INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

CaS: Collection-aware Segmentation

Raquel Garcez Coutinho Costa

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Júri

Presidente:	Professor Joção António Madeiras Pereira
Orientador:	Professor Alfredo Manuel dos Santos Ferreira Júnior
Co-Orientador:	Professor Manuel Joção Caneira Monteiro da Fonseca
Vogal:	Professor Carlos Antonio Roque Martinho

Novembro 2010

Acknowledgements

To my advisor, Professor Alfredo Ferreira, my sincere thanks for all the availability, support and guidance given during the evolution of this thesis and for introducing me on this area of the computer graphics .

Also a thank co-adviser Professor Manuel J. Fonseca for the support during the course of this thesis.

I am also grateful to the Portuguese Foundation for Science and Technology (FCT) who partially supported this work through 3DORuS project, with reference PTDC/EIA-EIA/102930/2008.

To all the users that accepted to participate in the tests and helped improving my solution a special thanks.

I would like to thank my family who supported me during these years. Without their support I would not have reached this moment.

I could not finishes, without thanking to all of my colleagues and friends at laboratory eleven, where most of this work was developed, that were also elaborating theirs master thesis and helped when I needed the most.

Lisbon, November 26, 2010
Raquel Costa

Resumo

Ao longo dos tempos, a segmentação de objectos 3D tem provado ser um desafio, dado que pode ser utilizada em diferentes áreas e a maioria dos resultados da segmentação depende da interpretação humana do objecto. Para cada contexto, diversas soluções foram propostas com diferentes objectivos, limitações e vantagens.

Com este trabalho propomo-nos a superar algumas dessas limitações usando o algoritmo de segmentação *Collection-aware Segmentation* (*CaS*). Este algoritmo identifica os segmentos dos objectos que se encontram numa coleções, baseando-se na sua individualidade e encontrava-se inicialmente integrado numa framework de recuperação de objectos 3D. O principal objectivo da solução descrita neste documento foi isolar e melhorar esse algoritmo do resto da aplicação, tornando-o capaz de decompor objectos 3D que se encontram embutidos numa coleção de objectos.

Para atingir esse objectivo, a solução foi melhorada, alterando a arquitectura e algumas características. Com essas alterações surge uma abordagem diferente, o *Adapted CaS* que usa a individualidade dos segmentos a fim de segmentar automaticamente uma colecção de objectos. Para isso, utiliza a abordagem *Hierarchical Fittinf Primitives* (*HFP*), que gera uma árvore binária. A decomposição da colecção de objectos segmentados é gerada ao descer nessa árvore, seleccionando os segmentos que são importantes para a decomposição, os quais são descobertos com base na sua singularidade relativamente á colecção a que pertence.

Após a execução do teste que foi realizado a fim de se compreender como os seres humanos segmentam uma coleção de objectos, foi possível perceber que alguns dos resultados produzidos por *Adaped CaS* estão sobre-segmentados. Assim, surge uma abordagem diferente, a *Geons-augmented CaS*, onde a principal diferença é a introdução de Geons para decompor um objecto.

A avaliação experimental revela que a nossa abordagem produz uma segmentação com resultados significativos para o ser humano e que melhorou relativamente ao *Original CAS*. Igualmente mostrou que, comparativamente com a abordagem original, gastou menos tempo de execução, mas em contra partida necessitou de mais memória. Ainda assim, é um bom resultado, dado que a memória é hoje em dia um recurso barato.

Abstract

Along the times, segmentation of 3D objects has proved to be a challenge because it has several applications in different areas and also because most of them depend on the human interpretation of the object. For each context, several approaches were proposed with different goals, limitations and advantages.

With this work we propose to overcome some of those limitations using the *Collection-aware segmentation* algorithm (CaS). This algorithm was initially integrated into a 3D object retrieval framework. It identifies segments of objects in collections based on their individuality. The main goal of the approach described in this document was to isolate and improve this algorithm from the rest of the application, making it capable of decomposing objects in 3D collections.

