of the z -axis, while the x -axis travels to the right. We stress that the canonical scale cannot be seen on the thumbnail images, which are generated using the extended bounding box (definition 4.3).

A query is selected simply by clicking on a thumbnail image. We provided another option for specifying the query, uploading from a local directory. Currently, the CCCC can accept the following 3D-file formats: VRML (Virtual Reality Modeling Language) [44, 14], DXF (Autodesk Drawing eXchange Format) [5], 3DS (3D Studio file format) [96], OFF (Object File format) [96], OBJ (Wavefront Object files) [96], and SMF (Simple Model Format) [37]. In the future, we will provide an option to draw 2D sketches of a 3D-object and use them as a query.

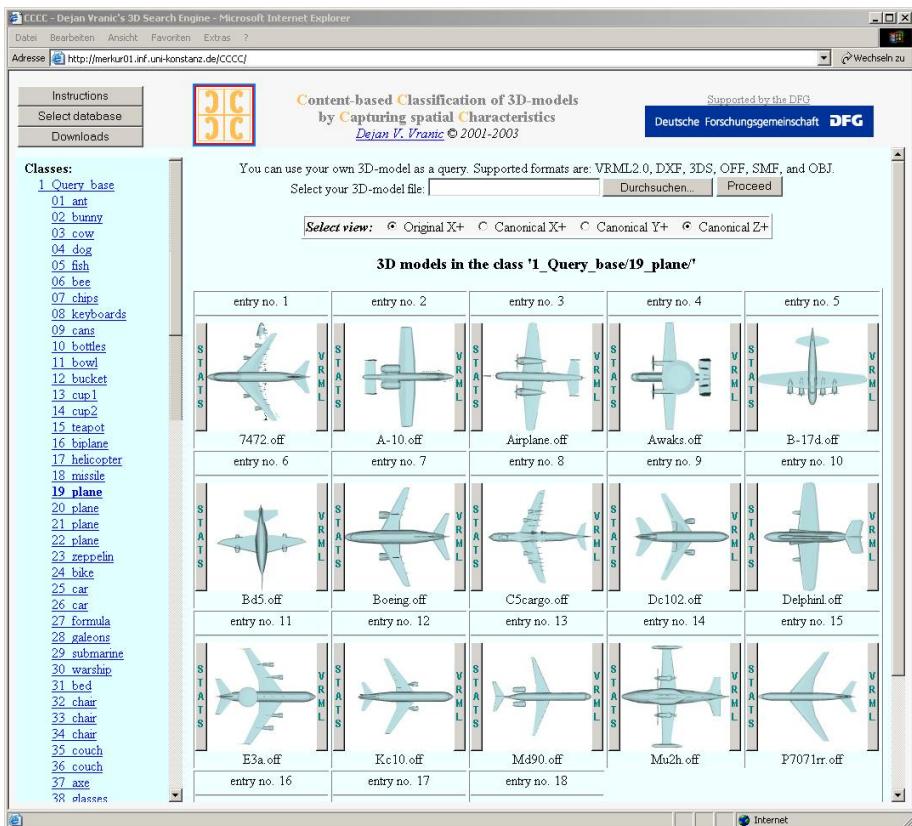


Figure 6.4: Visualization of normalized 3D-mesh models from the positive side of the z -axis, while the x -axis travels to the right.

After specifying the query model, by browsing or by uploading, the parameters for the search can be set (figure 6.5). We provide an initial setting, where our best hybrid descriptor (see section 5.4) and the l_1 distance metric (1.10) are selected by default. A user can change both settings and try different techniques, dimensions, and dissimilarity measures. In a very near future, we will provide a complete documentation about the offered 3D-shape descriptors as well as the references. Thus, we anticipate that the CCCC will have an educative role. The number of displayed thumbnail images per screen can also be set, because we consider that it is useful to have an option to reduce the number of shown images in the case of a slow Internet connection. After setting up the parameters, the similarity search is performed.

