SHAPE DESCRIPTION FOR 3D MODEL RETRIEVAL

Olga Symonova

February 2008

Technical Report # DISI-08-010

PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Shape Description for 3D Model Retrieval

Ol'ga Symonova

Advisor: Raffaele De Amicis

Fondazione Graphitech

March 2008

Abstract

Advanced design and scanning technologies facilitates the process of creation of 3D models. Still, designing a 3D model from scratch remains an effort and time consuming task. At the same time, the growth of the amount of 3D digital data available on the Internet and in corporate repositories gives the possibility to retrieve and reuse already existing models. The natural way to search for textual information is verbal description, whereas this can be ambiguous for digital shapes. Recently research in the field of shape representation and analysis proposes several solutions for shape description. Being exploited in different application domains like Industrial Design, Architecture, Medical Imaging and Game Industry shape descriptors might possess different properties important for shape retrieval. Some of them are robustness to noise, invariance to model's posture and position in space, ability to capture overall and partial similarity. At the same time it is important to have a compact shape descriptor to allow efficient retrieval process. Therefore there exists a tradeoff between completeness and compactness of a shape descriptor. The above-stated issues form a backbone of this PhD research which deals with description and retrieval of 3D models.

Differential topology and Morse theory are exploited to analyze and represent the topological structure of a model. By defining a Morse measuring function on the shape we are able to decompose the model into components representing critical and regular regions of the function. By iteratively bisecting saddle regions of the function and merging adjacent extrema and regular regions we maximize the regions feasible for the further shape analysis and minimize more complex saddle regions. Each component obtained after the segmentation and merging phases corresponds to a node in the Extended Reeb graph and the adjacency of two regions is reflected by the *presence of an edge between the corresponding nodes. Next, the shape of extrema and regular regions of the measuring function is analyzed with respect to several shape criteria. The shape analysis produces indices of shape classification which are used as the graph attributes.*

We propose to use a Hermitian matrix to represent the attributed topological graph in the retrieval process. By imposing several constraints on the elements of the Hermitian matrix we are able to mimic spectral properties of the Laplacian matrix. The Fiedler vector of such matrix is used for graph partitioning and further graph matching. Combination of the overall and partial graph matching increases the efficiency of the retrieval framework.

The proposed shape retrieval framework is examined using the benchmark of 3D models gathered from different domains and used for evaluation of other shape retrieval techniques. Performance evaluation measures prove the hight efficiency of our framework.

Finally, we propose a preliminary methodology for filling the gap between geometrical shape features and shape semantics. The proposed methodology is considered on the domain of furniture models but can be extended to other domains.

Keywords: 3D Shape Description, Shape Retrieval, Topological Structure, Shape Analysis, Graph Matching.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

The amount of digital data on the Internet grows exponentially over the last decade. Information freely available to each user includes different formats, text documents, graphical data, software applications etc. Any user in the network can access and reuse the data coming from the universe repository of the Internet. Search engines make easier the process of information seeking and retrieval. These engines take a textual query of a user and find the documents on the web with the highest number of occurrences of the provided query. Such a scheme for information retrieval is natural and efficient if a user seeks for textual documents.

Recent development of sophisticated technologies for creating and acquiring 3D data, such as separate 3D models and complete scenes of 3D models, resulted in the increasing amount of 3D graphical data available on the web as well as in corporate repositories. This facilitates the process of designing new 3D models by providing the possibility to reuse available data. While a textual query appears to be a natural description of textual documents in search and retrieval process, it proves to be inefficient for graphical data. Files containing 3D models cannot be found using a textual query. Textual information related to files with graphical data, such as the name of a file, author's annotations or the text on the page linking to the graphical data, is not sufficient for retrieval. Due to the use of different notations and languages, some files with desired 3D model will not match the query.

Recently research in the field of 3D shape retrieval has proposed a new approach to search for graphical information. Precisely, a sketch or an example 3D model can be used as a query for retrieval. 3D shape search engines analyze the shape of the query model using different mathematical techniques and compare it with the shape of the models in the repository. The models with the most similar shape are browsed to a user.

While analyzing the shape of a 3D model a search engine constructs a descriptor of the model. The descriptor is a compact and abstract shape representation used to compute similarity between two models. The descriptor can represent geometrical and topological features of the model. Depending on the way how a descriptor is constructed, it can be invariant to the position of a model in space, its posture, robust to noise and small shape inconsistence, it can represent a shape globally as well as integration of local features. Depending on the domain of 3D models some of the properties of the shape descriptor can be essential for efficient retrieval.

Once a query by sketch or by an example model is provided, a shape engine works only with geometrical representation of models, and any functional meaning is discharged. How to proceed from the geometry of a 3D model to its semantics, and how to learn about the geometry of a model knowing its functionality? To reveal the connection between the geometry of a shape and semantics intrinsically represented by the geometry, and to construct two-directional communication with 3D data is a challenge of the recent research in the field of advanced shape representation and retrieval.

1.2 Contributions

This thesis deals with the issues related to 3D model retrieval. Precisely, the contributions of the thesis are four-folded:

> We propose a new method to segment a shape of a 3D model which aims to extract the components of a model feasible for further shape analysis. Moreover, the proposed shape segmentation produces the components of a model which can be represented by the nodes of the topological Extended Reeb graph. The nodes corresponding to the neighbor components are connected by an edge in the graph. Such

topological graph can be used as a rough shape descriptor which is able to divide 3D models into topologically equivalent classes.

We propose a new scheme for local shape analysis of the components extracted on the previous segmentation step. Shape analysis is performed using three criteria, they are component's taper/enlargement, bending and average curvature. The shape criteria define shape classification, the indices of this classification can be used to attribute the topological graph representing the model.

We propose a new method to match attributed graphs. The graphs are topological shape descriptors attributed with indices produced after shape analysis which is performed locally on the components of a 3D model. The attributed graph is represented by a Hermitian matrix, which is constructed in the way to mimic the spectral behavior of a Laplacian. The eigenvector associated to the second smallest eigenvalue is used to partition the graphs into the set of meaningful subgraphs. The graph partition is done to cope with the problem of matching the graphs with different number of nodes. The similarity between the graphs is then measured as the distance between the second eigenvectors of the associated Hermitian matrices. The final measure of graph similarity is the combination of the overall and partial graph matching. Experimental results for 3D model retrieval prove high efficiency of the proposed graph matching and shape description methods.

We propose a new methodology to build the bond between the geometry and topology of the shape and its semantics. To construct this connection, 3D models should be considered and interpreted within a specified domain of models, because a shape can have different meaning depending on contest in which it is used. The proposed methodology considers the domain of furniture models. A simple shape descriptor is constructed for the models of the domain, and the vocabulary of the domain is used to map the descriptor to semantic labels. Using the ontology of the domain and semantic labels of a model, we can identify the class of the model and its relation with other models in the domain. Ontology of the domain allows two-directional interaction with 3D models, i.e. a user can query the system by providing the textual query which will be used to detect the desired class of 3D models (going from semantics to geometry), and conversely, when a user provides a 3D model and the system automatically detects the class of the model (going from geometry to semantics). The proposed methodology can be extended to other domains by exploiting different shape descriptors and constructing the vocabulary and ontology of the chosen domain.

1.3 Structure of the Thesis

The thesis is organized as follows:

In Chapter 2 we review the state of the art in the field of shape retrieval. First, we discuss the architecture and differences of several shape search engines available on the Internet. Then we overview the properties and types of shape descriptors used to represent a shape in retrieval process. We dwell on graph-based shape descriptors and methods for their matching. Finally, we list measures for evaluation of the efficiency of a shape retrieval process.

In Chapter 3 we give excerpts from differential topology and Morse theory, which form the theoretical background of the proposed shape segmentation and description techniques. We also discuss several functions used to map 3D shape to real-valued domain, and the techniques for construction topological graph-based shape descriptors. We conclude the chapter reviewing adjacency and Laplacian matrices used to represent a graph, and spectral properties of these matrices in the application to graph matching.

In Chapter 4 we describe the proposed framework for shape retrieval. First, we describe two approaches to construct the topological shape descriptor. Second, we give a scheme for local shape analyses for a segmented model. The result of the topological description and local shape analysis is the attributed graph based descriptor. Third, we describe the proposed graph partitioning and matching techniques. We conclude the chapter with the section, where we propose a new methodology for tying together the geometry and semantics of the shape.

In Chapter 5 we describe the results of the experiments of shape retrieval. We discuss the results of the proposed shape segmentation and analysis, and we also evaluate the efficiency of the whole retrieval system.

In Chapter 6 we review the work fulfilled in the thesis and highlight the directions of future work.

Chapter 2

State of the Art

An increasing number of 3D models on the Internet creates a potential to reuse this data. Having in mind a specific shape or an example model a user may query a 3D shape search engine by drawing a sketch or by uploading an example model. The search engine will analyze the query, extract important geometrical and topological features of the shape, understand its semantics and return to the user models with the similar shape or models with the same functional meaning. This scenario is a perfect solution of 3D shape retrieval. However, the problem of understanding a shape and, especially, the bond between the shape and intrinsic semantics is ambiguous and is usually studied with application to a restricted domain of models.

Shape analysis and understanding leads to construction of a shape descriptor, a specific structure (e.g. vector, matrix, graph structures) encoding geometrical, topological and possibly semantic characteristics of the shape. The purpose of a shape descriptor is to represent a shape in a simple and compact way in order to identify it as a member of a particular class of shapes. Thus a shape descriptor should encode those shape features which allow to distinguish the shape from other shapes in a database.

The process of shape retrieval consists of several phases. Once the architecture of a retrieval system is defined, the shape of models stored in a database or discovered as the result of crawling process is analyzed and their shape descriptors are extracted and stored along with the models. Given an input query model, the system analyzes its shape, constructs the shape descriptor and compare it with the shape descriptors of the models in the database. The result of such comparison is a set of real numbers which reflect the degree of similarity between the models. These numbers, allow to order the models in the database so that the user first browses the models most similar to the given query. Each of the phases of the depicted retrieval process are further reviewed in the sections of the current chapter.

2.1 Shape Search Engines

The growing amount of 3D data populating Internet and corporate databases requires a new technology to find, describe, index and store such information. Several prototypes of search engines of 3D models have been developed and now available on the Internet. Among them there are the search engine developed at Purdue University [3], Princeton 3D Model Search Engine [11], 3D Model Retrieval System based on LightField Descriptors [2]. All these search engines have similar architecture but differs by the ways how a query can be formed, by shape descriptors used to find similar models, by matching techniques for shape descriptors, and, finally, by the way the retrieval results are browsed to a user.

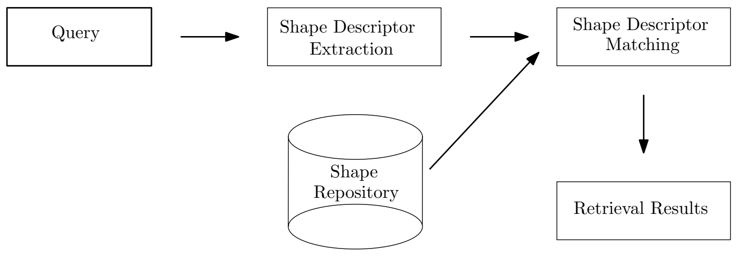


Figure 2.1: A scheme of a 3D model search engine.

Figure 2.1 represents a general scheme of a 3D shape search engine. On the first phase a query is formed. The query can be an example 3D model, 2D or 3D sketch. A query might contain keywords as well. To make use of textual queries the text on the page containing a link to the file with a 3D model is processed and matched against the query keywords. The name of the file of the model is also used in textual matching. On the second step the query is processed and a shape descriptor is constructed. On the third step the shape descriptor is matched against shape descriptors of models in the shape repository. These descriptors are usually extracted, indexed and stored on the preliminary off-line phase. The result of the matching of shape descriptors is usually a list of real numbers which indicate the similarity between corresponding models. Finally, when retrieved models are browsed to a user, they are ordered with respect to the computed matching rate.

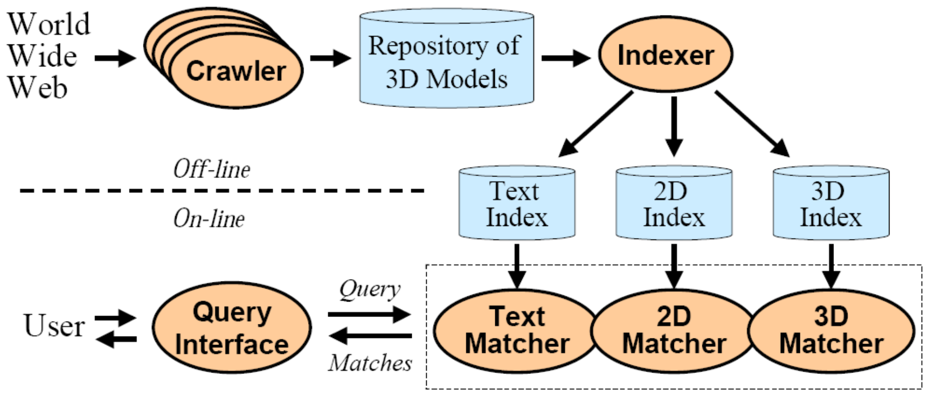


Figure 2.2: Organization of Princeton 3D Model Search Engine [40].

Figure 2.2 illustrates the architecture of Princeton 3D Model Search Engine [40]. The system execution consists of four steps: crawling, indexing, which are performed off-line, and querying and matching which are done on-line for each user query. The system allows textual queries, search by 2D and 3D sketches, and by an example model. The constructed shape descriptor is based on spherical harmonics of the shape function defined on the voxelized 3D model. The results are browsed to a user as an ordered list. A user has a possibility to refine a query by clicking the link "Find Similar Shape" which would lead to the multimodal query consisted of the entered keywords and of the shape information of a chosen model.

Figure 2.3 represents the architecture of the shape search engine developed at Purdue University. The search engine allows to query by an example model or by a sketch. The retrieved results are both 3D models and 2D legacy drawings. Additionally to a standard list of the retrieved models Purdue search engine proposes a new visualization paradigm where the models of the repository are located in 3D or 2D space. The coordinates of the retrieved models are computed by converting the similarity space into Euclidean space. Figure 2.4 illustrates the 3D navigation inter-

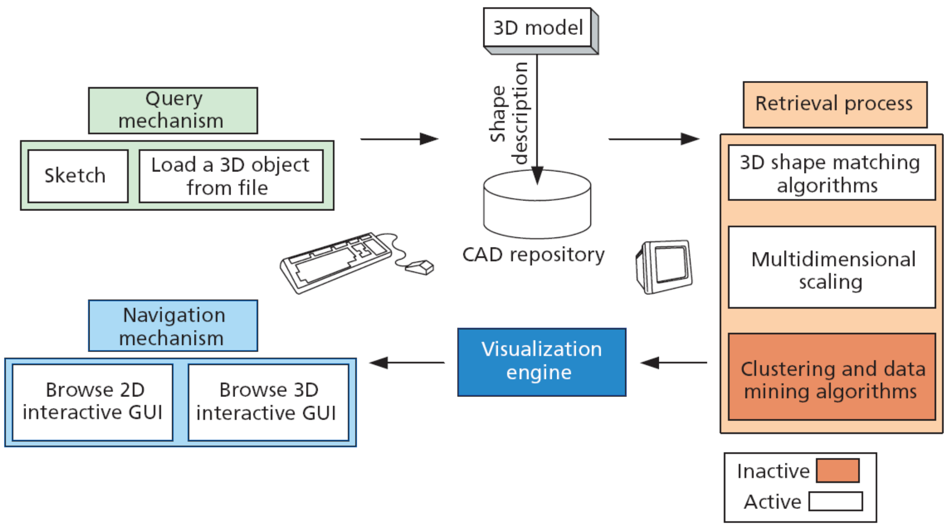


Figure 2.3: Organization of Purdue Navigation and Discovery Engine [74].

face.

The search engine developed at Technion University [56] proposes a user with a possibility to leave a relevance feedback regarding initial retrieval results by marking relevant items in the browsed list of retrieved models. Marked models participate further in refining the computation of shape similarity by separating relevant and irrelevant retrieved models. The refining procedure can take place until a user obtains satisfactory results. Figure 2.5 illustrates two phases of shape retrieval. On the first phase the retrieval is performed using only textual query, on the second phase the query was refined by marking relevant objects.

In the shape repository of AIM@SHAPE project [5] the models are organized using the metadata, i.e. ”knowledge about shape”, which is inserted inside the database together with a model by its owner. This metadata allows browsing models by different categories of shape representation, for example, boundary, raster or 3D animation shape representation. Moreover, this data is also exploited to organize the models in the repository using the developed ontology. Semantic search engine [6] use the ontology to retrieve models from the repository. Within AIM@SHAPE project there was also developed and now available online [4] a search engine based on the analysis of geometrical and topological shape features. This shape search engine takes as an input a query 3D model or a model from the

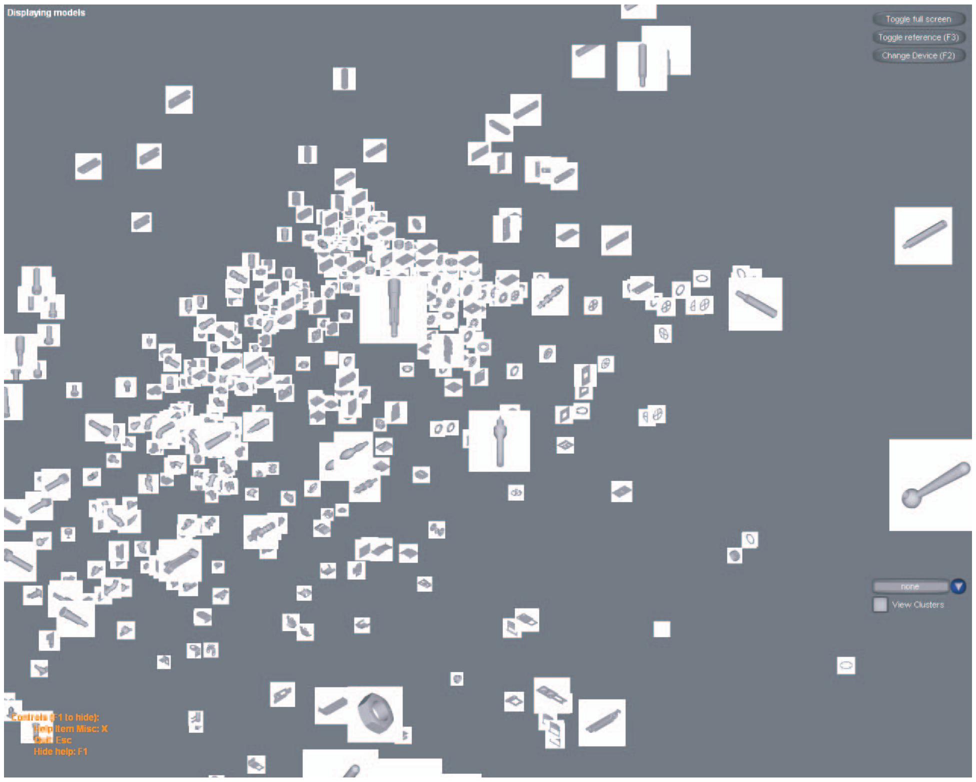
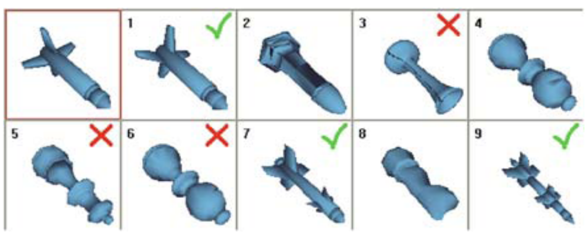


Figure 2.4: The 3D navigation interface. The models are positioned in space according to similarity distance. [74].

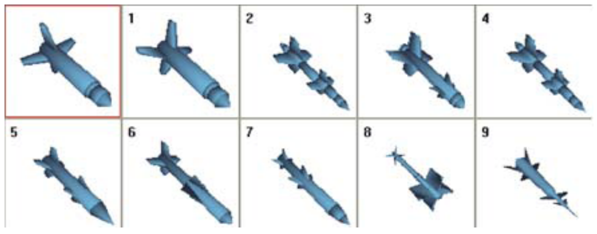
shape repository, and using Reeb graph based shape descriptor finds similar models (See Figure 2.6).

The efficiency of a retrieval system is usually evaluated using the measures from information theory. We discuss some of them in Section 2.3. The reviewed measures evaluate the performance of the whole system, where the number of relevant models is known in advance. The retrieval process, however, is composed from several complicate phases; some of them are construction of shape descriptor, matching of shape descriptors, representation of retrieval results. Efficiency of the whole system depends on the combination of the performances of each of the components. While designing a retrieval system, it is important to aim at increasing efficiency of each of the retrieval phases.

In the following section we review several shape descriptors, their properties and techniques for measuring similarity between them.



a) Intial search



b) Retrieval results after a relevance feedback iteration

Figure 2.5: Shape retrieval using relevance feedback [56].

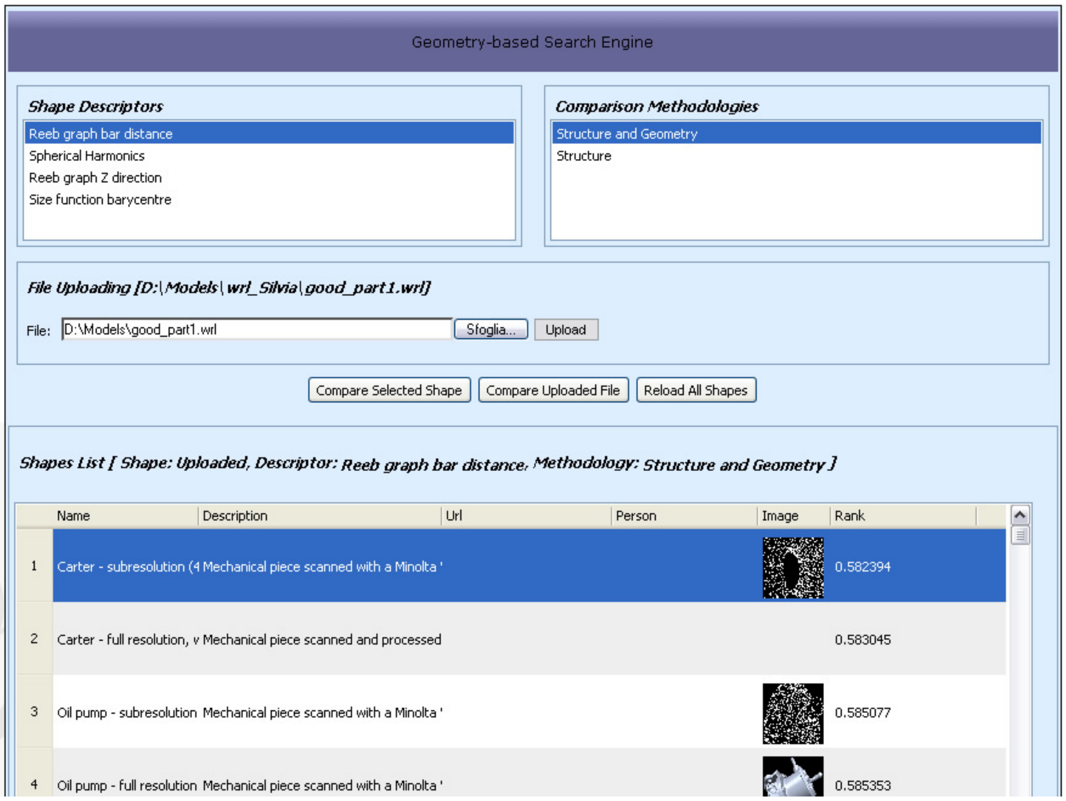
2.2 Shape Descriptors

Shape retrieval process consists of several steps, one of which is construction of a shape descriptor. Studies conducted in [64] demonstrate that the search of 3D shape using information stored in a shape descriptor significantly outperforms the search based on verbal description. This can be explained by the lack of textual information present in files of 3D models, by differences in language and notations. Conversely, a shape descriptor, or as it is also called, a shape signature, stores geometrical and/or topological information intrinsically represented by the shape of a 3D model.

The efficiency of a shape descriptor depends on its properties. To assure fast retrieval and indexing processes, a shape descriptor should have compact structure and allow fast matching. This property imposes a compromise between being descriptive and compact. The increase of descriptiveness of a shape signature usually leads to the growth of its size and as a consequence to a longer time needed to compute similarity between two signatures.

Often shape databases are collections of models which come from different designers. As the result the models can be located differently in space. A shape descriptor should be invariant to affine transformations. In the domain of articulated 3D models it is also important to have the same or similar shape descriptors for the models representing the same object but in different poses. Thus the descriptor should be posture invariant.

The majority of models which can be found on the Internet contain self-occlusions, holes, and noisy shape representation. Working with such models a shape descriptor should be robust to noise and small shape perturbations.


Geometry-based Search Engine

Shape Descriptors

- Reeb graph bar distance
- Spherical Harmonics
- Reeb graph Z direction
- Size function barycentre

Comparison Methodologies

- Structure and Geometry
- Structure

File Uploading [D:\Models\wrl_Silvia\good_part1.wrl]

File: D:\Models\good_part1.wrl

Shapes List [Shape: Uploaded, Descriptor: Reeb graph bar distance, Methodology: Structure and Geometry]

	Name	Description	Url	Person	Image	Rank
1	Carter - subresolution (4	Mechanical piece scanned with a Minolta '				0.582394
2	Carter - full resolution, v	Mechanical piece scanned and processed				0.583045
3	Oil pump - subresolution	Mechanical piece scanned with a Minolta '				0.585077
4	Oil pump - full resolution	Mechanical piece scanned with a Minolta '				0.585353

Figure 2.6: Geometry-based Shape Search Engine of AIM@SHAPE.

The other useful property which a shape descriptor can possess is the ability to describe the overall shape of a model as the composition of local shape descriptions. Such a descriptor will further provide a possibility to detect the overall as well as partial shape similarity.

Unfortunately, there is no a shape descriptor which possesses all the mentioned properties. Further in this section, we briefly review some shape descriptors which form state of the art of shape description and representation. We dwell on graph-based shape descriptors because they allow to combine topological and geometrical information of the shape and to detect overall and partial shape similarity. We use Extended Reeb Graph as a shape descriptor in our retrieval framework. Thus we will also review the applications of a Reeb graph in the fields of shape description and analysis.

Distribution Based Shape Descriptors use the distribution of a function defined on a shape as its signature. R. Osada et al. [69] investigated the use of four different functions calculated on points randomly sampled from the surface of a model. The $D2$ function of the distance between two random points gives the most discriminative shape description. In [43] the distribution of the geodesic distance function is used as a shape descriptor. The authors of [49] distinguish the distance values between two random points for which the connecting line segment passes inside a model, outside or both inside and outside the model. The authors also showed that the distribution based shape descriptor is able to discriminate gross shape differences, but tends to the normal distribution when a shape contains a lot of details.

Being computed on randomly sampled points and normalized, the distribution based shape descriptors have the advantage of being invariant to affine transformations and robust to noise and small shape inconsistencies. They are compact and fast to compute. Nevertheless, distribution based shape descriptors fail to distinguish the models with many shape features. Notice how similar the distributions of two left models on Figure 2.7.