To accomplish this goal, the approach was improved by changing the architecture and some characteristics. Therefor, we developed a different approach, the *Adapted CaS* which uses the individuality of the segments to segment automatically a collection of objects. For that, it uses the *Hierarchical Fitting Primitives* (*HFP*) that generates a binary tree. The decomposition of the segmented objects collection is generated by traversing this tree and selecting the important segments for the decomposition, these are found based on theirs singularity relatively to the collection where it is set in.

After the execution of the test that was performed to understand how human beings segments a collection of objects it was possible to notice some over-segmentation on the results produced by *Adaped CaS* when decomposing the collection. So, a different approach arises, the *Geons-augmented CaS* where the main difference is the introduction of Geons to decompose an object.

Experimental evaluation reveals that our approach produces meaningful segmentation for the human being and has improved comparatively to the *Original CaS*. It has also showed that, comparatively to the *Original CaS*, less execution time was spent but more memory was used. This is still a good trade-off, since memory is nowadays a cheap resource.

Palavras Chave
Keywords

Palavras Chave

Segmentação de Objectos 3D
Colecções
Automática
Semelhança
Signaficativo para o Ser Humano

Keywords

3D Object Segmentation
Collections
Automatic
Similarity
Meaningful to the Human Being

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mesh segmentation is used as a stage in many computer graphics applications such as, collision detection, animation or even indexing and retrieval of 3D objects, making these approaches application depended. As computer graphics is a growing area, several studies in this subject have been proposed with different algorithms. On this document we present limitations of the currents approaches, and still a different new approach. This is based on an algorithm previously presented by Alfredo Ferreira et al. in [10] which is embedded on an application of indexing and retrieval. The major goal of the work described in this dissertation was to isolate this algorithm from the rest of the framework and adapt it in order to create an independent segmentation approach. In this section we give a brief description of the motivation behind this work and also a description of the existing limitations related to the current approaches. We then briefly explain the proposed approach along with our main contributions, and finally an overview of the dissertation.

1.1 Motivation

Each human being is a unique individual, and thus interprets his environment in different ways to each other. This generates a huge range of possible interpretations of the world, and its components. Studies have been made in order to find a pattern and a measure that can embody the majority of the human being interpretations of the world. While interpreting the objects around him, it is common for a human to segment them into meaningful parts. Indeed, the object segmentation has been subject of study in different areas, from mathematics to philosophy, among others. In particular in computer graphics it is studied and used in different applications, from object indexing and retrieval to 3D animation or collision detection, not to mention CAD re-engineering.

1.2 Problem Description

Most of the proposed approaches were created in order to support different applications in different domains. Usually the segmentation algorithms are a stage of a bigger solution so they produce the best results that each application really needs, making the segmentation application and domain dependent. These and other limitations were highlighted during a study about the existent segmentation techniques and are described in chapter 2 where they were also evaluated with their advantages.

A major limitation of most approaches to 3D object segmentation is that they require interaction with users. They need inputs provided by the users to properly segment an object and this hinders an automatic segmentation. Another common problem with several approaches is the over segmentation.

After segmenting, the presented results have more segments than a human would consider, producing a meaningless result to the user.

Another common limitation occurs when we consider segmenting a collection of objects. These approaches only segment object by object instead of taking into account a collection of objects where they are included, and also, do not segment all objects simultaneously, so segmenting object by object can take more time than if it were done simultaneously.

1.3 Proposed Approach

To overcome the limitations we propose a different approach that extends the *Collection-aware Segmentation* (*CaS*) approach. This algorithm was originally proposed by Alfredo Ferreira et al. in [10] and was embedded in a framework for retrieval of 3D objects. On this thesis we intended to prove that this approach can be adapted into an independent segmentation algorithm that produces meaningful results to the humans. To that end, we will separate it from the framework, while improving and refining the algorithm. This approach uses the *Hierarchical Fitting Primitives* (*HFP*) [4] approach to hierarchically segment the collection of objects, and Spherical Harmonics (SHA) to compute the descriptors and calculate the similarity between segments. Because it is based on the *HFP*, the produced results are part based, that is, meaningful to the user.