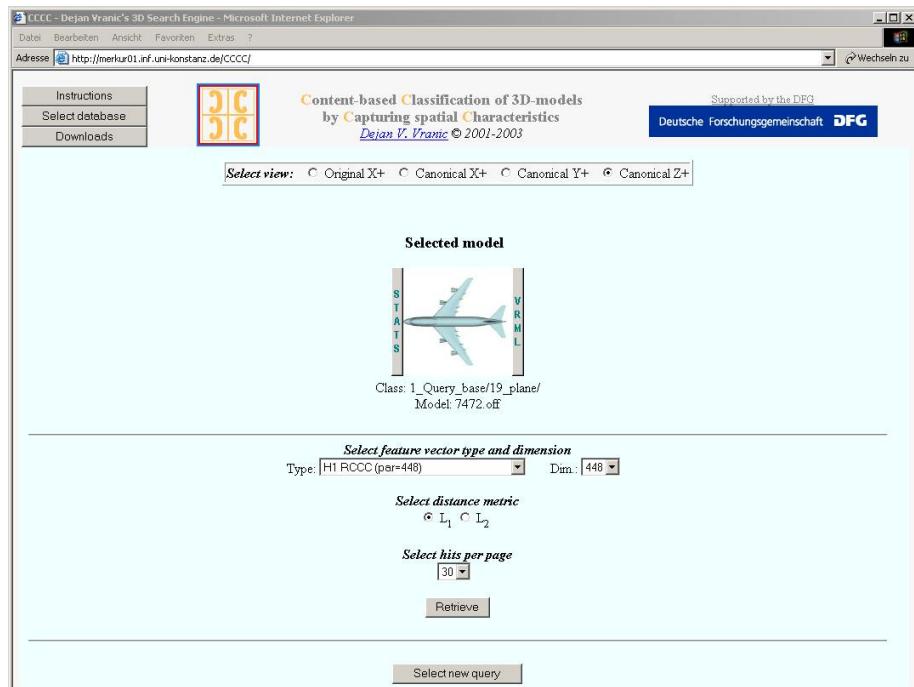


Figure 6.5: Setting up the parameters used for retrieval, feature vector type and dimension, the distance metric, and the number of displayed objects per screen.

In figure 6.7, the retrieval results for the specified query model, feature vector, dimension, and distance calculation are shown. The used descriptor is the hybrid feature vector of dimension 448, which is obtained as described in section 4.7. Objects that are relevant to the query model are denoted by the green color of the text on the buttons for displaying the statistics and rendering the model in a VRML-viewer. Conversely, the red color of the text denotes non-relevant objects. The match number 6 is considered as non-relevant in the used classification (O2 in table 5.3). We recall that the classification was not done by ourselves, but by our colleagues, without our influence.