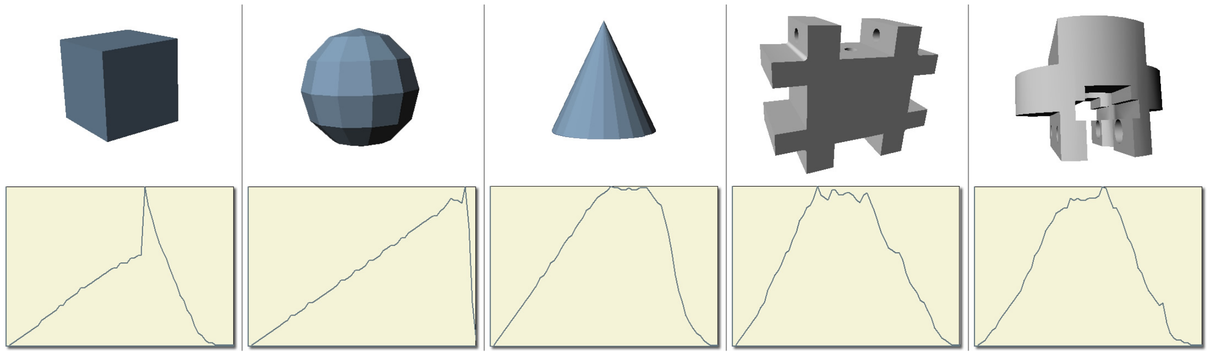


Figure 2.7: Distributions of $D2$ function for some 3D models. The more details are present in the shape of a model the less discriminative the distribution.

Similarity between the shape descriptors is simply Euclidean distance, or other distance measures like Minkowski L_N norms, Kolmogorov-Smirnov, Earth Mover's distances [69], computed between the vectors of the shape distributions of two models.

Transformation Based shape descriptors are descriptors which convert the function defined on the shape to frequency domain. These descriptors are usually 3D feature vectors.

The transformations that are used most often are Fourier Transform [95], decomposition into Spherical Harmonics [52], Angular Radial Transform [77], Zernike moments [68].

A general scheme for construction of a transform-based shape descriptor is the following. First, a 3D model is normalized in order to achieve invariance of the shape descriptor to affine transformations. The shape descriptor proposed in [52] is invariant to rotation due to the properties of spherical harmonics. Descriptors based on Fourier transformation require a preliminary alignment which is done through Principal Component Analysis [95]. Invariance to translation and scale is achieved by bringing a model into the origin of the reference frame and scaling it to the unit size.

Second, the shape is discretized inside voxel grid. The shape function is computed for the voxels composing the model. Voxelizing the model before construction of the shape descriptor make the descriptor more robust to noise and small shape impediments.

Finally, the discrete shape function is transformed into frequency domain. The shape descriptor is a vector composed of a certain number of low frequencies. Figure 2.8 shows the process of construction of a shape descriptor based on spherical harmonics.

Low frequencies of transformation based descriptors encode gross shape information whereas higher frequencies store detailed shape characteristics. Such multiscale particularity of transformation frequencies allow to measure similarity between two models at different scales. However, transformation and distribution based shape descriptors are able to reflect only the overall shape similarity and do not capture partial resemblance of models.

To find similarity between transformation based shape descriptors the L_1 and L_2 norms are usually computed between two descriptors [52, 95, 77, 100].

LightFiled shape descriptor is a view-based descriptor. Instead of analysis of the shape of a 3D model, a LightField shape descriptor represents a composition of descriptors of 2D image.

To construct the LightField shape descriptor proposed in [27], a 3D model is first placed to the origin of the system and scaled so that maximum length along one of the axes is one. 10 images of the model are taken from the vertices of a regular dodecahedron enclosing the model. The cameras used to take images should be distributed uniformly, and their

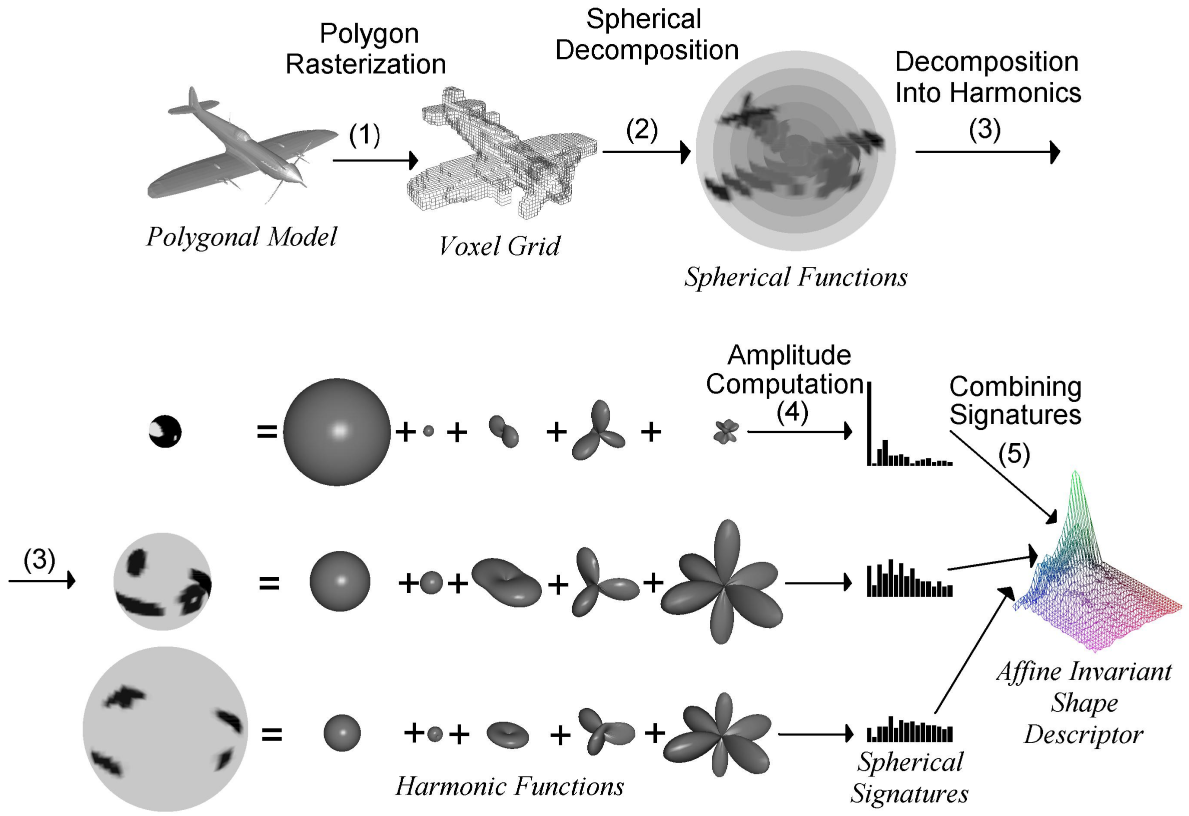


Figure 2.8: Computing the harmonic shape representation [52].

position is switched 60 times to find the most similar views. Moreover, 10 different rotations of the dodecahedron are considered in order to capture position difference between two 3D models. Zernike and Fourier rotation invariant descriptors are further extracted from obtained images of the model. Finally, the LightField shape descriptor of a 3D model is the vector of coefficients of Zernike and Fourier transformations of the images. Figure 2.9 illustrates the process of construction of the descriptor.

The LightField shape descriptor provides the best retrieval performance. Based on image description it is robust to noise and shape inconsistencies. Invariance to rotation, not exact but with small error, is achieved by choosing the most similar views among the set of images obtained after rotating the dodecahedron and switching camera system. However, the descriptor is not posture invariant and is not able to capture partial similarity of models.

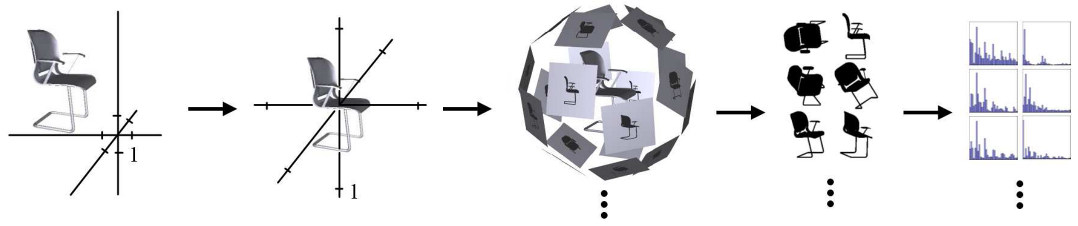


Figure 2.9: Computing the LightField descriptor of a 3D model [27].

The similarity between models represented by LightField Descriptors is measured as the smallest distance between the vectors composed of Zernike and Fourier coefficients for all obtained sets of views.

2.2.1 Graph Based Shape Descriptors

The majority of shape descriptors store information of the whole shape of a model and are not able to detach local shape details. Such descriptors are used to evaluate overall similarity between two models. In case when one model is a part of another model, shape descriptors often fail to detect their partial similarity. Figure 2.10 shows the cases of overall and partial shape similarity. Representation of a model by a graph-based shape descriptor allows to reveal partial similarity between models through detection of graph (sub)isomorphism.

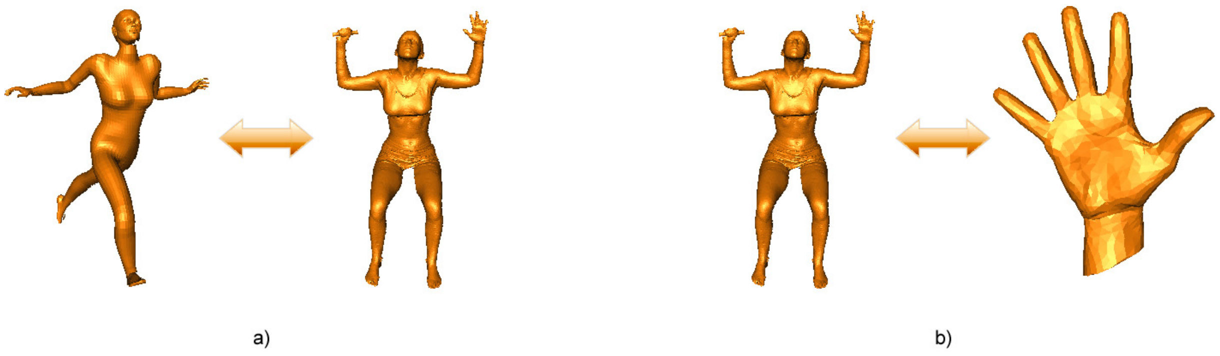


Figure 2.10: Overall (a) and partial (b) shape similarity.

Graph based shape descriptors encode both topology and geometry of a model in a compact way. The structure of the graph represents the topology of the model, and geometrical characteristics of the shape can be stored as attributes of nodes and edges of the graph. The advantage of efficient combination of topological and geometrical shape information pertains to graph-based shape descriptor. There exist several shape descriptors of a vector form, where some of vector components represent topological shape characteristics [56, 28], i.e. genus, number of connected components. However such descriptors do not reveal the structure and corresponding shape geometry.

Graph-based shape descriptors correspond to the natural perception of real solid objects by humans. Looking at a new object we perceive the shape of each part and the way how the parts are connected to each other. Thus, we perceive not only the substance of each part, but their integration. In a graph-based shape descriptor a node usually represents the substance of a part of an object, and the whole graph structure expresses relations between the parts.

On the other hand, matching graph based shape descriptors corresponds to (sub)graph isomorphism detection, which is a known NP-complete problem. Several heuristics were proposed to reduce the complexity of the problem of graph matching. Usually, information about a described shape is used to select nodes and edges in the graphs under comparison. Such selection results in an approximate but easier solution of the graph matching problem.

Medial Axis for shape description. Probably the most popular way to encode and represent compactly a shape is Medial Axis Transform (MAT). Medial Axis (MA) was first defined by H. Blum in 1967 in [24]. The author made analogy of MA and the place of extinguishing of grassfire fronts, where the fire is set on the boundary of an object. Mathematically, MA is defined as the loci of the centers of maximal balls within an object, where a maximal ball is the ball that is not contained in any other ball enclosed in the object. MAT is Medial Axis together with the value of the radius of the maximum ball corresponding to each point of the MA.

There exist three main approaches to construct Medial Axis. The first one is tracing approach [51, 29, 80]. Starting from a convex vertex on the surface of an object the method proceeds by sweeping along the incident seam until a junction point of MA is reached. Sweeping is performed using a tracing step and tracing direction, approximated representation of seams of MA is computed during sweeping. Once the junction point is reached, the sweeping procedure is repeated recursively for each of non visited adjacent seams.

The second approach uses Voronoi diagrams [12]. Since MA is a subset of vertices of Voronoi diagram of points sampled on the surface of an object, the approach first constructs Voronoi diagram of the object and then selects the subset of Voronoi vertices for medial axis approximation.

The third approach, also called homotopic thinning, discretizes an object on a voxel grid and then using distance transform consequently removes boundary voxels. While removing the voxels, or in other words, thinning the object, the check on the preservation of the object topology is done. Other control performed during object thinning is preservation of one-voxel thin and medial skeleton [14]. However, there exist a large number of homotopic thinning methods that produce non medial skeletons.

MA provides several advantages for the use in shape representation. First, it is topologically equivalent to the object itself, so it can be used to represent the topology of a model. As the consequence of this property, MA is invariant to the position of the model in space. Second, MA has compact structure. Third, the most important advantage is that it is possible to reconstruct an object from its MA by using distance transform.

MA is widely used in image representation, shape deformation and animation. However, its application to 3D shape representation is more difficult since MA is not any more the collection of points and curves but it can contain surface patches as well. The most known shortcoming of MA is that it is unstable to noise. Small protrusions on the boundary of an object can generate big changes in the MA. Moreover, changes of one vertex on the object boundary can lead to changes of MA in more than one place. Figure 2.11 illustrates the MA of an object and its changes due to small boundary perturbations.

To cope with instability to small boundary perturbations several values are associated to MA, which measure the substance and conductance of the components of the MA [51]. The components of the MA with higher substance value have more importance in the graph. The other way to reduce the instability of the MA is graph pruning [15], where more studies on graph structure should be done to decide which branches are more

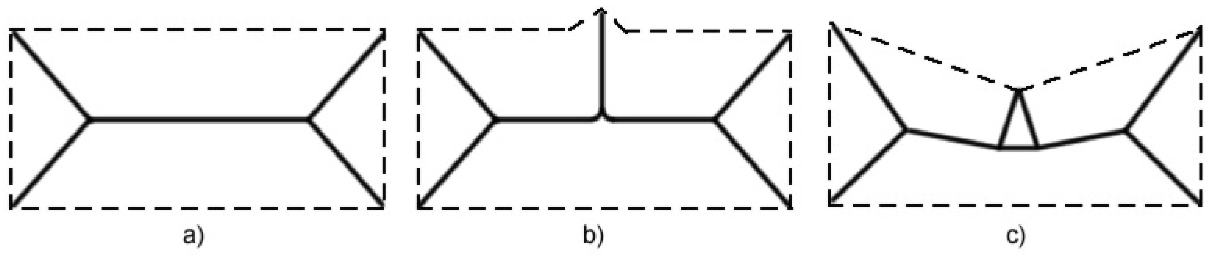


Figure 2.11: A shape (in dotted line) and its MA (in bold line). MA instability with respect to small boundary alterations, (b) a small protrusion produces large difference in MA; (c) boundary alteration in one place may change the MA in several places [51].

important and which are to be pruned.

Shock graphs are defined as the locus of singularities (i.e. shocks) of Blum's grassfire transformation. The position of shocks defines Medial Axis, the radius variation around a shock defines the label of the corresponding graph component. Shock graphs are directed acyclic graphs where the direction of arcs can be defined by the time of shock formation [84]. Figure 2.12 shows types of shocks used in [84] and an example shock graph.

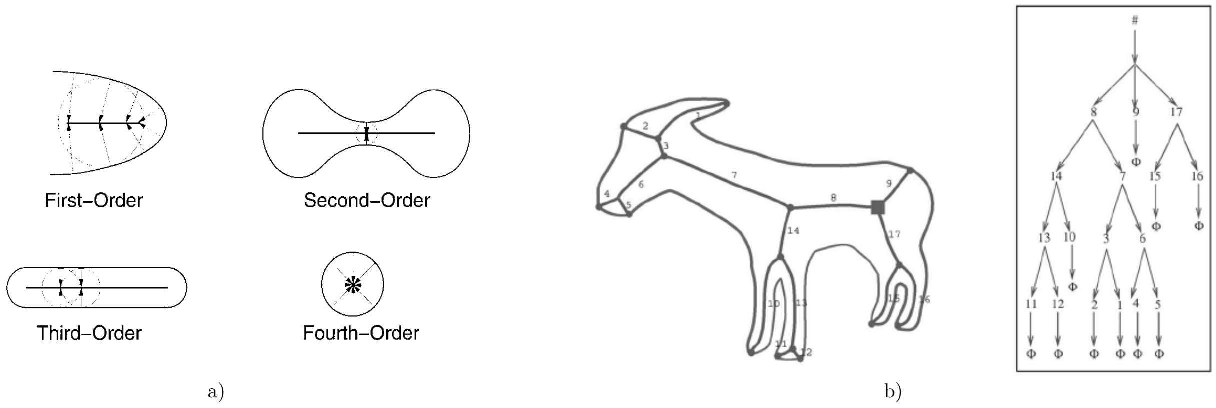


Figure 2.12: Shock graphs. a)Shock types [84]. b) a shock graph with arc direction opposite to the Blum's grassfire direction, image is taken from [78]

The authors of [78] use different shock classification which implies the directions of arcs in the graph.

Shock graphs inherit all the properties of Medial Axis and are further attributed by shock types. The edges of the graphs are directed arcs. These properties make shock graphs more amiable for shape description, recognition and matching.

2.2.2 Reeb Graphs

During last years Reeb graphs gain more and more popularity in the field of shape description and analysis which can be proved by an increasing number of research articles dealing with applications of Reeb graphs in different fields. First defined by French mathematician G.Reeb in 1946 [76], Reeb graphs were rediscovered in application to computer graphics in 1991 by Shinagawa et al [82]. A Reeb graph is used to represent the topological structure of the manifold of a shape. To this end a measuring function is defined in the manifold and Morse theory is exploited to study topology of the manifold through analysis of the measuring function. Morse theory provides powerful instruments to examine the changes in the topology of a manifold by tracing the evolution of critical points of a function defined on it. Therefore, the nodes of the Reeb graph represent the critical points of the function, and their configuration is reflected by the edges in the graph. We give theoretical definition of the Reeb graph together with some useful theorems from Morse theory in Section 3.1. The measuring function should obey several constraints which are also discussed in Section 3.1. Obviously, the structure of a Reeb graph depends on the chosen measuring function. For an example, the configuration of critical points of the height function differs from those of the width function, thus leading to different Reeb graphs of the same 3D model (see Figure 2.13). Several measuring functions used for Reeb graph construction are reviewed in Section 3.2.

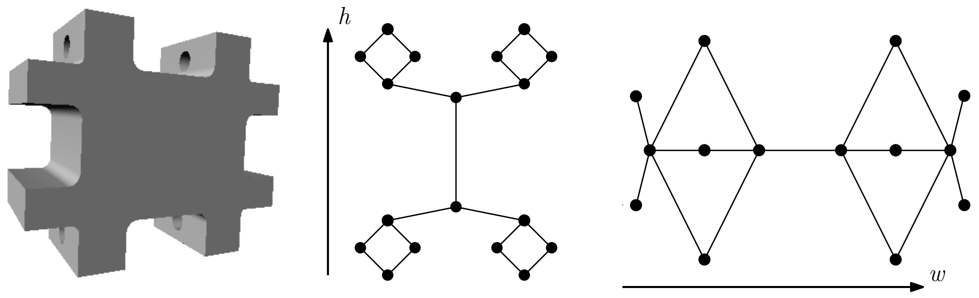


Figure 2.13: Reeb graphs with height and width measuring functions.

There are three main approaches to construct Reeb graphs. The first approach proposed in [46] is based on fine-to-coarse decomposition of the manifold of a shape into connected components. The second approach [21] analyses the evolution of levelsets of the measuring function defined on a shape. In [71] the authors propose a fast and robust way to construct a Reeb graph by exploiting efficiently the correspondence between data structures of the shape and the graph. In Section 3.3 we discuss all the three approaches in details.

Reeb graphs are widely used for shape encoding and description, shape matching and retrieval [46, 21, 19, 92, 90]. There is a number of works where Reeb graphs are used for visualization of complex scientific data [89, 88, 70, 34]. Reeb graphs are also exploited for shape segmentation [18, 99], topology repair [98], surface parameterization [72], texture mapping [101] and shape reconstruction [16]. In [34] the authors explore time-varying Reeb graphs, and in [50] the graphs are used for topology morphing.

2.2.3 Graph Matching

In the current section we review several method for matching graph-based shape descriptors. The most important property of graph-based shape descriptors is the ability to separate shape features and to represent them structurally. Consequently, these descriptors are used to detect partial shape similarity which is equivalent to the problem of subgraph isomorphism detection. The latter is a known NP-complete problem. Several heuristics were proposed in literature to reduce the complexity of the problem by exploiting additional information stored in the graph-based shape descriptors.

One of the methods used for detection of subgraph isomorphism finds Maximum Common Subgraph MCS_{G_1,G_2} of two graphs G_1 and G_2. MCS_{G_1,G_2} is defined as a common subgraph G of G_1 and G_2 such that there is no other common subgraph having more nodes than G. The first step in computation of MCS_{G_1,G_2} is to find all possible mappings M between two input graphs. Second, each pair of mapped vertices m from M is considered as a candidate for generating the set of common subgraphs. If a pair is a valid candidate the generated subgraphs will be expanded until no nodes can be added to the subgraph. This procedure is repeated for each candidate pair of mapped vertices. Finally, MCS_{G_1,G_2} is a graph with the maximum number of nodes among all generated common subgraphs.

Considering each pair of mapped vertices for subgraph generation, and further expansion of each graph in the set of all possible subgraphs lead to the exponential growth of the number of subgraphs which should be computed. In [61] the authors approximate the computation of MCS_{G_1,G_2} by reducing the subset of considered candidate pairs. This is done by selecting those vertices for which the relevance measure is higher than the mean relevance value of the nodes in the graph. The proposed relevance value of a vertex is a vector which represents information about the subgraph related the vertex. The distance between the relevance values of two vertices $d(v_i, v_j)$ is used to evaluate how the pair (v_i, v_j) contributes to the expansion of the common subgraph. Two different methods for selection of initial candidate pairs of vertices are used in [61] which lead to different MCS. The method based on grouping the initial set of candidates using vertex attributes resulted in associating semantically equivalent parts of two objects.

The second type of the methods used to detect subgraph isomorphism belongs to the class of inexact graph matching techniques which consider the vertices of graphs $G1$ and $G2$ as two disjoint sets of vertices of a bipartite graph and try to find the best assignment between the vertices from different sets. The smallest distance between the attributes associated to the vertices $d(v_1, v_2)$ is usually used as criterion for assignment two vertices $v_1 \in G_1$ and $v_2 \in G_2$. The problem of searching for the best correspondence, where a vertex from the first set of vertices can be mapped only to one vertex from the second set, and vice versa, is known as assignment problem. Formally, the assignment problem is stated as

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

, where c_{ij} are the coefficients representing the cost of assignment vertex $v_i \in G1$ to $v_j \in G2$, and x_{ij} are the elements of the permutation matrix:

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \sum_{i=1}^{n} x_{ij} = 1 \qquad x_{ij} \in \{0, 1\}.$$

There exist several algorithms aimed to solve assignment problem. Probably the most known of them is Hungarian algorithm originally proposed by H. W. Khun in [54].

If a shape is represented by a graph attributed with node and edge labels the graduated assignment algorithm is used to solve the problem of graph matching [41].

When the G_1 and G_2 are not large the brute-force method can be used to find the best assignment between the vertices. The vertices with smallest distance between their attributes are mapped to each other and the assessment of graph matching is the sum of the distances between all mapped vertices.

The lack of bipartite graph matching methods is that the connectivity relations between the nodes in a graph are completely discharged and only information stored in the labels of the nodes is used for matching.

The authors of [85] stored the connectivity information of each node as its label and used matching algorithm for bipartite graphs. Precisely, information stored along with each node is the sum of eigenvalues of the adjacency matrix of the graph rooted at that node. The eigenvalues intrinsically represent connectivity relations in the graph. Combining the adjacency information of each node together with bipartite matching algorithm the authors obtained good performance, retrieving in average 70% of desired models as top results.

Methods using the eigenvalues of a matrix associated to a graph belong to the class of spectral graph matching methods. The spectrum of a graph is the vector of eigenvalues of a matrix representing graph structure. Graph matching methods from these group gain more and more success for they allow to avoid NP problem of graph isomorphism detection. The cornerstone idea of spectral graph matching methods is that similarity between two graphs can be measured as the distance between their spectra. Moreover, the vectors of graph spectrum can be reduced to the same dimension by padding the spectrum of smaller graphs with zeroes, which corresponds to insertion of isolated dummy nodes to the graphs. This technique allows to perform efficient indexing methods for retrieval. Spectral graph matching methods were exploited not only in the field of 3D shape retrieval [83, 85, 32] but also for image [97, 32] and music retrieval [73]. We review spectral graph theory in more details in Section 3.4.

2.3 Evaluation of Efficiency of a Retrieval Process

There exist a large number of values to measure efficiency of shape retrieval process. Almost all of them were adopted from the field of information retrieval. The values of precision and recall are the most common measures used to evaluate efficiency of shape retrieval.

Figure 2.14 illustrates a set of all the objects of a shape repository U and two subsets, the set of the relevant objects A and the set of the retrieved objects B. The intersection of the two subsets $A \bigcap B$ indicates the subset of the relevant objects browsed to a user.

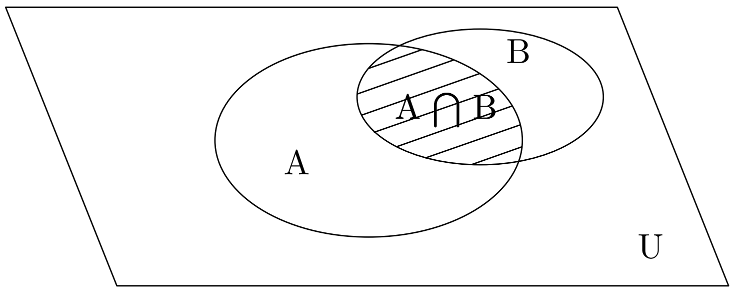


Figure 2.14: The set of all shapes U in a repository, A is the subset of all shapes relevant to a query, B is the subset of all retrieved shapes. The intersection $A \bigcap B$ is the subset of shapes relevant to the query and retrieved.

Precision is defined as the ratio of the number of relevant objects retrieved to the number of retrieved objects:

$$\text{Precision} = \frac{|A \bigcap B|}{|B|} \tag{2.1}$$

The value of precision is usually calculated for several tiers, i.e. for the first 10 retrieved objects, then for the first 20 etc. Precision_N means precision for the first N retrieved objects.

Recall is defined as the ratio of the number of relevant objects retrieved to the total number of relevant objects in a database:

$$\text{Recall} = \frac{|A \bigcap B|}{|A|} \tag{2.2}$$

Similarly to precision, Recall$_N$ means recall for the first N retrieved objects.

There are several combinations of the precision and recall values used to estimated the efficiency of retrieval process. Some of these values can be difficult to interpret because of the mixture of these two measures. One of the values is **Retrieval Efficiency** which is defined as precision while the number of retrieved objects is less than the number of all relevant objects $|B| < |A|$, and as recall otherwise, if $|A| < |B|$.

$$\text{Retrieval Efficiency} = \begin{cases} \text{Precision}, & \text{if } |B| < |A| \\ \text{Recall}, & \text{if } |A| < |B| \end{cases}$$

The other combined value is **F-Measure** which is by definition a harmonic mean of precision and recall values:

$$\mathrm{F-Measure} = \frac{2 \cdot \mathrm{Recall} \cdot \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}}$$

F-Measure is high if both precision and recall values are high.