Since the *Adapted CaS* approach is no longer application dependent, its architecture has changed in order to improve the approach, namely the execution time. As the *Original CaS*, the *Adapted CaS* has two main stages, the initialization, where all foundations for the segmentation are computed, and the iteration, where the segmentation is actually executed using the data structures built on the previous stage. The main difference between the two approaches resides on the data structures. The *Adapted CaS* no longer uses two binary trees and a shape pool. Instead it uses only one tree, two lists and a signature pool.

The initialization stage where a Hierarchical Segmented Mesh (HSM) is computed for each object of the collection using the Hierarchical Fitting Primitives. The HSM is a hierarchical binary tree that is built bottom to top, where the leafs are the triangles of the mesh which are considered the initial clusters and the root is the entire object. This is built by merging pairs of clusters with the minimum clustering cost. In this case, this cost is calculated by fitting primitives (cylinders, cones, spheres and planes) to these clusters. This structure helps guide through the segmentation because it indicates which are going to be the next segments to be processed.

After being computed, the HSM is saved on the HSM Set, a data structure inherited from *Original CaS*. Then the SHA signature for the root node of each HSM is computed . The resulting feature vectors are saved on the signature pool. These are later used to label a segment as decomposable or not.

To support the segmentation, we added two lists, The Non Decomposable List (NDL), which saves both all the segments that are going to be decomposed and all the segments that are labeled as not decomposable, and the Non Segmented List (NSL), where we save all the segments that are going to be processed on the next iteration. Also on the initialization stage the root nodes of the HSMs are saved on the NDL.

The iteration stage performs the segmentation of objects using the foundations built on the previous stage. Its execution is done by adding and removing segments from the NDL and NSL depending if the segments are decomposable or not. The iteration and the algorithm itself finishes when it reaches the last element of the NDL and the NSL is empty, returning the NDL with all the objects segmented.

After executing a test to evaluate how the human being segments a collection of objects and comparing these results with the ones produced by the *Adapted CaS* approach, we have concluded that most of the times, the approach produced over-segmented results, therefore a new feature was added to the algorithm.

Original CaS
Contribution
Adapted CaS
Contribution
Geons Augmented CaS

Figure 1.1: Evolution of the *CaS* Approach

This new feature uses a collection of Geons to label an object as decomposable or not. These are simple objects that have particular characteristics [8] and some are the same primitives used on the Hierarchical Fitting Primitives, such as the cylinder, the sphere and the cone. These were used by comparing each object to all Geons in the collection, so that if the segment is similar to any of the Geons, than it is labeled as not decomposable and so, the decomposition of that segment stops.

1.4 Contributions

On this work, one of the main contribution was to isolate the CaS approach of the rest of the indexing and retrieval application, thus creating an independent segmentation algorithm. By isolating the segmentation stage from the rest of the solution, the *Adapted CaS* (figure 1.1) returns the entire collection of segmented objects instead of returning just a shape pool with the segments found during the decomposition, as the *Original CaS* did.

Additionally to the isolated segmentation algorithm, Geons were introduced as a stop condition for decomposing the collection of objects (figure 1.1), solving the over segmentation problem that is present in various other segmentation approaches.

The overall contribution of this work was an algorithm for automatic segmentation that takes into consideration the geometric similarities of elements in the collection and that produces a meaningful segmentation to the user. Therefor, this approaches segments a collection of objects automatically without need for user input.

1.5 Dissertation Overview

This dissertation describes and validates the proposed approach. The document here presented starts by analyzing and studding the related work on the state of the art of three-dimensional object segmentation. This study, presented in chapter 2, helped to understand the existing problems.

To overcome the identified limitations in the existing segmentation algorithms, a new technique is proposed. This technique is based on the *CaS* algorithm originally proposed by Alfredo Ferreira et al. on [10]. Chapter 3 describes in detail the *Adapted CaS* algorithm, as well as the *Geon-augmented CaS*. The initialization and the iteration stages that compose the approach and how they work are also described in that chapter, as well as the main data structures used in the algorithm and the relations among them.

