International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Thesaurus-based 3D Object Retrieval with Part-in-Whole Matching

Alfredo Ferreira · Simone Marini · Marco Attene
Manuel J. Fonseca · Michela Spagnuolo
Joaquim A. Jorge · Bianca Falcidieno

Received: date / Accepted: date

Abstract Research in content-based 3D retrieval has already started, and several approaches have been proposed which use in different manner a similarity assessment to match the shape of the query against the shape of the objects in the database. However, the success of these solutions are far from the success obtained by their textual counterparts.

A major drawback of most existing 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. 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.

In this paper we present a novel approach to 3D shape retrieval that uses a collection-aware shape decomposition combined with a shape thesaurus and inverted indexes to describe and retrieve 3D models using part-in-whole matching. The proposed method clusters similar segments obtained trough a multilevel decomposition of models, constructing from such partition the shape thesaurus. Then, to retrieve a model containing a sub-part similar to a given query, instead of looking on a large set of subparts or executing partial matching between the query and all models in the collection, we just perform a fast global matching between the query and the few entries in the thesaurus. With this technique we overcame the time complexity problems associated with partial queries in large collections.

Keywords 3D Shape Retrieval · Part-in-whole Matching · Thesaurus · Segmentation

Alfredo Ferreira · Manuel J. Fonseca · Joaquim A. Jorge
Department of Computer Science and Engineering
INESC-ID/IST/Technical University of Lisbon
E-mail: {alfredo.ferreira, mjf, jaj}@inesc-id.pt

Simone Marini · Michela Spagnuolo · Bianca Falcidieno
Instituto di Matematica Applicata e Tecnologie Informatiche
Consiglio Nazionale delle Ricerche
E-mail: {simone, michi, bianca}@ge.imati.cnr.it

1 Introduction

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 the models in the collection, it is not easy to find the sought model. Aware of this, during the last decade, researchers proposed several approaches to retrieve 3D models based on shape similarity. Some of these content-based retrieval systems are able to find a model in a database from a sketched query, a set of keywords or using query-by-example. However, results produced by such systems are far from the successful query results obtained by their textual counterparts.

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. Existing 3D search engines work mostly by comparing the complete models: 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 approaches allow searching for a model by supplying as a query only a part of the desired model. However, a few of these solutions rely on representing only a few sub-parts of the model and not the complete model [1–3]. Indeed, considering a small set of sampled or even 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.

To overcome this problem, our research focused on a novel approach to 3D shape retrieval with part-in-whole matching, as proposed by Suzuki *et al.*[4, 5]. We aim 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 [6] 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 the models in that collection.

Basically, we intend to accomplish the reverse of the modelling by example method proposed by Funkhouser *et al.* [7]. In their work, existing objects in a database were used to model a new object, which will be composed by subparts of other models in the database. Indeed, authors claim that most objects can be assembled by interchanging parts with others when a sufficiently large database is considered. Following this rationale, in this work we assume that models in a collection share common parts and propose using these common parts in a similar manner text information retrieval system use to words in documents.

To classify and retrieve large collections of documents, successful text retrieval systems rely on word thesauri [8]. Likewise, we believe that through the use of an effective shape decomposition mechanism and a shape thesaurus with inverted indexes we will be able to describe and retrieve 3D models through whole-to-part searches. Indeed, use of thesauri in shape retrieval is not a recent idea [9] but it is still adopted by recent search engines. Indeed, while the basic concepts of the thesaurus remain unchanged, recent investigation focus, for instance, on improving thesaurus construction [10, 11].

Similarly to word thesaurus in text retrieval, which contains the list of words that compose the vocabulary used in the documents, the proposed shape thesaurus will contain the shapes that compose the indexed models. Conceptually, the shape thesaurus should consist of a list of segments extracted from the models in the collection and the inverted index consists on a list of terms in the thesaurus with the corresponding lists of occurrences in the indexed models. This approach will allow us to take advantage of some well known techniques from text information retrieval, such as the term frequency and inverse document frequency to rank the relevance of every subpart in the database.

However, despite the success of word thesauri in text documents [12], while words are explicit in text, subpart identification in a three-dimensional object is not trivial and the success of our approach depends greatly on the quality of this identification. Therefore, the core of our research focus is on devising techniques for model segmentation suitable to be used in as a basis for building a shape thesaurus.

Moreover, the number of words in a lexicon is usually measured in millions [13] but, to ensure time efficiency, our shape thesaurus should have just around a few thousand terms. Thus, instead of using all segments extracted from the models, we must group them according to their geometric properties and use 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. Therefore, we investigated techniques for clustering the shapes resulting from the models decomposition in order to create a shape thesaurus for a given collection.

Indexing of three-dimensional shapes using a thesaurus was a challenge we faced during our research, especially due to the novelty of our ideas. However, since our approach is based on the same fundament used in traditional textual information retrieval, we transposed concepts and techniques from this data to our field of work, i.e. 3D shapes. Thus, instead of a lexicon holding words from documents in the collection, we devised a lexicon where terms are the prototype for each cluster produced by the partitioning of the set of shapes issued by collection decomposition. Additionally, we implemented an inverted index to store the mapping between the terms and the corresponding models in the collection.

Based on the indexing methodology we adopted, the retrieval process is relatively straightforward. Essentially, we combine techniques inspired by textual information retrieval and 3D shape matching ones. From this combination we attained a solution that provides

efficient retrieval from large collections of models using partial queries.

In the remain of this paper we will present an overview of our methodology to construct shape thesauri for 3D model collections, and then describe the concepts and methods behind it. Next we will present the indexing and retrieval techniques we developed to support the partial queries on collections of models. Finally we will present a few conclusions and suggest some steps for future research.

2 Related Work

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 [14] 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.

Later, in 2001, Thomas Funkhouser and his team released the Princeton 3D model search engine [15]. This system is now the best known solution for shape retrieval, indexing more that thirty six thousand 3D models. Its authors even claim that they have developed the search engine to be the "Google^{TM} for 3D models" [16], although this might be considered an overstatement.

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 [17]. 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.

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 [18] 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. A simple interface is integrated in this search engine, allowing users to retrieve 3D models by drawing 2D silhouettes.

Vranić deployed a web-based retrieval system for 3D models [19], to serve as a proof-of-concept to methods and tools for content-based search for 3D-mesh models proposed during his PhD research. 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.

Based on two distinct approaches to description and matching of 3D objects, Assfalg *et al.* developed a content-based retrieval system for 3D shapes [20]. Using a curvature map of the shape surface, the authors propose, to subdivide the map into a grid of rectangular tiles and then use these to compute a shape histogram. In another approach, the map is segmented into regions of homogeneous curvature, and regions are described with weighted walkthroughs. This search engine allows users to perform queries-by-example through a web interface on a database of three-dimensional models.

More recently, in 2007, researchers from the FOX-MIIRE group released a on-line search engine for 3D content [21]. Their search engine implements the adaptive views clustering technique, proposed by the authors to index 3D models based on two-dimensional views. Besides the good retrieval results, this method has a unique feature when compared with previous approaches: this search engine is the first that accepts 3D models retrieval from photos [22] and that can be reached through a mobile device. Indeed, the idea of incorporating a 3D model retrieval system in a mobile device was proposed by Suzuki *et al.* [23]. They developed an experimental 3D shape retrieval system for cellular phones where users can search for a model similar to a given example.

Additionally, Qin Lv *et al.* introduced a toolkit that supports the construction of content-based similarity search systems [24]. The Ferret toolkit is a content-based similarity search engine for generic, multi-feature object representations. It was designed to solve the similarity search problem in high-dimensional spaces. Indeed, this solution can be used to successfully construct content-based similarity search systems for audio recordings, digital images, 3D shape models and genomic microarry data.

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 [25].

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.

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 [1]. 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 [26] 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.

A more recent approach [2] performs partial matching by comparing the salient features of two objects. This method first defines a set of local descriptors on the surface of the object. Each local descriptor, associated to a surface point of the object, consists of a quadric patch providing an approximation of the surface around the point. After the local descriptors have been computed, the method identifies the salient feature describing the shape of the object by grouping the local descriptors according to curvature variance and intensity. Finally each salient features is associated to an indexing vector and inserted in to a geometric hash table. These indices are used to access parts of objects through the hash table.