Average dynamic recall is defined as the average of the relevant objects within first i retrieved objects:

$$ADR = \sum_{i=1}^{N} \frac{|A_i \bigcap B_i|}{i}$$

0.5 Precision recall is defined as the recall value for which the precision is below 0.5.

Error Rate is the ratio of the number of non-relevant objects retrieved to the number of all retrieved objects

$$\mathrm{ErrorRate} = \frac{|(U - A) \bigcap B|}{|B|}$$

The average ranking is the average position of all relevant objects in the order they returned as the retrieval result. Low value of the average ranking means that relevant objects appear on the top of the list of retrieval results.

$$\mathrm{AverageReanking} = \frac{\sum_{i \in A} Rank(i)}{|A|}$$

However, the measure of the average ranking can be significantly influenced by the high rank of few relevant objects, inadequately reflecting the retrieval performance of the whole system.

Normalized Average Ranking varies in the interval $[0;1]$ and it takes on the value of 0 for ideal retrieval and approaches 1 as retrieval results worsen:

$$\overline{R} = \frac{1}{|U| \cdot |A|} \left(\sum_{i \in A} Rank(i) - \frac{|A| \cdot (|A| - 1)}{2} \right)$$

The percentage of success of the first and the second retrieved objects is defined as their probability to be relevant. The probability of success for the first and second retrieved objects is computed for each query, and then it is averaged over the whole set of queries.

The last place ranking indicates the position of the last relevant object in the list of retrieval results. This value is computed as

$$L = 1 - \frac{Rank(Last) - |A|}{|U| - |A|}$$

Defined in this way, the value of the last ranking varies within the interval $[0;1]$ and it takes on the value of 1 for the ideal retrieval process when $Rank(Last) = |A|$.

All the listed values can be calculated for each query separately. Usually the performance measures are further averaged over each category of the models in a database providing the possibility to analyze the performance of the whole system as well as the efficiency of a shape descriptor for different types of shapes. Finally, the performance measures are averaged over the whole database.

Often to evaluate the efficiency of a retrieval system the **Discounted Cumulated Gain vector** is used. This measure provides the advantage of having the lowest standard deviation among other standard measures.

Gain vector G is the vector whose entry G_i is 1 if the retrieved object of the rank i is relevant to the query, and 0 otherwise. Then Cumulated Gain vector CG is defined as:

$$CG_i = \begin{cases} G_1, & i = 1 \\ CG_{i-1} + G_i, & \text{otherwise} \end{cases}$$

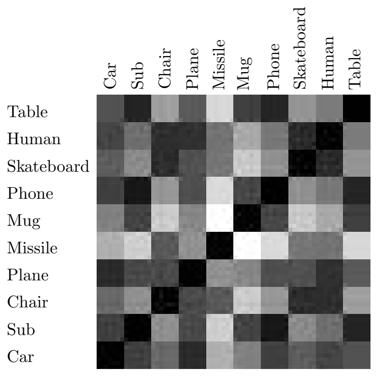


Figure 2.15: Distance matrix [69], which darkest diagonal entries correspond to the highest self-similarity values of the objects.

Discounted Cumulated Gain vector dampens the influence of less similar objects by adequately reducing the entry of the Gain vector:

$$DCG_i = \begin{cases} G_1, & i = 1 \\ DCG_{i-1} + G_i / \log_2 i, & \text{otherwise} \end{cases}$$

Graphical Performance Measures. There are several plots and histograms used to visualize the performance of a retrieval system.

One of the most used visual evaluation is the **distance matrix**, which gray scale entries reflect pairwise similarity between objects in a database. The rows and the columns of the distance matrix correspond to the objects in the shape repository which are usually grouped into relevant classes. The more similar two object i and j are, the smaller is the distance between them and the darker is the entry (i, j) of the matrix. Distance matrix allows to visually estimate which kind of shapes are more or less similar to each other according to exploited shape descriptor and matching techniques.

Precision plot and **Recall plot** are graphics of the precision and recall depending on the number of retrieved objects. From the definition of these measures (2.1), (2.2) it follows that recall grows as the number of retrieved objects increases whereas precision drops at the same time.

Precision Recall plot is a commonly used visual technique to compare efficiency of several retrieval systems. There exist a trade-off between recall and precision. The increase of the number of retrieved objects makes recall grow but at the same time reduces precision. The only way to increase both recall and precision is to improve the efficiency of a search engine by

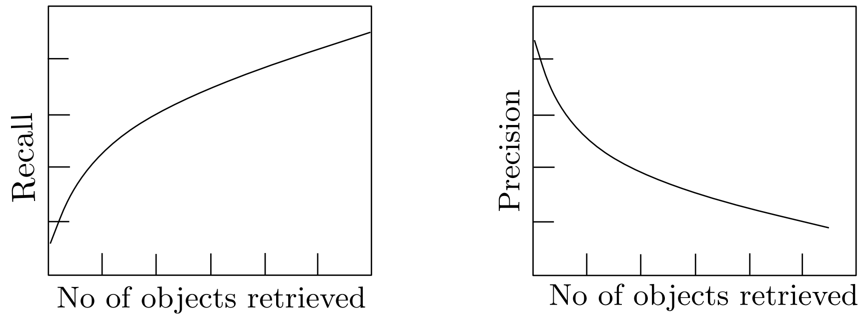


Figure 2.16: Recall and precision plots vs. the number of retrieved objects.

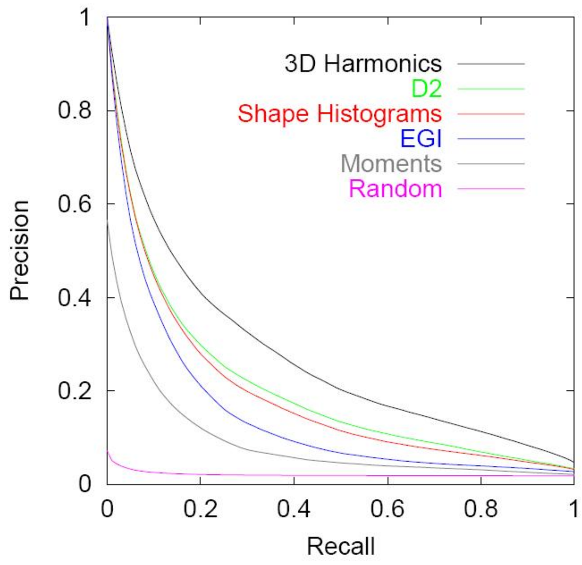


Figure 2.17: Precision Recall plot [52]. The top most plot corresponds to the most efficient retrieval.

introducing better techniques for shape representation and retrieval. The higher the efficiency of a retrieval system the more the Precision-Recall plot is shifted to the top-right.

In Chapter 5 we use some of the listed above evaluation measures to estimate the performance of our retrieval system.

2.4 Conclusions

In this section we have reviewed the main phases of the shape retrieval process.

The architecture of several prototypes of search engines available nowadays online was discussed. Mainly, functioning of a search engine can be divided into five phases, they are indexing of a shape repository, query formation, extraction of a shape descriptor from the query, search for the best matching of shape descriptors, and finally browsing the retrieval results to a user.

Different search engines use different shape descriptors to represent 3D models and to detect similarity between them. Shape descriptors can be divided into feature distribution based shape descriptors, transformation-based, view-based and finally graph-based shape descriptors. Descriptors from the latter group represent both topology and geometry of 3D models, allow to detect partial similarity of the models, but they are more difficult to compare. The methods used to detect partial similarity, or subgraph isomorphism of the descriptors, are usually approximation of exact graph matching techniques, where shape information stored in the descriptor is exploited to reduce the computation complexity.

Finally several measures for evaluation of efficiency of retrieval process were listed in this chapter. The most used measure are precision, recall, and discounted cumulated gain vector. Using precision-recall plot it is possible to compare efficiency of several search engines because the top and right most position of the plot indicates the greater efficiency.

In the following chapters we will reuse the Reeb graph based shape descriptor, and we will extend spectral graph matching techniques. Finally, to evaluate the efficiency of our framework we will use some of the listed evaluation measures.

Chapter 3

Theoretical Background

Differential topology together with Morse theory provides powerful tools for analysis and description of the shape of 3D models. These tools allow to study the topological properties of the manifold representing the shape through the critical points of a function defined on the manifold. This function is often called a measuring function as the value of the function at a point represents its measure and its significance in the context of the whole shape. A measuring function is also called a mapping function as it maps the points from the manifold of the shape to the codomain of the function. Configuration of critical points of the mapping function represents the topology of the shape. This configuration can be compactly encoded by a Reeb graph, which is used as a shape descriptor of 3D models. To estimate similarity between Reeb graphs of two different 3D models, spectral properties of the graphs can be exploited.

3.1 Differential Topology and Morse Theory

The theory of differential topology provides tools for shape analysis and description. In order to describe a shape we map the manifold representing a shape to a real-valued domain usually of lower dimension, called *chart*. To perform such a map we define a measuring function f on on the manifold. This function sets a correspondence between the points of the manifold and the chart $f : M \to \mathbb{R}$. Topological properties of the manifold M can be studied trough the analysis of critical points of the mapping function f.

A point p is a critical point of a function f, if all partial derivatives

$\frac{\partial f}{\partial x_i}(p)$ vanish at this point.

The Hessian matrix is the square matrix of partial derivatives of f:

$$H(f) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]. \tag{3.1}$$

A critical point p is non-degenerate if the determinant of the Hessian matrix is not zero:

$$\left|\frac{\partial^2 f}{\partial x_i \partial x_j}(p)\right| \neq 0. \tag{3.2}$$

The meaning behind a non-degenerate critical point is that the derivative $f'(x)$ takes on the value of 0 just once in the vicinity of the critical point p. Hence, non-degenerate critical points can be separated.

A smooth function defined on a smooth manifold is called *Morse* function if all its critical points are non-degenerate. A Morse function whose critical points have different values is called simple, otherwise it is complicated Morse function (see Figure 3.1 for illustration of degenerate and non-degenerate critical points). Complicated Morse functions can be easily turned into simple functions by slight local shape perturbation as shown in Figure 3.2b unless it is required to preserve the properties of invariant Morse function [39]. Consequently, any smooth function can be approximated by Morse function.

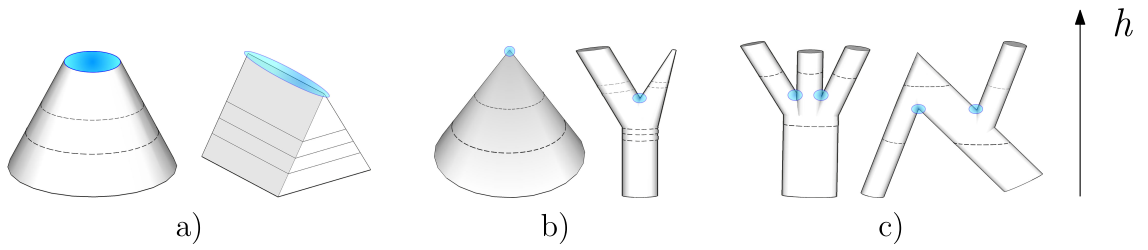


Figure 3.1: Critical points of the height measuring function. (a) degenerate critical points, (b) non-degenerate critical points with a single critical point per each critical value, (c) and with several critical points for the same critical value.

If a function used for analysis of the topology of a shape is a simple function, the shape manifold can be decomposed into regions which contain only a single critical point. In other words, the critical points of the mapping function can be separated. Such division of the manifold into

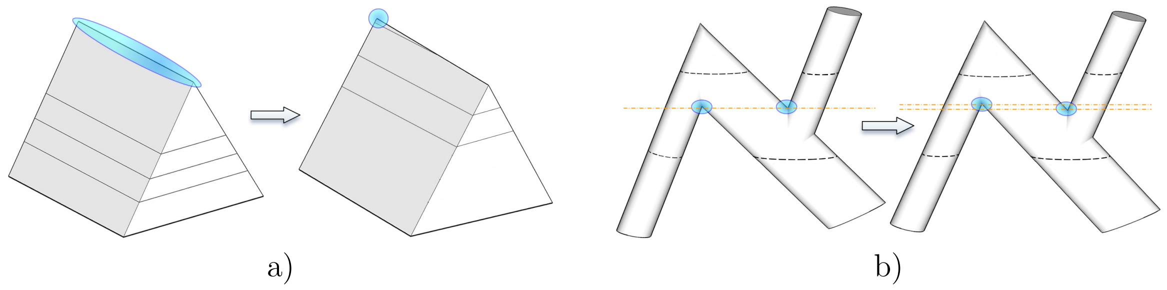


Figure 3.2: Degenerate critical points and non-degenerate critical points of a complicated Morse function. a) handling degenerate critical points by slight rotation; b) separating non-degenrate critical points with the same value of the mapping function by minor local shear.

critical regions corresponds to the decomposition into cells of different homotopy groups.

The number of negative eigenvalues λ of the Hessian matrix (3.1) at the critical point p is called the index of p. λ defines the type of the critical point: precisely, if $\lambda = 0$ (all eigenvalues are positive) then the critical point p is minimum; if all eigenvalues of the Hessian matrix are negative then p is maximum; otherwise p is a saddle point.

The index of a critical point is related to the homotopy type of the corresponding critical region, called λ-cell. If we define a level set f_i^{-1} of the function f as $f_i^{-1} = \{x|x \in M, f(x) = f_i\}$, then the topology evolution of the manifold M can be studied through the analysis of the evolution of level sets.

Suppose that an interval $M_{ab} = \{x|x \in M, f(a) \leq f(x) \leq f(b)\}$ does not contain a critical point of f. Then two level sets $f^{-1}(a)$ and $f^{-1}(b)$ are diffeomorphic. Moreover, if the level sets $f^{-1}(a)$ and $f^{-1}(b)$ are regular values of f (they are diffeomorphic to a circle), then the segment M_{ab} is diffeomorphic to a finite cylinder with two boundary components.

Consider now what happens when there is a critical point between two level sets $f^{-1}(a)$ and $f^{-1}(b)$.

Theorem 1 ([38]) *Let the function f be Morse function on a compact smooth n-manifold M (without a boundary), and lower set M_a be defined as $M_a = \{x|f(x) \leq a\}$.*

1. Suppose that 0 is a critical value of f, and that there is only one *critical point p in M_{ab} with index λ. Then M_b can be obtained from M_a by attaching a λ-cell.*

2. *The manifold M can be built by starting with the empty set, and attaching in succession a finite number of λ-cells, one λ-cell for each critical point of f of index λ. That is, f determines a cell decomposition of M.*

From the theorem it follows that two level sets $f^{-1}(a)$ and $f^{-1}(b)$ of the interval M_{ab} have different homotopy types if the interval contains a critical point. If the boundary of the interval M_{ab} ($a < b$), denoted as δM, is the union of two level sets $f^{-1}(a)$ and $f^{-1}(b)$, then if $f^{-1}(a)$ or $f^{-1}(b)$ is empty then there is a minimum, or respectively maximum in $M_{ab} - \delta M$, [47]. If we trace the changes in the sequence of the level sets $f^{-1}(x)$ of the interval M_{ab} of a 2D manifold $M(a \leq x \leq b)$ which are diffeomorphic to a circle, then each minimum corresponds to the introduction of a circle, i.e. attaching a 0-cell [96].

Similarly, according to the Theorem 1, extinguishing of a circle between two level sets $f^{-1}(a)$ and $f^{-1}(b)$ corresponds to the maximum occurrence in the interval M_{ab}, or, in other words, to attaching of 2-cell.

Finally, the union of two circles in the interval M_{ab}, with consequent formation of a new circle, corresponds to a saddle point on it, i.e. attaching 1-cell. Similarly, there is a saddle in the interval M_{ab} when there occurs a disconnection of one circle, which forms two new circles (the situation opposite to the described before).

Figure 3.3 illustrates introduction, extinguishing, union and split of circles which correspond to minimum, maximum and saddle points respectively.

Due to the fact that a Morse function can be defined on every compact smooth manifold, Theorem 1 implies that any compact smooth manifold has cell decomposition [38].

The graph with the nodes representing cells after decomposition of the manifold, and the edges reflecting the adjacency of two cells, is called Extended Reeb Graph (ERG). The ERG was defined in [21] as the configuration of critical areas of the surface.

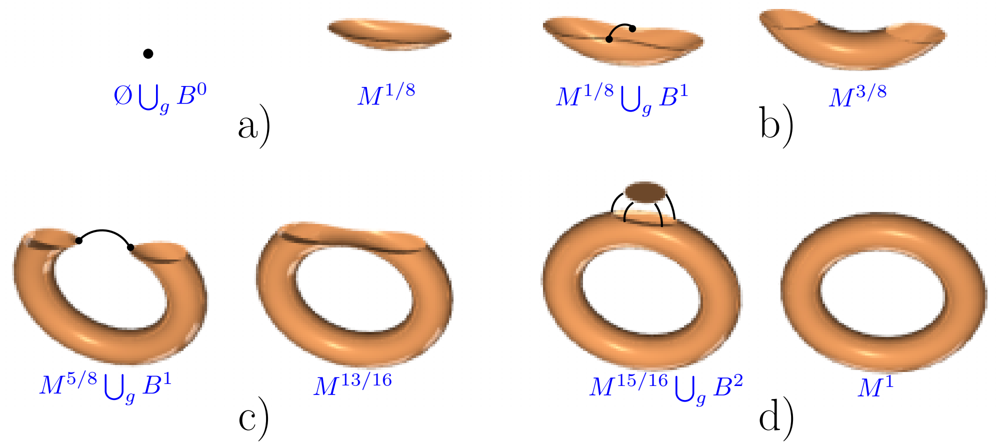


Figure 3.3: Cell decomposition of the torus. a) Attaching 0-cell corresponds to introduction of a circle. b) Attaching 1-cell corresponds to the union of two circles. c) Attaching of 1-cell corresponds to the disconnection of a circle and to formation of two new circles from one. d) Attaching 2-cell corresponds to extinguishing of a circle [45].

We, first, give a definition of Reeb Graph originally proposed in [76] and further studied in [82] with the application to the surface analysis and coding. Next, we will adduce the definition of the Extended Reeb graph.

Definition 1 (Reeb graph) *Let $f : M \to \mathbb{R}$ be a real valued function on a compact manifold M. The Reeb graph of M with respect to f is the quotient space of $M \times \mathbb{R}$ defined by the equivalence relation "$\sim$", which states that $(X1, f(X1)) \sim (X2, f(X2)$ if and only if:*

1. *$f(X1) = f(X2)$;*
2. *$X1$, $X2$ are in the same connected component of $f^{-1}(f(X1))$ (or $f^{-1}(f(X2))$).*

The Reeb graph encodes the evolution of level sets of the mapping function f. A node of the Reeb graph represents a level set where the function takes on a critical value; an edge between two nodes in the graph represents the adjacency of the corresponding critical level sets, no topological changes of the manifold occur along the edges of the graph. Figure 3.4 illustrates several level sets of the height mapping function of a model and the corresponding Reeb graph.

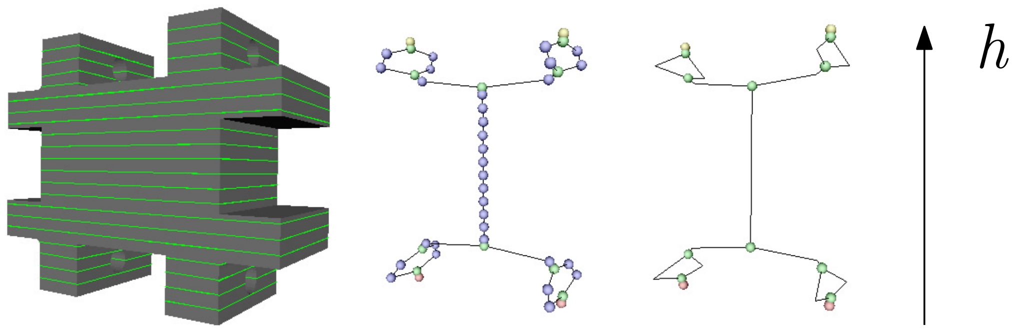


Figure 3.4: Contour driven approach for Reeb graph construction.

Extended Reeb Graph was first defined in [21]. It extends the equivalence relation given in Definition 1 from level sets to the critical areas.

Definition 2 (Extended Reeb Graph) *An Extended Reeb equivalence between two points $X1, X2 \in M$ is given by the following conditions:*

1. *$f(X1)$ and $f(X2)$ belong to the same critical area of a critical point $P \in M$;*
2. *$f(X1)$ and $f(X2)$ belong to the same connected component of $f^{-1}(f(p))$;*

ERG corresponds to the cell decomposition of the manifold, for it relates critical areas of the mapping function to topologically different regions of the manifold. Figure 3.5 illustrates cell decomposition of the model and corresponding ERG.

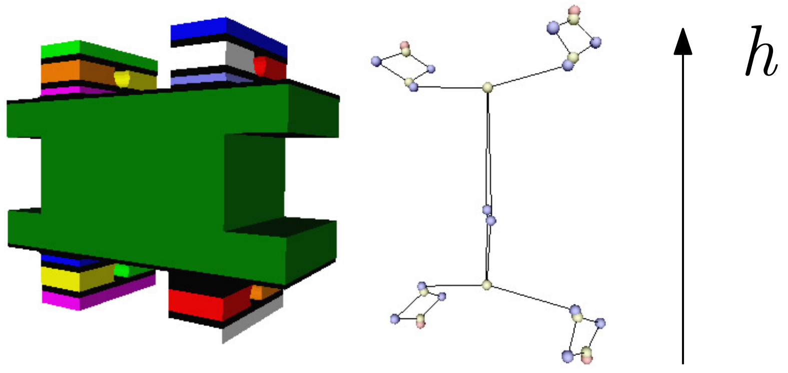


Figure 3.5: Cell decomposition approach for Extended Reeb graph construction.

3.2 Measuring Functions for Reeb Graph Construction.

Decomposition of a manifold into cells of different homotopy types and construction of a Reeb graph requires definition of a simple function on the manifold. In the current section we discuss several functions and their properties which are used to construct Reeb graphs. Some of them are not Morse and can have degenerate critical points. However, as illustrated on Figure 3.2 such cases can be resolved by slight local shape perturbations or by additional controls, as will be discussed further in Section 3.3. All of the listed below measuring functions, except height function, are invariant to rotation. The invariance to scale is usually achieved by normalizing the function so that it varies in the range [0..1].

Heigth function is a function which measures the elevation of a shape. After the shape is aligned as desired, the function represents just y-coordinate:

$$H(x, y, z) = y.$$

Barycenter distance function is a function measuring the distance between each point of a manifold and its center of mass. The center of mass is defined as $C(\bar{x}, \bar{y}, \bar{z}) = (\frac{\sum_{i=0}^{i<N} x_i}{N}, \frac{\sum_{i=0}^{i<N} y_i}{N}, \frac{\sum_{i=0}^{i<N} z_i}{N})$, where N is the number of vertices. Sometimes weighted center of mass is used: $C(\bar{x}, \bar{y}, \bar{z}) = \frac{\sum_{i=0}^{i<N} S_i(A_i+B_i+C_i)}{3S}$ where N is the number of triangles, S is the total surface area, S_i is the area of the i-th triangle, A, B and C are the vertices of the triangle.

The barycenter distance function is then computed as

$$B(x, y, z) = \sqrt{(x - \bar{x})^2 + (y - \bar{y})^2 + (z - \bar{z})^2}$$

.

Integral Geodesic distance function measures the average distance between a point and all the other points over the surface of a model:

$$G(v) = \sum_{i=0}^{N} geod\_dist(v, v_i)$$

The geodesic distance between two vertices on the surface is computed using Dijkstra algorithm which finds the shortest path between the two vertices on the surface made up by n vertices. The complexity of the algorithm is $O(n \log n)$. To reduce the complexity the authors of [46] approximated integral geodesic function which computes the distance between a point and all base points. The base points are scattered over the surface of a model almost uniformly, they occupy almost equal areas on the surface. The distance between a point on the surface and a base point is weighted with the area adjacent to the base vertex. Hence, the approximate geodesic distance function is computed as:

$$G\_apprv = \sum_{i=0}^{Nb_i} geod\_dist(v, b_i) \cdot area(b_i).$$

The complexity of computation of the approximate integral geodesic functions for all n vertices composing the mesh of a model and using b base points is $O(bn \log n)$.

Integral angular distance is a function which measures the average angle between the normal at a given vertex and the normals of all other vertices on the surface. The angle between to vertices with are not incident is computed as the smallest angle accumulated along the path connecting the vertices. Similarly to integral geodesic distance, integral angular distance can be approximated by calculating the shortest angular paths only from the base points.

$$A(v) = \sum_{i=0}^{Nb_i} (n_v, n_{b_i}),$$

where N_v is the unit normal at the vertex v computed as the average of normals of the triangles incident to v and normalized to a unit size.

Laplacian eigenfunction is usually the first eigenfunction of a Laplacian matrix defined on the shape. In [102] the authors overview several Laplacian matrices which encode geometrical shape information together with its topology. Once Laplacian matrix L is defined on a shape, the eigen decomposition of L can me computed to find the Laplacian spectrum $\{\lambda_0, \lambda_1 \dots \lambda_n\}$ together with eigenvectors $\{[e_0], [e_1] \dots [e_n]\}$. The i-th Laplacian eigenfunction on the vertex v which is represented by k-th row and column in the Laplacian matrix is computed as:

$$Lf_i(v_k) = \sqrt{\lambda_i} \cdot e_{ik}.$$

The complexity of computation of the Laplacian eigenfunctions depends on how sparse the Laplacian matrix and varies from $O(n \log n)$ till $O(n^2)$.

The listed measuring functions have different critical points and consequently lead to different description of a manifold. In some applications it is important to study and compare several measuring functions in order to ascertain which critical points are redundant and which are characteristic for the manifold in question. The easiest way to compare several measuring functions is to find the correlation between them:

$$Corr_{fg} = \frac{\sum_{i=1}^{n}(f_i - \bar{f})(g_i - \bar{g})}{\sqrt{[\sum_{i=1}^{n}(f_i - \bar{f})^2] \cdot [\sum_{i=1}^{n}(g_i - \bar{g})^2]}} \tag{3.3}$$

where $\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$ and $\bar{g} = \frac{1}{n}\sum_{i=1}^{n} g_i$ are the mean values.

However, the correlation between two functions defined on the mesh estimates the global similarity, and does not consider the connectivity between the vertices. In order to capture such information and to evaluate similarity between a set of measuring functions the authors of [35] propose a new measure. Computed for two functions at one point it evaluates the area of the parallelogram spanned by the gradients of these functions. Hence, the similarity measure between two functions computed on the manifold $\mathbb{M}$ is integrated over the surface:

$$\kappa(f, g) = \int_{x \in \mathbb{M}} ||\nabla f(x) \times \nabla g(x)|| \mathrm{d}x / area(\mathbb{M}) \tag{3.4}$$

The other approach to compare measuring functions was proposed recently in [23]. The defined correlation factor for the set of measuring functions $f_1, \cdots, f_n$ is computed as:

$$\sigma(f_1, \cdots, f_n) = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j \geq i}[a_{ij} - \bar{a}]^2}{n}} \tag{3.5}$$

where $\bar{a} = \frac{1}{n(n+1)}\sum_{i=1}^{n}\sum_{j \geq i} a_{ij}$ and a_{ij} is defined as $\kappa(f_i, f_j)$ in (3.4).

Figure 3.6 illustrates the color maps of scalar fields defined by some of the listed above measuring functions. Blue color corresponds to low and red to high values of a mapping function.

Table 3.1 shows the correlation (3.3) between some of the listed above functions. These values were computed for 10 different 3D models and

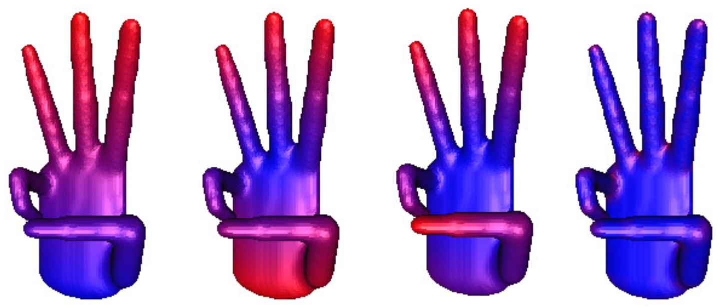


Figure 3.6: Color maps of measuring functions.

finally averaged. The negative values of the correlation coefficient between height and the other measuring functions reflect the fact that the points where the height function take on its minimal values are the points on maximum values for the other functions. The high correlation value between barycenter and integral geodesic distance indicates the fact that these functions have similar values in the same regions of a manifold.