To evaluate our approach four different tests were conducted. Two tests with users in order to discover a meaningful segmentation and to evaluate the algorithm and two other tests to measure the performance of the algorithm. On section 4 these tests are described and the experimental evaluation is present, along with an analysis and reflexion of the results of the different tests that were performed.

In the last section of this dissertation we present our conclusions and a reflection on the future work to develop.

Chapter 2

Related Work

With the evolution of computer graphics applications and fact that mesh segmentation was application dependent, several studies on this subject were proposed. Each application also requires different segmentation results depending on their purposes, so the segmentation techniques are classified in two different categories, the Part based and the Surface based described on the next section. Next we describe these different segmentation techniques and finally we make a study that compares these techniques by highlighting limitations and advantages.

2.1 Categorization of segmentation approaches

As most of the proposed approaches to 3D object segmentation are domain specific, the results of each segmentation techniques can differ. Nevertheless, some authors [1], [26] classify the segmentation techniques in two categories depending on the kind of segmentation accomplished based on a different point of view on the object being partitioned, illustrated in Figure 2.1. The part based segmentation algorithms take into account user's interpretation and produces a 3D volumetric view while the surface based segmentation rely on geometric features of the object surface and produces 2D surface view.

Part Based This type of segmentation decomposes the object into meaningful or semantic components, based on human perception, and so, it is necessary to understand the human perception of the objects. Many studies concluded that a human decomposes the object into smaller parts. Applying this concept on the segmentation, the part based algorithms decompose a 3D object into sub-meshes meaningful to the human that can be useful in shape matching, retrieval, and shape reconstruction applications [25].

Surface Based The surface based segmentation is sensitive to the shape surface, to the geometric properties of the shape boundary, like the planarity or the convexity. Hence, Surface-based segmentation algorithms use surface criteria to segment an object into fragments. This type of algorithms is usually used in texture mapping.

There are also cases in which both types can be combined. For instance, meaningful components are used by surface-type segmentation and also the opposite, in which meaningful components are found using surface-based attributes.

This categorization is used to differentiate the results of the approaches for a better evaluation and comparison. The next section describes these approaches and classifies them.

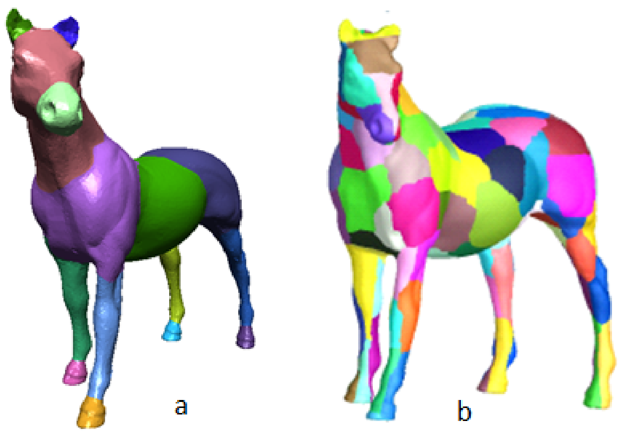


Figure 2.1: Representation of the result of the two different types of segmentation. a) Part Based Segmentation; b) Surface Based Segmentation. (Image taken from [26])

2.2 Current Approaches to 3D Object Segmentation

The 3D object segmentation is a problem that has been widely studied and different solutions that use distinct approaches have been proposed. Some of the existent and more relevant are presented on this section.

2.2.1 *Region Growing*

The process of *Region Growing* starts on a seed[1] and then traverses the mesh agglomerating the adjacent faces. There are many criteria to choose which faces should be agglomerated and there are also many stop or continue criteria, which are called termination or growing criteria, depending on the functionality. Every time the cycle stops a sub-mesh is found. Then, it restarts the process on another seed.

The solution proposed by Zuckerberger et al. on [32], uses a depth first or breadth first search to transverse the graph that represents the mesh. It starts in the node of this graph that represents the seed which is a face on the mesh and is chosen by the user.

