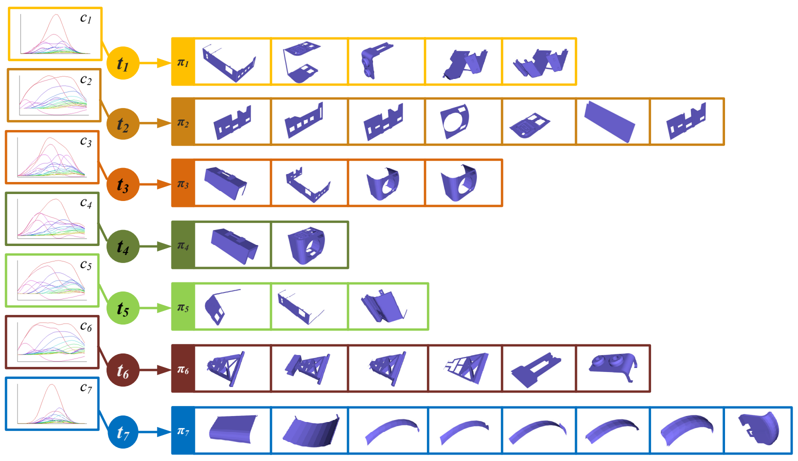
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



A Thesaurus-based Approach to 3D Shape Retrieval

Alfredo Manuel dos Santos Ferreira Júnior
(Master of Science)

Dissertation for the degree of Doctor of Philosophy in
Information Systems and Computer Engineering

July 2009

A Thesaurus-based Approach to 3D Shape Retrieval

Alfredo Manuel dos Santos Ferreira Júnior

A Thesis submitted to the Graduate School
for the degree of

Doctor of Philosophy in
Information Systems and Computer Engineering

Department of Information Systems and Computer Engineering
Instituto Superior Técnico - Technical University of Lisbon

Adviser

Doutor Manuel João Caneira Monteiro da Fonseca

Assistant Professor from the
Department of Information Systems and Computer Engineering
Instituto Superior Técnico - Technical University of Lisbon

Co-Adviser

Doutor Joaquim Armando Pires Jorge

Full Professor from the
Department of Information Systems and Computer Engineering
Instituto Superior Técnico - Technical University of Lisbon

This work was funded in part by the Portuguese Foundation for Science and Technology, grant **SFRH/BD/17705/2004** and the European Commission, through the FP6 Integrated Project **Euro-Tooling21** and the FP7 Specific Targeted Research Project **MAXIMUS**.

http://vimmi.inesc-id.pt/∼afj

alfredo.ferreira@inesc-id.pt

Resumo

À medida que a dimensão das colecções de modelos tri-dimensionais cresce, a recuperação destes modelos torna-se um verdadeiro desafio. Actualmente, uma colecção referente a um domínio específico pode conter milhares de modelos, enquanto que uma colecção genérica pode ser ainda maior. Este cenário cria a necessidade de motores de pesquisa baseados no conteúdo para modelos tri-dimensionais. Na realidade, a investigação em recuperação de modelos 3D baseada no conteúdo já existe e várias abordagens foram entretanto propostas. No entanto, o sucesso das soluções existentes fica bastante aquém do obtido pelas suas congéneres textuais.

A maior desvantagem dos actuais sistemas de recuperação tri-dimensional é o facto destes não suportarem pesquisas parciais. Presentemente uma interrogação submetida ao sistema deve corresponder a um modelo completo, não sendo possível interrogar o sistema usando apenas parte do modelo que se pretende, à semelhança do que acontece com os documentos textuais, onde a interrogação contém apenas algumas palavras e não todo o texto. Para resolver este problema, vários investigadores encontram-se a trabalhar em pesquisas parciais de objectos tri-dimensionais.

Neste âmbito, as abordagens propostas até agora focaram-se no reconhecimento de sub-partes semelhantes em objectos globalmente diferentes, no emparelhamento de sub-partes semelhantes em objectos estruturalmente idênticos, e na identificação de uma dada sub-parte num objecto. Não temos, no entanto, conhecimento de nenhuma solução que indexe modelos 3D com base em todas as suas sub-partes, permitindo dessa forma uma eficiente recuperação de modelos com recurso a interrogações parciais.

Na presente dissertação apresentamos uma nova abordagem à recuperação de modelos tri-dimensionais. Esta nova abordagem usa um mecanismo de decomposição de objectos sensível ao conteúdo da coleção, combinado com um *thesaurus* de formas e índices invertidos para descrever, indexar e recuperar modelos 3D. O algoritmo CAS realiza a decomposição de modelos com base nas semelhanças entre objectos na colecção, produzindo um conjunto de sub-partes dos modelos apropriado para a utilização com thesaurus de formas. Ao transpor conceitos bem sucedidos no contexto de documentos de texto para modelos 3D, resolvemos algumas das questões deixadas em aberto por abordagens anteriores. Na realidade, testes experimentais demostraram a validade da nossa abordagem. Assim sendo, acreditamos que a nossa solução providencia as bases para a construção de um motor de pesquisa 3D com suporte para interrogações parciais.

Palavras-Chave

Recuperação baseada no Conteúdo

Recuperação de Modelos 3D

Pesquisas Parciais

Segmentação de Objectos 3D

Thesaurus de Formas

Classificação e Indexação

Abstract

Index, search and retrieval of 3D models is becoming a key issue, as the size of available digital libraries of three-dimensional models grows. A regular domain-specific collection can contain thousands of items, and the number of generic 3D models available on generic collections is much larger. Such scenario leads to a demand for a content-based search engine for 3D models. Indeed, research in content-based 3D retrieval has already started, and several approaches have been proposed. However, the success of existing solutions is far from the success obtained by their textual counterparts.

A major drawback of most 3D retrieval solutions is their inability to support partial queries, that is, a query which does not need to be formulated by specifying a whole query shape, but just a part of it, for example a detail of its overall shape, just like documents are retrieved by specifying words and not whole texts. Recently, researchers have focused their investigation on 3D retrieval which is solved by partial shape matching.

The approaches developed so far address recognition of similar sub-parts in objects having different overall shape (partial matching) as well as recognition of similar sub-parts in objects that are both structurally globally similar (sub-part correspondence), and identification of a given sub-part in a model (part-in-whole matching). However, at the extent of our knowledge, there is still no 3D search engine that provides an indexing of the 3D models based on all the interesting subparts of the models, allowing efficient retrieval with partial queries.

In this dissertation we present a novel approach to 3D shape retrieval that uses the innovative collection-aware shape decomposition (CAS) combined with a shape thesaurus and inverted indexes to describe, index, and retrieve 3D models using part-in-whole matching. The CAS is an approach for automatic decomposition of models, which takes into account geometrical similarities among objects in the collection to produce a set of model sub-parts suitable for thesaurus-based indexing. Then, by transposing successful concepts from textual retrieval to 3D shape retrieval we overcome some issues faced by previous approaches. Indeed, experimental evaluation showed the validity of our solution in a controlled environment. Thus, we believe that our approach will provide the foundations for a fully-fledged content-based search engine for collections of 3D models with support for partial queries.

Keywords

Content-Based Retrieval

3D Model Retrieval

Partial Matching

Shape Decomposition

Shape Thesaurus

Classification and Indexing

Acknowledgements

First, I would like to express my appreciation to my adviser Professor Manuel J. Fonseca and co-adviser Professor Joaquim A. Jorge for their continuous support during the last years. To both I thank their availability to discuss my ideas, providing me constant comments and good guidance. At the end of this journey, I am glad to see that the initial advisory relations turned into friendships.

In second place, I would like to thank all the members of CNR IMATI-Ge, whom welcomed me so warmly in Genoa. Part of the work presented in this thesis was developed with their support. Indeed, some ideas and concepts come out from discussions with them. Despite my fellowship have fallen in a period of high work load, Bianca Falcidieno, Michela Spagnuolo, Simone Marini, Francesco Robbiano, Silvia Biasotti, Marco Attene, Daniela Giorgi, Michela Mortara and Chiara Catalano were wonderful and made my stay in Italy unforgettable, both from a professional and personal point of view.

Next, I would like to show gratitude to all the members of VIMMI group for their friendship and help, specially to Ricardo Jota for his companionship and his assistance during experiments, Bruno Araújo for his deep knowledge on Computer Graphics and his availability to share it, Tiago Guerreiro for being a superb office mate and Paula Monteiro for her assistance on bureaucratic stuff. Also, a special thank to Manuela Sado for her support during thesis writing.

Finally, I dedicate this thesis to my wife Sandra, who supported me unconditionally since when I was an undergraduate student (without her I had never get this far), to my son Tiago, with whom I do not share all the time he deserved, and to my aunt Bernardete for her love and support during all my life.

Lisboa, July 2009

Alfredo Ferreira

Contents

List of Figures

List of Tables

List of Publications

The work we developed within this PhD research has been the subject of original peer-reviewed publications. Although some work focused on topics marginal to the thesis core, the respective research contributed with valuable knowledge and experience. The relevant publications are listed below in reverse chronological order.

1. **Alfredo Ferreira**, Simone Marini, Marco Attene, Manuel J. Fonseca, Michela Spagnuolo, Joaquim A. Jorge and Bianca Falcidieno
   Thesaurus-based 3D Object Retrieval with Part-in-Whole Matching. International Journal of Computer Vision, Springer, Jun 2009.

2. Tiago Santos, **Alfredo Ferreira**, Filipe Dias and Manuel J. Fonseca
   Using Sketches and Retrieval to Create LEGO Models. Proceedings of the Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM'08), pages 89–96, Annecy, France, Jun 2008.

3. Manuel J. Fonseca, Elsa Henriques, **Alfredo Ferreira**, Joaquim A. Jorge and Rui Soares
   Assisting Mould Quotation through Retrieval of Similar Data. Chapter in "Digital Enterprise Technology - Perspectives and Future Challenges", Springer, June 2007

4. Manuel J. Fonseca, Elsa Henriques, **Alfredo Ferreira** and Joaquim A. Jorge
   Assisting Mould Quotation through Retrieval of Similar Data. Proceedings of the 3rd. International CIRP Sponsored Conference on Digital Enterprise Technology (DET'06) Setúbal, Portugal, September 2006

5. Manuel J. Fonseca, **Alfredo Ferreira** and Joaquim A. Jorge
   Generic Shape Classification for Retrieval. Springer Lecture Notes in Computer Science (LNCS 3926), Wenyin Liu and Josep LLadós (eds.), 2006.

Glossary

3DHT

Descriptor for 3D shapes based on the generalized 2D Hough transform.

3D SSD

Shape descriptor for 3D models based on the histogram of the shape index values.

CAH

Simple 3D shape descriptor that uses the length of vectors from the barycenter of the shape with its surface and corresponding angles with the coordinate axes.

CAS

Decomposition technique devised during this dissertation based on multilevel shape segmentation where objects are decomposed according to others in the collection.

CSG

Modelling technique that allows the creation of complex 3D models by using boolean operators to combine objects.

EGI

Histogram-based technique to represent the shapes of surfaces that define a 3D object.

ESB

Benchmark collection of 3D models for evaluating shape-based search methods relevant to the mechanical engineering domain.

HFC

Method to compute and represent the hierarchy of regions in a polygonal surface.

HFP

Automatic mesh segmentation algorithm that produces a binary tree of mesh segments using fitting primitives.

HSM

Binary tree structure representing an hierarchically segmented mesh.

PSB

Repository of 3D models and software tools for evaluating shape-based retrieval and analysis algorithms.

SHA

Numerical representation of 3D shapes commonly used as rotation invariant shape descriptor.

SHREC

A retrieval contest for 3D models widely accepted as a reference benchmark.

VEGI

Extension of the EGI technique that captures the volume distribution of an object without canonical alignment.

WTM

Collection of four hundred watertight mesh models collected from several web repositories.

1 Introduction

In the last decades the volume of multimedia information, such as images and video, stored into databases and over the internet has been growing. In particular, recent advances on modeling, digitizing and visualizing techniques led to a clear tendency to increase the number of 3D models both on the internet and in domain-specific databases. Three-dimensional models are used in a wide range of fields, such as engineering, virtual reality, medicine, molecular biology, geography or not to mention the growing entertainment industry. As a result, today we have a lot of 3D model collections available for usage on a wide range of areas.

The growing number of three-dimensional objects stored in digital libraries makes searching and browsing these collections a non-trivial task, since a regular domain-specific database can contain thousands of items. Indeed, unless effective meta-data have been assigned to models, it is not easy to find the sought model. Aware of this, during the last decade, researchers proposed several approaches based on shape similarity to retrieve 3D models. Some of these content-based retrieval systems are able to find a model in a database using query-by-example, a set of keywords, or a sketched query. However, results produced by such systems are far from the successful query results obtained by their textual counterparts.

The following section describes one of the problems that, in our opinion, contribute for the lack of success of 3D model retrieval systems and which we tackled during our research. Next, we suggest a solution for this problem, enumerating the key objectives and presenting a brief overview of our work. Finally, we summarize the main contributions and list the original publications presented to peer-reviewed journals and scientific conferences.

1.1 Problem Description

Existing approaches to content-based 3D shape retrieval work mostly by comparing the complete models [37]. Even those who use local features to represent a model in a collection usually do not allow matching of object subparts. Indeed, a major handicap of most existing retrieval systems is the fact that they only support queries of the complete object and do not allow partial queries to be formulated, which greatly hinders their usefulness. We can illustrate this problem with a parallel between 3D and text retrieval.

Most 3D search engines work mostly by simply matching entire objects: the query is a complete 3D object and the items against which it is matched are also complete objects. In a text-based system, this would mean to require that detailed specifications of pages, or even complete documents, are used as query initiators instead of typing a few words to a search engine to find the results sought. This might explain why 3D model retrieval systems enjoy limited usefulness and there is no equivalent of a Google^{TM} search engine for three-dimensional geometric shapes.

Recently, a few 3D shape retrieval approaches with partial matching capabilities have been proposed. These new approaches allow searching for a model by supplying as a query only a part of the desired model. However, regarding models in the collection, such solutions rely on representing only a few selected sub-parts of each of these models [59, 114] and not the complete models as a combination of all its sub-parts. Indeed, considering a small set of distinctive features of an object to classify it proved to be an efficient short-cut, but some eventually relevant object information is discarded in this process. Using again a parallelism with text documents, this kind of approach is similar to using only a few keywords to classify each document, which is clearly insufficient when compared to full text search.

On the other hand, the partial matching approaches that does not discard shape features [31, 125, 123] face the problem of time complexity, since they rely on performing comparisons with all shapes. When considering large collections of 3D models, such approach is not practical. In these cases, query time depends on the collection size. For instance, considering that a fast partial matching between two objects take around one tenth of a second, if we query a collection with several thousand elements using this technique it will need too much time to execute.

The main goal of our research is to devise a solution that overcomes the problems referred above: (1) for large collection of 3D models, retrieval is time-efficient only with complete object matching or (2) when only a smal set of features are considered in each model. Such solution should be able to perform partial queries whose execution time does not depend on collection size and where the complete models are included on the search process, instead of only a few selected sub-parts.

1.2 Thesis Statement

To attain the goal identified above, regarding efficient retrieval of three-dimensional shapes using partial queries, we propose a novel approach to retrieval with part-in-whole matching [1]. We aimed for a solution similar to the one adopted by existing text retrieval systems, which classifies all words from the entire document and not only a small set of selected keywords. In these systems the submitted queries usually contain just a couple of terms [118] and documents containing such words are retrieved. Likewise, our approach propose the retrieval of models from a collection based on geometrical similarity between a query and parts of the models in that collection.

Devising such retrieval solution faces two major challenges, besides those shared with the global matching approaches, such as query formulation and shape feature extraction. The first challenge is to devise an effective and efficient decomposition of models into sub-parts. The second is to find an effective way to index the extracted information to allow a fast and accurate search. In the present research work we dedicated a special attention to these challenges.

1.2.1 The ultimate goal

Ignoring for a while the practical problems, we present an overall description of what we think a fully fledged content-based 3D model retrieval system should be. In our opinion, an ideal system should be able to decompose models into its components, eventually in multiple scales, and classify all of them correctly. After an effective object decompo-

[1]In the literature, two distinct approaches to object retrieval with partial queries can be found. The partial matching aims on finding objects in a collection that share similar sub-parts with the query. The part-in-whole matching aims on retrieving models that contain the query.

sition and corresponding sub-part classification, it should be possible to submit a query to the system, representing a subpart of existing objects in the database, and it will return a list of models containing similar parts. Despite no such system had been devised yet,achieving such solution is the ultimate goal of research on 3D retrieval.

During our research we intended to develop novel techniques that provide some of the functionality necessary to build a retrieval system with partial queries. Assuming that models in the database are not previously decomposed by human operators, devising a system that behaves as described above is a complex task. Although our goal was not to devise a final solution for this problem, we believe to have provided some major contributions to it. Namely through a novel technique for automatic model decomposition and a methodology to clssify, index and retrieve 3D models.

1.2.2 Problem statement

Concerning a 3D search engine, performing retrieval with partial matching on large collections of three dimensional models using all of its features remains an open challenge. Although some techniques seems practical for indexing large models, and even large collections of complex models, these only consider a small set of relevant local features of each object. Such approaches do not fulfill the main goal of the research on shape retrieval with partial matching: the ability to find models with different global shape properties but having just some characteristics in common with the query, which might not even be the most relevant geometric features. In this dissertation research work we aimed at achieving a slightly different goal: 3D model retrieval with part-in-whole matching, *i.e.* find in a collection models containing sub-parts similar to teh query shape.

To that end, we followed a different approach from those proposed by other researchers and described ahead in Section 2.5. Like them, we aim to provide a solution that will allow successful searches on 3D databases with partial queries. However, we considered not only relevant parts of models, as some of them do, but the whole set of parts that compose objects. Moreover, we undertake the challenge of supporting large collections, and not only small sets with hundreds of models. Thus, we had to overcome a major problem: the time and space complexity associated to indexing all components of each model, even the irrelevant ones. This is even worst when considering a multi-scale approach to shape decomposition [97]. Nevertheless, indexing and searching contents of large collections have been already addressed with success in text information retrieval. In this dissertation, we suggested adapting techniques from this field to 3D shape retrieval.

Words are the basic elements of textual information retrieval. Every document in a collection is composed by words and these are used to classify and retrieve it. Thus, to transpose the textual information retrieval concepts to 3D shape retrieval, we should establish a parallel between words in document and parts from three-dimensional models. Since a model can be defined by merging its sub-parts, we considered that the basic element of our retrieval approach is a three-dimensional shape. However, text documents are based on a finite vocabulary of words, while models are defined in a continuous 3D space, thus not generally defined by a finite set of shapes. To overcome this divergence, we cluster model sub-parts according to their similarity and consider the resulting finite partition a vocabulary of basic shapes. Thus, to classify 3D models we use their sub-parts as words are used in textual information retrieval.

1.2.3 Research hypothesis

Well known indexing and matching techniques, widely-used in text information retrieval, produce successful practical results, such as the Google^{TM} search engine. In these systems, all words from the documents in the entire collection are classified and not only a small set of selected keywords. The queries usually contain just a couple of terms [118] and documents containing such words are retrieved. Likewise, our approach will retrieve models from a collection based on geometrical similarity between a query shape and parts of models in that collection.

We believe that through the use of an effective shape decomposition mechanism and a shape thesaurus with inverted indices we will be able to describe and retrieve 3D models using part-in-whole queries, similarly to what happens with word thesauri. This approach will allow us to take advantage of some well known techniques from text information retrieval, such as **tf-idf** [2] to rank the relevance of every subpart in the database or data compression techniques to reduce indices size.

[2]The term frequency and inverse document frequency (**tf-idf**) is a quantitative measure commonly used in information retrieval and text mining to determine the relevance of a word to a document in a collection, regarding to the contents of the whole collection.

Having identified the major challenges and how we intended to address them, we can now present the research hypothesis we formulated:

> **Retrieval of 3D objects with partial queries can be achieved by decomposing a model into a set of sub-shapes that describe the whole model, combined with a shape thesaurus for indexing.**

This hypothesis summarizes our main ideas and clearly indicates our final objective: to contribute to the development of a 3D shape retrieval solution that supports part-in-whole matching. But such goal could be decomposed into a set of research objectives in order to clearly identify the focus of our work. According to the hypothesis and taking into account the problems we faced, our research had three major goals:

1. Devise a solution to decompose a three-dimensional model according to the context where it lies (*i.e.* other objects in the collection), producing a set of segments (3D ”words”) suitable for the construction of a shape thesaurus;

2. Identify methods for shape matching and indexing that supports a thesaurus-based approach on 3D shape retrieval;

3. Develop a thesaurus-based framework for 3D shape retrieval with partial queries combining the methodologies referred above.

To achieve these objectives we devised a novel approach to 3D shape retrieval, described briefly in the next section and with more detail ahead in this document.

1.3 Approach Overview

In the dissertation work described in this thesis we researched the viability of transposing the matching and indexing approaches widely-used in text information retrieval to the 3D shapes field. Thus, we will start by briefly describe the concepts and basic ideas behind the text information retrieval approach used by successful search engines. After describing these concepts, we introduce our approach to shape retrieval.

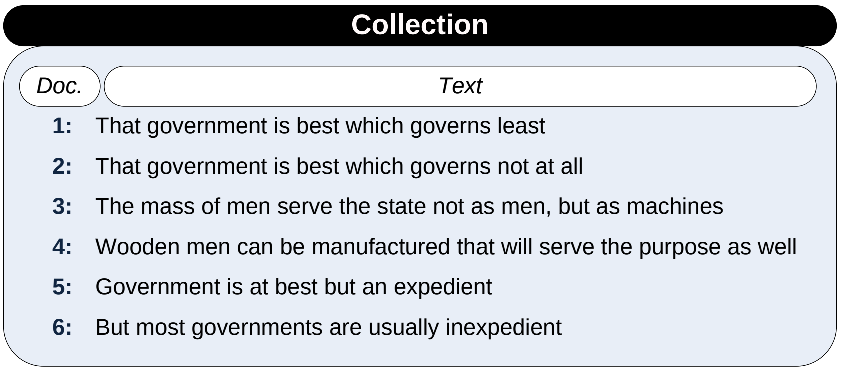
We will illustrate the description of text information retrieval concepts with a trivial example. For that end we use a simple collection presented in Figure 1.1, which contains only six text documents. Note that, despite its simplicity, this collection allows a clear representation of the basis of indexing mechanisms used by most text retrieval engines, which is our objective.

1.3.1 Indexing and Searching Text Collections

In theory, it is possible to query the contents of a textual documents collection by performing a sequential search. Sequential searches, often called serial scanning, consists in directly scan the document contents by finding the occurrences of a given text pattern - the query - in the documents. This strategy can be used when the collection is small (*i.e.* a few megabytes) and is the best choice when the collection is volatile (*i.e.* is frequently modified) [17].

However, if the collection grows, sequential searches are no longer feasible. In this case, data structures should be built over the text to allow efficient searches. Inverted indexes are widely used for this purpose in semi-static collections, providing good retrieval results with the cost of overhead space. These semi-static collections can be huge and updated, for instance, on a daily basis or at least indexed with such frequency even if thousands of documents are changed, inserted or removed every second. Indeed, web search engines adopted this approach and their success proves its goodness.

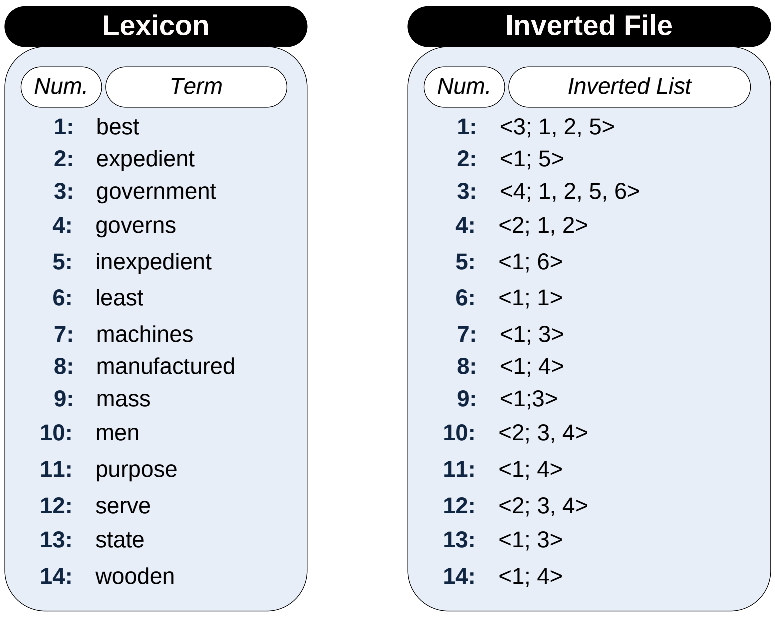
Conceptually, an inverted index (or inverted file) is a data structure composed by two



Collection

Doc.	*Text*
1:	That government is best which governs least
2:	That government is best which governs not at all
3:	The mass of men serve the state not as men, but as machines
4:	Wooden men can be manufactured that will serve the purpose as well
5:	Government is at best but an expedient
6:	But most governments are usually inexpedient

Figure 1.1: Example of a textual document collection.



Lexicon

Num.	Term
1:	best
2:	expedient
3:	government
4:	governs
5:	inexpedient
6:	least
7:	machines
8:	manufactured
9:	mass
10:	men
11:	purpose
12:	serve
13:	state
14:	wooden

Inverted File

Num.	Inverted List
1:	<3; 1, 2, 5>
2:	<1; 5>
3:	<4; 1, 2, 5, 6>
4:	<2; 1, 2>
5:	<1; 6>
6:	<1; 1>
7:	<1; 3>
8:	<1; 4>
9:	<1;3>
10:	<2; 3, 4>
11:	<1; 4>
12:	<2; 3, 4>
13:	<1; 3>
14:	<1; 4>

Figure 1.2: Example of a lexicon and corresponding inverted file for a textual document collection.

elements: the lexicon (or thesaurus) and the occurrences list, often called inverted file. To avoid misinterpretations, we will refer to these elements as thesaurus and inverted file for the remaining of this document. The thesaurus contains the vocabulary used in the collection, *i.e.* the words used in the documents. Respecting the concept of a lexicon, different forms of the same word clustered together, such as *run*, *ran* and *running*, which are different forms of the same lexeme - *run*. Thus, the construction of a word thesaurus is more than just listing all words in the documents.

For each entry in the thesaurus, called term, the occurrences of any form of that word in the collection are stored in the inverted file. To that end, an inverted list containing the number of occurrences and the documents where they occur is built for every term in the thesaurus. The inverted file is no more than the set of all these lists, as illustrated in Figure 1.2. In this example we show a sample lexicon, extracted from the collection presented in Figure 1.1 and the corresponding inverted file, where is represented a list of occurrences for each term in the lexicon.

Eventually, additional information can be included on this structure to improve its efficiency, such as referring the position in the document where each word occurs. The space complexity issue must also be addressed when indexing large collections of documents.

Indeed, several techniques are used to reduce space requirements of inverted indices, but since we only intend to explain the basic concepts, we will not take into consideration all the capabilities and problems of such structure. Instead, for more complete details and discussion on these topics we refer our readers to specialised literature [92, 17].

After indexing the collection,a three-stages algorithm is used to retrieve documents. It starts by performing a search in the lexicon, in which it identifies the terms that match the given query. Then, the corresponding list of occurrences is retrieved from the inverted file, identifying the documents where the query appears. In the last stage these occurrences are processed, usually using additional information stored in the inverted file, in order to improve the query results.

1.3.2 Thesaurus-based 3D Shape Retrieval

Our research aims at transposing to 3D retrieval the matching and indexing concepts widely-used in text information retrieval. We propose to use a shape thesaurus for model classification, indexing and retrieval. Similarly to a word thesaurus (or lexicon) in text retrieval, which contains the list of words that compose the vocabulary used in the documents, our shape thesaurus will contain shapes that compose the indexed models. Conceptually, the shape thesaurus should consist of a list of segments extracted from models in the collection and the inverted index will consist of a list of terms in the thesaurus with the corresponding lists of models containing that segment.

However, while words are explicit in text, subpart identification in a three-dimensional object is not trivial. Moreover, the success of our approach depends greatly on the quality of this identification. Unfortunately, no automatic detection of Geons - primitive shapes that, according to neuroscientist Irving Biederman, are used to construct structural representations of objects in human brain [32] - have yet been devised. Indeed, from a practical point of view, sub-part identification in 3D object is an hard task, not only due to its computational complexity but also because of the ambiguity of such decomposition. This task is even harder when aiming for automatically decompose 3D models. Therefore, we dedicated a significant part of our research on devising an effective technique for model segmentation, suitable to be used as a basis for building a shape thesaurus.

An ideal approach to model decomposition will segment the model according to semantic meaning of sub-parts or according to human visual perception of shapes. However, we consider that pursuing an automatic and time-efficient decomposition technique using these approaches in this stage of our research might be too ambitious. Therefore, we followed a distinct decomposition approach based on geometrical features of shapes to attain a solution that provides efficient automatic identification of sub-parts of models in a collection.

We developed a novel approach to shape decomposition that performs multilevel shape segmentation of each model based on the concept of decomposable regions. Decomposable regions are determined according to their distinctiveness regarding regions from all other models in the collection. We called this decomposition method Collection-aware Segmentation (CAS). This algorithm processes all models in the collection and produces a multilevel set of segments that compose these models. These segments (our ”3D words”) will be then used to construct the shape thesaurus, equivalent to a lexicon in textual information retrieval.

In a typical textual information retrieval system, the number of words in a lexicon is usually measured in millions [35]. However, to ensure time efficiency, we designed our shape thesaurus having just around a few thousand terms. To that end, instead of using directly all segments extracted from models, we grouped them according to their geometric properties and used these groups to construct the thesaurus. Indeed, something similar occurs in text indexing when different forms of the same word correspond to a unique term. Likewise, each term in the thesaurus of shapes represent a cluster of geometrically similar segments.

Thus, to make our approach useful and effective, segments obtained through model decomposition are meaningfully clustered. Terms for the thesaurus will spring up from these clusters and will be associated to all models containing segments in the correspond-

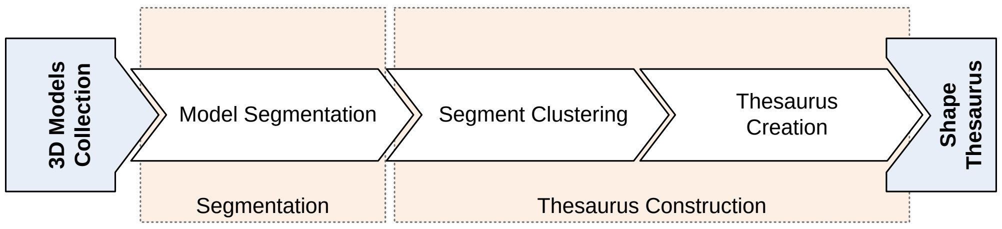


Figure 1.3: Pipeline for constructing a 3D shape thesaurus.

ing cluster. When looking for a model in a collection, the query is matched with the terms of the thesaurus and results are obtained using the measured similarities.

While the thesaurus is constructed by listing the estimated terms, the inverted index stores the list of segments belonging to the cluster that originated with each term. After creating a thesaurus, computing the inverted index is straightforward. We just look for segments associated to each term and create the corresponding inverted list with the models that contain these segments, *i.e.* the occurrences of that term in the collection.

Summarizing, our approach to index a 3D model collection using a shape thesaurus can be divided into three distinct steps, as depicted in Figure 1.3. The first is the segmentation of models in the collection, performed by the CAS algorithm. The second is the clustering of the resulting segments in order to identify the thesaurus terms. Finally, based on these terms, the last step creates a shape thesaurus and the corresponding inverted index. Basically, the classification process receives a 3D model and constructs the corresponding shape thesaurus to be used for 3D shape retrieval using part-in-whole matching.

The retrieval component of our approach combines 3D shape matching techniques with text information retrieval concepts. The overall idea can be described as a pipeline of two steps, illustrated in Figure 1.4. First, a typical shape matching is performed between the query and the terms. This matching process allows the identification of terms in the thesaurus geometrically similar to the given query. Notice that searching for similar terms in the thesaurus does not depend directly of the collection size. Instead, it only depends on the thesaurus size, which contains a small number of shapes, making the search process relatively fast.

After finding the similar terms, the remaining process is approximately the same as in text retrieval after identifying matches in the lexicon. The inverted index is used to identify models associated with the terms matching the query. These models correspond to the search result and can be identified in linear or even constant time, depending on indexing structure used and on the additional processing applied.

1.4 Contributions

The work described in this dissertation introduces a novel approach to 3D shape retrieval. It provides a solution for performing partial queries in large collections of three-dimensional models. This is achieved by combining concepts from textual information retrieval with 3D shape analysis and classification techniques in a modular framework for 3D shape retrieval. The major contributions of this research are the following:

- **Novel shape decomposition technique**
  To fulfill our needs of model decomposition, we devised a new technique to segment shapes automatically, without human intervention. Instead of considering only the model to be decomposed, the Collection-aware Segmentation (CAS) analyses all models in the collection and decompose them according to each other, *i.e.* model decomposition depends on the collection where it lies.

- **Collection indexing based on a shape thesaurus**
  A completely new concept was introduced during our research: a 3D shape thesaurus. This structure, together with an inverted-index, allows fast retrieval of 3D models with part-in-whole matching. The search time on a thesaurus-based retrieval solution is independent of the collection size, which makes this approach suitable for indexing large collections.

- **Generic framework for 3D shape retrieval with partial matching**
  The overall idea behind our approach diverges from existing approaches to shape retrieval. Thus the framework we devised combines a shape decomposition technique (that can be other than the one we suggest) with thesaurus-based indexing.

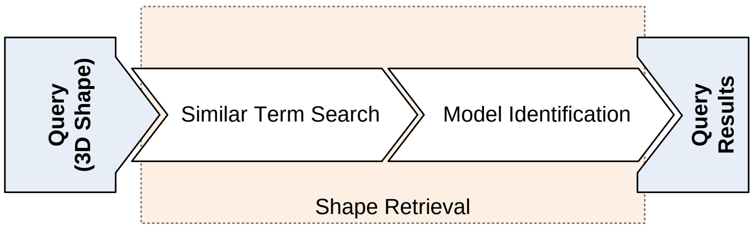


Figure 1.4: Pipeline for retrieving a 3D shape from the indexed collection.

1.5 Dissertation Outline

The remaining of this document is organized in five chapters. The next chapter presents the research background related with our work, namely by pointing the key players who developed pertinent and ground-breaking work in our field of research. In this chapter we also introduce the commonly used 3D model collections and benchmarks, as well as relevant details on query and matching techniques. Besides this research context information, this chapter includes surveys on the state-of-the-art in 3D shape description, shape segmentation and retrieval using partial queries.

In Chapter 3 we describe in detail the shape decomposition algorithm devised during this dissertation research. The CAS is a novel approach to shape segmentation and corresponds to an important contribution of our work. Chapter 4 describes our approach to retrieve 3D models with partial queries from large collections. This approach relies on a shape thesaurus and corresponding inverted index to provide efficient shape retrieval. We also describe the thesaurus construction, the indexing techniques and the retrieval methodology.

Chapter 5 presents experimental results obtained during our research, describing the behavior of the different algorithms. In this chapter we also present and discuss the results achieved by our retrieval solution as a whole. Finally, in Chapter 6 we present an overall discussion of our work, delineating conclusions and introducing perspectives for future research.

2 Background and State-of-the-Art

To familiarize the reader before deepen the description of our research, we present in this chapter the background we consider necessary to easily understand our work. We present the current 3D shape retrieval research context. To that end, we will firstly introduce the key players in this field of research, *i.e.* the groups whose research is closely related with ours and whose recent contributions are relevant to the progress of this research area in our opinion. We also present a list of existing 3D model collections, with special emphasis in shape benchmark collections.

Shape descriptors play a fundamental role in object analysis and classification, thus in 3D model retrieval. Therefore we dedicate part of this chapter to present a survey on 3D shape descriptors. Here, we introduce a taxonomy to classify shape descriptors and present the state-of-the-art on shape description techniques. Based on the conclusions of this survey we selected a shape descriptor to be used by our approach to 3D shape retrieval.

Additionally, in this chapter we will give the reader a quick overview of query strategies and similarity measures. These topics are not specific for 3D shapes, instead they are common for any kind of data. However, due to the importance of these subjects in our research, we consider important to provide the reader with the basic concepts behind querying and similarity measuring.

Finally, we present the more relevant solutions for content based retrieval of 3D models, followed by the state-of-the art in retrieval using partial queries. While the first will focus on solutions that support complete queries, the second will present several works that tackled the problem of partial matching.

2.1 Research Context

In this section we intend to introduce the reader into the current context of research on three dimensional shape analysis, classification and retrieval. To that end, we will start by presenting some people whose work is closely related or has a major relevance to our research. Additionally, we will overview existing model collections, with particular emphasis on benchmark collections.

2.1.1 The Key Players

Following the increasing importance of content-based retrieval of three-dimensional shapes, several research groups around the world focused their interests on this area. Producing an exhaustive list with all of them is difficult, with the additional risk of excluding some important researchers. Therefore, we will only present a short list containing those whose work we consider more relevant regarding our research focus.

Princeton During the last eight years, the Princeton Shape Retrieval and Analysis Group, leaded by Thomas Funkhouser, have been addressing key issues in shape-based retrieval and analysis of 3D models. Focusing on effective shape representations and query interfaces, they have developed a search engine for 3D polygonal models and later released a classified set of 3D models that can be used by researchers in this area. The Princeton Shape Benchmark is now widely accepted as a major benchmarking tool within 3D shape retrieval and classification.

PRECISE In Purdue University, Karthik Ramani leads the Purdue Research and Education Center for Information Sciences in Engineering (PRECISE). Their research lies at the intersection of design, shape analysis, and information sciences. They focus on developing representations for two and three-dimensional shapes for engineering and proteomics. In the last few years, they developed interfaces for querying, interacting, and navigating intelligently in 3D shape databases. Similarly to the Princeton team, researchers at PRECISE deployed a widely known benchmark focused on engineering models, the Engineering Shape Benchmark (ESB).

Konstanz The Multimedia Signal Processing Group chaired by Dietmar Saupe and the Databases, Mining and Visualization Group leaded by Daniel Keim, both from Konstanz University, join their efforts in a research project focusing the retrieval of three-dimensional shapes. The 3D Model Similarity Search project is part of the strategic research initiative on Distributed Processing and Delivery of Digital Documents and aims at effective content based model retrieval and efficient indexing and accessing methods.

CNR IMATI-Ge People at the Shape Modeling Group, a research team of the Institute of Applied Mathematics and Information Technology at Genova headed by Bianca Falcidieno, have been working on geometric modeling for several years. Their main research goal is to describe the shape of an object through the definition of geometric primitive entities and the classification of the reference context. To that end, they have been working in a variety of topics, ranging from graph comparison to free-form modeling, producing some interesting work on shape retrieval. Moreover, Bianca Falcidieno was coordinator of the AIM@SHAPE project.

AIM@SHAPE The Advanced and Innovative Models And Tools for the development of Semantic-based systems for Handling, Acquiring, and Processing knowledge Embedded in multidimensional digital objects (AIM@SHAPE) was a sixth framework program project that fostered the development of new methodologies for modeling and processing the knowledge related to digital shapes. This project embraced a multi-disciplinary field, which integrated Computer Graphics and Vision with Knowledge Technologies and built on using knowledge formalization mechanisms for linking semantics to shape or shape parts. Today, AIM@SHAPE turned into a Network of Excellence and a global reference to the our field of research.

Utrecht Also participating on AIM@SHAPE, Multimedia and Geometry group headed by Remco Veltkamp at the Center for Geometry, Imaging and Virtual Environments of Utrecht University has been working on multimedia information retrieval. Along with their research in areas such as music or image retrieval, they have developing some interesting work on 3D shape analysis and retrieval, namely on 3D facial models.

FOX-MIIRE In the Multimedia, Images, Indexing and Recognition (FOX-MIIRE) research group at the University of Sciences and Technologies of Lille, Jean-Philippe Vandeborre, Mohamed Daoudi and their teams are working on three-dimensional model indexing and topological analysis. The recently published FOX-MIIRE 3D-Models Search Engine based on adaptive views clustering algorithm is the first search engine that accepts 3D-Models retrieval from photos [10]. Additionally, they developed a 3D retrieval application for mobile devices, which were presented in the 2007 ACM International Conference on Image and Video Retrieval. The FOX-MIIRE group is a partner of DELOS Network of Excellence.

DELOS Partially funded by the European Commission in the frame of the Information Society Technologies Programme, DELOS is a Network of Excellence on Digital Libraries. The main goal behind DELOS is to provide global access to knowledge contained in the digital collections created by organisations and individuals around the world. To that end, DELOS is conducting a joint program of activities aimed at developing the next generation of Digital Library technologies, based on sound comprehensive theories and frameworks for the life-cycle of Digital Library information. Within the large number of research topics covered by DELOS, the work leaded by Alberto del Bimbo from the University of Florence is of major relevance for our research.

Firenze At the Media Integration and Communication Center of the University of Florence, researchers leaded by Alberto del Bimbo presented a prototype system for content based retrieval of 3D objects, briefly described in Section 2.4. Besides, they have been developing relevant research on curvature maps and spin images as descriptors for 3D shape retrieval.

Boğaziçi At the Image and Video Processing Group of Boğaziçi University, Bülent Sankur's team is developing, together with people from the multimedia, vision and graphics laboratory at Koç University relevant research on 3D shape retrieval. In the last two years, they focused their research on analysis and description of 3D shapes and on 3D face recognition, achieving remarking results.

2.1.2 Model Databases

Besides identifying the key players in three dimensional shape retrieval, it is important to identify the existing sources of three dimensional models to which shape retrieval can be useful. Indeed, as a result of recent advances on modeling, digitizing and visualizing techniques, there are a large number of 3D model collections available for usage both on the internet and in domain-specific databases. Since it will be hard and out of scope of this work to exhaustively enumerate all of this databases, we will refer in the following paragraphs just a few of these databases offering public access.

The Protein Data Bank (PDB) [21, 22], an archival for macro molecular structures, is an early example of such collections now considered the single worldwide archive for biological macro molecules. Another example of a 3D model collection was produced during the Digital Michelangelo Project [87, 86], an archive with digital models of Michelangelo sculptures and architecture containing an aggregate of nearly eight billion polygons.

The National Design Repository [107, 127, 108] is a collection of public domain computer-aided design data from a variety of sources. This data includes tens of thousand solid models and CAD files. Recently, Google^{TM} released the Google^{TM} 3D Warehouse [69], an online service that hosts 3D models of existing objects (mainly buildings) created in Google^{TM} SketchUp [68].

Besides the collections referred above, several research databases are available for public use. An exhaustive list of these databases will be too long to include on this document. However, Akgul compiled a rather complete list and published it in his thesis dissertation [4]. Relevant for our work are the databases associated with shape benchmarks, very useful for evaluating retrieval algorithms.

2.1.3 Benchmarking 3D shape retrieval

Like in other information retrieval research areas, effective testing and comparison of different techniques in multimedia information retrieval requires the existence of widely accepted evaluation frameworks, such as the TRECVid [119].Regarding 3D shape retrieval, a few benchmarks are already publicly available. Below we present most relevant with respect to our research.

Princeton Shape Benchmark In the three-dimensional models domain, the Princeton Shape Benchmark (PSB), deployed by Thomas Funkhouser team [115], has become the standard and is being widely used for evaluating various representation methods and shape retrieval techniques. The PSB provides a repository of around 1,800 models collected from the web and software tools for evaluating shape-based retrieval and analysis algorithms.

Shape Recognition Contest A retrieval contest for 3D models was established a few years ago, following the experience of other information retrieval research areas. Currently, Shape Recognition Contest (SHREC)is widely accepted as a reference benchmark in 3D shape retrieval and is expected to become an objective tool for evaluating and comparing 3D retrieval techniques. This contest has been organized by the network of excellence AIM@SHAPE.

AIM@SHAPE The AIM@SHAPE network of excellence, introduced in Section 2.1.1, released their shape repository which makes several 3D models available for researchers to compare shape matching algorithms. It is a shared repository populated with a collection of digital shapes and a integral part of the framework of tools and services for modeling, processing and interpreting digital shapes, developed within the AIM@SHAPE project.

SHREC 2009 partial retrieval track dataset The latest edition of the shape recognition contest (SHREC 2009) included a track on partial retrieval of 3D models. One of the objectives of this track is to evaluate solutions that perform retrieval of objects using partial queries. To that end, a set of queries and a collection of target objects are given. Unfortunately, these datasets were made available in December 2008, after we have finished our experiments.

Engineering Shape Benchmark Despite the existing work on shape repositories, there has been limited work in developing domain dependent benchmark databases for 3D shape searching. Indeed, the most popular benchmarks do not include model classes from specialized application domains. To overcome this in the engineering field, the PRE-

CISE group at Purdue proposed [71] a benchmark database for evaluating shape-based search methods relevant to the mechanical engineering domain. They developed a publicly available Engineering Shape Benchmark (ESB) for comparing various shape-based search algorithms.

Since our research focused on engineering CAD models and the Engineering Shape Benchmark (ESB) is widely accepted as a valuable benchmark in this domain, we used this collection during our experiments.

Benchmarking CAD search techniques Having also identified the problems referred above, Bespalov *et al.* presented benchmark datasets to assess the relevance of existing 3D shape retrieval techniques for engineering problems [23]. They proposed several distinctive repositories for evaluating techniques for automated classification and retrieval of CAD objects. These collections includes sets of primitives, industrial models and LEGO® models. The models on these datasets were extracted from the National Design Repository [107, 127, 108] and manually classified by their appearance, manufacturing process and function, since these are typical classification schemes for CAD models.

2.2 From Shapes to Descriptors

Three-dimensional models are useful in a wide range of domains. This application diversity lead to a myriad of domain-specific collections with several distinct forms of model representation [39]. However, these different representations can always be converted or approximated to a more generic one, such as a polygonal mesh, which could be interpreted by classification and retrieval algorithms.

These algorithms usually rely on a numerical representation to describe both models and queries. The most common numerical representation for 3D shapes are feature vectors. These feature vectors allow representing models and queries as points in a multi-dimensional space. Thus, searching for similar objects is reduced to a search in the feature-vector space instead of comparing objects directly. Indeed, the usage of feature vectors is the standard approach for multimedia retrieval [52]. Generally, a feature vector, also referred as descriptor or object signature, is a set of values extracted from a multimedia object that describe it numerically in a high-dimensional space.

An important goal of any 3D shape description approach is to preserve the maximum shape information on a feature vector with the lower dimensionality possible. Indeed, finding such computational representation of a shape is considered as the primary challenge in building a shape based retrieval system [55].

In this section we will briefly describe existing approaches to 3D shape description. Although we tried, in this document to produce a comprehensive survey, readers are encouraged to consult recent publications entirely dedicated to this topic, as the survey by Tangelder and Veltkamp [128] or the state-of-the-art review published by Iyer *et al.* [70].

2.2.1 Taxonomy of 3D Shape Descriptors

In a noticeable work, Bustos *et al.* extensively surveyed methods for 3D shape descriptor computation [37] and proposed a taxonomy for these methods. This taxonomy of 3D shape descriptors, although well-grounded, was not universally accepted due to the variety of distinct approaches to shape descriptor categorization. Nevertheless, we will follow a general classification scheme, very similar to the taxonomy proposed by Bustos *et al.*. This scheme, depicted in Figure 2.1, emphasizes the specific way to exploit the shape information contained in a 3D object and was suggested by Akgül [3].

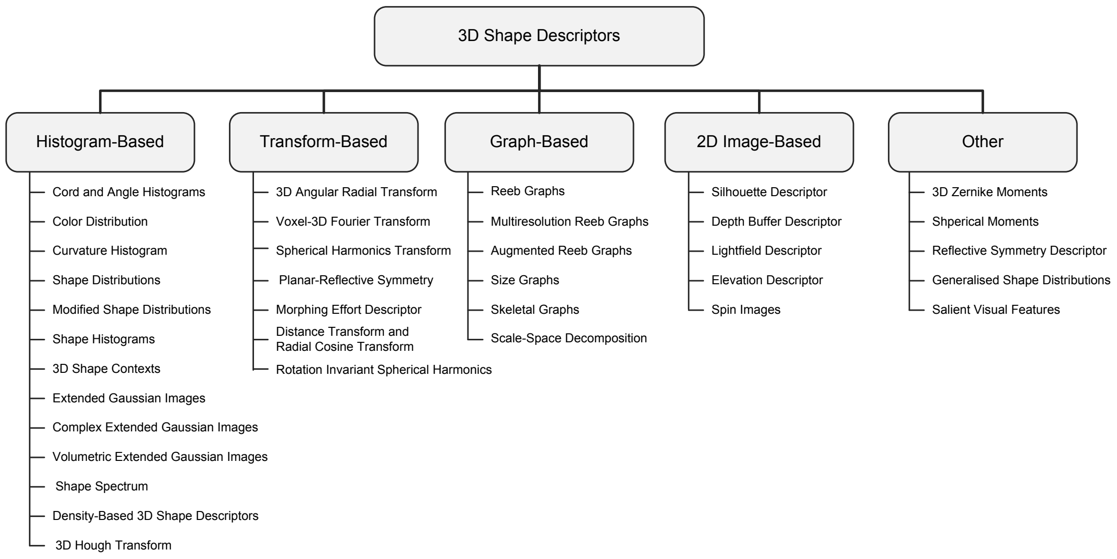

3D Shape Descriptors
Histogram-Based
Transform-Based
Graph-Based
2D Image-Based
Other
Cord and Angle Histograms
Color Distribution
Curvature Histogram
Shape Distributions
Modified Shape Distributions
Shape Histograms
3D Shape Contexts
Extended Gaussian Images
Complex Extended Gaussian Images
Volumetric Extended Gaussian Images
Shape Spectrum
Density-Based 3D Shape Descriptors
3D Hough Transform
3D Angular Radial Transform
Voxel-3D Fourier Transform
Spherical Harmonics Transform
Planar-Reflective Symmetry
Morphing Effort Descriptor
Distance Transform and Radial Cosine Transform
Rotation Invariant Spherical Harmonics
Reeb Graphs
Multiresolution Reeb Graphs
Augmented Reeb Graphs
Size Graphs
Skeletal Graphs
Scale-Space Decomposition
Silhouette Descriptor
Depth Buffer Descriptor
Lightfield Descriptor
Elevation Descriptor
Spin Images
3D Zernike Moments
Shperical Moments
Reflective Symmetry Descriptor
Generalised Shape Distributions
Salient Visual Features

Figure 2.1: Taxonomy of 3D shape descriptors.

The taxonomy we followed classifies descriptors according to the technique used to build the shape numerical representation. As a result of this classification methodology, the taxonomy adopted in this dissertation is divided into five categories: histogram based; transform-based; graph-based; image-based; and other shape descriptors. In the following sections we will present the most relevant approaches in each category.

2.2.2 Histogram-based Descriptors

Widely used in computer graphics to represent the color distribution in an image, the color histogram is computed by counting the number of pixels for each color. Adopted to 3D shape description, an histogram is often referred as an accumulator that collects numerical values of certain features calculated from the shape to represent. Based on this loose definition, many 3D shape descriptors can be considered as histogram-based methods, although they are not based on histograms in rigorous statistical sense.

Cord and Angle Histograms

The use of cord and angle histograms for 3D shape descriptors were presented by Paquet *et al.* in [103, 104]. The authors define a cord as a vector that goes from the centre of mass of an object to the center of mass of a bounded region on the surface of the object. Then, compute the descriptor based in a collection of three histograms. The first and second histogram represents the distribution of the angles between the cords and the first and second reference axis, respectively.The third histogram provides the distribution of the radius.

Despite its computational simplicity and efficiency, this approach simplifies triangles to their centers and does not consider the size and shape of the mesh triangles. Moreover, since only global features are used to characterize the overall shape of the objects this method is not very discriminating about objects details, but their implementation is straightforward. It is often used in object retrieval as an active filter, after which more detailed comparisons can be made, or can be used in combination with other methods to improve shape descriptors.

Color Distribution

Paquet and Rioux [103] proposed, along with the cord and angle histograms, a peculiar color based descriptor for 3D shapes. In their approach, a voxelised representation of the 3D object, where each voxel has a color value associated with it. This value is computed using information from the texture map, material properties and vertex color extracted from the object representation.

Authors suggest three distinct approaches to compute the histogram that describes the object. If color location is irrelevant, they compute the color histogram of the object or, alternatively, they use the dominant colors to compute three histograms. Otherwise, if color location is relevant, authors suggest using a wavelet approach based on a model with six dimensions: three for position and three for color. The six-dimensional wavelet transform is computed and then used to construct a histogram.

Curvature Histogram

Koenderink and van Doorn [81] defined the curvature index as a function of the two principal curvatures of the surface. This index gives the possibility to describe the shape of the object at a given point. However, it loses the information about the amplitude of the surface shape and is sensitive to noise. Later, Vandeborre *et al.* used this index to compute a curvature histogram of the shape [131], a local descriptor invariant to geometric transformations.

Shape Distribution

Osada *et al.* proposed a method for computing shape signatures for arbitrary 3D polygonal models [101]. They use a collection of shape functions computed with random sampling of the surface of the 3D object to describe. This shape function measures global geometric properties of the shape, based on distance, angle, area and volume measurements between random surface points.

Authors suggest five distinct one dimension functions to measure the object properties, which are quick to compute, easy to understand, and produce distributions that are invariant to rigid motions (translations and rotations). To compute these functions, the object surface is sampled, with a predetermined density. The shape descriptors are constructed from the histograms of a set of the above mentioned shape functions, controlling histogram accuracy through sampling density. The descriptor of shape distributions is fast, simple to implement, and useful for 3D shapes discrimination. However, the proposed shape functions are not adequate to fully describe the 3D shape effectively. Indeed, this approach distinguish models in broad categories very well, but perform poorly when used to discriminate between models with similar gross shape properties but vastly different detailed shape properties.

Modified Shape Distribution

Ohbuchi *et al.* [99] extended the $D2$ shape functions proposed by Osada *et al.*, by devising a set of shape features that are tolerant to topological variations and geometrical degeneration. In the proposed technique, the $mD2$ is similar to original $D2$, but authors used a quasi-random number sequence[1] to select points instead of the pseudo-number sequence[2] suggested by Osada *et al.*.

Shape Histograms

The shape histograms were proposed by Akerst *et al.* as an intuitive approach to describe 3D solid models [9]. In this approach the space where the object resides is partitioned using one of three space decomposition techniques: a shell model, a sector model and a spider web model (depicted in Figure 2.2).In any of these techniques, each cell of the decomposed 3D space correspond to a bin in the histogram. Then, the histogram can be constructed by accumulating the surface points in the bins.

The shape histograms method is an intuitive and discrete representation of complex spatial objects. However, authors illustrate the shortcomings of Euclidean distance to compare two shape histograms and make use of a Mahalanobis[3] quadratic distance measure taking into account the distances between histogram bins. On the other hand, this approach needs pose normalization to be performed in the pre-processing stage.

[1]The quasi random number sequence, is not really random since a predictable sequence of numbers is generated. However, the numbers generated will provide uniform sampling.

[2]A pseudo random number sequence exhibits statistical randomness while being generated by an entirely deterministic causal process.

[3]A short explanation of Mahalanobis distance can be found in Section 2.3.2

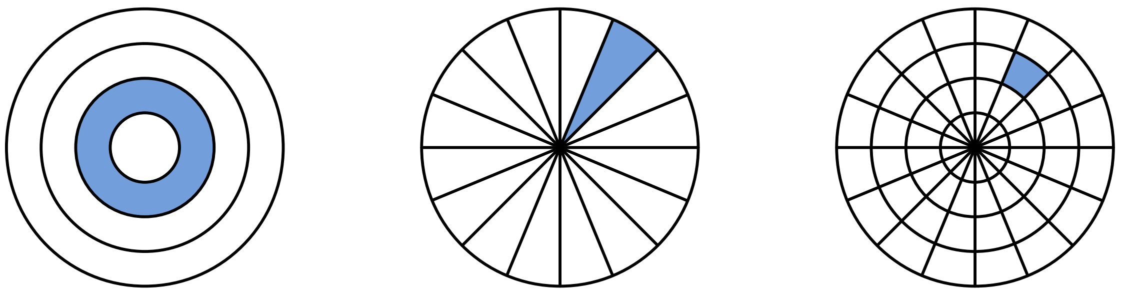


Figure 2.2: 2D examples of the space decomposition techniques proposed by Akerst *et al.* in [9]. With a single bin marked, are depicted, from left to right, shell, sector and spider web model.

3D Shape Contexts

The shape contexts was introduced by Belongie *et al.* [19] as a descriptor for computing similarity between 2D images. Using reference points, authors assign a shape context to each one, by capturing the distribution of the remaining points relative to it. Körtgen *et al.* extends the 2D shape contexts into a 3D shape description and combines it with the shape histogram, thus proposing a set of descriptors called 3D shape contexts [84].

To compute the 3D shape contexts, a set of points are sampled from the shape boundaries, as in the shape distribution approach [101]. Then, the vectors originating from one sample point to all other points in the shape are computed. Using the distribution over relative positions, is computed for this point a coarse histogram of the relative coordinates of the remaining points. Indeed, this histogram is an adapted version of one of the Ankerst's [9] shape histograms centred upon the sample point. This method is applied to all sampled points, producing a 3D shape descriptor that is composed by a set of histograms.

Extended Gaussian Images

Defined by Horn in [67], the Extended Gaussian Image (EGI) is a histogram-based technique to represent the shapes of surfaces that define a 3D object. Initially devised for recognition in machine vision systems, this approach was later adapted for pose determination and for computing 3D shape descriptors [75], by means of the complex extended Gaussian images (CEGI) representation.

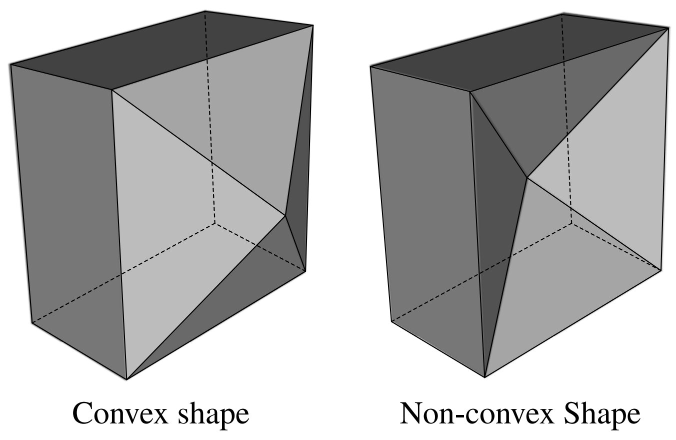


Figure 2.3: Two objects with the same EGI descriptor, but different volumes, thus different VEGI descriptors.