Table 3.1: Global correlation of measuring functions. Here H is the height function, B is the function of the distance from the barycenter, G_appr is approximate integral geodesic distance, and A is approximate integral angular distance

	H	B	G_appr	A
H	1	-0.09	-0.01	-0.09
B	-0.09	1	0.63	0.25
G_appr	-0.01	0.63	1	0.21
A	-0.09	0.25	0.21	1

3.3 Methods for Reeb Graph Construction.

Once the value of the measuring function is computed for all the vertices of a manifold, the Reeb graph can be constructed to describe its topology. There are three main approaches to construct a Reeb graph. The first approach is based on mutiresoltutional decomposition of the manifold of a shape into connected components. The decomposition proceeds in fine-to-coarse direction. In the beginning the manifold is subdivided into K

intervals according to the value of the function, where K is the finest resolution of the decomposition.

$$\bigcup_{i=0}^{K-1} [f_{min} + i \cdot h; f_{min} + (i+1) \cdot h], \tag{3.6}$$

where $h = \frac{f_{max} - f_{min}}{K}$ (see Figure 3.7). Each connected component after such division corresponds to a node in the Reeb graph of the K-th resolution. There is an edge between two nodes of the graph if the corresponding connected components are adjacent (Figure 3.7b). Next, the coarser resolution Reeb graph is constructed by unifying adjacent nodes (Figure 3.7c,d). The nodes of the coarser resolution are parents of the child nodes of the finer resolution. On Figure 3.7 the node n_5 is a parent of the nodes n_1 and n_0, the node n_6 is a parent of the node n_2. The relation between the graphs of different resolution is expressed by the presence of edges between the nodes of the Reeb graphs of different resolution. The nodes of graphs of different resolution correspond to different colors on the Figure 3.7; the edges between these nodes are depicted with red dotted lines on Figure 3.7e.

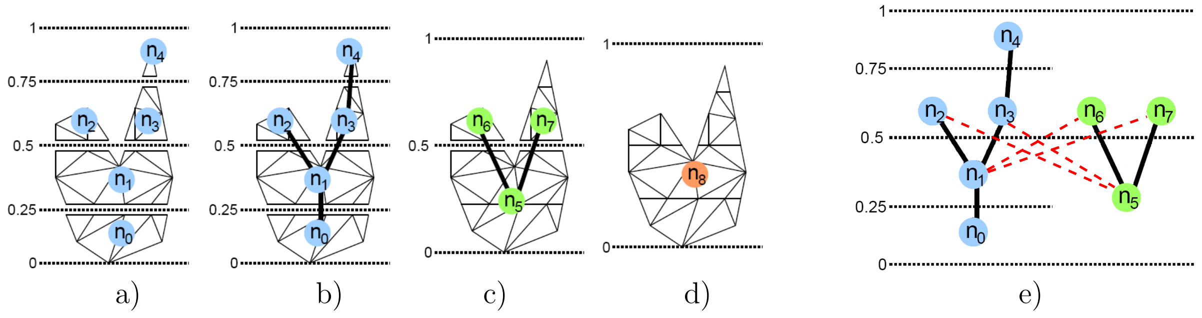


Figure 3.7: Multiresolutional Reeb Graph, K=4 [46]. (a) Manifold division and decomposition into connected components. (b) Construction of the Reeb graph of 4th resolution. (c) Construction of the Reeb graph of 2nd resolution. (d) Construction of the Reeb graph of 0th resolution (the whole model corresponds to one node). (e) Red doted lines are edges between the nodes of graphs of different resolutions.

The complexity of the construction of the Multiresolutional Reeb graph is $O(V)$ [46], where V is the number of vertices in the triangular mesh of a model. However, the precision of the graph description depends on K

value. In other words, the smaller the intervals of the mapping function (see formula 3.6), the higher the probability to reveal all topological details of a model. Therefore, if a 3D model contains small holes or other shape features which are completely included in the interval $[f_{min}+i\cdot h \; f_{min}+(i+1)\cdot h]$ then such topological details will not be revealed in the graph. Hence, Multiresolutional Reeb graph should be considered as an approximate description of the topology of a shape.

While matching Multiresolutional Reeb graphs the hierarchy between parent and child resolutions is exploited to avoid NP problem of graph isomorphism detection. Matching proceeds in coarse-to-fine direction. Two nodes can be matched if they belong to the same resolution level $[f_i; f_{i+1}]$ and if their parent nodes are matched. If two nodes are mapped to each other, the matching label is propagated in the graph in the direction of increase and decrease of the mapping function. This creates an additional constraint for matching, i.e. two nodes can be mapped if the lists of their matching labels coincide. The three constraints for matching of two nodes reduce the number of nodes to be compared, this consequently leads to the reduction of the complexity of graph matching.

The second method to construct the Reeb graph was proposed in [81, 82] and further studied and extended in [13]. The method is based on tracing the evolution of level sets of the mapping function. It is a direct application of Morse theory to discrete representation of shapes. First, a set of contours corresponding to K different level sets is inserted into the mesh. In [81] each inserted contour, or cross section as named by authors, corresponds to a node in the graph. An edge is inserted between two nodes corresponding to two subsequent level sets. When a level set is composed by more than one component the decision about connection of corresponding nodes with edges is based on the smaller distance between contours. Of course, such approximation of shape topology can introduce loss of important topological features.

The authors of [13] extended the approach proposed in [82]. Instead of tracing the evolution of level sets, the authors consider the homotype of each component obtained after insertion of contours corresponding to K level sets of the mapping function. A component containing a critical point, instead of a contour, is represented by a node in the Extended Reeb graph. The adjacency between two nodes in the ERG is defined by expanding the critical region corresponding to the first node until the second critical region is reached. Moreover, the authors defined the rules to handle degenerate critical areas avoiding small local shape perturbations. To distinguish between a regular region, flat maximum or handle-like maximum the value of the mapping function on the boundaries of the critical region is controlled. A regular region has different values of the mapping function on its boundaries, whereas generate maximum region has two or more boundary components with the same value of the mapping function and descending value in the adjacent regions. To discriminate handle-like and flat maximums, the inclusion of boundary components is controlled. Similar examinations are performed to distinguish between saddle and degenerate minimum or maximum regions. Additionally, the authors control whether all holes of the analyzed shape are revealed. This control is performed for each component. If there is a hole in a component, an additional contour is inserted inside the component. Constructed in this way ERG guarantees that all holes present in the shape of a 3D model are reflected in the graph structure. However small minimum and maximum regions which size is less then the interval between two subsequent level sets will be lost.

Recently the new approach to construct the Reeb graph was proposed in [71]. The method is fast and robust, independent on the dimension of the manifold and can work even with non-manifold shape representation. The proposed approach exploits efficient data structures for both shape and graph representations and links between them. The approach is based on the consideration that the Reeb graph is obtained by shrinking the level sets of a mapping function into a point. The graph construction is done on-fly while reading the sequences of vertices and triangles representing a model. For each new vertex, a Reeb node is constructed. For each three edges of a triangular, unless they have already been read, a new arc in the Reeb graph is inserted. Two paths connecting the highest and the lowest vertices in the triangle are merged as in merge sort algorithm. While merging the arcs in the graph, the position of the nodes is updated, it is averaged over the positions of nodes in the merged paths. To ease the traversal between the graph and the shape, the arcs in the graph point to the edges of the mesh intersected by the corresponding level set; an edge points to the highest arc in the graph which points to this edge. Figure 3.8 shows the main steps of the Reeb graph construction process.

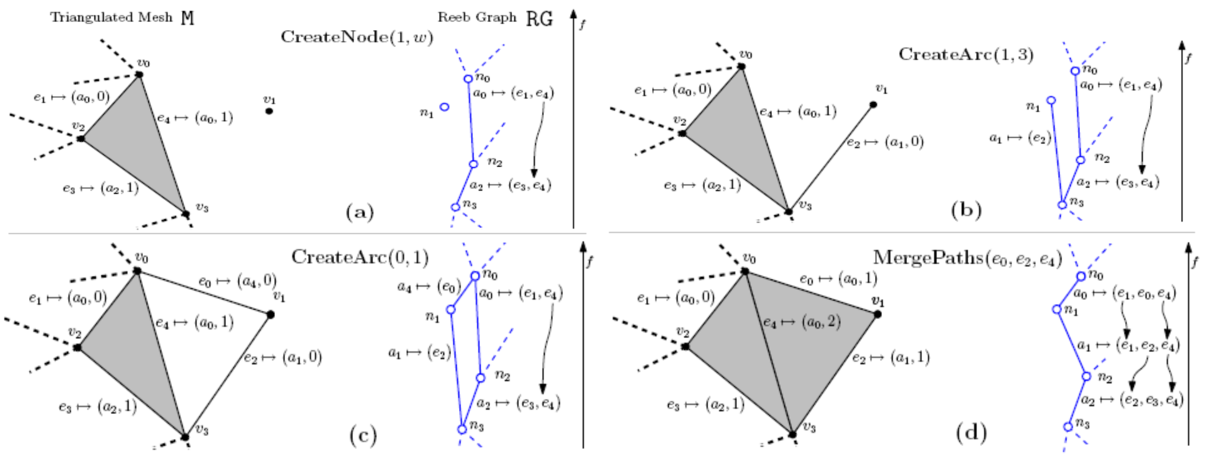


Figure 3.8: The main steps of the proposed in [71] approach for construction of the Reeb graph. a) For a new vertex v_1 insert a new node n_1 in the Reeb graph. b-c) For each new edge e_2 and e_0 insert a corresponding arc in the graph. d) For each triangle (after all its vertices and edges have been read) merge two paths a_1, a_4 and a_2, a_0 connecting the lowest and the highest vertices in the triangle.

The constructed in this way Reeb graph is embedded inside the shape repeating its outline. It is done by representing the arcs not by the two endpoints but by the sequence of buckets of intermediate position, where each bucket is represented as a couple of the number of corresponding vertices in the mesh and the vector of their average position. Figure 3.9 shows the embedding of the Reeb graphs.

The approach proposed in [71] avoids introduction of new edges inside the original mesh as the result of intersection of the mesh with level sets of the mapping function. Consequently, it reduces the complexity of the graph construction.

The Reeb graph of [71] exactly follows the shape in the increasing direction of the value of the mapping function, revealing all topological details of the shape. This makes the Reeb graph be applicable for topology control, when small defects are not visible and models are very big.

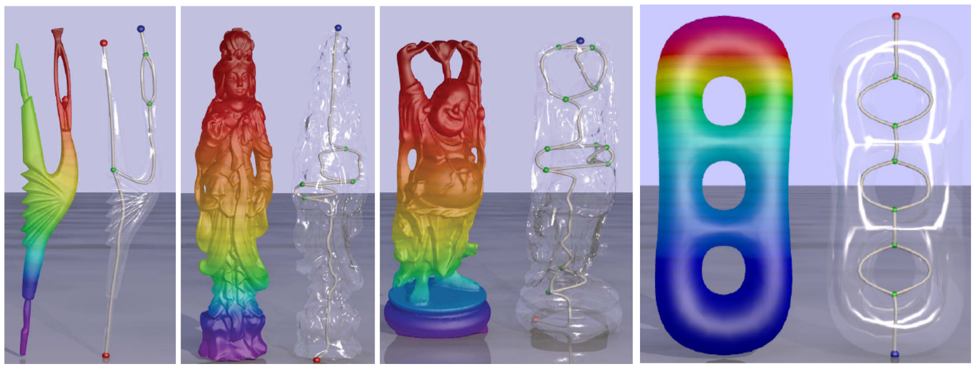


Figure 3.9: The Reeb graphs and their embedding [71]. The colors of the models shows the values of the mapping function. Blue corresponds to the low and red to the high values of the function.

3.4 Pattern Vectors for Graph Representation.

The next phase in the retrieval process after construction of shape descriptors is evaluation of their similarity. Working with the Reeb graph based shape descriptor, we are interested in the methods for detection of subgraph isomorphism. Some of the methods have been reviewed in Section 2.2.3. Recently, techniques from spectral graph theory gain more success for assessment of similarity between two graph structures, because they reduce the problem of the subgraph isomorphism detection to the computation of the distance between the vectors representing the graphs. In this section we review adjacency and Laplacian matrices which are used to represent graphs and discuss their useful properties which can be exploited in graph matching.

A graph G is an ordered pair (V, E), where V is a set of graph nodes, or vertices; and E is the set of pairs of vertices (v_i, v_j), called edges or arcs. The size of the graph G is the number of edges $|E|$, the order of the graph is the number of graph vertices $|V|$. The degree of a vertex v_i, denoted as $deg(v_i)$, is the number of incident edges.

Adjacency matrix $A(G)$ of the graph G is the matrix which rows and columns correspond to graph vertices, and which entry a_{ij} is defined as:

$$a_{ij} = \begin{cases} 1, & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases}$$

The adjacency matrix of the undirected graph is symmetric.

Consider a weighted undirected graph and let each edge $(v_i, v_j) \in E$ have an associated real number w_{ij} which satisfies the following conditions:

$$\begin{aligned} w_{ij} &= w_{ji}, \text{ where } v_i, v_j \in V & (3.7) \\ w_{ij} &\geq 0, \text{ where } v_i, v_j \in V & (3.8) \\ w_{ij} &\neq 0, \text{ iff } v_i \text{ and } v_j \text{ are adjacent in } G & (3.9) \end{aligned}$$

The adjacency matrix of the weighted graph is the matrix which entries a_{ij} are define as:

$$a_{ij} = \begin{cases} w_{ij}, & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases}$$

If we normalize the adjacency matrix so that the sum of each row equals to 1, i.e. $\sum_{j=0}^{n} a_{ij} = 1$, then the adjacency matrix can be interpreted as the matrix of probabilities to perform a walk in the graph. The main eigenvector of the matrix is known as the vector of steady-state probabilities which i-th entry indicates the probability of being in the vertex i in the steady state. This vector as well as the vector composed of eigenvalues of the adjacency matrix are often used to represent the graph in a compact way. The sorted vector of the eigenvalues $(\lambda_1, \lambda_2 \ldots \lambda_n)$, called graph spectrum, is permutation invariant. Due to this property graph spectrum is often used as its signature.

In case of a directed graph, the adjacency matrix is not symmetric and the eigenvalues are complex, hence the magnitudes of the complex eigenvalues are used as the graph signature.

The similarity between two graphs is calculated as the distance between its signatures.

$$Dist = \sqrt{\sum_{i=0}^{n} (\lambda_{1i} - \lambda_{2i})^2} \tag{3.10}$$

Suppose two graphs G_1 and G_2 have different number of nodes n and m, (let $n < m$) resulting in graph signatures of different length. To calculate the distance using (3.10) the signature of G_1 is padded with zeroes till the length of signature of $G2$. This corresponds to insertion of isolated dummy nodes in the original graph.

Permutation invariance of the graph spectrum guarantees that isomorphic graphs have the same spectrum. However, the opposite is not always true. In other words, non-isomorphic graphs can have the same spectrum. Two graphs are called cospectral if they have equal spectrum with respect to the same representation matrix. It was shown in [103] that among more than 2 million different trees with 21 nodes there is 21,3% of cospectral trees for adjacency matrix representation. This figure is reduced to 0,05% for Laplacian matrix representation. Additionally, it was shown that similar graphs have similar spectrum. Small graph perturbations, like edge and node deletion, lead to small changes in graph spectra. The results shown in [103] lead to the immediate conclusion that Laplacian spectrum provides a descriptive and stable way to represent a graph in matching process. Recently the properties of the spectrum and eigenvectors of Laplacian matrix are used in different fields of shape analysis, segmentation and matching.

Laplacian matrix $L(G)$ of the graph G is defined as $L(G) = D(G) - A(G)$, where $D(G)$ is the diagonal matrix which elements equal to the degree of the corresponding vertex $d_i = deg(v_i)$, and $A(G)$ is the adjacency matrix. The Laplacian matrix for a weighted graph is defined as the matrix whose elements satisfy the following conditions:

$$l_{ij} = \begin{cases} -w_{ij}, & \text{if } i \neq j \\ \sum_{j=0}^{n} w_{ij}, & \text{if } i = j \end{cases} \tag{3.11}$$

The weights w_{ij} in (3.11) should satisfy the conditions (3.7)-(3.9).

The smallest eigenvalue of the Laplacian matrix λ_1 is equal to 0. The multiplicity of this eigenvalues corresponds to the number of connected components in the graph. The second smallest eigenvalue of the Laplacian matrix λ_2 is known as algebraic connectivity of the graph. It carries probably the most important information about the graph. In [65] the author surveyed the bounds of several graph invariants which involve λ_2. The eigenvector associated to the second smallest eigenvalue is called Fiedler vector after the Czech mathematician Miroslav Fiedler who studied the properties of eigenvalues and eigenvectors of the Laplacian matrix for measuring the algebraic graph connectivity. Fiedler vector is often used to solve the problems of graph partitioning [75, 87].

3.5 Conclusions

In this chapter we have given the theoretical background for analysis of topology of a shape using Morse and differential topology theories. Several functions can be used for studying the topological properties of the shape. The functions with the lowest computational complexity and invariant to the rotation and posture of a model in space are the most valuable. Configuration of the critical points of the measuring function represents topological changes of the manifold, it can be encoded by the Reeb graph. We have reviewed three main techniques for Reeb graph construction. The method based on examining the homotopy type of each component of a shape is reused in the following section for construction of the Extended Reeb graph based shape descriptor and shape analysis. In this chapter we have also discussed the properties of the eigenvalues and eigenvectors of adjacency and Laplacian matrices. These matrices are often exploited in the estimation of similarity between two different graphs. In the following chapter we reuse some of these properties to match the graph based shape descriptors enriched with local geometrical shape characteristics.

Chapter 4

Shape Description and Retrieval

The process of shape retrieval consists of several phases, among which extraction of shape descriptors and their matching are the most essential.

Extraction of a shape descriptor requires careful analysis of both topology and geometry of the shape of a 3D model. These two shape characteristics can be integrated and represented using a graph structure. In the current chapter we describe how to represent the topology of a model with an Extended Reeb graph, and how enrich pure topological structure with local geometrical features.

Shape matching, where a shape is represented by the ERG, is analogous to the detection of graph isomorphism. To cope with the NP problem of searching graph isomorphism we propose to represent the ERG by the eigenvector of the Hermitian matrix associated to the graph. The distance between the vectors of two graphs represents the degree of similarity between them. Moreover, using the same vector we partition the graphs in order to solve the problem of matching graphs of different sizes.

Finally, in this chapter we present the methodology for synthesis of geometrical and topological description of a shape and its semantics. The methodology is proposed for models from furniture domain and uses a simple shape descriptor. However, it can be extended to other domains by exploiting more sophisticated techniques for shape analysis and representation.

4.1 Construction of a Reeb Graph

A Reeb graph represents the topology of a shape in a compact graph structure. Topological changes of the manifold of the shape are studied trough the analysis of the critical points of the mapping function. To reveal the critical points we trace the evolution of contours inserted inside the manifold, where each contour is a connected component of a level set of the function (for boundary representation of 3D models a contour is homeomorphic to a circle). Hence, constructing the Reeb graph we, first, compute the values of the mapping function on the vertices of the triangular mesh representing the model. Throughout this chapter we will use the term scalar field of a vertex to refer to the value of the mapping function at this vertex. The second step in the construction of the Reeb graph is definition of the number of the contours to be inserted inside the manifold. This number should be adapted to the complexity of the model in question. The more topological features are present in the model the more contours are needed to be inserted inside the mesh in order to reveal the features in the graph structure. After the insertion of the initial number of contours, the topology of each interval confined by two contours from consecutive level sets should be analyzed. If the interval contains a through hole which should be revealed in the graph, an additional contour can be inserted inside the interval.

Working with triangular mesh boundary representation of 3D models, contours are composed by vertices and edges. In order to construct a a contour for the particular value f_i of the mapping function f we search for the edges which endpoints v_1 and v_2 are located on the different sides from the current value $f(v_1) \leq f_i \leq f(v_2)$. Such edges are split in two new edges and the splitting point belongs to the corresponding level set. If scalar fields of the endpoints of an edge equal to the current value f_i then the edge is a constituent of the contour.

Working with accurate models which are represented by the consistent triangular mesh it is guaranteed that the models do not contain degenerate contours, which are tangent, open contours and contours consisting only of one point. A contour can be degenerate only when it is inserted at a critical value of the mapping function. In this case we can slightly rotate the model as shown on Figure 3.2 or shift the current contour corresponding to f_i value by a small quantity ϵ, so that the new current value of the mapping function is $f_i = f_i - \epsilon$.

Further, the construction of the Reeb graph can be proceeded using two alternative approaches. In the first approach we consider a contour as an approximation of the adjacent areas. Every contour is represented by a node in the graph. There is an edge between two nodes if the corresponding contours are reachable through the mesh without intersecting any other contour. This approach is described in more details in Section 4.1.1. We use the first approach in order to distinguish between the contours which represent a model and those representing its inner cavities. Consequently, we construct the Reeb graph of a 3D model, which is necessary connected because of the requirement for the connected and consistent mesh representation, and the Reeb graph for the complement of the model, which can be disjoint.

In the second approach a node of the Reeb graph represents a section of the shape confined between two contours from consecutive level sets. This kind of the Reeb graph was proposed in [20] and it is called Extended Reeb Graph (ERG). The nodes of the ERG encode the areas of the manifold where the topological changes occur. The edges of the ERG represent the adjacency relation of two neighbor shape segments. We explain the reasons for turning from the contour driven to the component driven approach in Section 4.1.2 and described the component driven approach in more details in Section 4.1.3.

4.1.1 Contour Driven Reeb graph

In applications of computer graphics where topology analysis is involved the use of the complement of an object, also called as background, is widespread [79], [44]. The analysis of topology of both an object and its complement captures important information about the entire object structure. Inspired by this approach we analyze the topology of a 3D model and its complement to obtain better shape description. Consequently, we construct two Reeb graphs. We require the first Reeb graph which describes the topology of the model to be connected, whereas the second Reeb graph representing the complement can be disconnected.

After the set of contours is inserted inside a model, we separate them into two groups. The first group of contours represents the model, or in other words, its outer surface. The second group of the contours represents the complement, or the surface of the inner cavities and holes. In order to separate the inserted contours in two mentioned groups we use the parity count method.

Parity count method is used in computer graphics for mesh simplification and repair [67], [37] when the detection of inner and outer parts of an object is required. We consider the set of the contours at each level set of the shape measuring function separately. For each contour from the current level set we draw three rays intersecting the contour, and we count how many times it intersects the other contours from the same level set. If the number of intersections of a ray and the contours, different from the considered contour, is even, then the contour represents the model, otherwise, it represents the complement. Our decision to draw three rays instead of one is determined by the possibility to draw a ray tangent to one of the contours from the same level set. In this case parity count wrongly defines the classification group. The final classification group of the contour is defined as the voting result of the three drawn rays. Figure 4.1 illustrates the parity count method.

After separating the contours into two groups we construct two different Reeb graphs for the contours representing the model and the complement.

While inserting the contours inside the mesh representing the model we also calculate several geometrical shape characteristics of the contours. We use these characteristics as labels of the corresponding nodes of the graph. The shape characteristics which we compute are perimeter, area and average curvature of the contour [92], [55].

Each contour inserted inside the mesh is represented by a node in the Reeb graph, geometrical characteristics of the contour are stored as labels of the node. The edges of the Reeb graph represent the adjacency of the contours. If one contour can be reached from another through the mesh without intersecting any other contour, then there is an edge between the corresponding Reeb nodes. In order to detect if two contours are adjacent we trace their connectivity by traversing the triangular mesh.

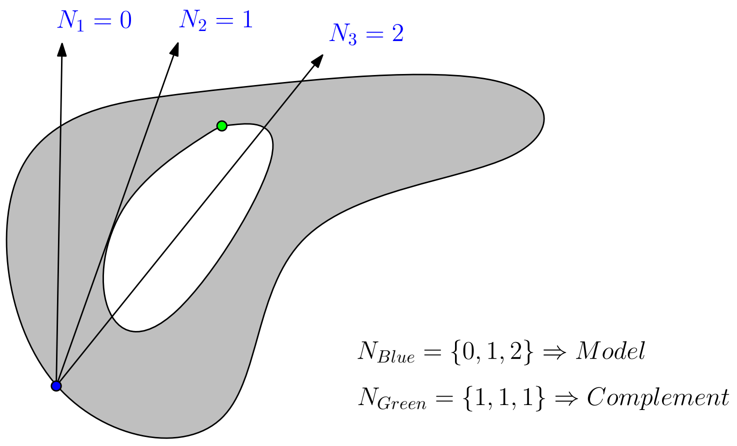

$N_1 = 0$
$N_2 = 1$
$N_3 = 2$
$N_{Blue} = \{0, 1, 2\} \Rightarrow Model$
$N_{Green} = \{1, 1, 1\} \Rightarrow Complement$

Figure 4.1: The triple parity count for the blue vertex of the contour results in 1,2 and 0 intersections. The voting result is that the contour represents the model (the majority of the intersection numbers is even). For the green vertex of the second contour the parity count gives 1 intersection for all three rays. Thus the second contour describes the complement.

Consider two contours from the consecutive level sets f_i and f_{i+1}, $f_i < f_{i+1}$. Starting from the triangles adjacent to the contour of f_i level set and expanding the mesh in the direction of increasing value of the mapping function we try to reach the contour of f_{i+1} level set. If it is possible we insert a new edge between the nodes representing the considered contours. While tracing the connectivity between two contours we mark all visited triangles to avoid looping by traversing always the same path. The mesh traversal is terminated once the contour of the higher level set f_{i+1} is reached or no unvisited triangles are left between the considered level sets. Figure 4.2 illustrates the process of tracing the connectivity between two contours.

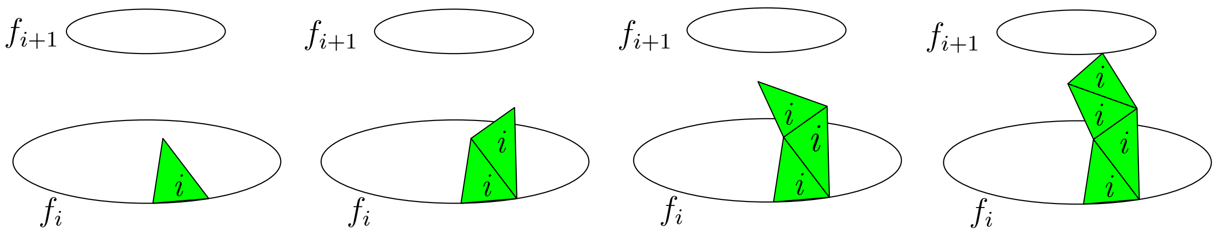


Figure 4.2: Tracing the connectivity of two adjacent contours.