Also in [3] important regions of the surface object are used to perform partial matching. A region of an object is considered important if it is useful to discriminate the object with respect to other objects of a given data-set. Distinctive regions are identified by randomly sampling the points on the object surface and by associating a shape descriptor to each sampled point. As descriptor, the authors propose the harmonics shape descriptor [27] computed at four different scales. Each descriptor is compared with all the others descriptors at the same scale in a data-set of objects grouped into object classes and the distinctiveness of each point is obtained from the discounted cumulative gain of the ranked list obtained from the comparison by measuring how often objects of the same class appear near the front of the list.

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 [28] 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 [29] 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 [30] it is based on a flexible matching framework based on the consolidated Reeb graph theory. This theory allows the use of different functions to analyze the shape, each one able to identify different relevant sub-parts of the object, finally geometric attributes are considered to drive the matching algorithm. In this case the structural descriptor is coded as attributed and directed graph and the matching is achieved through the construction of a common sub-graph (possibly not connected) between the two input graphs, that highlights where two shapes are similar or dissimilar.

Another recent method based on Reeb graphs has been proposed in [31]. This approach uses the Reeb graph to decompose the shape model into regions (Reeb charts) whose topology corresponds to a disk or an annulus. A shape signature, based on a parametrization technique, is extracted from each chart and a graph is generated by joining adjacent regions of the model. Shape comparison between models is obtained through a graph matching algorithm that reduce the computational complexity by matching nodes whose corresponding regions have the same topology, that is disks or annulus.

Suzuki *et al.* proposed [4,5] 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 [32]. To segment the shape, 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. 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. In our approach, described in following section we propose a novel methodology that, while also relying on shape decomposition, overcome the retrieval time complexity.

3 Approach Overview

Our research path aims at transposing to 3D retrieval the matching and indexing approaches widely-used in text information retrieval. We propose using a shape thesaurus for model classification and indexing, similarly to what happens in text documents. Unfortunately, while words can be easily extracted from documents, subpart identification in a 3D object is an harder task, unless the object representation already contains such information.

One example of 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 subpart identification. However, such representation is not commonly used on the myriad of existing domain-specific and generic collections that rely on several distinct forms of model representation [33]. Nevertheless, these different representations can always be converted or approximated to a more generic one, such as a polygonal mesh. Aiming for a representation independent approach, in the present work we assume that a 3D model is represented as a non-manifold polygonal mesh.

Thus, a crucial task of our technique consists on identifying object decomposition from its 3D mesh. The difficulties of this task came not only from its computational complexity but also from the ambiguity of such decomposition and the requirement of an automatic segmentation of entire collections.

Moreover, to be useful for our approach, the segments obtained through model decomposition must then be meaningfully clustered. The terms in the thesaurus will spring up from these clusters and will be associated to all models containing the segments in the corresponding cluster. Each term will be represented by a prototype, to be used for matching with the query when looking for a model in the collection. Therefore, computing shape clusters and corresponding prototypes is another issue that we tackled in our research.

Summarising, our approach to 3D shape retrieval with part-in-whole matching faces three major challenges, besides the ones shared with the global matching approaches.The first challenge is to devise an effective and efficient decomposition of models into subparts. The second is to attain a meaningful clustering of these segments and corresponding cluster representation. The third is to find an effective way to index the extracted information that allows fast and accurate search.

The system 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. A schematic overview of our system is depicted in Figure 1, illustrating the two parts of the system and corresponding components.

The classification part is composed by three individual components. The *Decomposer* processes the model collection, decomposing its models into segments that are stored in a so-called *Shape Pool*. The algorithm behind the *Decomposer* is described in Section 4.These segments are analysed by *Segment Clustering* component in order to compute a partition for the *Shape Pool*. The proposed methodology to cluster the *Shape Pool* is described in Section 5.1. Finally, this partition is used by the *Thesaurus Builder* to create the shape thesaurus and corresponding inverted index, as described in Section 5.2.

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 partially similar to the query. In Section 5.3 we explain our retrieval technique which allows fast model retrieval with part-in-whole matching.

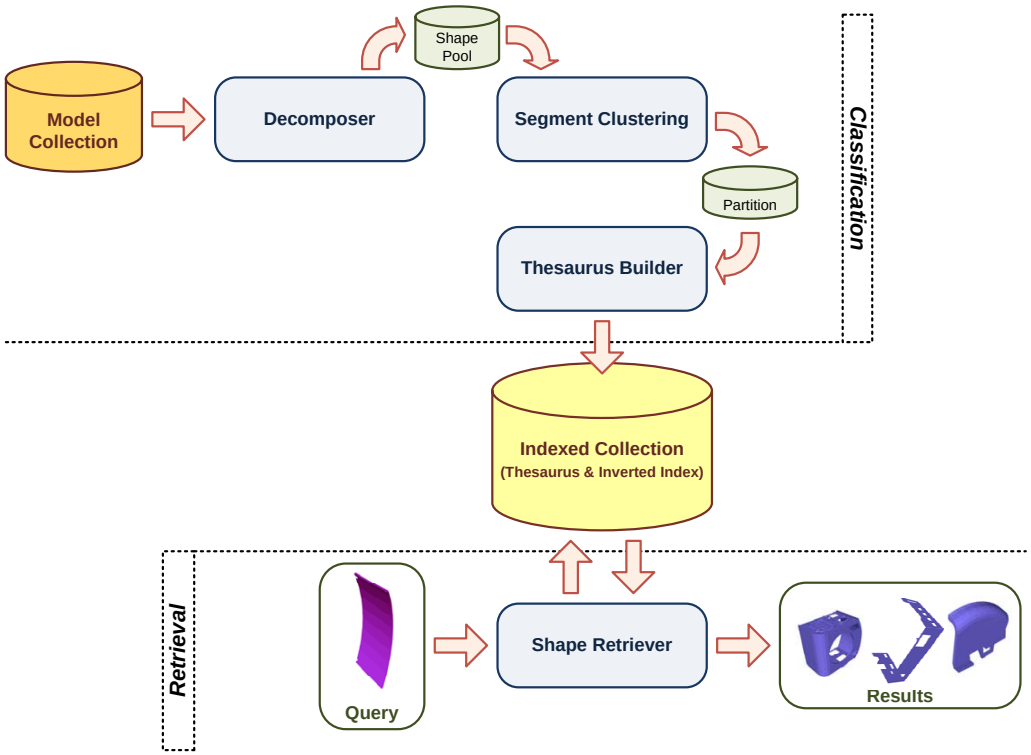

Fig. 1 Schematic overview of our approach.

4 Collection-Aware Segmentation

To segment models in the collection, we need a technique that provides not only automatic segmentation of all models, but also produces segments useful for the thesaurus construction. Several approaches to 3D shape decomposition have been published, with recognised success in some domain-specific models, such as articulated characters. However, independently of the methodology used, all these approaches only consider the model that should be decomposed, ignoring the context where it lies, namely the other models in the same collection. Inspired by a similar principle, in the present work we devised a novel approach to shape decomposition: the collection-aware shape segmentation, a method that takes into account the other objects in the collection while decomposing each model.

The collection-aware segmentation (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. A conceptual overview of this methodology, depicted in Figure 2, 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

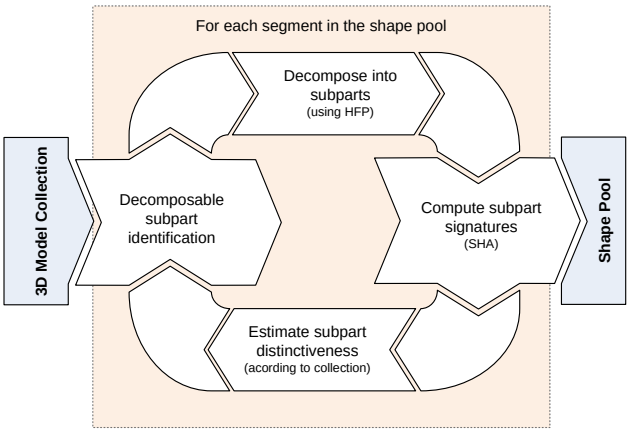

Fig. 2 Collection-Aware Segmentation pipeline

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.

We emphasise that the proposed segmentation algorithm does aim to produce decomposed versions of models in a collection. Instead its goal is to hierarchically extract segments from models in such a manner that uncommon segments are further segmented, while segments geometrically similar to many others are not.

This way we obtain a multilevel segmentation containing both distinctive and common subparts of models. Such result provide an adequate starting point for constructing the shape thesaurus.