Basically, the EGI of a 3D object is a histogram that records the variation of surface area with surface orientation. The CEGI concept, presented by Kang and Ikeuchi, extended the EGI approach by taking also into account the distance of faces to the origin.

More recently, Zhang *et al.* further extended the EGI representation to capture the volume distribution of an object without canonical alignment, while maintaining the translation, scale and orientation invariance. The Volumetric Extended Gaussian Image (VEGI) shape descriptor [144] is thus able to differentiate between convex and non-convex shapes that share the same EGI such as the objects depicted in Figure 2.3. Furthermore, VEGI directly extracts shape features avoiding pose normalization, since it does not depends on canonical alignment of shapes.

3D Hough Transform

Based on the 2D generalized Hough transform [18], Zaharia and Prêtoux proposed in [143] the 3D Hough Transform (TDHT) to represent a three-dimensional object. In their later work [142], developed canonical 3D Hough transform descriptor (C3DHT). These descriptors are constructed by accumulating points within a set of planes in 3D space. Indeed, TDHT can be considered as a generalized version of EGI. In fact, Agkül *et al.* have experimentally proven [6] that the TDHT descriptor captures the shape information better than the EGI descriptor.

Shape Spectrum

The shape spectrum was introduced by Dorai and Jani [49] as a view-based representation of 3D free-form objects. The shape spectrum characterizes quantitatively the object shape by summarizing the area on the surface of an object at each shape index value. The shape index is a local geometrical attribute of a 3D surface, expressed as the angular coordinate of a polar representation of the principal curvature vector. It provides a scale for representing salient elementary shapes and is invariant with respect to scale and Euclidean transforms. However, an important problem regarding the shape index, in fact all curvature-related quantities, is the estimation unreliability.

Such shortcoming was alleviated by Zaharia and Prêtoux in [142] by augmenting the shape index histogram by two additional attributes named planar surface and singular surface. Then, the proposed 3D Shape Spectrum Descriptor (TDSSD) was applied to 3D retrieval within the MPEG-7 framework for multimedia content description. The TDSSD of a 3D mesh is defined as the histogram of the shape index values, calculated over the entire mesh.

The TDSSD locally characterizes free-form surfaces represented as discrete polygonal 3D meshes. One major advantage of this descriptor is its generality, since 3D meshes may include open surfaces that have not an associated volume. Furthermore, inherited from the shape index properties, the TDSSD is invariant with respect to scale, translation, rotation and reflection transforms. On the other hand, this descriptor, as a simple local feature representation, is better to be combined with some global representation schemes to effectively describe 3D object for shape retrieval purposes.

Density-based shape descriptors

A density-based descriptor of a 3D object is defined as the sampled probability density function[4] (PDF) of some surface feature. The feature is local to the surface patch and treated as a random variable. This analytical framework was proposed by Akgül and Sankur [6, 7] to extract 3D shape descriptors from local surface features characterizing the object geometry. In [5], the authors suggest using as features the radial distance, the

[4]The probability density function [2], also called probability function or density function, is a function that represents a probability distribution in terms of integrals.

radial direction, the normal direction, the radial-normal alignment and the tangent-plane distance.

Akgul mentioned in [6] that the density-based descriptor shown very good results when compared with other well-known 3D shape descriptors.However, the features used in this descriptor are neither scale nor rotation-invariant and, since the method depends on them, pose normalization must be accomplished during the preprocessing phase.

2.2.3 Transform-based Descriptors

Instead of estimating the shape descriptor from the three dimensional space, some approaches rely on mathematical transformations to switch from the spatial domain to a more suitable one and compute from there a shape descriptor. These new spaces are usually the frequency domain, although some recent approaches use different spaces.

Voxel-based 3D Fourier transform

Vranić and Saupe suggest switching from the spatial domain to the frequency domain via a 3D Fourier transform [135] (3DFT). Authors start by performing pose normalization, then voxelise the object using the so-called bounding cube[5] (BC). This voxelisation is achieved by subdividing the BC into N^3 equal sized cubes (cells) and calculating the proportion of the total surface area of the object inside each cell. Then, authors apply a 3D discrete Fourier transform to the voxelised model, *i.e.* calculated values in cells, to compute the descriptor that represents the feature in the frequency domain.

Distance Transform and Radial Cosine Transform

Following the initial ideas from Vranić and Saupe of applying Fourier transforms to feature extraction, Dutagaci *et al.* proposed estimating a 3D discrete Fourier transform descriptor using two different voxel representations of 3D objects [50], namely, binary and continuous, as depicted in Figure 2.4. While in the first case the voxel values are simply

[5]Vranić and Saupe consider the bounding cube of a 3D-model as the tightest cube in the canonical coordinate frame that encloses the model, with the center in the origin and the edges parallel to the coordinate axes.

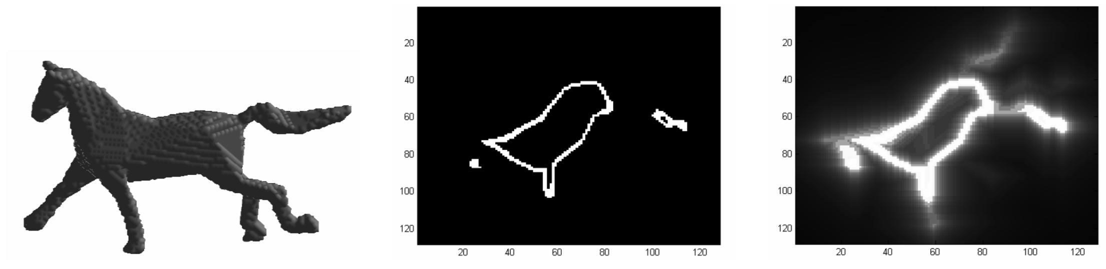


Figure 2.4: From left to right: voxelised 3D object; cross section of the binary function; cross section of the inverse distance function (Figures taken from [50] © 2005 IEEE).

set to 1 in the surface of the object and 0 elsewhere, in the continuous representation the space is filled with a function of distance transformation.

Additionally, Dutagaci *et al.* suggest the use of 3D radial cosine transform (RCT) as an alternative to 3DFT [50]. The RCT coefficients constitute a set of rotation invariant shape descriptors. Such descriptor represents a 3D model with a small number of features, thus being easy and fast to be calculated. However, the retrieval results of RCT are generally worse than 3DFT or some other approaches. Therefore, it is always considered to be mainly suitable to be used together with other descriptors as an preliminary filter.

Spherical Harmonics Transform

In their initial approach, Saupe and Vranić [134] took a coarse number of samples of the spherical extent function [6] to construct a feature vector. However, this simple technique is sensitive to small perturbations of the model. Therefore, to improve the robustness of this approach, they later proposed [111] extracting a dense sample of the spherical function and then compute spherical harmonics for this function to describe the shape.

The descriptor accuracy can be defined by changing the parameter that defines the sampling size, as well as the number of spherical harmonic coefficients to use. Figure 2.5 depicts the reconstruction of an object by using different number of coefficients. The shape descriptor is derived from these of coefficients, thus providing an embedded multi-resolution approach for 3D shape description.

[6] A 3D object can be characterized by a function on the sphere. To that end, rays are cast from the center of mass the object and is estimated, for each ray, the value of the distance from origin to the last point of intersection with the object surface. These values yield a sample of such function, called spherical extent function for a shape.

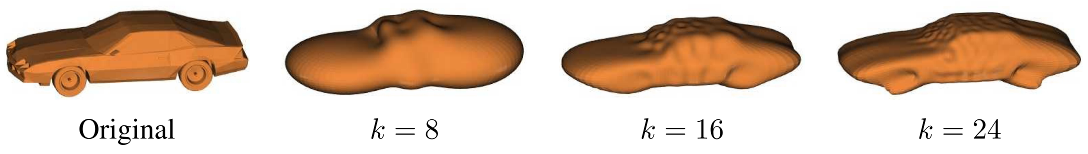


Figure 2.5: Multi-resolution representation of the spherical extent function applied to the original model using k^2 spherical harmonic coefficients (Figures taken from [111]).

Vranić and Saupe [135] improved the robustness of the proposed feature vector by taking samples of the spherical function at many points, but characterizing the map by just a few coefficients in the spectral domain. In [136], they enhanced the spherical harmonics transform approach described above by taking in account the orientation of the surface, along with the extent vector. Later, Vranić [132] proposed considering a set of concentric spheres with different radii, instead of a single one, thus using a collection of spherical functions to compute the descriptor.

Morphing Effort Descriptor

On a slightly different approach, Yu *et al.* proposed measuring the amount of effort required to morph[7] a 3D object into a canonical sphere [140] to describe the geometry and topology of the object. To that end, they use a descriptor similar to the spherical extent function together with a descriptor counting the number of intersections from a ray casted from the origin with the object surface. A surface penetration map is constructed by counting the shape surfaces intersected by a ray shot from the center of the sphere and provides information about object topology and concavity.

Rotation Invariant Spherical Harmonics

After identifying several limitations of canonical alignment used in other approaches, Kazhdan *et al.*proposed [78] an alternate method to obtain rotation invariant representation of three-dimensional objects based on spherical harmonics. First, the object to describe is converted into a voxel grid. Then the object is intersected with a set of concentric spheres and a spherical function is constructed from voxel values for each sphere. Next, the frequency decomposition of each one of these functions is computed, as well as

[7]Morphing is a technique that changes one object into another through a seamless transition, generally by producing a sequence of intermediate objects.

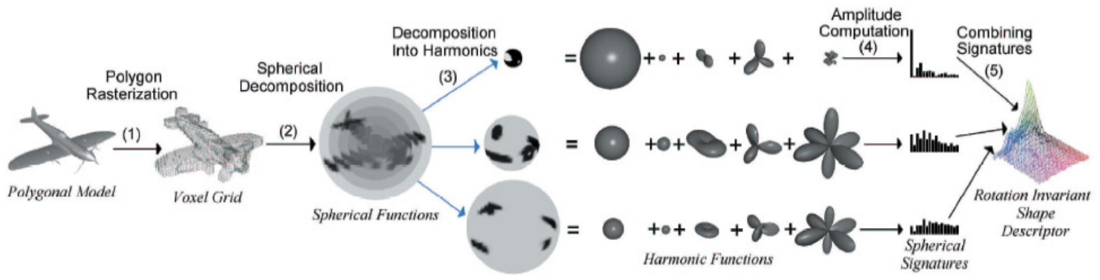


Figure 2.6: Princeton methodology for computing spherical harmonics shape descriptor (Figure taken from [56] © 2003 ACM).

the norms of each frequency component at each radius. The resulting rotation invariant descriptor is a 2D grid indexed by radius and frequency. Following this idea, the Princeton group derived a practical methodology [56], illustrated in Figure 2.6, to compute rotation invariant descriptor using spherical harmonics.

Papadakis *et al.* presented recently [102] a 3D shape retrieval methodology also based on spherical harmonics. The proposed model decomposition and feature extraction is very similar to previous approaches. They compute the spherical functions using not only the intersections of the surface with emanating rays but also points in the direction of each ray which are closer to the origin than the furthest intersection point.

3D Angular Radial Transform

Adopted as region-based shape descriptor in MPEG-7 standardization, the angular radial transform [34] is a moment based description method that expresses pixel distribution within a 2D region. Ricard *et al.* proposed a generalization of this descriptor to index 3D models [109]. The 3D angular radial transform (3D ART) descriptor preserves the properties of the original 2D descriptor, such as robustness to rotation, translation, noise and scaling. Moreover, the 3D ART produces compact descriptors and allows short retrieval times. Authors argue that their approach outperforms the spherical harmonics descriptor in speed while keeping a close accuracy.

Planar-Reflective Symmetry Transform

In [105] Podolak *et al.* introduced the planar reflective symmetry transform (PRST). The PRST is a transform from the space of points to the space of planes that captures

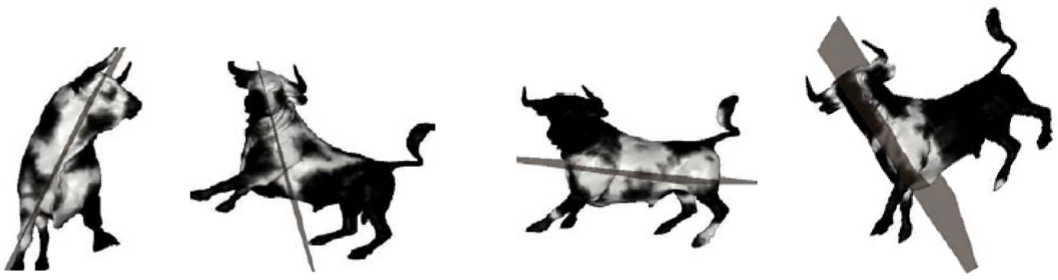


Figure 2.7: Symmetries with respect to planes representing the four strong local maxima of the PRST. (Figures taken from [105] © 2006 ACM).

a continuous measure of the symmetry of a shape with respect to all planes through its bounding volume. This transform combines and extends previous work that has focused on global symmetries with respect to the center of mass in 3D meshes [77, 79], briefly described in Section 2.2.6, and local symmetries with respect to points in 2D images [141].

Authors also provide an iterative refinement algorithm to find local maxima of the transform precisely. In Figure 2.7 triangles of the model are grayed to show how symmetric they are with respect to the plane of symmetry displayed. The triangles with highest symmetry values are lighter than the ones with less or no reflection in the given plane.

2.2.4 Graph-based Descriptors

The above referred approaches focus on describing the geometry of model to classify, ignoring or giving just few relevance to topological information. At most, these approaches attempt to integrate topological information in the shape descriptor of the object. In contrast, graph-based approaches extract both topology and geometry of 3D objects, focusing on topological relationships between object components and using graphs to represent such relationships. These approaches are generally more complex than the previous ones, but despite their ability to encode geometry and topology, they do not generalize for any type of 3D shape, forcing each approach to restrict its scope to a specific type of object. Therefore, graph-based approaches are not effective in general-purpose retrieval applications.

Furthermore, due to the complexity associated to graph matching, alternative solutions to graph isomorphism are used, such as application of techniques from spectral graph theory to convert graphs into numeric descriptors [54, 117]. However, during re-

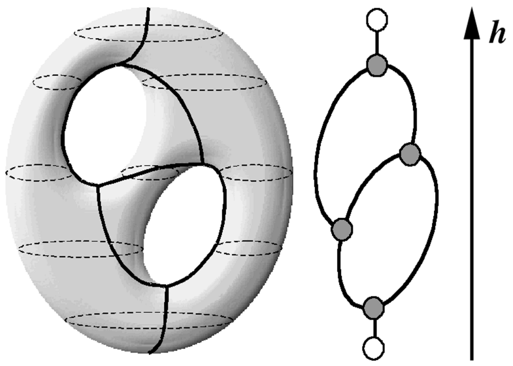


Figure 2.8: Reeb graph of a bi-torus and some cross sections. (Figure taken from [27])

cent years several researchers focused their attention in graph-based descriptors due to its potential. In the following paragraphs we will present the most commonly used approaches for graph-based descriptors.

Reeb Graphs

Back in 1946, Georges Reeb proposed considering a topological graph defined as a quotient space of a manifold[8] which, under opportune hypotheses defines the skeleton of the manifold itself [106]. Indeed, the Reeb graph is just a topological skeleton determined using a scalar function defined on an 3D object. To automatically construct a Reeb graph, Shinagawa and Kunii [116] proposed defining a scalar function and using a series of cross-sections of the object to determine nodes and arcs of the graph.Figure 2.8 illustrates a Reeb graph of a bi-torus computed using a height function as mapping function.

Several approaches to 3D shape classification and retrieval based on Reeb graphs were proposed in recent years. Biasotti *et al.* obtain graphs by using different quotient functions f and suggest that a good choice of f is necessary to achieve good matching results [30]. Indeed, they conclude that f function must be determined based on the object type, since the same function produces different matching performance for different kinds of models. For instance, the authors proved that using the integral geodetic distance as a quotient function is especially suited for articulated objects.

[8]A manifold is an abstract mathematical space that is locally Euclidean. This means that around every point there is a neighborhood that is topologically the same as the open unit ball in $\mathbb{R}^n$, *i.e.* the neighborhood resembles Euclidean space, but the global structure may be more complicated.

Multiresolution Reeb Graphs

Hilaga *et al.* introduced the concept of topology matching for 3D object retrieval [65], describing a matching method best suited for articulated objects. Their method constructs Reeb graphs at multiple levels of resolution of a function, the Multiresolution Reeb Graph (MRG). According to the chosen function, the resulting descriptor has certain properties for a different kinds of models. To quickly determine similarity between polyhedral models they compare the graphs using a coarse-to-fine strategy while preserving the consistency of the graph structures, which results in results in establishing a correspondence between the parts of objects. This graph matching is achieved through sophisticated heuristics proposed by authors and improved later by Tung *et al.* [130].

Augmented Reeb Graphs

Tung and Schmitt [129] took further the approach by Hilaga and augmented the Reeb graph by storing geometric attributes in each node, since the original method only takes into account topological information, which is often insufficient for effective shape matching. In this approach authors use, as geometrical information, features such as the cord histograms, local curvature and volume associated with each node. Moreover, they also provided a new topological coherence condition to improve the graph matching.

Using the proposed Augmented Reeb Graph (ARG), Tung and Schmidt could overcome some issues raised during matching with Hilaga's MRG. For instance, graph edges topologically similar might not really be geometrically similar, thus being wrongly matched, as depicted in Figure 2.9. This figure illustrates the gain obtained by introducing geometrical information in the nodes. While matching without geometrical information (left) legs can be matched with arms, since they are topologically equivalent, by adding geometrical information, arms and legs are well matched.

Scale-space Decomposition

Aware of MRG drawbacks when models become geometrically and topologically detailed, Bespalov *et al.* studied the application of Hilaga's method to matching of complex machined parts [24]. They stated that such solution produces poor results when directly

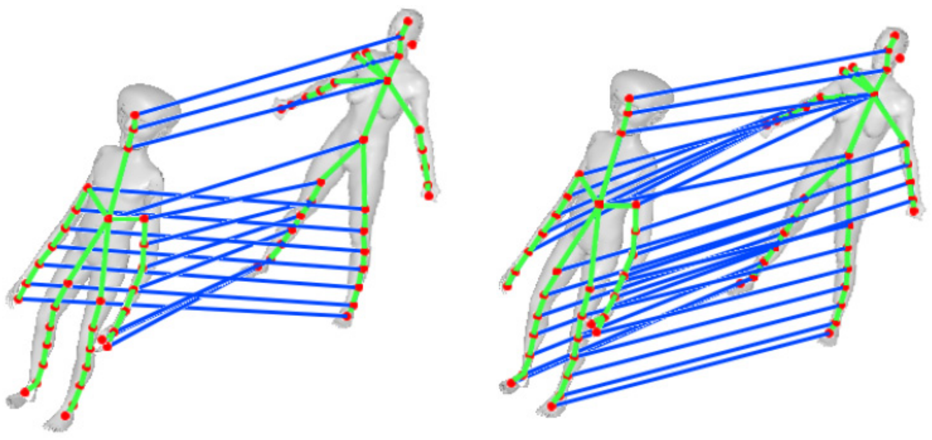


Figure 2.9: Results of shape matching with MRG *versus* ARG (Figure taken from [129] © 2004 IEEE).

applied to 3D solid models in engineering databases. Since for this kind of models topological insensitivity is important, they conclude that some improvements should me made to MRG technique.

Thus, they present in [26] an alternative to Hilaga's approach. Their method computes the scale-space decomposition of a shape, represented as a rooted tree. Through spectral decomposition the problem is of matching reduced to that of computing a mapping and distance measure between vertex labeled rooted-trees. Indeed, authors claim that their method represents a computationally efficient approach to matching of 3D models, enabling highly accurate matching of solid models of 3D mechanical parts.

Size Graphs

Following the previous approaches, Biasotti *et al.* use the Reeb graph to construct a centerline skeleton of a 3D model and apply a size function to create a size graph [29]. Their idea is to associate with a 3D object a graph (G^f, ϕ), where G^f is the centerline skeleton computed using the quotient function f and ϕ is a measuring function labeling each node of the graph with local geometrical properties of the model.

In this approach, authors consider four distinct mapping functions f, namely the distance from the barycenter, the distance from the center of the bounding sphere, the integral geodetic distance and the topological distance from curvature extrema [96]. Based on these functions, a centerline skeleton is extracted from the original model. Then, for each node of this skeleton, the value of function ϕ must be calculated to obtain the size graph. Biasotti *et al.* suggest measuring a set of features of the corresponding region on the model, such as the area of the region or the minimum, maximum and average distance of the barycenter of the region to region vertices. To compare models authors use the matching between their size functions, as discussed by d'Amico *et al.* in [46].

Skeletal Graphs

In a slightly different approach, skeletons can be derived from solid objects and represented as a direct acyclic graph (DAG). These skeletons capture important information about the object. However, when using shape skeletons in 3D object retrieval, two major challenges arise. Suitable skeleton computation algorithms and similarity functions should be defined. Sundar *et al.* presented a framework that provides both [121]. They propose, as shape descriptor for three-dimensional models, a skeletal graph encoding geometrical and topological information of the object. Then they apply graph matching techniques to match the skeletons and compare them.

2.2.5 Image-based Descriptors

An approach completely different from the previous ones consider representing a three dimensional model in a set of two dimensional spaces. The basic idea behind such approach is that when two 3D models are similar, images captured from the same points of view are also similar. From this idea several researchers were able to reduce the problem of comparing 3D shapes to image matching. Thus, taking advantage of the existing reasonable amount of work in this area capable of producing good retrieval results.

Spin Images

Johnson and Herbert proposed a 3D object recognition system [73] based on matching surfaces using the spin image representation. To produce spin images authors use oriented points on the model surface, *i.e.* points associated with the surface normal at that point. Each oriented point corresponds to a spin image and defines a local coordinate system using the tangent plane.

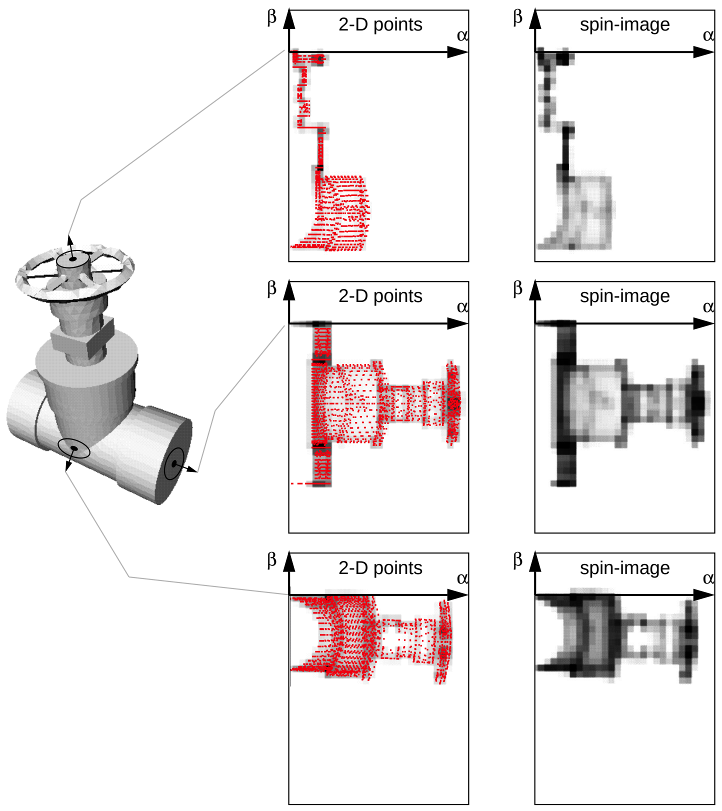


Figure 2.10: Spin-images for three oriented points on the surface of a model of a valve (Figure taken from [72]).

To create the spin image, is created a 2D accumulator indexed by two coordinates defined with respect to the oriented point. This accumulator can be thought of as an image where dark areas in the image correspond to bins that contain many projected points. Figure 2.10 shows the projected coordinates and spin images for three oriented points on a model of a valve. For 3D object matching, spin images can be constructed for every vertex in the surface mesh, producing a set of two-dimensional histograms representing the object geometry.

The spin images approach to 3D shape retrieval was later improved by de Alarcón *et al.* [47]. Instead of compression method proposed by Johnson and Herbert to solve the high space complexity of their approach, Alarcón *et al.* suggest data reduction by clustering the spin image set using a self organising map algorithm to group similar spin images, followed by a clustering algorithm. More recently, Assfalg *et al.* suggested a retrieval method based on spin images, but using global features [14]. In their approach, spin images are used to derive a view-independent object description.

Silhouette Descriptor

Vranić presented in [133] a 3D shape descriptor based on 2D silhouettes. In this approach, a axis aligned 3D-object is projected on the coordinate hyperplanes, in order to generate three monochrome images as depicted in Figure 2.11. Next, author founds the outer contour of each image, approximating it by a polygonal line. Then, a commonly used technique on 2D shape description, the discrete Fourier transform, is used to represent the shape features in the spectral domain. The absolute values of the obtained coefficients are used to form the silhouette-based feature vector.

While perfroming a PCA preprocessing makes the silhouette descriptor pose and scale invariant, author stresses that it is also invariant to rotation due to properties of the discrete Fourier transform. However, it is not possible to determine a single unique contour of the silhouette image for 3D models with holes or disjoint parts, only the longest contour is processed. Therefore, certain parts of 3D-objects might not be described.

Depth Buffer Descriptor

When silhouette images of 3D-objects are created, all the information about shape is contained in contour points.Therefore, in order to capture 3D-shape characteristics Vranić considered other approaches for creating 2D images from 3D-objects [133]. He proposed another feature vector, which is obtained from six depth-buffer images formed using the faces of an appropriate cuboid region. After rendering the six images, the three-dimensional discrete Fourier transform is used to represent the image in the spectral domain instead of spatial domain. Figure 2.12 illustrates the extraction of the depth buffer-based shape descriptor.

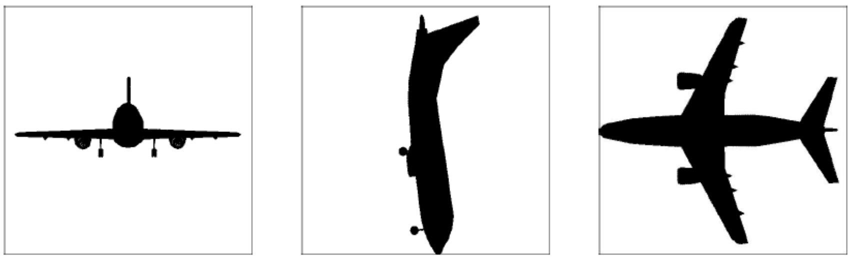


Figure 2.11: Silhouette images of an aeroplane model obtained by projecting the model on the coordinate hyper-planes (Figure taken from [133]).

Elevation Descriptor

Shih *et al.* proposed [112] a descriptor that shares the basic idea behind the depth buffer descriptor referred above. To compute the elevation descriptor, six different views of the 3D object are captured.These views encode elevation maps describing the altitude information of the model relative to the corresponding view plane and are represented as gray-scale images. Then, each one of these images is decomposed into a set of concentric circles around the centre point and the elevation descriptor is obtained by taking the difference between the altitude sums of two successive circles.

Authors are aware that performing a full matching between two models will require a large number of elevation comparisons. Therefore, to reduce matching time Shih *et al.* provide an efficient similarity computation that finds the best match for a given query model.

Light Field Descriptor

Following the idea that if two 3D models are similar, they also look familiar from all viewing points, Chen *et al.* [44] proposed a descriptor based on silhouettes from many different viewing directions. The light field descriptor encode one hundred orthogonal projections of an object, excluding symmetry, with both Zernike moments and Fourier descriptors to produce feature vectors that describe the object.

To improve robustness against invariance a set of ten light field descriptors is applied

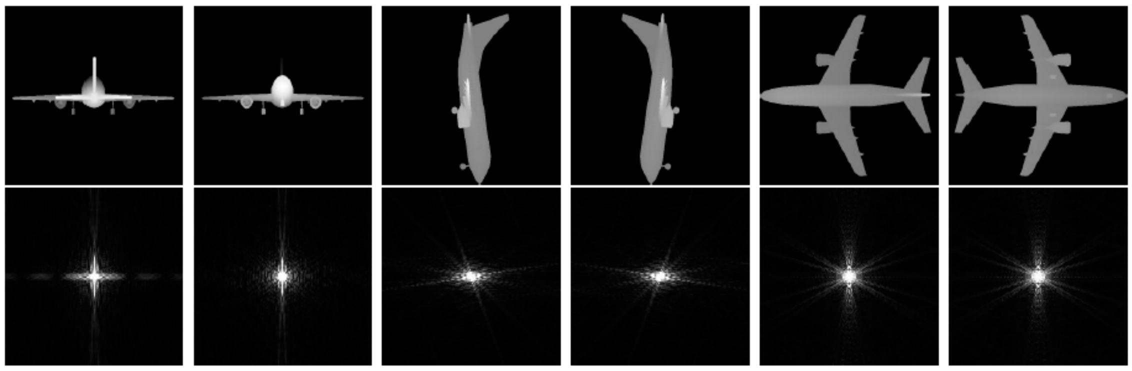


Figure 2.12: Extraction of the depth buffer-based shape descriptor: depth-buffer images in top row and corresponding two-dimensional Fourier transforms in bottom row. (Figure taken from [133]).

to each 3D model, which lead to the one hundred orthogonal projections. These ten descriptors are created from different camera system orientations. Thus, the dissimilarity between two models is the minimum difference between all combinations light fields.

2.2.6 Other Methods

In the previous sections, we described several methods to represent 3D shapes. These were classified according to the technique behind the descriptor computation, may it be histograms, transforms, graphs or images. However there are some approaches that does not fit on these classifications, since the computation of these shape descriptors does not use any of these techniques or combines different methods to achieve a distinct result. In this section we will briefly describe a few of such techniques.

3D Zernike Moments

Novotni and Klein [98] advocate the usage of a specific kind of shape moment that has the advantage of capturing global information about the 3D shape and not requiring closed boundaries as boundary-based methods. The 3D Zernike descriptors are a projection of the function defining the object onto a set of orthonormal functions within the unit ball. They can be considered as the magnitudes of a set of orthogonal complex moments of the 3D shape and the natural extensions of spherical harmonics based descriptors.

Spherical Moments

Based on the concepts underneath moment-based method proposed by Saupe [111], Wei and Yuanjun introduced spherical moments as a shape comparison method for 3D model retrieval [137]. This method employs a multi-level spherical moments analysis approach relying on voxelization and spherical mapping of the 3D models. Authors claim that, despite the simplicity of this method, it outperforms in retrieval performance many previously proposed ones.

To compute the shape descriptor of a model, firstly a pose normalization step is done to align it into a canonical coordinate frame. Afterwards, them model is rasterised into a cubic voxel grid, then a series of homocentric spheres centered at the center of the voxel

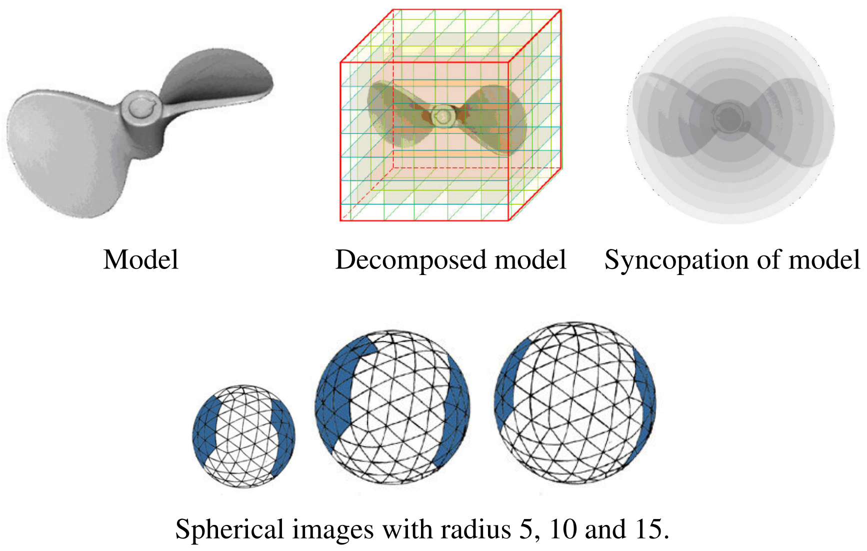


Figure 2.13: Computing spherical images from a 3D model (Figures taken from [137]).

grid are used to produce a series of spherical images, by simply checking the intersection between trigonal pixels on the spheres surface and the object voxels and labeling the pixels accordingly, as illustrated in Figure 2.13. Finally moments of each sphere are computed and the moments belong to all spheres constitute the descriptor of the model. To estimate the similarity between models, Wei and Yuanjun suggest comparing feature vectors using Euclidean distance.

Reflective Symmetry Descriptor

Kazhdan *et al.* proposed describing 3D models by measuring its amount of symmetry [77]. While such approach in two dimensions is quite simple, since it works by averaging an image against itself reflected along a line of symmetry, with 3D shapes the symmetry computation is more complex. Indeed, the reflective symmetry descriptor [79] of a 3D model is a collection of functions that measure the rotational and reflective symmetry with respect to every axis passing through its center of mass. Kazhdan *et al.* present an efficient algorithm for computing the reflective symmetry descriptor from a 3D voxel representation of a model, and show that,in addition, planar symmetries can be used for alignment of 3D meshes.

Moreover, authors suggest using this approach to improve existing shape descriptors with symmetry information. In particular, they describe the symmetry augmented descriptor, based on the spherical harmonic representation, described in Section 2.2.3. According to Kazhdan *et al.*, this augmented descriptor provides a highly discriminating representation of the shape.

Generalized Shape Distributions

Liu *et al.* presented last year a combined approach to 3D shape description [88]. The Generalized Shape Distributions (GSD) takes advantage of both local and global shape signatures. The start by generating spin images, on meshes, producing a set of local shape descriptors, which are then clustered in what authors call ”words” in a ”dictionary” of local shapes. This way, they represent a global 3D shape as the spatial configuration of a set of specific local shapes by computing the distributions of the Euclidean distance of pairs of local shape clusters. Then, they store the descriptor in an indexing data structure to reduce the space complexity of the proposed shape descriptor.

Authors claim that their approach is robust to non-trivial shape occlusions and deformations and is more discriminative than a simple collection of local shape signatures since the spatial layouts of a global shape are explicitly computed. The robustness to shape occlusions and deformations comes from the fact that there are statistically a large number of chances that some local shape signatures and their spatial layouts are unchanged and users can easily identify those unchanged parts. Indeed, their preliminary experiments show the effectiveness of the proposed technique for shape comparison and analysis.

Salient Visual Features

Sharing the ”word” and ”dictionary” concepts Ryutarou Ohbuchi *et al.* proposed a slightly different approach [100] that uses salient visual features for shape-based 3D model retrieval. These features are computed based on 2D range images captured from a set of uniformly sampled viewpoints laid down on a pre-defined view sphere. In each of these range images a set of interest points are identified and then features are extracted from these interest points. The resulting local features are placed in a bag-of-features, which is then clustered to estimate a set of ”visual words” which are then used to compute a histogram of occurrences of each ”visual word” in the model. This histogram is then used to obtain the feature vector that will represent the model.

2.2.7 Comparative Studies

As we have already shown, there are many different techniques to describe three-dimensional shapes, each one with its own strengths and drawbacks. Due to differences between existing approaches to shape description, comparing these is a difficult task. Nevertheless, to assess the effectiveness of shape description methods several comparative studies has been carried out recently. In the following paragraphs, we will briefly describe three of these studies published last year.

Bustos *et al.* presented in 2006 [38] an experimental effectiveness comparison of methods for 3D similarity search. In this study, authors surveyed some approaches to 3D shape retrieval and presented an extensive experimental effectiveness and efficiency evaluation of these techniques, using several 3D collections. Among a total of sixteen shape descriptors, they studied the rotation invariant spherical harmonics descriptor, shape distribution descriptor, shape spectrum descriptor, silhouette descriptor and depth-buffer descriptor.

After comparing the computational complexity of the analyzed descriptors, they discuss its retrieval performance. Authors concluded that there is a number of descriptors that have good database-average effectiveness and work well in general, while others work better with some specific model classes but have poorer results on generic models. Finally, authors argue that most descriptors can be considered robust, as they can effectively retrieve similar objects with different level of detail.

In a distinct study, Alberto del Bimbo and Pietro Pala performed a comparative analysis of a few different solutions for description and retrieval by similarity of 3D models [33]. For this study, authors selected descriptors that are representative of the principal classes of approaches. From the class of histogram-based descriptors, authors selected the curvature histograms and the shape functions, while Spin-image signatures and light field descriptors were used to represent the image-based approaches. Authors also included on the comparative study the geometric moments [51] used by Elad *et al.* to describe 3D shapes.

Bimbo and Pala focused their experimental analysis on comparing the four methods referred above according to their robustness to deformations and their ability to capture the structural complexity of 3D objects, as well as the resolution at which models are considered. To that end, authors used two different shape databases. The *Art-Model* database composed by around three hundred high-resolution models from miscellaneous sources was used to test the robustness to geometric deformations. To that end, these models were subjected to special deformations, generating an extra set of deformed versions of each original model. The other collection used in this comparative study was the well known Princeton shape database.

From the results obtained in this experiment, authors could achieve an extensive set of conclusions. Particularly, they concluded that the light fields descriptor and spin image signatures have superior capability to capture the structural peculiarities of the models, with highest insensitivity noise provided by the light fields representation. Moreover, Bimbo and Pala claim that geometric moments are not able to capture salient and discriminating features of 3D objects and the histogram-based approaches never provide better retrieval performance than the other solutions. However, the computational complexity were not considered in this study, which will eventually uncover additional drawbacks of the image-based descriptors when compared to the histogram-based approaches.

Together with the ESB Jyanti *et al.* presented a comparative study between twelve different shape descriptors evaluating the effectiveness of these representations on the mechanical engineering domain [71]. Within the set of tested descriptors are spherical harmonics, shape distributions, shape histogram and light field descriptors, among others. In these experiment, authors performed an unusual test. They compared the results retrieved by using every shape descriptor against the random retrieval method. As expected, all shape representation methods outperformed the random retrieval.

Additionally, in a paper published in 2007, Bustos *et al.* present two recently proposed approaches to shape description and discuss methods for benchmarking the 3D retrieval systems' qualitative performance [36]. Indeed, they suggest as best options for shape retrieval evaluation the Princeton Shape Benchmark and the actual version of the benchmark used in the 3D shape retrieval contest (SHREC).

2.2.8 Discussion on Shape Descriptors

As we have seen above, there are several three dimensional shape descriptors. In a variety of approaches, these algorithms for feature extraction have one thing in common. All aim at producing a feature vector that provides good discriminating power while keeping the time and space consumption relatively low while computing the descriptor. Although some of the techniques we described in this chapter are outdated, the technique proposed in such approaches remain useful. Indeed, many recent algorithms are basically an evolution of older methods. A summary of all analysed methods for shape description is available in Table 2.1.

To provide a organized view of this research area, we divided the 3D shape description algorithms into five distinct categories. The histogram based approaches simply accounts one or more shape features and constructs histograms with them. These histograms are then used to estimate the corresponding set of feature vectors. The transform-based approaches rely on mathematical transformations to switch from the spatial domain to a more suitable one and compute from there a shape descriptor. For instance by applying the Fourier transform or the spherical harmonics transform. The graph-based approaches computes a graph representing the topology of the model and then uses one or a combination of several histogram-based or transform-based descriptors to code the shape features for each node of the graph. The image-based approaches rely on multiple 2D representations of the shape to compute the descriptor. Finally, there are a few algorithms that do not fit on any of these four categories, which we classify as other.

Depending on the purposes and scope of the retrieval system, some shape descriptors can perform better than other. As a matter of fact, there are no shape descriptor that is clearly better than all the others. Instead, some are best suited for some kind of 3D models or for specific needs of the retrieval system. While, in some cases the major concern is the effectiveness of the shape descriptor, in others the most important factor could be the efficiency of shape description techniques.

The effectiveness of a shape descriptor indicates the amount of shape information it is able to represent. More effective shape descriptors store more information about the shape. On the other hand, the efficiency of a shape description technique regards on the time and space necessary to compute and store the resulting feature vector. Larger and more complex shape descriptors usually led to slower classification and retrieval. In the current implementation of our approach we use the spherical harmonics descriptor, due to its good balance between descriptive power and space-complexity.

	Descriptor	**Feature**	**Refs.**
Histogram-based	Cord and Angle Histograms	Distribution of length and angles of cord rays	[103, 104]
	Color Distribution	Voxel colors computed from shape texture	[103]
	Curvature Histogram	Principal curvatures at each face of the mesh	[81, 131]
	Shape Distribution	Collection of shape functions measuring several shape features using randomly selected surface points.	[101]
	Modified Shape Distribution	Shape functions measuring a pair of features using a quasi-random point selection.	[99]
	Shape Histograms	Surface points in cells of decomposed model.	[9]
	3D Shape Contexts	Distribution of sampled points relative to each other.	[84]
	Extended Gaussian Images	Variation of surface area with surface orientation.	[75]
	3D Hough transform	Variation of surface area with surface orientation.	[143]
	Shape Spectrum	Area of surface object with respect to shape curvature.	[49, 142]
	Density-based Shape Descriptor	Variation of surface area with surface orientation.	[6, 7, 5]
Transform-based	Voxel 3D Fourier Transform	Proportion of total surface area inside each cell of voxelised model.	[135]
	Distance and radial cosine transform	Binary and continuous voxel-based distance from a point to the object surface.	[50]
	Spherical Harmonics Transform	Distance from the object surface to the surface of enclosing sphere.	[111]
	Rotation Invariant Spherical Harmonics	Spherical functions of concentric sphere based on voxelised model.	[78]
	Planar-Reflective Symmetry transform	Symmetry of a shape with respect to all planes through its bounding volume.	[105]
Graph-based	Multi-resolution Reeb Graphs	Reeb graphs at multiple levels of resolution of a function over the surface	[65]
	Size Graphs	A centreline skeleton with nodes labelled with local geometric properties.	[29]
	Skeletal Graphs	A directed acyclic graph associated with a set of geometric features and a signature vector.	[121]
Image-based	Spin Images	Surface points projected on planes defined by oriented points on model surface	[73, 14]
	Silhouette Descriptor	Object projections on coordinate hyperplanes.	[133]
	Depth Buffer	Mappping of surface distances into the six faces of the object bounding cube.	[133]
	Lightfield Descriptor	Object silhouettes captured by cameras on the vertices of a dodecahedron.	[44]
	Elevation Descriptor	Decomposition in concentric circles of surface elevation with respect to the six faces of bounding cube.	[112]
Other	3D Zernike Moments	Magnitudes of a set of orthogonal complex moments of the object.	[98]
	Spherical Moments	Moments of spheres mapping voxelised models.	[137]
	Reflective Symmetry Descriptor	Collection of functions measuring rotational and reflective symmetry with respect to every axes passing on barycenter.	[77]
	Generalised Shape Distributions	Clustering of spin images.	[88]
	Salient Visual Features	Clustering of range images.	[100]

Table 2.1: Summary of 3D shape descriptors.

2.3 Query and Matching

As we have already mentioned, 3D shape descriptor computation is crucial on a shape based retrieval system. This explains the large amount of work developed in this particular topic, which we surveyed in the previous section. However, effective shape representation is not the only challenge to overcome when developing 3D model retrieval systems. Another important part of a 3D shape retrieval solution, as of any content-based retrieval system, is the query and matching processes. Descriptors extracted from objects are usually represented as feature vectors on a multidimensional space. This is truth not only for 3D models, but also for other data types, such as images or music. Effective content-based retrieval of information indexed on multidimensional spaces depends greatly of query and matching techniques.

2.3.1 Query Types

Regardless of the data stored on the database or used as a query, several authors agree on a basic set of different types of queries [58, 64, 41] for multidimensional spatial data. Following the enumeration of query types presented by these authors, we will briefly describe the four most common categories of queries.

Exact match query This type of query aims on finding all objects that have exactly the same spatial extent as the spatial query object. Indeed, exact match queries are only of moderate interest in content-based retrieval , and, when applied on content-based retrieval, are usually based on metadata, managed by a traditional database management system. An example of such query is ”find all alloy 17 inches-radius wheels”.

Range search query Content-based retrieval approaches rely mostly on retrieval-by-similarity queries. One way to accomplish this is by performing a range search query, *i.e.* find all objects that are within a given range, usually a hyperrectangle [9]. Such query can be specified as finding objects in the multidimensional space that have at least one

[9]A hyperrectangle, also called orthotope, is parallelotope whose edges are all mutually perpendicular. Indeed, a hyperrectangle is a generalization of the rectangle to higher dimensions [138]. For instance, the cuboid is a 3-orthotope.

common point with a query volume in that space. For instance, the following sentence is a range search query: ”find all MRI models showing a tumor of size between x and y”.

k-nearest-neighbor search Another way to perform retrieval-by-similarity is to search for a given number of objects similar to the query. Measuring this similarity among objects is an important issue on retrieval that we discuss briefly on Section 2.3.2. Usually such similarity is seen as a distance between object representations in multidimensional space. Thus, a k-nearest-neighbor query is specified by finding k objects with the shorter distance to the given query. An example of such query is given by the sentence: ”Find the twenty tumors most similar to a specified example”.

Within-distance (or α-cut) This third way to query by content is quite similar to the previous one. The main difference is that in this type of query instead of a pre-defined number of results, the search must return all the results within a given distance to the query. Thus, the objective is to find all objects with a similarity score better than α with respect to a query. This means to find all objects whose representation in multidimensional space has a distance smaller than a threshold from the query representation in that space. An α-cut query can be specified, for instance, by the sentence:”Find all the MRI containing tumours having a given similarity with respect to the example provided”.

2.3.2 Similarity Measuring

To achieve effective matching, resemblance between objects must be measured. Usually, similarity measuring among objects is done through distance functions that estimate the distance between the closest points of their representations in the multidimensional space.

As a matter of fact, an extensive theory lays behind distance functions and there are a large set of possible approaches to measure the distance between two points in a multidimensional space. However, this issue is not within the main scope of our work. Therefore, we will only focus on a short list of the most useful distance functions. Basicaly, we will present some particular cases of the family of Minkowsky distances and the Mahalanobis distance.

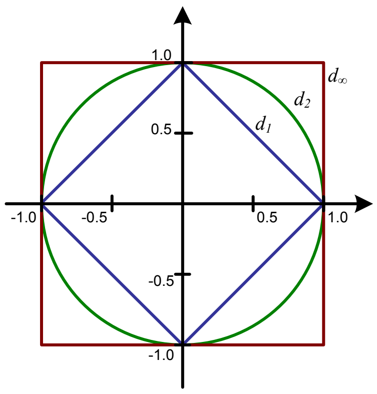


Figure 2.14: The unit sphere under Manhattan (d_1), Euclidean (d_2) and Chebychev (d_∞) distances.

Minkowsky distances

The Minkowsky distance of degree p, also called p-distance, between two points in a n-dimensional space is given by:

$$d_p = (\sum_{i=0}^{n}(x_i - y_i)^p)^{\frac{1}{p}}. \tag{2.1}$$

Indeed, this general equation is not applied in practice. Instead, the parameter p is fixed in a few values commonly used. Thus, the Minkowsky distance of degree 1 ($p = 1$) is called Manhattan distance, the usual Euclidean distance is the distance of degree 2 ($p = 2$) and with $p = \infty$ we obtain the Chebychev distance. The difference among these three distance functions is shown in Figure 2.14, where we depict the unit sphere in each one of these metric spaces. The corresponding distance functions are defined as:

- **Manhattan distance:** $d_1(x, y) = \sum_{i=0}^{n}(x_i - y_i)$;
- **Euclidean distance:** $d_2(x, y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$;
- **Chebychev distance:** $d_\infty(x, y) = max_{i=0..n}(x_i - y_i)$.

These three Minkowsky distances are simple, fast to compute and can be generically used. However, in some cases the results obtained by this measurements do not fulfil the needs of retrieval solutions. Thus, to solve problems caused by poorly scaled or highly correlated coefficients of a vector, is often used the Mahalanobis [91] distance.

Mahalanobis distance

The Mahalanobis distance is a computationally expensive generalisation of the Euclidean distance widely used in cluster analysis and other classification techniques to measure the distance between probability distributions. This measurement is based on correlations between variables by which different patterns can be identified and analysed.

Mahalanobis distance is a useful way of determining similarity of an unknown sample set to a known one. It differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant. The Mahalanobis distance is defined in terms of a covariance matrix C, which measures a tendency to vary between two features, as given by:

$$d_M(x, y) = det|C|^{\frac{1}{d}}(x - y)^T C^{-1}(x - y). \tag{2.2}$$

There are some other distance functions that are often used in content based retrieval solutions, but we prefer not to mention them all here. Instead, we refer our readers to the comprehensive explanation of similarity measures for retrieval published by Castelli [41] or to a theoretical description of distance function presented by Hervé Abdi [1].

In our approach we will use the quadratic Euclidean distance to measure the distance between the shape signatures. Although more expensive in computational terms than the Manhattan distance, the quadratic Euclidean distance function is faster to compute than the Euclidean distance and provides the precision necessary for our research purposes.

2.4 Content-based Retrieval of 3D Models

Having introduced in the previous sections some of the existing works regarding description of three-dimensional shapes, generic query methodologies and commonly used similarity determination techniques, next we will present the more relevant solutions for content-based retrieval of 3D models.

2.4.1 Nefertiti

During recent years, several 3D shape search engines have been introduced. One of the earliest of such systems was proposed by Paquet and Rioux in 1997. Nefertiti [103] is the first well documented query by content software for three-dimensional model databases. It incorporates a set of retrieval algorithms that allows database searches by scale, shape, color or any combination of these parameters.

2.4.2 Princeton 3D Model Search Engine

Later, in 2001, Thomas Funkhouser and his team released the Princeton 3D model search engine [56]. This system is now the best known solution for shape retrieval, indexing more that thirty six thousand models. Its authors claim that they have developed the search engine to be the ”Google$^{\text{TM}}$ for 3D models” [55]. However, despite its success within the research community, backed up by a powerfull shape description and query mechanism, the usability of this search engine failed to fullfill the authors expectations.

2.4.3 Purdue 3D Engineering Shape Search

Unlike the Princeton team, whose search engines aims on generic 3D models, the PRECISE group at Purdue University developed a search engine for a specific domain [90]. The 3D Engineering Shape Search system integrates a set of existing shape description techniques to compute the feature vectors of a model. This search engine incorporates a 3D interface that allows users to submit a shape as a query, to select the feature vectors that will be used for shape representation and to search the database by browsing.

2.4.4 NTU 3D Model Retrieval System

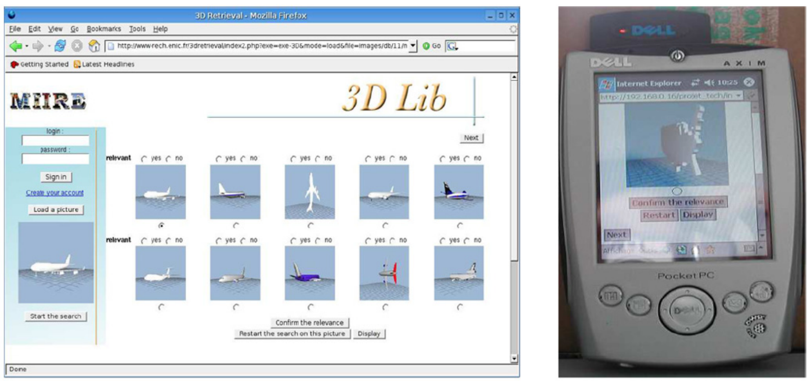
Starting from the idea that if two 3D models are similar they also look similar from all viewing angles, Chen *et al.* introduced a retrieval system [44] based on the light field descriptor. The 3D Model Retrieval System from National Taiwan University is available on the web and its database contains more than ten thousand publicly available 3D generic models.

2.4.5 CCCC

To serve as a proof-of-concept to methods and tools for content-based search for 3D-mesh models proposed during his PhD research [133], Vranić deployed a web-based retrieval system for 3D models. The Content-based Classification of 3D-models by Capturing spatial Characteristics (CCCC) 3D search engine uses a set of model databases, including the Princeton Shape Benchmark test and training databases, providing around three thousand classified objects.

2.4.6 FOX-MIIRE Search Engine

More recently, in 2007, researchers from the FOX-MIIRE group released a on-line search engine for 3D content [11]. Their search engine implements the adaptive views clustering technique, a method proposed by the authors to index 3D models based on two-dimensional views. Besides the good retrieval results offered by the FOX-MIIRE search engine, it has a unique feature when comparing with previous approaches. This search engine is the first that accepts 3D-Models retrieval from photos [10] and can be reached through a mobile device. Figure 2.15 depicts both the standard and the mobile device interfaces of the FOX-MIIRE search engine. Indeed, the idea of incorporating a 3D model retrieval system in a mobile device was proposed by Suzuki *et al.* [124]. They developed an experimental 3D shape retrieval system for cellular phones where users can search for a model similar to a given example.



(a) (b)

Figure 2.15: MIIRE search engine on PC (a) and PDA (b).

2.5 Retrieval using Partial Queries

Knowing that the list of works on shape retrieval presented above is not exhaustive, one can state that there is plentiful work on 3D shape retrieval. However, most of the methods for 3D object comparison discussed in the literature approach the problem of shape similarity as a global matching problem. These methods estimate the similarity between two objects by returning as output a real number obtained by analysing the overall shape of the two objects instead of considering similar sub-parts shared by the two objects [37].

In the last decade several approaches to partial matching have been proposed, but none of them provide a definitive solution to this problem. The methodologies for the estimation of partial matching can be grouped into two coarse categories: based on local shape descriptors and based on structural descriptors.

2.5.1 Spin Images

In the first category, one of the most important method that inspired many other approaches, uses the spin-images to provide a set of local shape descriptors [74]. This method samples the object surface into a set of oriented points (3D points with surface normals) and associates to each sampled point a 2D description of the surface around it: the spin-image. A similarity measure between 2D images is used to evaluate the similarity between two spin-images and thus between two oriented points of the compared objects. In this way a point-to-point correspondence between the two objects is provided. In [110] the similarity measure defined among spin-images is used to group oriented points into patches. This latter approach allows the correspondence between patches instead of points.

2.5.2 Salient Geometric Features

A more recent approach [59] performs partial matching by comparing the salient features of two objects. In their approach to partial matching, Ran Gal and Daniel Cohen-Or [59] shown that a relatively small number of salient geometric features can describe a three-dimensional model with sufficient detail for various applications of content-based shape retrieval. Based on this idea they introduced the abstraction of salient geometric features and presented a method to extract these features from polygonal meshes.

The first step of this method is computing a sparse set of local surface descriptors across the surface and use these to measure similarity between regions of the model, even if they have dissimilar polygonal meshes. Then, these descriptors are clustered in order to locally describe a nontrivial region of the surface. Each one of these clusters form a compound higher-level descriptor that represent a salient geometric feature characterising a local partial shape. In this approach trivial regions of the model are considered irrelevant and discarded.

A major challenge facing the Gal and Cohen-Or was correctly identifying the salient features. To that end, they start by making a loose definition of salient geometric feature. In this definition, a salient geometric feature is a region of the object surface with a non-trivial shape. Based on this definition, they select regions with high curvature relative to their surroundings and high variance of curvature values as geometrically salient. Indeed, such option is grounded on previous work by Hoffman and Singh [66]. They have found that human vision defines boundaries along negative minima of the principal curvatures on surfaces. From this, Hoffman and Singh suggest that salience of a region depends on its size relative to the whole object, the degree to which it protrudes, and the strength of its boundaries.

Authors identify salient regions by growing, for each descriptor from the sparse set, a cluster of descriptors. Such cluster is constructed by incrementally adding descriptors from its neighbourhood that maximise the saliency of the cluster until the contribution of neighbour cluster become insignificant.

After estimating all clusters, authors select from these a set of clusters with higher values of saliency grade and use them to identify the set of salient geometric features of the model. This set should include model regions that are salient and interesting compared with other parts of the model. Figure 2.16 illustrates the result of applying this method to four different models and selecting as salient the top ten percent clusters ordered according to saliency grade.

In this approach each model is represented by a set of descriptor clusters corresponding to the salient geometric features of the object. Ran Gal and Cohen-Or associate each

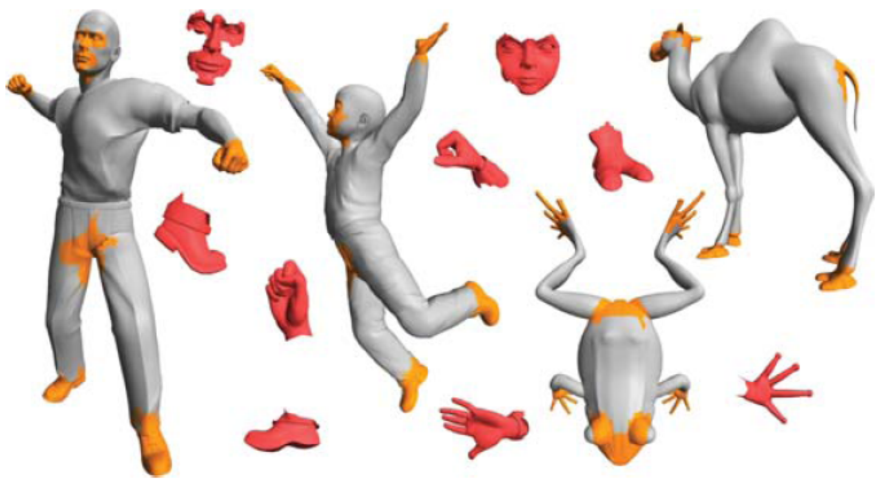


Figure 2.16: Salient geometric features from four models and corresponding individual sub-parts (Figure taken from [59] © 2006 ACM).

one of these features with a vector index (a signature) and insert it in a geometric hash table[10]. Authors recognise that elaborate indices, such as normalised moments can be used to describe the geometric features. However, they simply use the terms employed for defining the saliency grade to construct the vector index, reinforcing their claims for the efficiency of salient features in shape retrieval.