The control if one contour is reachable from another through the mesh is performed for each pair of contours from two subsequent level sets. As a result of constructing the contours and tracing the connectivity between them we obtain a graph structure (V, E) of the vertices and edges. However such graph representation is not unique for a 3D model, because the number of nodes and edges depends on the number of considered level sets. To avoid this ambiguity we perform graph smoothing. We keep the nodes of the graph representing the contours of critical values of the mapping function and remove the nodes representing regular values of the mapping function. A contour represents a maximum value of the mapping function if there is no any contour from the higher level set which can be reachable from it. Similarly, a contour represents minimum if it is not reachable from any contour of the lower lever set. A contour corresponds to a saddle critical point of the measuring function if it is adjacent to more then two contours from both higher and lower level sets. If a contour is reachable from two contours, where one of them is from the higher and the other is from the lower level sets, then the considered contour represents a regular value of the mapping function. While smoothing the graph, we remove regular nodes and edges incident to them. At the same time in order to preserve the topology representation we insert a new edge between the two nodes incident to the removed regular node. Figure 4.3 shows the result of graph smoothing. In this way we guarantee the topological equivalence of the 3D model in question and its Reeb graph. The final graph representation is unique, each node represents a critical point of the mapping function and the edges represent their adjacency.

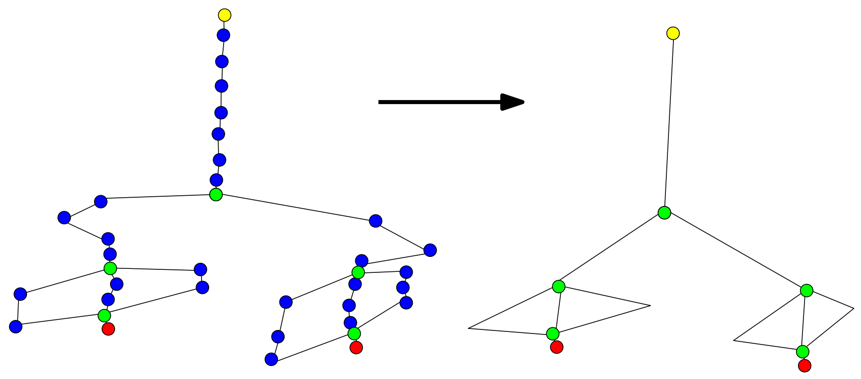


Figure 4.3: Smoothing the Reeb graph.

Constructing the Reeb graph for both model and its complement we are able to reveal all topological features of the shape. Moreover, the geometrical shape characteristics stored as labels of the nodes can be used to distinguish topologically equivalent models with different geometry.

To compute the complexity of construction of the Reeb graph we need to consider the complexity of contour insertion, parity check, mesh traversal and graph smoothing separately.

To make less time consuming the insertion of the contours inside the mesh, the edges can be ordered with respect to the value of the mapping function on the endpoints. The edge ordering requires $O(n \log n)$ operations, where n is the number of vertices composing the mesh. If we denote with n_C the number of vertices composing all inserted contours, then the number of checks for the edges to be split is $O(n+m) = O(\max(n,m))$. The number of edge intersections is equal to the number of the vertices composing the contours, i.e. $O(m)$. Therefore, the overall insertion of contours inside the mesh is $O(m+n\log n+\max(n,m)) = O(\max(n\log n, m))$.

The parity check method considers the contours from each level set, and checks for the intersection of the ray and each edge composing the contour. Hence, integrating overall contours the complexity of the parity check is $O(m)$.

While tracing the connectivity between contours and traversing the mesh, the visited triangles are marked in order to avoid lopping. Therefore, the complexity is $O(n)$ (the number of vertices, edges and triangles in the mesh are of the same order).

The complexity of graph smoothing depends on the number of regular nodes in the graph structure, which is of much lower order than the number of vertices n.

Finally, the whole process of Reeb graph construction is $O(\max(n\log n, m)) + O(m) + O(n) = O(\max(n\log n, m))$.

4.1.2 From Contour Driven to Component Driven Reeb graph

The construction of the Reeb graphs for a 3D model and its complement gives good description of the whole topological structure of the model. However, geometric information stored as weights of the nodes of the Reeb graph is not appropriate for the entire shape description. Usually geometric characteristics of the contours such as perimeter, area, curvature, texture are associated to the corresponding Reeb nodes and are used in further matching process. For models with simple topology a few number of contours is sufficient to reveal all topological changes. However the geometric characteristics calculated only for a few number of slices can be insufficient for the entire shape representation. In order to describe properly the topology of the shape together with its geometry we decided that a node of the Reeb graph should represent a section of the shape confined between two contours from consequent level sets. In this case the edges represent adjacency of two sections. Geometric characteristics stored as attributes of the Reeb graph are calculated for the entire section, which is a volumetric part and not a planar contour. This approach was also used in [13], [46], [92].

Representing a model with one graph which reveals all topological characteristics gives also an advantage when measuring similarity between two models. Obviously, the task of graph isomorphism detection is easier to perform if a model is represented by a unique connected graph than by two graphs where one of them can be disjoint, i.e. the graph representing the complement.

Keeping in mind these observations in the following section we propose the new approach for constructing Extended Reeb Graph, which nodes represent volumetric components of a model.

4.1.3 Component Driven Extended Reeb Graph

Extended Reeb graph is a graph which nodes represent the critical areas, i.e. the regions containing a critical point of the mapping function. Hence, the nodes of the ERG represent the volumetric components of a model unlike the nodes of the Reeb graph representing cross sections. The method for the ERG construction was first proposed in [21] and further studied in [20]. The novelty of our approach described in the current section consists in decomposition of a model into components where topological changes occur while maximizing cone- and cylinder-like components and minimizing branching parts. Our decision of such decomposition is determined by the fact that the shape of cone- and cylinder-like parts is easier to analyze than the shape of branching parts. Due to the same reason in [66] several geometrical shape characteristics are proposed for cone and cylinder-like segments while providing just a few for branching components. Constructing a shape descriptor which will be exploited in further retrieval process we need to find the best compromise between exhausting shape description and better shape compression. To this end, it is important to maximize the regions which can be easier described geometrically (called simple) and to minimize the regions difficult for the further analysis and representation (called complex). In this section we explain the segmentation process which decomposes a model into large cone- and cylinder-like components and small branching parts. In the next section we propose shape classification for the first two types of the components.

The idea of the shape segmentation consists in iterative bisection of the surface of a 3D model. The iterative bisection is represented in the tree structure "left child - right sibling" (see Figure 4.4). Connected components obtained after bisection of the model are stored as the child nodes in the tree. The choice of such representation of the segmentation is determined by the fact that we do not know in advance how many connected components we obtain after a bisection step. It means that we cannot forecast the number of child nodes created at each iteration. We decide also to associate to a node the subpart of the triangle mesh representing the corresponding segment. By doing this we simplify the further process of shape analysis which is described in the following section. In addition, we associate to a node of the tree the list of indices of boundaries of the corresponding segment which will be useful during the further process of simplification of the segmentation.

In the beginning the whole model represents the root of the tree. We define the initial partition of a model M into K intervals by inserting $K-1$ contours.

$$M = (M_{min}, M_1) \bigcup (M_1, M_2) \bigcup \dots \bigcup (M_{K-1}, M_{max})$$

The difference of the value of the mapping function f on the boundaries of each interval in this case is $\triangle f = \frac{f_{max}-f_{min}}{K}$. The process of the segmentation of the model is equivalent to its decomposition into a set of disjoint components. When a contour is inserted into the mesh, the constituting

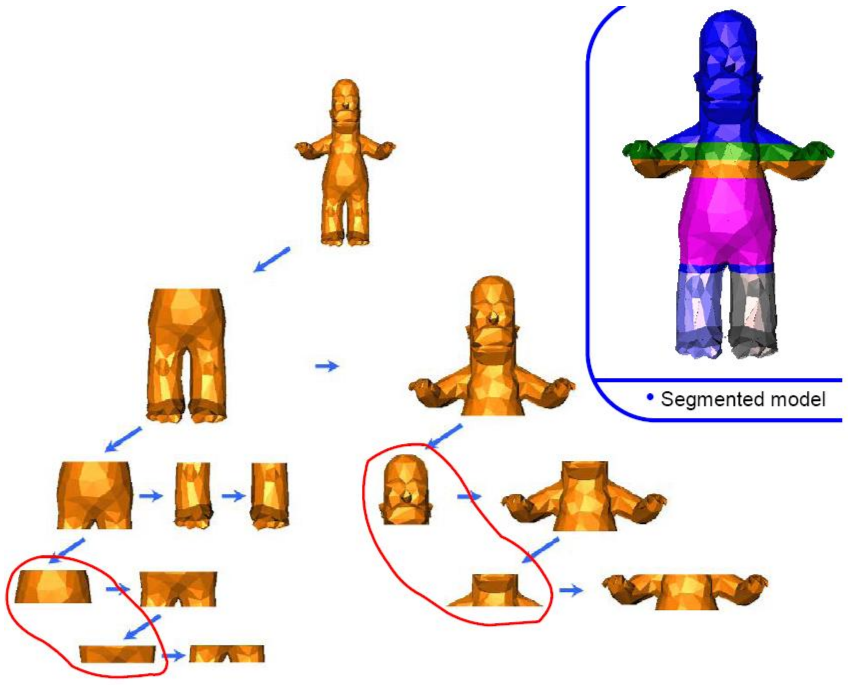


Figure 4.4: Iterative segmentation and postprocessing merging. The model is taken from AIM@SHAPE Shape Repository [5].

edges and vertices are labeled with the ordinal number of the contour. For the decomposition of the model we require that each segment has its own boundary. To achieve this, vertices and edges composing the boundary should be duplicated, so that each vertex, edge and triangle belong only to one segment. In this way we decompose a model into the set of disjoint segments. The connectivity relations between edges and the endpoints as well as between triangles and the edges should be updated.

When a model is decomposed into a set of disjoint components we check each of the components if it has a non-zero genus or represents a saddle region. Such components are classified as complex regions. We calculate the genus of each component as $g = \frac{E-V-T-B+2}{2}$ [60], where E, V , T, and B are the number of edges, vertices, triangles, and boundaries correspondingly. The components representing saddle regions have more than three boundaries from two different level sets. We continue bisecting complex components until the difference of the measuring function $\triangle f$ on the boundaries of the component reaches the predefined threshold. Components obtained at each segmentation step are stored as child nodes of the complex bisected component. In order to make more efficient the use of memory, we do not keep the triangle mesh representing the processed component, because now it can be represented as the union of the triangulations of its children. The iterative bisection terminates when all components associated to leaf nodes of the segmentation tree are classified as simple or have the size less then the predefined threshold.

Figure 4.4 shows that after segmentation several adjacent components with less then three boundaries can be generated. These components can be merged on the post-processing step. To this end, we create the list of the leaf nodes representing simple components. The possibility to merge these components is verified for each couple of nodes in the list. Two components can be merged if they have a common boundary or, in other words, if the intersection of their lists of boundary indices B_i is not empty

$$B_i \cap B_j \neq \emptyset. \tag{4.1}$$

After the process of the segmentation and simplification is terminated, each of the components is represented by a node in the ERG. The adjacency of the components is revealed by the edges in the ERG. Precisely, there is an edge between two nodes in the ERG, if the intersection of the lists of boundary indices of the corresponding components is not empty (4.1), or in other words, if these components share a common boundary. The shape of each components of the ERG can be classified as cone-, cylinder-like or branching which corresponds to the components with one, two and more boundaries. Cone-like components represent the regions of maximum or minimum values of the mapping function. Cylinder-like components represent regions of regular values of the mapping function. The components with two boundaries may as well represent degenerate or handle-like maximum or minimum regions, what can be easily detected controlling the values of the mapping function on the boundaries. However, the shape analysis described in the following section for handle-like and degenerate maximum regions with two boundaries is the same as for cylinder and cone-like components correspondingly. The components with branching shape are those having more than two boundaries, they represent saddle or degenerate minimum or maximum regions.

Here we would like to underline that the resulting ERG contains the nodes representing regular regions of the mapping function. Such regions may be revealed in the ERG only if they are adjacent to two saddle regions. This is done with the goal to avoid multiple edges in the ERG. The graphs without multiple edges can be represented by more sophisticated matrix structure which can be exploited in further spectral graph matching.

Iterative bisection of a 3D model increases the number of generated segments but at the same time minimizes the complex areas. The merging step reduces the number of the created segments by enlarging the size of simple areas. As the result of such segmentation the model is divided into large simple and small complex components. The segmentation described in this section guarantees that all topological characteristics of a shape are revealed in the ERG, unless they are concentrated in relatively small areas with respect to the overall size of the model.

An additional advantage of the proposed segmentation is that it directly implies the possibility of parallel implementation, i.e. the iterative process of bisection of saddle regions can be performed simultaneously for each of the regions.

To compute the complexity of shape segmentation and ERG construction, we need to consider the complexities of each of steps separately. These steps are contour insertion, mesh division, extraction of connected component, check for adjacent simple components and their merging.

Let us denote with n the number of vertices composing the original mesh, and with m the number of vertices composing all the contours. We define n_i as the average number of vertices composing a branching component, and m_i as the average number of vertices composing the contours of one level set, so that $\sum_{i=1}^{N_{ls}} m_i = m$. We denote with s the number of saddle regions of the model, with $Iter$ the number of iterations needed to perform before the bisection process terminates, and with ls_0 the initial number of considered level sets used to insert seed contours inside the original mesh.

Similarly to the complexity of contour insertion discussed in Section 4.1.1, we compute the complexity of iterative contour insertion. In the beginning we consider ls_0 number of level sets of the mesh composed by n vertices, this requires $O(m_i ls_0 + n \log n)$ operations. Further, we continue to insert the contours inside branching regions until the iterative process terminates. It corresponds to $s \cdot Iter$ insertions of contours inside saddle region, where each insertion has the complexity $O(m_i + n_i \log n_i)$. Therefore the overall complexity of the iterative insertion of contours requires operations $O(m_i l s_0 + n \log n + sIter(m_i + n_i \log n_i))$. We would like to make a remark here about the dependency of the method on the number of saddle regions. When a 3D model has a simple topology with no branching regions, the iterative insertion of contours is just $O(m_i l s_0 + n \log n) = O(\max(n \log n, m))$. When a model has many saddle regions the more iteration are required to bisect them, hence the complexity is determined by the second term $O(sIter(m_i + n_i \log n_i))$.

The process of mesh division consists of duplicating the vertices and edges of contours and updating the adjacency relation of the incident triangles. Therefore the process depends on the number of vertices composing the contours and has the complexity $O(m)$.

Extraction of connecting components after mesh division requires traversing all triangles of the mesh which has $O(n)$ complexity (the number of triangles are of the same order as the number of vertices). Similarly, to iterative contour insertion, the number of extractions of connected components depends on the number of branching regions and on the number of iterations needed to perform before the segmentation terminates. Therefore, the complexity of the whole process of extraction of connected components is $O(n + sIter \cdot n_i)$.

After the iterative segmentation terminates, all extracted simple components are checked on the possibility to be merged. Merging of mesh components is inverse to mesh division and is linear on the number of vertices composing the boundary. Each couple of simple components is checked on the possibility to be merged, which in the worst case leads to the complexity of merging $O(N_{sc}^2 m_i)$, where N_{sc} is the number of simple components extracted after segmentation.

Finally, the total complexity of the iterative segmentation and merging has the complexity $O(m_i l s_0 + n \log n + sIter(m_i + n_i \log n_i)) + O(m) + O(n + sIter \cdot n_i) + O(N_{sc}^2 m_i)$, which is after simplifying

$$O(\underbrace{(m_i l s_0 + n \log n)}_{Initial\ segmentation} + \underbrace{sIter(m_i + n_i \log n_i)}_{Iterative\ segmentation} + \underbrace{N_{sc}^2 m_i}_{Merging})$$

. Here, the complexity of the segmentation of models without branching regions, for which $N_{sc} = ls_0 + 1$, will be determined by terms $O(n \log n + ls_0^2 m_i)$. The complexity of the segmentation of models with many branching regions will be determined by the term $O(sIter(m_i + n_i \log n_i))$.

4.2 Geometric Characteristics and Shape Analysis

The ERG constructed in the previous section represents the topological structure of a 3D model, hence it can be used as a rough filter for dividing models into topologically equivalent classes. Using the ERG as a shape descriptor in retrieval process we can find models with the same topology but having different geometrical characteristics. In order to distinguish both topology and geometry of different 3D models we can analyze locally the geometry of the components of a model, which are represented by the nodes in the graph. The revealed geometrical information can be stored as labels of the corresponding nodes and can be used in similarity estimation between different shape descriptors.

The Reeb graph enriched with geometrical information was proposed in [46], [13], [92], [22], [17]. In [66] the authors segmented 3D models into cone-, cylinder- and branching parts and they proposed several geometrical shape characteristics for the first two types of segments while providing just a few for the third type. In [17] the authors assign the weight to each node of the topological graph which encodes the shape of level curves defined on a topologically homogeneous part of a 3D model. However, the aim of such encoding is shape reconstruction and not shape retrieval.

We propose to use several non-dimensional measures to describe the overall geometry of a segment, and also to perform detailed shape analysis for cone- and cylinder- like parts. These shape characteristics together with the topological Reeb graph can be used further as the shape descriptor in the retrieval process.

Each component of a model obtained after segmentation can be characterized by the following values:

- the number of boundary components N_b;

- the nondimensional measure of segment's weight in the whole model

$$SW = \frac{A_C}{A_M} \tag{4.2}$$

defined by the relation of the surface area of the component A_C and the surface area of the whole model V_M. This value can be used in graph matching to estimate the influence of a node on the whole graph structure, i.e. bigger segments have greater impact;

- convexity measure $Conv = \frac{A_C}{A_{CH}}$ which is defined as the ration of the surface area of the component A_C to the surface area of its convex hull A_{CH} [28];

- hull packing $H_p = 1 - \frac{V_C}{V_{CH}}$ defined as the convex hull volume V_{VH} not occupied by the volume of the component V_C [28];

- compactness measure $Comp = \frac{{A_C}^3}{{V_C}^2}$ defined as the ratio of the surface area cubed over the volume of the component squared.

For cone- and cylinder-like components we propose to perform more detailed shape analysis. The main idea of the proposed shape analysis scheme is that the components with one and two boundaries have two reference points which define the main axis of the component. For the components with one boundary, the first reference point is the tip of the component, which is the point with the maximum/minimum value of the measuring function. The second reference point is the barycenter of the boundary component. For the components with two boundaries, both reference points are the barycenters of the boundaries. The two reference points define the line which we propose to use as the main axis for further shape analysis. Further, intersecting the component with equally spaced planes perpendicular to the main axis we obtain the set of contours. The analysis of the evolution of these contours gives us advanced information about the shape of the component. We provide three tables for shape analysis for components both with one and two boundaries. The tables contain the name of a component and the image of its standard shape with corresponding characteristics.

Table 4.1 gives the description of component's taper/enlargement through tracing the changes of the area of the cross sections. If the area of the cross sections is constant to the extent of a predefined threshold then the component belongs to the first type of the proposed classification. If the cross area changes (increases or decreases) we analyze the average speed of the cross area alteration. Suppose we locate $N-1$ cross sections at equal distances along the main axis of a model component M. The inserted contours define N equal intervals

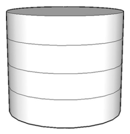
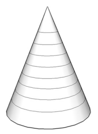
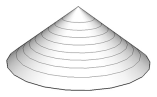
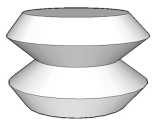
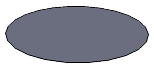
$$(M_{min}, M_1)\bigcup(M_1, M_2)\bigcup\ldots\bigcup(M_{N-1}, M_{max}).$$

We calculate the average cross area alteration and we define that the alteration is smooth if the apex angle of the induced cone or pyramid is less than 90°, which corresponds to the inequality $\frac{r_i}{h_i} < 1$, where r_i is the average radius of the i-th cross section and h_i is the height of the corresponding cone. The area of the cross sections increases/decreases sharply in case when $\frac{r_i}{h_1} > 1$. If the alteration of cross areas changes its behavior, for example increasing and then decreasing, then we define the component as having multiple area alterations. If the main axes has almost zero length, the component is flat, which corresponds to the case when the apex angle of the induced cone is 90°.

Table 4.2 gives the description of component's bending, specifying if it is straight or curved. In order to classify a component according to its bending we keep track of the position of the main axis with respect to each cross section. If the axis passes through all sections then the component is straight. If the axis is located outside several consequent sections then the component has single bending. In the case when the axis is outside two or more sections while passing through the sections located in between, the component has multiple bending.

Table 4.3 classifies components with respect to the average curvature of each cross section. We distinguish only three kinds of cross sections, namely sections having the average curvature $60^\circ, 90^\circ$ and more degrees. If the curvature of the cross sections is preserved along the main axis, then the component has the constant curvature, otherwise it has curvature alteration.

Table 4.1: Analysis of a component's taper/enlargement

	Constant cross area	Cross sections have approximately equal areas
	Smooth increasing\ decreasing cross areas	The area of cross sections changes smoothly increasing or decreasing
	Sharp Increasing\ Decreasing cross area	The area of cross sections changes sharply increasing or decreasing
	Multiple cross area alterations	The area of cross sections changes increasing and decreasing, characterized by the number of alterations
	Flat component	The length of the main axes is almost zero

Shape classification given in Tables 4.1, 4.2 and 4.3 is not exhaustive and can be extended according to the precision of the retrieval system. The intersection of the three defined categories produces 76 kinds of shapes which can be used for rough approximation of each segment of a model. Figure 4.5 demonstrates the example of such approximation.

The final topological graph augmented with the geometrical characteristics can be stored in the following structure

$$\begin{aligned}(V,E) \;=\; &(\{(N_{bi}, SW_i, H_{cp_i}, H_{p_i}, H_{ci}, Class1_i, Class2_i, Class3_i) | v_i \in V\},\\ &\{(v_i, v_j) | (v_i, v_j) \in E\}).\end{aligned}$$

In the next section we describe the process of similarity estimation between two different shape descriptors, where the shape descriptor is represented by the ERG enriched with local geometrical information.

Table 4.2: Analysis of a component's bending

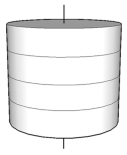
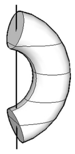
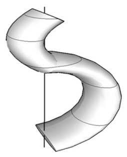
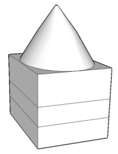
	No bending	The main axis of the component passes through all cross sections
	Single bending	The medial axis passes through the cross sections which are close to the reference points
	Multiple bending	The main axis passes through the cross sections occasionally. Characterized by the number when the main axis is outside the cross section

Table 4.3: Analysis of component's curvature alteration

	Constant curvature	The average curvature is preserved throughout all sections. The single curvature value characterizes the whole component
	Curvature alteration	The average curvature changes along the component. The component is characterized by the values of curvature in the vicinity of reference points

4.3 Graph Matching

In the current section we propose a new method for matching the ERGs enriched with local geometric information. The method uses spectral properties of the Hermitian matrix representing the ERG. Imposing several constraints on the elements of the Hermitian matrix, we are able to mimic the behavior of the Laplacian matrix, known for its wide application in graph matching and partitioning.

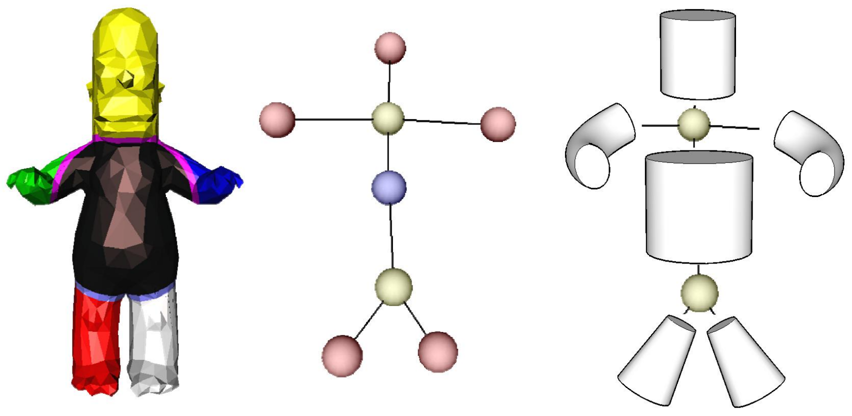


Figure 4.5: Rough approximation of a model as the result of shape classification.

4.3.1 Shape Representation for Shape Retrieval

To represent a 3D model during retrieval we use the ERG constructed in Section 4.1.3. The nodes of the graph represent volumetric parts of the model which can be classified as simple (having one and two boundaries) and complex (having more than two boundaries). Due to the iterative segmentation and further merging, two simple components can never be adjacent, otherwise they should be merged.

When a model has simple topology, as for example the model on Figure 4.6, after segmentation it is represented by two simple adjacent components, one minimum and one maximum regions. If we merge these components, then the model will be represented by a single node, and the matrix representing the empty ERG will just a zero 1×1 matrix. To improve the descriptiveness of the ERG we do not merge these two simple components, but insert in the middle a regular interconnecting node.

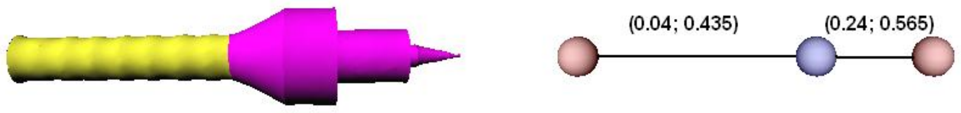


Figure 4.6: Segmentation of a model without saddle regions and its ERG.

Similarly, two saddles can never be adjacent. Iterative bisection of branching regions leads to division of saddle points concentrated in the region and as a consequence to the insertion of regular regions between two saddles. A complex saddle region is not separated and as the consequence is represented by a single node in the ERG only when saddle points are concentrated in a small area which size is less then the predefined threshold.

In the ERG representing a 3D model simple nodes are connected to saddle nodes and vice versa. An edge in the ERG is always incident to one simple or interconnecting node. This allows to attribute an edge with the shape characteristics of the incident simple region proposed in Section 4.2. Figure 4.7(c) illustrates the ERG with the attributed edges after segmentation (a) and merging (b) phases.

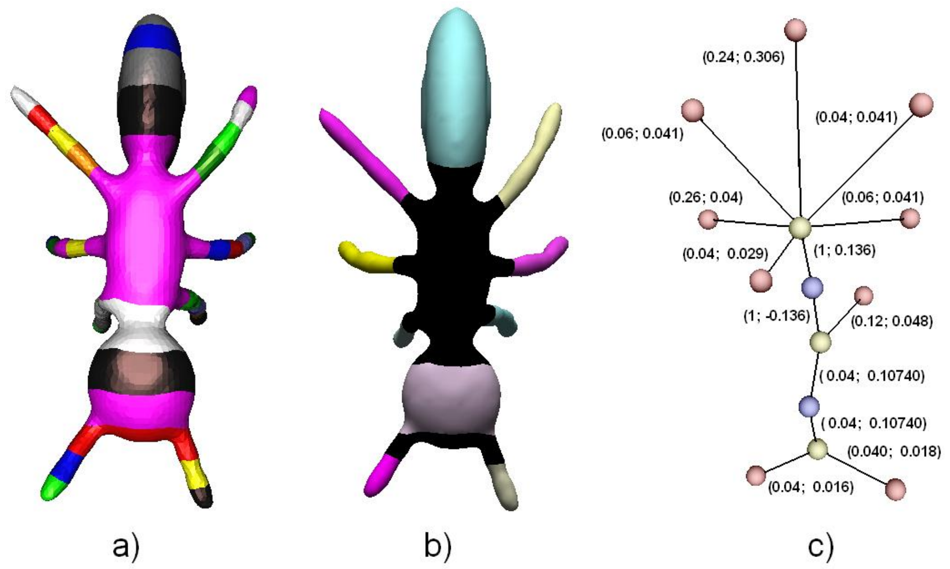


Figure 4.7: ERG construction. (a) Shape segmentation. (b) Merging simple adjacent components. (c) ERG with edge attributes: shape index and segment weight.

4.3.2 Graph Representation

After shape segmentation and shape analysis each 3D model is represented by the attributed ERG. To estimate the similarity between the graphs we follow the ideas of [97] and propose to associate the Hermitian matrix which encodes edge attributes together with graph connectivity. By imposing several constraints on the elements of the Hermitian matrix we succeed to mimic the spectral properties of the Laplacian matrix.