From the description, above it should be clear that the proposed approach is supposed to work with generic collections of 3D models, and also supports different shape segmentation and description techniques. to validate our framework, we selected a well-defined setup and focused on a specific decomposition technique that facilitates the creation of a thesaurus for 3D shape retrieval. Therefore, three major constraints were defined:

- **Collection type:**
  CAD models;
- **Segmentation algorithm:**
  Hierarchical fitting primitives [34];
- **Shape description:**
  Rotation invariant spherical harmonics [35].

For our experiments we selected collections of CAD models instead of generic collections. For an easier comprehension of the algorithm we use, in this document as explanatory example, a very small set of models from the engineering shape benchmark collection (ESB) introduced in [36]. The examples are shown in Figure 3. When choosing which existing segmentation algorithm would have been the most appropriate to conduct our tests, we considered that (1) our experiments are focused on tessellated CAD models and (2) a multilevel/hierarchical segmentation is necessary in our framework. These two requisites led us to the choice of the hierarchical fitting primitives (HFP) segmentation algorithm, which indeed proved to behave particularly well on CAD models and is intrinsically hierarchical [37,38].

To create the feature vectors we use the spherical harmonics (SHA), a widely accepted rotation invariant shape descriptor published by the Princeton team [35]. Nevertheless, we plan to add more shape descriptors in future research, combining them in order to improve the accuracy of our similarity measurements.

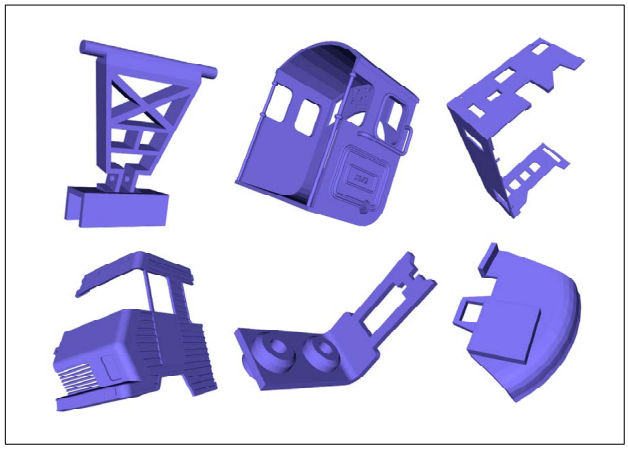

Fig. 3 Example collection with six models extracted from Purdue's Engineering Shape Benchmark [36].

4.1 Hierarchically Segmented Meshes

From the constraints referred above we developed the collection-aware algorithm based on 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 [34]. 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. This iteration repeats until there is only one cluster remaining, representing the whole model, and which will become the tree root. The resulting tree is called the hierarchically segmented mesh (HSM) and contains the whole multilevel decomposition of the segmented mesh.

In order to allow a fast iterative decomposition of models in the collection, the CAS/HFP algorithm takes advantage of the fact that HSM trees contain the whole multilevel segmentation. This way each model just need to be segmented once using HFP algorithm, and the resulting trees are then used to iteratively decompose all models in the collection. Indeed, as illustrated in Figure 4, the CAS/HFP algorithm is divided into two completely distinct stages (initialization and iteration) and the estimation of HSM trees, i.e. running the HFP algorithm, happens once for each model at the initialization stage. These trees are stored in the HSM set, constructed during the initialization.

4.2 Shape Pool

In the initialization stage, besides the HSM trees, also shape signatures are computed for all models in the collection, as the one depicted in Figure 5, along with the respective model. 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, the segments resulting from such decomposition and corresponding signatures are added to

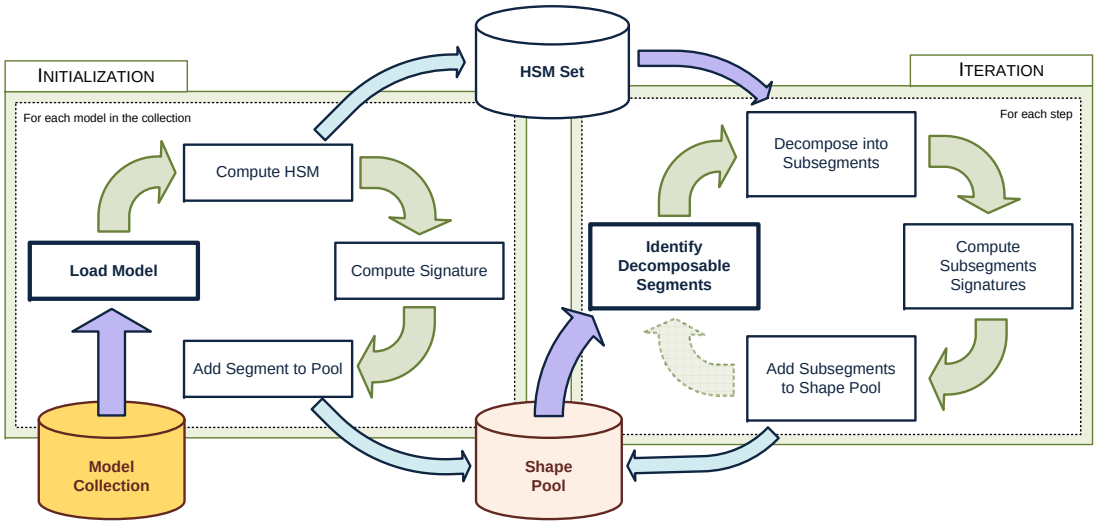

Fig. 4 CAS/HFP block decomposition

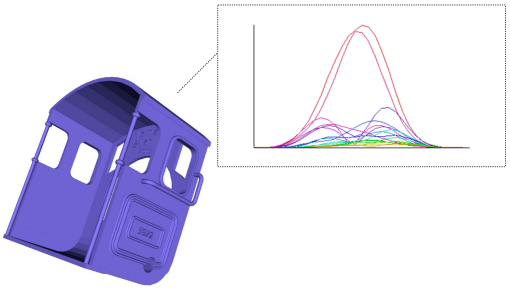

Fig. 5 3D model and corresponding signature.

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.

Indeed, the *Shape Pool* is a highly important part of the proposed 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. 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. 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.

4.3 Identification of Decomposable Segments

At the beginning of the iteration, the decomposable segments in the *Shape Pool*l must be identified in order to be further decomposed if necessary. This is indeed the major challenge of the proposed algorithm. It is not trivial to automatically identify which segments should be further decomposed. For this purpose, we suggest to use the shape signatures of each segment, to measure the Euclidean distance between them, and then count, for every segment, the number of segments within a given range, defined by the so-called *similarity threshold*. If this count is below a given value, the so-called *similar count threshold*, the segment should be further decomposed, since there are not enough similar shapes in the pool to flag it as a recurrent shape part in the given collection. In an extreme situation, all models share a common shape: the triangle. However, the idea behind this approach is to decompose models into a meaningful and not trivial set of shapes, since decomposing a model to the triangle level is quite useless for our approach.

Considering the *Shape Pool* as a multidimensional dataset, 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. Then, for each segment in the *Shape Pool*l we just need to apply the k-NN algorithm and check if all returned items are within the *similarity threshold*. An alternative way to achieve the same is using within-distance or α-cut search algorithm which will identify all segments within a given distance, which

should be set to the *similarity threshold* value and check if the number of returned segments is above the similar segments count.

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. However, 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 [39], which allows k-NN searches in sub-linear time of $O(\log N)$, where N is the cardinality of the *Shape Pool*.

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, which will increase the CAS/HFP algorithm memory requirements and at this point of our research we are mainly concerned in validate the proposed approach with collections containing hundreds models. In the future, to adapt the algorithm to larger collections, more efficient k-NN search methods can be used.

Nevertheless, to improve the behavior of our linear search and considering 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. Although theoretically it still runs in linear time, in practice it is much faster since it needs to measure much less distances. Moreover, we are aware that with some changes we can further improve the execution time without using any additional complex structures or, using such structures, achieve a sub-linear time complexity.

Summarizing, to determine if a segment is decomposable we apply a search algorithm in the signature space, using as parameters two values: the *similarity threshold*, which sets when two shapes are considered similar, and the *similar count threshold*, which defines how many segments should be similar in order to be considered not distinctive. Indeed, these are the two parameters used to tune our decomposition algorithm.

4.4 Sub-segments

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 stage and stored in the HSM set. Since these trees contain the whole multilevel segmentation of the model, it is fast and simple to determine 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 segment keeps a reference to the matching node, this task is accomplished in constant time.