2.5.3 Distinctive Regions

Also in [114] important regions of the surface object are used to perform partial matching. However, researchers at the Princeton 3D shape retrieval group followed a slightly different path. Instead of identifying the salient regions of an object, Shilane and Funkhouser [113] suggest selecting the distinctive regions of a 3D surface. The basic idea behind their approach is to focus the shape matching process on local features of shapes that are consistent among objects of the same class and distinctive relative to object of other classes.

Instead of using global descriptors, which represent global features of the model and

[10]Geometric hashing is an highly efficient technique with low polynomial complexity developed for matching geometric features against a database of such features [85]. This technique uses a grid-based hash table to store every feature of every object but only a limited number of features is used to determine a mapping into the hash. During a query, the remaining features are used when hash collisions exist. With this technique matching is possible even when the recognisable database objects have undergone transformations or when only partial information is present [139].

fail when local properties of an object distinguishes it from others, in their approach authors use local shape descriptors. However, computing and storing local shape descriptors for the whole shape is time consuming and space expensive. To overcome this, they proposed a method for finding distinctive features of an object that are more relevant for shape retrieval.

In their method, Shilane and Funkhouser [114] define a distinctive region as a region with features that are only found on objects of a single class, while a not distinctive region is a region common to many objects of different classes. Therefore, in this approach to find the distinctive regions of an object the complete model database should be initially classified into object types. Otherwise it will not be possible to establish which are the objects of the same class. And such relationship is necessary to identify common features.

The distinctive region identification process starts by randomly sample each mesh on the database in order to obtain a set of spherical regions, covering the object at different scales. For every region, authors compute the corresponding shape descriptor that represents the distribution of surface area within that region. Next, by comparing all the descriptors of the database, they produce a ranked list of matches for each descriptor and use it to produce measures of region distinctiveness, thus identifying the most distinctive regions of each model.

Identifying distinctive regions is, therefore, a pipeline of relatively simple steps. Although other sampling methods could be used, authors propose selecting points randomly with uniform distribution with respect to surface area. Likewise, several shape descriptors can be used, but authors suggest describing the shape of every spherical region using rotation invariant spherical harmonics[11] [78]. Figure 2.17 illustrates the different stages of the process of partitioning a model into distinctive regions with respect to a set of object classes in a given database. In the final result, regions in red are the most distinctive while regions in blue are least distinctive.

To perform partial matching retrieval on large model databases, Funkhouser and Shilane proposed a priority-driven search algorithm [57]. This kind of backtracking search algorithm considers only partial matches that can possibly lead to the lowest cost matching, as in the widely known shortest path algorithm by Dijkstra [48]. Therefore, authors

[11]Rotation invariant spherical harmonics were briefly described in Section 2.2.3.

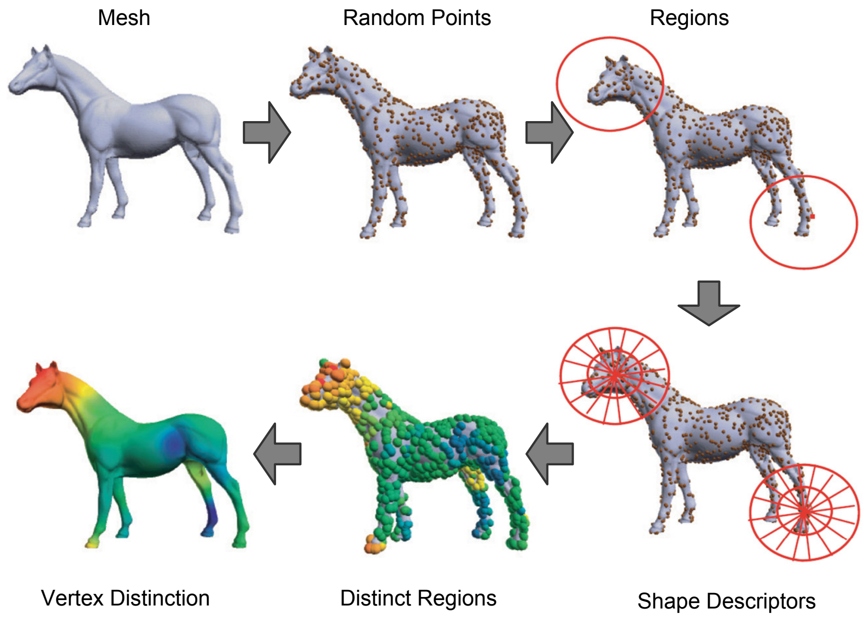


Figure 2.17: Selecting distinctive regions of an object (Figures taken from [114] © 2007 ACM).

use a cost function that accounts for both feature dissimilarity and geometric deformation to order the list of pairwise matches between features of query and of objects in database. The proposed algorithm produces a list of best target objects sorted by the similarity of a subset of matching features between the object and the query.

2.5.4 Structural Descriptors

The previous methods describe 3D objects as a set of local shape descriptors, on the contrary structural descriptors describes 3D objects as a graph-like skeleton representing the relevant part of the object and their adjacency relationships. While local shape descriptors drop out the information on the overall shape, the structural descriptor provide at the same time global and partial information on the shape of the object. Beside the identification of shared similar sub-parts, between two objects, and their correspondence, the information associated to the structural descriptor makes easier the estimation of the global similarity based on the overall shape of the objects. The following are some example of partial matching methods based on structural descriptors.

The methods proposed in [26] represent the shape object as binary tree obtained by recursively subdividing the object into two parts. The recursive subdivision of the objects is obtained by analyzing the geodetic distance among the vertexes of the triangular mesh representing the object and the angle among triangles. The similarity measure between two objects is obtained by matching the two trees, Beside the sub-part correspondence is induced by the node mapping provided by the matching algorithm.

The structural-based framework for 3D shape matching proposed in [45] uses a many-to-many matching algorithm that works with skeletal representations of 3D volumetric objects. The matching between two 3D skeletons is obtained by using an extension of the Earth Mover's Distance (EMD) where skeleton transformations are considered.

The approach proposed in [31] is based on a flexible matching framework based on the consolidated Reeb graph theory. Biasotti *et al.* described an interesting method for partial shape matching that couples geometry and structure in a single descriptor. Based on the theory of Reeb graphs, as an alternative to commonly used skeletal graphs, authors compute the so-called structural descriptor. They suggest [93] encoding the shape and all its relevant sub-parts in a graph which represents the structure of the object and its geometry at the same time.

The proposed extended Reeb graph (ERG) [120] generalises the original Reeb graph definition to a surface on which a finite set of contour levels given by a mapping function f is defined. In their work, authors compare two distinct mapping functions, since choosing this function is an important aspect of the proposed method. One option is using the distance from the centre of mass of the object as a mapping function, which makes f rotation invariant, but sensitive to pose changes. The other option is estimating f as suggested by Hilaga *et al.* in [65], using the integral geodesic distance to the surface centre, which is also pose invariant. Biasotti *et al.* conclude that the latter is best suited for retrieving articulated objects disregarding its pose, while the first option distinguishes articulated models in different poses.

Using the selected mapping function, the ERG is constructed and represents the topology of the model. Then, the corresponding value of f and a geometric descriptor is assigned to each node of the graph, which represents a sub-part of the model. To compute the geometric descriptor assigned to each node, authors use spherical harmonic analysis of the corresponding sub-part. The rotation invariant spherical descriptor used in this ap-

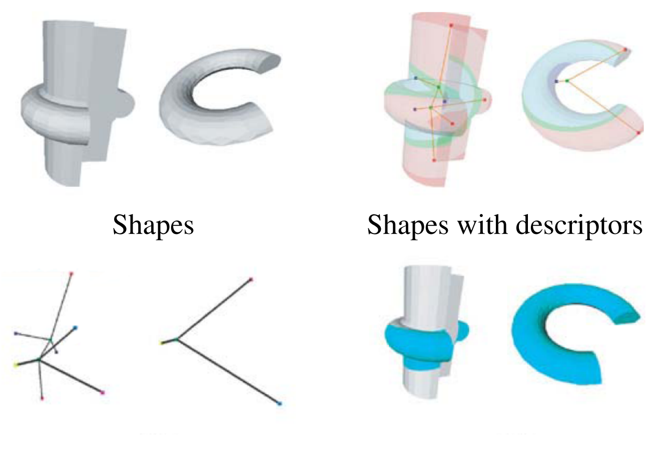


Figure 2.18: Sub-part correspondence of two mechanical parts (Figures taken from [31] © 2006 Elsevier Ltd.).

proach has been defined by Kazhdan *et al.* in [78] and is briefly described in Section 2.2.3. Additionally, each sub-part is uniformly scaled separately before computing the descriptor to guarantee that retrieval is scale invariant. Indeed, due to the necessity of finding similar sub-parts with different sizes, scale invariance is an important feature in retrieval with partial matching approaches.

Since the structural descriptor is coded as a directed attributed graph, the sub-part correspondence between models is obtained by matching its descriptors, *i.e.* matching its graphs. Using inexact graph matching, the authors adapted the algorithm proposed by Marini [94] for the computation of the maximum common sub-graph between two directed, acyclic graphs with attributes. The specialised version of this algorithm produces a set of all common sub-graphs between two extended Reeb graphs, considering not only the topological structure but also node attributes such as the geometric descriptor. The similarity estimation between models is obtained by considering the size of the common sub-graphs with respect to the size of the corresponding graphs and the similarity distance between the nodes belonging to the common sub-graphs.

An example of the above described technique is shown in Figure 2.18. To obtain partial matching between two models the ERG are extracted from each object and the structural descriptor are computed based on it. Then, a graph matching technique is applied to compare the structural descriptors, identifying the common sub-graphs. Finally, the similar subparts are identified in both objects by comparing the common sub-graphs.

2.5.5 Scale-space Feature Extraction

Focusing on mechanical CAD models, Bespalov *et al.* [25] proposed a partial matching technique for finding similarities across part models constructed from data acquired in 3D scanners. For that end they propose a feature extraction technique based on recursive decomposition of polyhedral surfaces into patches which applies the method introduced by Novatnack *et al.* for extracting and integrating shape features in the discrete scale-space [12] of a 3D mesh model [97]. The discrete scale-space of a three dimensional model is constructed by unwrapping the shape surface onto a planar domain, as a two dimensional image of surface normals. After this initial step, the scale-space operator used in image processing can be applied to the 3D shape.

However, the parametrization of original mesh to the planar domain that produces the surface unwrapping is not isometric, introducing distortion in the image. As a result of this distortion, relative geodesic distances between points on the original 3D model are not equivalent to relative distances between corresponding points on the 2D normal map. Therefore, to correct this distortion, authors compute the distortion for each point in the 2D image and then construct a dense distortion map with these values. Then, this map is used to approximate the geodesic distances between two points in the two dimensional image representing the unwrapped model surface. Finally, the discrete scale-space of the original model is constructed from finer to coarse by iteratively convolving the normal map with a distortion adapted Gaussian kernels, as commonly done when computing the scale-space of a two dimensional image.

After the discrete scale-space of the model has been constructed, scale-dependent shape features can be extracted in a similar manner to image feature detection. To that end, a gradient of the normal map that correctly accounts for the distortion is defined. This gradient is then used to detect edges and corner of the original shape in the normal map. Since a 3D corner is a point with geometric changes in more than one direction, these points can be detected in the normal map by identifying large local changes in the

[12]Scale-space is widely used by the computer vision and image processing communities for handling image structures at different scales. With this framework, the fine-scale features are iteratively suppressed while the level in the scale-space representation increases. The idea behind this theory is that objects are composed by different structures at different scales. For instance, it is appropriate to represent a dog at the scale of meters, but not the hair of its fur or the molecules that compose its skin, which should be represented at much finer scales. Therefore, the scale-space approach consider multiple descriptions for an object at different scales to be able to capture its complete description.

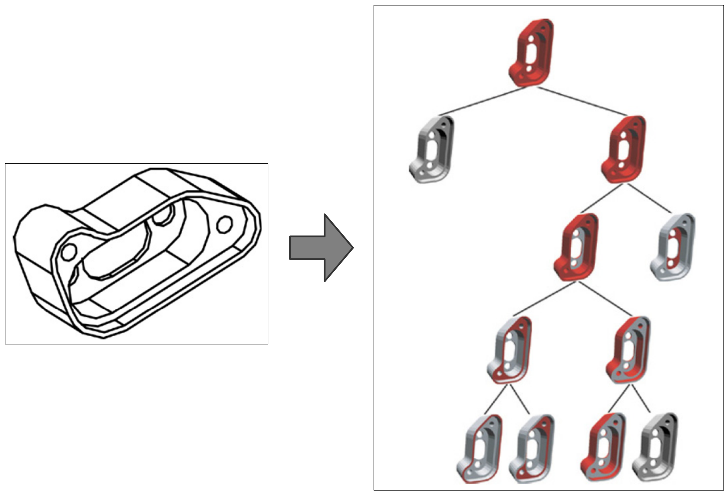


Figure 2.19: Scale-space decomposition of a mechanical part (Figures taken from [25] © 2006 Elsevier Ltd.).

normal directions. On the other hand, an edge in the 3D model corresponds to a line of points with significant changes in the surface geometry. Therefore, edges are detected by finding maxima along gradients previously computed. Indeed, to detect corners and edges authors suggest methodologies analogous to the Harris corner detection algorithm [63] and the Canny edge detector algorithm [40] respectively.

Once the features have been extracted at individual scales these are combined into a unified feature set which encodes the scale-dependent geometric structure of the shape, providing a concise representation of the original model. Authors argue that, with the appropriate parameters, the method can be tuned to extract local features of engineering relevance from CAD mechanical models. Thus, they adapted feature extraction in scale-space proposed by Novatnack [97] discussed above by replacing the geodesic distance function by a new distance function computed with respect to triangular faces of the model. This function measures the maximum angle between adjacent faces on the shortest path between two surface polygons.

In practice, the maximum angle function introduced by Bespalov *et al.* quantifies the smoothness of the surface, since smaller angles correspond to smoother surfaces. Using this function, CAD mechanical models are decomposed and the resulting combined feature set is used for partial matching of 3D models. Figure 2.19 illustrates a scale-space decomposition of a CAD model. In this example the presented tree are not full, since it will be hard to understand the results if the whole tree was depicted.

2.5.6 Part-in-Whole Matching

Suzuki *et al.* proposed [125, 123] a solution that follows a different approach. They aim for part-in-whole matching instead of partial matching. To that end, the 3D model is initially decomposed into its sub-components and then shape descriptors for these shapes are computed using a rotation invariant shape descriptors they proposed earlier for their similarity retrieval system [122].

There are a multiplicity of different ways to decompose a 3D object. Indeed, besides the impractical user-assisted 3D model decomposition, several automatic techniques have been proposed. These usually rely on object attributes such as color, texture or shape curvature. Detailed explanations of these techniques can be found in several papers [43, 42, 145]. In their work, Suzuki *et al.* apply a simple and automatic decomposition technique. They decompose 3D models into several parts by comparing angles created by normal vectors of each polygonal face, and the technique finds sharp angles and cuts polygonal faces into parts based on a typical clustering approach. To tune the decomposition granularity is used a threshold for the angle size. A wide angle size produces a large number of shape parts while a sharp angle size produces a small number of components.

To compute the shape descriptors for extracted components, Suzuki *et al.* used a rotation invariant shape descriptors they proposed earlier for their similarity retrieval system [122]. In this method the object part is initially normalised for scale and then for orientation by using principal component analysis pose normalisation. Next it is voxelised and inserted into a cube divided in a three dimensional grid. The number of voxels contained in each cell are computed and then a clustering technique is applied. Finally, the descriptor are constructed from a voxel distribution function.

Although their decomposition technique is fully automatic, authors acknowledge that occasionally the algorithm can not efficiently handle highly complex 3D models. Additionally, time complexity is also a problem of the proposed method, since the decomposition process is a time consuming task and shape matching requires a considerable amount of time due to the high number of shape descriptors for each model.

More recently, Suzuki *et al.* improved their decomposition method and partial shape descriptors construction algorithm to attain better similarity retrieval results [126]. One of the decomposition enhancements was the use of the area proportion to identify irrelevant parts that should be merged into other. Other improvement in this approach was the use of multiple bounding boxes in descriptor computation. Authors use a bounding box for each decomposed part, instead of only one for the entire object used in their previous solution. However, despite for most models this approach proved better, the time complexity problems were not solved and when 3D models does not have visually irrelevant parts the previous technique works better.

In our approach to 3D shape retrieval, described in this dissertation, we propose a novel methodology that, while also relying on shape decomposition and part-in-whole matching, overcome the retrieval time complexity. The technique proposed by Suzuki uses a *per object* decomposition that looks at surface angles and areas of each model to determine its decomposition. Our approach decomposes each model in terms of features considered distinctive with respect to other models in the collection. Moreover, to retrieval a model Suzuki approach searches the similar sub-parts in a dataset containing all detected segments in every model of the collection, which makes it hardly scalable for large collections. In our approach we use a thesaurus of shapes together with an inverted index, which greatly reduces the number of comparisons required in a query, thus providing scalability.

2.6 Summary

In this chapter we familiarized the reader in the research subjects covered by this dissertation. Namely, we listed the key players in shape analysis, classification and retrieval in order to provide an overview of those who have been providing the latest advances in this area. Obviously, such list is far from complete, since it will be hard to produce an exhaustive list of all research groups involved in a constantly evolving field. We include in this list only those whose work we consider more relevant regarding our research. Next we presented some existing 3D model collections and described in detail the best known shape benchmarks, with special focus on the ESB which we used intensely in our experiments.

Due to core importance of shape description techniques for our research, we reviewed with special attention the existing approaches to 3D shape description. From this study we identified the technique that best fits our purposes. The rotation invariant spherical harmonics descriptor [78] was selected, despite the time-complexity of the estimation algorithm, because of its descriptive power. However, this shape description technique has another drawback just identified as such during the evaluation of the algorithm. The Rotation Invariant Spherical Harmonics (SHA) descriptor produces an extremely long signature, when compared with other approaches. As discussed in Chapter 6, the large size of the signature, while providing a powerful description of the shape, increases the time and space complexity of the whole approach, threatening its scalability.

To complement the background required to fully understand the matters discussed in this dissertation we also presented basic concepts for query types and similarity measurement techniques. Here we described the exact match and range search queries, as well as the k-NN search and α-cut distance. Indeed, these concepts are fundamental to comprehend any retrieval solution. The same applies to the similarity measurement techniques. Although we used in all our work the quadratic Euclidean distance, we described here other commonly used approaches to measure distances between two points in space.

Finally, we present existing solutions that allow content-based retrieval of 3D models. However, most of these systems support only complete matching, *i.e.* queries by the entire object. Indeed, the subject of partial matching in 3D shapes is relatively recent and most relevant work has been published in the last six years. Thus, we finished this chapter by presenting the relevant work on partial matching, starting by the solution proposed by Correa *et al.* [110] through the latest advances in this topic. However, despite the several approaches to partial matching, no definitive solution for this issue had been proposed. In our research we devised a novel approach that overcomes some problems faced by existing approaches, which we will describe in the remaining of this document.

3 Collection-Aware Segmentation

As we have referred previously, our approach to shape retrieval relies on a correct and meaningful decomposition of the models in the collection. Since our shape thesaurus is composed by a set of terms, and these terms are computed from sub-parts of models in the collection, segmentation plays an important role in our methodology to index and retrieve three-dimensional shapes with partial queries.

To segment models in the collection, we need a technique that will provide not only automatic segmentation of all models, but will also produce useful segments for the thesaurus construction. Several approaches to three-dimensional shape decomposition have been published in the past, with recognized success in some domain-specific models, such as articulated characters. However, independently of the methodology used, all approaches only consider the model to be decomposed, ignoring the context where it lies, namely the other models in the collection.

Although existing decomposition techniques could be very effective in many cases, they do not completely fulfill our needs to decompose models in a collection in order to classify them using a shape thesaurus, because some need per object parameterization, do not produce multilevel decomposition, or even require human intervention during segmentation. Therefore, in the present work we devised a novel approach to shape decomposition: the Collection-aware Segmentation (CAS). This method takes into account other objects in the collection while decomposing each model. The CAS decomposition is fully automatic and produces a multi-level segmentation of all models in the collection.

3.1 Algorithm Overview

The CAS is a decomposition algorithm that performs multilevel shape segmentation of each model based on the concept of decomposable regions. Decomposable regions are determined according to their distinctiveness regarding regions of all other models in the collection. In our algorithm, distinctive regions of models are further decomposed in our multi-level segmentation, while common regions are not. The key idea behind this heuristic is that if a feature is shared by many objects it will correspond to a term in the thesaurus as well as unusual features, and the last should be further decomposed into more common subparts in order to become itself a set of common regions. Thus, it is important to clearly comprehend what is a distinctive region in a model.

The concept of distinctive region in 3D meshes was first introduced by Philip Shilane and Thomas Funkhouser in [114]. In their approach, the distinctive regions of each object are identified, by comparing objects in a collection and by selecting those regions that are consistent with objects of the same type and different from regions in objects of other type. Although they achieve interesting results, their approach requires a pre-classified collection, where objects are organized into categories.

Therefore, the method proposed by Shilane and Funkhouser to identify distinctive regions is not appropriate for our purposes. We need a technique that works on unclassified collections, identifying automatically distinctive regions with the intent of segmenting the model. To that end, we consider distinctive a region, or segment, of a model whose geometric features do not occur frequently in the collection. In our approach, less distinctive segments are geometrically similar to many others in the collection, while a more distinctive segment share geometrical similarities with few segments in the whole collection.

After introducing our interpretation of distinctive segments, we can present with some detail our overall approach to model decomposition. A conceptual overview of this methodology, depicted in Figure 3.1, can be given as follows: each model in the collection is decomposed into subparts; then, shape descriptors for each subpart are computed and used to determine the subpart distinctiveness; next, based on this information, the algorithm identifies which subparts of each model should be further decomposed and the iteration starts over, considering now the recently decomposed subparts.

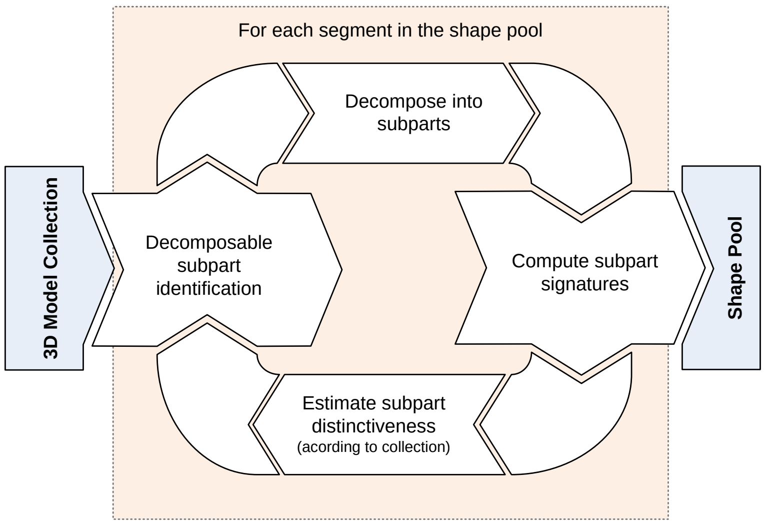

For each segment in the shape pool
Decompose into subparts
Decomposable subpart identification
Compute subpart signatures
Estimate subpart distinctiveness (acording to collection)
3D Model Collection
Shape Pool

Figure 3.1: Collection-Aware Segmentation pipeline

From the description, above it should be clear that the proposed approach is supposed to work with generic collections of 3D models, and also support different shape segmentation and description techniques. However, in the context of this thesis and to validate our framework, focused on a specific decomposition technique that facilitates the creation of a thesaurus for 3D shape retrieval and selected a well-defined setup:

- **Collection type:** CAD models;
- **Segmentation algorithm:** Hierarchical fitting primitives [16];
- **Shape description:** Rotation invariant spherical harmonics [78].

For our experiments we decided to restrict the collection type to engineering CAD models instead of generic 3D models. Naturally, the choice of a benchmark collection for our studies will reflect this restriction. To that end, we will mainly use the Purdue's Engineering Shape Benchmark (ESB) collection, introduced by Jayanti *et al.* [71] and described in detail in Section 2.1.3. Additionally we will also use a dataset of LEGO® models extracted from the National Design Repository [107, 127, 108] and the collection of watertight models used in the SHREC 2008 stability track [28].

By selecting CAD models we reduced the problem of shape segmentation to a specific domain and consequently we make the selection of existing segmentation methods easier. Indeed, our collection-aware shape decomposition technique is based on a traditional segmentation algorithm. We decided to use the Hierarchical Fitting Primitives (HFP) segmentation algorithm for multilevel decomposition of models, since it has proven good results in decomposition of engineering models and produces results that serve well our needs.

To create the feature vectors we use the SHA, a widely accepted rotation invariant shape descriptor published by the Princeton team [78]. This descriptor overcomes several limitations of many other descriptors at the cost of time complexity. Indeed, the SHA descriptor computation is very time consuming, but, since the classification process will be executed in batch and during the retrieval phase only the descriptor for the query will be computed, this is not an issue in present work.

Nevertheless, we plan to add more shape descriptors in future research, combining them in order to improve the accuracy of our similarity measurements. Indeed, we had already implemented and tested the CAS algorithm with the Cord and Angle Histogram (CAH) [103, 104] both stand-alone and together with SHA. However, from preliminary experiments we found no improvement in results that could justify the additional processing time and memory requirements. So, we decided to postpone the use of CAH descriptor.

For an easier comprehension of our approach we use, in this document as explanatory example, a very small set of six models from the ESB collection. The sample collection $\mathcal{D}_{eg} = \{M_1, \cdots, M_6\}$ is shown in Figure 3.2, together with a visual representation of the corresponding SHA signatures. These shape descriptors are constructed from the first

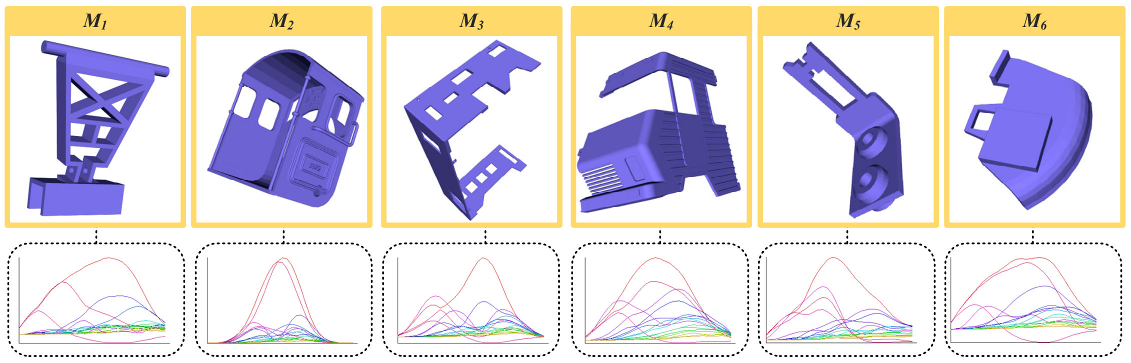


Figure 3.2: Sample collection $\mathcal{D}_{eg}$ with six models extracted from Purdue's Engineering Shape Benchmark [71] and corresponding SHA signatures.

sixteen harmonic components of thirty-two spherical functions that represent the shape. Thus, each SHA signature corresponds to a point in a 32×16 dimensional space and is represented in this document as a series of sixteen two-dimensional functions. From a quick analysis of the depicted signatures, it is possible to perceive the descriptive power of the adopted shape representation.

3.2 Hierarchically Segmented Meshes

To obtain a multi-dimensional decomposition of 3D models, we developed our collection-aware algorithm based on the hierarchical fitting primitives method. We named this algorithm Collection-aware Segmentation with Hierarchical Fitting Primitives (CAS/HFP). The HFP is a traditional mesh segmentation algorithm that produces, for a given model represented as a triangle mesh, an iteratively generated binary tree of clusters each of which is fitted by one of the predefined fitting primitives [16]. This completely automatic algorithm is a variation of the Hierarchical Face Clustering (HFC) method. The HFC, proposed by Garland *et al.* [60], represents a polygonal mesh as a hierarchy of surfaces. This hierarchy is produced by a face clustering algorithm that merges neighboring triangles into representative clusters, which are approximated by fitting planes. Authors suggested several applications for the HFC algorithm, such as surface simplification or multi-resolution radiosity, but did not apply it for segmentation.

The HFP algorithm extended the concepts used in the HFC with the purpose of performing mesh segmentation. In this approach, instead of using fitting planes, a finite set of primitives are used to compute the face clusters. As its predecessor, the HFP represents the mesh as a hierarchy of face clusters, more precisely, as a binary tree in which each node corresponds to a face cluster. This tree, called Hierarchically Segmented Mesh (HSM) contains the whole multilevel decomposition of the segmented mesh. This structure, plays an important role in our approach for model segmentation, as is computed for all models in the collection using the HFP algorithm in the initial stage of our method.

On a brief description, the HFP algorithm works as follows: initially each triangle of the mesh represents a single cluster; at each iteration, all pairs of adjacent clusters are considered; and the pair that can be better approximated with one of the fitting primitives forms a new single cluster, which represents a parent node at the above level in the tree.

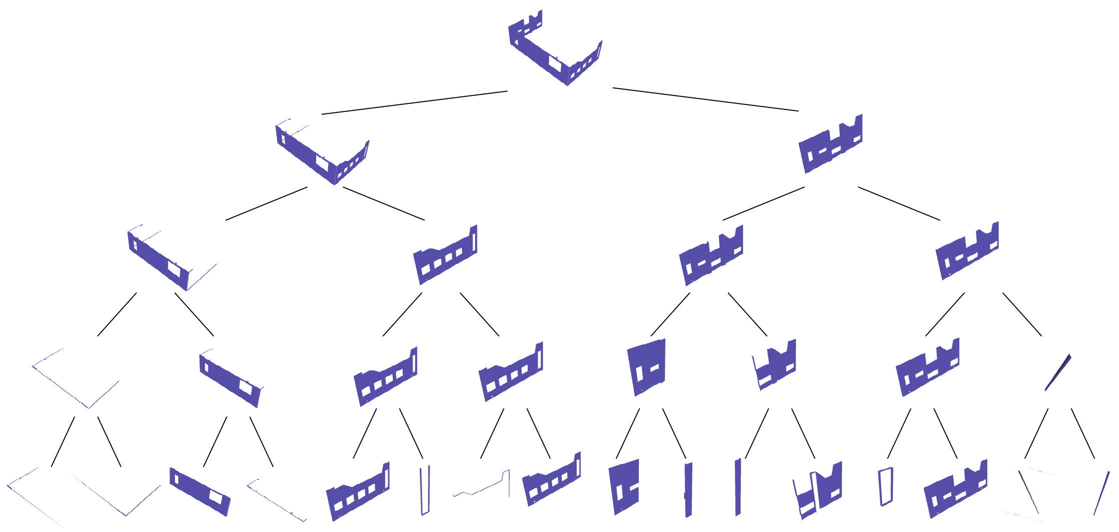


Figure 3.3: Five upper levels of the HSM tree for a 3D model.

This iteration repeats until there is only one cluster remaining, representing the whole model, and which will become the tree root. The resulting HSM tree provides a multilevel decomposition suitable for our segmentation purposes.

An excerpt of a HSM tree for a three-dimensional model is show in Figure 3.3. The interpretation of an HSM tree happens in reverse order regarding its construction by the HFP algorithm, *i.e.* it starts by the root node - the last cluster computed by the algorithm. Thus, in the given example are depicted only the four upper levels of the HSM tree, since including all levels of the tree will be impractical. Nevertheless, it is possible to perceive from this partial tree that segment complexity is not similar among all nodes in the same level, which makes automatic identification of relevant mesh segments an hard task. In our research we solve this issue by combining the HFP algorithm with the CAS decomposition method.

In order to provide automatic decomposition of models in the collection, the proposed CAS/HFP algorithm takes advantage of the fact that HSM trees contain the whole multilevel segmentation of the object to speed up decomposition. Indeed, our algorithm is divided in two phases: the initialization and the iteration. The first phase initialize the data structures used in the algorithm while the second performs the segmentation based on the information stored in these structures.

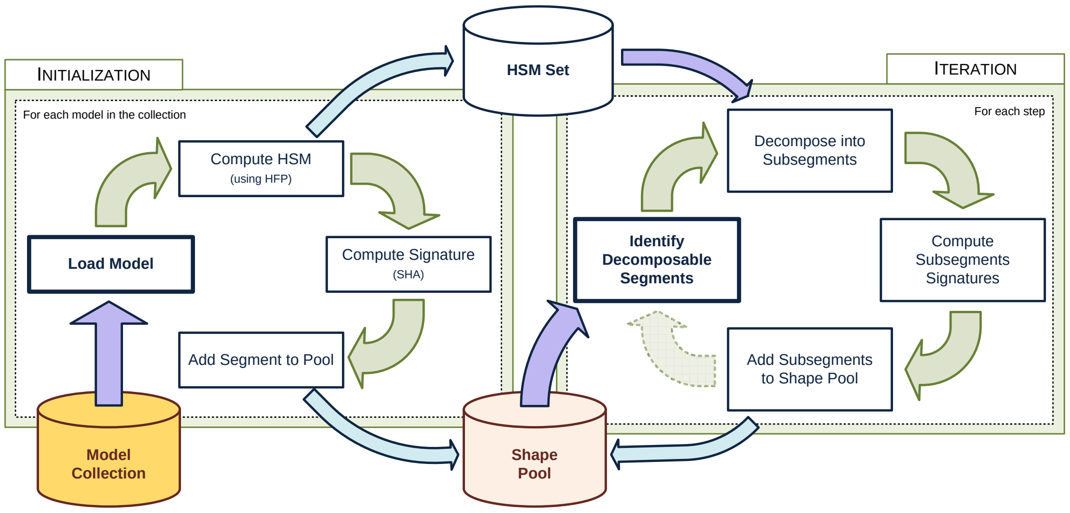

INITIALIZATION
For each model in the collection
Compute HSM (using HFP)
Load Model
Compute Signature (SHA)
Add Segment to Pool
Model Collection
HSM Set
Shape Pool
ITERATION
For each step
Decompose into Subsegments
Identify Decomposable Segments
Compute Subsegments Signatures
Add Subsegments to Shape Pool

Figure 3.4: Block diagram of CAS/HFP alorithm.

During the initialization phase all models in the collection are processed and results are stored for later use in the next phase. This way, each model just need to be segmented once using the HFP algorithm, and the resulting segmentation trees are then used to iteratively decompose all models in the collection. Indeed, as illustrated in Figure 3.4, the estimation of HSM trees, *i.e.* running the HFP algorithm, happens only once for each model at the initialization stage. These segmentation trees are then stored in the *HSM set*, one of the data structures constructed during the initialization phase, that is used to create the *shape pool*.

3.3 Shape Pool

In the initialization stage, besides the HSM trees, also shape signatures are computed for all models in the collection. These signatures are stored in a structure, which we called *shape pool*, together with the corresponding segments - at this stage, the whole models. Moreover, during the iteration stage, data stored in the *shape pool* are used to identify decomposable segments. Then, segments resulting from such decomposition and corresponding signatures are also added to the *shape pool*. Therefore, the *shape pool* is a dynamic structure that will store the most important information produced by the algorithm: the pairs segment-signature that will be later used to construct the *shape thesaurus*.

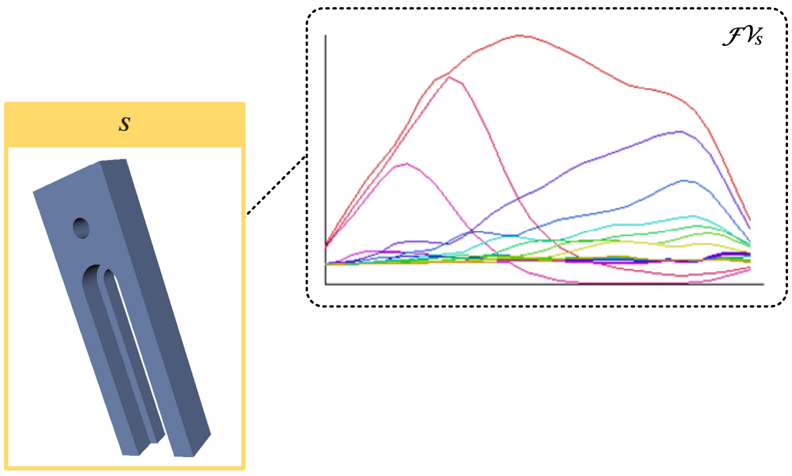


Figure 3.5: 3D model S and corresponding signature $\mathcal{FV}_S$.

Indeed, the *shape pool* is a vary important part of our approach, not only because it keeps the final result of the segmentation algorithm but mainly because the efficiency of the CAS/HFP algorithm depends greatly on it. This happens because the determination of the segments to decompose is a key step, executed at every iteration for each segment in the *shape pool*.

Basically, the *shape pool* contains a set of pairs segment-signature, as the one depicted in Figure 3.5. In practice, the segment in the pool is not really a mesh or a shape. Instead, it is just a reference to a node in the HSM tree stored in the *HSM set*. On the other hand, the signature is the feature vector produced by the SHA descriptor. Conceptually, the *shape pool* can be seen as a multidimensional dataset, where each point in space corresponds to a segment. In truth, this structure was implemented as a n-dimensional space, where n is the signature length and depends only of the shape descriptor used. In our case, and since we are using SHA descriptors, we have $n = 544$ dimensions.

In a formal definition, a collection $\mathcal{D}$ of m models is specified as

$$\mathcal{D} = \{M_1, M_2, ..., M_m\}$$

and the segments resulting from the model decomposition are referred as $S_{i,node}$, where i indicates the model to which the segment belongs and $node$ indicates the corresponding HSM tree node.

Considering that a three-dimensional shape S is represented by a single n-dimensional signature as described in Section 2.2, the corresponding feature vector $\mathcal{FV}$ is specified as

$$\mathcal{FV}_S = \{f_{S_1}, f_{S_2}, ..., f_{S_3}\},$$

where each f_{S_i} is a floating point value that corresponds to a shape feature. Thus, the *shape pool* is a set $\mathcal{SP}$ of pairs

$$SP_{i,node} = \langle S_{i,node}, FV_{S_{i,node}} \rangle$$

where i and $node$ maps to the corresponding segment, *i.e.* a node in the model HSM tree. At the end of the initialization phase, $\mathcal{SP}$ contains the pairs correspondent to the segments that represent the whole models, *i.e.* the HSM tree roots, and corresponding signatures. Then, during the iteration phase the CAS algorithm append and search this multidimensional space in order to determine the model decomposition.

3.4 Identification of Decomposable Segments

At the end of the initialization phase, the *shape pool* contains segment-signature pairs that correspond to the whole model of every object in the collection. To decompose these objects we apply an iterative process over the segments in the *shape pool*. In each step of the iteration phase, the information at the *shape pool* is updated according to the decomposition heuristics. Conceptually, what happens in this process is the construction of the CAS decomposition trees of models in the collection. These are created based on the corresponding HSM trees. Figure 3.6 depicts the decomposition of a single model using our approach. In the first iteration, a CAS decomposition tree is created with the whole model as the tree root. In each iteration the nodes are identified as ”decomposable”, ”decomposed” or ”not decomposable” depending to its distinctiveness regarding all other models in the collection. Then, nodes are expanded according to its classification, proceeding to the next iteration. The decomposition finishes when no more nodes are identified as decomposable. In the following paragraphs we detail this process.

At the beginning of each iteration step, the decomposable segments in the *shape pool* must be identified in order to be further decomposed if necessary. This is indeed the major challenge of our algorithm. It is not trivial to automatically identify which segments

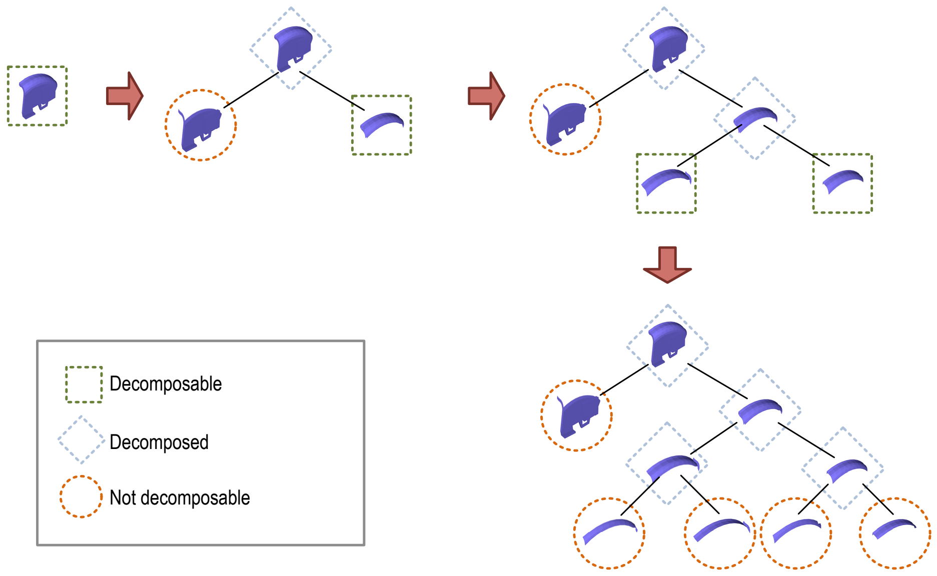


Figure 3.6: Evolution of model decomposition during the iteration phase.

should be further decomposed, it is necessary to determine the distinctiveness of each segment, which depends on the number of segments similar to it in the *shape pool*.

To determine the decomposability of segments, we suggest use segment signatures, measure the distance between them, and then, for every segment, count the number of segments whose distance is within a given range. We called this value *similarity threshold*, σ, and it determines how much geometrically similar two objects has to be in order to be considered similar.

If the count of similar segments is above a given value, which we called *similar count threshold*, τ, the segment should be further decomposed, since there are enough similar shapes in the pool to flag it as a recurrent shape part in the given collection. Indeed, due their importance, these two values are the main parameters of the CAS algorithm. While the *similarity threshold*, σ, represents a distance in the signature space, the *similar count threshold*, τ, corresponds to a percentage of the *shape pool* size.

In an extreme situation, with an extremely low similarity threshold ($\sigma \approx 0.0$) and unreasonable high similar count threshold ($\tau \approx 1.0$), the only sub-part similar in models is the triangle, the basic element of segments mesh, and all objects will share this common shape. However, the idea behind this approach is to decompose models into a meaningful and not trivial set of shapes. Decomposing a model to the triangle level is quite useless for our approach. Thus, using such values produces weak decomposition results. Indeed, determination of σ and τ values that provide an effective decomposition was an issue tackled during our research.

3.4.1 Nearest Neighbor Search

Considering the *shape pool* as a multidimensional dataset $\mathcal{SP}$, we do not need to measure the distance between all segments to flag a segment as decomposable, which would be a time consuming task. This can be done by using a k-nearest neighbor (k-NN) search algorithm, setting the k value to the given similar segment count threshold. Thus, to determine the decomposability of segment in the *shape pool*, we just need to apply the k-NN algorithm to each not already decomposed segment $S_{i,node}$, obtaining a list of similar segments ordered by signature distance. This neighbors list is given by

$$\mathcal{N}_{FV_{i,node}} = kNN(\mathcal{SP}, FV_{i,node}, k),$$

where $kNN(\mathcal{SP}, FV_{i,node}, k)$ performs a k-NN search in the dataset $\mathcal{SP}$ that returns the k nearest neighbors of point $FV_{i,node}$. To determine the correct k for the every step of the CAS iteration step, it is necessary to compute the value correspondent to the percentage of the size of the *shape pool*,

$$k = Size(\mathcal{SP}) \times \tau. \tag{3.1}$$

After estimating k for the present iteration, we check if all items in the neighborhood $\mathcal{N}_{FV_{i,node}}$ are within the *similarity threshold*. In practice, it is sufficient to check if the distance between $FV_{i,node}$ and the farthest element of $\mathcal{N}_{FV_{i,node}}$ is smaller than σ. Since this list is ordered, this comparison can be done in constant time with respect to the neighborhood size. Formally, considering $d(\alpha, \beta)$ the distance between two points in space and setting $k = \tau$, the decomposability of a segment is given by

$$Decomposable(SP_{i,node}) = \begin{cases} 1, \text{ if } d(FV_{i,node}, Last(\mathcal{N}_{FV_{i,node}})) < \sigma, \\ 0, \text{ otherwise} \end{cases}$$

where $Last(\psi)$ returns the last element of list ψ, in this particular case it corresponds to the signature of the segment less similar to $S_{i,node}$, according to SHA shape descriptor. To improve time efficiency, the quadratic Euclidean distance is used to compute $d(\alpha, \beta)$. In Section 2.3.2 we described with some detail this and other similarity measurements.

An alternative way to achieve the same goal is using within-distance (or α-cut) search algorithm instead of the k-NN search technique. This algorithm will identify all segments within a given distance. In this case the cut-off value should be set to the *similarity threshold* value, *i.e.* $\alpha = \sigma$, and to determine if a segment is decomposable we check if the number of returned segments is above the *similar segments threshold*. After preliminary experiments we conclude that no relevant improvements over k-NN search are achieved by using this method. Therefore, focused on the k-NN search technique.

The simplest approach to k-NN determination is the linear search, also referred as the naive approach, which is similar to our initial suggestion since it basically measures all distances and keep the closest segments. This method has a linear running time, but different approaches were suggested for the k-NN search problem with better time complexities, such as the ones based on spatial-partitioning methods. A quite simple and commonly used example is the kd-tree [20], which allows k-NN searches in sub-linear time with respect to dataset size. However, this and other sub-linear approaches behave badly in a very high dimensional spaces, such as our signature space.

Therefore, despite of its poor time-complexity, linear regarding *shape pool* size, we are using a naive approach since other methods require additional complex data structures. Besides increasing the CAS/HFP algorithm memory requirements, these structures hinder the effectiveness of sub-linear algorithms. Moreover, in our research we are mainly concerned in validate the proposed approach with benchmark collections. In the future, more efficient k-NN search methods can be used.

3.4.2 Within Range Search

Nevertheless, to improve the behavior of our linear search and considering that we do not really need to know the nearest neighbors but only if there are enough similar segments, we slightly changed the algorithm. Instead of searching for the k nearest neighbors or determine the number of segments within a given range, our version just tries to find out if there are k segments within a given threshold. We call it the k-within range (k-WR) estimation.

The proposed algorithm to determine if exist at least k points within a range r of query Q in dataset ξ is detailed below in KWR. It sweeps over all elements in the dataset, measuring the distance between each of them and the query and checking if this distance is within the given range. When enough points in the vicinity are found the iteration stops and the algorithm produces a positive result. If not enough points within range are found, a negative response is given.

```
K-WITHIN-RANGE($\xi, Q, k, r$)
1  $n \leftarrow 0$
2  for each $P$ in $\xi$
3  do
4      if $d(P, Q) < r$
5        then $n \leftarrow n + 1$
6              if $n = k$
7                then return $1$
8  return $0$
```

Although theoretically K-WITHIN-RANGE still runs in linear time with respect to the dataset size, in practice it is much faster than a linear k-NN search since it measures much less distances. Moreover, we are aware that with some changes the execution time can be further improved without using any additional complex structures or, using such structures, achieve a sub-linear time complexity.

The improvement obtained by using k-WR technique instead of linear k-NN search when determining the decomposability of a segment is illustrated in Figure 3.7, where a over-simplified two-dimensional signature space is considered. The difference is that, while the k-NN search must compare all the points in the dataset, *i.e.* measure the similarities between the query segment and all other segments in the *shape pool*, the k-WR just compares points until k similar points are found. In the depicted example, considering the range δ and $k = 3$, the k-WR search will just visit P_1 to P_5, the first five elements of the dataset, represented by darker points in the figure. Since P_2, P_3 and P_5 are within range, the answer to the k-within range search can be given positively as soon as P_5 is compared with Q, dismissing further comparisons.

The proposed CAS algorithm works with any of the search techniques referred above or others that provides the necessary answer regarding similar segments count. However,

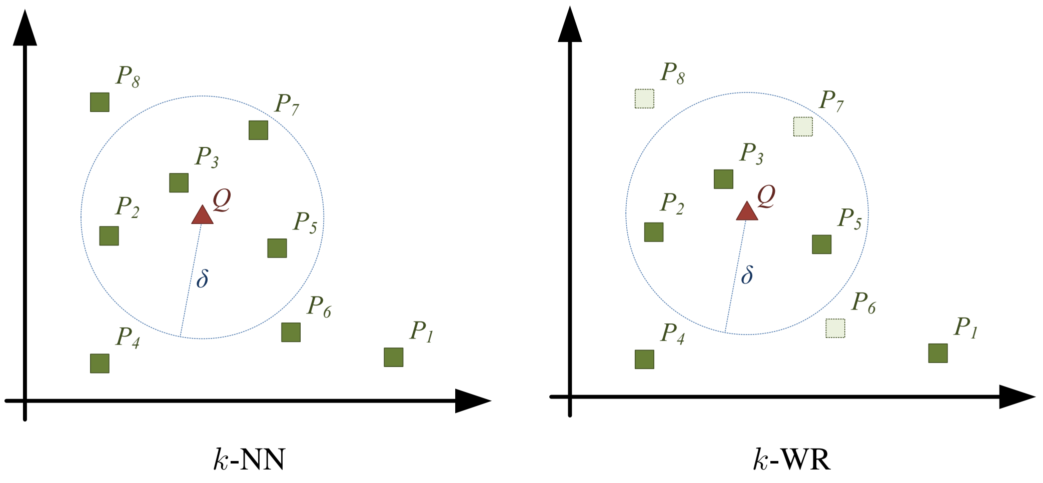


Figure 3.7: Visited points during the decomposability determination for segment represented by signature Q, using the k-NN search (left) or the k-WR technique (right). While the k-NN search algorithm visits all points in the dataset, the k-WR search algorithm just visits the darker points.

due to its simplicity and efficiency, when compared with linear k-NN search, we suggest the use of the k-WR algorithm to determine segment decomposability. Moreover, by using the k-WR technique instead of a k-NN search, the decomposability determination of a segment is immediately obtained, without additional measurements and comparisons.

Summarizing, to determine if a segment is decomposable we apply a k-WR search algorithm in the signature space, using as parameters two values: the *similarity threshold* (σ), which sets when two shapes are considered similar, and the *similar count threshold* (τ), which defines how many segments should be similar in order to be considered not distinctive. These are the two parameters used to tune our decomposition algorithm. An additional parameter is the *maximum depth* (λ), also referred as *iteration cut-off*, which preset a maximum iteration count for the second stage of the CAS/HFP, the iteration phase.

3.5 Sub-segments

In each iteration of the CAS algorithm, after identifying the segments that should be decomposed we must determine their decomposition. To that end, we use the HSM trees created during the initialization phase and stored in the HSM set. Since these trees contain the whole multilevel segmentation of the model, it is fast and simple to determine

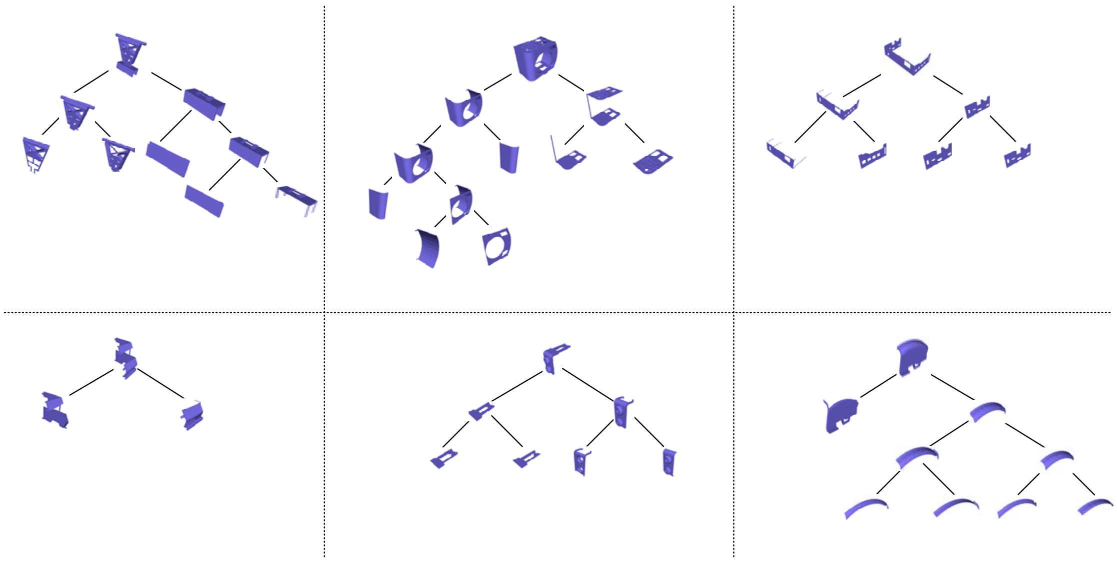


Figure 3.8: Decomposition trees produced by CAS/HFP for models in collection $\mathcal{D}_{eg}$.

the decomposition of a segment. This is done simply by looking at the corresponding node in the tree and using the child nodes as sub-segments. Moreover, since every pair segment-signature $SP_{i,node}$ in the *shape pool* keeps a reference to the corresponding node, this task is accomplished in constant time.

For every newly created sub-segment, its signature is computed using the SHA shape descriptor. This signature, $FV_{i,node}$, is then attached to the sub-segment $S_{i,node}$, as well as the reference to the corresponding node in the HSM tree, thus creating a new pair segment-signature $SP_{i,node}$. Then, all this information is added to the *shape pool*. When all segments identified as decomposable have been decomposed and the originated sub-segments added to the *shape pool*, the iteration starts over by identifying again the decomposable segments. This cycle continues unless one of the stop conditions has been verified. Basically, there are two conditions that may stop the cycle: when a pre-defined iteration count is achieved or if there are no more decomposable segments.

The iteration phase of CAS algorithm is not complex. Indeed, each iteration step of this phase relies on two different cycles, as detailed in CAS-HFP-ITERATION-PHASE. The first cycle identifies the decomposable segments in the *shape pool*, using DETERMINE-K to compute the k value as described in Equation 3.1. The second cycle iterates over the decomposable segments identified previously. For each of these segments it is necessary to identify the left and right nodes in the respective HSM tree, which is accomplished through GET-SUBSEGMENTS and to compute their signatures, create the corresponding shape-segment pairs and add them to the *shape pool*.

```
CAS-HFP-ITERATION-PHASE($\mathcal{SP}, \mathcal{HSM}, \sigma, \tau, \lambda$)
 1  $c \leftarrow 0$
 2  repeat
 3        $\xi =$ empty set of decomposable segments
 4        $c \leftarrow c + 1$
 5        $k \leftarrow$ DETERMINE-K($\mathcal{SP}, \tau$)
 6        for each $S_{i,node}$ in $\mathcal{SP}$
 7        do
 8           if $S_{i,node}$ not already decomposed
 9             then
10                if K-WITHIN-RANGE($\mathcal{SP}, S_{i,node}, k, \sigma$)
11                  then
12                       Add $S_{i,node}$ to $\xi$
13
14        for each $S_{i,node}$ in $\xi$
15        do
16           $\{S_{i,child_{left}}, S_{i,child_{right}}\} \leftarrow$ GET-SUBSEGMENTS($\mathcal{HSM}, i, node$)
17           $FV_{i,child_{left}} \leftarrow$ COMPUTE-SHAPE-DESCRIPTOR($S_{i,child_{left}}$)
18           $FV_{i,child_{right}} \leftarrow$ COMPUTE-SHAPE-DESCRIPTOR($S_{i,child_{right}}$)
19           Add $\langle S_{i,child_{left}}, FV_{i,child_{left}} \rangle$ to $\mathcal{SP}$
20           Add $\langle S_{i,child_{right}}, FV_{i,child_{right}} \rangle$ to $\mathcal{SP}$
21     until IS-EMPTY($\xi$) $\vee \neg(c < \lambda)$
22  return $\mathcal{SP}$
```

During the process described above, CAS decomposition trees for each model are conceptually built. Indeed, such trees does not exist explicitly in the data structures but the approach behind the algorithm is based on them. In Figure 3.8 we depict the set of decomposition trees produced by the CAS/HFP algorithm for the models in the example collection $\mathcal{D}_{eg}$. Despite being an extremely small sample, it is possible to observe that the trees are unbalanced but the shapes on the tree leafs have similar complexity.

Concluding, at the end of the CAS/HFP algorithm our approach has produced a *shape pool* containing all the segments from every model in the collection, as depicted in Figure 3.9. This *shape pool* is later used to create the *shape thesaurus*. However, to achieve useful results for the shape thesaurus creation, we must identify a good value for an important parameter of the CAS/HFP algorithm, the *similarity threshold*.

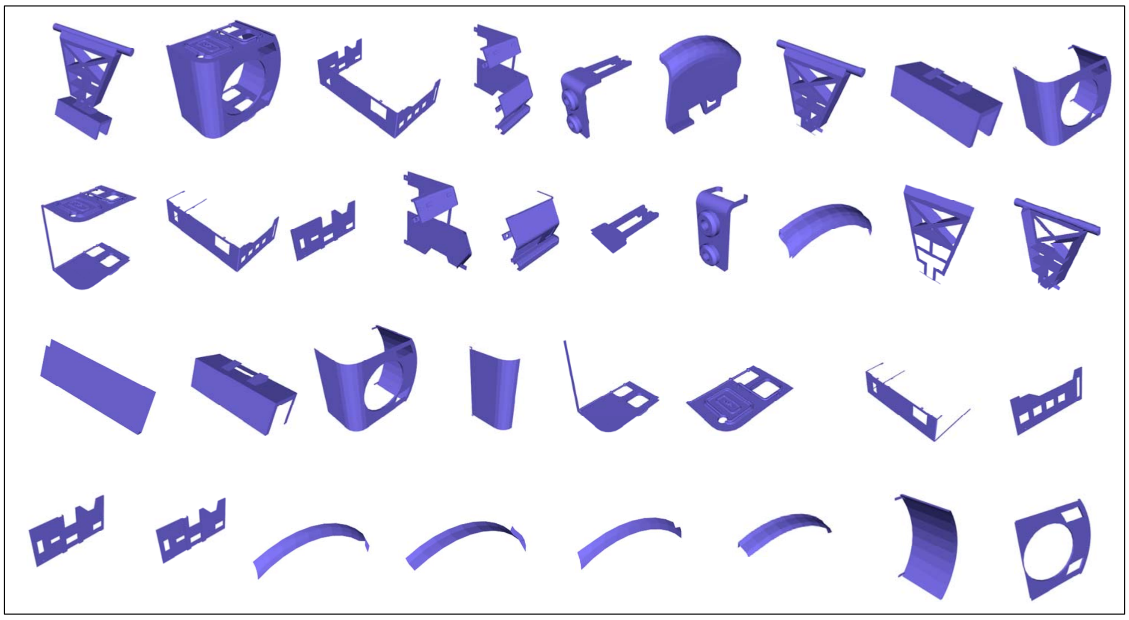


Figure 3.9: *Shape pool* $\mathcal{SP}_{eg}$ resulting from the decomposition of collection $\mathcal{D}_{eg}$.