When it is traversing the graph it collects faces while they are forming a convex area, meaning that this is the growing criteria, and the termination criteria is when the convexity property is violated. From this traversal a segment is created and then it restarts traversing on an unvisited node of the graph forming a new segment.

On [31] proposed by Y. Zhang et al. the *Region Growing* is a stage of the solution. As shown in Figure 2.2 the solution is constituted by four stages. In the first stage the Gaussian curvature is computed for each vertice on the surface. Then, on the boundary detection stage, each vertice with a highly negative curvature is labeled as boundary, using a predefined threshold, and the remaining vertices are labeled as seeds. The next step on boundary detection eliminates the isolated vertices. Then it passes to the next stage, the *Region Growing*, where it uses the seeds that were found previously to create the patches, and uses the Gaussian curvature and the boundary vertices as termination criteria. Finally, a last stage of post-processing is executed. In this stage, the vertices that were not labeled are assigned to a patch and patches that have fewer vertices than a predefined number of vertices are eliminated.

Both proposed solutions use the region growing to produce meaningful results in order to be used in retrieval, metamorphosis and simplification applications. So, this approach is classified as part based segmentation.

[1]A seed can be a point, a face or region of the mesh and can be chosen by the user or automatically

2.2.2 *Clustering*

The main idea of this approach is to cluster the mesh faces based on some rules. The main difference between *Clustering* and *Region Growing* is that *Region Growing* starts only in one point, agglomerating the adjacent faces, while *Clustering* approach starts on several points and agglomerates the adjacent faces to each point generating several segments at the same time. There are two main Clustering types:

Hierarchical Clustering

In *Hierarchical Clustering* each face of the mesh initially represents a cluster. Then it starts a cycle where first, the costs of grouping the faces are computed, and with these costs it chooses the pair of faces that cost less to agglomerate. The cycle proceeds until we have as final result, the initial object. This method generates a hierarchical tree where the root is the entire object, the leafs are the triangles of the mesh and the intermediates nodes are the several segments which resulted of each iteration of the cycle. There are many ways to calculate these costs. One of them is by approximating a primitive to the new cluster generated by grouping the clusters. Other is calculating the convexity of the cluster.

The final result of this initial technique is a binary tree whose leaf nodes are the original model vertices and the rest represents two child nodes clustered, that is, whenever two vertices are merged, a new node is created on the mesh that is the parent of the two merged nodes that represent the vertices of the mesh. In the last solutions presented that use the face clustering, a dual graph whose propose is to support the segmentation is usually used. Along with the graph a tree who's leafs are the clusters that represent the faces and this tree is built along the algorithm. Whenever two nodes of the dual graph are fused, a parent node is created on the tree and the algorithm proceeds as was previously explained. For each pair of nodes, graph edge, a cut value is assigned, and the edge with lower value that is going to be contracted is chosen. The cut value is calculated based on a primitive, that is, given a cluster, it is approximated, in this case, to a plane. In [4], Attene et al., also uses other kinds of primitives, like spheres and cylinders to calculate the clustering cost.

In his latest works, Attene et al. in [6] gives emphasis to the convexity of the object. Therefore, a hierarchy similar to the previously described is created, where the cost of clustering is calculated based on the cluster convexity.

This approach is surface based because it is based in using the criteria of the surface as the convexity, or using the primitives to compute the clustering cost. These proposed solutions are used in reverse engineering, mesh denoising and deforming shape features applications.

Iterative Clustering / K-means

The main idea is to choose the best segmentation of k clusters. Hence, the total number of k clusters that are going to be generated by the algorithm is provided by the user. Initially a representative group of k clusters is created, then each face of the mesh is associated to a cluster until it reaches r radius. This radius is the distance between the seed and the boundary of the patch.