For every newly created sub-segment a signature is computed. For that purpose the sub-segment is treated as an independent shape and the corresponding SHA descriptor is computed. The resulting signature is then attached to the sub-segment, as well as the reference to the corresponding node in the HSM tree. 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 (a parameter of the CAS/HFP algorithm) or if there are no more decomposable segments. At the end of the algorithm one has a *Shape Pool* containing all segments of every model in the collection, such as the pool depicted in Figure 6. This *Shape Pool* is then used to create the shape thesaurus.

5 Thesaurus Construction and Shape Retrieval

Although the core of our research has focused on the collection segmentation, the remaining of the classification process, namely the thesaurus construction and the retrieval process have an important role in our approach to shape retrieval with part-in-whole matching. Thus, in the next sections we will describe these methods, starting by the essential *Shape Pool* clustering.

5.1 Shape Pool Clustering

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 unlabelled data into groups in an unsupervised manner [1] . Depending on the approach, these groups, also called clusters,

[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 [40]. While in supervised classification data labels and corresponding mapping functions are

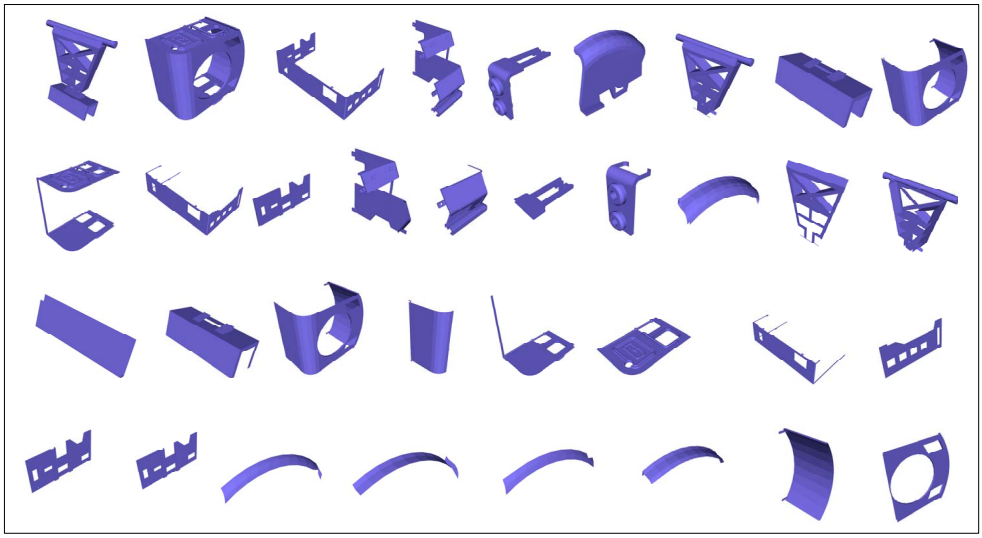

Fig. 6 *Shape Pool* resulting from the decomposition of example collection presented above.

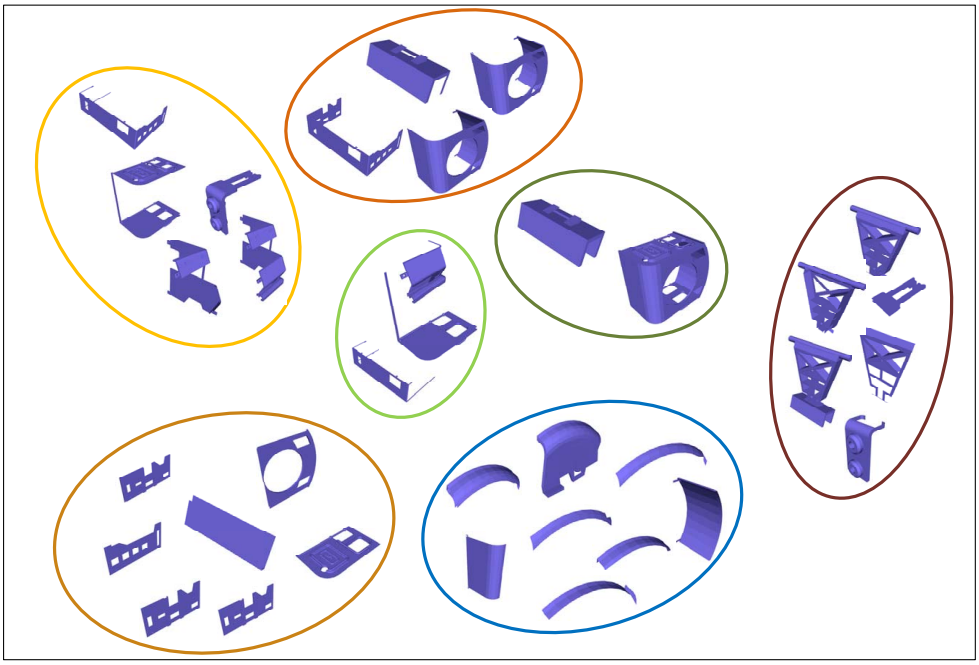

Fig. 7 Seven clusters partition produced from the *Shape Pool* depicted above.

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.

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 [41], with a vast family of algorithms.

In order to provide a tight control on the size of the lexicon 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 [42] proposed by Kanungo *et al.* [43]. As a result of applying this algorithm to the *Shape Pool* we ob-

used, in unsupervised classification - clustering - no labeled data are available.

tain a set of clusters grouping similar shape segments together, as depicted in Figure 7.

The number of clusters in the partition is directly related with the size of the thesaurus. Indeed, it corresponds to the number of terms in the thesaurus, as explained ahead. Thus, the definition of a suitable cluster count is determined by the desired size of the *Shape Thesaurus*. It should be large enough to allow a good representation of geometrical features and small enough to allow fast searches in the thesaurus terms.

5.2 Thesaurus Creation

After having the segments in the *Shape Pool* grouped according to its geometrical similarity we are able to create the shape thesaurus. Indeed, the shape thesaurus we devised is based on the well known thesaurus concept used in text information retrieval. According to Baeza-Yates and Ribeiro Neto [44] 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 pre-compiled 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 shapes that comprises it.

Starting from the partition computed based on the *Shape Pool*, we create the 3D shape thesaurus by considering each cluster as a term, T_i, constructing a list of segments that comprises each cluster and attributing a signature to each term. The term corresponds to the prototype for the grouped segments and the signature of a term is the centre of the corresponding cluster of signatures. An example of a shape thesaurus, constructed from the partition referred in previous section, is depicted in Figure 8.

In this example a thesaurus with seven terms was constructed from the partition with seven cluster depicted in Figure 7. To each term is assigned a list containing the segments in the corresponding cluster. Additionally, to each term is attributed a multidimensional signature based on the SHA signatures of the shapes in the cluster. As referred above, this signature corresponds to the centre of the cluster that originated the respective term.

After creating the shape thesaurus, and following the concepts used in text information retrieval, we produce the inverted file that will support the retrieval process. The inverted file is simply 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 in the thesaurus and, for each one, a list of models that contain segments in the corresponding segment list, *i.e.* in the corresponding *Shape Pool* partition cluster. In Figure 9 we illustrate an example of the inverted index relative to the example collection, constructed based on the thesaurus depicted previously.

In the depicted example the terms in the thesaurus are indexed according to the decomposition and clustering results. To that end, for each term in the thesaurus, we swept the corresponding segment list and identified the models to which every segment belongs and created with them a list of models assign to the respective term T_i in the inverted index.

5.3 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 10. The first step 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, the rotation invariant spherical harmonics (SHA).

As a result of feature extraction, we obtain the signature of the query shape. In order to retrieve models partially similar to the query, the nearest neighbors of this signature in the signature space of the shape thesaurus must be found. To that end a k-NN search is 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 can be used. Independent of the algorithm efficiency, any k-NN search will return the terms more similar to the query shape.