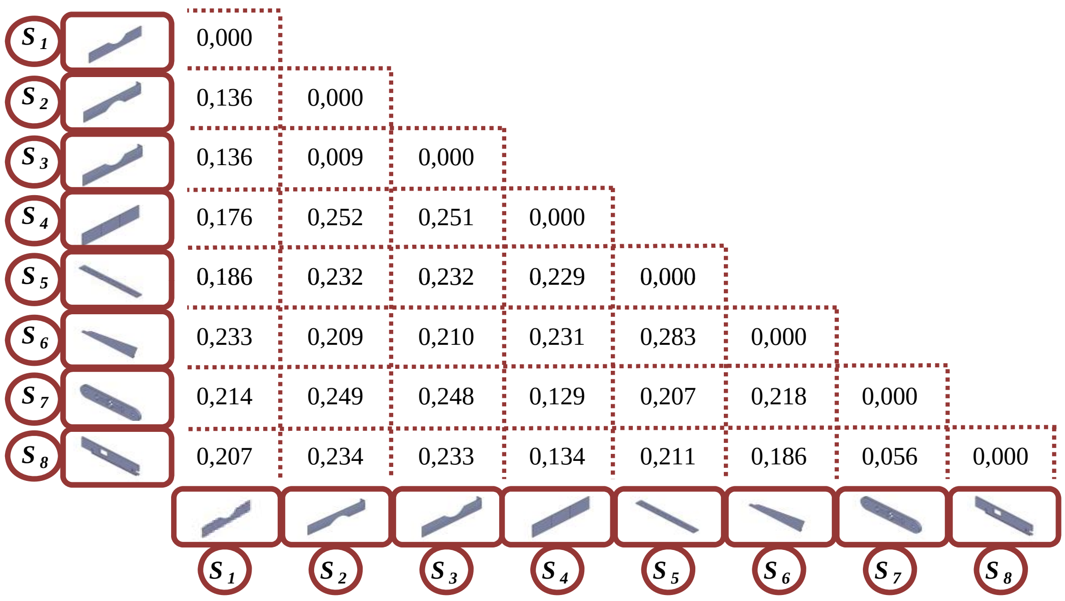
3.6 Similarity Threshold Determination

As referred above, the *similarity threshold* (σ) plays an important role in the CAS algorithm. Is based on this threshold that the algorithm determines if two shapes should be considered similar and uses this information to decide if a give shape should be further decomposed. Formally, we consider two shapes S_1 and S_2, represented by the feature vectors FV_1 and FV_2, respectively, as similar when

$$d(FV_1, FV_2) < \sigma.$$

The determination of a reasonable σ value, or at least a reasonable range for it, was an issue studied during our research and the reader can find a detailed description of this study in Annex A. Still, we briefly describe it in the next paragraphs while presenting the conclusions we deduced from this study.

Considering that the SHA descriptor is used by CAS/HFP to compute the signature of our shape, we analyze the behavior of the resulting feature vectors with different sets of models from the ESB collection. More precisely, we measured the distance between feature vectors of specific 3D shapes, examining the similarity between these shapes. This



	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
S_1	0,000							
S_2	0,136	0,000						
S_3	0,136	0,009	0,000					
S_4	0,176	0,252	0,251	0,000				
S_5	0,186	0,232	0,232	0,229	0,000			
S_6	0,233	0,209	0,210	0,231	0,283	0,000		
S_7	0,214	0,249	0,248	0,129	0,207	0,218	0,000	
S_8	0,207	0,234	0,233	0,134	0,211	0,186	0,056	0,000

Figure 3.10: Similarity values between geometrically similar shapes in a set ξ_{sim_1}.

similarity, $s_{i,j}$, is given by

$$s_{i,j} = d_2(FV_i, FV_j),$$

where FV_i and FV_j are the SHA signatures of shapes S_i and S_j respectively. Following a widely used methodology for similarity measurement, $s_{i,j}$ is a real number in the range

$$0.0 < s_{i,j} < 1.0,$$

where $s_{i,j} = 0.0$ if shapes S_i and S_j are equal. Further details on similarity measuring were provided in Section 2.3.2.

The first studied set (ξ_{rand}) contained randomly chosen shapes and, after computing the SHA descriptors for these shapes we measured their similarity. As expected from a random sample, the similarities between the shapes are distributed by the whole range of possible values, but most of them fall around the mid-point. Indeed, the average for the similarity among our sample of random shapes was $\overline{s_{\xi_{rand}}} = 0.54$.

However, the information above is not sufficient to define a valid range for σ. Thus, we studied sets composed of similar shapes. These shapes were selected based on the similar parts provided by the ESB search engine [90]. As for the random shapes set, we computed the SHA descriptors and measured similarities. The results obtained with one of the studied sets of similar shapes are shown in Figure 3.10. In this example we used a set containing eight shapes, $\xi_{sim_1} = \{S_1, ..., S_8\}$, and measured the similarity between these shapes. A quick analysis of the depicted distance chart allows the reader to check the behavior of $s_{i,j}$ with similar shapes.

From the analysis of given ξ_{sim_1} signature distances is possible to observe that, abiding by the definition of similarity measurement, when a shape is compared with itself the distance $s_{i,i} = 0.0$ and when two very similar shapes are compared this value is very low, as in $s_{2,3} = 0.009$. Indeed, although it might not be asserted from the depicted views, shapes S_2 and S_3 are quite similar except for the position of a curved carving in the shapes. In contrast, when two less similar shapes are compared this value is higher, as with shapes S_2 and S_7. But even in this case the measured similarity, $s_{2,7} = 0.25$, is far below the average similarity of the sample set containing random models.

The behavior described in the previous paragraph for the set ξ_{sim_1} depicted in Figure 3.10 is observed in other sets ξ_{sim_i} containing shapes considered similar by the ESB search mechanism. In our study all these sets contained eight objects similar to a given query according to ESB and were manually inspected, to guarantee the geometrical similitude of the shapes. In every studied set ξ_{sim_i}, the maximum distance we observed between shapes was less than 0.3, as illustrated in Figure 3.11. This chart presents the distance values measured between SHA feature vectors of shapes extracted from ESB organized into two distinct samples: a sample containing a set of randomly selected shapes (ξ_{rand}) and a sample containing six distinct sets ξ_{sim_i} of similar models.

From the analysis of samples ξ_{rand} and ξ_{sim_i} it is clear that distances between feature vectors of similar models are generally smaller than between random models, as expected. In this aspect no relevant information was obtained except confirmation of the obvious. However, based on the gathered measurements we estimated that the average for the feature vector Euclidean distance among similar shapes was $\overline{s_{\xi_{sim}}} = 0.191$. Based on this value and extrapolating this behavior for other sets of geometrically similar models in the collection, we suggest that a good value for the *similarity threshold* should be

$$\sigma \approx 0.2.$$

Indeed, the extrapolation referred above must be done because it is not feasible to determine the set containing geometrically similar shape for every model in a collection containing more than eight hundred models, since this operation requires manual valida-

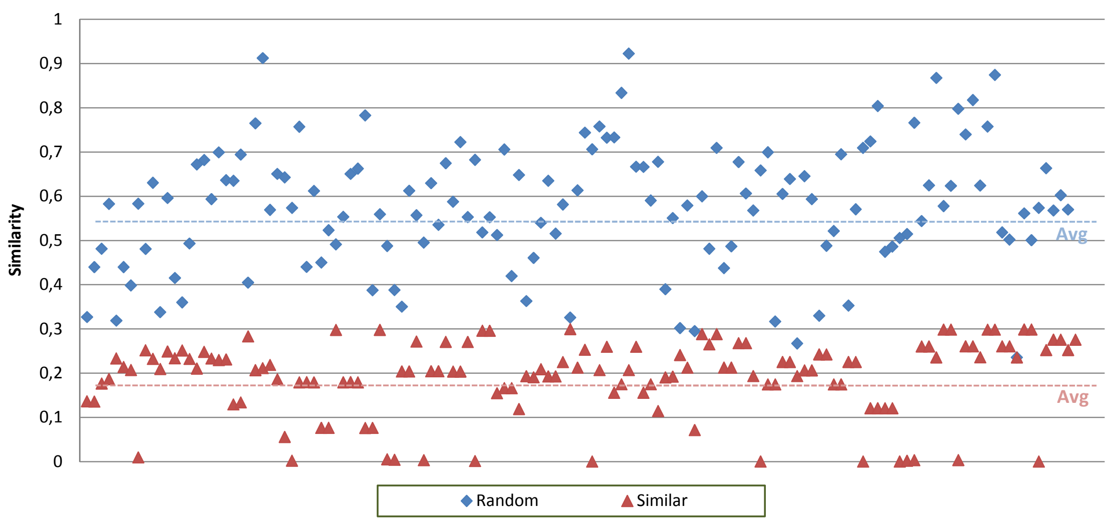


Figure 3.11: Similarity values between shapes from ESB organised into two distinct samples: randomly selected shapes and sets of similar shapes.

tion due to false positives given by the ESB query system that should be removed. Thus, to investigate further the behavior of the SHA descriptor we adopted a distinct approach. We estimated the feature vector for every model in the ESB collection and determined for each one the five nearest neighbors in the signature space.

To identify the nearest neighbors we executed linear k-NN searches using the feature vector of each model from the ESB collection as a query. From the results produced by the k-NN algorithm we constructed, for each query, a set containing the five nearest feature vectors. Each of these sets corresponds to the five more similar shapes to the query model, according to SHA descriptor. The distances between the queries and the respective nearest neighbors are depicted in Figure 3.12, where each point in the chart corresponds to a distance between a query and one neighbor. In this chart it is possible to observe that although some distances are above 0.3, a vast majority is beneath this value.

Notice that in this measurements no geometrical similarity can be guaranteed. Indeed, even if a shape is unique and no similar shape exists in the collection, the five nearest neighbors of its feature vector will be found, probably representing dissimilar models. This fact justifies the existence of larger distances than the observed when measuring sets of geometrically similar shapes that were manually verified. Nevertheless, the overall behavior does not differ greatly between the two cases. As presented in Table 3.1, more than ninety percent of the measured distances are $d \leq 3.0$, which validates our previous

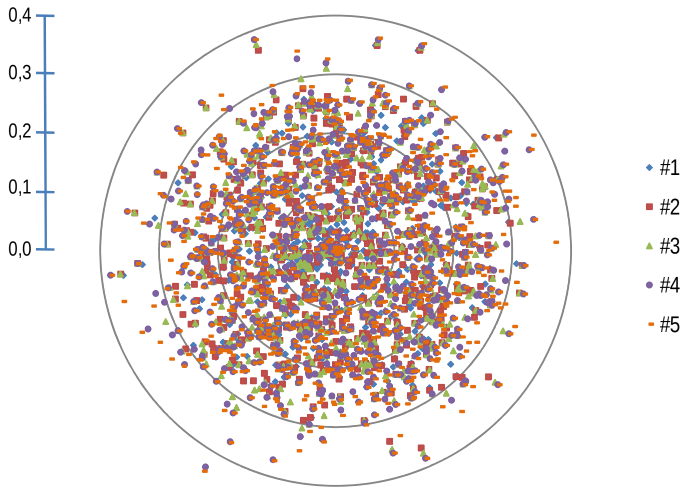


Figure 3.12: Feature vector distances for the five nearest neighbors of every model in the ESB collection.

affirmation regarding the maximum distance between feature vectors of similar shapes, considering that the remaining percentage correspond to geometrically dissimilar models.

The analysis of the feature vector distances also confirm the mean value we observed in the measurement with geometrically similar shapes. Considering the fifth element returned by the k-NN search, the average distance between this feature vector and the corresponding query is $d = 0.194$. Such observation corroborates the approximate value we suggested previously for the *similarity threshold*, $\sigma \approx 0.2$.

	#1	#2	#3	#4	#5
$0.0 \leq d < 0.1$	541	308	210	148	102
$0.1 \leq d < 0.2$	185	301	339	345	328
$0.2 \leq d < 0.3$	120	225	273	308	352
$0.3 \leq d < 0.4$	7	19	31	48	67
$d \geq 0.4$	0	0	0	4	4
$\overline{d}$	0.087	0.147	0.179	0.181	0.194
$var(d)$	0.00841	0.00834	0.00742	0.00717	0.00703
$d \leq 0.3$	99.2%	97.8%	96.4%	93.9%	91.7%
$0.15 \leq d \leq 0.25$	22.5%	36.2%	44.4%	45.7%	46.0%

Table 3.1: Statistics of feature vector distance, d, for the five nearest neighbours considering all models in the collection.

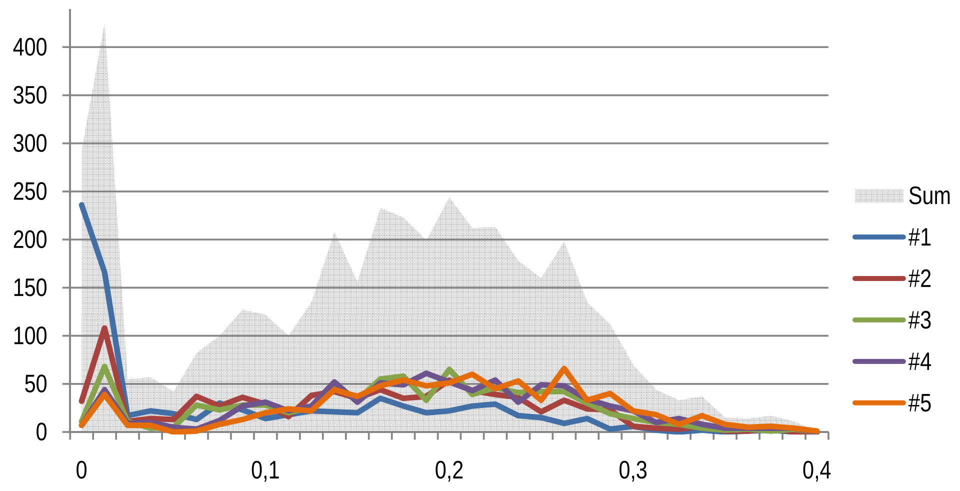


Figure 3.13: Histograms of the feature vector distances for the five nearest neighbors of every model in the ESB collection.

Additionally, we used the data gathered when comparing the nearest neighbor feature vectors to define a reasonable range for the *similarity threshold*. To that end, we analyze the distances distribution for the five nearest neighbors of each query. Figure 3.13 depicts the histograms of these distances. These histograms depicts the number of distances found within a range $0.0 < d < 0.4$. As expected, peaks are evident near the lower bound of the range, which corresponds to distances among shapes extremely similar that exist in the collection. But these peaks does not provide worthful information for our purposes.

On the other hand, when looking at the distance count in the interval $d \in [0.1; 0.3]$ it is possible to observe that in the neighborhood of 0.2 other peaks exist, especially if we sum the count of all five neighbors. Since we do not want do have only very similar shapes neither vaguely similar shapes to be considered similar, we analyzed the sum histogram and, using also the information presented in Table 3.1, we identified a reasonable range for the *similarity threshold*:

$$0.15 < \sigma < 0.25.$$

This interval provided a basis for experiments with CAS/HFP algorithm. Indeed, as presented in Chapter 5, to obtain good decomposition results the values of σ will always be within this interval, otherwise the segmentation will be visibly ineffective.

3.7 Comparing CAS/HFP with HFP

Although the CAS/HFP algorithm can be used to simply decompose 3D objects, its main objective is the creation of a *shape pool* containing the sub-parts of all models in the collection, to be used in the thesaurus construction. Thus, it might seem inappropriate to compare results of the proposed CAS/HFP algorithm with the HFP technique. However, since the CAS/HFP can produce, at the end of its execution, not only the *shape pool* but also decomposed models, we analyzed the behavior of the two approaches.

To that end we compare the segmentation obtained with the collection-aware decomposition based on hierarchical fitting primitives with the segmentation obtained using the original hierarchical fitting primitives algorithm. To compute the segmentation with the HFP algorithm we used the EfpiSoft tool [15] developed by Marco Attene.

To segment an object, the EfpiSoft tool starts by computing its HSM tree. Recall that the HSM tree of an object contains a complete hierarchical decomposition of the corresponding mesh, which means that the leaf of such trees refer to the atomic elements of the mesh, the triangles. Thus, to properly segment the object it is important to determine a method to prune the tree. Thus, to obtain the object segmentation a single parameter must be specified: the number of desired segments, N_{seg}. This value indicates the number of tree nodes to be considered for the segmentation, cutting the tree in order to obtain that number of leafs.

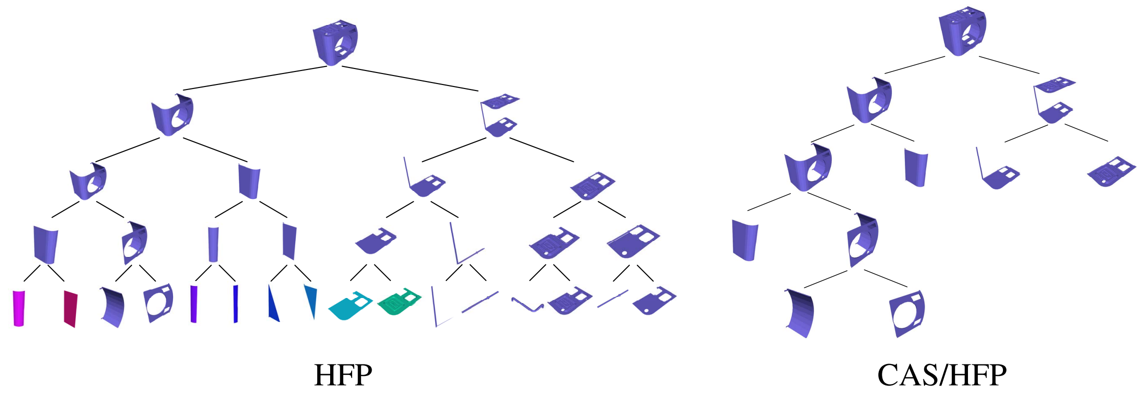


Figure 3.14: Decomposition trees produced by the HFP (left) and the CAS/HFP (right) algorithms.

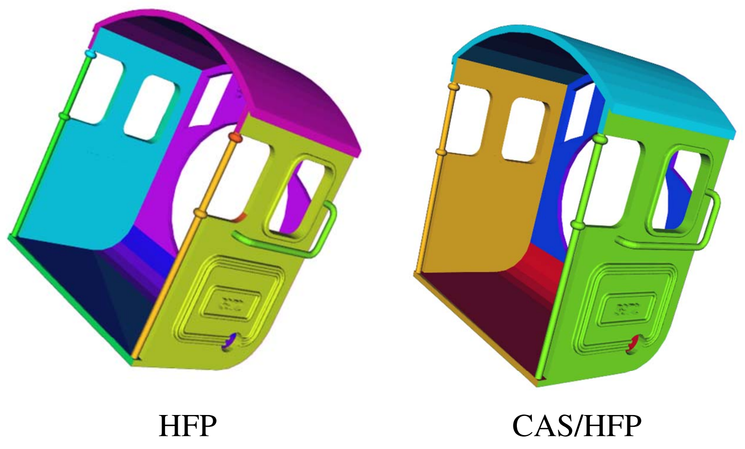


Figure 3.15: Segmentation of a 3D model using the HFP (left) and the CAS/HFP (right) algorithms.

Despite the fact that the segmentation using HFP relies on the desired number of segments, to allow easier comparison of results with our algorithm we preferred to use a pre-determined tree depth. In practice, this issue was solved by estimating the number of segments according to that tree level. This is simple, since the hierarchical segmentation produces a binary tree. In practice, we just had to compute the number of leafs in a balanced binary tree for the given tree level.

We experimentally compared the behavior of both algorithms with a set of models randomly selected from the ESB collection. In this experiment we defined the *similarity threshold* as $\sigma = 0.2$ and the maximum tree depth $\lambda = 4$, which means that $N_{seg} = 2^{\lambda} = 16$. We observed that the same object is decomposed differently, even if sometimes is just a slightly difference. One example of such difference is depicted in Figure 3.14, where we show decomposition trees of the same model with both methods. In this example, although the two segmented models, illustrated in Figure 3.15, seem apparently similar, but a closer analysis reveals significant difference. Furthermore, when analyzing the decomposition trees produced by the application of each algorithm the difference between the results produced by each algorithm is quite clear.

Comparing the two trees, it is easy to conclude that the segmentation produced by CAS/HFP algorithm is more concise than the one produced by using only HFP. But more important than this is the fact that, in our approach, all leaf segments have similar granularity. On the other hand, in the HFP tree some leaf segments are just planar patches while others are still complex parts. This leads to the unbalanced models decomposition that occurs when trying to stop HFP segmentation based on a pre-defined value instead of setting it manually during decomposition, as allowed by EfpiSoft. Based on these results we concluded that, for automatic decomposition of models, our method produces better results than the hierarchical fitting primitives algorithm alone, but needs much larger computation times per model. Still, we underline that our aim is constructing a thesaurus for shapes, a task that is supposed to run in batch mode, without user intervention. So, it is acceptable that our CAS/HFP algorithm takes some time to process a collection.

3.8 Summary

In this chapter we introduced a novel approach to decompose three-dimensional models in a collection. The Collection-aware Segmentation (CAS) performs the decomposition of each model taking into account all the other models in the collection. To that end, it relies on the concept of distinctive region to identify which segments of a model should be further decomposed while performing an iterative decomposition of the model. The distinctive segments are iteratively subdivided into subparts while common segments are left unchanged. As a result the CAS algorithm produces not only the segmented models, but, more important, a set of shapes that will be used to construct the thesaurus.

After presenting a conceptual overview of the proposed methodology, we define some conditions to validate our approach. Theoretically, the CAS algorithm works with generic collections of three-dimensional models, using different techniques to perform the basic shape segmentation and any shape descriptor to measure the similarity between shapes. However, to devise a practical solution we restricted the collection type to 3D CAD models, we computed the shape feature vectors through rotation invariant spherical harmonics [78] and used the hierarchical fitting primitives [16] as a basic segmentation tecnhique. Therefore, we called our decomposition algorithm collection-aware segmentation based on hierarchical fitting primitives (CAS/HFP).

The CAS/HFP algorithm is divided in two distinct stages. The initialization stage and the iteration stage. In the initialization stage we compute the hierarchically segmented meshes (HSM) for every model in the collection. The HSM is the outcome produced by the hierarchical fitting primitives segmentation (HFP) algorithm. Basically, it is a binary tree of face clusters that contains the whole multilevel decomposition of the model from the complete model to every face on the mesh. Besides the HSM trees, during the initialization stage it is also created the *shape pool*.

The *shape pool* is a data structure that contains a set of segment-signature pairs. Thus, in the initialization stage each model is considered a single segment, its signature is computed and the corresponding pair is stored in the *shape pool*. This signature is the feature vector that represents the rotation invariant spherical harmonics (SHA) descriptor for that segment. In practice, the *shape pool* is represented as a multidimensional dataset in the feature vector space, where each point maps to the corresponding segment. This structure plays an important role in the decomposition algorithm.

Indeed, the second stage of CAS/HFP algorithm - the iteration stage - consists on iteratively grow the *shape pool* while processing the HSM trees produced in the initialization stage. To that end, in each step of the iteration the segments in the shape pool are inspected in order to identify those which must be docomposed further. This identification is accomplished through a search algorithm we developed specifically for this purpose. The within range search (k-WR) algorithm determines in a simple and quick manner if at least a given number of points are within a given range. By applying this search algorithm to every point in the *shape pool* we immediately identify the decomposable segments.

Having identified all decomposable segments, the second step of the iteration takes advantage of the multilevel decomposition represented in the HSM trees to compute the two sub-segments of each identified segment. Since our data structures were designed for this objective, such computation is obtained directly in constant execution time. Indeed, the segment in the *shape pool* refers to a node in the corresponding HSM tree and the sub-segments are immediately available by simply following that reference. These newly found segments are added to the *shape pool*, making it grow and then, after all decomposable segments have been processed, the iteration starts over again.

This iterative loop stops in two conditions: when no more decomposable segments are found or if a given maximum iteration is reached. In theory this second stop condition is not necessary, since in a worst case the triangles that constitute the leafs of the HSM trees will not be decomposable and the iterative loop will stop when only these segments remain. By the definition of HSM tree, these segments exist are within a finite depth in the tree. Thus the loop will certainly stop in finite time, but this might be unnecessarily long. To avoid this we suggest the use of a maximum iteration limit.

Besides the maximum iteration limit, a more important parameter should be provided to the algorithm, the *similarity threshold*. This value defines the degree of similarity between two shapes to be considered as similar, when determining if a segment is decomposable. To identify a reasonable range for this parameter, we studied the behavior of the SHA shape descriptor within the engineering shape benchmark collection and presented it in this chapter.

Finally, we presented a comparison between our decomposition methodology and the original HFP approach. Indeed, the main goal of CAS/HFP is the creation of a set of shapes upon which the thesaurus will be built, rather simply decompose the models. Nevertheless, our approach produces the decomposition of the models and we compared it with the results obtained when decomposing the models through basic HFP. From this comparison we concluded that our approach produces better results if the decomposition is accomplished without human intervention but it takes more time to compute. However, since our goal is to have an automatic decomposition technique to be executed in batch mode, the CAS/HFP method fulfills better our needs.

4 Thesaurus-based 3D Shape Retrieval

One innovative contribution of the present research work is the introduction of a totally different approach to three dimensional model retrieval. Most existing solutions retrieve a shape by comparing complete models, applying time-expensive partial matching algorithms to all objects in a collection or considering only small portions of each model discarding the remaining information. However, these approaches are inappropriate to perform partial queries on large collections, unless a large amount of geometric features are ignored. In our approach, we use all geometric information of all models in the collection to retrieve a 3D object. To overcome the time complexity inherent to such approach we transposed the thesaurus concept used in text information retrieval to 3D objects, introducing the *shape thesaurus*.

Conceptually, the *shape thesaurus* consists on a set of terms that represent sub-parts of models in the collection. Each of these terms correspond to several segments that share geometrical features. Therefore, the construction of a effective *shape thesaurus* relies on a multilevel decomposition of the models in the collection and on a correct clustering of the resulting sub-parts. We use an hybrid clustering algorithm to compute a partition for the *shape pool* produced by the Context-aware Segmentation (CAS) method, described in previous chapter. From this partition we determine the terms of the thesaurus and construct the indexing structures.

In this chapter we will start by presenting a framework for 3D shape retrieval with partial queries. Then, we introduce in detail the *shape thesaurus* and the concepts behind it. Finally, we will describe the construction of such structure and how it can be used to efficiently retrieve 3D objects using part-in-whole matching.

4.1 Overview of the Framework for 3D Shape Retrieval

During our research on 3D shape retrieval with partial matching we aimed at solving both scalability, complexity and matching problems faced by existing works. To that end we developed a framework that differs from those commonly used by 3D object retrieval approaches, since we are using an indexing technique from text retrieval, but transposing the concepts to 3D shapes. In our opinion, the use of a thesaurus together with an inverted index, similarly to what happens in text information retrieval, provides a good solution for the scalability and complexity problems. In truth, thesaurus-based approaches are employed with recognized success in text collections, but not in the 3D shapes context.

Our approach to information retrieval shares the same basic principle of most information retrieval solutions. The framework, depicted in Figure 4.1, is divided into two distinct parts: the classification component and the retrieval component. While the first indexes the collection, the second performs queries in the indexed collection. Indeed, this is valid for almost any information retrieval system. However, since in our solution the indexing mechanism is based on a thesaurus of three-dimensional shapes, both classification and retrieval components are very specific for this approach.

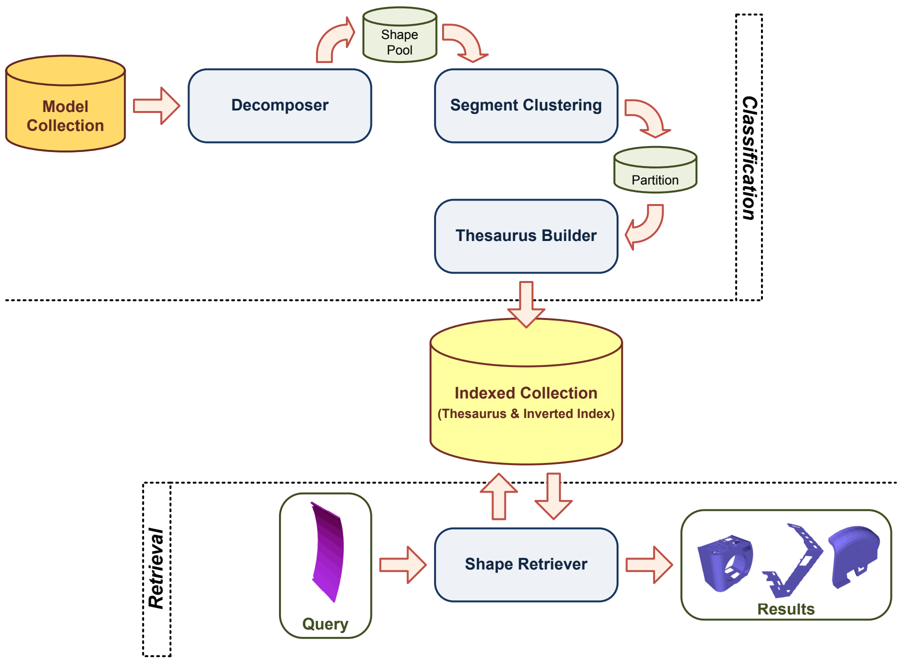


Figure 4.1: Overview of thesaurus-based 3D shape retrieval framework.

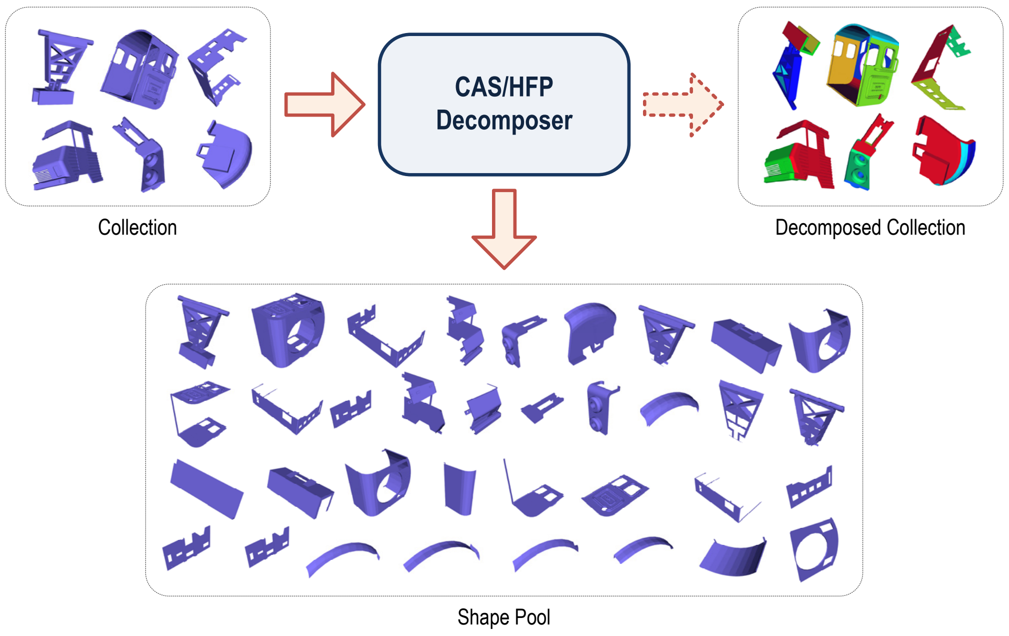


Figure 4.2: Decomposition of collection $\mathcal{D}_{eg}$ using the CAS/HFP algorithm.

4.1.1 Classification

The classification part is composed by three individual components: decomposer; segment clustering; and thesaurus builder. The *Decomposer* processes the models collection, performing a multilevel decomposition using the CAS algorithm and storing the resulting sub-parts in the *shape pool*. In Figure 4.2 we illustrate the application of CAS/HFP algorithm to the collection $\mathcal{D}_{eg}$ introduced in previous chapter. As shown, two outputs are produced: a set of decomposed models and the *shape pool*. For the rest of the classification part only the *shape pool* is necessary, the decomposed models itself are ignored, since we are only interested in their sub-parts.

In the second stage of the classification part, the segments in the *shape pool* are analyzed by the *Segment Clustering* component in order to compute a partition for the *shape pool*. This partition contains the shapes in the pool grouped according to its geometrical similarity. Section 4.3 describes the method we applied to create this partition.

Finally, the resulting partition is used by the *Thesaurus Builder* to create the *shape thesaurus* and corresponding inverted index. Since these structures mimic the homonym concepts in text retrieval, their construction follows closely the principles widely used in this field, as described in Section 4.4.

4.1.2 Retrieval

On the other hand, the retrieval part of our 3D shape retrieval system consists on a single component, the *Shape Retriever*. This component receives as input the 3D shape to be used as an example-query and retrieves from the indexed collection the corresponding query results, which is a list of models that are fully or partially similar to the query.

Succinctly, the proposed retrieval pipeline starts by computing the descriptor of the query shape and then finds in the thesaurus the similar terms. Next, the inverted index is accessed in order to identify, for each of these terms what are the models that contain them. Finally, a list containing the identified models properly ordered is constructed and returned as a query result. In Section 4.5 we explain in detail our retrieval technique which allows a fast model retrieval with part-in-whole matching. Meanwhile, in the following sections we will describe the structures and methods which support this technique.

4.2 Shape Thesaurus

Currently, existing 3D shape retrieval systems that supports partial queries search the contents of object collections by performing sequential searches. Indeed, whether they inspect the complete shapes or only few of their parts, these approaches generally perform a serial scanning of the indexed contents. Eventually they use mechanisms and data structures that provide efficient ways to search in a multidimensional space constructed with the shape signatures. But, even in these cases, the signature space contains the signatures of all elements in the collection, or of all considered parts. Thus, although these approaches can be somehow practical with small collections, when the collection get larger, the signature dataset grows accordingly, making the search on this space too expensive in terms of computation time.

To overcome this problem we devised a novel approach to three-dimensional shape retrieval. We transposed the lexicon concept widely used in textual information retrieval, by introducing the concept of *shape thesaurus*. While in text information retrieval the lexicon contains the vocabulary used in the collection, *i.e.* the words used in the documents, the *shape thesaurus* will contain the shapes used to construct the models in the collection. In some very specific domains the identification of the ”3D words” is easy,

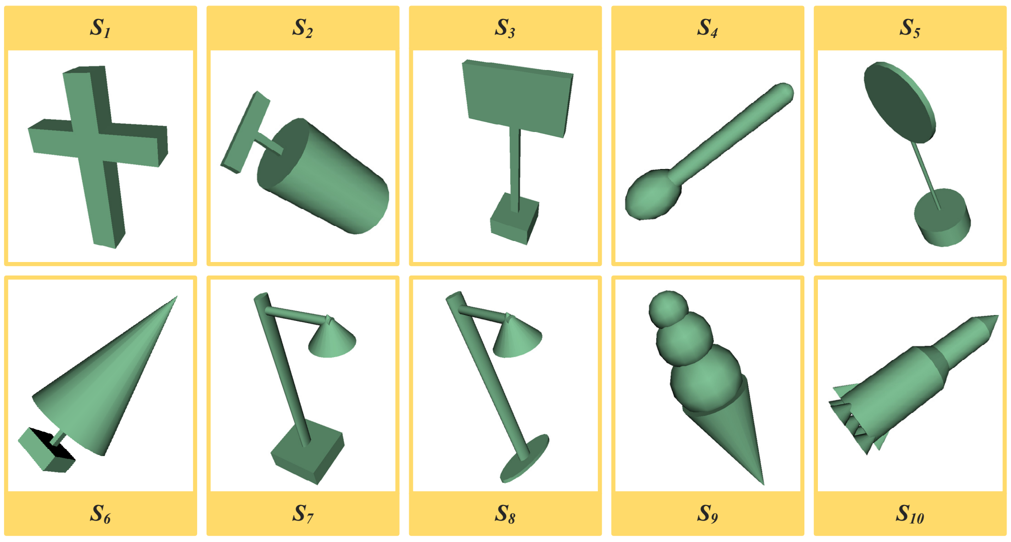


Figure 4.3: Collection $\mathcal{D}_p$ of models represented using primitive instancing.

e.g. in collections composed by solid 3D objects modeled using Constructive Solid Geometry (CSG) where the construction primitives are explicit in the model. However, for general 3D objects the isolation of the ”3D words” is hard.

An example of a collection $\mathcal{D}_p$ constituted exclusively by solid objects modeled using primitive instancing [53] with a small set of primitives (box, sphere, cylinder and cone) is presented in Figure 4.3. Since the models in this collection rely on a primitive instancing representation, the shapes used to construct them are explicitly available. Thus, the construction of the *shape thesaurus* is straightforward, as described concisely in the following paragraphs and detailed in Annex B.

When a collection contains models represented by primitive instancing, the thesaurus for that collection will be composed by the set of primitives used by the objects in the collection. Since the terms of the thesaurus (3D shapes) are explicit in the collection, the same way words are explicit in text collections, building a *shape thesaurus* for such collection happens similarly as building a text thesaurus.

Therefore, for the collection $\mathcal{D}_p$, the thesaurus will contain the four primitives used to represent the models, while the inverted file will store the instances of each primitive in every model, as depicted in Figure 4.4. For each term, the inverted file has an occurrences list indicating in which models the term is used, *i.e.* the occurrences of each term. In these cases, both thesaurus and inverted file are trivial to construct.

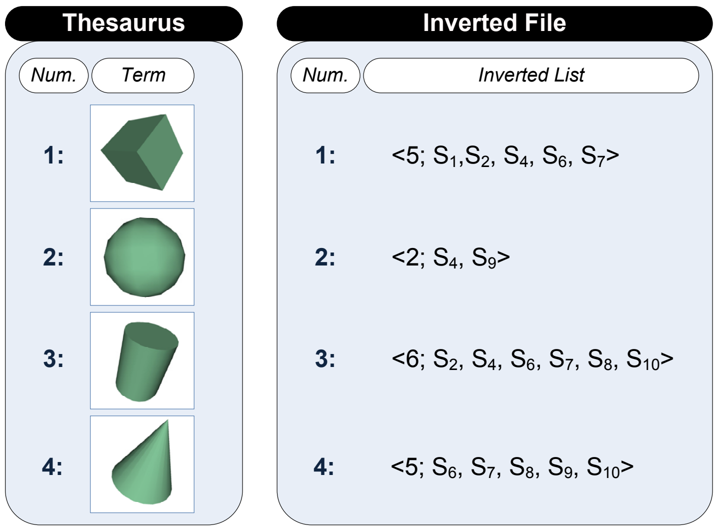

Thesaurus
Num.
Term
1:
2:
3:
4:
Inverted File
Num.
Inverted List
1: <5; S1,S2, S4, S6, S7>
2: <2; S4, S9>
3: <6; S2, S4, S6, S7, S8, S10>
4: <5; S6, S7, S8, S9, S10>

Figure 4.4: Shape thesaurus and inverted file for collection $\mathcal{D}_p$.

However, in our approach we want to retrieve objects in collections of CAD models independently of the modeling technique used to create and to represent them. Thus, we consider that these models use a boundary representation. Indeed, most 3D collections contain models represented as triangular meshes of the object surface, such as ESB [71] or Princeton Shape Benchmark (PSB) [115] collections. In this context, the determination of the shapes used to construct 3D models is not possible, at least in an automatic manner.

Therefore, instead of primitives shape the *shape thesaurus* will contain segments of the models in the collection. Nevertheless, while words in a text document, construction blocks in a CSG model, or primitive shapes in models represented by primitive instancing, are explicit, these segments are not explicitly identified within models. Thus, unlike in the example presented above, the construction of the *shape thesaurus* for a collection using boundary representation is not trivial, mainly because the terms should be properly identified without human intervention. In the previous chapter we described a method to automatically identify the parts that compose the models in a collection, creating a pool of such shapes. Indeed, we build our thesaurus using this *shape pool*.

4.3 Shape Pool Clustering

In text information retrieval the thesaurus contains terms that symbolize the words that comprise the vocabulary of the collection. Each of these terms usually correspond not to a single word but instead to different forms of the same word. Even though, the number of words in a lexicon of a web search engine is measured in millions [35]. While dealing with such a large number of terms in a text retrieval system is possible, handling a thesaurus of shapes of the same magnitude is hardly feasible. In the next subsections we explain how we overcome this problem.

4.3.1 Shape signature as term of the thesaurus

In our solution, each term in the thesaurus is represented by a shape signature, which corresponds to a feature vector produced by a shape descriptor. These feature vectors are points in multidimensional space, usually with very high dimensionality. For instance, in the CAS/HFP algorithm we used the SHA shape descriptor, which produces a feature vector with dimension $d = 544$. Considering such dimensionality, it is clear that a thesaurus of shapes could not grow as their textual counterparts.

Therefore, in our approach the thesaurus could not have more than a few thousand terms, or otherwise the search will take too much time. Moreover, if a large amount of features from every model in the collection is indexed directly in the thesaurus, the major benefit underlying our technique will be lost, since a search in the thesaurus will almost correspond to a search in the complete collection. The idea behind our approach is exactly to avoid that by creating a thesaurus with much entries than the sum of all segments extracted from the models in the collection.

As referred above, the construction of the thesaurus is based on the segments in the pool produced by the CAS algorithm. This shape pool is essentially a dataset in the multidimensional feature vector space containing the segment signatures. For instance, in Figure 4.5 we illustrate the *shape pool* $\mathcal{SP}_{eg}$ for the collection $\mathcal{D}_{eg}$ introduced in the previous chapter, giving the proper relevance to the shape signatures. The thesaurus for the collection $\mathcal{D}_{eg}$ is computed from $\mathcal{SP}_{eg}$, which is a dataset of shape signatures. The first step to determine the terms of the thesaurus is the computation of a partition for such dataset. This partition should contain the signatures clustered together according to the geometry of the corresponding segments. Thus, the next challenge is to devise an efficient way to cluster the shape signatures.

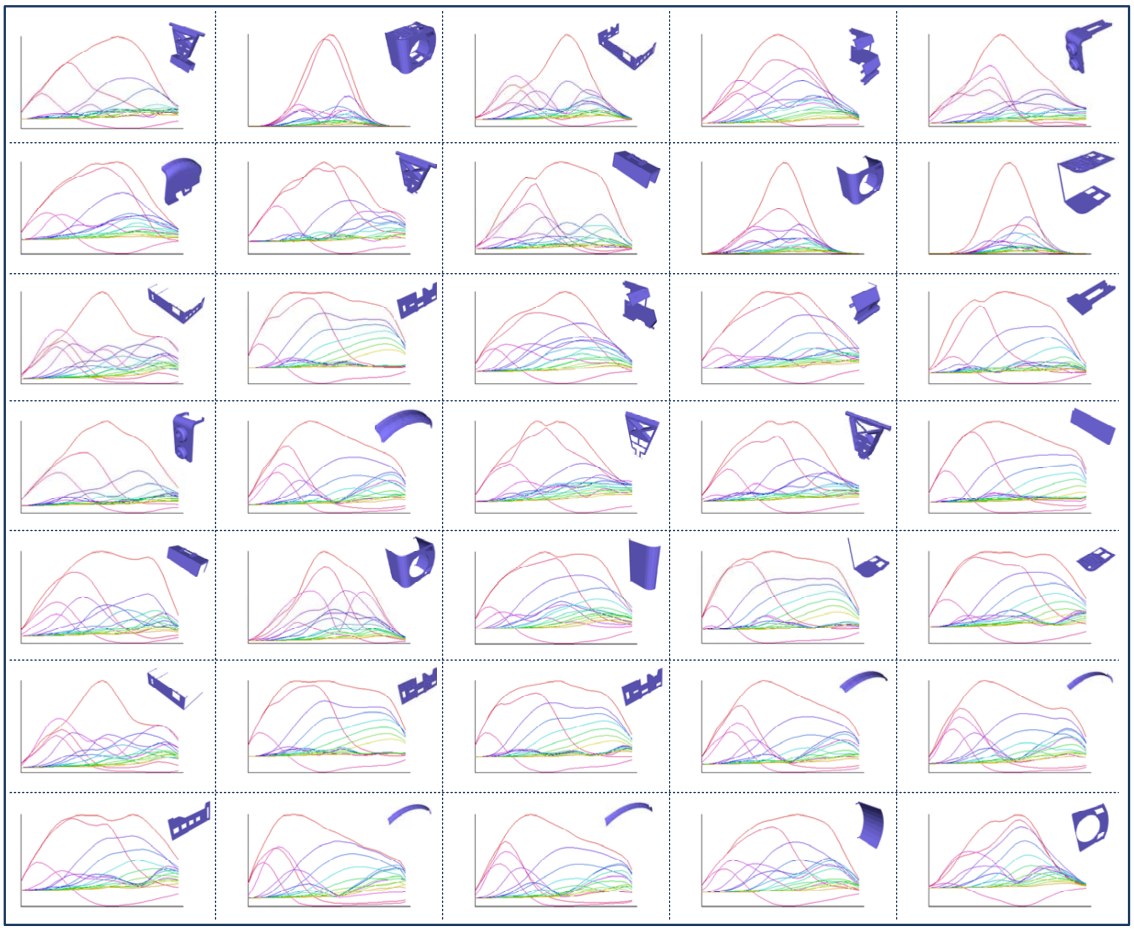


Figure 4.5: Shape pool $\mathcal{SP}_{eg}$ produced by CAS/HFP after processing collection $\mathcal{D}_{eg}$. In practice, it consists on a multidimensional dataset containing the signatures of the segments, where each point corresponds to a segment in the shape pool.

4.3.2 Clustering High-Dimensional Data

Due to its utility in a wide variety of fields, a large number of clustering algorithms are available, based on several distinct approaches. Nevertheless, the fundamental goal of any of these algorithms is to partition unlabeled data into groups in an unsupervised manner [1]. Depending on the approach, these groups, also called clusters, can be found according to a predefined number of expected groups or according to a given data *similarity threshold* within each group, among other less common approaches.

[1]Usually, when referring to supervised partition of data the more generic "classification" term is used. Indeed, data classification can be defined as the process of grouping these data into a set of groups or categories, independent of the method applied for that purpose. Note that under this definition, data can be classified even manually by humans [8]. While in supervised classification data labels and corresponding mapping functions are used, in unsupervised classification - clustering - no labeled data are available.

In any case, dataset clustering is a NP-complete problem and optimal solutions can only be found in exponential time. Thus, existing practical clustering solutions rely on sub-optimal algorithms, such as the iterative methods to determine a partition for a dataset. Among the iterative methods the k-means clustering is by far the most popular partitioning methodology [82], with a vast family of algorithms.

The basics of the k-means algorithm are as follows. Given a set ξ containing m points in $\mathbb{R}^n$, the goal of the algorithm is to determine a partition

$$\Pi = \bigcup_{i=1}^{k} \pi_i$$

for ξ, where each π_i represents a cluster of points in ξ, such that the centroids of these clusters minimize the mean squared distance from each point in π_i to the corresponding centroid $c(\pi_i) \in \mathbb{R}^n$.

This type of clustering is closely related to the Euclidean k-medians [12] which aims on minimizing the sum of distances to the nearest center. Thus, formally the centroid for each cluster is defined as

$$c(\pi_i) = arg\ min\{\sum_{a \in \pi_i} d(x, a), x \in \mathbb{R}^n\},$$

and the goal of the algorithm is to find a partition

$$\Pi^{min} = \bigcup_{i=1}^{k} \pi_i^{min}$$

that minimizes the value of a quality function $Q(\Pi)$ given by

$$Q(\Pi) = \sum_{i=1}^{k} Q(\pi_i),$$

where

$$Q\left(\pi_i\right) = \sum_{a \in \pi_i} d\left(c\left(\pi_i\right), a\right).$$

As stated above, there are no efficient solutions known for this minimization problem. Although Matousek [95] introduced an asymptotically efficient approximation for the k-means clustering problem, it is inappropriate for the present application, since our dataset contains a large number of high-dimensional points, not well handled by this approach. Thus we looked at alternative methods to perform clustering.

4.3.3 Lloyd's Algorithm

There are several algorithms that use heuristics to solve the k-means minimization problem, relying on a simple iterative approach that finds a locally minimal solution. One of the most popular is the algorithm proposed by Lloyd [89], initially for scalar data[2] and later generalized for datasets in multidimensional real numbers space.

Considering an initial set of centers $C = \{c_1, \cdots, c_k\}$, the Lloyd's algorithm starts by computing the neighborhood, $V(c_i)$, for each one of these points. This neighborhood consists on the points in the dataset ξ for which c_i is the nearest neighbor in C. In each iteration of the Lloyd's algorithm, every center point c_i is moved to the centroid of $V(c_i)$, producing a new set C' of centers. Then it starts again by recomputing the neighborhod for each new center. These iterations continues until a predefined criteria is fulfilled and a "good" partition is found.

Notice that, since the Lloyd's algorithm does not specify the initial distribution of centers, a set C of centers should be provided as input. Moreover, due to the nearest neighbor computation costs, a direct application of the Lloyd's algorithm for clustering large and high-dimensional data can be quite inefficient. Nevertheless, it is commonly used in this context as a postprocessing stage to improve the final results of other clustering algorithms.

4.3.4 Clustering Shape Signatures

In order to provide a tight control on the size of the thesaurus that will be created from the *Shape Pool* partition we choose to adopt a partitioning approach based on a predefined number of expected clusters. Therefore, in our approach we used a k-means clustering algorithm based on a combination of local search and Lloyd's algorithm [89] proposed by Kanungo *et al.* [76].

Indeed, despite its popularity, the Lloyd's algorithm can converge to a local minimum that is far from the optimal solution. One example is depicted in Figure 4.6, where a

[2]The algorithm originally proposed by Lloyd was devised for signal processing, namely for quantization in pulse-code modulation. Since it is basically a method to distribute evenly samples or objects in a given space, it was generalized for other applications. Namely, for the estimation of Voronoi diagrams and for high-dimensional dataset clustering.

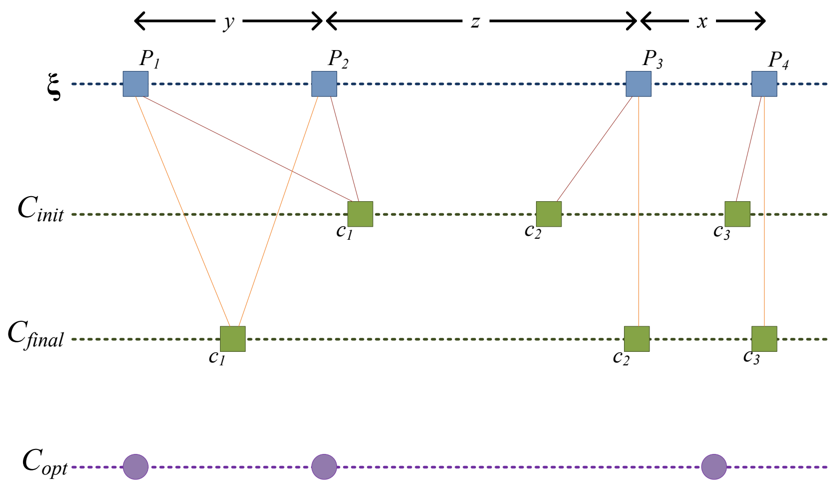


Figure 4.6: Example of a far from optimal partition produced by Lloyd's algorithm for a very simple dataset.

small dataset ξ in $\mathbb{R}$ is partitioned for $k = 3$ using a given initial centroids set C_{init}. The resulting center produced by the original heuristic, C_{final} is far from the optimal centroids center set, C_{opt}. This happens because in the first iteration the nearest centers for each point $P_i \in \xi$ are determined. Since $x < y < z$ the nearest neighbors will be estimated as pointed in the figure. Then, the centroids are recomputed accordingly. At the second iteration no changes are detected in nearest neighbors, causing the algorithm to stop, producing a far from optimal solution.

To overcome this drawback, Kanungo *et al.* [76] followed the approach used in approximation algorithms [12, 83] for a minimization problem. Based on the heuristic for k-medians proposed by Arya *et al.* [13] they devised a practical solution that produces a sub-optimal partition in a feasible time, even in high dimensions. This method is based on local search and uses a simple swapping process where the centers are swapped in and out of the set C. These swaps are random and are accepted if it provides better quality $Q(\Pi)$ to the partition Π, otherwise they are ignored. This simple technique provides a better solution than the original Lloyd's algorithm.

In our approach to cluster signatures in the *shape pool* we use an hybrid version of the k-means clustering algorithm proposed by Kanungo that combines the Lloyd's algorithm with center swapping. The technique performs some number of swaps followed by some number of iterations of Lloyd's algorithm. Additionally, it also includes a technique

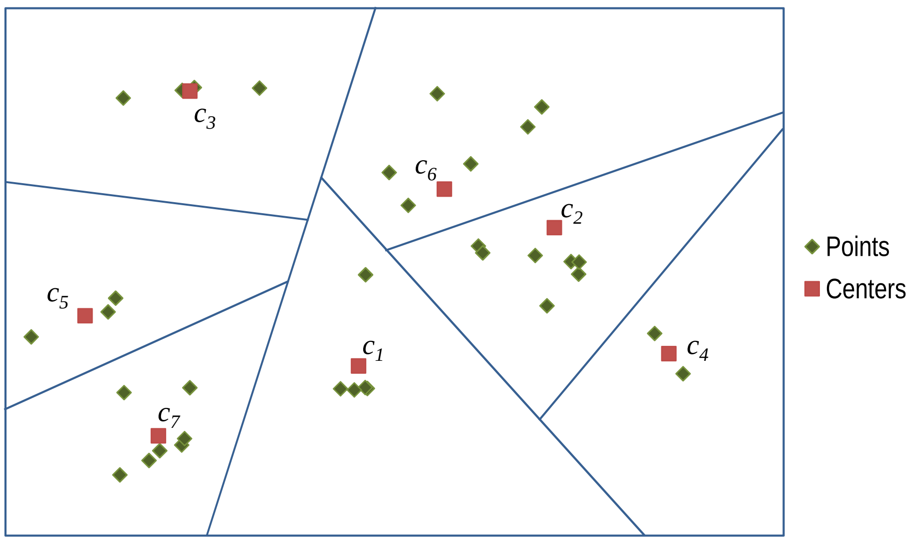


Figure 4.7: Partition of a 2D signature space corresponding to the shape pool $\mathcal{SP}_{eg}$ produced by CAS/HFP after processing collection $\mathcal{D}_{eg}$.

similar to simulated annealing [3] in order to avoid getting trapped in local minima.

To illustrate the method used we will only depict the signature as a two dimensional feature vector[4]. In Figure 4.7 we represent, as points in $\mathbb{R}^2$, the simplified signatures for the example shape pool $\mathcal{SP}_{eg}$ presented in previous chapter, and depict the corresponding space partition.

This partition was computed by applying the hybrid clustering algorithm to the two-dimensional dataset comprised by the reduced signatures of the segments in the shape pool. For this example we set $k = 7$ and obtained the seven cluster centers

$$C_{eg} = \{c_1, \cdots, c_7\},$$

which correspond to the partition presented in Figure 4.8, given by

$$\Pi_{eg} = \{\pi_1, \cdots, \pi_7\}.$$

[3]Simulated annealing is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global minimum of a given function in a large search space [80].

[4]In this particular example we constructed the feature vector using only the 105^{th} and 387^{th} dimensions of the complete SHA signature. These were carefully selected, considering the collection D_{eg}, in order to produce clustering results compatible to the partition of corresponding *shape pool* using the complete signatures.

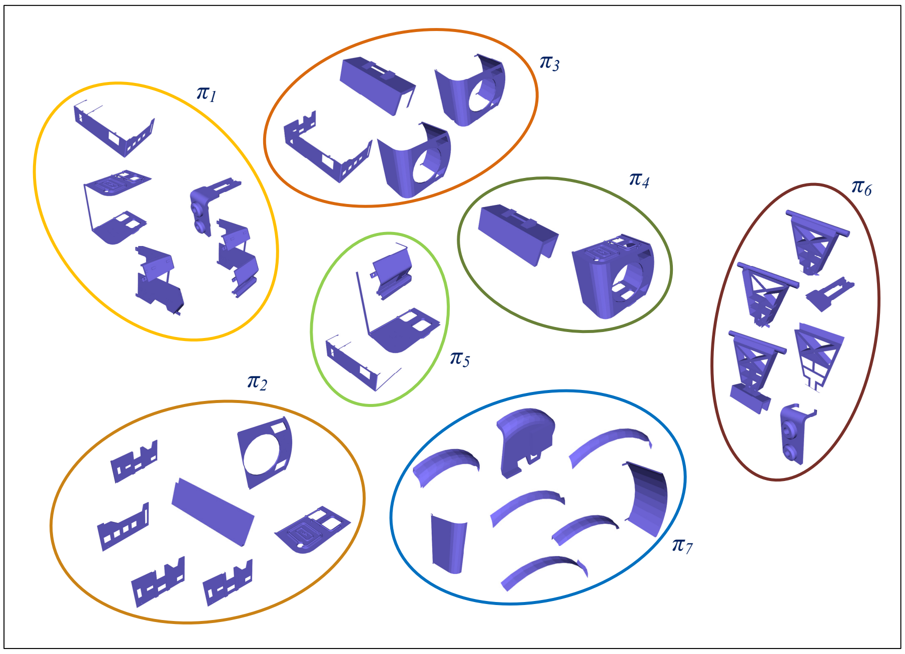


Figure 4.8: Seven clusters partition Π_{eg} produced from the shape pool $\mathcal{SP}_{eg}$ created by CAS/HFP after processing example collection $\mathcal{D}_{eg}$.