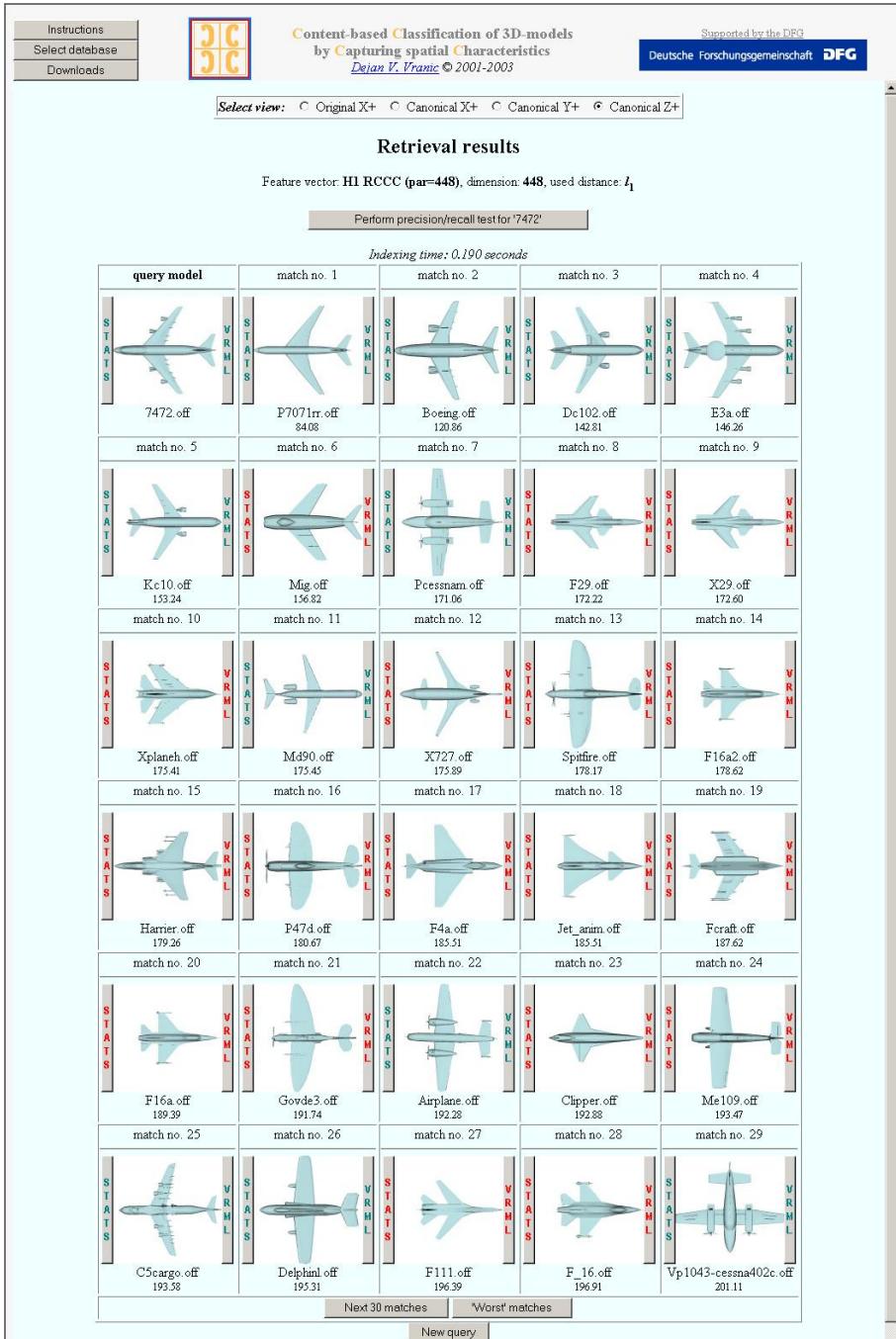


Figure 6.6: Inspection of retrieved results.

Thus, in figure 6.6, the first 29 retrieved models are airplanes, but some of them are not considered relevant to the query, by the selected ground-truth. A type of voting will be provided to a user, in order to alter the ground truth. Besides inspecting the retrieved models using thumbnail images of four different views, a user can obtain more information about the model by displaying a window with statistics as well as by rendering the model in a VRML-viewer (figure 6.7).

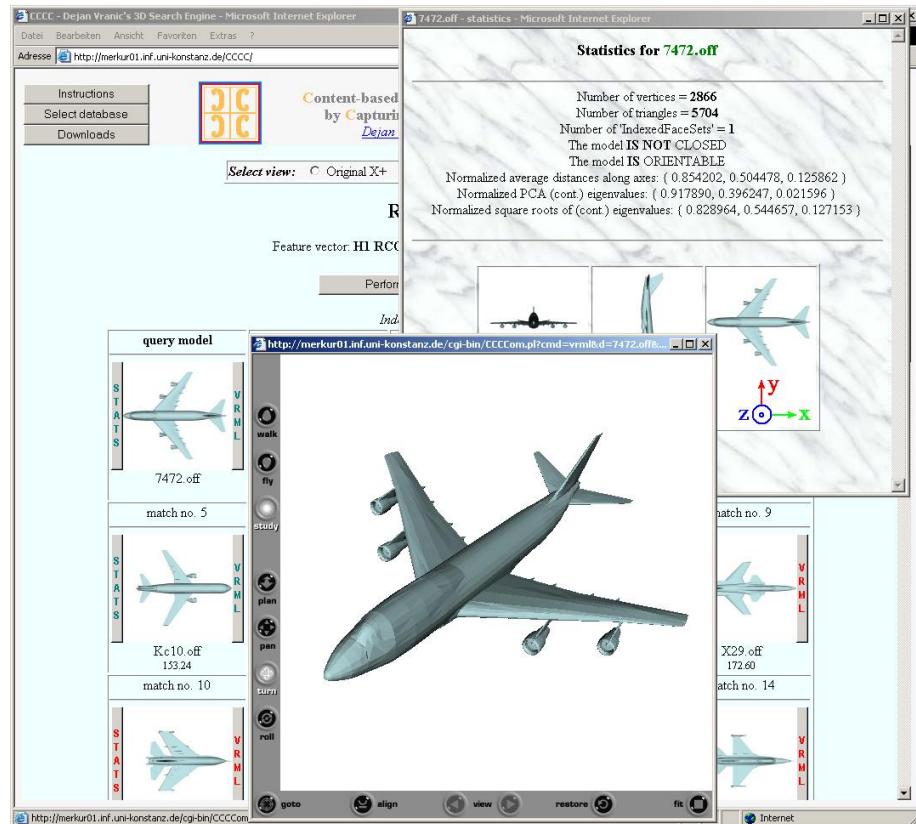


Figure 6.7: Showing of the window with basic statistics about the model and rendering using a VRML-viewer.

Interactive generation of precision-recall diagrams is provided, as well (figure 6.8). At the moment, two precision-recall curves can be generated, a curve for the query model and an curve curve for the class containing the query. Hence, a user can be informed if the class contains several models similar to the query or the query possesses an untypical 3D-shape for the class. In the future, the average precision-recall diagram of all queries, obtained by using the selected descriptor, dimension, and distance measure, will also be displayed. Thus, a used will be offered to compare descriptors at three levels: model-wise, category-wise, and global.