From the list of thesaurus 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 constant time if structures that map directly the terms in the thesaurus with the entries in inverted index were used. This option was not followed because our main concern was to guarantee as independence among all components in order to facilitate changes on the algorithms. Moreover, as part of

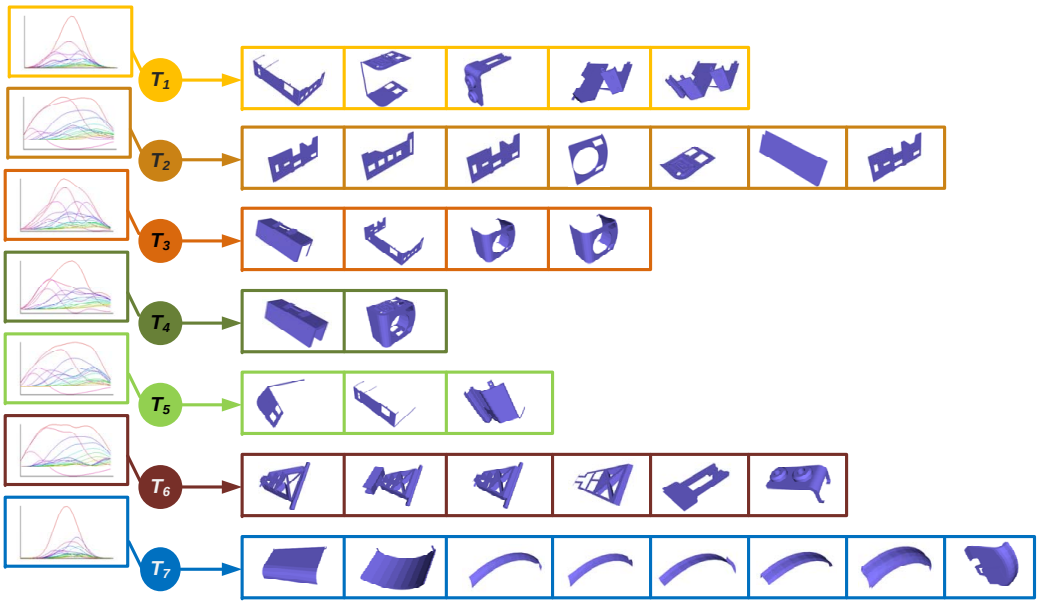

Fig. 8 Example of a shape thesaurus created from the previous partition.

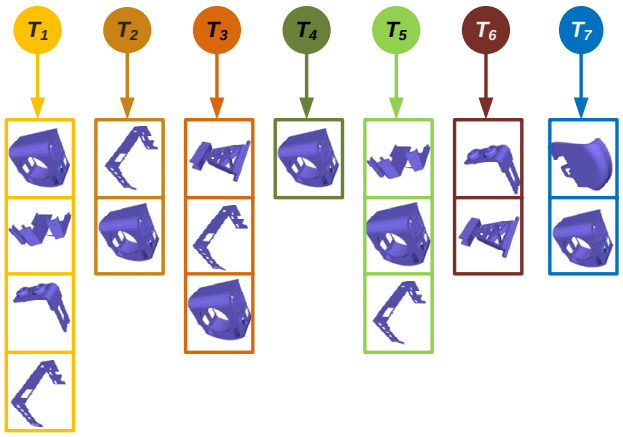

Fig. 9 Example of an inverted index created for the example collection based on the corresponding thesaurus.

future research, we plan to perform partial matching between the query and each model in the inverted list to improve the results.

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. Thus, the retrieval process execution time depends on query complexity and thesaurus size.

6 Experimental Results

As referred previously, the core of our approach is the identification of terms for the thesaurus. To identify these terms we must decompose all models in the collection, which is the most important part of the thesaurus construction process. We suggested using a collection-aware segmentation (CAS) method to compute such decomposition. More precisely, we developed a CAS algorithm based on the hierarchical fitting primitives (HPF) segmentation method.

Thus, one of our initial concerns, after implementing the proposed decomposition algorithm, was to compare the results produced by our method with the results produced by the HFP algorithm. Indeed, HFP is the basis of our algorithm, so it is important to check if

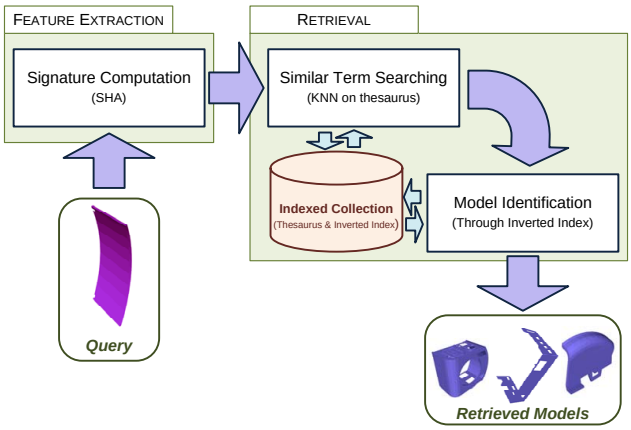

Fig. 10 Shape retrieval pipeline.

any improvement were achieved. To that end we used the small collection referred above to study the behavior of both algorithms. After asserting CAS/HFP effectiveness, we addressed the scalability problem. First, by simulating algorithm behavior with a worst case situation. Next, by evaluating the CAS algorithm with a 3D model benchmark collection. In the following sections we describe this work and discuss the obtained results.

6.1 Comparing CAS/HFP with HFP

We used our prototype to compute the segmentation with the collection-aware decomposition based on hierarchical fitting primitives. To compute the segmentation with the HFP algorithm we used the EfpiSoft tool [45], an application that implements the hierarchical fitting primitives segmentation algorithm and provides. To run the HFP algorithm within EfpiSoft a single parameter must be specified: the number of desired segments. However, in order to allow easier comparison of results we prefer to obtain a pre-determined tree depth. This issue was solved by defining the number of segments according to that tree level, which is simple since the hierarchical segmentation produces a binary tree.

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, while in the HFP tree some leaf segments are just planar patches and others are still complex parts. This leads to the unbalanced models decomposition that occurs when trying to stop HFP segmentation based on a predefined value instead of setting it manually during decomposition, as allowed by EfpiSoft. Instead, using the CAS/HFP algorithm the tree leafs have similar complexity, providing a balanced model decomposition, where all simpler segments are at the same complexity level.

Based on these results we concluded that, for our needs of automatic decomposition of models, our method produces better results than the original hierarchical fitting primitives algorithm. However it needs larger computation times per model. Still, we underline that our aim is constructing a thesaurus for shapes, a task that is supposed to run off-line, without user intervention. So, it is acceptable that the proposed CAS/HFP algorithm takes some time to process a collection.

6.2 Mesh complexity in ESB models

While studying the behavior of proposed CAS/HFP algorithm we needed to have some information about the mesh complexity of models in the engineering shape benchmark (ESB) collection [36]. Although the number of faces in a polygonal mesh might not be proportional to model complexity, we considered for our purposes that complex objects contain more polygons than simpler ones and face count is easy and fast to estimate, providing a good approximation to object complexity.

From this estimation, we concluded that a vast majority of models has more than one thousand faces and, within these objects, most of them have less than ten thousand faces, as illustrated in Figure 12. Additionally, the complexity histogram depicted in Figure 13 reinforces these conclusions.It is there clear that most models contain between one and six thousand faces, with a peak around the one thousand polygons count.

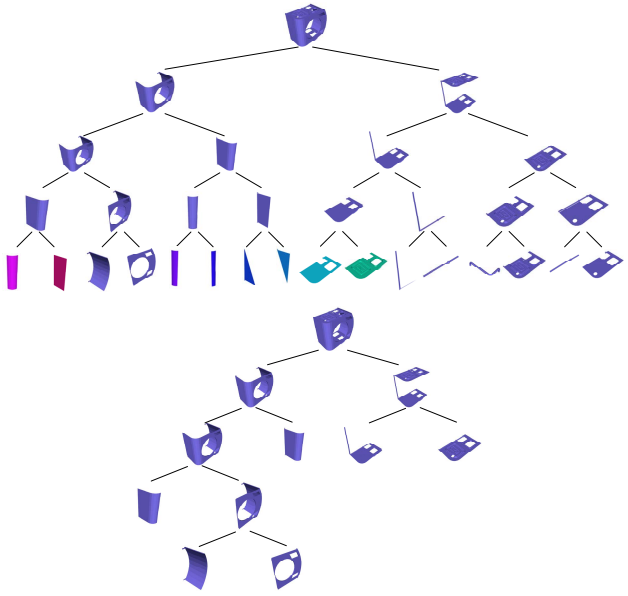

Fig. 11 Decomposition trees produced by HFP (top) and CAS/HFP (bottom)

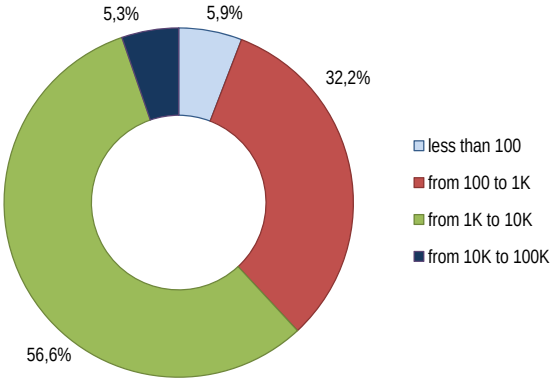

Fig. 12 Distribution of mesh complexity in ESB collection, according to polygon count per model.