4.4 Thesaurus Construction

After having the segments from the *shape pool* grouped according to its geometrical similarity we are able to create our indexing structures, by identifying the ”3D words” (or terms) that compose the thesaurus.

According to Baeza-Yates and Ribeiro Neto [17] a thesaurus is a data structure composed of a pre-compiled list of important words in a given domain of knowledge and, for each word in this list, a list of related (synonym) words. In our approach, a 3D shape thesaurus is a list of terms that represent groups of similar shapes extracted from models in a given collection and, for each group in this list, a list of models that contains it. In the next subsections we explain the creation of the *shape thesaurus* and corresponding inverted file.

4.4.1 Creating the shape thesaurus

Starting from the partition computed from the *Shape Pool* with the hybrid segment clustering algorithm, we create the 3D shape thesaurus by considering the center of each cluster π_i as a basis for a term, T_i, constructing a list of segments that comprises the cluster and attributing a signature to each term. Conceptually, the term should correspond to a prototype for the grouped segments. In our approach we use as a prototype for a group of signatures the point in the center of that cluster. This way we can directly obtain the term signature from the partition without further processing. Thus, the term signature is the center c_i, of the corresponding cluster.

Formally, we define the shape thesaurus as the set

$$\mathcal{T} = \bigcup_{i=1}^{k} t_i,$$

where k is the size of the thesaurus, which should be equal to the number of clusters in the shape pool partition, and t_i is a term entry of the thesaurus. This term entry is given by the pair

$$t_i = \langle c_i, \pi_i \rangle.$$

To create a thesaurus from the results produced by the shape pool clustering, namely the centers set C and partition Π, we devised the algorithm BUILD-THESAURUS. This simple algorithm sweeps over the k clusters π_i in the partition Π, identifying for each one of these clusters the corresponding center and adding in each iteration the corresponding pair composed by the partition and the respective center. At the end it will return the shape thesaurus, which is basically a set containing these pairs.

The algorithm BUILD-THESAURUS runs in linear time with respect to the number of cluster in the partition, asymptotically, it executes in $O(k)$. Since the number of clusters, which corresponds to the desired size of the thesaurus, is controlled by the user when indexing the collection and it should not be a very high number in order to allow fast queries, this algorithm suits perfectly in the requirements of the classification process.

```
BUILD-THESAURUS($C, \Pi$)
1  $\mathcal{T} \leftarrow \emptyset$
2  for each $\pi_i$ in $\Pi$
3  do
4      $c_i \leftarrow c_j \in C : i = j$
5      Add $\langle c_i, \pi_i \rangle$ to $\mathcal{T}$
6
7  return $\mathcal{T}$
```

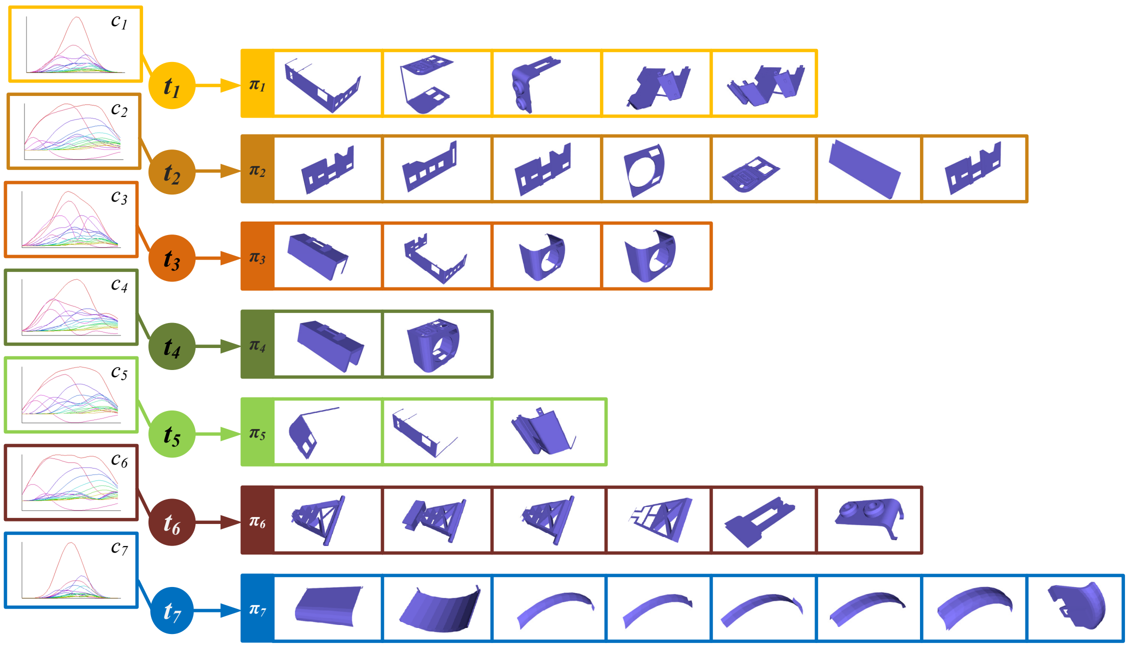


Figure 4.9: Shape thesaurus $\mathcal{T}_{eg}$ created from the partition Π_{eg} produced for the collection $\mathcal{D}_{eg}$.

The shape thesaurus $\mathcal{T}_{eg}$ produced for the collection $\mathcal{D}_{eg}$ introduced in the previous chapter using the algorithm BUILD-THESAURUS, is depicted in Figure 4.9. In this example a thesaurus with seven terms was constructed from the partition Π_{eg}.

To each term t_i is assigned a list containing the segments in the corresponding cluster. This list consists in the elements of the cluster, which means that it is essentially π_i. Additionally, to each term is attributed a multidimensional signature based on the SHA descriptors of the shapes in the cluster. As referred above, this signature corresponds to the center of the cluster that originated the respective term.

In a more formal manner, the shape thesaurus is specified as a set of terms, corresponding each term to a pair center-cluster. Thus, the thesaurus for the collection $\mathcal{D}_{eg}$ is given by

$$\mathcal{T}_{eg} = \{\langle c_1, \pi_1\rangle, \cdots, \langle c_7, \pi_7\rangle\},$$

where c_1 to c_7 correspond to the centers of clusters π_1 to π_7, respectively. These clusters are defined in partition Π_{eq} computed previously.

4.4.2 Building the inverted index

After creating the shape thesaurus, and following the concepts used in text information retrieval, we produced the inverted file that will support the retrieval process. The inverted file is an index composed of a vocabulary of terms and a list of occurrences of each particular term in models from the indexed collection. In practice, the inverted index necessary for 3D shape retrieval does not differ from its counterpart in text retrieval. The 3D shape inverted file contains a list of the terms from the thesaurus and, for each one, a list of models that contain segments belonging to the corresponding *shape pool* partition Π cluster π_i.

The formal definition of the inverted index is given as follows:

$$\mathcal{I} = \bigcup_{i=1}^{k} \langle t_i, \mathcal{O}_i \rangle,$$

where $\mathcal{O}_i$ is the occurrences list given by

$$\mathcal{O}_i = \{M \in \mathcal{D} : \exists_{S \in \pi_i} S \subseteq M\},$$

where $\mathcal{D}$ is the collection that originated the partition Π and the operator $\subseteq$ indicates if a shape is contained in another shape. More formally, considering two 3D shapes S_A and S_B the operator $\subseteq$ is defined by

$$S_A \subseteq S_B \Leftrightarrow S_A \text{ is a subpart of } S_B,$$

where we use a relaxed definition for subpart. In our work we consider that a shape S_A is a subpart of S_B if $S_A = S_B$ or if S_A belongs to the decomposition of S_B.

The algorithm devised to build the inverted index $\mathcal{I}$ for the collection $\mathcal{D}$, BUILD-INVERTED-INDEX, consists on a cycle over the terms t_i in the thesaurus $\mathcal{T}$ and, for each of these terms, it sweeps the segments in the corresponding cluster π_i, determining the models to which these segments belong. This algorithm receives as input the collection $\mathcal{D}$ and the corresponding shape thesaurus $\mathcal{T}$, and produces a structure $\mathcal{I}$ containing the inverted index.

```
BUILD-INVERTED-INDEX($\mathcal{D}, \mathcal{T}$)
 1  $\mathcal{I} \leftarrow \emptyset$
 2  for each $t = \langle c, \pi \rangle$ in $\mathcal{T}$
 3  do
 4      $\mathcal{O} \leftarrow \emptyset$
 5      for each $S_{i,node}$ in $\pi$
 6      do
 7          $M_i$ is model to which $S_{i,node}$ belongs
 8          Add Unique $M_i$ to $\mathcal{O}$
 9      Add $\langle t, \mathcal{O} \rangle$ to $\mathcal{I}$
10  return $\mathcal{I}$
```

Despite its two nested cycles, the BUILD-INVERTED-INDEX algorithm runs in linear time regarding the size of the shape pool. Indeed, while the outer cycle sweeps over the clusters, the inner one iterates over the segments in those clusters. Since each segment just exist in a single cluster, basically the algorithm visits all the segments in the shape pool to compute the inverted index. Therefore, asymptotically it runs in $O(Size(\mathcal{SP}))$, which might eventually led to high computation times if the shape pool grows too much.

Indeed, uncontrollable growth of the shape pool is an issue tackled during our research. This topic is properly addressed in Chapter 5, where we show that the shape pool does not grow behind a reasonable limit.

The inverted index $\mathcal{I}_{eg}$ relative to the collection $\mathcal{D}_{eg}$, constructed based on the thesaurus $\mathcal{T}_{eg}$ consists on a set of pairs, where each one contains a term of the thesaurus and the corresponding list of occurrences. This inverted index can be specified as follows:

$$\mathcal{I}_{eg} = \{\langle t_1, \mathcal{O}_1 \rangle, \cdots, \langle t_7, \mathcal{O}_7 \rangle\},$$

where

$$\begin{aligned}
&\mathcal{O}_1 = \{M_2, M_3, M_4, M_5\},\\
&\mathcal{O}_2 = \{M_2, M_3\},\\
&\mathcal{O}_3 = \{M_1, M_2, M_3\},\\
&\mathcal{O}_4 = \{M_2\},\\
&\mathcal{O}_5 = \{M_2, M_3, M_5\},\\
&\mathcal{O}_6 = \{M_1, M_5\}, \text{ and}\\
&\mathcal{O}_7 = \{M_2, M_6\}.
\end{aligned}$$

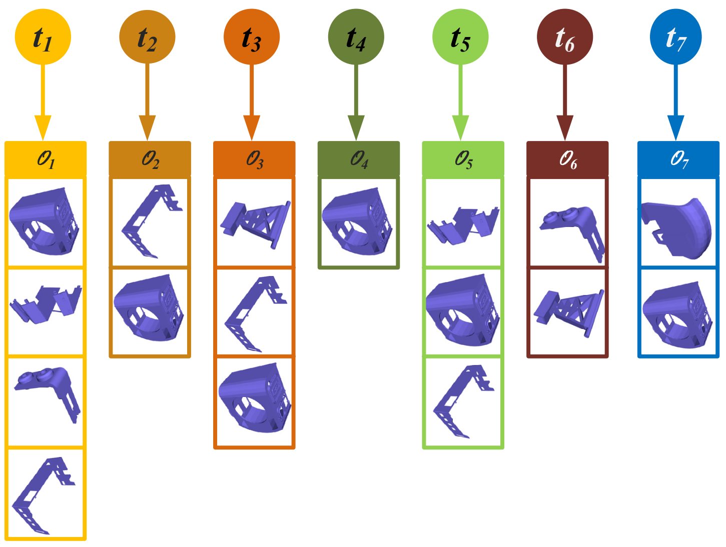


Figure 4.10: Inverted index $\mathcal{I}_{eg}$ created for the collection $\mathcal{D}_{eg}$ based on the corresponding thesaurus $\mathcal{T}_{eg}$.

This inverted index,illustrated in Figure 4.10, was constructed from the terms in the thesaurus, which are indexed according to the decomposition of the models in the collection $\mathcal{D}_{eg}$ and corresponding partition Π_{eg} produced by the shape clustering algorithm. To that end, for each term t_i in the thesaurus, we swept the corresponding segment list πeg, identified the models to which every segment belongs and created with them a list of occurrences $\mathcal{O}_{eg}$ assigned to the respective term t_i in the inverted index.

The creation of the shape thesaurus and corresponding inverted index, represent the conclusion of the collection classification process. Indeed, the construction of such structures are one of the main goals of the our research. From these structures it will be possible to perform partial queries with part-in-whole matching.

4.5 Shape Retrieval

With the collection properly indexed it is now possible to execute partial queries. The retrieval process consists on a three-stage pipeline that receives as query a 3D shape and produces a list of models that satisfy the given query, as depicted in Figure 4.11. This is as modular process since each stage works independently of the other. Thus, it is easy to substitute the methods used in these stages by others, ensuring they receive a given input and produce an expected output.

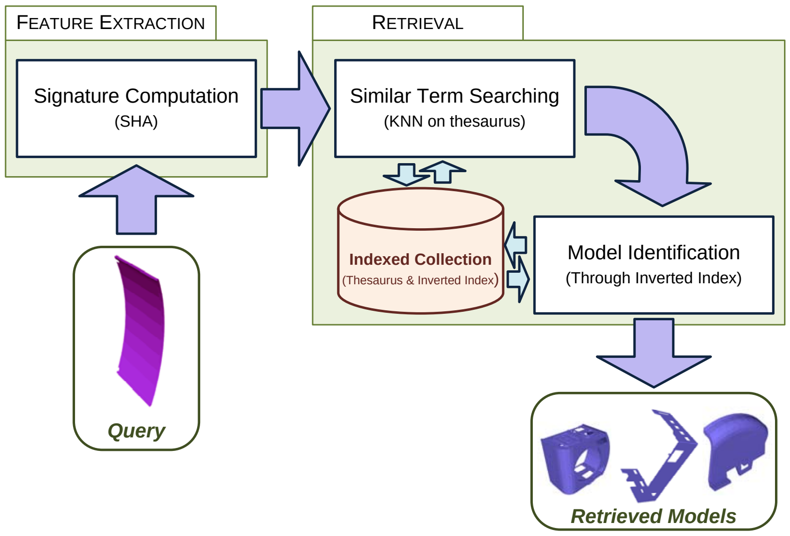


Figure 4.11: Shape retrieval pipeline.

Basically, the first stage produces a feature vector that describes the query, the second uses that feature vector to produce a list of terms similar to the query, and the third uses that list to determine the models that satisfy the query with part-in-whole matching.

More specifically, in our current approach we forged these stages in order to comply with the options we made and corresponding set-up specified in previous chapter. Namely, regarding the shape description technique and multi-dimensional space search algorithm. In next subsections we describe in detail each of these stages.

4.5.1 Signature Computation

The first step of the retrieval process consists of extracting the geometric features of the query shape by computing its signature. To that end we use the same shape descriptor used to calculate segment signatures during the thesaurus construction, the rotation invariant spherical harmonics (SHA). As a result of feature extraction, we obtain the signature $\mathcal{FV}_Q$ of the query shape Q, as given by

$$FV_Q = SHA(Q),$$

where $SHA(Q)$ estimates a signature for shape Q using the rotation invariant spherical harmonics, proposed by Kazhdan *et al.* [78]. An example of such feature vector is depicted in Figure 4.12, along with the corresponding query shape.

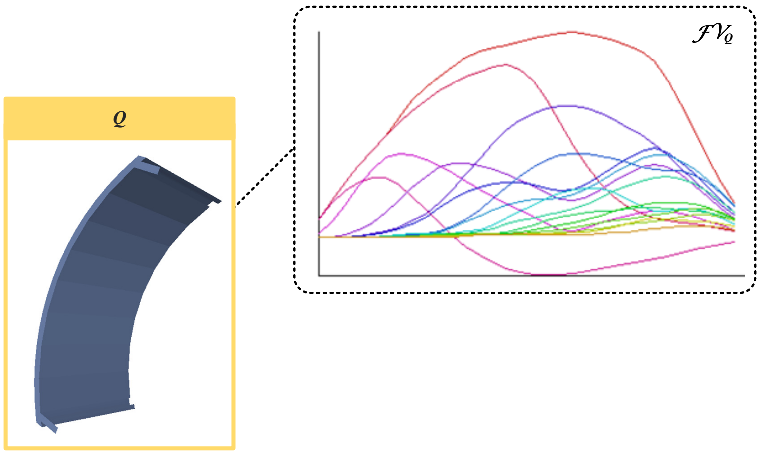


Figure 4.12: Query Q and corresponding shape signature $\mathcal{FV}_Q$.

4.5.2 Similar Term Searching

In order to retrieve models partially similar to the query, the nearest neighbors of this feature vector in the signature space of the shape thesaurus must be found. To that end, the second stage consists on a k-NN search performed on the shape thesaurus. In the current implementation of our solution a linear search algorithm is used, but to improve the retrieval efficiency a faster sub-linear approach [5] can be used. Nevertheless, since the number of terms in the thesarus is controlled and within a reasonable size, the search in this dataset will be fast, even when using high dimensionality signatures, as the SHA feature vector.

Independently of the algorithm efficiency and shape descriptor used, any k-NN search will return a set ξ containing the k terms more similar to the query shape, as given by

$$\xi = kNN(\mathcal{T}, \mathcal{FV}_Q, k),$$

where k is the number of neighbors to retrieve, a pre-defined setting of the retrieval system, and $kNN(\alpha, \beta, \kappa)$ consists on a k-NN search in the dataset α that returns the κ nearest neighbors of point β, as described in Section 3.4.1.

[5]Indeed, in our research work we choose to use a linear approach not only due to its simplicity but also because more efficient search approaches rely on usually complex datastructures, such as the kd-tree [20], which demand additional memory requirements when using a high dimensionality dataset. Since we intend to make a proof-of-concept implementation of our approach and the behaviour of linear k-NN satisfies our purposes, we prefer to avoid additional complexity in exchange for some efficiency.

4.5.3 Matching Models Identification

From the list ξ of terms produced by the k-NN algorithm is quite straightforward to find the models that partially satisfy the query. Using the inverted index, the models assigned to the resulting terms are gathered and ordered, thus producing a list of partially similar models. In the present implementation this process executes in linear time since it consists on a single swap through the inverted index terms. Nevertheless, this process can be executed on constant time if structures that map directly the terms in the thesaurus with the entries in the inverted index were used. This option was not followed because our main concern was to guarantee independence among all components in order to facilitate changes on the algorithms.

Therefore, our current solution for retrieval of models in a collection indexed using the *shape thesaurus* is described by the algorithm RETRIEVE-MODEL. It receives as input a query shape Q, a thesaurus $\mathcal{T}$ and the corresponding inverted index $\mathcal{I}$. As a result it produces a list $\mathcal{R}$ of models that are partially similar with the given query.

```
RETRIEVE-MODEL($Q, \mathcal{T}, \mathcal{I}$)
 1  $\mathcal{R} \leftarrow$ empty set of search results
 2  $\mathcal{FV}_Q \leftarrow=$ COMPUTE-SHAPE-DESCRIPTOR($Q$)
 3  $\xi \leftarrow kNN(\mathcal{T}, \mathcal{FV}_Q, k)$
 4  for each $t$ in $\xi$
 5  do
 6      $\mathcal{O} \leftarrow$ GET-OCCURRENCES-LIST($t, \mathcal{I}$)
 7      for each $M$ in $\mathcal{O}$
 8      do
 9          Add Unique $M$ to $\mathcal{R}$
10
11  return $\mathcal{R}$
```

Similarly to its homonym in the CAS/HFP algorithm, the method COMPUTE-SHAPE-DESCRIPTOR computes a feature vector for the given shape. In the present approach, both use the rotation invariant spherical harmonics, as stated above. Furthermore, the method GET-OCCURRENCES-LIST returns the list of occurrences of a given term in a given inverted index.

Considering that the signature computation depends only of the complexity of the submitted query, the similar term search in the thesaurus runs on linear (or eventually sub-linear) time, $O(N)$, where N is the thesaurus size, and identifying models using the inverted index can be accomplished in linear (or constant) time, $O(K)$ or $O(1)$, it should be clear to the reader that the retrieval time efficiency does not depend directly on the size of the collection, but rather on the size of the thesaurus and query complexity.

4.6 Summary

Our approach to retrieval of three-dimensional models using part-in-whole queries follows a path distinct from those proposed in other 3D shape retrieval solutions. We transposed to the 3D context the basic concepts of text information retrieval. Namely the concept of a thesaurus of terms: a lexicon in text collections, now a shape thesaurus in collections of three-dimensional models. This chapter introduces not only the shape thesaurus and related data structures but also a framework to support the collection classification and shape retrieval using our methods.

The framework for thesaurus-based 3D shape retrieval with partial queries is divided in two main components. The classification component processes and indexes the models in the collection using a shape thesaurus and corresponding inverted index. The retrieval component accesses the indexed collection to find a set of models partially similar to a given query.

To classify the collection, three distinct steps are necessary. The first and more important step consists on decomposing the objects in the collection, identifying the segments that comprises them. From this decomposition is produced a structure called *shape pool* that contains all segments of all models in the collection. The algorithm we devised to carry out this decomposition was called Context-aware Segmentation (CAS), and is the kernel of our research (see the previous chapter).

The *shape pool* produced by the CAS algorithm plays an important role in the classification process, since it is the source upon which the shape thesaurus is built. To that end, in the second step of the classification process, the segments in the pool are clustered together according to their feature vectors. The partition of the shape pool is obtained using a k-means clustering algorithm which combines local search techniques with Lloyd's clustering algorithm.

Since each cluster of the *shape pool* partition corresponds to a term in the *shape thesaurus*, the construction of the thesaurus is straightforward. Indeed, to build the thesaurus it is only necessary to create a set of terms, which correspond to the cluster centers, and assign to each term the segments that constitute the respective cluster. This task is performed by the third step of the classification component, along with the creation of the corresponding inverted index.

The inverted index contains, for each term in the thesaurus, the list of models that contain shape segments associated with that term. Since the segments in the *shape pool* contain information indicating from which model they belong, it is not complicated to build the inverted index. Indeed, it can be done in linear time regarding the *shape pool* size, by sweeping through the terms in the thesaurus and for each term construct a list of occurrences, *i.e.* a list containing the models in which the segments associated to that term occurs.

Having the shape thesaurus and corresponding inverted index properly built, it is possible to perform part-in-whole queries on the collection. In the devised framework, the retrieval component is a three-staged pipeline that receives a 3D shape as a query and returns a list of partially similar objects.

In the first stage of the retrieval component a feature vector is extracted from the query shape. In the second stage this feature vector is used as a query to a k-NN search in the dataset composed by the signatures of the terms in the thesaurus. The resulting nearest neighbors correspond to terms in the thesaurus that are similar to the query. Finally, the query results are constructed simply by consulting the inverted index occurrences lists for the terms returned by the k-NN search.

With our solution, a query is executed in approximately linear time regarding the thesaurus size. Indeed, this corresponds to the k-NN in the thesaurus signature space, whose execution consumes the larger slice of the query time. On the other hand, the classification is a time consuming task. However, this is not an issue since classification is supposed to be done sparsely and in batch mode.

5
Experimental Results

During our research we performed a wide set of experiments aiming a multitude of objectives. Besides those whose goal was to validate our ideas, we also did some experiments to help defining the research path based on well-grounded decisions.

After identifying the type of 3D collections to work with and selecting representative benchmarks for that type, we studied these benchmarks regarding model complexity. Results from this study allowed us to adapt our classification strategy to suit the desired purposes. In a different perspective, but with the same goal, we simulated the behavior of our decomposition algorithm, in the worst case scenarios.

To evaluate and validate our approach we performed several experiments. These experiments focused both on the classification mechanism as a whole, and on in its components separately. To that end, we developed a set of prototypes, described in Annex C and, using a selected set of benchmark collections, we tested our approach to thesaurus-based 3D shape retrieval with partial queries.

The experiments directly related to our research objectives include the study of the behavior of the Collection-Aware Segmentation (CAS) algorithm, the segment clustering method and the thesaurus building technique. Ultimately, we tested the retrieval process by performing partial and complete queries with a set of 3D object collections.

In the remainder of this chapter we describe the most relevant experiments performed during our research, presenting and discussing the observed results. All experiments were performed on a Hewlett-Packard workstation, model xw4200, equipped with a $3\ GHz$ Intel® Pentium 4 processor, $1\ GB$ of memory, and a Windows XP operating system.

5.1 Mesh Complexity in ESB and PSB

While planning our approach to 3D shape retrieval we needed some information about the complexity of models that compose 3D collections. However, if we consider this complexity as the perceivable visual complexity of a 3D shape, such measurement is notoriously difficult [61]. One approach to obtain an approximate quantification of 3D objects complexity is to rely on the complexity of the representation used to describe the object. In the context of our research, the models are represented by polygonal meshes.

Therefore, although the number of faces in a polygonal mesh might not be proportional to model complexity, we considered for our purposes that this measure quantifies the complexity of the shape. To that end we assumed that meshes for more complex objects contain more polygons, while meshes with fewer triangles represent simpler models. Indeed, a simpler object might contain more faces than a more complex one, depending on the tessellation. However, this is not common in the collections we tested and the face count is easy and fast to estimate.

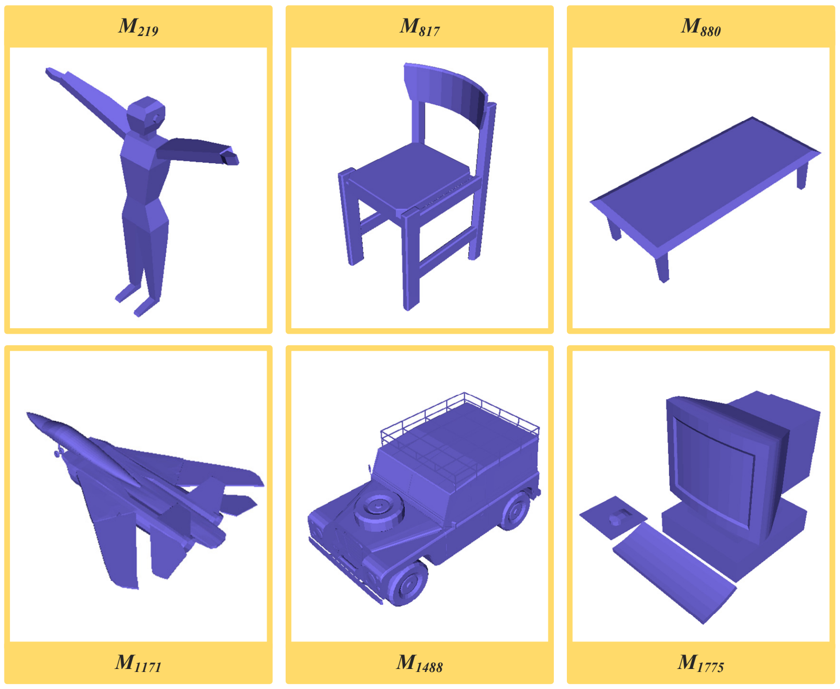


Figure 5.1: Six models sampled from the PSB collection.

5.1.1 Experiment Description

As referred in the previous chapters, we focused our work on collections of engineering models and used for research purposes the engineering shape benchmark (ESB) collection [71]. However, in order to have additional information regarding generic models rather than only CAD models from ESB, we also analyzed the complexity of models in the Princeton shape benchmark (PSB) collection [56].

Six models randomly sampled from this collection are depicted in Figure 5.1, together with the respective model names. This sample allow us to minimally illustrate the diversity of models that can be found in this collection, which led us to include it in this experiment.

The model complexity analysis is quite straightforward since it consists on counting the number of mesh polygons of each model in both collections and then analyze the gathered values. To perform this count we developed a small prototype, described in Annex C that sweeps over the collections and recollects this information.

From the gathered information, we obtained some interesting results, namely considering the distribution of complexity within these collections. To that end, we grouped the models according to its polygon count, p, within five intervals with logarithmic width, where the limits l_i of these intervals are given by $l_i = 10^i$, with $i = 1, \cdots, 5$. Then, we counted the number m of models that contain the quantity of polygons expressed in each intervals for both collections and obtained the results depicted in Table 5.1.

	ESB		PSB	
		m		m
$p < 10$	(0.00%)	0	(0.00%)	0
$10 \leq p < 10^2$	(5.86%)	50	(3.80%)	69
$10^2 \leq p < 10^3$	(32.24%)	275	(32.64%)	592
$10^3 \leq p < 10^4$	(56.62%)	483	(43.05%)	781
$10^4 \leq p < 10^5$	(5.28%)	45	(19.79%)	359
$p \geq 10^5$	(0.00%)	0	(0.72%)	13

Table 5.1: Distribution of mesh complexity in ESB and PSB collections. For each interval is shown the number m of models whose polygon count falls into that interval.

5.1.2 Analysis and Results

The analysis of the model complexity allow us to conclude that, in both collections, a vast majority of models has more than one thousand faces and, within these objects, most of them have less than ten thousand faces. Besides a noticeable number of simple models, *i.e.* with less than one thousand models, we also found a relevant number of models with more than ten thousand polygons, especially in PSB, since in ESB just around five percent of models fall into that classification. These conclusions are easily intelligible in the charts illustrated on Figure 5.2, which depict the distribution of the model complexity in each collection.

Additionally, we computed the complexity histograms for both collections, depicted in Figure 5.3. These histograms reinforce the conclusions taken based on complexity analysis. By simply looking at the histograms it is clear that most models contain between one and six thousand faces, with a peak around the one thousand polygons count. Peculiarly, both histograms have their peaks on the same intervals. This data allow us to say that, regarding time and space complexity analysis of our algorithm, we can safely consider that models to be processed contain a few thousand polygons each. From this conclusion we procceed to test the behaviour of the CAS/HFP decomposition algorithm.

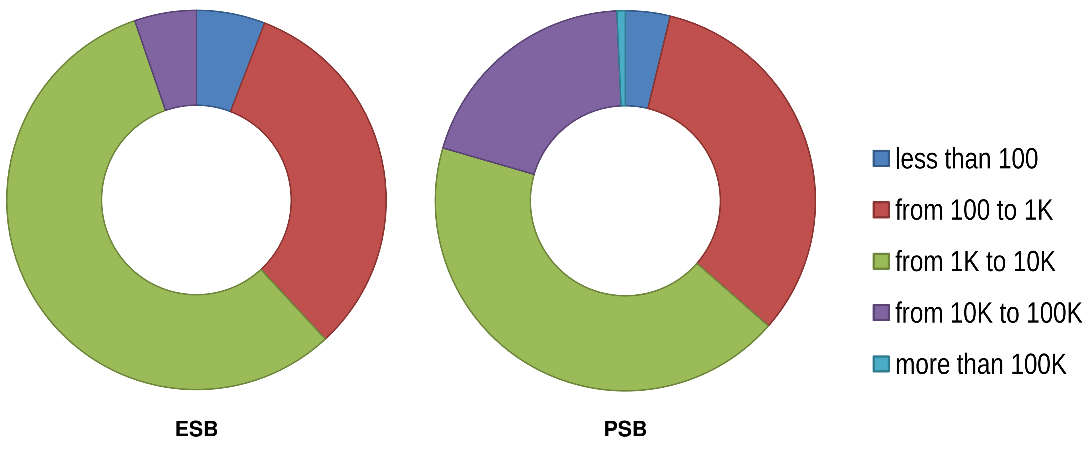


Figure 5.2: Distribution of mesh complexity in both benchmark collections, according to polygon count per model.

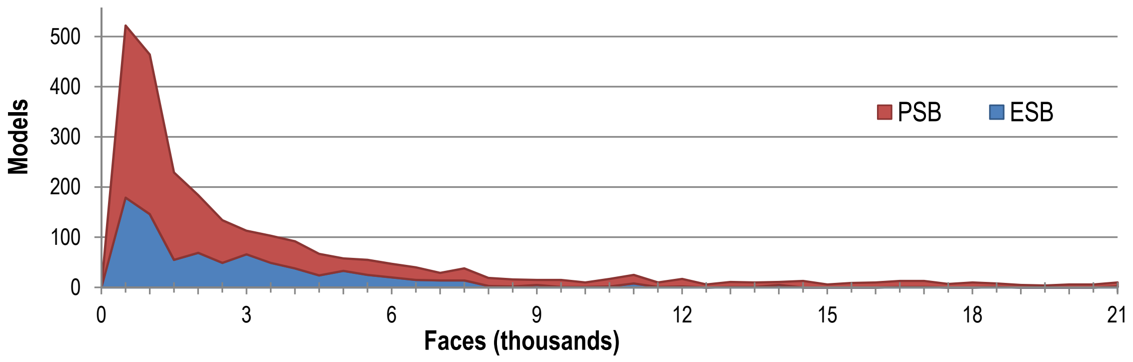


Figure 5.3: Mesh complexity histogram for both benchmark collections.

5.2 Worst Case Simulation

After testing our approach with a small collection and obtaining positive feedback, we simulated the CAS/HFP decomposition algorithm behaviour in a worst case situation. Therefore, we consider that all elements of the collection have more than two thousand polygons and there are no similarities between the first three hundred segments extracted from each model during segmentation by HFP algorithm.

Although the first assumption is perfectly acceptable, as shown in previous section, the whole scenario is highly improbable in a model collections unless is used an extremely low value for the *similarity threshold* (σ), *i.e.* to be considered similar two segments should be basically equal, or an illogically high value for the *similar count threshold* (τ), *i.e.* to be considered not decomposable a segment should be similar to all the other segments in the collection. Nevertheless, we simulated such unreal condition.

5.2.1 Experiment Description

Since in our approach we are using the ESB, we set up our simulation according to this collection cardinality. Thus we simulated two scenarios, one with the approximate size of the smaller cluster [1] of the ESB and other with the approximate size of the whole ESB collection.

The worst case simulation experiment focused on the major issues faced by our decomposition algorithm, execution time and memory space requirements in a worst case

[1]The ESB collection is divided into three clusters (or categories), described in Section 5.3.

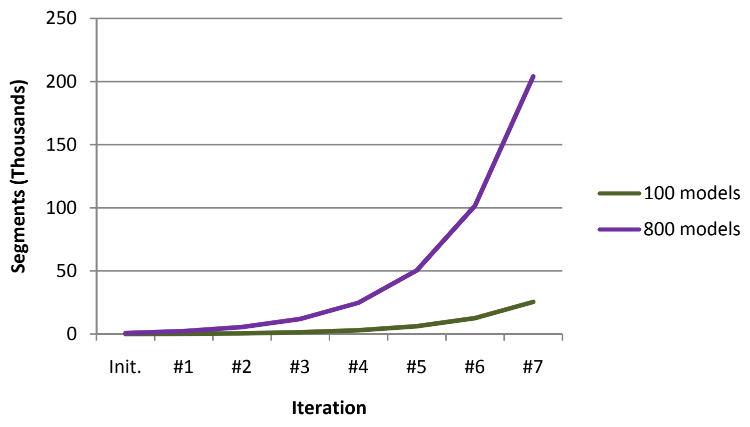


Figure 5.4: *Shape Pool* growth in worst case simulations.

situation. However, the existing benchmark collections does not comply with the worst case scenario we intended to study. Therefore, we considered two hypothetical collections with one hundred and eight hundred 3D models respectively and simulated the algorithm behavior with $\sigma \approx 0.0$ and $\tau \approx 1.0$. During the simulations we measured the memory used and time spent to process each collection.

5.2.2 Shape Pool Growth

The results obtained in these simulations were not surprising. As expected, the number of segments in the *shape pool* grew exponentially, as illustrated in Figure 5.4. Such growth is easily justified by two facts:

1. in each interaction of the CAS/HFP algorithm every decomposable segment in the shape pool originates two more segments;

2. with the σ and τ values used in the simulations, practically all segments are decomposable.

Indeed, the observed behavior of the *shape pool* reflects exactly that. After the initialization, it contains exactly the same number of segments as models in the collection, but at each iteration the number of new segments doubles. Therefore, from this we can state that the size of the *shape pool* for such simulation is given by:

$$Size(\mathcal{SP}) = Size(\mathcal{D}) \times (2^{i+1} - 1),$$

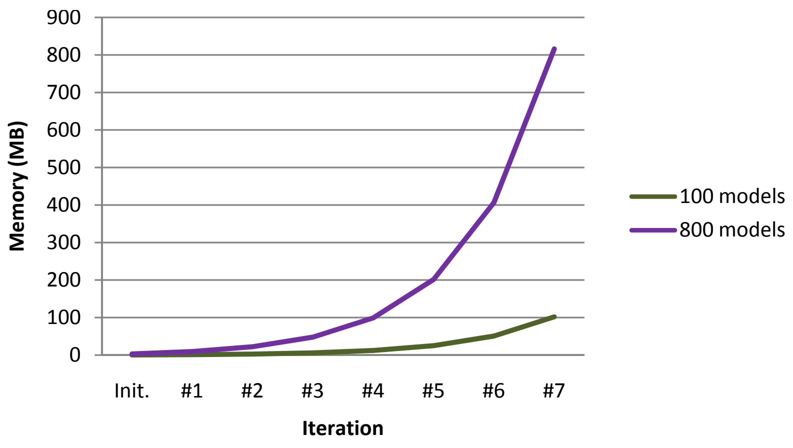


Figure 5.5: Memory required to store the *shape pool* in worst case simulations.

where i corresponds to the iteration count, considering that $i = 0$ refers to the initialization phase.

Naturally, the memory required to store segment information grows similarly to the shape pool. Computing the rotation invariant spherical harmonics (SHA) descriptor according to authors suggestion, its signature is represented by a feature vector with dimension $d = 544$. This means that each signature requires slightly more than 2KB of memory. As explained previously, besides the signature there are other information to be stored, such as the reference to the originating model and reference to the corresponding node in the HSM tree.

In practice each pair segment-signature uses approximately 4KB of memory. From our simulation we observed that, at seventh iteration, more than eight hundred megabytes of memory are necessary just to store segment information, as depicted in Figure 5.5. While such values does not seem unreasonably high, its growth can compromise the scalability of our decomposition algorithm.

5.2.3 Decomposability Determination Time

Another topic studied during this simulation is the time required to identify if a segment should be further decomposed. Indeed, the decomposability determination is based on a neighbor search and sub-linear algorithms for this purpose are widely-known. However, as described in Section 3.4, in present implementation of the CAS/HFP algorithm we used an adapted version of the k-NN linear search, called k-within range (k-WR).

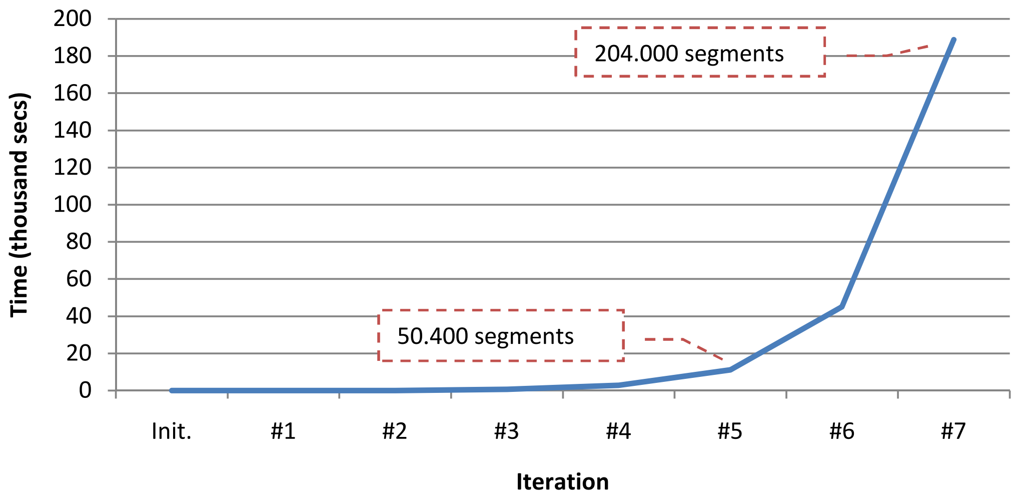


Figure 5.6: Decomposability determination for a single segment in eight hundred models collection.

This version outperforms k-NN, allowing decomposability identification of a segment in reasonable time, but runs also in linear time with respect to dataset size. Thus, if the size of the *shape pool* grows exponentially we will face unacceptable computation times.

For instance, at the seventh iteration of CAS/HFP at the worst case simulation the *shape pool* has more than two hundred thousand segments. Determining at this point the decomposability of a single segment takes less than one second, but considering that this operation should be repeated for all segments, it might take more than fifty hours to determine which of these are decomposable, as illustrated in Figure 5.6. Even if this processing time might eventually seem acceptable for batch processing, it will take too much time to analyze an exponentially larger *shape pool*.

We are aware that the time complexity issue identified above should be addressed in a near future in order to make our approach scalable to very large collections of 3D models. Indeed, we are already investigating some techniques to avoid the exponential growth of the *shape pool*, along with more efficient algorithms for decomposability determination. However, even for the worst case simulations, the results we obtained are acceptable for testing the proposed approach with the *shape pool* size below fifty thousand segments. Therefore, we still use the k-WR algorithm in the remaining of our tests of the CAS/HFP algorithm.

5.3 Decomposing a Benchmark Collection

As expected, when applying the CAS/HFP algorithm with reasonable σ and τ parameters to a benchmark collection, the results are quite different from the suggested by the worst case simulation. For this experiment we used as test collection the ESB [71], described in Section 2.1.3. This collection is divided into three clusters, as presented in Table 5.2.

	Name	*Models*
FTC	Flat-Thin Components	105
RCP	Rectangular Cubic Prism	273
SoR	Solid of Revolution	475

Table 5.2: Clusters of ESB collection

5.3.1 Experiment Description

In this experiment we processed the clusters from ESB with the CAS/HFP decomposition algorithm, first separately and then together, *i.e.* the whole collection. The decomposition set up used for all tests is described in Table 5.3, where the values for the algorithm parameters are shown. These values were determined based on the study presented in Section 3.6 and from the analysis of results obtained by several tests made throughout our research.

During the ESB collection decomposition experiment we focused our attention on time and memory required to decompose the collections, since this was a major issue identified during the development of our approach. To that end, we measured the time spent in each step of the algorithm but also the *shape pool* size, which is directly proportional to the required memory.

	Setting	*Value*
Similarity threshold	σ	0.2
Similar count threshold	τ	0.01
Iteration cut-off	λ	10

Table 5.3: CAS/HFP settings for benchmark collection decomposition experiment.

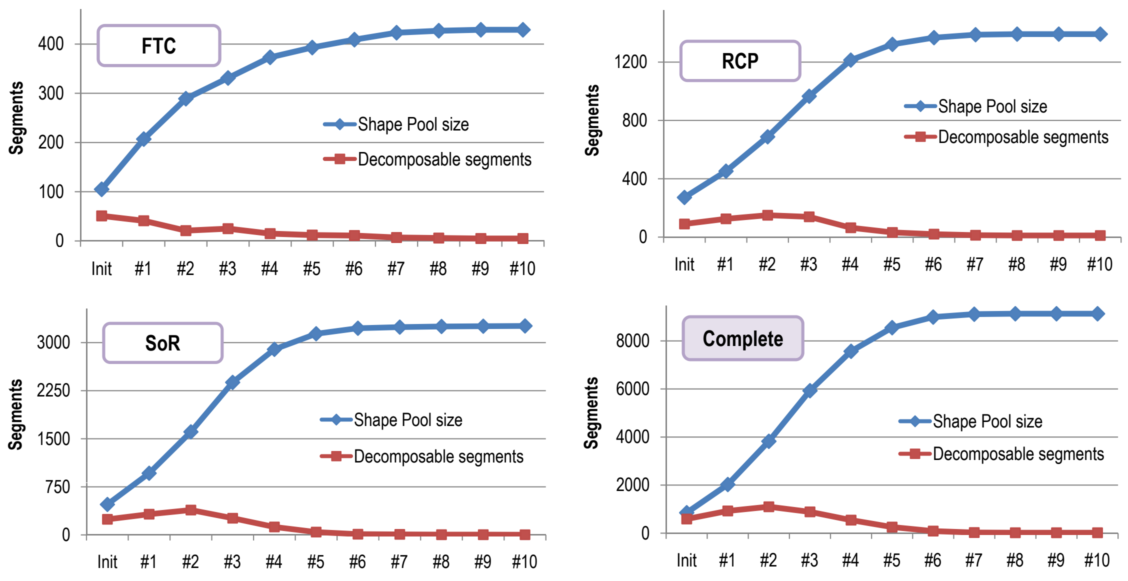


Figure 5.7: Shape pool growth

5.3.2 Shape Pool Growth

The observed behavior was very encouraging, since in all cases the *shape pool* growth was far below the measured in the worst case simulations. More important, the growth rate of the *shape pool* starts decreasing soon after the initialization, which means that the *shape pool* does not increase to impractical size.

The shape pool growths measured in these tests are depicted in Figure **??**. Here is possible to observe that the number of segments in the *shape pool* stabilize around the sixth iteration and that the *shape pool* growth rate starts decreasing after the second iteration.

Exceptionally, the above referred observation is not exact when processing the FTC cluster. This happens mainly due to its small size. However, although the *shape pool* size took longer to stabilize in this case, the overall behavior is the same.

Hence, for the remaining clusters (RCP and SoR), as well as for the whole collection, the shape pool size never grows above a very clear ceiling. This ceiling is not defined or forced by the algorithm. Instead it is a natural consequence of the fact that, for a given similarity threshold, there are a certain number of segments above which no more new dissimilar shapes are found, thus no more segments are identified as decomposable.

Indeed, we observed similar *shape pool* growth behavior in all other tests performed until now with this and other collections, which is by itself a very positive outcome. Considering this behavior and the results obtained in the worst case simulations, we reasoned that the proposed algorithm will be able to produce valid results in suitable time and consuming a reasonable amount of memory.

5.3.3 Used Memory

From the measurements made during this experiment we verified that the memory required to process the collection is within reasonable values in all cases. As formulated in Section 3.3, in our approach a single signature is assigned to each segment in the *shape pool*. Thus, *the shape pool* basically consists on a dataset in the signature space, with the same dimensionality, containing the signatures of all segments. Thus it is trivial to conclude that the memory required to store the signature space is directly proportional to the size of the *shape pool* and to the dimension of the signature.

As explained in Section 3, the CAS/HFP algorithm relies on the rotation invariant spherical harmonics (SHA) introduced by Michael Kazhdan *et al.* [78] to numerically describe three-dimensional shapes. This entails that the segments stored in the *shape pool* are represented by the SHA descriptor. The corresponding feature vector needs slightly more than 2KB of memory and is used as a signature for the segment. Since additional

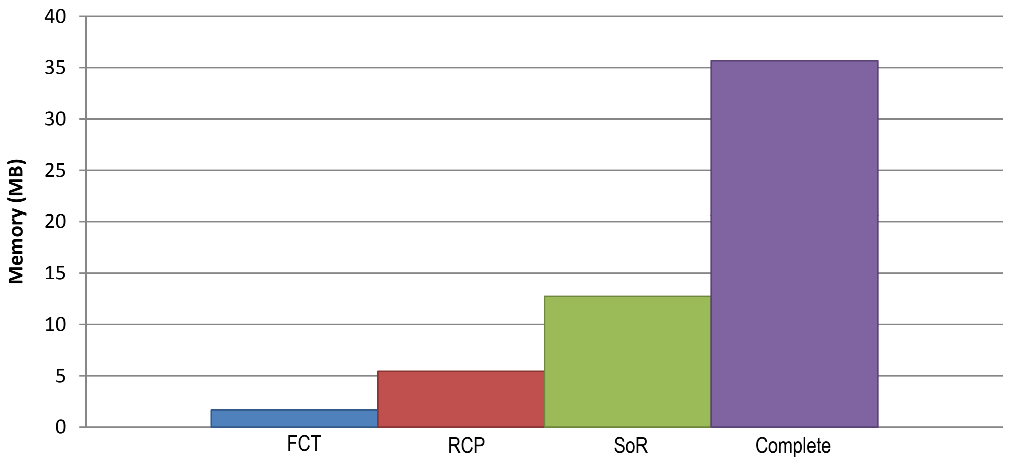


Figure 5.8: Memory used by the *shape pool* produced by the CAS/HFP algorithm after processing each ESB cluster and the complete ESB collection.

information is necessary for a segment, besides its signature, in practice each item in the *shape pool* requires 4KB.

Thus, in this experiment the memory used for the *shape pool* was always naturally low, as illustrated in Figure 5.8. This chart depicts the memory used by the *shape pool* produced at the end of the CAS/HFP algorithm when processing the ESB clusters as well as the complete collection. For instance, considering the decomposition of the complete ESB collection, in previous section we showed that the corresponding shape pool stabilized with around nine thousand segments, which explain the use of around 35MB of memory to store it.

Indeed, while processing benchmark collections, the memory requirements were always quite acceptable. Eventually, when processing a very large collection containing tens of thousand models, this might be an issue. However, the main challenge we faced during our research regards the execution time of the CAS/HFP algorithm.

5.3.4 Execution Time

The CAS/HFP algorithm is divided into two stages. The initialization stage, where the HSM trees for all models in the collection are computed and corresponding root segments are added to the *shape pool*, and the iteration stage, where the decomposability of segments in the shape pool is determined and those that are identified as decomposable are decomposed, while the shape pool is updated accordingly.

We studied the time spent both in the initialization stage and in every iteration of the iteration stage. For that matter is is now relevant to introduce the system were the experiment took place. The tests referred in this section were executed in a Intel® Pentium® 4 CPU at 3.6GHz with 1GB of memory. The chart depicted in Figure 5.9 illustrates the measured execution times when processing the complete ESB collection. The observed execution times mimic the behavior of the shape pool growth. After the third iteration the iteration time starts decreasing and achieves a very low value after a few iterations.

Indeed, a similar behavior is also observed in other collections. However, the time necessary to process the collection is proportional to the collection size. This means that a larger collection will have larger computation times. In Figure 5.10 we present the

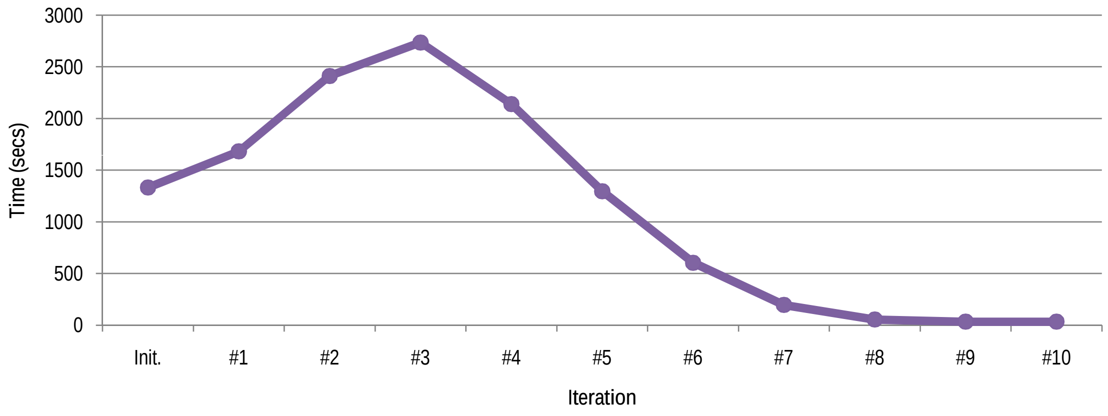


Figure 5.9: Detailed execution time of CAS/HFP algorithm when processing the complete ESB collection.

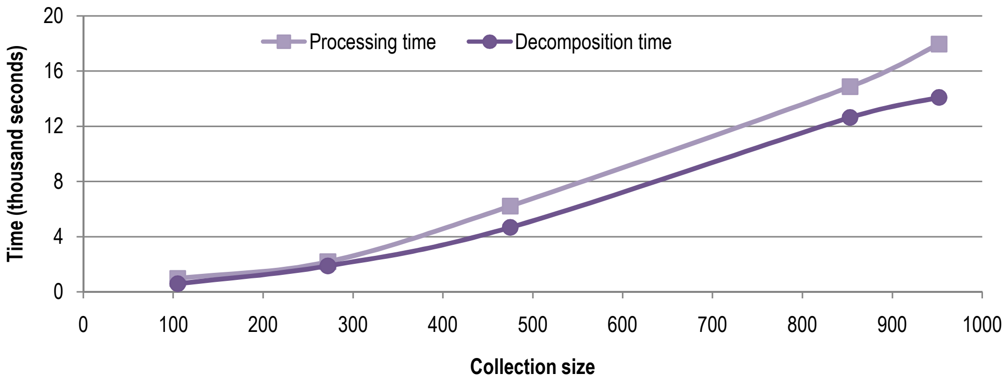


Figure 5.10: Execution time of CAS/HFP algorithm when processing collections with different size.

execution time for collections with different sizes. The smaller collections were the FTC, RCP and the SoR clusters of the ESB collection and then the complete ESB followed by a larger collection with almost one thousand models, comprising all ESB models joined with the one hundred LEGO^{TM} parts available at the National Design Repository [108]. We underline that each collection were fully processed, no data re-utilization was made from one to another, which would reduce drastically the measured execution times.

Considering that the execution time for a collection with around one thousand models will fall around the four hours, we concluded that the CAS/HFP algorithm will be able to produce valid results in suitable time, regarding that this algorithm is supposed to be batch processed, we think that such computation times are perfectly acceptable for this type of collection. However, to handle larger collections it might be necessary to reduce the time requirements of our approach.

5.3.5 Time Distribution

To identify which parts of the decomposition process are consuming more time, we measured the time spent by each task separately. Regardless of the stage in the CAS/HFP algorithm, we identify three main tasks executed during decomposition. The HSM estimation task consists on computing the HSM tree for a model. The signature computation task comprises the determination of the feature vector to represent a shape. Finally, the similarity estimation task consists on determining how many segments are similar to a given segment.

From the measured times we observed that, as depicted in Figure 5.11, most of the processing time is spent computing the shape signatures. Indeed, in preliminary tests we noticed that, even for simple shapes, the estimation of the SHA signature took between one and two seconds. Thus it was expectable that when the shape pool grows, computing all the signatures turns into a time consuming task. However, the SHA shape descriptor offers great descriptive power and we prefer to take advantage of it at the cost of execution time.

Nevertheless, to process larger collections another descriptor should be considered. We suggest a simpler descriptor, with a smaller signature and with shorter computation time. Such shape descriptor will most certainly provide less descriptive power that the SHA descriptor. On the other hand, using it will entail less memory for *shape pool* storage and shorter execution times.

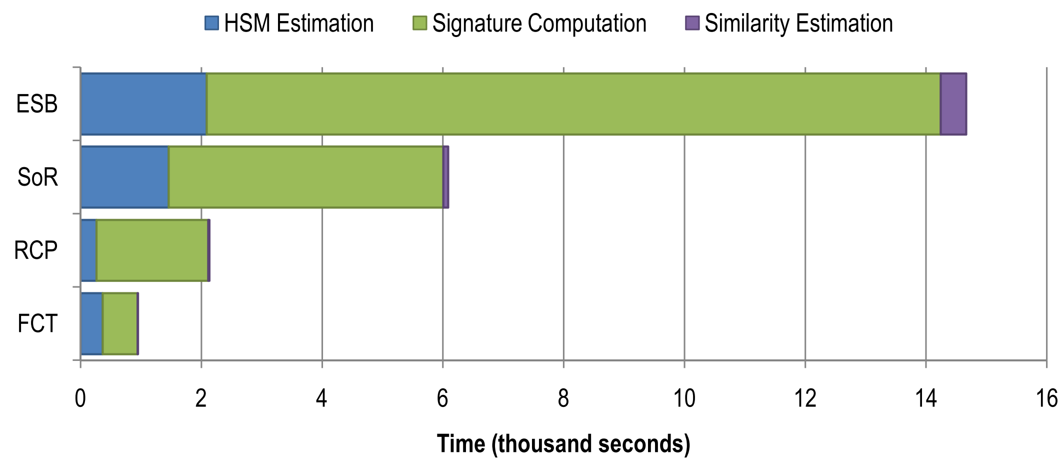


Figure 5.11: Time spent by each stage of CAS/HFP algorithm.

5.4 Updating a Decomposition

As shown in previous section, decomposing a collection of three-dimensional models using the CAS/HFP algorithm is a time-consuming task. However, having the collection processed, updating the decomposition after adding models to the collection is faster. To verify the algorithm behavior stated in last sentence, we performed a set of tests described in this section.

5.4.1 Experiment Description

The goal of this experiment was to study the behavior of the CAS/HFP algorithm when updating the decomposition after adding new models to the collection. To that end we considered the decomposition of the whole ESB collection described in previous section as the basis for these tests.

As source for models to be added to the ESB collection we used the unclassified LEGO® dataset from the National Design Repository at Drexel University [108] and the Watertight Models collection (WTM) introduced at the 2007 Shape Retrieval Contest [62].

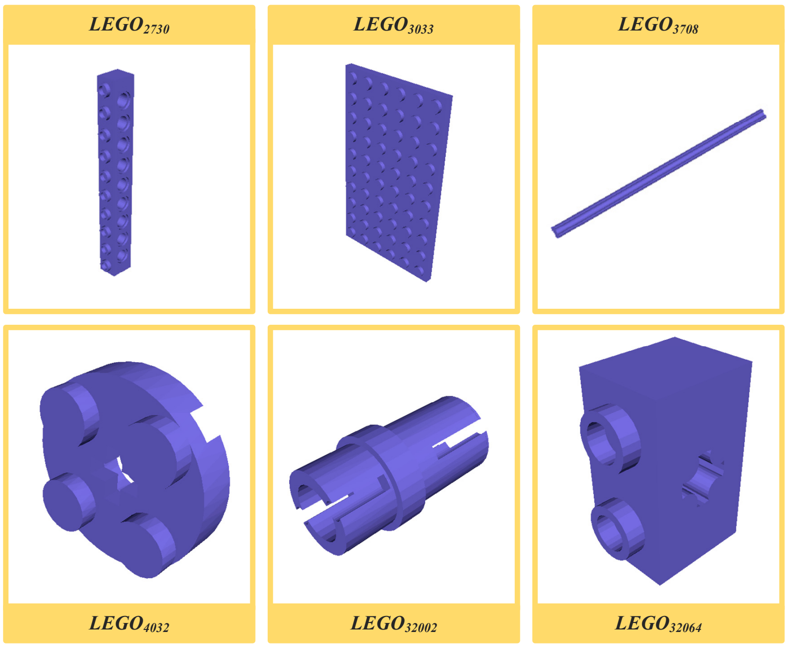


Figure 5.12: Six models sampled from the LEGO dataset.