A Hermitian matrix is the square matrix that is equal to its conjugate transpose, i.e. $H_{ab} = \overline{H_{ba}}$, where $\overline{H_{ba}}$ denotes the conjugate of the complex number H_{ba}. The complex entries of the Hermitian matrix can be written in the polar form as $H_{ab} = -w_{ab}e^{iy_{ab}}$, where w_{ab} is the magnitude and y_{ab} is the phase of the complex number. To mimic Laplacian spectral properties the magnitude w_{ab} of the complex entries should satisfy the conditions (3.7-3.9). A phase of the complex number should satisfy the following conditions:

$$y_{a,b} = -y_{b,a} \tag{4.3}$$
$$-\pi < y_{a,b} < \pi \tag{4.4}$$

The first condition (4.3) ensures that H is equal to its own conjugated transposed matrix. By obeying the second constraint (4.4), phase wrapping can be avoided.

To make the sum of magnitudes of the entries of each row sum up to zero we set the on-diagonal entries to be real numbers defined as:

$$H_{aa} = \sum_{b \neq a} W_{a,b} \tag{4.5}$$

The Hermitian matrix defined in this way mimics the behavior of the Laplacian matrix from spectral point of view. Precisely, the spectrum and the eigenvectors of the Hermitian matrix can be used for graph partitioning and graph matching.

We now choose the graph attributes which could be encoded in the Hermitian matrix and thus should satisfied the imposed constraints.

Magnitude of Hermitian elements. As was described in Section 4.3.1, an edge in the graph connects a saddle node and a simple node. As a consequence, the calculated shape characteristics of the simple segment can be stored as the attributes of the edge incident to the corresponding simple node. We decide to use indices of the shape classification described in Section 4.2 as the magnitude of the entries of the Hermitian matrix.

The values of shape indices are chosen based on visual shape similarity as well as on the alteration of criteria used for shape analysis. Table 4.4 illustrates the list of indices of all possible shape types produced by combining cross area and bending criteria. As it can be seen from the

Cross Area	Bending	Shape Index
constant	zero	0.04
constant	single	0.06
constant	multiple	0.08
alteration	zero	0.12
alteration	single	0.14
alteration	multiple	0.16
smoothly decreasing	zero	0.24
smoothly decreasing	single	0.26
smoothly decreasing	multiple	0.28
smoothly increasing	zero	0.32
smoothly increasing	single	0.34
smoothly increasing	multiple	0.36
flat		0.64
sharply decreasing	zero	0.68
sharply decreasing	single	0.7
sharply decreasing	multiple	0.72
sharply increasing	zero	0.76
sharply increasing	single	0.78
sharply increasing	multiple	0.8
body		1

Table 4.4: Shape indices.

table, the variation of the cross area criterion has a greater influence on the difference in shape index than the bending criterion. This choice can be explained by the fact that semantically equivalent subparts of articulated models frequently have different bending properties, e.g. straight and bended arms. The curvature criteria was not considered in the choice shape indices because working with articulated models it does not produce distinct shape types. However, working with CAD models this shape criteria can be involved and more shape indices can be defined.

Mapping the shape indices to the interval $(0; 1]$ allows their use as magnitude of complex entries of the Hermitian matrix. The positive shape index of the segment is invariant to edge direction thus satisfying the constraints (3.7)-(3.9).

Phase of Hermitian elements. The segment weight SW (4.2) can be used as the phase value y_{ab} for the complex entries of the Hermitian matrix. To obey the antisymmetric condition (4.3) we set y_{ab} as:

$$y_{ab} = \begin{cases} SW_{ab}, & \text{if } deg(a) > deg(b) \\ -SW_{ab}, & \text{if } deg(a) < deg(b) \end{cases}$$

The value of SW_{ab} is bounded by the interval $(0;1)$. Scaling this interval up to $(0;\Pi)$ and using the above definition of y_{ab} we satisfy the constraint (4.4) for the phase value of the complex entries of the Hermitian matrix.

4.3.3 Fiedler Vectors for Graph Matching.

The spectrum of the defined Hermitian matrix mimics the behavior of the Laplacian matrix from the spectral point of view. Precisely, the spectrum of the Hermitian is equal to the spectrum of the Laplacian matrix which entries are the magnitudes of the complex Hermitian entries. Consequently, the information stored in the phases of the complex elements is not preserved in the Hermitian spectra. The eigenvectors of the Hermitian matrix are complex-valued vectors which encode the graph attributes stored both in magnitudes and phases of the elements of the Hermitian matrix. We propose to use the eigenvector of the Hermitian matrix associated to the second smallest eigen value. The corresponding eigenvector for the Laplacian matrix is called Fiedler eigenvector. We preserve this name for the Hermitian matrix as it is analogous to the Laplacian one from spectral point of view.

Differently from the matrix spectrum, the eigenvectors are not permutation invariant. In order to use the Fiedler vector as the graph signature in the retrieval process we, first, order its entries lexicographically. Precisely, suppose a graph has n nodes, then the Fiedler vector of the Hermitian matrix is n-dimensional complex-valued vector. We order this vector lexicographically and then we interlace the real and imaginary parts in the following way:

$$\begin{pmatrix} a_k + ib_k \\ a_l + ib_l \\ \vdots \\ a_n + ib_n \end{pmatrix} \quad \rightarrow \begin{pmatrix} a_l \\ b_l \\ a_k \\ b_k \\ \vdots \\ a_n \\ b_n \end{pmatrix} \tag{4.6}$$

if $a_l < a_k$, or $a_k = a_l$ and $b_l \leq b_k$.

As the result we transform n-dimensional complex-valued into $2n$-dimensional real-valued vector which we use for the retrieval process.

The similarity between two models represented by the graphs of the same size is computed as the Euclidean distance between the transformed Fiedler vectors. However, in practice the graphs representing even similar models are of different size which leads to the Fiedler vectors of different dimensions. Working with the graph indexing using spectra it is a common approach to pad with zeroes the spectra of a smaller size till the size of the largest spectrum. This technique is equivalent to inserting dummy disconnected nodes in the graph, and consequently increasing the multiplicity of the smallest, i.e. zero, eigenvalue.

Following this approach we pad with zeroes the smaller transformed Fielder vector to the size of the Fiedler vector of the bigger graph. The similarity between two models represented by the attributed graphs is calculated as the distance between the Fiedler vectors of the same size:

$$d_{\text{full}}(G1, G2) = \sqrt{\sum_{i=1}^{2n} (F_1(i) - F_2(i))^2} \tag{4.7}$$

where F_1 and F_2 are the sorted, interlaced complex Fiedler vectors of graphs $G1$ and $G2$ respectively.

The described techniques is sensitive to the different level of detail which can be revealed during the segmentation phase. Suppose two similar models are represented by similar graphs of different size. This may happen because of more details present in the shape of one of the models. As the result a larger distance will be found between the corresponding Fiedler vectors as the result of the distance calculation between dummy and real nodes. Moreover, matching the Fiedler vectors of two whole graphs does not take into account partial matching.

Still, Fiedler vector as the graph signature give good results for the overall graph matching. Following these considerations we propose to use Fiedler vector both for overall and partial matching. In the next section we describe how Fiedler vector is employed in partitioning the graph into non-overlapping subgraphs. Further the combination scheme for overall and partial graph matching is proposed in Section 4.3.5.

4.3.4 Fiedler Vectors for Graph Partitioning.

A common way to calculate partial similarity is to construct all possible subgraphs of the graphs to be matched and to compute similarity between the subgraphs of the same size [62, 31]. This approach, however, leads to the exponential growth of all possible constructed subgraphs, and as the consequence to the excessive computation of similarity between them. Moreover, some of the possible subgraphs could be semantically meaningless. Figure 4.8 illustrates a model of a dog and the corresponding ERG together with two subgraphs, where the first subgraph represents the front part of the model, and the second subgraph represents the interconnection of different parts of the model without reflecting semantics of any single part. In [31] the authors reduced the complexity of similarity computation for the cases of integral spectra variation [86, 53]. Unfortunately, in practice the integral spectra variation during extraction of all possible subgraphs happens quite rarely.

In this section we follow the idea of [75] where the authors use the Fiedler vector of the Laplacian matrix for graph partitioning into non overlapping supercliques. The superclique is composed of a central node an its immediate neighbors. A central node is a node with the highest assigned importance, where the importance of the node proposed in [75] depends on the value of the corresponding entry in Fiedler vector, the degree of the node and the position of the node in the graph. The nodes which are located closer to the graph perimeter and have higher degree and Fiedler

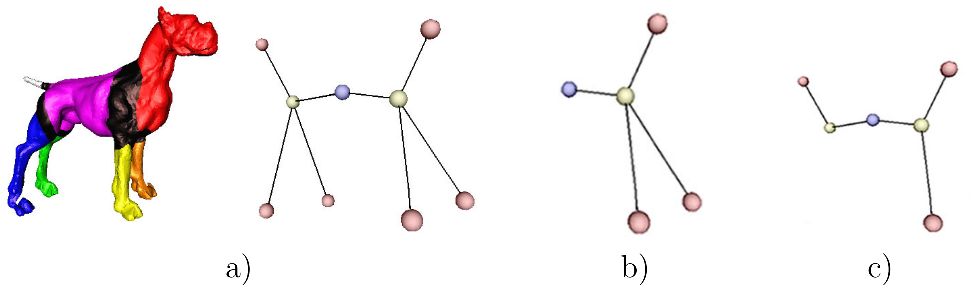


Figure 4.8: Subgraphs of the ERG. a) A segmented model of a dog and the corresponding ERG. b) The subgraph representing the front part of the dog. c) The meaningless subgraph.

entry gain more importance and, thus, are chosen as central nodes.

Considering that the ERG are the graphs often without edges which could constitute the graph perimeter, we assign the importance of the node depending on its degree and the value of the corresponding Fiedler entry of the Hermitian matrix.

The main steps of the graph partitioning are the following:

1. Sort the entries of the complex Fiedler vector lexicographically in decreasing order $\Pi = (a_1 + jb_1, a_2 + jb_2, \ldots, a_n + jb_n)$, where $a_1 > a_2$, or $a_1 = a_2$ and $b_1 \geq b_2$.

2. Associate a *score* with every node in the graph: $F(i) = \alpha \times \text{degree}(i) + \beta/\text{rank}(i)$, where $\text{rank}(i)$ is the position of the node in the sorted Fiedler vector Π, and α and β are two balancing factors. For the experiments reported in Chapter 5 we used the balancing factors $\alpha = \beta = 0.5$

3. Traverse through the list of the sorted Fiedler vector Π and select center nodes. A node is a center node if its score is higher than the scores of all its neighbors.

4. Remove the center node and its adjacent nodes from the list; together they form a subgraph. Continue processing the graph this way until all nodes are in a subgraph.

The described procedure partitions the graph in non-overlapping meaningful subgraphs. For an example, see Figure 4.9, where a segmented model

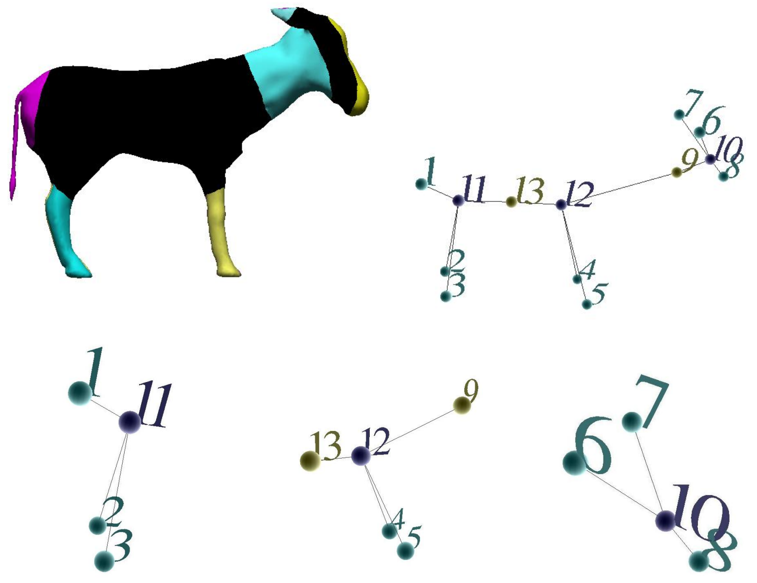


Figure 4.9: Subgraph decomposition of a model of a goat.

of a goat is displayed, together with a view of its 3D graph and extracted subgraphs. The first extracted subgraph corresponds to the back (tail and back legs), the second subgraph corresponds to the middle part (body, neck and fore legs) and the third subgraph corresponds to the front part (head, nose and ears).

To be able to capture small alterations in the structure of subgraphs we introduce the fifth step which decomposes yet more the subgraphs extracted using the above procedure.

5. Decompose each subgraph even further by reporting every possible combination of the center node and its adjacent nodes of size 2 up to the size of the original subgraph.

Finally, each of the extracted subgraphs is represented by their transformed Fiedler vector (4.6). The distance between two graphs g_1 and g_2 is then defines as the average of the weighted pairwise subgraph distances. The subgraph distances are calculated only between the subgraphs of the same size and are weighted with the corresponding subgraph size:

$$d_{\text{sg\_d}} = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} d_{\text{full}}(g_1^i, g_2^j) \tag{4.8}$$

where g_1^i and g_2^j are the $i - th$ and $j - th$ subgraphs of size d of graphs g_1 and g_2, and n and m are the number of subgraphs of size d that were extracted from g_1 and g_2.

A subgraph distance contributes more to the total similarity if the size of the subgraphs is larger. The total distance between graphs g_1 and g_2 based on their subgraphs is defined as

$$d_{\text{sg}}(g_1, g_2) = \frac{1}{\text{MaxD} \times (|g_1| + |g_2|)} \sum_{d=1}^{MaxD} d \times d_{\text{sg\_d}} \tag{4.9}$$

where $|g_1|$ and $|g_2|$ are the number of nodes in g_1 and g_2 respectively, and where $MaxD$ is the smallest maximal subgraph that could be decomposed from either g_1 or g_2.

Using this distance function, a ranked list of all the database objects can be produced for a given query. Together with the ranked list based on the comparison of complete graphs described in Section 4.3.3, two complementary ranked lists are produced for one query. These two ranked lists can be combined to produce better retrieval results which capture both overall and partial similarity. In the next section the we propose the way to combine two ranked lists.

4.3.5 Combination of Two Ranked Lists

There is a large amount of research around combination of similarity measures or ranked lists produced by different techniques which aims at the best retrieval performance. In [25] the authors experimented several combination schemas of distance measure produced by different shape descriptors. The first schema assumes that all shape descriptors produce equally important ranking lists, thus the resulting combined distance measure between a query and an object $d^c(q, o)$ is an unweighted sum of normalized distances computed using different shape descriptors:

$$d^c(q,o) = \sum_{i=1}^{N} b_{C_i} \frac{d_i(q,o)}{dmax_i(q,o)},$$

where N is the number of the shape descriptors, $d_i(q,o)$ is the distance measure produced by the i-th shape descriptor, and b_{C_i} is a binary coefficient which indicates if the i-th shape descriptor participates in the combination c of the descriptors.

The other schema studied in [25] assumes that a shape repository is preclassified and hence a relevant measure $purity(f_i, q, k)$ can be calculated for each shape descriptor f_i. This measure indicates the maximum number of relevant models present among first k retrieval results using f_i descriptor. The combined distance measure in the second schema is the weighted sum of all distances:

$$d^c(q,o) = \sum_{i=1}^{N} (purity(f_i, q, k) - 1) \frac{d_i(q,o)}{dmax_i(q,o)},$$

In [64] the authors combined the ranks produced by estimating textual and shape similarity. Four combination schemas were experimented: linear weighted average combination, minimum matching score, minimum rank, and confidence limits method. The combination method which uses the minimum normalized score improved the retrieval performance by 5.8%.

In Sections 4.3.3 and 4.3.4 we have defined two distance functions (4.7) and (4.9). The values of these distances can vary greatly, because each of the distances depends on the value of entries of the graph spectrum, which in its turn depends on the number of nodes in the graph. Hence the value of the distance of the overall graph matching is usually much higher than the value of the distance for the partial graph matching. Observed this, we decide to combine the ranks produced by the two distances, and not distances themselves. The two ranked lists appears very complementary, i.e. the retrieved models relevant to the query are usually on the top of one of the ranked lists. Therefore, combining the ranked lists we would like to increase the influence of the higher and to damp the influence of the lower ranks. The range of the values

If we denote with $\mathrm{r_{full}}(\mathrm{i})$ the position of the i-th model in the ranked list produced using the distance for complete graph matching, and with $\mathrm{r_{sg}}(\mathrm{i})$ the position in the ranked list for partial graph matching, then we combine the two ranks:

$$\mathrm{score}(i) = \log(\mathrm{r_{sg}}(i)) + \log(\mathrm{r_{full}}(i)).$$

The value $\mathrm{score}(i)$ defines the position of the model i in the new list of retrieved items. The logarithms in the formula decrease the influence of lower ranks on the final score.

The results of the proposed in this section shape representation, matching and retrieval techniques are described and discussed in the following chapter.

4.4 Semantics and Shape Descriptor

In this section we propose the preliminary methodology to relate geometrical and topological shape characteristics to shape semantics. The research in the field of knowledge structuring suggests to use ontology for describing the knowledge of a chosen domain [26]. The author of [42] defines ontology as a specification of a representational vocabulary for a shared domain of discourse which may include definitions of classes, relations, functions and other objects. Therefore, if we know the domain in which the 3D shapes are constructed, the ontology of the domain can be built. Then mapping between low level features and ontology concepts is performed. Finally, 3D shapes are annotated and become well-defined structure under human-perspective.

In [59, 63] the authors build the vocabulary that maps the low level geometrical description of regions on an image to semantic labels. The considered geometric characteristics are shape, position, color and texture of a region. The vocabulary represents the ontology of objects and scenes which might be present on an image. In [58] the wide visual concept ontology was proposed to map domain knowledge to low-level vision numerical descriptors. In [91] the ontology of 3D shapes was introduced with the purpose to fill the gap between knowledge-level representation of shapes in the human mind and their low-level shape representations.

Within the Network of Excellence Aim@Shape the ontology of shapes has been developed. The purpose of the ontology is to integrate concepts and properties which can be used to annotate the models in Aim@Shape shape repository and at the same time be present in the developed domain ontologies. The authors of [36] developed a tool, called TopMesh, which automatically extracts topological properties of the shape formalized in Shape Acquisition and Processing ontology.

In this section we propose the methodology for relating the distribution based shape descriptor [69] to the semantic labels of s chosen domain, that is the domain of furniture models. In the next subsections we, first, state several assumptions about the models of the chosen domain. Second, we explain which shape descriptor is used to represent a model on low level. Third, we describe how low-level shape descriptors can be mapped to semantic labels. Next, we show how a shape can be retrieved using the domain ontology. We conclude the section discussing the possibilities to extend the proposed methodology to other domains using other shape descriptors.

4.4.1 Assumptions on Shape Representation

The fist assumption which we would like to take regardless the chosen domain is that a model can be completely described by connectivity relations between its constituents and their shape. This assumption means that the color and texture of a model do not influence its semantic. Resting the domain of models to the furniture one allows us to assume that models are created using Constructive Solid Geometry (CSG) approach. Thus the furniture models are assemblies of meaningful atoms that are similar to geometric primitives. To prove that this assumption does not constrict too much the number of 3D models which can be used in the proposed approach we performed a search of 3D furniture models in Internet. We have downloaded 98 furniture models from Princeton Shape Benchmark [11] and Free Stuff of 3D Cafe [8]. After examining them we found that 63% of furniture models are compound models (here we notice that 88% of models from Princeton Benchmark are compound), and 75% of compound models are models composed from geometrical primitives. We suppose that these figures can increase when a 3D database is created by designers from the same industrial domain. Consequently, our second assumption will be valid for the majority of CAD models, because assembly modelling is effective approach, which allows designers to work together on a complex model and gives a possibility of the further reuse of designed objects.

4.4.2 Shape Descriptor

Assuming that the models from furniture domain are compositions of conceptual parts, we start the analysis of a model from its decomposition into the constituents. We load the triangle mesh, representing the given model, and then we decompose it into connected components. Next, we analyze the shape of each constituent of the model using the approach suggested in [69]. For each constituent we construct the vector of shape distribution. The choice of the distribution-based shape descriptor is determined by simplicity of construction, invariance to affine transformations and good discriminative results for the models similar to geometrical primitives, like cubes, spheres, cylinders, etc [48]. According to the assumptions stated in Section 4.4.1, we consider the models that are the compositions of geometrically simple objects. Hence we can build the finite set of the geometric primitives, which can be used to construct CAD models. For each of such geometrical primitives we extract the distribution based shape descriptor, and we label primitives with the corresponding name. The phase of the construction of the database of geometrical primitives and labeling them with corresponding names is done manually. The number of geometrical primitives which can be used for the composition of furniture model is finite, thus once the database has been constructed it can be used for annotating the constituents of a model without user intervention. Figure 4.10 illustrates some geometric primitives which can be used as atoms for designing furniture models, and their shape descriptions. After the decomposition of a given 3D model into constituents, we compare each part with geometrical primitives from Figure 4.10. The smallest distance between the shape distribution vectors identifies the shape of the analyzed constituent. The constituent inherits the label of the most similar geometric primitive. The process continues for all parts of the model. As a result the shape de-

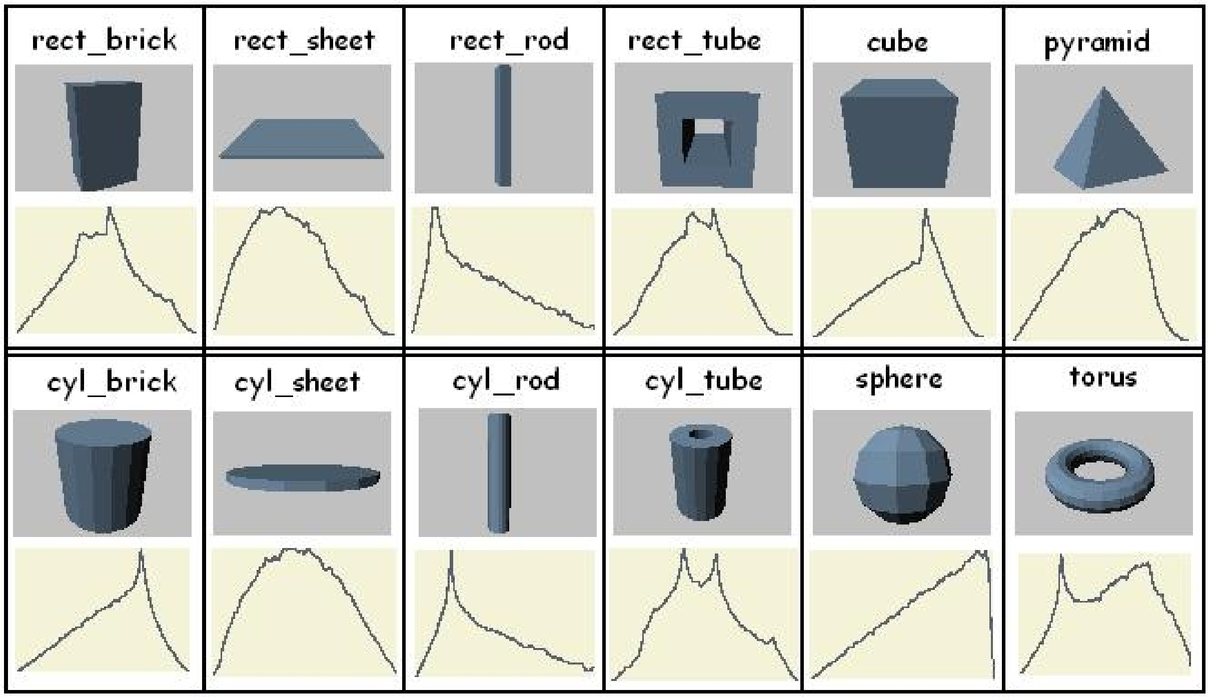


Figure 4.10: Geometrical primitives for Constructive Solid Geometry and their shape distributions.

scriptor of a model is a vector which elements are the names of constituent parts. For better description we also analyze the connectivity relations between the parts. We propose to compute pairwise angles between the main axes of each component found through the Principal Component Analysis. In this way we obtain $n \times (n-1)$ values of angles between model constituents, where n is the number of connected components.

To illustrate the proposed shape analysis we consider the example of a 3D model of a table. Figure 4.11 shows this process.

In the next section we propose a method to map the shape descriptor to the vector of semantic labels.

4.4.3 Mapping Shape Descriptor to Semantic Labels

In order to map geometrical and topological features of a model from a chosen domain to semantically meaningful labels we propose to create a vocabulary describing all models of the domain. Table 4.5 illustrates the

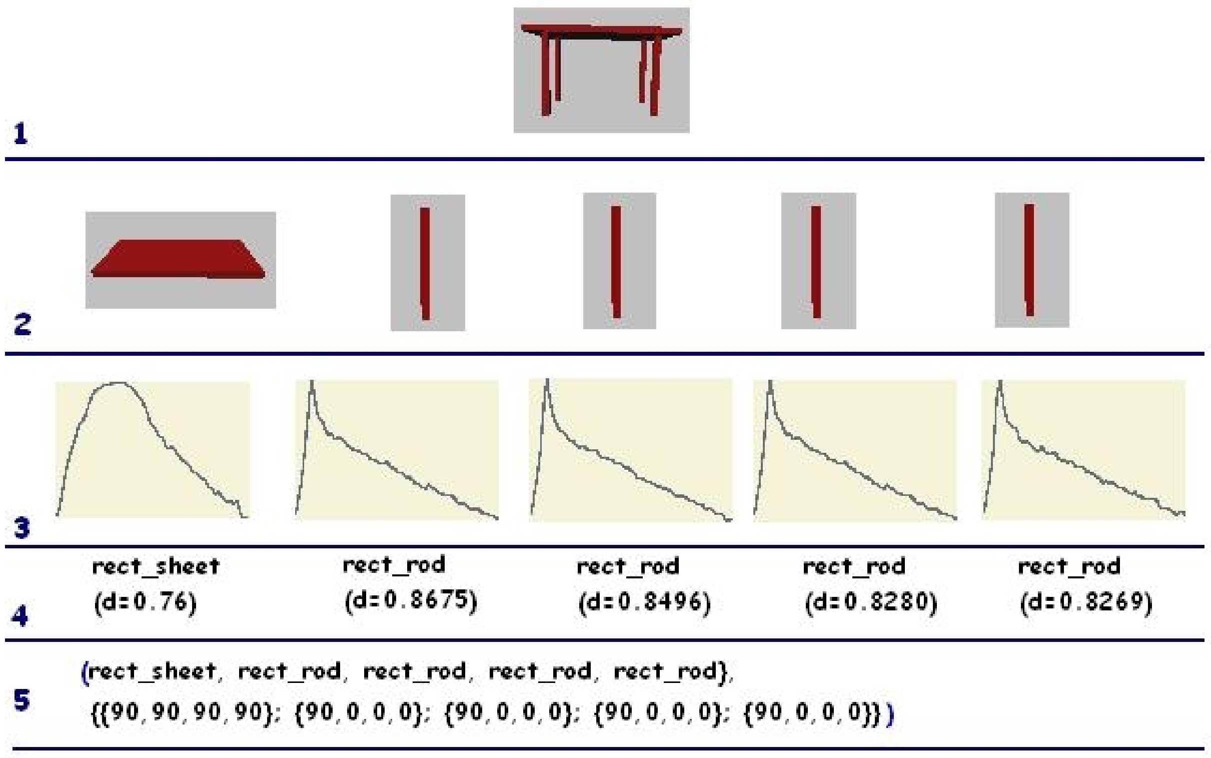


Figure 4.11: Feature vector extraction. 1) Input model. 2) Model decomposition. 3) Shape distributions of the model's constituents. 4) Labelling model's constituents as shape primitives. 5) Output feature vector.

description of models of the table on Figure 4.11.