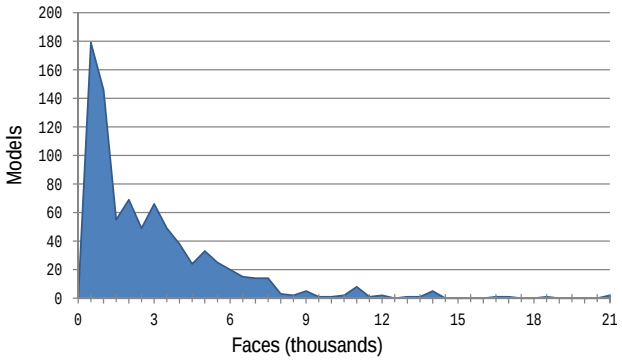

Fig. 13 Mesh complexity histogram for ESB collection.

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. Such conclusion is relevant because the execution time of HFP segmentation, performed at the initialization stage time of the CAS/HFP algorithm, depends on the number of polygons in the mesh as well as the size of the resulting HSM tree.

6.3 Worst case simulation

Due to the high-dimensionality of the SHA signature, the number of segments in the *Shape Pool* might turn into an issue. Therefore, we simulated the proposed decomposition algorithm behaviour in a worst case situation. We considered that all elements of the collection have much 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*, i.e. to be considered similar two segments should be basically equal, or an illogically high value for the *similar count threshold*, 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.

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 clusterof the ESB and other with the approximate size of the whole ESB collection. Therefore, we considered two hypothetical collections with one hundred and eight hundred 3D models respectively and simulate the algorithm behavior.

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 14. Naturally, the memory required to store segment information grows similarly. Computing the rotation invariant spherical harmonics 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.

Another problem identified during this simulation is the time required to determine if a segment should be further decomposed. Indeed, the decomposability identification is based on a neighbor search and sub-linear algorithms for this purpose are widely-known. However, as described in Section 4.3, in present implementation

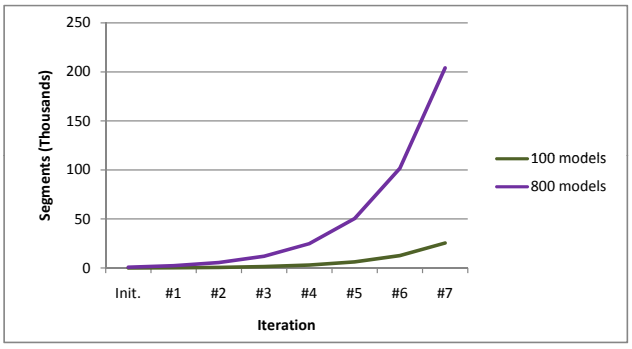

Fig. 14 *Shape Pool* growth in worst case simulations.

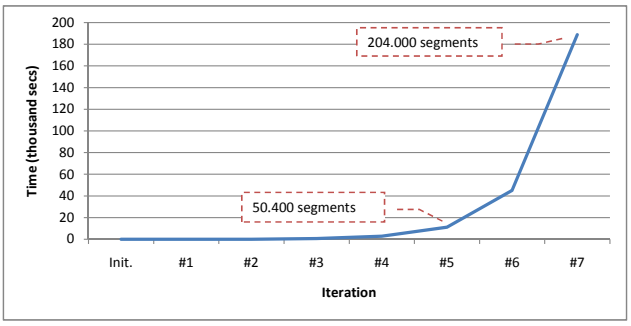

Fig. 15 Decomposability determination for a single segment in eight hundred models collection.

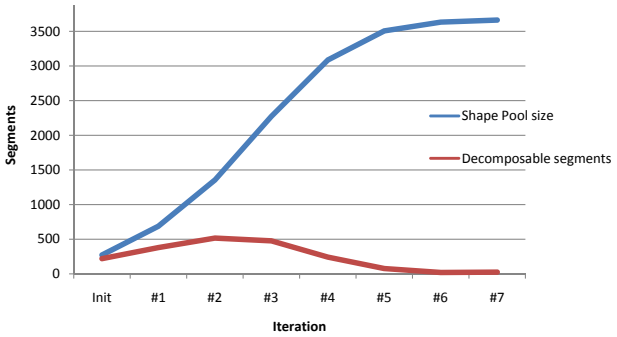

Fig. 16 *Shape Pool* growth for the RCP cluster of the ESB collection

of the CAS/HFP algorithm we used an adapted version of the k-NN linear search, called k-within range (k-WR). This version outperforms k-NN, allowing decomposability identification of a segment in reasonable time, but runs also in linear time *wrt Shape Pool* size. Thus, if the size of the *Shape Pool* grows exponentially we will face unacceptable computation times. For instance, in 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 15. 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.

6.4 Decomposing a benchmark collection

As expected, when applying the CAS/HFP algorithm with reasonable parameters to a benchmark collection, the results are quite different from the suggested by the worst case simulation. We used as a test collection, the clustered version of the engineering shape benchmark from SHREC 2008. This collection contains three clusters, described below:

- **Flat-Thin Components (FTC):**
  105 models whose envelope is largely a solid of revolution;
- **Rectangular Cubic Prism (RCP):**
  273 parts with a largely rectangular or cubic prism envelope;
- **Solid of Revolution (SoR):**
  475 objects with thin-walled sections and shell-like components;
- **Complete ESB collection:**
  853 models, i.e. all of the above.

We evaluated the algorithm with each one of the presented clusters separately. Its behavior was very encouraging, since in all tested cases the *Shape Pool* growth was far below the obtained in the worst case simulations. More important, the growth rate of the *Shape Pool*l starts decreasing around third iteration, which means that the *Shape Pool* does not increase to impractical size. For instance, when processing the RCP cluster with the proposed CAS/HFP we noticed that the number of segments in the *Shape Pool* stabilize below the four thousand elements, as depicted in Figure 16. Indeed, we observed similar behavior in all other tests performed until now, which is by itself a very positive result.

Considering this behavior and the results obtained in the worst case simulations, we concluded that the proposed algorithm will be able to produce valid results in suitable time and consuming a reasonable amount of memory. Analysing the simulations of decomposability determination process, as the one which is depicted in Figure 15, we see that for more than fifty thousand segments it took less than twenty seconds to determine if a segment is decomposable, while the *Shape Pool* size does not even reach one tenth of that. Moreover, from these simulations we saw that for a *Shape Pool* contain-

ing more than five thousand segments it will took less than fifteen seconds to determine the decomposability of all these segments. Even if we consider a larger *Shape Pool* with around fifty thousand segments, the time necessary to perform one iteration of the CAS/HFP algorithm in this circumstances is less than four hours. Since this algorithm is supposed to be batch processed, we think that such computation times are perfectly acceptable.

Furthermore, the memory requirements are also within reasonable range. Indeed, in the proposed approach, a single signature is assigned to each segment in the *Shape Pool* and 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*. As explained in Section 4, the CAS/HFP algorithm relies on the rotation invariant spherical harmonics introduced by Michael Kazhdan *et al.* [35] to numerically describe the three-dimensional shapes stored in the *Shape Pool*. The corresponding feature vector needs slightly more than 2KB of memory and is used as a signature for the shape. Since additional information is necessary for a segment, besides its signature, each item in the *Shape Pool* requires 4KB. This means that for a *Shape Pool* containing fifty thousand segments our technique will require around two hundred megabytes of memory to store it. Although this *Shape Pool* size is far above the expected when processing the benchmark collections, the amount of required memory is quite acceptable.

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 segmented, 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. The chart depicted in Figure 17 illustrates the measured execution times when processing the complete ESB collection. As expected, 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.

To understand which part 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 identified 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.

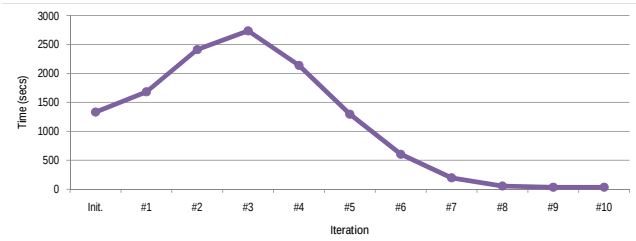

Fig. 17 Detailed execution time of CAS/HFP algorithm when processing the complete ESB collection.