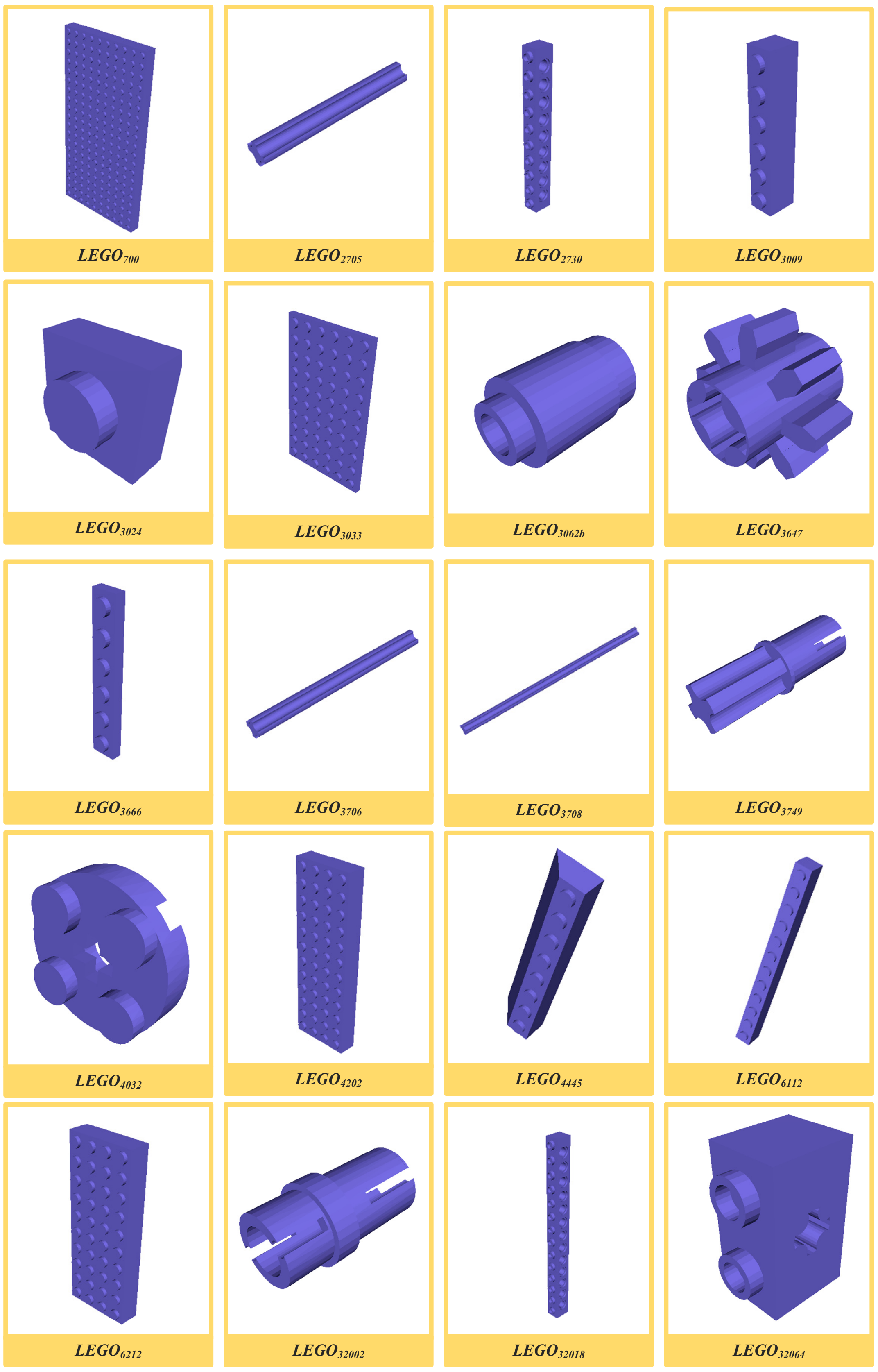


Figure 5.13: Twenty models sampled from the LEGO dataset.

The experiment was divided into four different tests, in the first two tests we randomly selected six and twenty objects from the LEGO dataset, creating a set of models for each test. The contents of these sampled sets are depicted, together with the corresponding object name, in Figures 5.12 and 5.13. In both cases we started the test with the already decomposed ESB collection and, after adding the corresponding sample set to the collection, we updated the decomposition.

In the third test, instead of a sampled set, we added the complete LEGO dataset to the whole ESB collection and then updated the decomposition. The LEGO dataset we wtilized contains one hundred models of LEGO® parts, available at the National Design Repository.

Finally, the fourth tests consisted on adding the LEGO dataset together with the complete WTM collection. The WTM collection contains four hundred high resolution models of diverse objects collected from a variety of sources. Thus, in this test five hundred models were added to an already decomposed ESB collection and the decomposition was updated.

During this experiment we analyzed not only the time required to update the decomposition after adding new models to the collection, but also the differences on the decomposed models. In the following sections we present and comment the observed results.

5.4.2 Execution Time

While the time consuption of the decomposition proccess might seem a drawback of our approach to shape retrieval, we argue that this is just an issue while decomposing the collection for the first time. Indeed, assuming that small sets of new models are sparsely added the collection, updating the decomposition will be a relatively fast task. Thus,

		Execution time
Decompose ESB collection	853 models	14.872 secs.
Update collection	+6 models	614 secs.
Update collection	+20 models	817 secs.

Table 5.4: Time necessary to decompose the ESB collection and to update the resulting decomposition after adding six and twenty models to the collection.

we considered a scenario of six to twenty new models added per day to the collection and a daily decomposition update and measured the time such update will take. The results obtained are presented in Table 5.4, where is possible to observe that updating the decomposition is much faster than compute it for the first time.

The shorter update execution time is justified by a simple reason. While updating the decomposition after adding a few models to the collection the most time consuming steps of the process are drastically reduced. Indeed, for the models already decomposed, the HSM tree is already computed, as well as the SHA signatures for the decomposed segments. Thus, is just necessary to compute the HSM for the new models and the signatures for the corresponding segments. Additionally, it might be necessary to compute a few more signatures, corresponding to possible new segments for models already decomposed and whose decomposition will change.

Therefore, as depicted in Figure 5.14, during decomposition update the HSM estimation time falls to unnoticeable values and the signature computation tooks just a fraction of the time spent in the initial collection decomposition. As expected, the similarity estimation, necessary to determine the decomposable segments, consumes practically the same time in both cases.

From the analysis of execution times measured during this experimente we concluded that updating the decomposition does not require excessive execution time, unless a large number of models have been added. Indeed, updating the decomposition after adding hun-

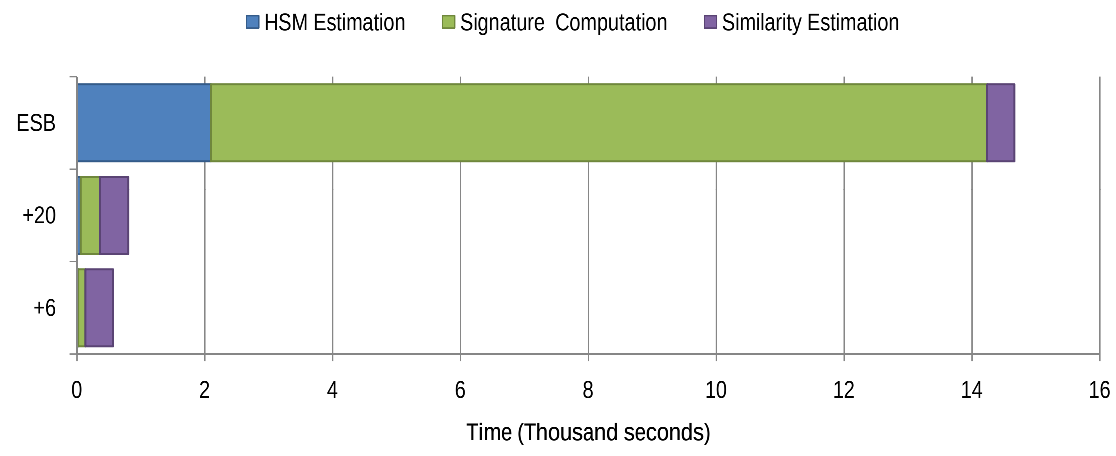


Figure 5.14: Execution times measured while decomposing the whole ESB collection and when six and twenty models are added to the collection and the decomposition is updated.

dreds of new models will require the computation of HSM trees for all those new models and the calculation of corresponing segment signatures, which is a time-consuming task. In such context the decomposition update can take a few hours. However, when adding around ten or twenty models, the decomposition will be updated in a matter of minutes.

5.4.3 Decomposition Stability

Besides the execution time, we studied the stability of the decomposition after adding new models to the collection. To that end, we measured the differences in the model segmentation after updating the decomposition with respect to the initial collection. To assess these differences we compared the segments that comprise each model before and after updating the decomposition. Then, we examined the comparison results and count the number of models that suffered no changes, those that were further decomposed (had more segments) and those whose decomposition was simplified (had less segments).

The observed values shown that, as depicted in Figure 5.15, while the number of new models in the collection remains low, none or few changes happen in the segmentation of already decomposed models. As the number of new models grows, increasing changes in decomposition are detected. In practice, after adding six models (0.7% of the collection) and updating the collection, no changes exist in the initial decomposition. On the other hand, when adding five hundred modes, which correspond to 58% of the initial collection size, the decomposition is different for more than four hundred models, *i.e.* around half

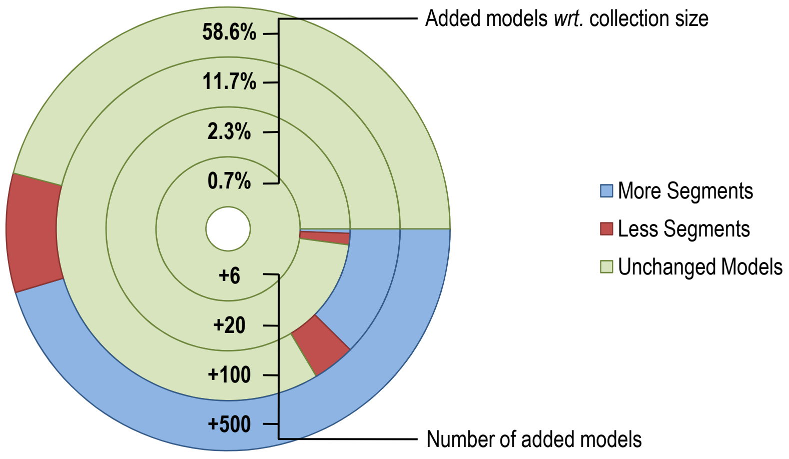


Figure 5.15: Stability of the decomposition when six, twenty, one and five hundred models are added to the collection.

	Models	
$diff < 0.5\%$	719	(84.3%)
$0.5\% \leq diff < 1.0\%$	67	(7.9%)
$1.0\% \leq diff < 2.0\%$	46	(5.4%)
$2.0\% \leq diff$	21	(2.5%)

Table 5.5: Segmentation difference distribution while adding five hundred models to the collection. Most of model segmentation suffered none or just a slight change after updating the decomposition.

of the models in the initial collection have different segmentation after adding the new models to the collection.

Even in the case when five hundred models are added to the collection, the changes found in each model segmentation is minimal. Indeed, only two models in the ESB collection have more than 4% difference between the initial segmentation and the one obtained after updating the decomposition. The segmentation of all other models suffered even slighter changes. Concretizing, as depicted in Table 5.5, the decomposition of approximately around eighty five percent of models in the collection did not change or suffered changes below 0.5%, while just 2.5% of he models present a segmentation difference greater than two percent.

From the analysis of the results presented above we conclude that the CAS/HFP algorithm offers a stable decomposition and, most important, that adding new models to an already decomposed collection and updating the decomposition practicaly does not alter the segmentation of initial models. Indeed, although we were not able to prove it with the collections currently available, we believe that when the decomposed collection is sufficiently large, adding new models and updating the decomposition will cause no changes in the decomposed models.

5.5 Building a Shape Thesaurus

Although we focused part of our research on the hierarchical decomposition of models in collections, this was just the groundwork for the construction of shape thesaurus. Indeed, this thesaurus is the basilar structure of our approach to 3D model retrieval. This thesaurus will be used to index the collection, thus supporting an efficient methodology to perform partial queries. In this section we present the results of our experiments on creating the shape thesaurus and corresponding inverted index for a collection of 3D models.

5.5.1 Experiment Description

Considering that the decomposition of models in the collection is the first stage of the thesaurus building process, as described in Section 4.1, the creation of a *shape thesaurus* from the resulting *shape pool* requires two additional stages. The second stage corresponds to the determination of a partition for the *shape pool*, while the third stage consists on the construction of the *shape thesaurus* from that partition.

The herein presented experiment was aimed to assess the behavior of our approach to index a collection, *i.e.* computing the *shape thesaurus* and the corresponding *inverted index*. This experiment comprised two distinct tests. While in the second test we processed the whole ESB collection and measured the execution time separately for each stage of the thesaurus building process, in the first test we studied the behavior of the segment clustering algorithm alone, since it is the most time-consuming task after the collection decomposition.

5.5.2 Segment Clustering Time

To cluster the segments in the *shape pool* we used, as explained in Section 4.3, a generic k-means clustering algorithm based on a combination of local search and Lloyd's algorithm. In our approach, the dataset to be clustered contains the signatures of segments in the shape pool. Basically, it consists on a set of points in a high-dimensional space. More precisely, due to the shape descriptor we applied, this space has 544 dimensions. Indeed, is the large dimensionality of the dataset to cluster that makes the whole clustering process more complex.

During this experiment we studied the behavior of the clustering algorithm with collections of different sizes, while modifying the number of clusters in the produced partition. The number of clusters directly related with the size of the thesaurus, since each cluster correspond to a term in the thesaurus.

The larger dataset tested contained less than twenty thousand points, and can be stored in less than fifty megabytes of memory. Since the memory required to compute the parti-

	Description	*Models*
FTC	Flat-Thin Components cluster from ESB	105
RCP	Rectangular Cubic Prism cluster from ESB	273
SoR	Solid of Revolution cluster from ESB	475
ESB	Whole ESB collection	853
ESB+LEGO	ESB collection plus LEGO models	953
ESB+LEGO+WTM	Above collection together with WTM	1353

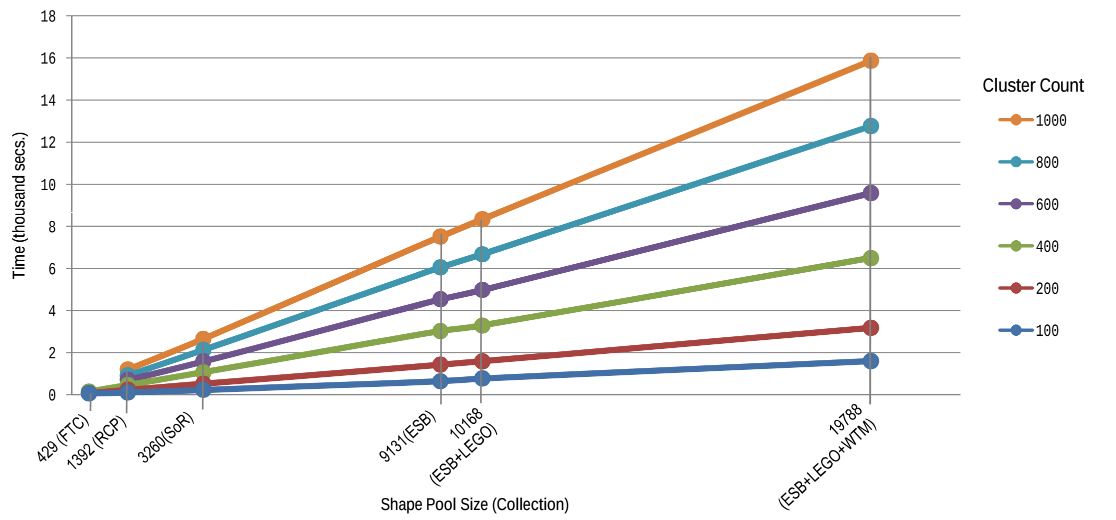
Table 5.6: Collections used in the segment clustering experiment.

tion of a dataset is just slightly higher that the memory used by the dataset and the centers of the clusters, the memory consumption of the clustering algorithm is not an issue in our approach. Thus, in this experiment we focused our attention on the execution time of the segment clustering algorithm.

The clustering algorithm is expected to perform with linear time-complexity with respect to the number of cluster and the number of points in the dataset. To assert this, we measured the execution time for a set of collections of different sizes, described in Table 5.6. Namely, the FTC, RCP and SoR clusters of ESB collection presented in Section 5.3, the whole ESB collection, a collection containing the ESB models plus the LEGO models introduced above in this section, and a collection containing the ESB models, the LEGO object and the WTM dataset.

The observed execution times confirmed the linear time-complexity of the clustering algorithm. As depicted in Figure 5.16, the execution time grows proportionally to shape pool size and cluster count. Thus, when fixing the number of clusters to produce and processing the different collections, we check that the execution time grows linearly. For instance, to build a thesaurus with eight hundred terms, which corresponds to a partition with the same number of clusters, the segment clustering algorithm took around six thousand seconds to process the *shape pool* resulting from the decomposition of the whole ESB collection. In our tests, this such pool contains 9131 segments. To build a thesaurus with the same size for a larger collection, the ESB+LEGO+WTM, to which corresponds a shape pool with almost twenty thousand segments, the segment clustering algorithm took less than thirteen thousand seconds.

The same behavior is observed when fixing the collection and varying the desired thesaurus size. In this case, the execution time of the algorithm grows linearly with the



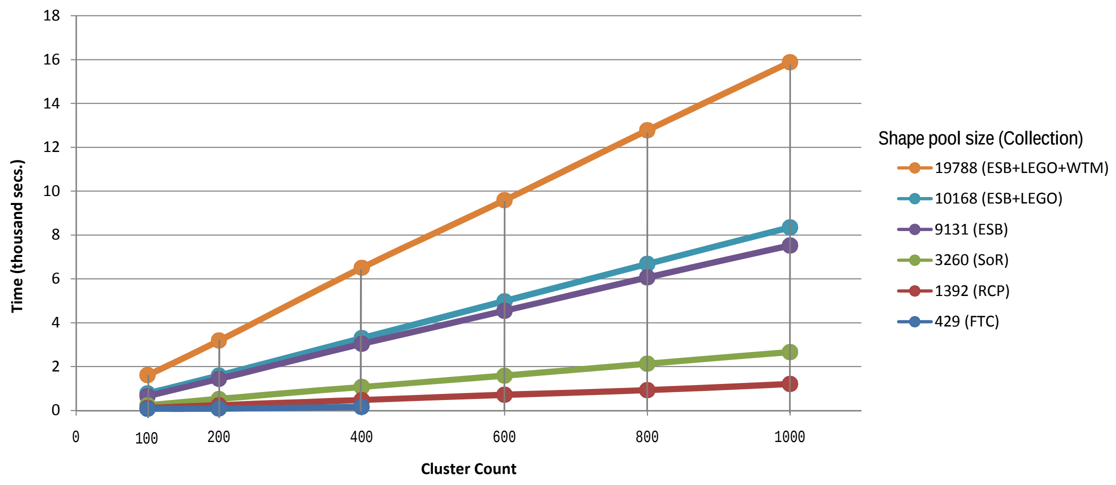


Figure 5.16: Execution time of segment clustering for partitions with different cluster count while changing shape pool size (top), and for different collections, while varying the number of clusters to be created (bottom).

number of clusters in the partition. For example, considering the whole ESB collection, while determining a partition with one hundred clusters took around six hundred seconds, computing a partition with eight hundred clusters took around six thousand seconds. This observation strengthens the importance of a reasonable size for the *shape thesaurus*. Indeed, a larger thesaurus will correspond to a heavier segment clustering process but led to a more efficient retrieval, while a smaller thesaurus will produce larger inverted lists in the inverted index, which will reduce the efficiency of the retrieval process.

From the statements above one can wrongly conclude that the thesaurus should contain as many terms as possible. However, this is not absolutely true: an excessively large thesaurus can easily become ineffective. In an extreme situation, a thesaurus will have the same number of entries than the segments in the pool, *i.e.* each term corresponds to a single segment. In this case. there is no point in using the thesaurus. It will be similar to search directly in the shape pool, which is exactly what we intend to avoid. Later we will discuss this topic and present some suggestions regarding the determination of a ”good” thesaurus size. For now, we will consider that the shape thesaurus to index the ESB collection will have eight hundred terms.

5.5.3 Execution time

To appraise the behavior of our approach to index a collection of 3D objects, we studied the process of creating a *shape thesaurus* and corresponding *inverted index* from a set of models represented as triangle meshes. To that end, and in accordance with previous experiments performed within our research, we used the collection of engineering models set up by Karthik Ramani team, the Engineering Shape Benchmark. As explained earlier in this document, this collection contains 853 CAD models stored as separate STL files. Each file contains an unstructured triangulated surface of the model. Indeed, the format itself is not relevant for the present purposes, but it is important to underline that, in the current implementation of the algorithm we deal with surfaces, a common way to represent CAD models.

In practice, the presented classification technique creates an indexing structure for a collection of three-dimensional surfaces. This indexing structure is composed by a *shape thesaurus* which contains a set of terms, and an *inverted index*, which maps the terms in the thesaurus with the models in the collection. Basically, each term corresponds to a prototype of surface patches, or segments, and is mapped to the models to which these segments belong. In the test described in this section we studied the computation of this indexing structure. Namely by measuring the execution time and analyzing the produced structures.

As described in previous chapter, the computation of the indexing structure for a collection of models is divided in three distinct stages: the collection decomposition; the segment clustering; and the thesaurus building. Although some of these stages can be further divided into significant steps, we will avoid that level of detail, since it has been

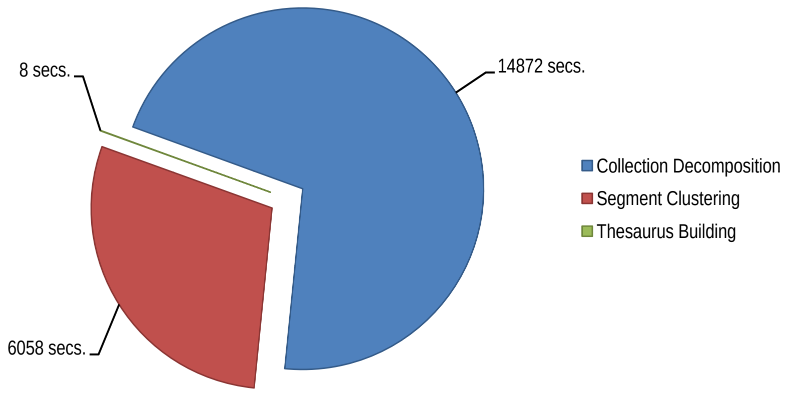


Figure 5.17: Time spent to index the ESB collection, *i.e.* to build the *shape thesaurus* and corresponding *inverted index* from the set of 853 models that compose the collection.

discussed separately earlier. Instead, we will provide a view of the behavior of the whole classification process.

In this test we measured the execution time of each step when processing the ESB collection. The results of these measurements are depicted in Figure 5.17. In this chart is easily comprehensible that the time required by the third stage is negligible and that the major time-consuming tasks are the decomposition and clustering stage. Indeed, while decomposing the collection took around fifteen thousand seconds and clustering the segments took six thousand seconds, building the thesaurus and creating the corresponding *inverted index* took only eight seconds. Hence, the whole classification process took twenty thousand seconds, *i.e.* around five and a half hours.

Considering that the classification process is performed just once and in batch, we claim that five and a half hours is a perfectly acceptable time. Moreover, when adding more items to the collection, updating the indexing structures is a much faster process. Regarding the decomposition stage,in Section 5.4 we shown that adding models to the collection and updating the collection decomposition is a relatively fast process and that the shape pool just suffer minor changes. Thus, knowing that if the clustering process starts with an existing partition, computing an updated partition for the same dataset with some changed points is not a time-consuming task, we can assert that our approach to index a collection suits the proposed purposes.

5.5.4 Thesaurus and Inverted Index

In our research work, we introduced a novel approach to three-dimensional shape retrieval. This new approach is based on concepts transposed from the text-information retrieval field. Namely, efficient indexing techniques widely used with text documents, more precisely the use of thesaurus and inverted indexes. Thus, we introduced the concept of a *shape thesaurus* and adapted the *inverted index* to fulfill our needs. Indeed, above we have already discussed the time-complexity of constructing such structures without looking at the resulting structures itself.

As described in Section 4.2, the *shape thesaurus* stores the signatures of the terms, *i.e.* the cluster centers, and to each term is assigned an unique identifier. In practice, the *shape thesaurus* is as a dataset in the signature space where each multi-dimensional point corresponds to a term. These points are associated to the term identifiers that are used in the *inverted index* to map terms with the models to which the segments associated with that term belong. To guarantee an efficient retrieval, these structures should be stored in memory, since the first step of the retrieval process is a k-NN search in the thesaurus signature space and the second step is to browse the inverted index.

Indeed, the space necessary to store the *shape thesaurus* is not a real issue, assuming that it has a reasonable size. Considering that each SHA signature requires around two kilobytes of memory, a thesaurus with one million terms will consume about two gigabites of memory. However, such a large thesaurus will fail to attain the purpose of our approach, since searching in such large dataset will be unacceptably slow. Moreover, the idea behind the thesurus is to group the segments that compose the models io the collection in a set of terms that represents them, which implies that the number of terms should be much smaller than the number of segments.

On the other hand, the size of the *inverted index* depends on the number of segments in the *shape pool* and on the size of the *shape thesaurus*. Basically, to each entry in the thesaurus will correspond a list of models in the inverted index. This list of models is no more than a list of model identifiers. The *inverted index* structure is relativey small (in the present experiment it requires less than fourty kilobytes) and its growth when handling really large collections can be minimized through data compression techniques, as occurs with its counterpart in text information retrieval.

Besides these two basic structures, a third support structure exists. This structure, despite its importance on the classification process, plays a secondary role in the retrieval phase. However, having in mind the information it contains, this structure is useful to improve the quality of the results returned by the retrieval mechanism. We are referring to the *shape pool*, which contains the signatures of all segments extracted from the models in the collection. For now, in the shape retrieval process the information stored in the shape pool is used only to order the retrieved results according to the distance of the corresponding segment signatures to the query signature. We believe that future research will identify further advantages for the retrieval process that can be obtained from the data stored in the *shape pool*.

For the example studied in this experiment, the ESB collection, all structures described above are relatively small when compared with the collection size. Indeed, the thesaurus containing the eight hundred signatures and corresponding identifiers occupies less than two megabytes, while the inverted index needs around forty kilobytes. The structure that stores the shape pool, a dataset containing more than nine thousand signatures uses about eighteen megabytes of memory. Summing, the memory required to store the whole indexing structure for the ESB collection will be within the tens of megabytes.

5.6 Retrieval of 3D models

The main goal of our research is to devise a scalable solution to 3D shape retrieval that supports partial queries. To that, end we developed a set of methods, structures and techniques to classify collections of 3D models and retrieve objects partially similar to a given query. The suitability of these techniques for indexing purposes was shown by the experiments described above in this chapter. In this section we focused on the retrieval process itself.

Indeed, despite the importance of the classification component in our approach to shape retrieval, the success of our solution depends on the results produced by the retrieval process. When a query is submitted, a successful retrieval system should provide an accurate answer in short time. To verify if our solution satisfies this premise, we tested our approach as described below.

5.6.1 Experiment Description

To evaluate the retrieval efficiency and accuracy of the thesaurus-based 3D shape retrieval introduced on this research work, we used the ESB collection. This collection was decomposed and indexed as described previously in the current chapter. To test the behavior of our solution we submitted five distinct queries to the system. From these queries, three (shapes Q_3 to Q_5) were subparts of existing models in collection, while two (shapes Q_1 and Q_2) were complete models from the ESB collection. These segments and models are depicted in Figure 5.18 and were randomly selected from the shape pool and collection, respectively.

Since time is a major concern of retrieval solutions, we measured not only the time to execute the query, but also the time to load the indexing structures into memory. In practice, the structure loading is done only once during retrieval system startup. After loaded, these structures remain in memory for as long as necessary, allowing the submission of innumerous queries without the necessity of reloading. This just must occur after changes in the collection with corresponding thesaurus update.

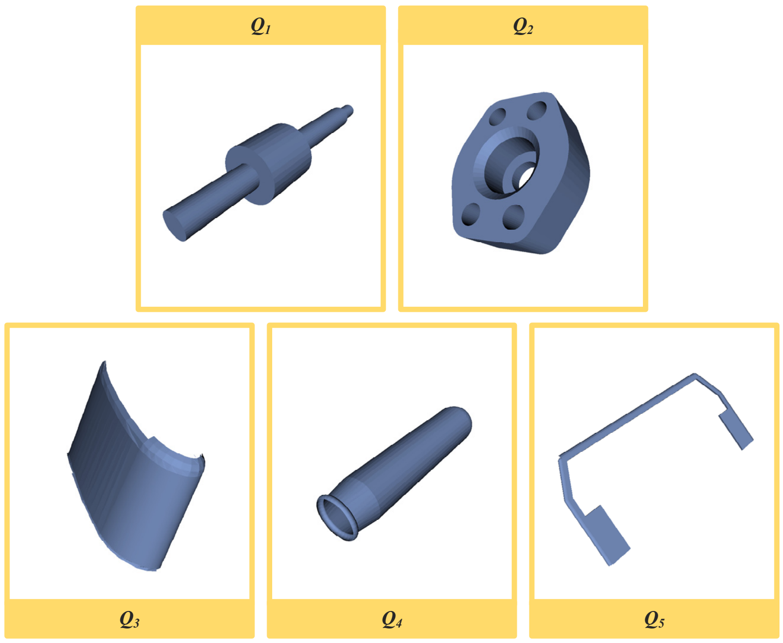


Figure 5.18: Query shapes submitted to the retrieval system.

5.6.2 Search Time

The retrieval prototype that implements our approach starts by loading into memory the indexing structures. Reading these structures from the non-volatile storage device is a relatively fast process. However, besides simply loading the information we also perform a set of validation and data coherency checks. In this experiment, the time we measured as loading time refers to all this process, not only reading from disk into memory.

In our approach, when a query shape is submitted the first step consists on estimating its SHA signature. Due to its inherent complexity, the computation of this signature is a time consuming process. Moreover, in the current implementation of our prototype we use an external application for this task. Using this external application involves additional execution time, but allows greater independence regarding the shape descriptor technique, when compared to a solution that incorporates the signature computation. Nevertheless, in a definitive solution, the signature computation time can be drastically reduced.

Having computed the signature of the query shape, the next step of the retrieval process consists on finding the more similar terms in the thesaurus. This is performed trough a linear k-NN search in the terms signature space. Despite its high dimensionality, this space has a controlled cardinality, which assures that the search for similar terms is really fast. Based on the search results and using the inverted index, the models containing segments similar to the query shape are identified.

In this experiment we measured the descriptor computation time separately from the time spent searching for similar terms in the thesaurus and finding the associated models, as presented in Table 5.7. We emphasize that the first depends exclusively on the query shape and shape descriptor, while the second is affected by several factors, such as the number of terms in the thesaurus, signature dimensionality or shape pool size.

		Queries				
		Q_1	Q_2	Q_3	Q_4	Q_5
Loading	121.64					
SHA computation (a)		1.182	1.187	1.085	1.163	1.120
Search (b)		0.162	0.172	0.181	0.165	0.161
Query $(a + b)$		**1.344**	**1.359**	**1.266**	**1.328**	**1.281**

Table 5.7: Measured time (in seconds) while loading the indexing structures into memory and performing the queries using a given shape.

The observed results show what we were already expecting, the computation of the query shape signature consumes most of the time spent while querying the collection. Indeed, while estimating the query signature took more than one second, in average, searching for models that satisfy the query took less than two tenths of a second. Moreover, none of these values depend directly on the collection size, which means that larger collections will only slightly affect the query time.

5.6.3 Memory Usage

We have shown above the efficiency, in terms of time complexity, of the retrieval process. In the present experiment, the indexing structures require around twenty megabytes of memory, including the signatures of all segments in the *shape pool*. Thus, the space complexity is not an issue. When the collection grows, the structures that will grow are the *inverted index*, which is always relatively small (around thirty kilobytes in this experiment), and the *shape pool*, which can be left on disk since no searches are made in this structure, just a few direct access to given signatures in each query.

Nevertheless, with a *shape pool* with less than ten thousand segments, the corresponding signature space occupies eighteen megabytes. Even if the *shape pool* contains one million terms, it might be loaded into memory, since it will require less than two gigabytes, which is available in today's middle-range desktop computers. In both cases, it can be entirely loaded into physical memory, thus improving the access time over the "stored on disk" approach.

Therefore, the retrieval prototype loads the indexing structures, as well as the *shape pool*, into memory, using around twenty megabytes. The *shape pool* is used only to obtain the segment signatures when ordering the retrieved models through direct access to the signatures. Indeed, after identifying the segments associated with each term similar to the query, our approach sorts the models according to the similarity between the matching sub-part and the query. The signature of the matching sub-part is obtained by following a pointer stored in the inverted index to the signature in the *shape pool*. This ordering task is remarkably fast, due to its simplicity and to the small number of models to sort, and greatly improves the results produced by the retrieval solution.

5.6.4 Retrieval Results

Since describing here exhaustively the results produced by the system for all queries will be long and repetitive, we selected only two queries to illustrate the retrieval capabilities of our approach. The first query, Q_1, corresponds to a complete object from the ESB collection, while the second, Q_5, corresponds to a sub-part of a model. These two queries cover the most valuable features of our solution. The ability to retrieve objects completely similar to the query and retrieve models partially similar to the query.

The output produced by our shape retrieval mechanism when query Q_1 is submitted is depicted in Figure 5.19. As expected, since the query shape corresponds to a model in the collection, the first returned result, $R_{1,1}$, is exactly that model. Obviously, it is the more similar to the query shape, since it is the same object. The following model returned by query, $R_{1,2}$, satisfies a part-in-whole matching with the query, *i.e.* a sub-part of $R_{1,2}$ is very similar to Q_1.

The remaining returned models are less similar to the query shape. Indeed, while $R_{1,3}$ and $R_{1,5}$ are, to some extent, globally similar to the query shape, only a sub-part of $R_{1,4}$ has some similarity with the query. As often occurs, the matching sub-part is not easily

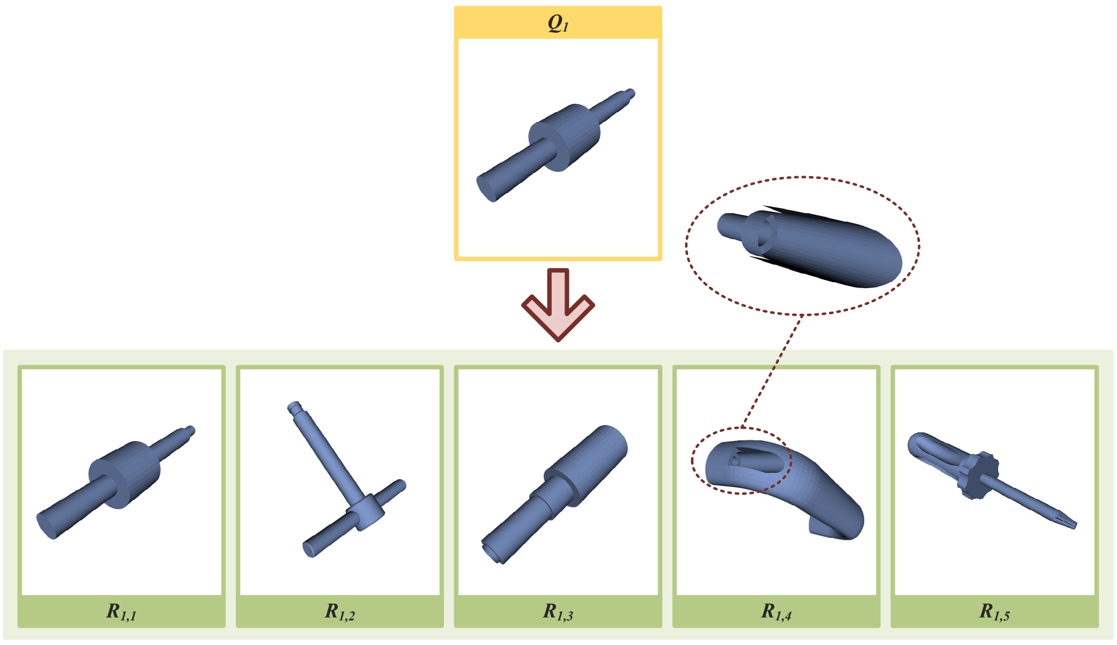


Figure 5.19: Query results for Q_1.

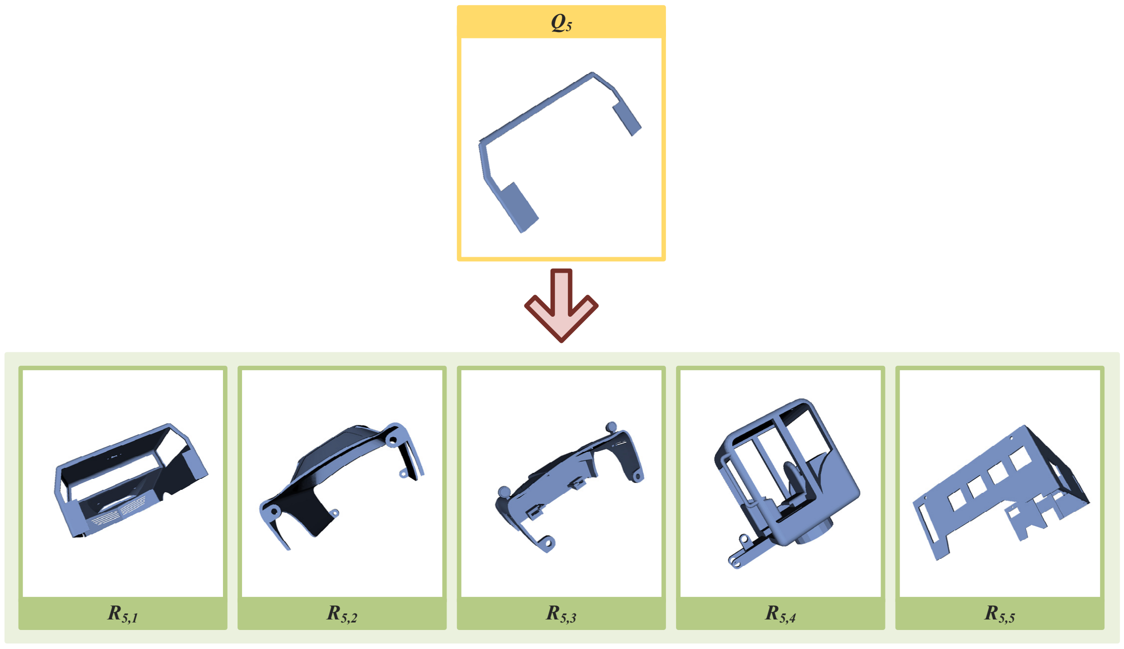


Figure 5.20: Query results for Q_5.

recognizable in the returned model. To assist the reader in the recognition of the similar sub-part, we highlighted it. We recall that this sub-part was automatically detected by the CAS algorithm during the classification phase as a distinctive segment of the original model.

In the second example, the shape Q_5 is itself a sub-part of a model. The outcome produced by our retrieval solution when Q_5 is submitted as a query is illustrated in Figure 5.20. As expected, the first retrieved model is the one to which this sub-part was extracted, the model $R_{5,1}$. The other retrieved models, $R_{5,2}$ to $R_{5,5}$, contain sub-parts similar to the query. Thus, in this example, all retrieved models satisfy a part-in-whole match with the query.

The ability to quickly retrieve models that are globally similar to a query or just partially similar, shown in the two examples described above, is one of the major achievements of our approach. Considering that through the use of a thesaurus and inverted index the scalability of this approach is unquestionable, we believe that we showed a possible path for a functional 3D retrieval system. While the classification component of our approach require further improvements to effectively support very large collections, the retrieval component is already able to provide quick and accurate answers to 3D queries.

5.7 Summary

In this chapter we presented the most relevant experiments performed during our research. Besides the experiments that aimed directly on validating and evaluating our approach we described here some of the supplemental tests that supported our research.

In the first of these experiments we studied the mesh complexity of models that comprise two widely used benchmark collections. From this study we concluded that a vast majority objects in these collections are defined by meshes with less than six thousand polygons. Based on this conclusion we assumed that a polygon reduction algorithm is not necessary in our approach, since our algorithms behave well with such mesh complexity.

The second experiment focused on simulating worst case conditions. In practice such conditions should never occur, but the tests performed within this experiment were useful to identify the limitations of our approach.

The other experiments described in this chapter focused on the algorithms devised during our research. Using a set of research prototypes that implement the algorithms presented in this dissertation, we studied the behavior of our approach when classifying benchmark collections and its effectiveness to retrieve models using partial queries.

As a result from these experiments we concluded that, while the space complexity is not a real problem in any of the proposed algorithms, the same is not true regarding time complexity. Indeed, the CAS/HFP decomposition algorithm and the segment clustering algorithm might take too long to compute very large collections. One reason for this fact is the high dimensionality of the SHA signature. However, for collections with a less than a thousand models, the execution time can be considered acceptable.

Regarding the retrieval process, the experiment presented in this chapter demonstrated the viability of our approach to 3D shape retrieval with partial matching. Not only the queries were executed in short time but also the quality of the returned results are quite good. Indeed, we found the retrieval results encouraging to further develop our approach, namely by improving the time-efficiency of the classification process. Nevertheless, based on the observed results we consider that they prove the validity of our approach.

6 Conclusions and Perspectives

The thesis work described in this dissertation focused on 3D object retrieval. More precisely, we aimed for a solution that supports partial queries, with part-in-whole matching. To that end, we devised a novel approach based on a thesaurus of shapes, a successful concept transposed from the textual information retrieval. In this chapter we summarize the main contributions of our work, stressing the more important results, and suggesting perspectives for future research.

6.1 Dissertation Overview

The core of the present dissertation can be divided into three distinct parts. In the first part we provide a research context in 3D shape analysis, classification and retrieval, together with some theoretical background required to better understand our work. The second part consists on a detailed description of our approach to 3D shape retrieval using a *shape thesaurus*. Finally, in the third part of this dissertation we present the experimental results.

6.1.1 Research Background

To familiarize the reader with the context of our research we started Chapter 2 by presenting a short list of the more relevant players in this field. This list is far from complete, since it contains only those researchers whose work we consider more relevant for our research. Moreover, in such a constantly evolving field it is impossible to compile a complete and fair list.

Additionally we listed the best-known 3D model collections, describing succinctly their contents. We also presented with some detail the more relevant benchmark collections, important to evaluate and validate our work. Furthermore, to provide the reader with a couple of important concepts on information retrieval, we included a quick overview of most common query strategies and similarity measurement techniques.

Due to the importance of shape descriptors in our field of research we included on the second chapter an extensive review of existing shape description techniques. In this review we began by presenting a taxonomy to classify 3D shape descriptors and then we described the distinct techniques, grouping them according to that taxonomy.

We concluded the research background part of this dissertation by presenting the most relevant 3D object retrieval solutions and the state-of-the-art concerning techniques to support shape retrieval with partial queries, which is indeed the focus of our work.

6.1.2 Approach Description

Our approach to 3D shape retrieval with partial queries can be divided into two main subjects. The first is the decomposition of models in a collection. The second is the construction of a thesaurus from a collection and the corresponding retrieval mechanism.

We used the *shape thesaurus* as a fundamental structure for our approach to shape retrieval. Such thesaurus should be composed by terms, which correspond to segments of the models. Therefore, the identification of that terms is a crucial part of our methodology. However, these terms are not explicit in the objects. Thus, the decomposition of objects in a collection is a primordial problem in our approach. In Chapter 3 we described in detail the solution we devised for this problem, the CAS algorithm.

After introducing our technique to identify the segments that compose the models in a collection, we presented, in Chapter 4 our approach to classify, index and retrieve such objects. In this chapter we described in detail the concept of *shape thesaurus* and explain how it is constructed from the pool of shapes produced by the CAS algorithm. We also presented the concept of *inverted index* and described the method devised to create such structure. Finally, we presented the methodology to retrieve 3D objects using partial queries, which relies on the *shape thesaurus* and corresponding *inverted index* to provide a fast and accurate retrieval.

6.1.3 Experimental Results

To assist us in decisions regarding our approach, we start performing experiments at the early stages of our work. In Chapter 5 we described the most relevant of these experiments, together with the tests made to evaluate and validate our approach. We began by presenting a study regarding model complexity in two widely used benchmark collections. Next we described a simulation of the behavior of our decomposition algorithm in the worst case scenarios, to identify its limitations and check its scalability.

Results of preliminary experiments provided valuable information that allowed the definition of a stable and robust set of techniques and algorithms. To evaluate and validate these, we developed several prototypes and used them to perform practical experiments. Detailed descriptions of these experiments and corresponding results are the core of the experimental results chapter.

6.2 Conclusions and Discussion

The experimental results provided us a valuable feedback regarding the validity of our approach. From these results we were able, not only to confirm the worth of our contributions, but also identify the limitations of our approach. In this section we will present and discuss both.

6.2.1 Contributions

While the majority of the research work in the 3D shape retrieval field focuses on shape description or matching techniques, we tackled a wider topic. Our research focused on a complete solution for 3D shape retrieval supporting partial queries. To achieve such solution we attacked distinct flanks, from shape decomposition to collection indexing. Although we obtained positive results in all of them, we highlight the collection-aware segmentation algorithm and the *shape thesaurus* as the most important contributions of our work.

Nevertheless, we consider that our approach has a whole is the major outcome of our research. Using the prototypes developed, and backed up by the experimental results we can state confidently that our idea - use of a thesaurus of shapes to index collections of 3D models in order to allow retrieval with partial queries - is not only valid but also capable of bear a fully functional search engine of three-dimensional shapes.

We believe that a thesaurus of shapes and corresponding *inverted index* are the most appropriate indexing solution for a shape retrieval system that aims on providing real scalability, partial query support, and retrieval efficiency. Indeed, successful textual information retrieval systems rely on a similar solution, which we consider as a good indicator for the potential of our approach.

6.2.2 Benefits

Several approaches to shape description have been proposed in the last decade, as shown in Chapter 2. The same is valid for shape matching techniques. Either complete matching, partial matching or part-in-whole matching techniques were published. However, to the best of authors knowledge, no integrated solution for retrieval of 3D models with partial queries that overcomes the scalability problem was published.

In our approach we propose a scalable approach for shape retrieval with partial matching. This approach relies on a thesaurus of shapes as an indexing structure to guarantee the scalability of our solution. Indeed, the retrieval time will not be directly influenced by the size or complexity of the collection of models, contrary to what happens in other approaches.

To construct such thesaurus it is necessary to identify the terms that comprise it. In a rough explanation, these terms correspond to segments that compose models in the collection, the same way words in a document correspond to terms in a lexicon. This is not a trivial task, since such segments are not explicit, as in text documents.

We devised an efficient algorithm that decomposes automatically, *i.e.* without human intervention, all the models in a collection. This algorithm constitutes a innovate approach to 3D shape segmentation and can be seen as a positive benefit to that field. Using the CAS algorithm it is simple to compute the segments that will be applied in the thesaurus construction.

Depending on the size and complexity of the collection the CAS algorithm can be relatively time-consuming. The same observation applies to the segment clustering, necessary to determine the terms in the thesaurus. However, since the whole classification process just have to be executed sparsely and can run in batch, time is not a major issue.

Therefore, the major benefit of our approach lies on the fact that the time consuming tasks are performed only during classification, while the retrieval process is fast and independent of the collection size.

6.2.3 Limitations

The current implementation of our approach has some limitations, such as supporting only non-manifold models, that can be easily overcome without major effort. However, some choices that we made during the development of our approach are indeed open issues to be tackled by future research.

The most noticeable of these limitations is related with the adopted shape descriptor. Although powerful in terms of descriptive capacity, the rotation invariant spherical harmonics descriptor has two major drawbacks. One is the time required for its computation. The other is the size of the computed signature. These facts undermine its use with a very large number of shapes, in terms of both time and space required to store them.

As we shown, in our approach the number of segments resulting from the multilevel decomposition tends to stabilize when the number of level grows. Indeed, in our experiments we observed that this stabilization happens around the sixth iteration of the CAS algorithm, that is when models were decomposed into seven or less levels. However, regardless of its growth, the total number of segments depends directly of the collection size, since the initial *shape pool* contain one segment for each model in the collection. Therefore, the size of the *shape pool* is directly proportional to the number of models in the collection.

To construct the *shape thesaurus*, we must compute a partition for the *shape pool*. This means clustering the signatures of the segments detected by the multilevel decomposition. Due to the high-dimensionality of the shape signatures, we used an hybrid k-means clustering technique that combines local search with the LLoyd's algorithm [89]. The execution time of such partition computation grows linearly with the size of the dataset, in the present case the number of segments in the *shape pool*.

From the two limitations referred above is easy to conclude that our approach has a linear classification time regarding the collection size. When aiming for a real scalability this might not seem the best of the conclusions. However, this can be solved by gaining more control on the *shape pool* growth.

6.3 Perspectives

The current limitations of our approach might threaten the attainability of our ultimate goal - to devise a scalable, robust, effective and fast solution to shape retrieval with partial queries. Although it is not within the scope of the present PhD research to find such solution, we believe that our approach can evolve in that direction and already identify research paths to be followed. In the next paragraphs we summarize some of these perspectives for future work.

Low-Dimensional Signatures To overcome the computation time and storage space required by the SHA signature, we suggest, as a path for future research, the investigation of shorter, yet powerful and fast to compute shape descriptors. Indeed, existing shape description techniques either are feeble descriptive or complex to compute. Nevertheless, aiming at immediate results, our approach can easily be adapted to use other shape descriptor, eventually not so powerful as the SHA descriptor but simpler to compute and with a low-dimensional signature.

Early Clustering The relation between the size of the collection and the number of segments in the *shape pool* can be reduced if similar segments are clustered immediately as they are detected during the CAS algorithm execution. This idea, we called early clustering[1], will drastically reduce the number of segments in the *shape pool*, thus reducing the space required to store the corresponding signatures and the number of comparisons required to determine the decomposability of a segment. In an extreme evolution, the segment clustering step executed after CAS decomposition can even be discarded.

[1]The original idea for early clustering was raised in a exchange of views with Francesco Robbiano and Marco Attene from the CNR IMATI-Ge.

Sub-linear Clustering Due to complexity of data structures used by sub-linear clustering algorithms, the current high dimensionality of the SHA signature makes its use pointless. However, if a shape descriptor with a shorter signature is adopted, more time-efficient clustering algorithms might be used. This fact strengthens the importance of investing on more space-efficient shape descriptor techniques, while looking into better data clustering algorithms.

Approach Optimization In this dissertation we shown that basic concepts from textual information retrieval can be successfully transposed to 3D shape retrieval. However, we believe that more advanced concepts and techniques can also be adapted from textual information retrieval field to improve the accuracy and efficiency of our approach. For instance, term frequency-inverse document frequency (tf-idf) weight can be used to quantify the relevance of segments to shapes in a collection, or combine a local Bernoulli compression method with Golomb coding to compress the indices [17].

Fully Partial Queries Our approach focused on part-in-whole matching. This means that, when a shape is submitted as a query, the retrieval system will return the models in the collection that has a segment similar to that query. This type of partial query considers that the submitted shape should be treated as a whole. We suggest that, for fully partial queries, the retrieval system should return not only the models that contain segments similar to the query object, but also those which contain segments similar to parts of the query object.

In practice, adapt our approach to support fully partial queries is simple. However, the goodness of such methodology should be carefully studied, namely regarding user needs. In an analogy with the textual information retrieval, a fully partial query is the same as using the word ”airplane” as query and the system returns documents with the words ”air” and ”plane”. While in text documents such might seem questionable, we make no assumption regarding the 3D model context. We leave this as an open question for future research.

6.4 Final Remarks

The ultimate goal of our research field is to devise fully-fledged content-based 3D model retrieval system. Such system should be able to retrieve models from very large collections. These models can be globally or only partially similar to the submitted query. Basically, we aim for a search engine for 3D shapes similar to existing solutions for text documents. Currently, this goal was not reach by anybody, and is beyond the scope of the this thesis work. In this dissertation we introduced a novel approach that provides the groundwork of a future 3D search engine.

Our approach uses a thesaurus for classification, indexing and retrieval. This thesaurus will contain the shapes that compose the models of the indexed collection, similarly to a lexicon in text retrieval, which contains the terms that compose the vocabulary used in documents. Combined with an *inverted index*, this thesaurus provides an efficient retrieval solution. Performing a query with our approach comprises a short pipeline of simple and fast steps. First, the signature of the query object is computed. Then, the terms in the thesaurus with the nearest signatures are found. Finally, the models associated to each term are properly ordered according to query-segment similarity. As we have shown in Chapter 5, this retrieval process is both fast and accurate.

While the simplicity of the retrieval process relies on the *shape thesaurus*, the construction of such structure is not a trivial task. Indeed, unlike text documents, where terms are explicit and lexicons can be easily built from them, the terms of the *shape thesaurus* are not explicit in the models. To determine the terms that constitute the thesaurus we group sub-parts of models in the collection according to their geometric similarity. Thus, each term in the thesaurus correspond to a point in the signature space and is associated, through the *inverted index*, to models that contain sub-parts resembling that shape. The major challenge in the thesaurus building process is the identification of that sub-parts.

Sub-part identification in 3D object is a complex task, due not only to its computational complexity but also to the ambiguity of such decomposition. Moreover, in our approach we need to decompose all models in the collection automatically, without human intervention. To achieve such goal we devised a novel approach to shape decomposition that performs multilevel shape segmentation. Instead of decomposing each model independently, the Collection-aware Segmentation algorithm, described in Chapter 3, extracts sub-parts of a model taking into account the segments of all other models. This algorithm performs a multi-level decomposition of models in the collection, producing a set of sub-parts. The *shape thesaurus* is build from these sub-parts.

A set of research prototypes implemented the different components of our approach, from sub-part identification to shape retrieval. These prototypes, presented in Annex C, served to evaluate our work through a set of experiments. Indeed, in these experiments we identified the drawbacks of our approach, which will be tackled by future research. But, above all, these experiments shown the validity of our approach, thus fulfilling the research hypothesis defended in this dissertation.

Concluding, while prototypes and experiments developed within our research focused on validating the proposed ideas, a functional retrieval solution can be easily implemented from them. Such system will be yet far from the ultimate goal. However, having shown the validity of our ideas and pointed paths for future research, we believe that concepts and techniques presented in this document have the potential to evolve into a fully-fledged 3D shape retrieval solution.

A
Shape Similarity Study

In this annex we present in detail the study we performed in order to understand the behaviour of the spherical harmonics (SHA) descriptor while measuring the similarity between models from the Engineering Shape Benchmark (ESB) collection [71]. To that end, we compute the multidimensional feature vector corresponding to SHA representation of every studied model and measured the Euclidean distance between these feature vectors.

The study was divided in three distinct parts. The first part focused on studying the behaviour of the SHA descriptor in a set of random models. In the second part we examined sets of geometrically similar models and studied the behaviour of feature vectors produced by SHA in these cases. Finally, in the third part all models in the ESB collection were considered as a query to the collection in the SHA feature vector signature space. In the following sections we report in detail these three parts of our study, describing the tests and the results obtained in each one of these.

A.1 Random shapes

In the first part of the study we investigate the similarity between models randomly extracted from the ESB collection. We arbitrarily select twenty three-dimensional objects to create a set ξ_{rand} of models. Since this set contains more than two percent of the whole collection, we assumed that it holds a representative sample.

The models that comprise the set ξ_{rand} are depicted in Figure A.1, indicating the name assigned to each one in the ESB collection, and in Figure A.2 are presented the corresponding SHA signatures. These models are also listed in Table A.1, which contains the description of each model according to the classification scheme presented bin [71].

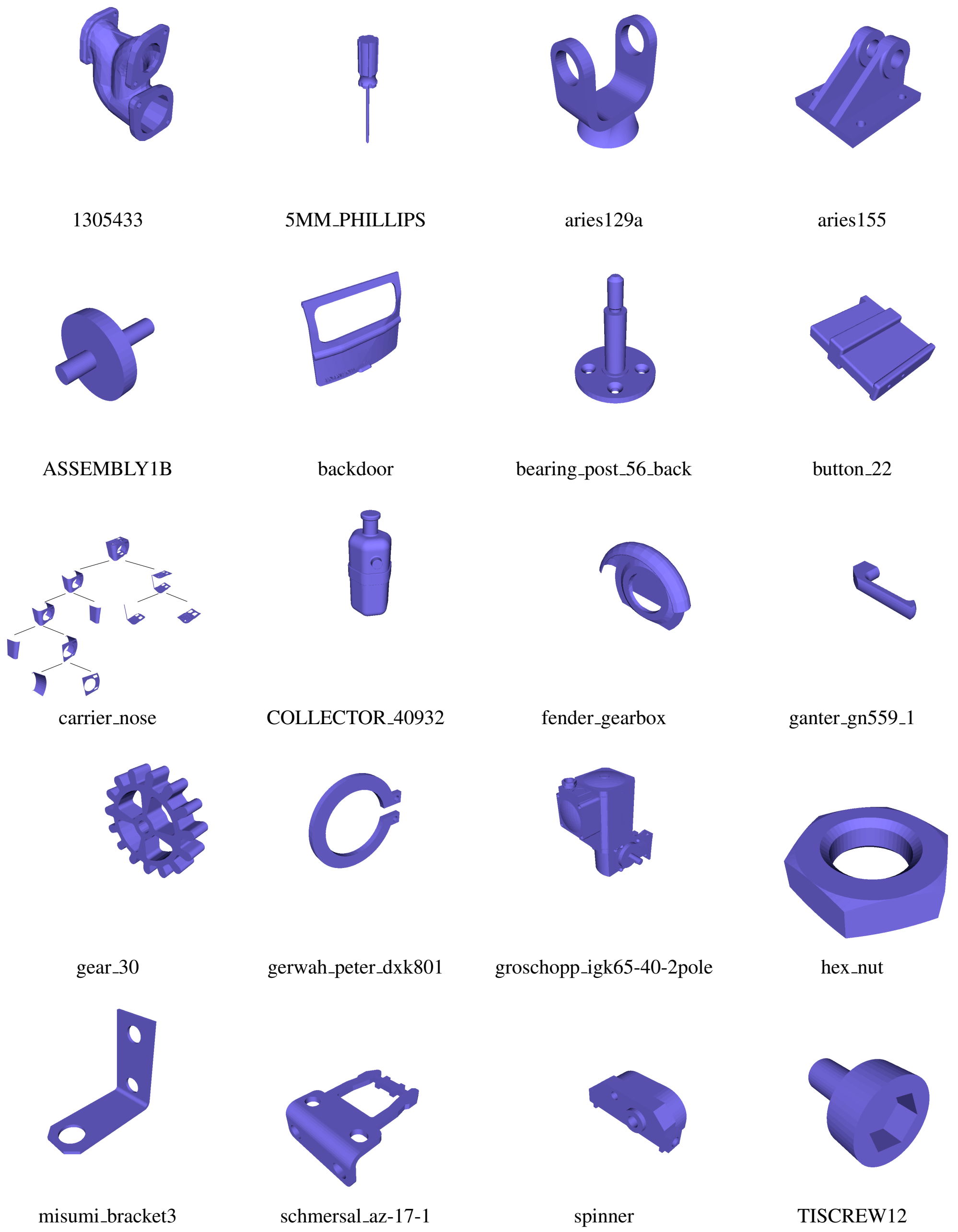


Figure A.1: Models randomly selected from the ESB collection.