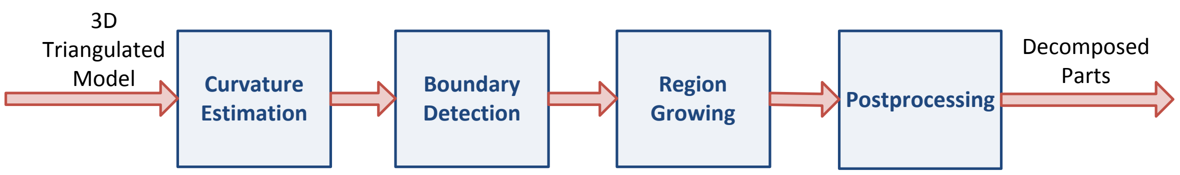


Figure 2.2: Representation of the four steps of the solution proposed by Y. Zhang on [31].

After having the first segments, the algorithm restarts the execution changing the representative group of clusters until there are no more changes on the segments found. The algorithm goes iteratively detecting what the best segmentation is given a number of clusters. In [2] the *Iterative clustering* is a stage, and this proposed solution contains four stages. The first calculates the feature value to each point of the mesh, in this case, a value that characterizes the surface based on the curvature, which is called shape index. This value states if the point belongs or not to a concave, hyperboloid or convex shape. In the second stage, the algorithm builds, for each node, a histogram that has the values of the neighbors inside a radius r, which is the distance between the seed and the boundary of the patch. These values are then saved on a feature vector. The third stage just reduces the size of the feature vectors and transforms the feature spaces in coefficient values. The last stage applies the k-means algorithm to the coefficient vectors. This approach is surface base because it uses the shape of the surface to segment, in this case, the concavity, the hyperboloid and convexity to calculate the feature value of each point and their goal was to create a segmentation that is useful for object recognition tasks.

In [16] the *Iterative Clustering* is used on the last stage of the algorithm. Initially a decomposition using the Iterative Clustering is done. To each pair of faces a distance is attributed, and this value is used to calculate the probability of a given face belonging to a segment of the mesh. Next, the *Iterative Clustering k-means* agglomerates the faces that have more probability of belonging to a given segment, which results in having a zone where the probability of a face belonging to a given segment is very near the probability of the same belonging to another segment, getting a fuzzy area, as depicted in Figure 2.3. Lastly the fuzzy decomposition is transformed into a final decomposition by refining the limits using an appropriate algorithm. For this proposed solution it was important to produce a meaningful segmentation that could be used in matching, retrieval, metamorphosis and computer animation applications. So, this is classified as part based.

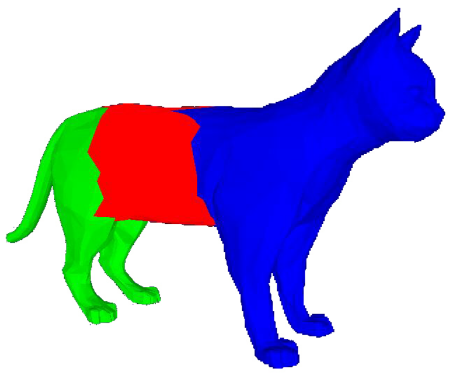


Figure 2.3: Fuzzy zone represented in red. (Image taken from [16] ©ACM 2003)

2.2.3 *Skeleton Based* segmentation

This approach uses the skeleton of the object to determine the segmentation. The method most used to extract the skeleton is the *Reeb Graph* [2]. For instance, in [30] an enhanced topological skeleton is extracted, for which are initially are found the feature points, located on the extreme of the object, and found using the geodesic distances. These are used to create a function that indicates the distance from a given point of the mesh to each feature point of the mesh. Then the constraints are identified and the *Reeb Graph* is built, which uses these constraints to subdivide its components. After having the skeleton, the object is segmented in the areas where each skeleton node corresponds to a segment of the object. This approach is classified as part based because the authors attempt to create a meaningful decomposition from a human point of view so it can be used in shape modeling, deformation and retrieval applications.