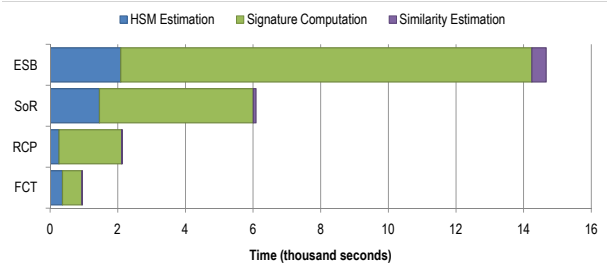

Fig. 18 Time spent by each stage of CAS/HFP algorithm.

From the measured times we observed that, as depicted in Figure 18, 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 expected 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.

6.5 Building a Shape Thesaurus

The ultimate goal of our research was to devise a methodology to index a collection of 3D models in such a way that is possible to retrieve objects using partial queries. To that end we proposed the use of a shape thesaurus with the corresponding inverted index. To study the construction of such structures we tested our approach

with the ESB collection. Using the methodology described in this document, we classified the set of 853 engineering models that compose the ESB [2]decomposing its models, computing the terms to represent the extracted segments and creating a shape thesaurus with this terms along with the corresponding inverted index.

Considering that the first step (decomposition of models in the collection) has finished, the creation of a *shape thesaurus* from the resulting *shape pool* requires two additional steps. The second step corresponds to the determination of a partition for the *shape pool*, while the third step 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*.

To cluster the segments in the *shape pool* we used 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, this dataset 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 datasets of different sizes, while modifying the number of clusters in the produced partition. The larger dataset tested contained less than ten thousand points and corresponds to the *shape pool* produced by the decomposition of models in the complete ESB collection, which occupies less than thirty megabytes of memory. Since the memory required to compute the partition 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 observed execution times confirmed the linear time-complexity of the clustering algorithm. As depicted in Figure 19, the execution time grows proportionally to shape pool size and cluster count. For example, considering the whole ESB collection, whose shape pool contains 9131 segments, determining a partition with one hundred clusters took around six hundred seconds, while computing a partition with eight hundred clusters took around six thousand seconds. This observation strengthens the importance of a reasonable size for

[2] Indeed, the ESB collection contains a total of 866 models, but due to technical problems reading a few files, we discarded thirteen models.

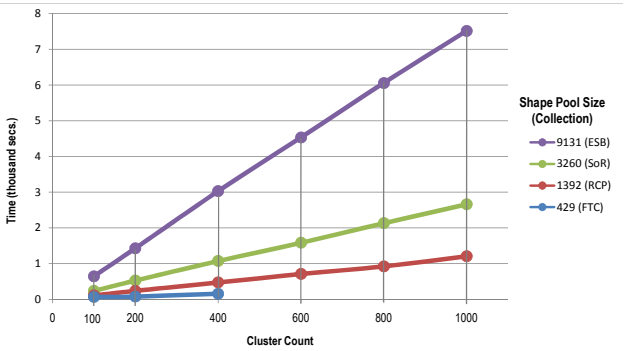

Fig. 19 Execution time of segment clustering for different collections, while varying the number of clusters to be created).

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 want to avoid. In the near future we intend to study this topic and present some suggestions regarding the determination of a "good" thesaurus size. For now, based on empirical tests, we will consider that the shape thesaurus to index the ESB collection will have eight hundred terms.

Finally, construction of the *shape thesaurus* and corresponding *inverted index* from the resulting partition is a straightforward task accomplished in insignificant time (less than ten seconds) when compared to the *Shape Pool* partition partition computation. On the other hand, the indexing structures are relatively small when compared with the real collection size. Indeed, the thesaurus containing the eight hundred signatures and corresponding identifiers requires 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.

Fig. 20 Query results.

6.6 Retrieval of 3D models

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.

To evaluate the retrieval efficiency and accuracy of the thesaurus-based 3D shape retrieval introduced on this research work, we submitted five distinct queries to the system and studied its behaviour. From these queries three (Q_3 to Q_5) were subparts of existing models in collection, while two (Q_1 and Q_2) were complete models from the ESB collection. These shapes are depicted together with the returned results in Figure 20 and were randomly selected from the shape pool and collection, respectively.

In our approach, when a query shape is submitted the first step consists on estimating its SHA signature. The next step of the retrieval process consists on finding similar terms in the thesaurus and retrieving the associated models. We measured the descriptor computation time separately from the searching time, as presented in Table 1. We emphasise that the first depends exclusively on the query shape and shape descriptor, while the second is affected by several factors, such as

	Queries				
	Q_1	Q_2	Q_3	Q_4	Q_5
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 1 Measured time (in seconds) while performing the queries using a given shape.

the number of terms in the thesaurus, signature dimensionality or shape pool size.

The observed results show that the computation of the query shape signature,due to its inherent complexity, 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.

When a complete object from the collection, such as query Q_1, is submitted to the system the first returned result, $R_{1,1}$, is exactly that model, as expected. 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 $R_{1,5}$ has visible similarities with the query shape, only a unnoticeable sub-part of $R_{1,4}$ has some similarity with the query. As often occurs, the matching sub-part is not easily recognisable in the returned model. To assist the reader in the recognition of the similar sub-part, we illustrate it in Figure 21. 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. During the retrieval process, after being identified as associated to a matching thesaurus term, the geometric similarity between the subpart and the query shape is measured using the SHA descriptor. The estimated distance $d_{SHA}(Q_1, R_{1,4})$ between the two signatures represents the partial similarity between the two models.

In another example, the shape Q_5 is itself a sub-part of a model. 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.

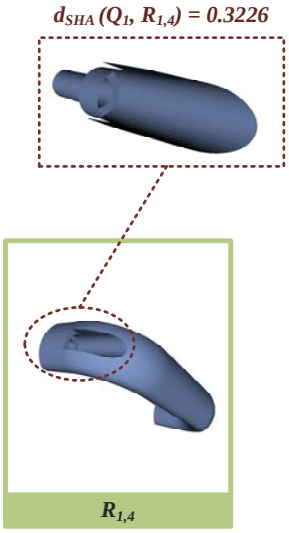

Fig. 21 Detail of retrieved model $R_{1,4}$.

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.

On the other hand, we need to measure the retrieval accuracy of our system, behind the examples presented above, for instance using precision-recall diagrams - a prevalent indicator to assess effectiveness of retrieval systems. The most common experimental methodology to evaluate the accuracy of a generic information retrieval system relies on a test collection, a set of queries, and a set of relevance judgements indicating which results are relevant for each query [46]. In the present context, the ESB is considered as a test collection, while the shapes Q_1 to Q_5 are seen as the query set. However, no relevance judgement exist for this or, to the extent of

our knowledge, for any other query set regarding partial matching. Indeed, we are planning to perform, in short term, a user study to establish well-grounded relevance judgements for a set of queries regarding part-in-whole matching in the ESB collection.

7 Conclusions and Future Work

As stated above, the tests we've made pointed out to reasonable *Shape Pool* sizes while processing the benchmark collections. Every time a segment is segmented the *Shape Pool* grows, thus the probability of finding more similar segments gets larger. This means that, after a few iterations of the algorithm, the number of segments identified as decomposable starts decreasing due to the growth of the *Shape Pool*. Such fact is important for our research because it ensures that the *Shape Pool* size does not increase behind a reasonable limit, as depicted in Figure 16. Based on this conclusion and now that we have a working prototype, we aim at experiencing the proposed approach with larger collections in a near future.

Roughly speaking, we can state that our approach to construct a shape thesaurus for 3D shape retrieval with part-in-whole matching is done. Indeed, further testing is needed and we will do it as part of future research, but as a proof of concept both our CAS/HFP algorithm and thesaurus based retrieval approach apparently work. Analysing the work we developed and what is missing, we believe that the initial goal was attained. Indeed, the developed prototype is capable of producing a shape thesaurus from a collection of 3D models and retrieve models with partial queries.

However, the described approach supports only whole-to-part searches, *i.e.* the query is matched as a whole against parts of models in the collection. Still, with some additional effort it will be easy to achieve full partial matching, where will be possible to find models with parts similar to parts of the query. To that end, a partial matching algorithm can be used instead of globally match the query with the terms in the thesaurus. Alternatively, the query itself can be decomposed by the CAS algorithm and the resulting subparts will be then matched against the terms in the thesaurus.