After computing the SHA descriptor for every model in ξ_{rand}, we measure the distance, $d_{i,j}$, between the corresponding feature vectors. To that end we used the applications *getsig* and *cmpsig* developed by Michael Kazhdan [78] to compute the SHA signatures of the models and to compare these signatures, respectively. This way we obtained the distances that represent the model similarity for the whole sample. These distances will allow us to better understand the behaviour of the SHA descriptor with CAD models.

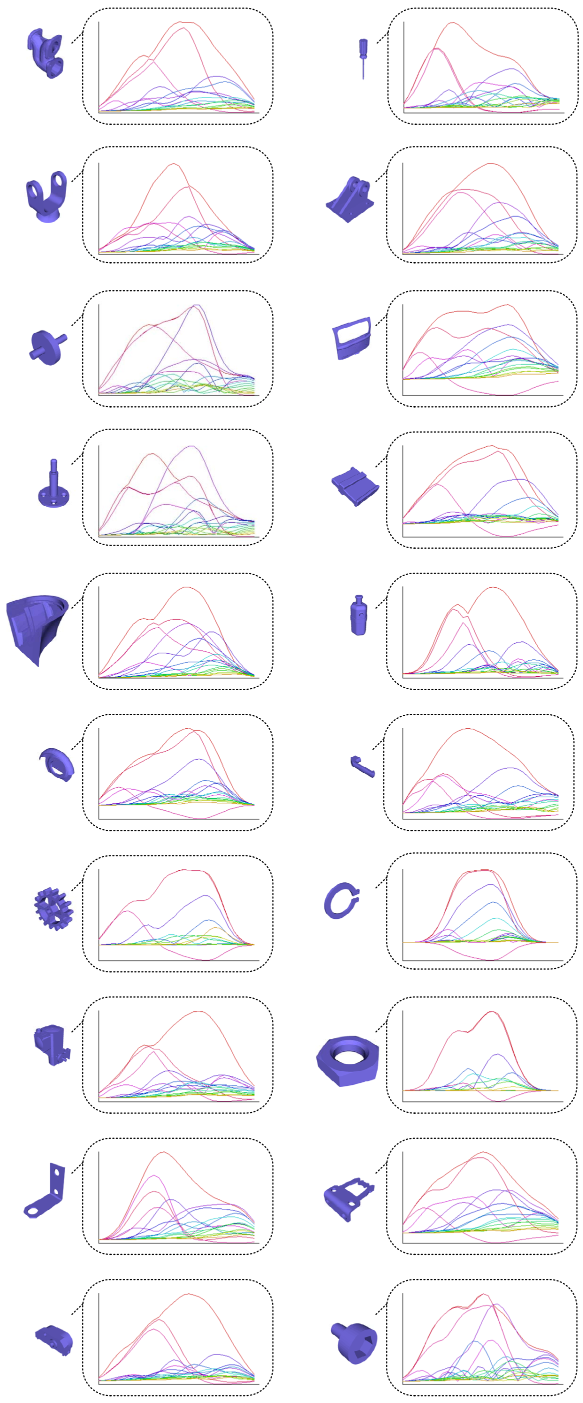
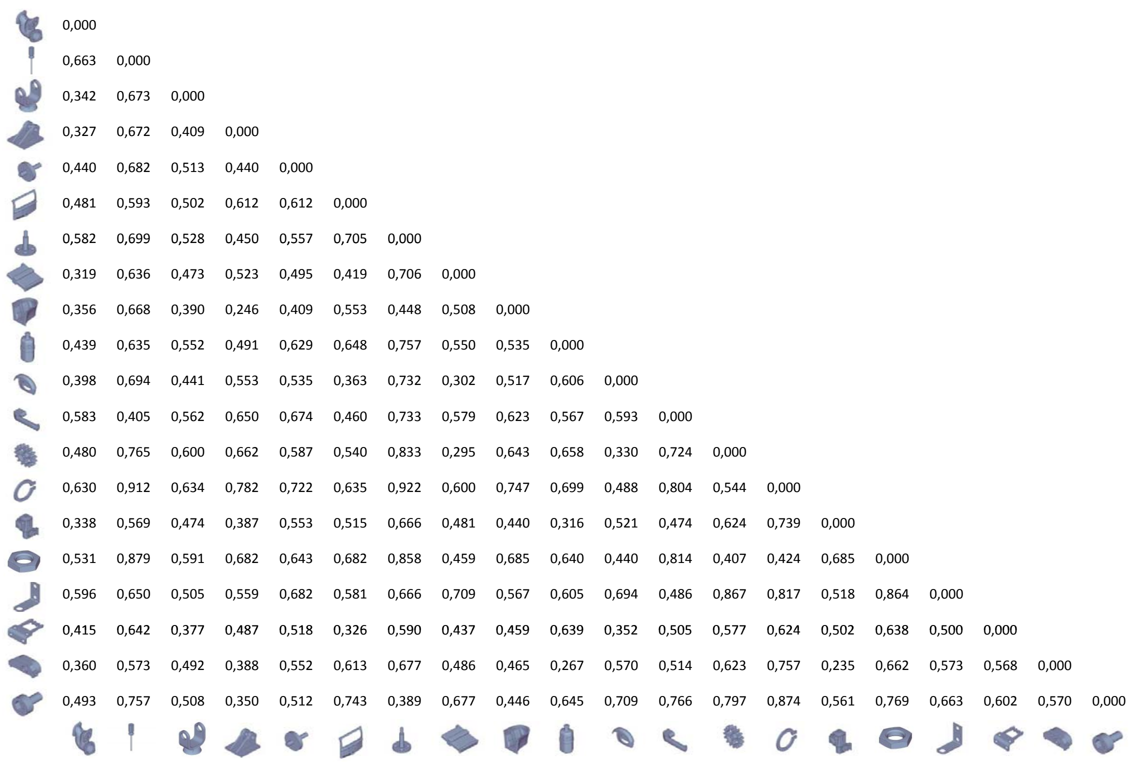


Figure A.2: SHA signatures of models in ξ_{rand}.

	Model Name	ESB Cluster	ESB Class
1	1305433	Solid Of Revolution	Intersecting Pipes
2	5MM_PHILLIPS	Solid Of Revolution	Long Pins
3	aries129a	Rectangular-Cubic Prism	Miscellaneous
4	aries155	Rectangular-Cubic Prism	Miscellaneous
5	ASSEMBLY1B	Solid Of Revolution	Posts
6	backdoor	Flat-Thin Wall components	Back Doors
7	BEARING_POST_56_BACK	Solid Of Revolution	Posts
8	button_22	Rectangular-Cubic Prism	Thick Plates
9	carrier_nose	Flat-Thin Wall components	Miscellaneous
10	COLLECTOR_40932	Solid Of Revolution	Container Like Parts
11	fender_gearbox	Flat-Thin Wall components	Curved Housings
12	ganter_gn559_1	Rectangular-Cubic Prism	U shaped parts
13	gear_30	Solid Of Revolution	Spoked Wheels
14	gerwah_peter_dxk801	Flat-Thin Wall components	Clips
15	groschopp_IGK65-40-2pole	Rectangular-Cubic Prism	Motor Bodies
16	HEX_NUT	Solid Of Revolution	Nuts
17	misumi_bracket3	Flat-Thin Wall components	Bracket like Parts
18	schmersal_AZ-17-1	Flat-Thin Wall components	Contact Switches
19	SPINNER	Rectangular-Cubic Prism	Machined Blocks
20	TISCREW12	Solid Of Revolution	Bolt Like Parts

Table A.1: Description of models in ξ_{rand}.

The distances $d(FV_i, FV_j)$ measured between all pairs of models S_i and S_j in the set ξ_{rand} are presented in Figure A.3. These values represent the similarity $s_{i,j}$ between shapes, where a smaller value means that the shapes are more similar while larger values means that the shapes are less similar.



0,000																			
0,663	0,000																		
0,342	0,673	0,000																	
0,327	0,672	0,409	0,000																
0,440	0,682	0,513	0,440	0,000															
0,481	0,593	0,502	0,612	0,612	0,000														
0,582	0,699	0,528	0,450	0,557	0,705	0,000													
0,319	0,636	0,473	0,523	0,495	0,419	0,706	0,000												
0,356	0,668	0,390	0,246	0,409	0,553	0,448	0,508	0,000											
0,439	0,635	0,552	0,491	0,629	0,648	0,757	0,550	0,535	0,000										
0,398	0,694	0,441	0,553	0,535	0,363	0,732	0,302	0,517	0,606	0,000									
0,583	0,405	0,562	0,650	0,674	0,460	0,733	0,579	0,623	0,567	0,593	0,000								
0,480	0,765	0,600	0,662	0,587	0,540	0,833	0,295	0,643	0,658	0,330	0,724	0,000							
0,630	0,912	0,634	0,782	0,722	0,635	0,922	0,600	0,747	0,699	0,488	0,804	0,544	0,000						
0,338	0,569	0,474	0,387	0,553	0,515	0,666	0,481	0,440	0,316	0,521	0,474	0,624	0,739	0,000					
0,531	0,879	0,591	0,682	0,643	0,682	0,858	0,459	0,685	0,640	0,440	0,814	0,407	0,424	0,685	0,000				
0,596	0,650	0,505	0,559	0,682	0,581	0,666	0,709	0,567	0,605	0,694	0,486	0,867	0,817	0,518	0,864	0,000			
0,415	0,642	0,377	0,487	0,518	0,326	0,590	0,437	0,459	0,639	0,352	0,505	0,577	0,624	0,502	0,638	0,500	0,000		
0,360	0,573	0,492	0,388	0,552	0,613	0,677	0,486	0,465	0,267	0,570	0,514	0,623	0,757	0,235	0,662	0,573	0,568	0,000	
0,493	0,757	0,508	0,350	0,512	0,743	0,389	0,677	0,446	0,645	0,709	0,766	0,797	0,874	0,561	0,769	0,663	0,602	0,570	0,000

Figure A.3: Similarity values between random models.

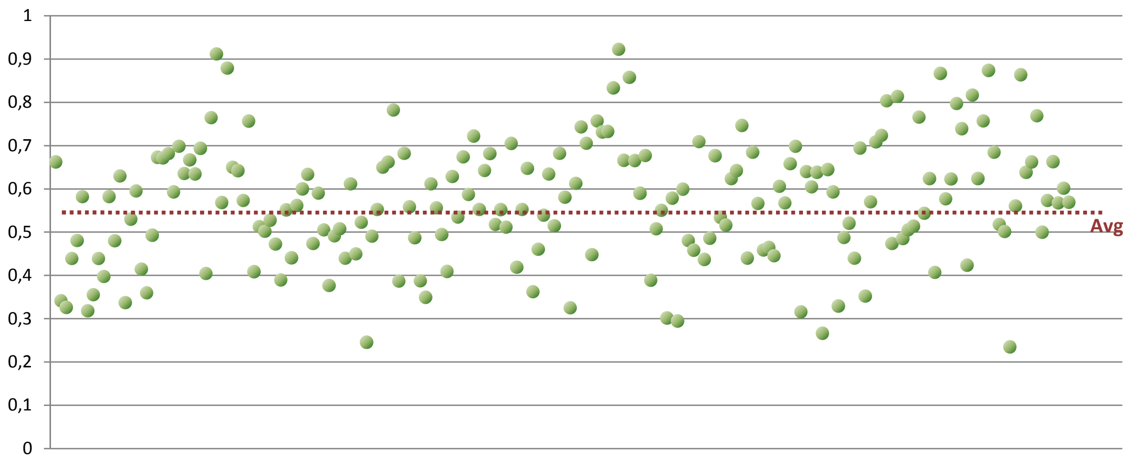


Figure A.4: Chart representing similarity values between random models.

As expected, the obtained similarity values range from almost complete dissimilarity ($d_{14,7} = 0.92$) to relative similarity ($d_{15,19} = 0,24$). Indeed, in such a randomly obtained small sample it is not expected to have really similar shapes, which justifies the absence of smaller values. From the measured distances we estimated that $\overline{d_{\xi_{rand}}} = 0.542$, with $var(d_{\xi_{rand}}) = 0.034$, as depicted in the scatter chart presented in Figure A.4, which contains all measured distances between pairs of feature vectors.

The analysis of the obtained results shown that, in average, the distance between feature vectors will fall around the middle point of the possible range. Indeed, the similarities are distributed by a wide range of possible values but most of the distances are in the vicinity of $d_{mid} = 0.5$, as depicted in Figure A.5.

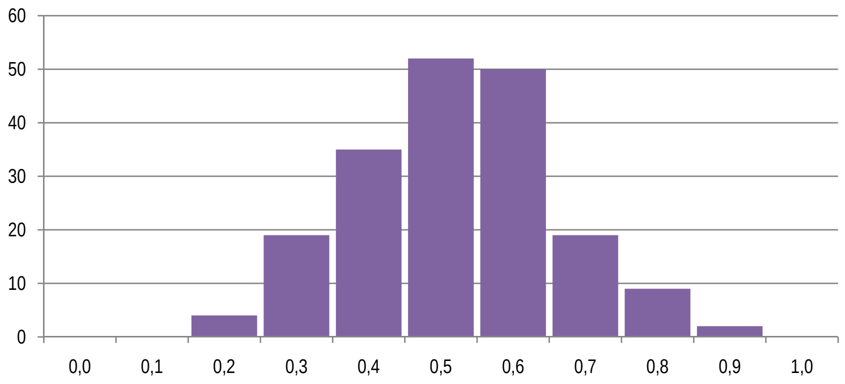


Figure A.5: Distribution of similarity values between random models.

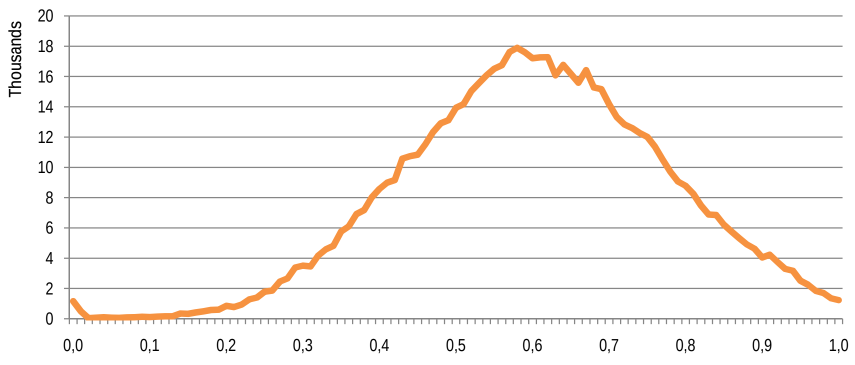


Figure A.6: Distribution of similarity values between all models in the collection.

A.1.1 Similarities among all models

To check the validity of the conclusions above we performed an additional test. Instead of using a sample set and compare the distance between the shapes, we used a set ξ_{all} containing all models of the ESB collection. Then, we computed the signatures for all objects in ξ_{all} and measured all similarities between them. Such test involved estimating the feature vectors for the 853 models that constitute the ESB collection and then perform more than 360 thousand comparisons among all shapes in order to obtain the complete similarity map.

This similarity map is a matrix $\mathcal{M}_{sim}$ that contains the distance between all models. Due to distance properties, the size of this matrix was reduced by using a triangular matrix. Thus, we employed a triangular matrix $\mathcal{M}_{sim}$ with 853 rows and 853 columns to store the distances between all models in the ESB collection.

From data stored in $\mathcal{M}_{sim}$ we computed the average distance, $\overline{d_{\xi_{all}}} = 0.598$, and the distribution of the similarity. The histogram depicted in Figure A.6 illustrates that distribution. The analysis of this distribution confirms the assertion made above regarding the similarity values in random shapes.

Indeed, the peak of the histogram is around $d = 0.6$, but this happens because the number of dissimilar shapes in the whole collection is much larger than the number of similar models. If we consider a vicinity $\mathcal{V}$ with a radius $\delta = 0.15$ we observe that more than 62% of the similarities lie in the interval $0.45 < d < 0.75$, as illustrated in the chart depicted in Figure A.7. Here is possible to see that the remaining similarities are distributed equally at left and right of vicinity $\mathcal{V}$.

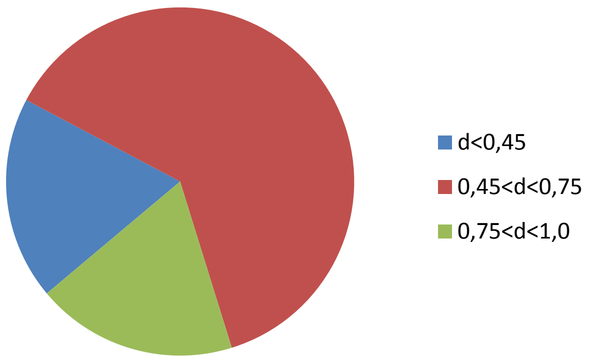


Figure A.7: Distribution of similarity values between all models in the collection.

A.2 Geometrically similar shapes

For the second part of the study we used the query mechanism provided with the ESB to create sets of similar shapes. To that end we select a small set of six shapes randomly chosen from the ESB collection and use them as queries to the ESB search system. From the answers produced by the system we created the sets of similar models ξ_{sim_i} for our tests.

Each set ξ_{sim_i} contained eight models S_{i_j} identified as similar by the ESB query mechanism. While in Table A.2 we describe these models according to their classification in the ESB colection, in Figures A.8 to A.18 we present the different sets, illustrating the models that campose each set $\xi_{sim_i} = \{S_{i_1}, ..., S_{i_8}\}$ and corresponding signatures FV_{i_1} to FV_{i_8}, and indicating the respective names in the ESB collection.

As in the previous tests, the feature vectors which represent the models were computed through the SHA shape descriptor, using the same tools. After computing these signatures for all models, we estimated the similarity

$$s_{i_{j,k}} = d_2(FV_{i_j}, FV_{i_k})$$

between them, given that i is the index of the shape set while j and k are the indices of two shapes in that set. The measured distances are shown in Figures A.9 to A.19, where for each model we indicate the similarity to all other models in the same set.

	Model Name	ESB Cluster	ESB Class
S_{1_1}	bed2_left_sidepannel	Flat-Thin Wall Components	Slender Thin Plates
S_{1_2}	bed2_right_sidepannel	Flat-Thin Wall Components	Slender Thin Plates
S_{1_3}	bed_left_sidepannel	Flat-Thin Wall Components	Slender Thin Plates
S_{1_4}	bed_tailgate	Flat-Thin Wall Components	Slender Thin Plates
S_{1_5}	CLAMP_FINGER_PRT	Rectangular-Cubic Prism	Long Machine Elements
S_{1_6}	PART_13_PRT	Flat-Thin Wall Components	Slender Thin Plates
S_{1_7}	PART_4_PRT	Rectangular-Cubic Prism	Slender Links
S_{1_8}	SLIDER_.STL	Flat-Thin Wall Components	Slender Thin Plates
S_{2_1}	2473396	Solid Of Revolution	Oil Pans
S_{2_2}	2473396_1	Solid Of Revolution	Oil Pans
S_{2_3}	2487583	Solid Of Revolution	Oil Pans
S_{2_4}	2487583_1	Solid Of Revolution	Oil Pans
S_{2_5}	2487583_3	Solid Of Revolution	Oil Pans
S_{2_6}	2494009	Solid Of Revolution	Oil Pans
S_{2_7}	2494009_1	Solid Of Revolution	Oil Pans
S_{2_8}	sh-r44357-000-u	Solid Of Revolution	Flange Like Parts
S_{3_1}	1546387	Rectangular-Cubic Prism	Rocker Arms
S_{3_2}	1546387_1	Rectangular-Cubic Prism	Rocker Arms
S_{3_3}	2360536	Rectangular-Cubic Prism	Rocker Arms
S_{3_4}	2360536_1	Rectangular-Cubic Prism	Rocker Arms
S_{3_5}	7n3433	Rectangular-Cubic Prism	Rocker Arms
S_{3_6}	7n3433_1	Rectangular-Cubic Prism	Rocker Arms
S_{3_7}	9y4757	Rectangular-Cubic Prism	Rocker Arms
S_{3_8}	9y4757_1	Rectangular-Cubic Prism	Rocker Arms
S_{4_1}	SPACER_87_BACK_SHORTS	Solid Of Revolution	Discs
S_{4_2}	SPACER_87_FRONT_SHORTS	Solid Of Revolution	Discs
S_{4_3}	SPACER_87_LINK1_43_LINK1_42	Solid Of Revolution	Discs
S_{4_4}	SPACER_87_LINK_43_LINK_42	Solid Of Revolution	Discs
S_{4_5}	SPACER_87_LINK_43_LINK_42	Solid Of Revolution	Discs
S_{4_6}	SPACER_87_LINK_43_LINK_42	Solid Of Revolution	Discs
S_{4_7}	SPACER_87_LINK_43_LINK_42	Solid Of Revolution	Discs
S_{4_8}	SPACER_87_LINK_43_LINK_42	Solid Of Revolution	Discs
S_{5_1}	assem_spool_gear	Solid Of Revolution	Gear like Parts
S_{5_2}	GEAR43	Solid Of Revolution	Gear like Parts
S_{5_3}	gear_30	Solid Of Revolution	Spoked Wheels
S_{5_4}	GEAR_PRT	Solid Of Revolution	Gear like Parts
S_{5_5}	LOW_REV_GEAR	Solid Of Revolution	Gear like Parts
S_{5_6}	LOW_REV_GEAR_PRT	Solid Of Revolution	Gear like Parts
S_{5_7}	MS_SEC_GEAR	Solid Of Revolution	Gear like Parts
S_{5_8}	MS_SEC_GEAR_PRT	Solid Of Revolution	Gear like Parts
S_{6_1}	BEARING_SHAFT_57_LINK1_43	Solid Of Revolution	Long Pins
S_{6_2}	BEARING_SHAFT_57_LINK_42	Solid Of Revolution	Long Pins
S_{6_3}	BEARING_SHAFT_57_LINK_43	Solid Of Revolution	Long Pins
S_{6_4}	CLUSTER_COUNTERSHAFT	Solid Of Revolution	Long Pins
S_{6_5}	CLUSTER_COUNTERSHAFT_PRT	Solid Of Revolution	Long Pins
S_{6_6}	ganter_gn698	Solid Of Revolution	Round Change At End
S_{6_7}	MOTOR	Solid Of Revolution	Round Change At End
S_{6_8}	POTENTIOMETER	Solid Of Revolution	Round Change At End

Table A.2: Description of models in sets ξ_{sim_1} to ξ_{sim_6}.

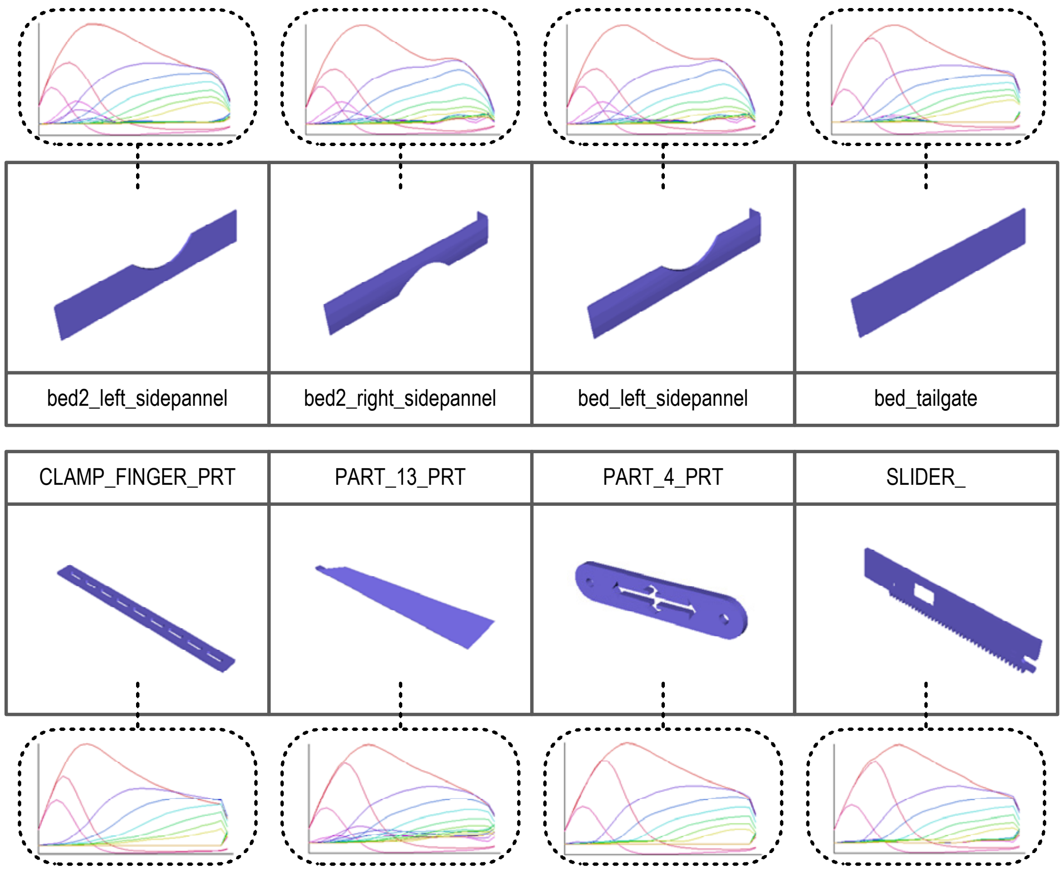
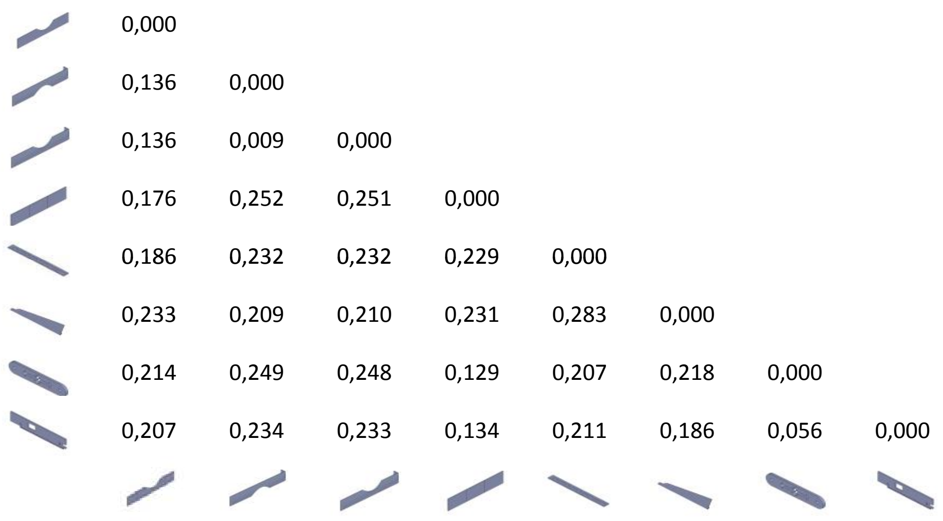


Figure A.8: Set ξ_{sim_1} of similar shapes and corresponding SHA signatures.



0,000							
0,136	0,000						
0,136	0,009	0,000					
0,176	0,252	0,251	0,000				
0,186	0,232	0,232	0,229	0,000			
0,233	0,209	0,210	0,231	0,283	0,000		
0,214	0,249	0,248	0,129	0,207	0,218	0,000	
0,207	0,234	0,233	0,134	0,211	0,186	0,056	0,000

Figure A.9: Distances between SHA feature vectors of models in ξ_{sim_1}.

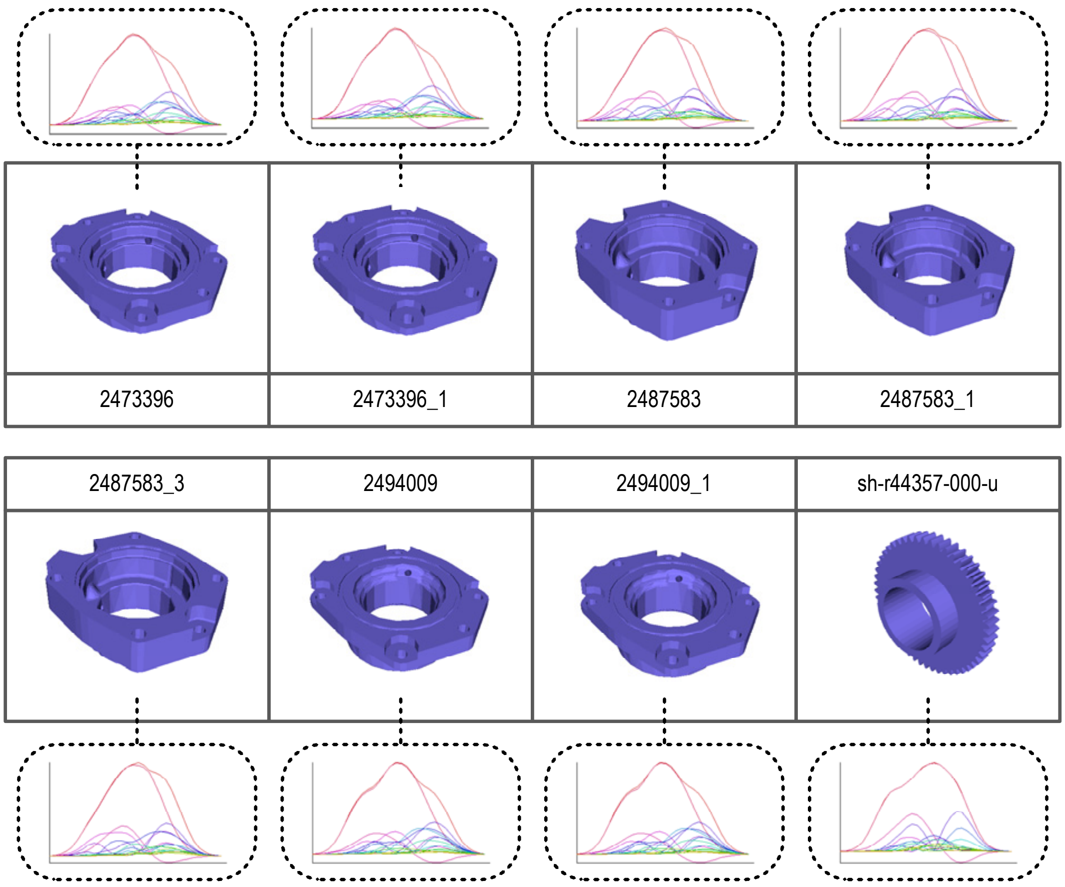
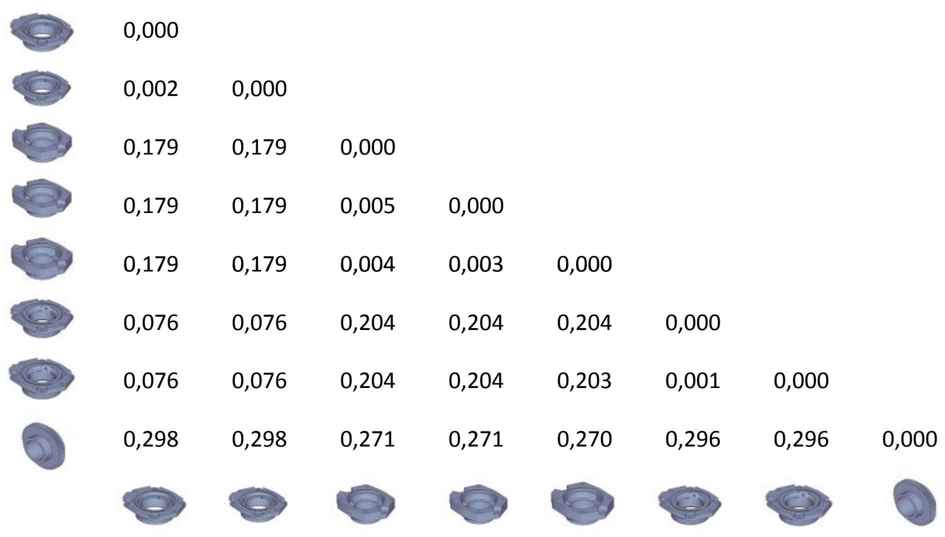


Figure A.10: Set ξ_{sim_2} of similar shapes and corresponding SHA signatures.



0,000							
0,002	0,000						
0,179	0,179	0,000					
0,179	0,179	0,005	0,000				
0,179	0,179	0,004	0,003	0,000			
0,076	0,076	0,204	0,204	0,204	0,000		
0,076	0,076	0,204	0,204	0,203	0,001	0,000	
0,298	0,298	0,271	0,271	0,270	0,296	0,296	0,000

Figure A.11: Distances between SHA feature vectors of models in ξ_{sim_2}.

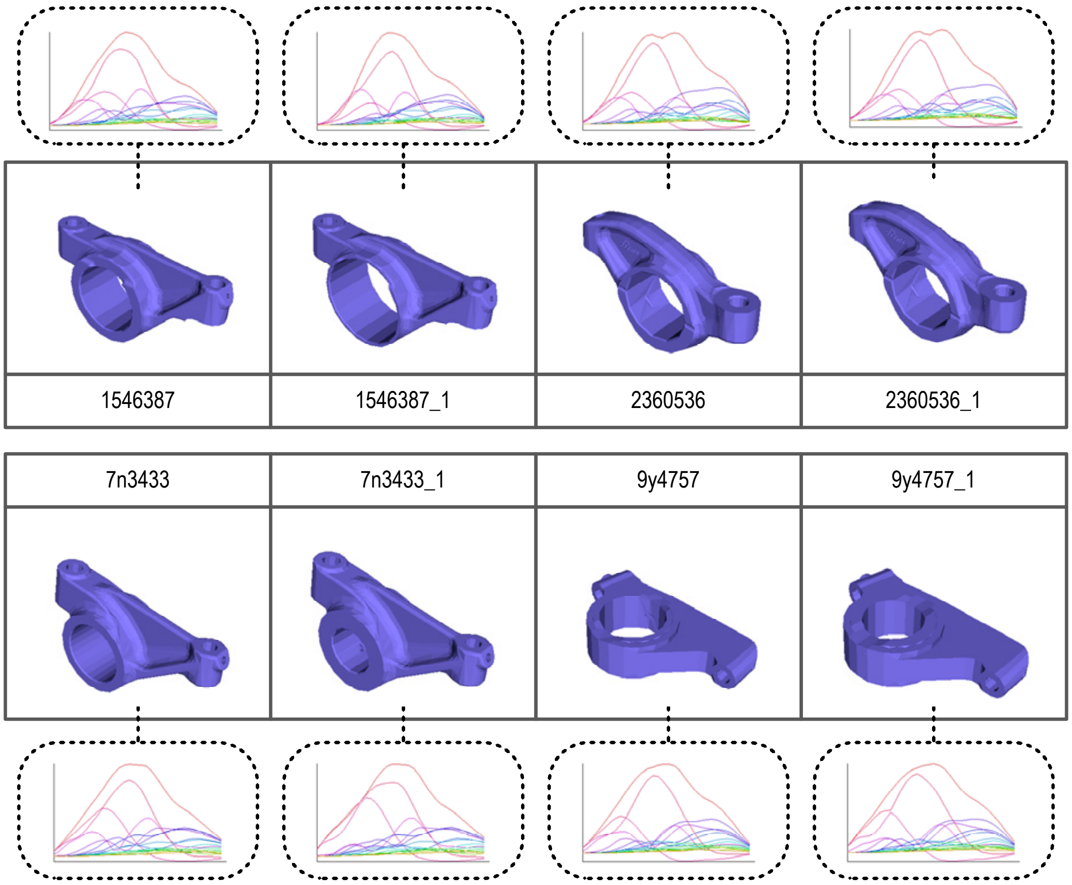
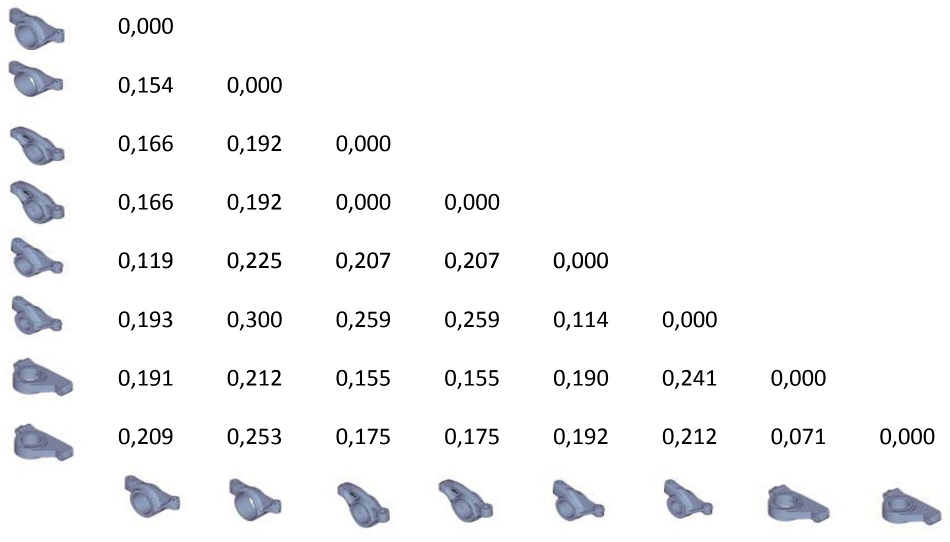


Figure A.12: Set ξ_{sim_3} of similar shapes and corresponding SHA signatures.



0,000							
0,154	0,000						
0,166	0,192	0,000					
0,166	0,192	0,000	0,000				
0,119	0,225	0,207	0,207	0,000			
0,193	0,300	0,259	0,259	0,114	0,000		
0,191	0,212	0,155	0,155	0,190	0,241	0,000	
0,209	0,253	0,175	0,175	0,192	0,212	0,071	0,000

Figure A.13: Distances between SHA feature vectors of models in ξ_{sim_3}.

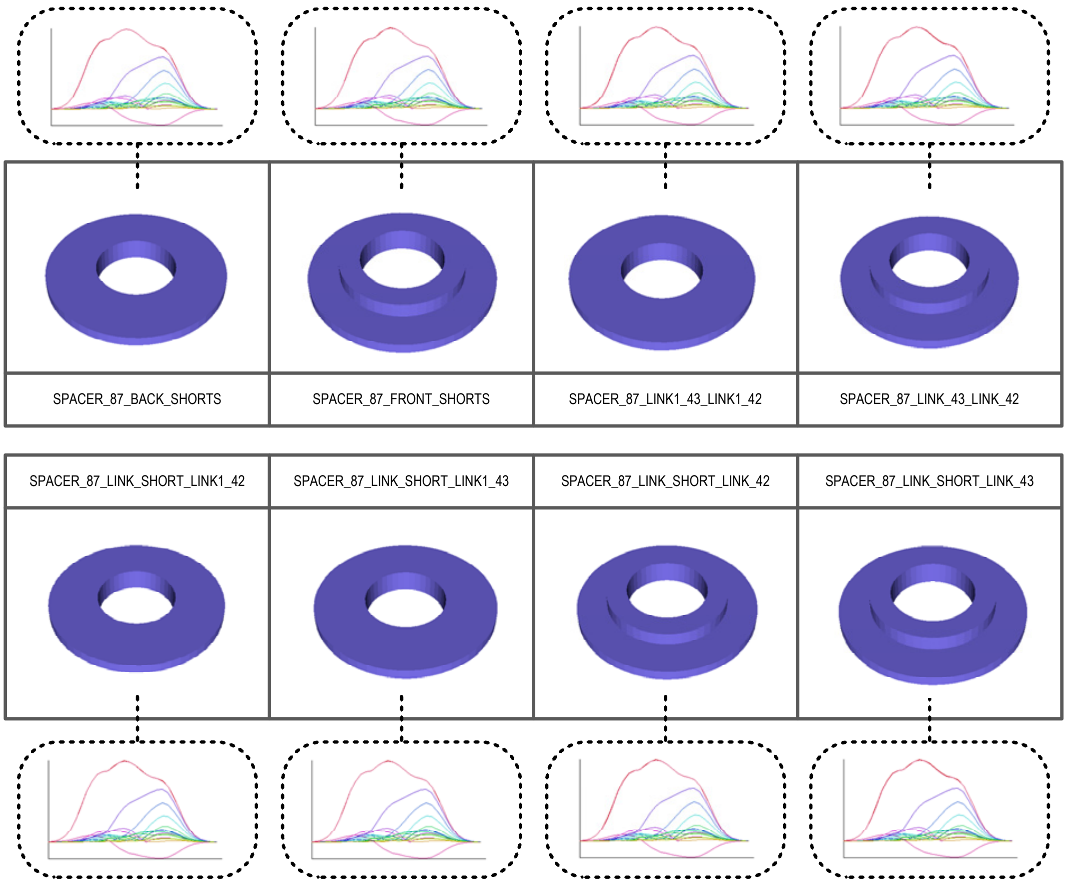
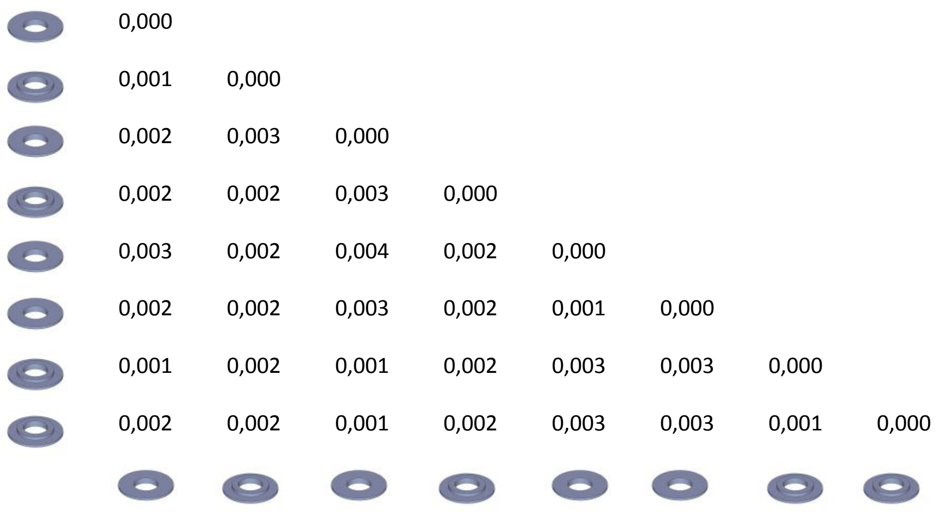

SPACER_87_BACK_SHORTS
SPACER_87_FRONT_SHORTS
SPACER_87_LINK1_43_LINK1_42
SPACER_87_LINK_43_LINK_42
SPACER_87_LINK_SHORT_LINK1_42
SPACER_87_LINK_SHORT_LINK1_43
SPACER_87_LINK_SHORT_LINK_42
SPACER_87_LINK_SHORT_LINK_43

Figure A.14: Set ξ_{sim_4} of similar shapes and corresponding SHA signatures.



	0,000							
	0,001	0,000						
	0,002	0,003	0,000					
	0,002	0,002	0,003	0,000				
	0,003	0,002	0,004	0,002	0,000			
	0,002	0,002	0,003	0,002	0,001	0,000		
	0,001	0,002	0,001	0,002	0,003	0,003	0,000	
	0,002	0,002	0,001	0,002	0,003	0,003	0,001	0,000

Figure A.15: Distances between SHA feature vectors of models in ξ_{sim_4}.

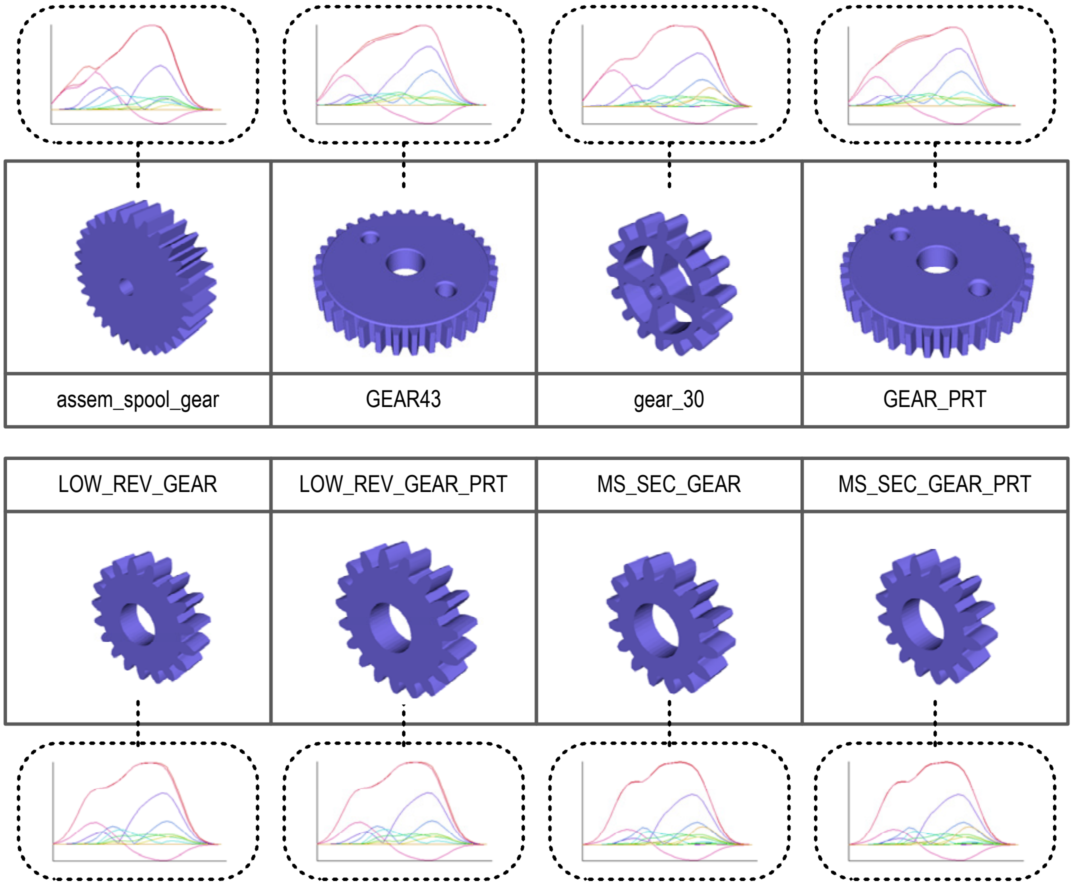
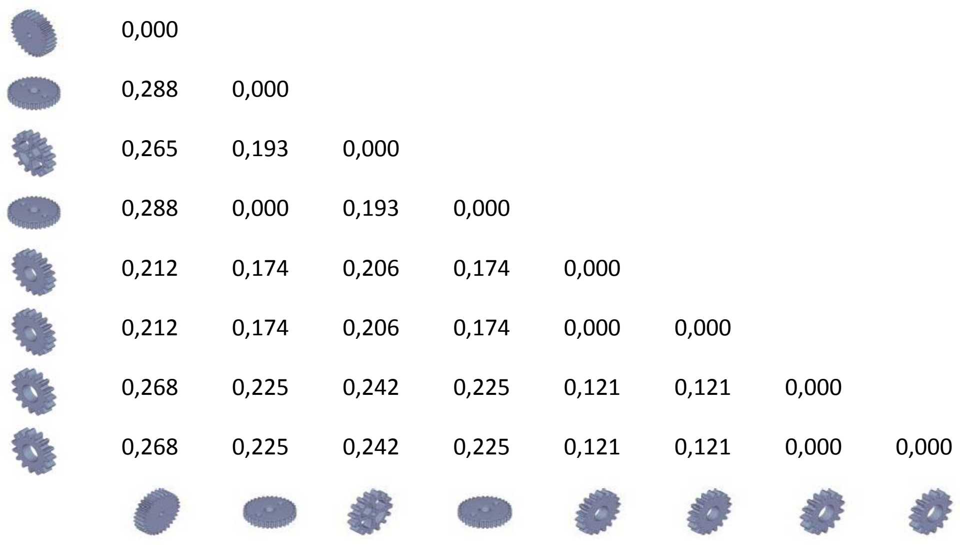


Figure A.16: Set ξ_{sim_5} of similar shapes and corresponding SHA signatures.



0,000							
0,288	0,000						
0,265	0,193	0,000					
0,288	0,000	0,193	0,000				
0,212	0,174	0,206	0,174	0,000			
0,212	0,174	0,206	0,174	0,000	0,000		
0,268	0,225	0,242	0,225	0,121	0,121	0,000	
0,268	0,225	0,242	0,225	0,121	0,121	0,000	0,000

Figure A.17: Distances between SHA feature vectors of models in ξ_{sim_5}.

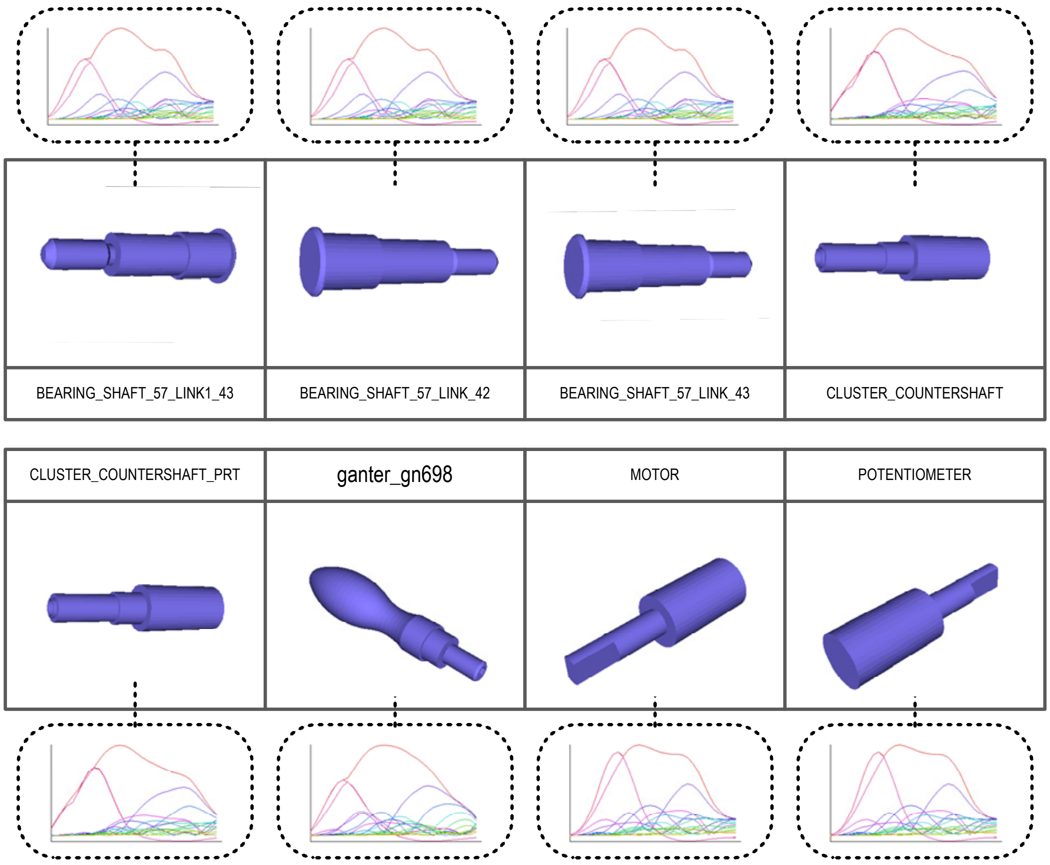


Figure A.18: Set ξ_{sim_6} of similar shapes and corresponding SHA signatures.

	0,000							
	0,002	0,000						
	0,003	0,003	0,000					
	0,260	0,261	0,261	0,000				
	0,260	0,261	0,261	0,000	0,000			
	0,235	0,236	0,236	0,252	0,252	0,000		
	0,298	0,298	0,299	0,276	0,276	0,288	0,000	
	0,298	0,298	0,299	0,276	0,276	0,288	0,000	0,000

Figure A.19: Distances between SHA feature vectors of models in ξ_{sim_6}.

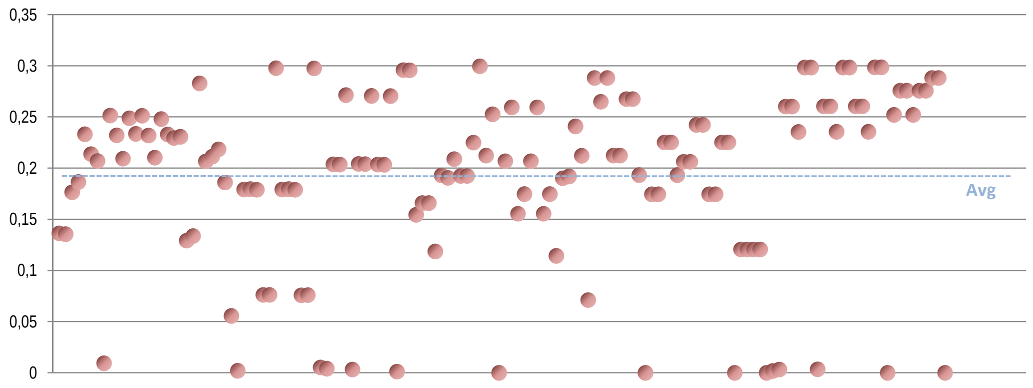


Figure A.20: Chart representing similarity values between models in each set ξ_{sim_i}.

From the analysis feature vector distances in sets ξ_{sim_i} we observed that when two extremely similar shapes are compared the similarity value is very low, as in $s_{4_{6,5}} = 0.00096$, while less similar shapes have higher values, as in $s_{2_{8,1}} = 0.298$. Indeed, in the tested sets the measured similarities ranged in a well defined interval

$$0.0 \leq s_{i_{j,k}} < 3.0.$$

This fact is clearly visible in the chart depicted in Figure A.20, where we show graphically the distances measured in this test, along with average similarity

$$\overline{s_{\xi_{sim}}} = 0.191.$$

A.3 Nearest Neighbours

It is not feasible to apply te test described in the previous section for every model in a collection containing more than eight hundred models because it requires human intervention. Thus, to validate the observed behaviour we performed a slightly different test. This third part of our study consisted in analysing the similarity of all models in the ESB collection with their nearest neighbours in the feature vector space.

To that end, we computed the SHA descriptor of every object in the collection and created with the corresponding feature vectors a multidimensional dataset. Then for each model we executed a linear k-NN search in this dataset with $k = 5$. The result $kNN(Q) = \{R_{1_Q}, \cdots, R_{5_Q}\}$ produced by the algorithm when applied to each model S_i

	#1	**#2**	**#3**	**#4**	**#5**
$0.0 \leq d < 0.1$	541	308	210	148	102
$0.1 \leq d < 0.2$	185	301	339	345	328
$0.2 \leq d < 0.3$	120	225	273	308	352
$0.3 \leq d < 0.4$	7	19	31	48	67
$d \geq 0.4$	0	0	0	4	4
$\overline{d}$	0.087	0.147	0.179	0.181	0.194
$var(d)$	0.00841	0.00834	0.00742	0.00717	0.00703

Table A.3: Statistics of feature vector distance, d_2, for the five nearest neighbours considering all models in the collection.

correspond to a set ξ_{kNN_i}. These sets contain the five nearest neighbours to each models of the ESB collection. Thus, we ended up with 853 sets of k-NN query results,

$$\xi_{kNN} = \{\xi_{kNN_1}, \cdots, \xi_{kNN_{853}}\}.$$

For each set ξ_{kNN_i} we estimated the similarity between the query and every corresponding result using Euclidean distance

$$s_{i_j} = d_2(FV_{S_i}, FV_{R_{j_i}})$$

where FV_{S_i} is the feature vector of the query model and $FV_{R_{j_i}}$ is the feature vector of the corresponding j^{th} query result, with $1 \leq j \leq 5$.

From the analysis of this data we observed that only four of the 4265 distances mesured were above 0.4 and that the majority (more than ninety percent) of the distances lay below 0.3, as shown in Table A.3. Moreover we also detected that the a large majority (almost seventy percent) of the similarity values for the first nearest neighbour were under 0.1. This means that a large number of objects in the collection have at least one very similar model, according to the geometric features considered by the SHA descriptor.

In Figure A.21 we depict the similarities between every model and each one of its five nearest neighbours. In this chart it is possible to observe that there are a large number of distances $d_2 \approx 0$, which correspond to extremely similar models in the collection, and the rest of the distances are distributed mostly in a wide vicinity of $d_2 = 0.2$, which correspond to similar models. The distances measured for each query result are shown in charts depicted in Figures A.22 to A.26.