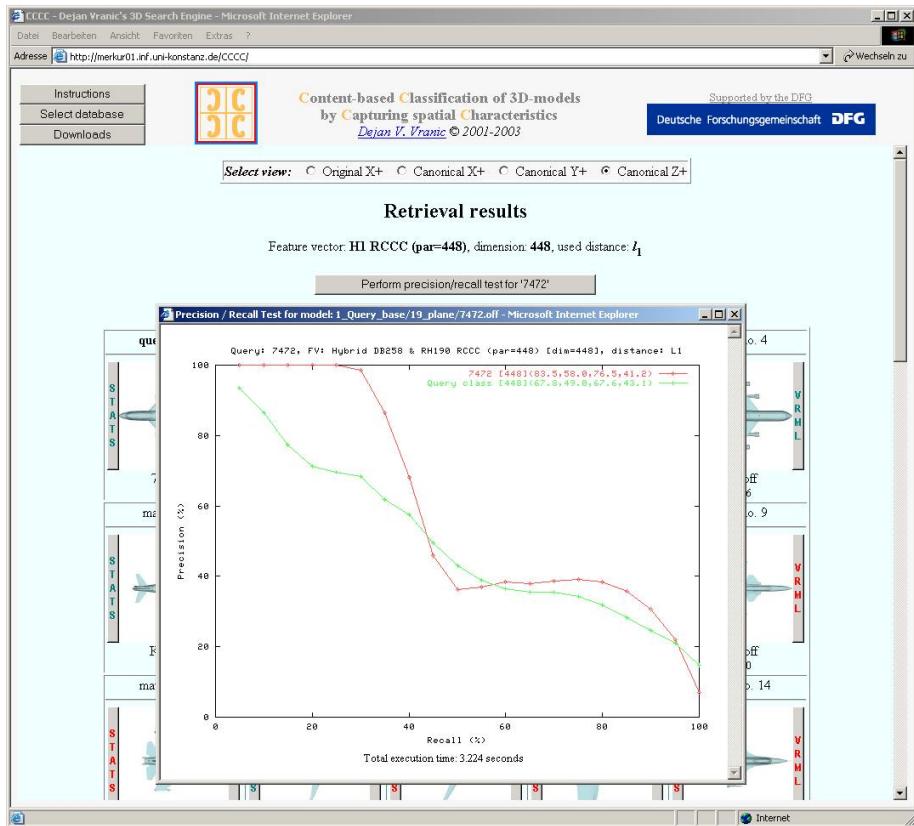


Figure 6.8: Precision-recall diagrams for the used query and the average curve for the class of 3D-models containing the query object.

Besides offering our classification of 3D-models, extraction tools for a selection of our descriptors are also available for download. We plan to update the download page periodically. Planned modifications of the CCCC 3D-model search engine include: implementation of three representations of retrieval results (textual, using thumbnail images, and using a 3D-viewer), incorporation of a Web-crawler (to enrich the set of collected 3D-models), complete documentation for a variety of techniques, creation of a personalized query interface (settings for a user will be stored and continuously updated), visualization of certain feature vectors (e.g., silhouette and depth buffer images), and implementation of an adaptive query processor.

We regard the CCCC as complementary to other 3D-model search engines such as the 3D-model search engine of the Princeton Shape Retrieval and Analysis Group [108], National Design Repository Drexel University [110], Nefertiti [99], and Ogden [124].

Biography



Dejan V. Vranić was born on July 2, 1970 in Zaječar, Serbia. He is son of Vukola and Živka Vranić, his hometown is Negotin, Serbia. He is married to Djilija Vranić and they have a daughter Aleksandra. He finished primary and secondary school in Negotin and received special acknowledgments for achieved results (Tesla's and Alas's diploma). Among numerous acknowledgments (over 30) for achieved results at different competitions in Mathematics and Physics at regional and state level the most important ones were two third prizes at Republic competitions of young mathematicians of Serbia. He received his Dipl.-Ing. in computer science (1994) and "Magistar" title (2000) from the University of Niš. He started his Ph.D. studies in October 1999 at the University of Leipzig, Germany.

He won several student prizes and scholarships, including a scholarship from the Ministry for Science and Technology in Serbia, a scholarship from the Institut für Feinwerktechnik, Technische Universität Wien, and a scholarship from the DFG (Deutsche Forschungsgemeinschaft).

He was associate researcher and university assistant at the Faculty of Electronic Engineering, University of Niš, associate researcher at the Institut für Feinwerktechnik, Technische Universität Wien, and associate researcher at the Institut für Informatik, Universität Leipzig. Currently, he is a research assistant at the Department of Computer and Information Science, University of Konstanz, Germany. He is author or co-author of more than 20 scientific papers.