Additionally, several unsolved issues were identified and must be tackled in the future. Among others, we highlight three major subjects to look at. One is experiment the use of supervised classification algorithms to create the shape thesaurus instead of clustering. We believe that taking advantage of additional shape segment information might lead to better thesaurus, which will provide sounder retrieval results. Another is the use of more efficient k-NN search methods to improve retrieval speed. Indeed, we used a naive k-NN algorithm that runs in linear time with respect to thesaurus size, but several sub-linear approaches are available, such as the kd-tree [39] or the nb-tree [47]. The third is the computation of segment relevance in the original model, considering, for instance, the percentage of surface area that corresponds to the segment. Using such relevance information models could be ranked in the inverted lists, providing more accurate query results.

References

1. Johnson, A.E., Hebert, M.: Recognizing objects by matching oriented points. In: CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), p. 684. IEEE Computer Society, Washington, DC, USA (1997)
2. Gal, R., Cohen-Or, D.: Salient geometric features for partial shape matching and similarity. ACM Transactions on Graphics **25**(1), 130–150 (2006)
3. Shilane, P., Funkhouser, T.: Distinctive regions of 3d surfaces. ACM Transactions on Graphics **26**(2), 7 (2007)
4. Suzuki, M.T., Yaginuma, Y., Yamada, T., Shimizu, Y.: A partial shape matching method for 3d model databases. In: Proceedings of the Ninth IASTED International Conference on Software Engineering and Applications (SEA2005), pp. 389–394. ACTA Press, Phoenix, USA (2005)
5. Suzuki, M.T., Yaginuma, Y., Shimizu, Y.: A partial shape matching technique for 3d model retrieval systems. In: ACM SIGGRAPH 2005 Posters, p. 128. ACM Press, New York, NY, USA (2005)
6. Silverstein, C., Marais, H., Henzinger, M., Moricz, M.: Analysis of a very large web search engine query log. SIGIR Forum **33**(1), 6–12 (1999)
7. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Transactions on Graphics **23**(3), 652–663 (2004). DOI http://doi.acm.org/10.1145/1015706.1015775
8. Ruge, G.: Automatic detection of thesaurus relations for information retrieval applications. In: Foundations of Computer Science: Potential - Theory - Cognition, to Wilfried Brauer on the occasion of his sixtieth birthday, pp. 499–506. Springer-Verlag, London, UK (1997)
9. Joyce, T., Needham, R.M.: The thesaurus approach to information retrieval. Readings in information retrieval pp. 15–20 (1997)
10. Curran, J.R., Moens, M.: Improvements in automatic thesaurus extraction. In: Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition, pp. 59–66. Association for Computational Linguistics, Morristown, NJ, USA (2002). DOI http://dx.doi.org/10.3115/1118627.1118635
11. Milne, D.N., Witten, I.H., Nichols, D.M.: A knowledge-based search engine powered by wikipedia. In: CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 445–454. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1321440.1321504
12. Crouch, C.J.: A cluster-based approach to thesaurus construction. In: SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 309–320. ACM, New York, NY, USA (1988). DOI http://doi.acm.org/10.1145/62437.62467

13. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1-7), 107–117 (1998)
14. Paquet, E., Rioux, M.: Nefertiti: A query by content software for three-dimensional models databases management. 3dim **00**, 345 (1997)
15. Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., Jacobs, D.: A search engine for 3d models. ACM Trans. Graph. **22**(1), 83–105 (2003)
16. Funkhouser, T., Kazhdan, M., Min, P., Shilane, P.: Shape-based retrieval and analysis of 3d models. Communications of the ACM **48**(6), 58–64 (2005)
17. Lou, K., Prabhakar, S., Ramani, K.: Content-based three-dimensional engineering shape search. International Conference on Data Engineering p. 754 (2004)
18. Chen, D.Y., Tian, X.P., Shen, Y.T., Ouhyoung, M.: On visual similarity based 3d model retrieval. Computer Graphics Forum **22**(3), 223–232 (2003)
19. Vranić, D.V.: 3d model retrieval. Ph.D. thesis, University of Leipzig, Germany (2004)
20. Assfalg, J., Bimbo, A.D., Pala, P.: Content-based retrieval of 3d models through curvature maps: a cbr approach exploiting media conversion. Multimedia Tools and Applications **31**(1), 29–50 (2006)
21. Ansary, T.F., Daoudi, M., Vandeborre, J.P.: A bayesian 3D search engine using adaptive views clustering. IEEE Transactions on Multimedia **9**(1), 78–88 (2007)
22. Ansary, T.F., Daoudi, M., Vandeborre, J.P.: 3d-models search engine from photos. In: Proceedings of ACM International Conference on Image and Video Retrieval (CIVR 2007). Amsterdam, The Netherlands (2007)
23. Suzuki, M.T., Yaginuma, Y., Sugimoto, Y.Y.: A 3d model retrieval system for cellular phones. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 3846–3851. IEEE Computer Society (2003)
24. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Ferret: a toolkit for content-based similarity search of feature-rich data. SIGOPS Oper. Syst. Rev. **40**(4), 317–330 (2006)
25. Bustos, B., Keim, D.A., Saupe, D., Schreck, T., Vranić, D.V.: Feature-based similarity search in 3d object databases. ACM Computing Surveys **37**(4), 345–387 (2005)
26. Ruiz-Correa, S., Shapiro, L.G., Meila, M.: 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, p. 1126. IEEE Computer Society, Washington, DC, USA (2003)
27. Kazhdan, M.: Shape representation and algorithms for 3d model retrieval. Ph.D. thesis, Princeton University (2004)
28. Bespalov, D., Shokoufandeh, A., Regli, W.C., Sun, W.: Scale-space representation of 3d models and topological matching. In: Proceedings of the 8^{th} ACM symposium on Solid Modeling and Applications, pp. 208–215. ACM Press, New York, NY, USA (2003)
29. Cornea, N.D., Demirci, M.F., Silver, D., Shokoufandeh, A., Dickinson, S.J., Kantor, P.B.: 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, pp. 368–373. IEEE Computer Society, Washington, DC, USA (2005)
30. Biasotti, S., Marini, S., Spagnuolo, M., Falcidieno, B.: Sub-part correspondence by structural descriptors of 3d shapes. Computer-Aided Design **38**(9), 1002–1019 (2006)
31. Tierny, J., Vandeborre, J.P., Daoudi, M.: Partial 3d shape retrieval by reeb pattern unfolding. Computer Graphics Forum **28**, 41–55 (2009)
32. Suzuki, M.T., Kato, T., Otsu, N.: A similarity retrieval of 3d polygonal models using rotation invariant shape descriptors. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 2946–2952. IEEE Computer Society, Nashville, TN, USA (2000)
33. Campbell, R.J., Flynn, P.J.: A survey of free-form object representation and recognition techniques. Comput. Vis. Image Underst. **81**(2), 166–210 (2001)
34. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. **22**(3), 181–193 (2006)
35. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: L. Kobbelt, P. Schroder, H. Hoppe (eds.) Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 156–164. Eurographics Association, Aire-la-Ville, Switzerland (2003)
36. Jayanti, S., Kalyanaraman, Y., Iyer, N., Ramani, K.: Developing an engineering shape benchmark for cad models. Computer-Aided Design **39**(9), 939–953 (2006)
37. Attene, M., Biasotti, S., Mortara, M., Patan, G., Spagnuolo, M., Falcidieno, B.: Computational methods for understanding 3d shapes. Computers & Graphics **30**(3), 323 – 333 (2006). DOI DOI: 10.1016/j.cag.2006.02.007
38. Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N., Azariadis, P.: 3D mesh segmentation methodologies for cad applications. Computer-Aided Design and Applications **4**(6), 827–841 (2007)
39. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
40. Anderberg, M.R.: Cluster analysis for applications. Academic Press, New York (1973)
41. Kogan, J.: Introduction to Clustering Large and High-Dimensional Data. Cambridge University Press, New York, NY, USA (2007)
42. Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982)
43. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(7), 881–892 (2002)
44. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)
45. Attene, M.: 'efpisoft'. http://efpisoft.sourceforge.net/ (2006)
46. Cormack, G.V., Palmer, C.R., Clarke, C.L.A.: Efficient construction of large test collections. In: SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 282–289. ACM, New York, NY, USA (1998). DOI http://doi.acm.org/10.1145/290941.291009
47. Fonseca, M.J., Jorge, J.A.: Indexing High-Dimensional Data for Content-Based Retrieval in Large Databases. In: Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA'03), pp. 267–274. IEEE Computer Society Press, Kyoto, Japan (2003)