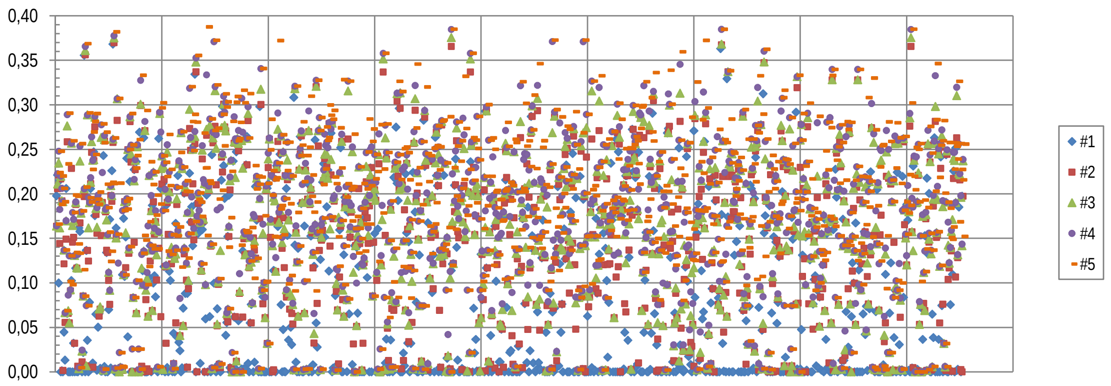


Figure A.21: Chart representing similarity values between models and corresponding top five nearest neighbours.

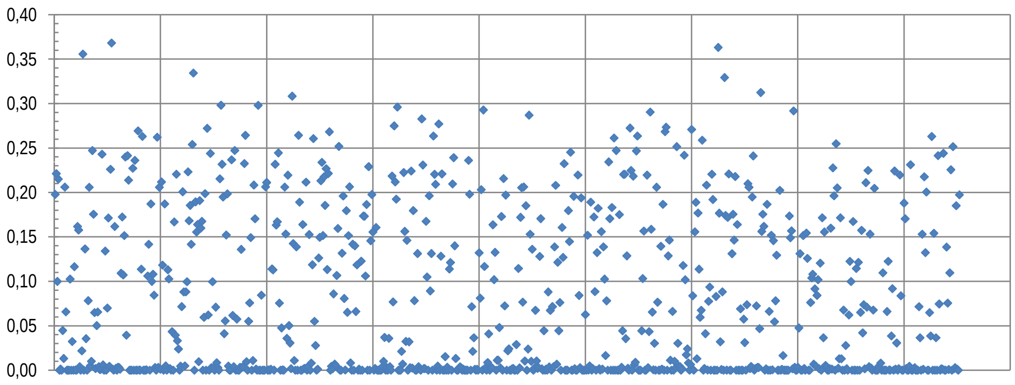


Figure A.22: Chart representing similarity values between all models and corresponding 1st nearest neighbour.

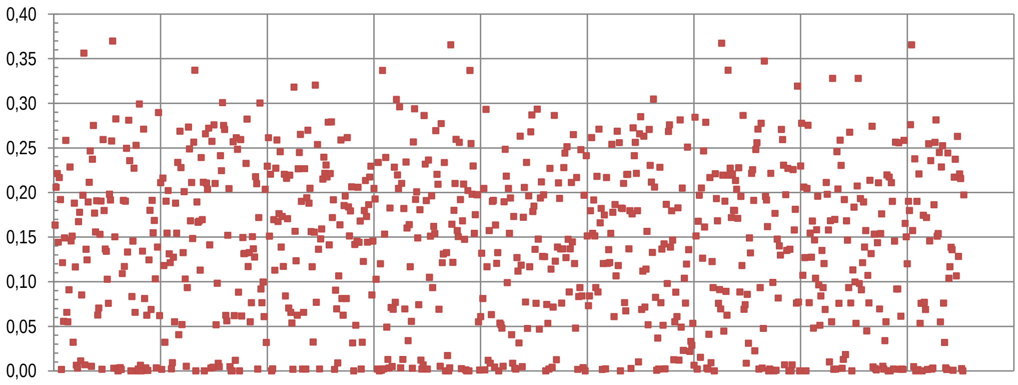


Figure A.23: Chart representing similarity values between all models and corresponding 2nd nearest neighbour.

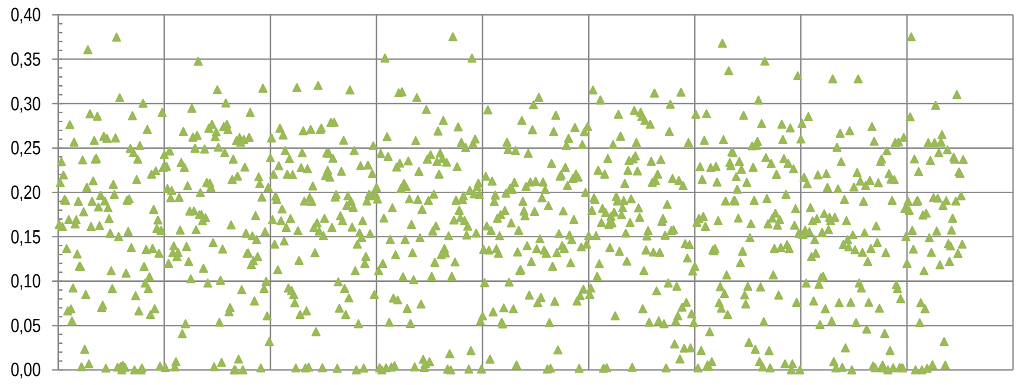


Figure A.24: Chart representing similarity values between all models and corresponding 3rd nearest neighbour.

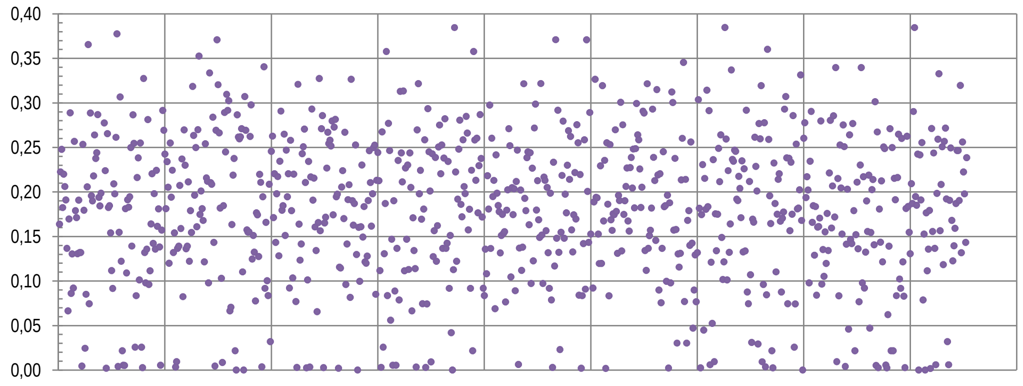


Figure A.25: Chart representing similarity values between all models and corresponding 4th nearest neighbour.

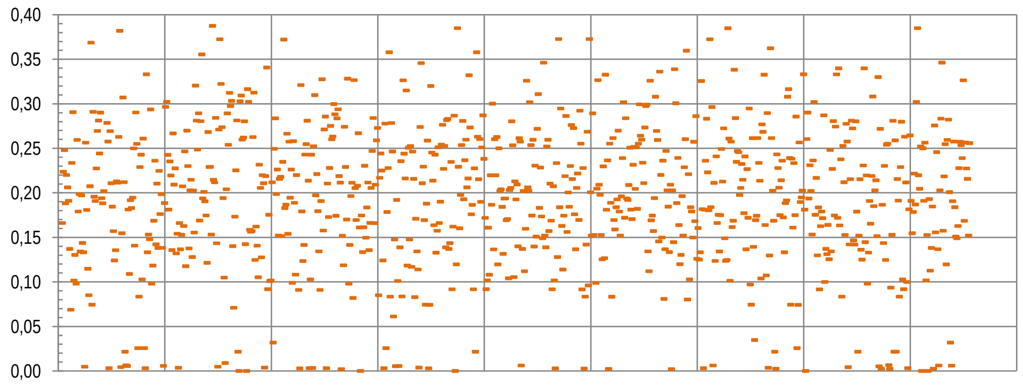


Figure A.26: Chart representing similarity values between all models and corresponding 5th nearest neighbour.

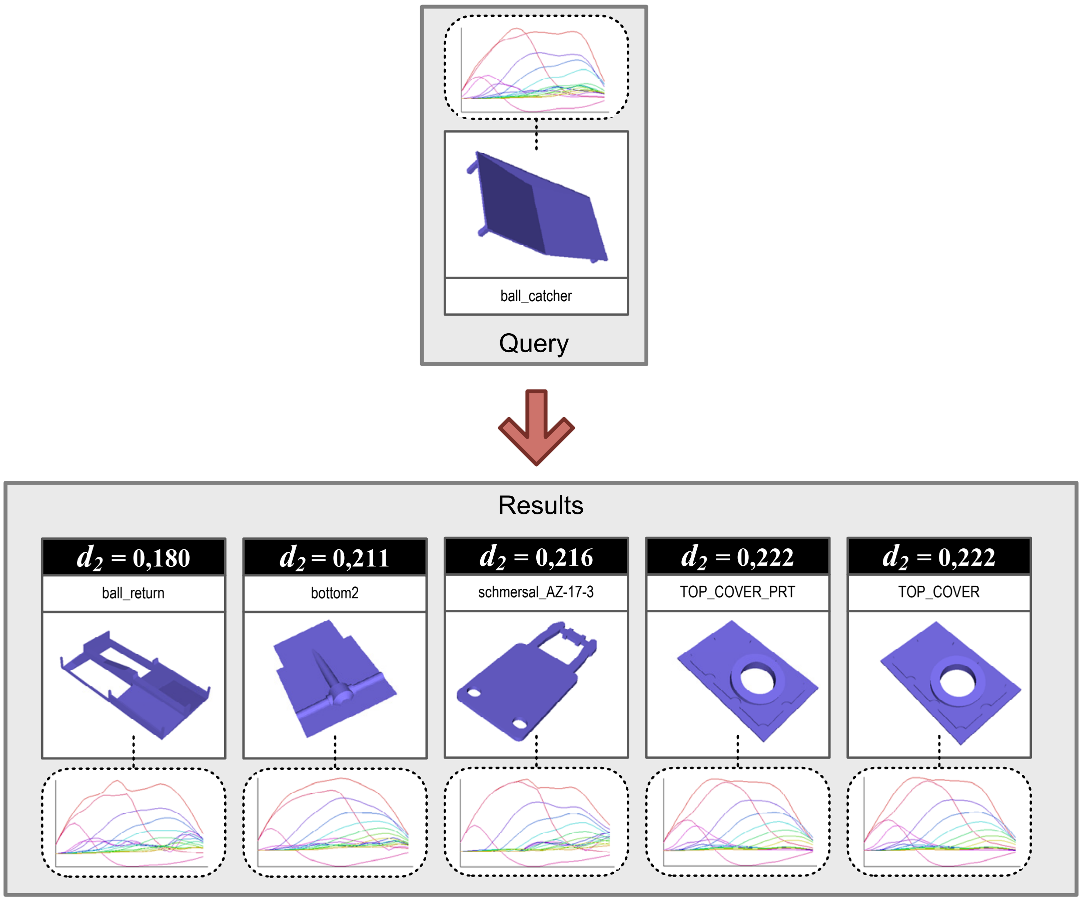


Figure A.27: Example of k-NN query, with $k = 5$, depicting the shapes, the corresponding SHA signatures and the Euclidean distances, d_2, from each result to the query.

To illustrate the obtained results, we randomly selected one model from the ESB and show here the five nearest neighbours for a query made with that object, as we did in this test for all models in the collection. In Figure A.27 we depict the query and the neighbours, presenting not only a view of each shape but also the respective SHA signature and the Euclidean distance, d_2 measured between the query and each result.

As expected, in this example is possible to check that the nearest neighbour is a geometrically similar shape and that the similarity decreases in the following neighbours. Additionally, the fourth and fifth neighbours correspond to two extremely similar objects. Thus, the distance to te given query is the same. This similarity is an example of the typical case that justifies the large number of almost absolute similarity observed in this test. There are several models with a twin in the ESB collection.

A.4 Summary

In this annex we presented with some detail a set of tests we made with the ESB collection in order to comprehend the behaviour of the SHA shape descriptor when applied to CAD models. The presented study was composed of three different parts, always focused on similarity among shapes but with distinct approaches. In the first part we study a set of random models. The second part focused on sets of geometrically similar models, according to the ESB search system. Finally, in the third part we examined the results produced by a k-NN search for each model in the collection.

From this study it was possible to perceive the behaviour of the SHA descriptor with the ESB collection. Namely regarding the range on which the distance among feature vector varies and how these distances are distributed. The results obtained during this study were used to define a reasonable range for the *similarity threshold*, an important parameter of the CAS/HFP algorithm proposed in this dissertation.

B
Building a Thesaurus from Primitive Shapes

The present dissertation introduces a technique to index collections of three-dimensional models. This technique transposes to the 3D context concepts widely used on text information retrieval. The central concept behind our approach to 3D object indexing is a shape thesaurus, equivalent to a lexicon in text collections.

Unfortunately, while words are explicit in text documents, the shapes that compose a model are not generally easy to determine, unless the object representation already contains such information. One example for such exception is when models in the collection are represented by primitive instancing. In this particular case, the shapes used to construct the object are explicit in the model, which greatly simplifies the thesaurus construction.

Despite the fact that such representation are used on very specific domains, we exemplify in this annex the construction of a shape thesaurus for a simple collection with ten models represented by primitive-instancing. These models were created using just four basic primitives depicted in Figure B.1: the box, the sphere, the cylinder, and the cone

The collection containing the models represented by primitive instancing using this

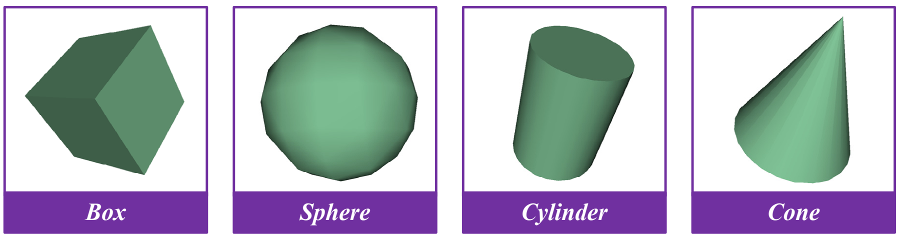


Figure B.1: Primitives used in the example collection.

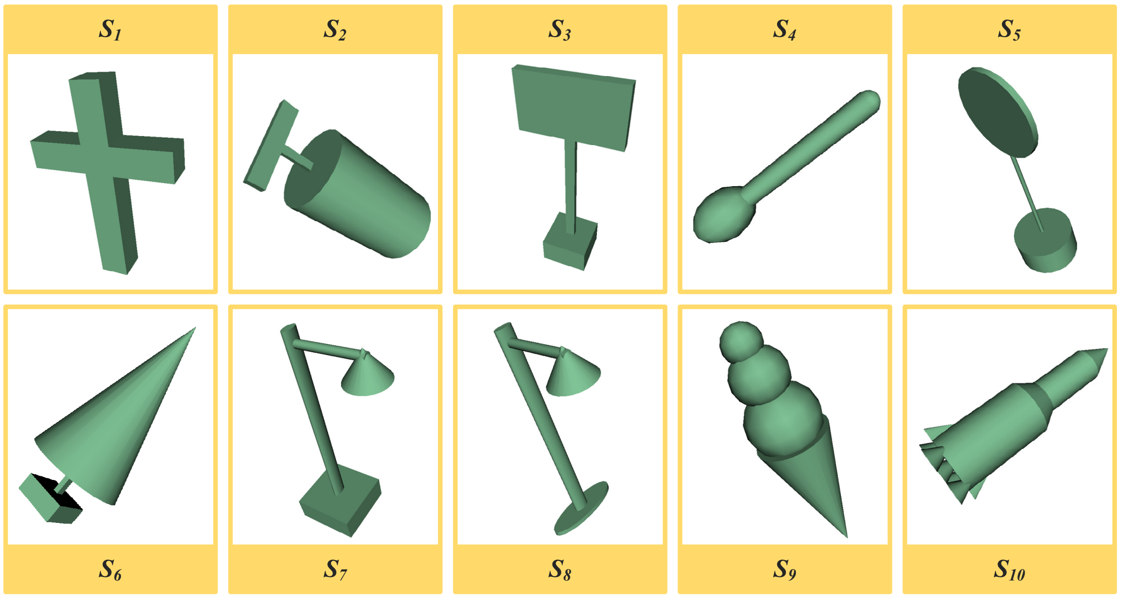


Figure B.2: Collection of models represented using primitive instancing.

primitive set is depicted in Figure B.2. To index this collection using a shape thesaurus we must, first of all, identify the building blocks for each model. Indeed, unlike the generic approach described in this dissertation, in this case no decomposition algorithm is necessary. The shapes that compose each model are explicitly available in the object representation.

Therefore, in this type of collection, the CAS/HFP algorithm and subsequent shape pool clustering, introduced in this dissertation, are not required to determine the terms of the shape thesaurus. Instead, the terms of the thesaurus are the primitive shapes used in the representation scheme. In this small example we use four primitives, but this number can be much higher in some cases, such as engineering models represented by primitive-instancing, where the shape primitives usually correspond to mechanical parts, like screws or bolts. Nevertheless, the working principle presented in this annex to create the shape thesaurus is applied similarly in these cases.

To identify the primitives used in each object it is sufficient to inspect the corresponding scene graph and check with shapes are assigned to the leaf nodes. These shapes are the primitives used in the model. In order to clearly show the primitives used by each object in the example collection we present in Figures B.3 to B.12 the corresponding scene graphs.

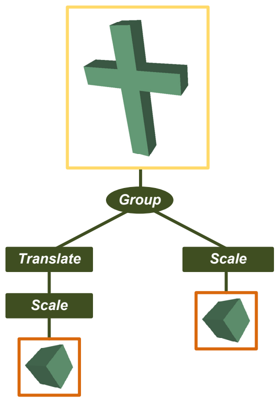


Figure B.3: Scene graph for model S_1.

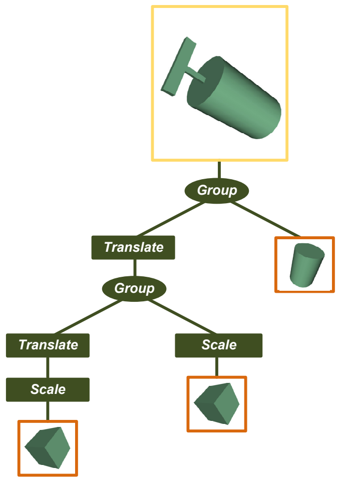


Figure B.4: Scene graph for model S_2.

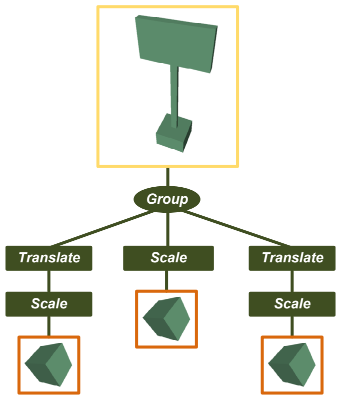


Figure B.5: Scene graph for model S_3.

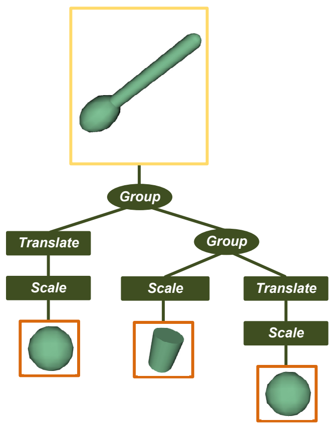


Figure B.6: Scene graph for model S_4.

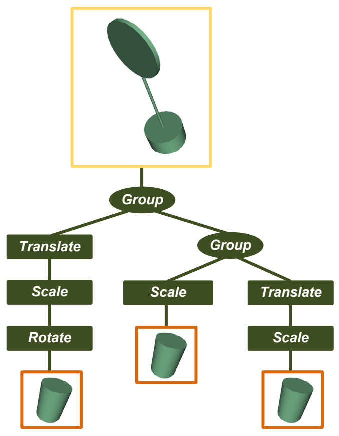


Figure B.7: Scene graph for model S_5.

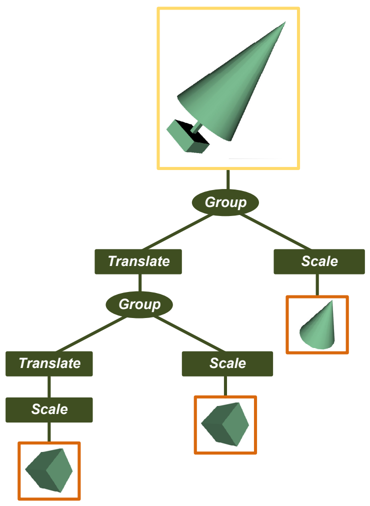


Figure B.8: Scene graph for model S_6.

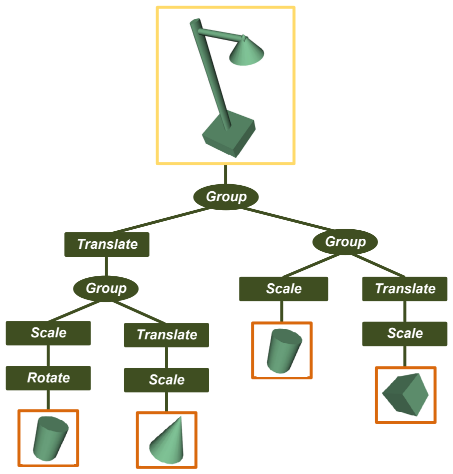


Figure B.9: Scene graph for model S_7.

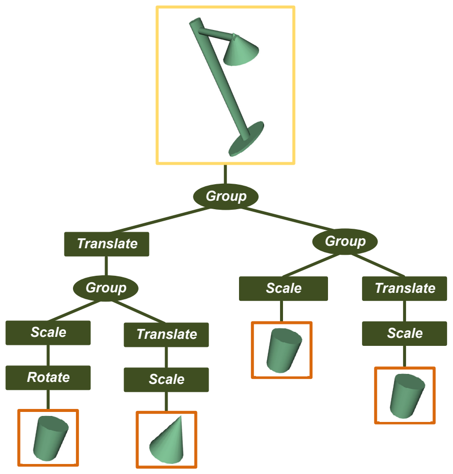


Figure B.10: Scene graph for model S_8.

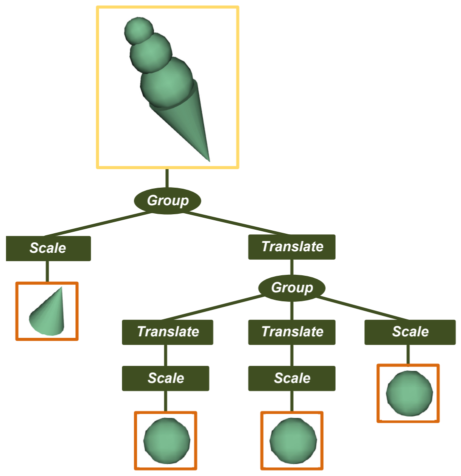


Figure B.11: Scene graph for model S_9.

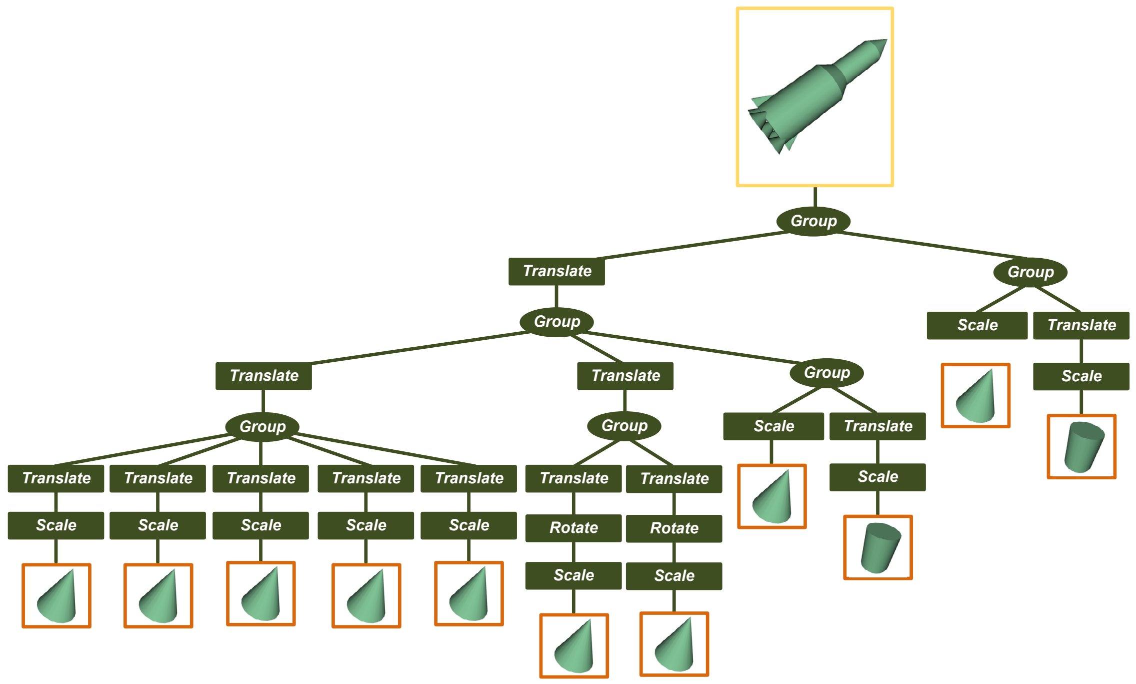


Figure B.12: Scene graph for model S_{10}.

As explained above, the shape thesaurus is constructed directly using as terms the primitive shapes. Following the terminology introduced in Chapter 4, the thesaurus in this example is defined as

$$\mathcal{T} = \{t_1, \cdots, t_4\},$$

where

$$t_1 = Box, t_2 = Sphere, t_3 = Cylinder, \text{ and } t_4 = Cone.$$

From the identification of primitives used in each object, which is directly obtained from the analysis of the model representation, we build the inverted index

$$\mathcal{I} = \{\langle t_1, \mathcal{O}_1\rangle, \langle t_2, \mathcal{O}_2\rangle, \langle t_3, \mathcal{O}_3\rangle, \langle t_4, \mathcal{O}_4\rangle\},$$

where

$$\mathcal{O}_1 = \{S_1, S_2, S_4, S_6, S_7\},$$
$$\mathcal{O}_2 = \{S_4, S_9\},$$
$$\mathcal{O}_3 = \{S_2, S_4, S_6, S_7, S_8, S_{10}\}, \text{ and}$$
$$\mathcal{O}_4 = \{S_6, S_7, S_8, S_9, S_{10}\}.$$

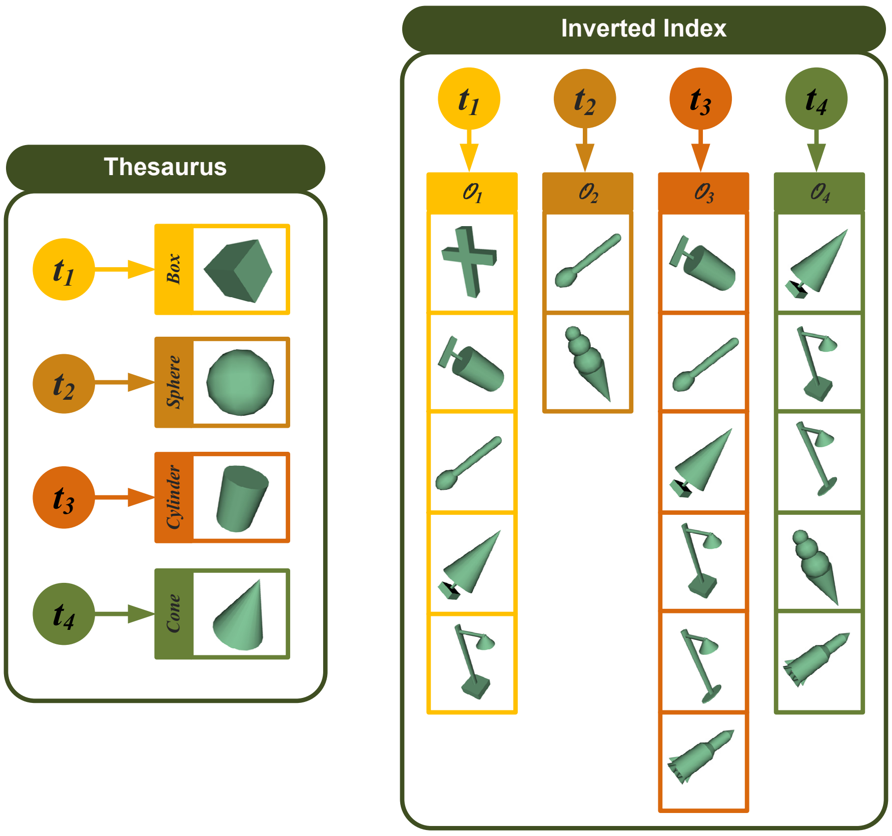


Figure B.13: Shape thesaurus and inverted index for the example collection.

For an easier interpretation, we depict, in Figure B.13, the computed thesaurus and corresponding inverted index. Although less formal than the definitions above, this illustration shows clearly how the models of the collection are indexed with a shape thesaurus.

While the example given in this annex refers a very small set of basic primitives, usually an application of primitive instancing is far more complex. Typically these primitives are parameterized on several properties. for instance, on a geometric modeling tool, a primitive object may be a regular prism with parameterized base face count. In this case, to specify a pentagonal prism, this parameter might be set to five.

Another example of primitive instancing can be found on some CAD modeling, where the primitives are complex objects. For instance, bolt or gear can be primitives and the parameters can range from the dimensions of the object to specific information such as the number of teeth in a gear.

The approach presented in this annex for thesaurus construction works with collections containing models represented as described in the previous paragraph. However, despite the wide use of primitive instancing for modeling purposes in some domains, models found in collections are rarely represented using primitive instancing and, when they are, they commonly use proprietary formats.

Therefore, although simple, this approach has a very limited number of applications. A generic approach should be able to classify collections independently of model representation. To that end, in our research work we consider that models are specified as polygonal meshes. This assumption is safe since other representations of 3D models can be easily converted to a polygonal mesh.

However, a polygonal mesh does not contain explicit information about the shapes that compose the model. Thus, the creation of a shape thesaurus and corresponding index is much more complex than the method presented above. In the present dissertation we proposed a solution for this issue.

C
Prototypes

During our research work we developed several software prototypes. From file format converters to the 3D shape retrieval with partial matching tool, we developed a wide variety of small applications. Some of these, although not directly related with our research goals provided relevant support to our work.

C.1 Support Prototypes

In this section we will describe the most noticeable support prototypes developed during our research. Although apparently simple and unrelated to our final goal, the prototypes herein presented were used not only in our research but also for other purposes by other researchers.

C.1.1 File conversion (and more)

This tool was initially devised as a simple format converter (from OFF to VRML). However, during our research work, lots of extra functionalities were added. Among these are the ability to write the model in other formats, as STL or PLY, or the capability to read from STL files as well.

Besides providing format conversion functionalities, the OFFtoVRML tool offers a lot of other extra features. It is possible to represent in the VRML file the face and vertex normals, as well as create a point cloud instead of a mesh and compute and include special shapes, such as convex hull or the minimum axis-aligned bounding box of the model.

C.1.2 Counting faces on meshes

To help us creating statistics on collections of 3D models we developed a prototype that counts the faces of surface meshes stored in STL (or OFF) files. The **FaceCounter** can be used to analyze a single model or a collection of models. The first is done by specifying the corresponding STL file and the second by specifying the folder that contains the collection.

Indeed, this a very small application that implements a single functionality available in many other much more complete (and complex) tools. However, when the only information needed is the number of polygons in a mesh there is no point in installing and using powerful tools for that.

C.1.3 Visual representation of a shape signature

To help us visualizing the shape signature of 3D objects we developed the **SHaVisRep** prototype. It produces visual representations of spherical harmonics (SHA) signatures. It receives as input the binary signature created by the executable for computing the SHA descriptor of 3D models provided by Kazhdan [78] and produce a visual representation for this descriptor. This visual representation can be a 3D column chart or a 2D line chart in VRML or BMP format, respectively, as depicted in Figure C.1.

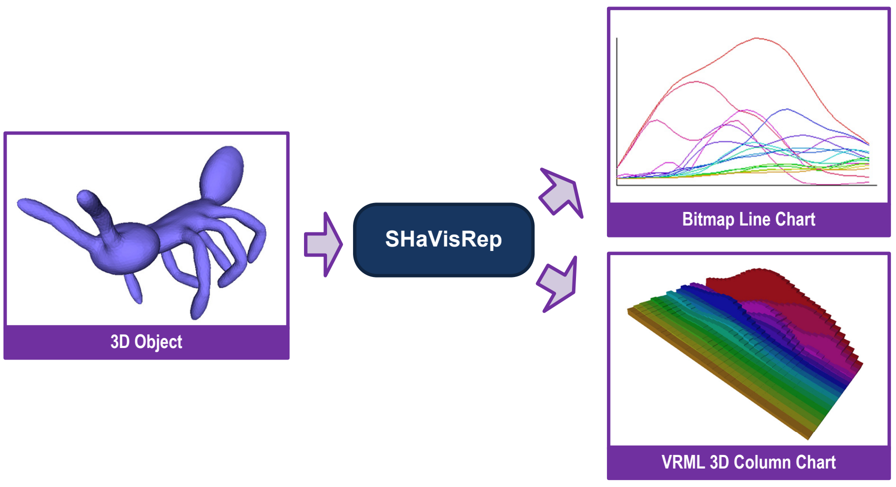


Figure C.1: Diagram of SHaVisRep prototype.

C.2 Research Prototypes

Although prototypes presented above are useful and have several applications, they do not cover the research work presented in this dissertation. To validate and evaluate our approach we developed a set of prototypes that implemented the algorithms and techniques devised during our investigation.

These research prototypes were created according to the methodology followed during our research, including the inter-algorithmic independence. Thus, we ended up with four prototypes corresponding to the decomposition, segmentation, thesaurus building and shape retrieval components of our approach.

C.2.1 Collection Decomposition

As we have described in this thesis dissertation, our approach to 3D object retrieval uses indexing concepts transposed from the textual information retrieval field. However, while text documents can be immediately decomposed into atomic elements (words), such does not occur with three-dimensional models. Unless models include explicitly its decomposition, it is not trivial to identify the segments that compose a generic 3D model. To compute the decomposition of models in collections we presented the Collection-Aware Decomposition algorithm.

The **CASdecomposer** prototype implements the CAS algorithm. This prototype decomposes all models in a collection according to given parameters. Trough these parameters is possible, among other functionalities, to tune the algorithm behavior, by setting the values of the similarity threshold, similar count threshold and maximum iteration count.

Receiving as input a set of 3D models, the **CASdecomposer** prototype produces two distinct outputs. The most important is the pool that stores the segments that compose the models in the collection and corresponding signatures, the *shape pool*. This pool is basically a dataset in the SHA signature space where each point corresponds to a segment. The other output is a set containing the decomposed models. Here, besides the representation used in our approach, based on HSM trees, the prototype can produce VRML files containing the decomposed models.

C.2.2 Segment Clustering

The output produced by the collection decomposition prototype serves as input for the **SegmentClustering** prototype. In text information retrieval the words are grouped into terms. Similarly, in our approach to 3D object retrieval, to compute a thesaurus for shapes the segments should also be clustered into terms. To that end, a partition for the *shape pool* has to be calculated.

The segment clustering prototype computes this partition trough an adapted implementation of the k-means clustering algorithm proposed by Kanungo [76]. The center of each cluster in the partition corresponds to a term in the thesaurus. Thus, the **SegmentClustering** prototype produces a file containing the partition of the *shape pool*. This file contains, for each cluster in the partition, the coordinates of the cluster center and the identification of the segments belonging to that group.

C.2.3 Thesaurus Building

Based on the output produced by the two research prototypes presented above, the **TBuilder** prototype computes the *shape thesaurus* and corresponding inverted index. To that end it assembles the clusters present the *shape pool* partition into a set of thesaurus terms. This set of terms constitutes the *shape thesaurus*.

Having the *shape thesaurus* properly computed, the prototype identifies the models in the collection that contain segments associated with each term in the thesaurus. This information is compiled into an *inverted index* that maps the terms in the thesaurus with models in the collection.

C.2.4 Shape Retrieval

Our last, but not less important, research prototype implements the retrieval part of our approach. The current version of the **3DSRetrieval** prototype is based on query-by-example methodology. Thus, to query the collection the user provides a 3D model, more specifically an OFF file containing the model to use as query. The prototype will search for models that contain segments similar to that query, performing part-in-whole matching using our thesaurus-based approach.

The **3DSRetrieval** prototype starts by loading into memory the necessary indexing data and then allows innumerous queries to be submitted. The indexing data to load is comprised by the dataset containing all the segments in the *shape pool*, which will be used to order the retrieved results, the *shape thesaurus* and the *inverted index*.

The indexing data should fit in computer physical memory to provide efficient retrieval experience. Nevertheless, this easily occurs since, according to our experiences, the size of such structures for a collection with around a thousand elements does not exceed the fifteen megabytes.

For each query submitted to the *3DSRetrieval* prototype, a list of results is compiled and presented properly ordered to the user. This list of results contains some additional information, such as the signature distance between the closer matching segment and the query. This additional information, useless for a common user, is of major importance for research purposes.

C.3 Conclusions

In this annex we listed the most relevant prototypes developed during our research. The three support prototypes provide functionalities such as file format converting, face counting on meshes and producing visual representation of signatures. The four research prototypes implement the algorithms and techniques that comprise our approach to object retrieval.

The prototypes herein presented are freely available in the internet. This way we expect to improve the dissemination of our research work and to provide helpful tools for researchers and general users worldwide.

Bibliography

[1] Hervé Abdi. Distance. *Encyclopedia of Measurements and Statistics*, 2:280–284, 2007.

[2] Milton Abramowitz and Irene A. Stegun. Probability functions. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*, pages 925–964, 1974.

[3] Ceyhun Burak Akgül. 3d shape descriptors and similarity learning. Phd thesis progress report, Boḡaziçi University, Istanbul, June 2006.

[4] Ceyhun Burak Akgül. *Density-based Shape Descriptors and Similarity Learning for 3D Object Retrieval.* PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris an Boḡaziçi University, Istanbul, December 2007.

[5] Ceyhun Burak Akgul, Bulent Sankur, Francis Schmitt, and Yucel Yemez. Multivariate density-based 3d shape descriptors. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 3–12, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Ceyhun Burak Akgül, Bülent Sankur, Yücel Yemez, and Francis Schmitt. A framework for histogram-induced 3d descriptors. In *Proceedings of the 14th. European Signal Processing Conference (EUSIPCO'06)*, Florence, Italy, September 2006.

[7] Ceyhun Burak Akgül, Bülent Sankur, Yücel Yemez, and Francis Schmitt. Density-based 3d shape descriptors. *EURASIP Journal on Advances in Signal Processing*, 2007:Article ID 32503, 16 pages, 2007. doi:10.1155/2007/32503.

[8] Michael R. Anderberg. *Cluster analysis for applications.* Academic Press, New York, 1973.

[9] Mihael Ankerst, Gabi Kastenmï¿½ller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *SSD99: Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 207–226, London, UK, 1999. Springer-Verlag.

[10] Tarik Filali Ansary, Mohamed Daoudi, and Jean-Philippe Vandeborre. 3d-models search engine from photos. In *Proceedings of ACM International Conference on Image and Video Retrieval (CIVR 2007)*, Amsterdam, The Netherlands, July 2007.

[11] Tarik Filali Ansary, Mohamed Daoudi, and Jean-Philippe Vandeborre. A bayesian 3D search engine using adaptive views clustering. *IEEE Transactions on Multimedia*, 9(1):78–88, January 2007.

[12] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 106–113, New York, NY, USA, 1998. ACM.

[13] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[14] Jürgen Assfalg, Alberto Del Bimbo, and Pietro Pala. Retrieval of 3d objects by visual similarity. In *MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 77–83, New York, NY, USA, 2004. ACM Press.

[15] Marco Attene. 'efpisoft'. http://efpisoft.sourceforge.net/, October 2006.

[16] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.*, 22(3):181–193, 2006.

[17] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[18] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 714–725, 1987.

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

[20] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[21] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[22] F.C. Bernstein, T.F. Koetzle, G.J. Williams, E.F. Meyer Jr, M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank. a computer-based archival file for macromolecular structures. *European Journal of Biochemistry*, 80:319–324, 1977.

[23] Dmitriy Bespalov, Cheuk Yiu Ip, William C. Regli, and Joshua Shaffer. Benchmarking cad search techniques. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling (SPM '05)*, pages 275–286, New York, NY, USA, 2005. ACM Press.

[24] Dmitriy Bespalov, William C. Regli, and Ali Shokoufandeh. Reeb graph based shape retrieval for cad. *Proceedings of DETC'03 2003 ASME Design Engineering Technical Conferences*, September 2003.

[25] Dmitriy Bespalov, William C. Regli, and Ali Shokoufandeh. Local feature extraction and matching partial objects. *Computer-Aided Design*, 38(9):1020–2037, September 2006.

[26] Dmitriy Bespalov, Ali Shokoufandeh, William C. Regli, and Wei Sun. Scale-space representation of 3d models and topological matching. In *Proceedings of the* 8^{th} *ACM symposium on Solid Modeling and Applications*, pages 208–215, New York, NY, USA, June 2003. ACM Press.

[27] Silvia Biasotti. Topological techniques for shape understanding. Central European Seminar on Computer Graphics, Bratislava, Slovakia, 2001.

[28] Silvia Biasotti and Marco Attene. Shrec08 entry: Report of the stability track on watertight models. In *IEEE International Conference on Shape Modeling and Applications 2008*, New York, USA, June 2008. IEEE Computer Society.

[29] Silvia Biasotti, Ennio De Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Size functions for 3d shape retrieval. In Konrad Polthier and Alla Sheffer, editors, *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, pages 239–242, Cagliari, Sardinia, Italy, June 2006. Eurographics Association.

[30] Silvia Biasotti, Simone Marini, Michela Mortara, Giuseppe Patanï¿½, Michela Spagnuolo, and Bianca Falcidieno. 3d shape matching through topological structures. In Ingela Nystrï¿½m, Gabriella Sanniti di Baja, and Stina Svensson, editors, *Discrete Geometry for Computer Imagery*, volume 2886 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 2003.

[31] Silvia Biasotti, Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. Subpart correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

[32] Irving Bierderman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, April 1987.

[33] Alberto Del Bimbo and Pietro Pala. Content-based retrieval of 3d models. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):20–43, 2006.

[34] Miroslaw Bober. Mpeg-7 visual shape descriptors. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 11(6):716–719, 2001.

[35] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[36] Benjamin Bustos, Daniel Keim, Dietmar Saupe, and Tobias Schreck. Content-based 3d object retrieval. *IEEE Computer Graphics and Applications*, 27(4):22–27, 2007.

[37] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan V. Vranić. Feature-based similarity search in 3d object databases. *ACM Computing Surveys*, 37(4):345–387, 2005.

[38] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan V. Vranić. An experimental effectiveness comparison of methods for 3d similarity search. *International Journal on Digital Libraries*, 6(1):39–54, 2006.

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

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

[41] Vittorio Castelli. Multidimensional indexing structures for content-based retrieval. *Image Databases*, 2002.

[42] Bernard Chazelle, David P. Dobkin, Nadia Shouraboura, and Ayellet Tal. Strategies for polyhedral surface decomposition: an experimental study. In *Proceedings of the eleventh annual symposium on Computational geometry (SCG '95)*, pages 297–305, New York, NY, USA, 1995. ACM Press.

[43] Bernard Chazelle and Leonidas Palios. Decomposition algorithms in geometry. *Algebraic Geometry and its Applications*, pages 419–447, 1994.

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

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

[46] Michele d'Amico, Patrizio Frosini, and Claudia Landi. Using matching distance in size theory: A survey. *International Journal of Imaging Systems and Technology*, 16(5):154–161, 2006.

[47] Pedro A. de Alarcón, Alberto D. Pascual-Montano, and José M. Carazo. Spin images and neural networks for efficient content-based retrieval in 3d object databases. In *CIVR '02: Proceedings of the International Conference on Image and Video Retrieval*, pages 225–234, London, UK, 2002. Springer-Verlag.

[48] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[49] Chitra Dorai and Anil K. Jain. Shape spectrum based view grouping and matching of 3d free-form objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(10):1139–1146, 1997.

[50] Helin Dutagaci, Bulent Sankur, and Yucel Yemez. Transform-based methods for indexing and retrieval of 3d objects. In *3DIM '05: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 188–195, Washington, DC, USA, 2005. IEEE Computer Society.

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

[52] Christos Faloutsos. *Searching Multimedia Databases by Content.* Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[53] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice in C.* Addison-Wesley Professional, second edition, August 1995.

[54] Manuel J. Fonseca. *Sketch-Based Retrieval in Large Sets of Drawings.* PhD thesis, Instituto Superior Técnico / Universidade Técnica de Lisboa, 07 2004.

[55] Thomas Funkhouser, Michael Kazhdan, Patrick Min, and Philip Shilane. Shape-based retrieval and analysis of 3d models. *Communications of the ACM*, 48(6):58–64, 2005.

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

[57] Thomas Funkhouser and Philip Shilane. Partial matching of 3D shapes with priority-driven search. In *Symposium on Geometry Processing*. Eurographics, June 2006.

[58] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, 30(2):170–231, 1998.

[59] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, 2006.

[60] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, New York, NY, USA, 2001. ACM.

[61] John Gero and Vladimir Kazakov. On measuring the visual complexity of 3d objects. *International Journal of Design Sciences and Technology*, 12(1):35–44, 2004.

[62] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. Shape retrieval contest 2007: Watertight models track, 2007.

[63] C. Harris and M.J.Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–152, 1988.

[64] Ho Min Wong Hee Kap Ahn, Nikos Mamoulis. A survey on multidimensional access methods. Technical Report UU-CS-2001-14, Institute of Information and Computing Sciences, Utrecht University, 2001.

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

[66] Donald D. Hoffman and Manish Singh. Salience of visual parts. *Cognition*, 63(1):29–78, April 1997.

[67] B. K. P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686, 1984.

[68] Google Inc. 'google sketchup'. http://sketchup.google.com, 2006.

[69] Google Inc. '3d warehouse'. http://sketchup.google.com/3dwarehouse, 2007.

[70] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37:509–530, April 2005.

[71] Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, Natraj Iyer, and Karthik Ramani. Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 39(9):939–953, September 2006.

[72] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

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

[74] Andrew Edie Johnson and Martial Hebert. Recognizing objects by matching oriented points. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 684, Washington, DC, USA, 1997. IEEE Computer Society.

[75] S. B. Kang and K. Ikeuchi. The complex egi: A new representation for 3-d pose determination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(7):707–721, 1993.

[76] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[77] Michael Kazhdan, Bernard Chazelle, David Dobkin, Thomas Funkhouser, and Szymon Rusinkiewicz. A reflective symmetry descriptor for 3d models. *Algorithmica*, 38(1):201–225, 2003.

[78] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In Leif Kobbelt, Peter Schroder, and Hugues Hoppe, editors, *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 156–164, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[79] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Symmetry descriptors and 3d shape matching. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 115–123, New York, NY, USA, 2004. ACM Press.

[80] S. Kirkpatrick, Jr. Gelatt, C. D., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[81] Jan J. Koenderink and Andrea J. van Doorn. Surface shape and curvature scales. *Image Vision Comput.*, 10(8):557–565, 1992.

[82] Jacob Kogan. *Introduction to Clustering Large and High-Dimensional Data*. Cambridge University Press, New York, NY, USA, 2007.

[83] Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM Journal on Computing*, 37(3):757–782, 2007.

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

[85] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proceeedings of the Second International Conference on Computer Vision*, pages 238–249, December 1988.

[86] Marc Levoy. The digital michelangelo project. In *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling*, pages 2–11, Ottawa, Ont., Canada, October 1999.

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

[88] Yi Liu, Hongbin Zha, and Hong Qin. The generalized shape distributions for shape matching and analysis. In *Proceedings of the IEEE International Conference on*

Shape Modeling and Applications 2006 (SMI'06), page 16, Washington, DC, USA, 2006. IEEE Computer Society.

[89] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar 1982.

[90] Kuiyang Lou, Sunil Prabhakar, and Karthik Ramani. Content-based three-dimensional engineering shape search. *International Conference on Data Engineering*, page 754, 2004.

[91] P.C. Mahalanobis. On the generalized distance in statistics. *National Institute of Sciences of India*, 2(1):49–55, 1936.

[92] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, July 2008.

[93] Simone Marini, Biasotti Silvia, and Falcidieno Bianca. Partial matching by structural descriptors. In Tim Crawford and Remco C. Veltkamp, editors, *Content-Based Retrieval*, number 06171 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[94] Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. From exact to approximate maximum common subgraph. In Luc Brun and Mario Vento, editors, *Graph-Based Representations in Pattern Recognition*, volume 3434 of *Lecture Notes in Computer Science*, pages 263–272. Springer, 2005.

[95] Jiri Matoušek. On approximate geometric k-clustering. *Discrete and Computational Geometry*, 24(1):61–84, December 2000.

[96] Michela Mortara and Giuseppe Patanè. Shape-covering for skeleton extraction. *International Journal of Shape Modeling*, 8(2):139–158, 2002.

[97] John Novatnack, Ko Nishino, and Ali Shokoufandeh. Extracting 3d shape features in discrete scale-space. In *Proceeding of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 946–953, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[98] Marcin Novotni and Reinhard Klein. 3d zernike descriptors for content based shape retrieval. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 216–225, New York, NY, USA, 2003. ACM Press.

[99] Ryutarou Ohbuchi, Takahiro Minamitani, and Tsuyoshi Takei. Shape-similarity search of 3d models by using enhanced shape functions. In *TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003*, page 97, Washington, DC, USA, 2003. IEEE Computer Society.

[100] Ryutarou Ohbuchi, Kunio Osada, Takahiko Furuya, and Tomohisa Banno. Salient local visual features for shape-based 3d model retrieval. In *IEEE International Conference on Shape Modeling and Applications 2008 (SMI 2008)*, pages 93–102, June 2008.

[101] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, 2002.

[102] Panagiotis Papadakis, Ioannis Pratikakis, Stavros Perantonis, and Theoharis Theoharis. Efficient 3d shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recogn.*, 40(9):2437–2452, 2007.

[103] E. Paquet and M. Rioux. Nefertiti: a query by content software for three-dimensional models databases management. In *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, volume 0, page 345, Washington, DC, USA, 1997. IEEE Computer Society.

[104] Eric Paquet, Marc Rioux, Anil Murching, Thumpudi Naveen, and Ali Tabatabai. Description of shape information for 2-d and 3-d objects. *Signal Processing: Image Communication*, 16:103–122, September 2000.

[105] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3d shapes. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH*, pages 549–559, New York, NY, USA, 2006. ACM Press.

[106] Georges Reeb. Sur les points singuliers dúne forme de pfaff completement integrable ou dúne fonction numï¿$\frac{1}{2}$rique. *Comptes Rendus des séances de lÁcadémie des sciences*, 222:847–849, 1946.

[107] William C. Regli, Cheryl Foster, Erik Hayes, Cheuk Yiu Ip, David McWherter, Mitchell Peabody, Yuriy Shapirsteyn, and Vera Zaychik. National design repository project: A status report. In *International Joint Conferences on Artificial Intelligence (IJCAI) and AAAI/SIGMAN Workshop on AI in Manufacturing Systems*, Seattle, Washington, USA, August 2001.

[108] William C Regli and Daniel M Gaines. A repository for design, process planning and assembly. *Computer-Aided Design*, 29(12):895–905, December 1997.

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

[110] Salvador Ruiz-Correa, Linda G. Shapiro, and Marina Meila. A new paradigm for recognizing 3-d object shapes from range data. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1126, Washington, DC, USA, 2003. IEEE Computer Society.

[111] Dietmar Saupe and Dejan V. Vranić. 3d model retrieval with spherical harmonics and moments. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 392–397, London, UK, 2001. Springer-Verlag.

[112] Jau-Ling Shih, Chang-Hsing Lee, and Jian Tang Wang. A new 3d model retrieval approach based on the elevation descriptor. *Pattern Recogn.*, 40(1):283–295, 2007.

[113] Philip Shilane and Thomas Funkhouser. Selecting distinctive 3d shape descriptors for similarity retrieval. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, page 18, Washington, DC, USA, 2006. IEEE Computer Society.

[114] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3d surfaces. *ACM Transactions on Graphics*, 26(2):7, 2007.

[115] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape Modeling International*, June 2004.

[116] Yoshihisa Shinagawa and Tosiyasu L. Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11(6):44–51, 1991.

[117] A. Shokoufandeh, S. Dickson, K. Siddiqi, and S. Zucker. Indexing Using a Spectral Encoding of Topological Structure. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pages 2491–2497. IEEE Computer Society, 1999.

[118] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[119] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Trecvid: evaluating the effectiveness of information retrieval tasks on digital video. In *Proceedings of the 12th annual ACM international conference on Multimedia (MULTIMEDIA '04)*, pages 652–655, New York, NY, USA, 2004. ACM Press.

[120] Michela Spagnuolo, Silvia Biasotti, Bianca Falcidieno, and Simone Marini. Structural descriptors for 3d shapes. In Tim Crawford and Remco C. Veltkamp, editors, *Content-Based Retrieval*, number 06171 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[121] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. *Proceedings of the Shape Modeling International 2003*, page 290, 2003.

[122] Motofumi T. Suzuki, Toshikazu Kato, and N. Otsu. A similarity retrieval of 3d polygonal models using rotation invariant shape descriptors. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 2946–2952, Nashville, TN, USA, 2000. IEEE Computer Society.

[123] Motofumi T. Suzuki, Yoshitomo Yaginuma, and Yasutaka Shimizu. A partial shape matching technique for 3d model retrieval systems. In *ACM SIGGRAPH 2005 Posters*, page 128, New York, NY, USA, 2005. ACM Press.

[124] Motofumi T. Suzuki, Yoshitomo Yaginuma, and Yuji Y. Sugimoto. A 3d model retrieval system for cellular phones. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3846–3851. IEEE Computer Society, October 2003.

[125] Motofumi T. Suzuki, Yoshitomo Yaginuma, Tsuneo Yamada, and Yasutaka Shimizu. A partial shape matching method for 3d model databases. In *Proceedings of the Ninth IASTED International Conference on Software Engineering and Applications (SEA2005)*, pages 389–394, Phoenix, USA, November 2005. ACTA Press.

[126] Motofumi T. Suzuki, Yoshitomo Yaginuma, Tsuneo Yamada, and Yasutaka Shimizu. A 3d model retrieval based on combinations of partial shape descriptors. In *Proceedings of IEEE North American Fuzzy Information Processing Society Annual Conference (NAFIPS 2006)*, pages 273–278, Montreal, Canada, June 2006. IEEE Computer Society.

[127] S. Szykman, R.D. Sriram, C. Bochenek, J.W. Racz, and J. Senfaute. Design repositories: engineering design's new knowledge base. *Intelligent Systems and Their Applications, IEEE*, 15(3):48–55, 2000.

[128] Johan W.H. Tangelder and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications*, 39:441–471, 2008. 10.1007/s11042-007-0181-0.

[129] Tony Tung and Francis Schmitt. Augmented reeb graphs for content-based retrieval of 3d mesh models. In *Proceedings of the Shape Modeling International 2004 (SMIÓ4)*, pages 157–166, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[130] Tony Tung and Francis Schmitt. The augmented multiresolution reeb graph approach for content-based retrieval of 3d shapes. *International Journal of Shape Modeling*, 11(1):91–120, 2005.

[131] Jean-Philippe Vandeborre, Vincent Couillet, and Mohamed Daoudi. A practical approach for 3D model indexing by combining local and global invariants. In *1st IEEE International Symposium on 3D Data Processing Visualization Transmission (3DPVT'02)*, Padova, Italy, June, 19-21 2002.

[132] Dejan V. Vranić. An improvement of rotation invariant 3d-shape based on functions on concentric spheres. In *Proceedings of the 2003 IEEE International Conference on Image Processing (ICIP 2003)*, volume 3, pages 757–760, Barcelona, Spain, September 2003.

[133] Dejan V. Vranić. *3D Model Retrieval*. PhD thesis, University of Leipzig, Germany, June 2004.

[134] Dejan V. Vranić and Dietmar Saupe. 3d model retrieval. In *Proceeedings of Spring Conference on Computer Graphics and its Applications (SCCG2000)*, pages 89–93, Budmerice, Slovakia, May 2000.

[135] Dejan V. Vranić and Dietmar Saupe. 3d shape descriptor based on 3d fourier transform. In *Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001)*, pages 271–274, Budapest, Hungary, 2001.

[136] Dejan V. Vranić and Dietmar Saupe. Description of 3d-shape using a complex function on the sphere. In *Proceeedings of 2002 IEEE International Conference on Multimedia and Expo (ICME '02)*, volume 1, pages 177–180, 2002.

[137] Liu Wei and He Yuanjun. 3d shape similarity comparison using multi-level spherical moments. *Computer-Aided Design Applications*, 3(1–4):307–314, 2006.

[138] Eric W. Weisstein. Orthotope. From MathWorld–A Wolfram Web Resource.

[139] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Comput. Sci. Eng.*, 4(4):10–21, 1997.

[140] Meng Yu, Indriyati Atmosukarto, Wee Kheng Leow, Zhiyong Huang, and Rong Xu. 3d model retrieval with morphing-based geometric and topological feature maps. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, 02:656, 2003.

[141] Hagit Zabrodsky, Shmuel Peleg, and David Avnir. Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1154–1166, 1995.

[142] T. Zaharia and F. J. Prêteux. 3D shape-based retrieval within the MPEG-7 framework. In Edward R. Dougherty and Jaakko T. Astola, editors, *Proceedings of the SPIE conference on Nonlinear Image Processing and Pattern Analysis XII*, volume 4304, pages 133–145, May 2001.

[143] T. Zaharia and F. J. Prêteux. Hough transform-based 3D mesh retrieval. In L. J. Latecki, D. M. Mount, A. Y. Wu, and R. A. Melter, editors, *Proceedings of the SPIE conference on Vision Geometry X*, volume 4476, pages 175–185, November 2001.

[144] Jiqi Zhang, Hau-San Wong, and Zhiwen Yu. 3d model metrieval based on volumetric extended gaussian image and hierarchical self organizing map. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 121–124, New York, NY, USA, 2006. ACM Press.

[145] Emanuel Zuckerberger, Ayellet Tal, and Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733–743, 2002.