The vocabulary should describe all concepts present in the ontology of the domain. Thus, querying it by the feature vector we can output as a result the vector of semantic labels. For instance taking the model of the table of the previous example, we get $\{top, leg\times 4\}$, and having the given semantic vector we can identify the category of the model in the ontology of the domain.

4.4.4 Ontology for Shape Annotation

Before building an ontology we should define its scoping, i.e. its domain, and its purpose, i.e. the intended usage [93]. In our case the domain that the ontology will formalize is that of furniture. In the first phase the intended usage of the furniture ontology is the annotation of models

Table 4.5: Mapping from low-level features to semantic labels.

Component	Geometrical primitive	Connectivity (pairwise angle between components	Cardinality	Semantic label
1	rect_sheet	{90,90,90,90}	1	top
2	rect_rod	{90,0,0,0}	4	leg

in the database with ontology concepts. In a second phase we want to investigate the possibility of retrieving the models by textual queries. We regard the 3D models as a syntactic domain and the ontology language as a semantic domain. An interpretation function will assign to each "3D model" a concept from ontology. In this way we can say that a certain 3D model is a "Chair", while another 3D model is a "Table", where "Chair" and "Table" are concepts in our ontology. Since the classes of models are distinguished at the syntactic level by the feature vectors extracted and explained in Section 4.4.2, there are two interesting questions that an ontology based shape annotation system should answer:

1. What is the system precision? The precision of the system in this case is defined as:
$$P = \frac{M_{Cann}}{M_{ann}} \times 100\% \tag{4.10}$$
where M_{Cann}- is the number of correctly annotated models and M_{ann} is the number of all annotated models. Since the system we propose is not implemented we cannot quantify its precision, but we can make an interesting observation. The upper boundary of what can be achieved is already known. If the properties that distinguish two ontology concepts cannot be mapped to distinct sets of syntactic features then the system will fail to correctly annotate the models. Let's suppose for example that there are two concepts named "YellowChair" and "BlueChair" in our ontology. Both concepts have as their superclass the concept "chair" and they are distinguished only by the color: respectively yellow and blue. Because the above mentioned algorithm cannot extract the color of an object the system will fail to correctly annotate "BlueChair" and "YellowChair" models. However, assuming

that for designers the shape of a model is a more important matter than its color, we suppose that the feature vector extracted on the previous step completely describes a model.

2. The second relevant parameter is the recall of the system that is defined as:

$$R = \frac{M_{ann}}{M_T} \times 100\% \tag{4.11}$$

where M_T is the total number of models in a dataset. If all models are well formed the recall will be 100%.

We can start building the furniture ontology using Wordnet Domains [57]. Developed at IRST, Wordnet Domains, is PWN (Princeton Wordnet) 1.6 [33] augmented with a set of Domain Labels. PWN 1.6 synsets have been semi-automatically linked with a set of 200 domain labels taken from Dewey Decimal classification, the world most widely used library classification system. The domain labels are hierarchically organized and each synset received one or more domain labels. We are interested in the synsets that are annotated with the domain "furniture". Because PWN is a linguistic resource and many concepts found there are not suitable for building an ontology of furniture we want to make use in our work of other ontologies and specialized thesauri.

We can encode our ontology in OWL language. In the beginning the ontology is a simple taxonomy enriched with a relation "hasPart" that specifies the parts of objects in the furniture domain. We make use also of cardinality restrictions as the following example, which describes the entry for the concepts "BackRestChair" and Back Rest Chair "BackRestChair-WithFourLegs", shows:

```
<owl:Class rdf:ID="BackRestChair">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:ObjectProperty rdf:
                         ID="hasPartLeg"/>
       </owl:onProperty>
```

```
        <owl:someValuesFrom rdf:
                          resource="#Leg"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:
                     resource="#BackRest"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:
                     ID="hasPartBackRest"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:
           ID="BackRestChairWithFourLegs">
    <rdfs:subClassOf rdf:
               resource="#BackRestChair"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:
                       about="#hasPartLeg"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#int"
        >4</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

The above OWL representation says that a ”BackRestChair” has as a part exactly one ”BackRest” and that a ”BackRestChairWithFourLegs” IS-A

"BackRestChair" and has exactly four legs. The only kind of inference needed in the example is the "inheritance" of properties from super-classes to their subclasses.

4.4.5 Retrieval through Annotations

After we annotated 3D models with ontology concepts users have two possibilities. First they can make retrieval of 3D objects by textual query. A query can be typed by the user or can be formed by ontology browsing. For example, a user interested in "barber chair" models can input the concept in a text box. Alternatively, he can browse the ontology and select the appropriate concept. The system will answer the user query by returning all the models annotated with the input concept or with a subconcept of the input concept. An enhanced retrieval system based on textual queries can take advantage of Boolean operators.

The second possibility is to query by an example model. Here a user can browse all models within the category of the input model and autonomously search for more similar models. Such approach groups all objects into quite large classes. The other way to search for similar models is to find the smallest dissimilarity measure (i.e. the smallest distance value) between feature vectors of the constituents of a sample model and corresponding parts of models from the same category. Such approach reduces the number of comparisons needed to retrieve similar models. As a result the overall dissimilarity measure will be the sum of dissimilarities between corresponding constituent parts.

4.4.6 Extension to Other Domains

In the current section we have presented the methodology for the synthesis of shape description and ontology-based annotation and retrieval. Performing shape analysis we decompose a 3D model into its constituents and we analyze the shape and connectivity between each of the parts of the model. As a result we output the feature vector describing the 3D model. Using a vocabulary defining all concepts of the ontology of the given domain (here furniture), we map the extracted feature vector to the vector of semantic labels. Finally, the ontology of the considered domain can be used in model annotation and then key word based retrieval of furniture models. More detailed ontology of the considered domain will result in more accurate retrieval process. The proposed method offers two options to a user: textual query and query by a sample model. As a result, the proposed method succeeded in term of shape-to-text (shape annotation) and text-to-shape (query shape by text) schemes.

The proposed method of shape annotation for furniture models can be easily extended to other domains, using possibly different techniques for shape segmentation and analysis. Recently the method for automatic segmentation and annotation of a virtual human was proposed in [30]. To annotate segmented shape of a scanned body, a special dictionary was defined which tie together geometrical characteristics of the shape of parts of human body and their semantics.

A shape should be interpreted within a specific domain to avoid possible ambiguities in its meaning. Consequently, the rules for shape annotation should be derived depending on the chosen domain of shapes. The vocabulary of the domain represents the annotation rules for the constituents of 3D models; the constituents, models and relations between them are further represented formally in the domain ontology.

4.5 Conclusions

In the current chapter we have presented all phases of the retrieval framework. The framework can be queried by an example 3D model, which is represented by manifold triangle mesh. The shape of the query is segmented into simple and complex regions. As the result of the segmentation the Extended Reeb Graph is constructed to represent the topological structure of the model. The shape of simple regions is further studied locally. Three shape classification schemas are defined according to taper, bending and curvature of the region. The indices of the shape classification are stored as attributes of the ERG.

To avoid NP problem of graph matching we exploit spectral properties of the ERG. In order to store as more information as possible we use the

Hermitian matrix for graph representation, where shape information is encoded both in the magnitude and the phase of the complex entries. By obeying several constraints on the construction of elements, the Hermitian matrix mimics spectral behavior of the Laplacian.

The problem of matching graphs of different size is solved by partitioning the graphs into smaller subgraphs. The partitioning is accomplished using topological information stored in the eigenvector of the Hermitian matrix corresponding to the second smallest eigenvalue. We call this vector as Fiedler vector of the Hermitian matrix. Consequently, each 3D model is represented by the complete ERG and smaller subgraphs. Each of these graphs is further represented by Fiedler vector. Two distances are calculated for a pair of 3D models, the distance between Fiedler vectors of complete ERG graphs and the average normalized distance between Fiedler vectors of the subgraphs. The ranks produced by these two distances are combined in the way to increase the influence of more similar models (small distance value) and damp impact of less similar models (bigger distance between Fiedler vectors).

Finally, we presented the methodology to bridge the gap between the geometrical and topological description of a shape and its semantics. To this end, the domain of the considered 3D models should be defined and the vocabulary describing the models of the domain should be constructed. The vocabulary developed within the ontology of the domain is used to map geometrical shape characteristics to semantic labels. Finally, the ontology formalizing both concepts of the domain and the relations between them can be used for model annotation and text-based retrieval. We presented the methodology for the domain of furniture models but it can be applied to other domains using appropriate shape segmentation and description techniques.

Chapter 5

Experimental Results

The retrieval process consists of many phases. The efficiency of each of the phases influences the final performance of the retrieval system but hardly can be evaluated separately. Some observations can be done about the correctness of the shape segmentation for some particular classes of models, as well as about the local shape analysis of extracted components. However, the efficiency of both these phases will be revealed while evaluating the performance of the whole framework. To test the framework we use the benchmark of 3D models prepared for Watertight Model Track of SHREC 2007, SHape REtrieval Contest organized in conjunction with Shape Modeling International Conference. The benchmark is the collection of 400 models divided in 20 classes, 20 models per each class. The models were collected from National Design Repository at Drexel University [10], the AIM@SHAPE repository [5], the Princeton Shape Benchmark [11], the CAESAR Data Samples [7], the McGill 3D Shape Benchmark [9], the 3D Meshes Research Database by INRIA GAMMA Group [1], the Image-based 3D Models Archive. The collection was divided into classes manually by the organizers of Watertight Models Track [94]. Figure 5.1 shows the classified benchmark.

The classes of the collection were defined using semantics of the models. Consequently, the shape of the models within the same class can vary greatly, including diverse position and posture of models and different substantial shape features.

5.1 Shape Segmentation

In this section we show the results of shape segmentation using iterative bisection method described in Section 4.1.3. The results are shown for 3D models from different classes and using different mapping functions. Precisely, we used height, distance from the barycenter and integral geodesic distance functions for the segmentation. The experiments demonstrate that the height function gives better results for CAD models. Here saying "better results" we mean the segmentation which gives large simple segments feasible for the further shape analysis. Such results can be explained by the fact that CAD models are usually correctly oriented during the design process. The Figure 5.2 shows the results of the segmentation of a mechanic model using height, distance from the barycenter and integral geodesic distance measuring functions. However, the segmentation based on the height mapping function is not invariant to the position of a model in space and, in general, requires a preliminary alignment.

On the contrary, for free-form models, like animals, toys and humans, better results are obtained when the measuring function is the distance from the barycenter or integral geodesic distance. Figure 5.3 illustrates segmentation of a free form model using height, barycenter and integral geodesic distance functions.

The function of the distance from the barycenter as well as geodesic distance function gives the advantage of being invariant to rotation. Moreover, the latter is also robust to different postures of a model but it requires more computational time. Figure 5.4 shows segmentation results using integral geodesic distance function for a model located differently in space and having diverse postures.

5.2 Shape Analysis

The results of shape analysis described in Section 4.2 are shown for some models on Figure 5.5.

The robustness of the shape analysis depends on the number of contours inserted inside each component and on the threshold used to decide weather the shape is increasing or decreasing. Inserting a few contours we risk to miss some shape features identifying, for example, the models of spring with multiple bending as components with single bending. On the other hand, if we increase the number of inserted contours and at the same time reduce the threshold for taper/enlargement decision we may reveal insignificant local shape features which influence the final shape classification.

Moreover, shape analysis depends on the measuring function used to segment a shape, because it defines the boundary of components. The function of the distance from the barycenter as well as integral geodesic distance function produces curved boundaries. The reference points defined by such boundaries can be very close to each other, leading to flat classification of the region, even when it is of high curvature. Such regions can be detected while separating complex saddle regions.Similar situation can occur for regions of maximal minimum values of the geodesic distance function(see Figure 5.6). In oder to avoid extraction of thin segments between two saddle regions the threshold used in shape segmentation for iterative bisection should be increased.

5.3 Efficiency of the 3D Model Retrieval Framework

The results of shape retrieval which will be presented further in this section are obtained by segmenting models of the benchmark using iterative bisection method and integral geodesic distance function with approximately 200 base points scattered over the surface of a model. The shape of simple components was analyzed according to the proposed in Section 4.2 schema using only four intermediate contours. Finally, the shape is represented by the attributed ERG, as shown in Section 4.3.1. To evaluate efficiency of the framework and to compare it with others methods proposed by participants of SHREC 2007, we computed the same performance measures. These measures are precision and recall after 20, 40, 60 and 80 retrieved models, average dynamic recall (ADR), percentage of success for the first (PS1) and the second (PS2) retrieved items, average ranking (AR), last place ranking (LPR), cumulated and discount cumulated gain vectors (CGV and DCGV). We refer to Section 2.3 for the definitions of these measures.

Table 5.1: Performance values for some selected classes of models.

	Human	Ant	Mechanic	Bearing	Bust
P_{20}	0,94	0,66	0,47	0,51	0,40
R_{20}	0,94	0,66	0,47	0,51	0,40
P_{40}	0,49	0,46	0,35	0,36	0,29
R_{40}	0,98	0,91	0,70	0,72	0,58
P_{40}	0,33	0,33	0,28	0,26	0,21
R_{40}	0,99	0,99	0,83	0,79	0,63
P_{80}	0,25	0,25	0,22	0,21	0,18
R_{80}	1	1	0,90	0,85	0,70
ADR	0,95	0,87	0,65	0,67	0,63
LR	0,99	0,92	0,23	0,20	0,21
AR	12	17	51	59	87
PS1	100%	100%	100%	100%	100%
PS2	90%	100%	90%	80%	80%

Each model from the benchmark is used as a query and the listed values are calculated for each query. The computed values are further averaged over the classes and finally over the whole benchmark.

Before we list the values of performance values computed for the whole benchmark, we would like to adduce here observations over some separate classes.

As can be seen from figures in Table 5.1 our framework performs better for free-form models with substantial shape features. The classes of humans, ants, armadillos, teddies etc. are the classes for which the framework gives the best retrieval results. This can be explained by the presence of elongated shape components which are revealed after shape segmentation, and for which the schema of the proposed shape analysis fits perfectly.

Let us now consider the class of bearings. The performance of this class is similar to the classes of springs and glasses, where the models have simple topology. After the segmentation phase, the models are usually represented by two simple components, corresponding to maximum and minimum regions of the mapping function. The resulting ERG graph is a graph with only two edges. Hence, using only the topological information we would not be able to discriminate so simple models. The indices of shape analysis stored as labels of the ERG significantly improves the performance.

Still, the small size of the graphs leads to lower performance values for these three classes.

The last observation we would like to make here is about the class of busts. Our framework shows the worst performance for this class. The first and obvious reason is that the shape of models within this class varies greatly. As can be seen from Figure 5.7 some models from this class are just heads, others have some clothes and hat features, some are without hair and others have sophisticated coiffure. The second reason of such a low performance is that our segmentation method reveals larger shape features (e.g. a hat, curls, pedestal) that are not essential for busts, and thus can be absent in other models of the class.

On average, however, our segmentation, shape analysis and shape matching techniques perform very well, surpassing almost all the participants of watertight SHREC 2007 contest. According to the computed performance measures, our shape retrieval techniques ranks behind only one participant, the method where the authors used Multiresolution Reeb graph [92]. The performance measures calculated over the whole benchmark together with the values of the other SHREC participants are shown in Tables 5.2, 5.3 and 5.4. Values for LPR and AVGR were not given exactly in [94], but plotted in charts. Hence these values in the Table are read from these charts.

Figure 5.8 illustrates the retrieval results for some queries. The results show that the majority of models within the same class are identified correctly regardless the position and posture of the models. Due to robust shape analysis, the models with slight shape deformations, e.g. the models from armadillo class, are also retrieved correctly.

Table 5.2: Precision after 20, 40, 60 and 80 retrieved items. CFV refers to the proposed approach, using complex Fiedler vectors.

method	P_{20}	P_{40}	P_{60}	P_{80}
Akgul et al.	0.63	0.37	0.26	0.21
Chaouchet al.	0.55	0.33	0.24	0.19
Napoleon et al.	0.60	0.37	0.26	0.21
Daras et al.	0.56	0.35	0.25	0.20
Tung et al.	0.71	0.41	0.29	0.23
CFV	0.60	0.40	0.29	0.23

Table 5.3: Recall after 20, 40, 60 and 80 retrieved items. CFV refers to the proposed approach, using complex Fiedler vectors.

method	R_{20}	R_{40}	R_{60}	R_{80}
Akgul et al.	0.63	0.73	0.79	0.82
Chaouch et al.	0.55	0.66	0.72	0.76
Napoleon et al.	0.60	0.73	0.79	0.82
Daras et al.	0.56	0.69	0.76	0.80
Tung et al.	0.71	0.83	0.87	0.90
CFV	0.60	0.80	0.88	0.93

Table 5.4: Further performance measures. ADR = Average Dynamic Recall, LPR = Last Place Ranking, AVGR = Average rank of relevant item, PS1 = percentage of success for the first retrieved item, PS2 = percentage of success for the second retrieved item.

method	ADR	LPR	AVGR	PS1	PS2
Akgul et al.	0.79	0.54	48	100%	94.0%
Chaouch et al.	0.72	0.51	68	100%	92.8%
Napoleon et al.	0.78	0.61	52	100%	95.9%
Daras et al.	0.75	0.55	54	100%	93.3%
Tung et al.	0.86	0.74	31	100%	97.7%
CFV	0.77	0.69	35	100%	92.3%

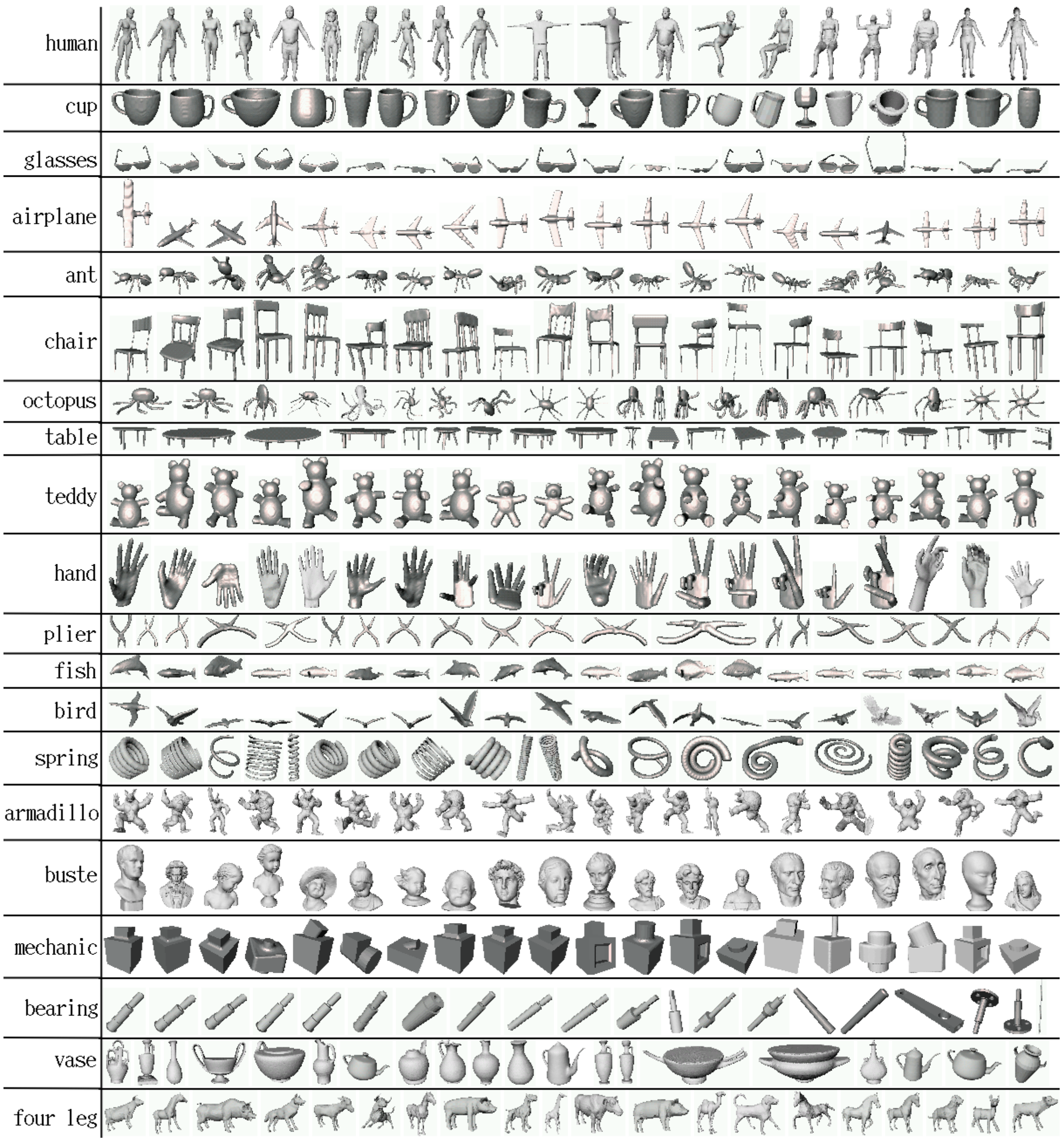


Figure 5.1: Watertight Shape Benchmark of SHREC 2007 [94].

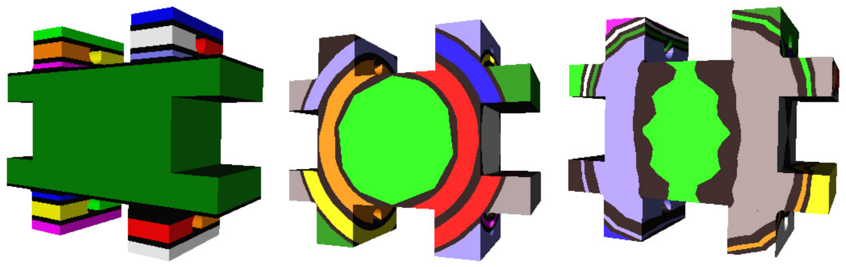


Figure 5.2: A mechanical model segmented using height, distance from the barycenter and geodesic distance measuring functions.

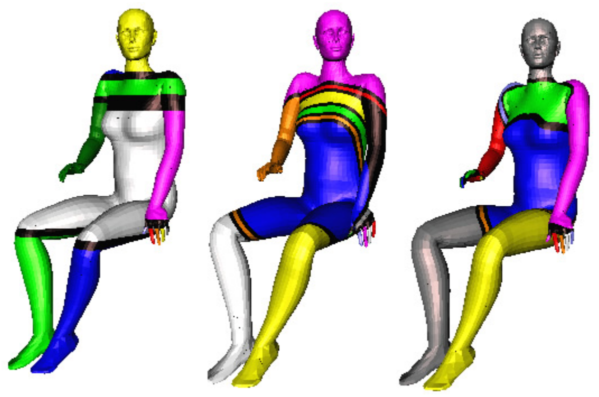


Figure 5.3: A model of a human is segmented using height, distance from the barycenter and geodesic distance measuring functions.

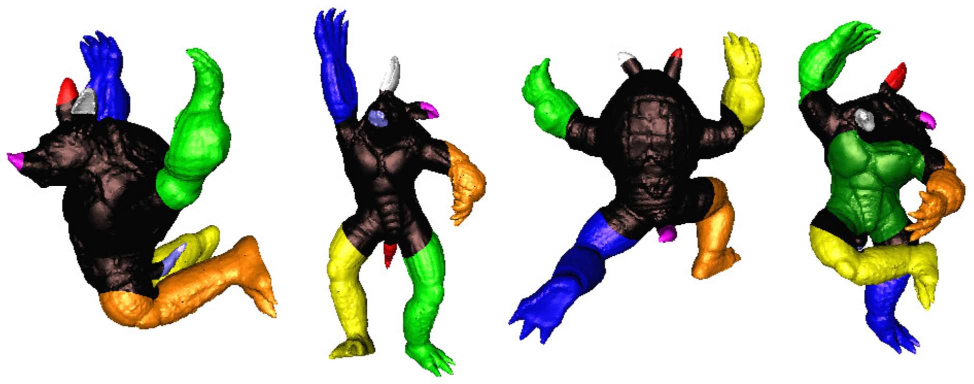
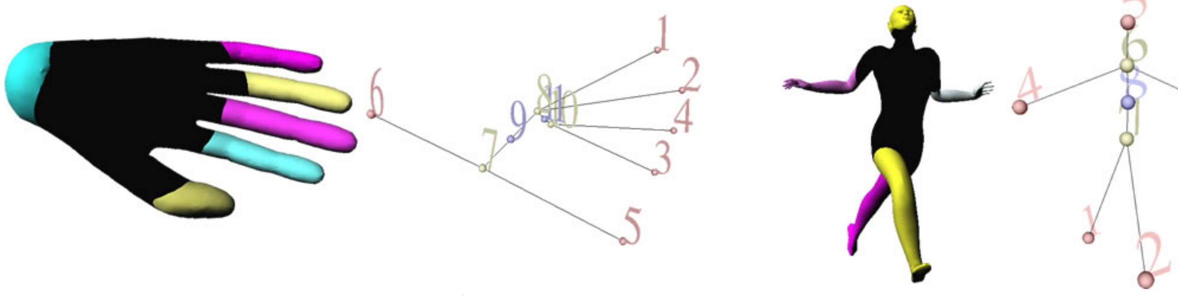


Figure 5.4: Armadillo's models located differently in space and having diverse postures are segmented using integral geodesic distance function.



Shape Analysis of a model of a hand

Node	1	2	3	4	5	6	7	8	9	10	11
SW	0.06	0.07	0.09	0.10	0.05	0.09	0.12	0.22	0.17	0.20	0.21
Cl1	1	1	1	1	1	4	Br.	Br.	Int.	Br.	Int.
Cl2	1	1	1	1	1	1	Br.	Br.	Int.	Br.	Int.
Cl3	3	3	3	3	3	3	Br.	Br.	Int.	Br.	Int.

Shape Analysis of a model of a human

Node	1	2	3	4	5	6	7	8
SW	0.17	0.16	0.06	0.06	0.06	0.13	0.36	0.24
Cl1	4	4	6	6	4	Br.	Br.	Int.
Cl2	2	2	2	2	1	Br.	Br.	Int.
Cl3	3	3	3	3	3	Br.	Br.	Int.

Figure 5.5: Shape analysis of segmented models. In the tables: SW = Segment Weight, Cl1 = classification according to component's taper/enlargement (1 - Constant, 4 - Smooth Decreasing, 6 - Multiple), Cl2 = classification according to component's bending (1 - No bending, 2 - Single Bending), Cl3 = classification according to component's curvature (3 - > 90°). The entry *Int.* stands for "Interconnecting node" and *Br.* for "Branching node"

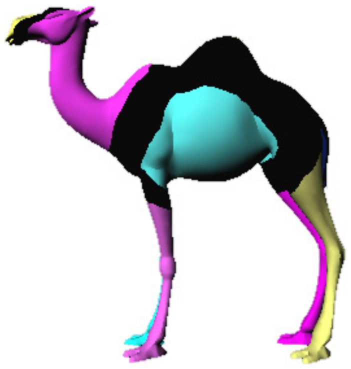


Figure 5.6: Segmentation of a model of a camel using geodesic distance function. Curved boundaries can lead to flat classification of curved regions, as for the green region of camel's paunch.

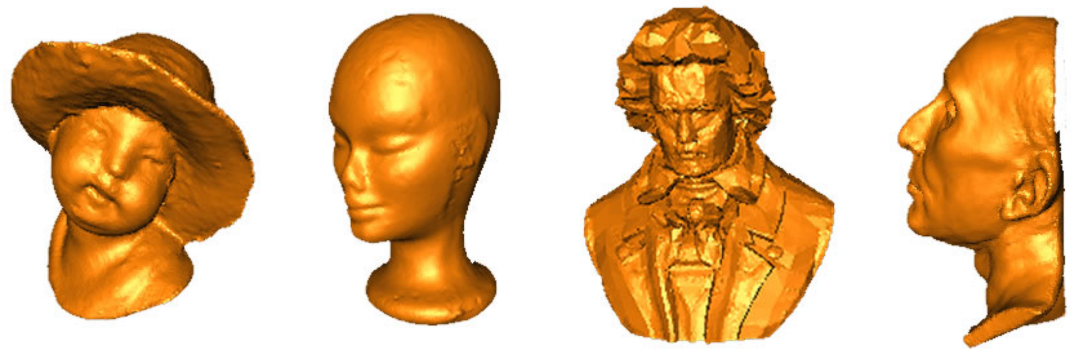
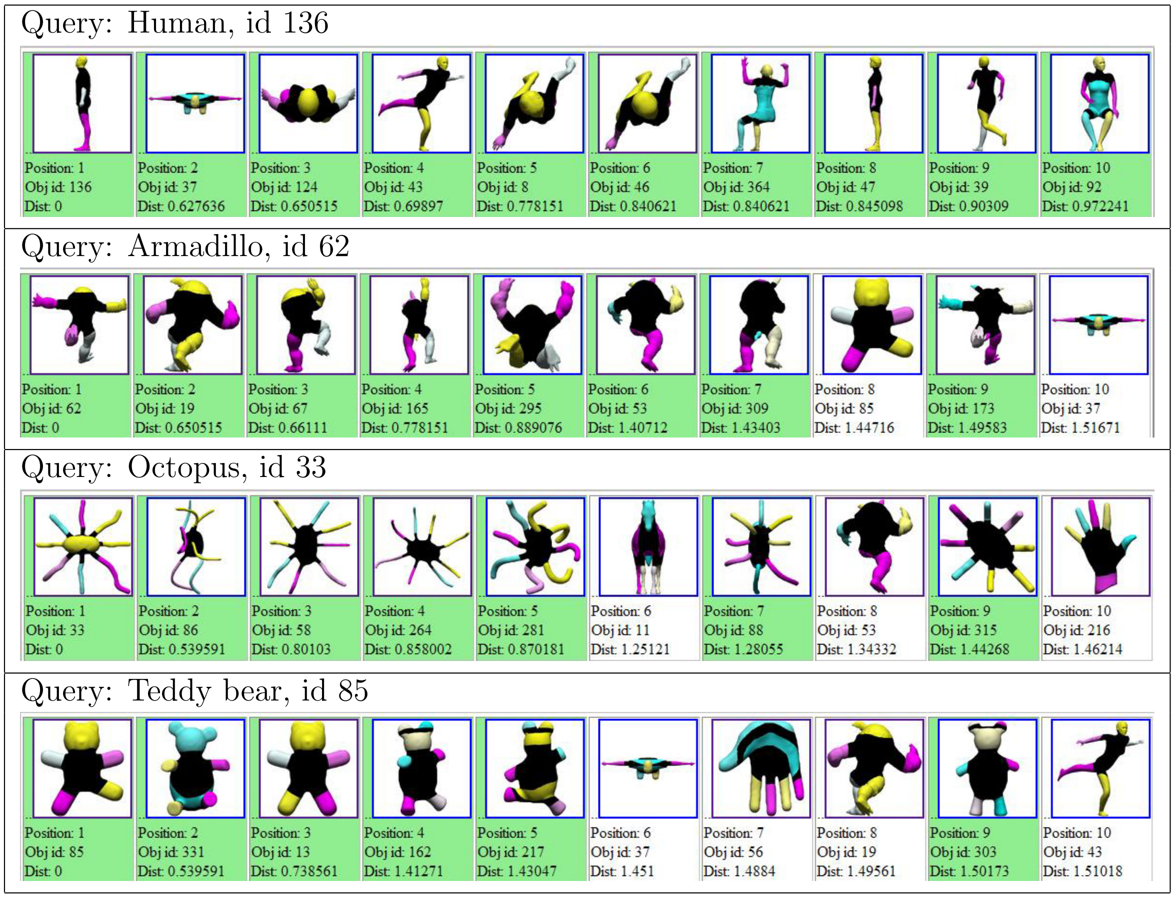


Figure 5.7: Variety of shape of the models within class of busts.



Query: Human, id 136

Position: 1	Position: 2	Position: 3	Position: 4	Position: 5	Position: 6	Position: 7	Position: 8	Position: 9	Position: 10
Obj id: 136	Obj id: 37	Obj id: 124	Obj id: 43	Obj id: 8	Obj id: 46	Obj id: 364	Obj id: 47	Obj id: 39	Obj id: 92
Dist: 0	Dist: 0.627636	Dist: 0.650515	Dist: 0.69897	Dist: 0.778151	Dist: 0.840621	Dist: 0.840621	Dist: 0.845098	Dist: 0.90309	Dist: 0.972241

Query: Armadillo, id 62

Position: 1	Position: 2	Position: 3	Position: 4	Position: 5	Position: 6	Position: 7	Position: 8	Position: 9	Position: 10
Obj id: 62	Obj id: 19	Obj id: 67	Obj id: 165	Obj id: 295	Obj id: 53	Obj id: 309	Obj id: 85	Obj id: 173	Obj id: 37
Dist: 0	Dist: 0.650515	Dist: 0.66111	Dist: 0.778151	Dist: 0.889076	Dist: 1.40712	Dist: 1.43403	Dist: 1.44716	Dist: 1.49583	Dist: 1.51671

Query: Octopus, id 33

Position: 1	Position: 2	Position: 3	Position: 4	Position: 5	Position: 6	Position: 7	Position: 8	Position: 9	Position: 10
Obj id: 33	Obj id: 86	Obj id: 58	Obj id: 264	Obj id: 281	Obj id: 11	Obj id: 88	Obj id: 53	Obj id: 315	Obj id: 216
Dist: 0	Dist: 0.539591	Dist: 0.80103	Dist: 0.858002	Dist: 0.870181	Dist: 1.25121	Dist: 1.28055	Dist: 1.34332	Dist: 1.44268	Dist: 1.46214

Query: Teddy bear, id 85

Position: 1	Position: 2	Position: 3	Position: 4	Position: 5	Position: 6	Position: 7	Position: 8	Position: 9	Position: 10
Obj id: 85	Obj id: 331	Obj id: 13	Obj id: 162	Obj id: 217	Obj id: 37	Obj id: 56	Obj id: 19	Obj id: 303	Obj id: 43
Dist: 0	Dist: 0.539591	Dist: 0.738561	Dist: 1.41271	Dist: 1.43047	Dist: 1.451	Dist: 1.4884	Dist: 1.49561	Dist: 1.50173	Dist: 1.51018

Figure 5.8: Example retrieval results: for four queries the top 10 retrieved items are displayed. Colored border means successful hit.

Chapter 6

Conclusion and Perspective

6.1 Conclusions

An increasing number of 3D models on the Internet and in corporate repositories provides a strong potentiality to reuse this information and to simplify design process. However, organizing, classifying, indexing and searching 3D models are challenging problems. Text-based description of 3D shapes creates many ambiguities and is not efficient for retrieval. A shape descriptor aims to encode topological and/or geometrical properties of a 3D model and represent it in shape retrieval process.

In this thesis we worked mainly on four issues related to 3D model retrieval.

First, we explored the topological structure of a 3D model, and we represented it using Extended Reeb graph. Precisely, we proposed a new iterative bisection algorithm for segmenting a shape. The results of the segmentation are shape components appropriate for the further shape analysis. Each of the components is represented by a node in the ERG, and nodes representing neighbor components are connected by an edge.

Second, we proposed a schema for local shape analysis of simple components of a model extracted on the previous segmentation step. The shape analysis is performed using three criteria, they are shape taper/enlargement, bending and curvature. These three criteria produces more than 76 shape types, which can be used for shape representation in retrieval process. To this end, we attribute the ERG representing topology of a model, with the indices of the shape classification.

Third, we proposed a new method for shape matching which are represented by the attributed ERGs. Precisely, we constructed a Hermitian matrix for the graph-based shape descriptor, and by imposing several constraints on the elements of the matrix we succeeded to mimic spectral behavior of the Laplacian matrix. Hence, we exploited the properties of the eigenvalues and eigenvectors of the Hermitian matrix for graph partitioning and matching. We calculated two similarity measures for each couple of graphs. The first measure is the similarity between two complete graphs, whereas the second measure is the similarity between two sets of their subgraphs. The two measures are mutually complementary, thus we combine them to obtain a unique rank by dampening the influence of less similar and increasing the impact of more similar models.

Finally, we proposed the methodology for bringing together geometry, topology and semantics of a shape. Similarly to ambiguity of words having different meaning depending on the context in which they are used, there exists ambiguity of a shape. By specifying the domain we can build the vocabulary of models of the domain. The vocabulary is used for mapping geometrical and topological shape characteristics to semantic labels. The ontology of the domain can be used for model annotation and text-based retrieval.

6.2 Future Work

The work fulfilled in this thesis suggests several directions for future work.

6.2.1 Shape segmentation and analysis

The segmentation and shape analysis techniques proposed in Sections 4.1 and 4.2 are dependent on the exploited mapping function, and moreover they provide an approximate representation of a shape. Even if the function of the integral geodesic distance appears to be quite robust to the position and posture of a model, the final segmentation may vary for strong shape deformations caused by different poses of the model. The shape segmentation and analysis are also dependent on the number of contours used for mesh decomposition. For simple models a few number of contours can be sufficient to produce correct shape description. Working with complex models we need to consider more level sets of the mapping function in order to reveal all shape details. On the other hand, higher number of level sets may lead to noisy shape descriptors as the result of detection of small insignificant shape perturbations. Hence, in future we would like to explore techniques for more precise shape description. We are interested in investigating the methods for construction a graph-based shape descriptor regardless the number of considered level sets. Another challenging question is combination of different measuring functions for shape description. Here the research on efficient shape description versus computational complexity should be conducted. Finally, we are interested in studying shape description techniques for the models which are represented by non-manifold meshes, as the number of such models on the Internet continues to increase.

6.2.2 Graph matching

In the thesis we have proposed a new technique for graph matching exploiting the spectral properties of the Laplacian matrix. By partitioning the graphs and further matching the sets of the subgraphs we succeeded to capture the similarity between graphs with slightly different structure. As can be seen from Figure 5.8 the retrieved models can have different segmentation, and hence they are represented by the ERGs with distinct structure and of different size. Our technique for graph matching succeeds to captures partial similarity between these graphs. In future we would like to investigate the ability of the proposed technique to detect partial similarity between the graphs with significantly different number of nodes, or with substantially diverse graph structure but having a few common components, e.g. the model of a hand and a model of a human.

6.2.3 Encapsulating shape properties into Domain Ontology.

In this work we have proposed the methodology for bringing together geometrical and topological properties of the shape and its semantics. The vocabulary of a chosen domain can be build with the goal to map the shape characteristics to semantic labels. An interesting research field is investigation of the way topological properties of a shape can be encode in the ontology of the domain. The only relation which is commonly used in ontologies and shape analysis is "hasPart". Other topological and geometrical relations between constituents of a 3D model, like overlap, underlap, tangent etc, can be formalized and encapsulated into the domain ontology using mereotopology theory. As the consequence the richer ontology will result in more precise shape annotation and retrieval. The issue of bridging the gap between shape and its intrinsic semantics has been addressed by European NoE project Aim@Shape, and continues to be a hot research topic involving experts from the fields of shape reasoning and understanding, knowledge processing and structuring.

Bibliography

[1] 3d meshes research database by inria gamma group. http://www-c.inria.fr/gamma/download/.

[2] 3d model retrieval system based on lightfield descriptors. http://3d.csie.ntu.edu.tw/.

[3] 3d sketch-based system for conceptual design. http://shapelab.ecn.purdue.edu/search.aspx.

[4] Aim@shape geometric search engine. http://gse.aimatshape.net/AimShapeApplication-war/.

[5] Aim@shape sape repository. http://shapes.aimatshape.net/.

[6] Aim@shape semantic search engine. http://dsw.aimatshape.net/sse/Search.jsp?ontology=shapes.

[7] The caesar data samples. http://www.hec.afrl.af.mil/HECP/Card1b.shtml.

[8] Free stuff on 3d cafe. http://www.3dcafe.com.

[9] Mcgill 3d shape benchmark. http://www.cim.mcgill.ca/shape/benchMark/.

[10] National design repository at drexel university. http://www.designrepository.org.

[11] Princeton 3d model search engine. http://shape.cs.princeton.edu/search.html.

[12] N. Amenta, S. Choi, and R.K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2):127–153, 2001.

[13] M. Attene, S. Biasotti, and M. Spagnuolo. Shape understanding by contour-driven retiling. *The Visual Computer*, 19(2-3):127138, 2003.

[14] J. August, K. Siddiqi, and S. W. Zucker. Distance-ordered homotopic thinning: a skeletonization algorithm for 3d digital images. *Computer Vision and Image Understanding*, 72(3):404–413, 1998.

[15] J. August, K. Siddiqi, and S. W. Zucker. Ligature instabilities in the perceptual organization of shape. *Computer Vision and Image Understanding*, 76(3):231–243, 1999.

[16] S. Baloch, H. Krim, D. Zenkov, and I. Kogan. 3d object representation with topo-geometric models. In *In 13 European Signal Processing Conference*, Antalya, Turkey, September 2005.

[17] S. Baloch, H. Krim, D. Zenkov, and I. Kogan. 3d object representation with topo-geometric models. In *13 European Signal Processing Conference*, Antalya, Turkey, September 2005.

[18] S. Berretti, A. Del Bimbo, and P. Pala. Partitioning of 3d meshes using reeb graphs. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, Hong Kong, China, August 2006.

[19] D. Bespalov, W. C. Regli, and A. Shokoufandeh. Reeb graph based shape retrieval for cad. In *Proc. of the 2003 ASME Design Engineering Technical Conferences*, Chicago, Illinios USA, September 2003.

[20] S. Biasotti. *Computational Topology Methods for Shape Modelling Applications*. PhD thesis, DIMA, Universit di Genova and CNR-IMATI, 2004.

[21] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Extended reeb graphs for surface understanding and description. In *Proceedings of the 9th Discrete Geometry for Computer Imagery Conference*, Uppsala, Sweden, December 2000.

[22] S. Biasotti and S. Marini. Sub-part correspondence using structure and geometry. In *4th Eurographics Italian Chapter*, Catania, Italy, February 2006.

[23] S. Biasotti, G. Patané, M. Spagnuolo, and B. Falcidieno. *Analysis and Comparison of Real Functions on Triangulated Surfaces.* Nashboro Press, 2007.

[24] H. Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[25] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranic. Automatic selection and combination of descriptors for effective 3d similarity search. In *ISMSE '04: Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04)*, Washington, DC, USA, December 2004.

[26] B. Chandrasekaran, J. R. Josephson, and R. V. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–26, 1999.

[27] D.Y. Chen, X.P. Tian, Y.T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. In *Proceedings of Eurographics 2003*, Granada, Spain, September 2003.

[28] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks, and R. Macleod. Coarse filters for shape matching. *Computer Graphics and Applications, IEEE*, 22:65–74, 2002.

[29] T. Culver, J. Keyser, and D. Manocha. Exact computation of the medial axis of a polyhedron. *Computer Aided Geometric Design*, 21(1):65–98, 2004.

[30] F. Dellas, L. Moccozet, N. Magnenat-Thalmann, M. Mortara, G. Patané, M. Spagnuolo, and B. Falcidieno. Knowledge-based extraction of control skeletons for animation. In *IEEE International Conference on Shape Modeling and Applications*, Lyon, France, June 2007.

[31] M. F. Demirci, R. H. van Leuken, and R. C. Veltkamp. Indexing through laplacian spectra. *Computer Vision and Image Understanding*, pages 21–26, 2007. Online.

[32] M. F. Demirci, R. H. van Leuken, and R. C. Veltkamp. Shape indexing through laplacian spectra. In *In Proc. of the 14th International Conference Image Analysis and Processing*, Modena, Italy, September 2007.

[33] C. Fellbaum (Ed.). Wordnet: An electronical lexical database. MIT Press, May 1998.

[34] H. Edelsbrunner, J. Harer, A. Mascarenhas, and V. Pascucci. Time-varying reeb graphs for continuous space-time data. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, Brooklyn, New York, USA, June 2004.

[35] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Local and global comparison of continuous functions. In *IEEE Visualization*, Austin, Texas, USA, October 2004.

[36] L. De Floriani, A. Hui, and L. Papaleo. Topology-based reasoning on non-manifold shapes. In *First international Workshop on Shapes and Semantics*, Matsushima, Japan, June 2006.

[37] J. Folley, A. vanDam, K. Feiner, and J. Hughes. *Computer Graphics Principles and Practice.* Addison-Wesley, 1997.

[38] A. T. Fomenko and T. L Kunii. *Topological Modeling for Visualizatio.* Springer-Verlag Tokyo, 1997.

[39] A.T. Fomenko. Special lecture: Computer visualization for the topology of integrable cases in rigid body dynamics. In *Computer Graphics International*, Hannover, Germany, June 1998.

[40] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3d models. *ACM Transactions on Graphics*, 22(1):83–105, 2003.

[41] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

[42] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[43] A. Hamza and H. Krim. Geodesic object representation and recognition. In *Discrete Geometry for Computer Imagery.*, Naples, Italy, November 2003.

[44] X. Han, C. Xu, U. Braga-Neto, and J.L. Prince. Graph-based topology correction for brain cortex segmentation. In *IPMI '01: Proceedings of the 17th International Conference on Information Processing in Medical Imaging*, London, UK, June 2001.

[45] J. C. Hart. *Mathematical Visualization*, chapter Morse Theory for Implicit Surface Modeling. Springer, Berlin, 1998.

[46] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *In SIGGRAPH 01: 28th Annual Conference on Computer graphics and interactive techniques*, New York, USA, August 2001.

[47] M. W. Hirsch. *Differential Topology.* Springer-Verlag, 1976.

[48] T. Hong, K. Lee, S. Kim, C. Chu, and H. Lee. Similarity comparison of mechanical parts. *Computer-Aided Design and Applications*, 2(6):759–769, 2005.

[49] C. Ip, D. Lapadat, L. Sieger, and W. Regli. Using shape distributions to compare solid models. In *Seventh ACM Symposium on Solid Modeling and Applications.*, Saarbrücken, Germany, June 2002.

[50] P. Kanonchayos, T. Nishita, S. Yoshihisa, and T. L. Kunii. Topological morphing using reeb graphs. In *CW '02: Proceedings of the First International Symposium on Cyber Worlds*, Tokyo, Japan, November 2002.

[51] R. A. Katz and S. M. Pizer1. Untangling the blum medial axis transform. *International JOURNAL of Computer Vision*, 55(2-3):139–153, 2003.

[52] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Symposium on Geometry Processing.*, Aachen, Germany, June 2003.

[53] S. Kirkland. A characterization of spectral integral variation in two places for laplacian matrices. *Linear and Multilinear Algebra*, 52(2):79–98, 2004.

[54] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[55] S.C. Lee, Y. Wang, and Lee E.T. Compactness measure of digital shapes. In *IEEE'04 Region 5 Conference: Annual Technical and Leadership Workshop*, Norman, Oklahoma, USA, April 2004.

[56] G. Leifman, R. Meir, and A. Tal. Semantic-oriented 3d shape retrieval using relevance feedback. *The Visual Computer*, 21(8-10):865–875, 2005.

[57] B. Magnini and G. Cavaglia. Integrating subject field codes into wordnet. In *2nd International Conference on Language Resources & Evaluation*, Athens, Greece, May-June 2000.

[58] N. Maillot, M. Thonnat, and A. Boucher. Towards ontology-based cognitive vision. *Machine Vision and Applications*, 16(1):33–40, 2004.

[59] N. Maillot, M. Thonnat, and C. Hudelot. Ontology based object learning and recognition: application to image retrieval. In *16th IEEE International Conference on Tools with Artificial Intelligence*, Sophia Antipolis, France, November 2004.

[60] M. Mäntylä. *Introduction to Solid Modeling.* Computer Science Press, 1988.

[61] S. Marini, M. Spagnuolo, and B. Falcidieno. From exact to approximate maximum common subgraph. In *Proceedings of GbRPR: Graph-Based Representations in Pattern Recognition*, Poitiers, France, April 2005.

[62] B. T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979–1998, 1999.

[63] V. Mezaris, I. Kompatsiaris, and M.G. Strintzis. An ontology approach to object-based image retrieval. In *International Conference on Image Processing*, Barcelona, Spain, September 2003.

[64] P. Min, M. Kazhdan, and T. Funkhouser. A comparison of text and shape matching for retrieval of online 3d models. In *European Conference on Digital Libraries*, Libraries, Bath, UK, September 2004.

[65] B. Mohar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics and Applications*, 2:871–898, 1991.

[66] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for the multi-scale analysis and decomposition of triangle meshes. *Algorithmica, Special Issues on Shape Algorithms*, 38(2):227248, 2004.

[67] F.S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. on Visualization and Computer Graphics*, 9(2):191–205, 2003.

[68] M. Novotni and R. Klein. 3d zernike descriptors for content based shape retrieval. In *The Eighth ACM Symposium on Solid Modeling and Applications*, Seattle, Washington, USA, June 2003.

[69] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3d models with shape distributions. In *Shape Modeling International*, Genova, Italy, May 2001.

[70] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *IASTED conference on Visualization, Imaging, and Image Processing*, Marbella, Spain, September 2004.

[71] V. Pascucci, G. Scorzelli, P.T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26:58, 2007.

[72] G. Patané, M. Spagnuolo, and B. Falcidieno. Para-graph: Graph-based parameterization of triangle meshes with arbitrary genus. *Computer Graphics Forum*, 23(4):783–797, 2004.

[73] A. Pinto, R. H. van Leuken, M. F. Demirci, F. Wiering, and R. C. Veltkamp. Indexing music collections through graph spectra. In *In Proc. of the 8th International Symposium on Music Information Retrieval*, Vienna, Austria, September 2007.

[74] J. Pu, Y. Kalyanaraman, S. Jayanti, K. Ramani, and Z. Pizlo. Navigation and discovery in 3d cad repositories. *IEEE Computer Graphics and Applications*, 27(4):38–47, 2007.

[75] H. Qiu and E. R. Hancock. Graph partition for matching. In *Graph-based Representations in Pattern Recognition*, Springer Berlin, Heidelberg, January 2003.

[76] G. Reeb. *Sur les points singulièrs dune forme de Pfaff complètement intégrable ou dune fonction numèrique.* Comptes Rendu de lAcademie des Sciences, 1946.

[77] J. Ricard, D. Coeurjolly, and A. Baskurt. Generalizations of angular radial transform for 2d and 3d shape retrieval. *Pattern Recogn. Lett.*, 26(14):2174–2186, 2005.

[78] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, 2004.

[79] D.W. Shattuck and R.M. Leahy. Automated graph based analysis and correction of cortical volume topology. *IEEE Trans. Med. Imaging*, 20(11):1167–1177, 2001.

[80] E. C. Sherbrooke, N. M. Patrikalakis, and E. Brisson. An algorithm for the medial axis transform of 3d polyhedral solids. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):44–61, 1996.

[81] Y. Shinagawa and T. L. Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Comput. Graph. Appl.*, 11(6):44–51, 1991.

[82] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Comput. Graph. Appl.*, 11:66–78, 1991.

[83] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S.W. Zucker. Indexing hierarchical structures using graph spectra. *Pattern Analysis and Machine Intelligence*, 27(7):1125– 1140, 2005.

[84] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International JOURNAL of Computer Vision*, 35(1):13–32, 1999.

[85] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine Vision and Applications*, 2007. Online.

[86] W. So and R. Merris. Rank one perturbation and its application to the laplacian spectrum of a graph. *Linear and Multilinear Algebra*, 46(3):193–198, 1999.

[87] D. A. Spielman and S.H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, Burlington, Vermont, October 1996.

[88] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14(3):181192, 1995.

[89] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66:24–49, 2004.

[90] J. Tierny, J.P. Vandeborre, and M. Daoudi. Reeb chart unfolding based 3D shape signatures. In *Eurographics*, Czech Technical University in Prague, September 2007.

[91] Y.A. Tijerino, M. Yoshida, S. Abe, and F. Kishino. A shape knowledge representation scheme and its application on amulti-modal interface for a virtual space teleconferencing system. In *4th IEEE In-*

ternational Workshop on Robot and Human Communication, Tokyo, Japan, July 1995.

[92] T. Tung and F. Schmitt. Augmented reeb graphs for content-based retrieval of 3d mesh models. In *SMI '04: Proceedings of the Shape Modeling International*, Genoa, Italy, June 2004.

[93] M. Uschold and M. King. Towards a methodology for building ontologies. In *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada, August 1995.

[94] R. C. Veltkamp and F. B. ter Haar. SHREC2007: 3D Shape Retrieval Contest. Technical Report UU-CS-2007-015, Utrecht University, 2007.

[95] D. V. Vranic and D. Saupe. 3d shape descriptor based on 3d fourier transform. In *EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services.*, Budapest, Hungary, September 2001.

[96] A. Wallace. *Differential Topology: First Steps*. W.A. Benjamin Inc., 1968.

[97] R. C. Wilson, E. R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *Pattern Analysis and Machine Intelligence*, 27(7):1112–1124, 2005.

[98] Z. Wood, H. Hoppe, M. Desbrun, and P. Schroder. Removing excess topology from isosurfaces. *Transactions on Graphics*, 23(2):190–208, 2004.

[99] Y. Xiao, P. Siebert, and N. Werghi. A discrete reeb graph approach for the segmentation of human body scans. In *Fourth International Conference on 3-D Digital Imaging and Modeling*, Banff, Canada, October 2003.

[100] T. Zaharia and F. Prêteux. Shape-based retrieval of 3d mesh models. In *Proceedings 2002 IEEE International Conference on Multimedia and Expo (ICME'2002)*, Evry, France, August 2002.

[101] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1), 2005.

[102] H. Zhang, O. van Kaick, and R. Dyer. Spectral methods for mesh processing and analysis. In *Eurographics*, Prague, Czech Republic, September 2007.

[103] P. Zhu and R. C. Wilson. A study of graph spectra for comparing graphs. In *British Machine Vision Conference*, Oxford Brookes University, Oxford, September 2005.

Appendix A

Publications

Van Leuken R. H., Symonova O., Veltkamp R. C. Topological and Directional Layout Indexing Using Hermitian Spectra. *SPPRA Signal Processing and Pattern Recognition Applications 2008.* Innsbruck, Austria, February, 2008.

Symonova O., De Amicis R. Shape Analysis for Augmented Topological Shape Descriptor. *EG07 Eurographics 2007.* Prague, Czech Republic, September, 2007.

Symonova O., De Amicis R. Shape Segmentation For Shape Description. *IADIS International Conference on Computer Graphics and Visualization.* Lisbon, Portugal, July, 2007.

Symonova O., Dao M.-S., Ucelli G., De Amicis R. Ontology Based Shape Annotation and Retrieval. *The ECAI'06 International Workshop on Contexts and Ontologies: Theory, Practice and Applications.* Riva del Garda, Italy, August, 2006.

Symonova O., Dao M.-S., De Amicis R., Ucelli G.. Topological Descriptor for CAD Model with Inner Cavities. *The 4th Eurographics Italian Chapter Conference.* Catania,Italy, February, 2006.

Hernádvölgyi I.T., Ucelli G., Witzel M., Symonova O., Delpero L., De Amicis R. Shape Semantics from Shape Context. *KI2004 Workshop on Modeling and Retrieval of Context of the 27th German Conference on Artificial Intelligence.* Ulm, Germany, September, 2004.