A recent approach, referred in section 3.1, uses the *Shape Diameter Function* (*SDF*) values, which represents the shape diameter, and uses it to extract the skeleton [28]. The algorithm starts by choosing N points distributed over the object, then calculates the *SDF* values and divides them in half. Next it projects the point to a point inside the object, that is, in the opposite way of the object normal on that point, for which the distance between the new and the old points is the previous calculus. The skeleton is then built by uniting the different points found on the previous step. As this approach uses the shape diameter it is classified as surface based segmentation.

2.2.4 *Geometry and Structure-Based* Segmentation

The segmentation approaches that are based on the geometry and structure of the object use this kind of descriptors, which can capture the main features of a given surface and must be based on the shape of the object. The *Taylor* approach proposed by Mortara et al. [21], starts with a point on the model chosen by the user, around which is drawn a sphere, having the point as center and radius r, which will progressively increase, generating cuts on the object in the intersection areas between the sphere and the object. The radius stops increasing when there is a drastic change on the cut, for instance, in the size of the cut or if it passes from one cut to two cuts, which represents a bifurcation of the object. With the evolution of the radius of the sphere, it is possible to have information about the shape of the object structure. If the sphere just creates one cut, that is, if there is just one intersection between the sphere and the object, we can conclude that the surface around that point can be considered a disc and that the segment produced by the cut can be considered a sharp or round zone or in the presence of a blended vertices. If there are two cuts, it means that we are in a presence of a cylinder or a conic zone. This changes according to the threshold between the two cuts rays. If the sphere intersects the object more than twice, it means that we have a ramification of the object.

As an extension to the *Taylor* approaches there is the *Plumber* approach whose goal is to find the tubular features of the object. The previous algorithm is initially executed to identify the seeds vertices of the possible cylindrical or conical sections. By agglomerating the candidate seeds, the seed zones. The first seed zone chosen is called medial loop, whose goal is to find the tubular zones are created. In this method the sphere center is the center of the loop and it grows until it sweeps the entire tubular zone and then it passes to the next possible tubular or bifurcate zone. The loop stops when the size of the

[2] *Reeb graph* was initially proposed by the mathematician George Reeb in 1946 in [24]. This approach was quickly adopted by the computer graphics because it has showed that it is a good structure to solve problems of shape correspondence and morphing. This kind of structure [7] are used in segmentation and many other applications. It can be considered a shape descriptor because it saves the topological information of the object. Each node of the *Reeb Graph* represents an extreme or a critical point of the object and the edges the connections among them. The critical nodes are points where there is some change on the shape of the object, for example, the increase or decrease of the width in some place on the shape or where it has a concavity, etc. There is an extension to this approach which instead of using critical points, uses critical and regular areas. The critical areas are the extreme zones, minimum, maximum and for example areas with concavities. In this extension the graph nodes are the centers of these areas and are extracted based on the shape contours. The edges are obtained by expanding the critical areas [3].

intersection is bigger or smaller than a given threshold, when it reaches the end of a tubular or a bifurcate zone. In the end of the algorithm, the zones that were not detected as being tubular are classified as body parts of the object.

Mortara et al. in [22] uses the *Plumber* approach to find the tubular zones in objects that represent the human body. They present a solution that finds the human body members, based on the relationship between them, for instance, the size comparatively to the group of tubular features, the distances between the others zones. For example we can think of a hand, the five fingers are going to form small tubes with sizes not very different that are connected to the hand palm, and with this information the algorithm can realize that these small tubes are the fingers. This information of the spatial relation between the tubular zones is obtained based on a connection graph, whose nodes represents those zones and whose edges represent the relations of adjacency between them. Both solutions presented are surface based because they use the surface criteria to segment in this case the intersection of the sphere with the object.

2.2.5 *Watershed Based* Segmentation

In the beginning this approach was used for segmenting 2D images. It was proposed by Ioannis Pratikakis in [23] and later extended to the 3D object segmentation by Mangan et al. in [20]. The basic idea of this technique is to fill up the object with water as it were a recipient. The water goes filling the watersheds of the object and separation lines are created in the junction points. In the end, when the watersheds are flooded, the process stops. As result the object is segmented and the watersheds represent the object segments. Zuckerberger et al. in [32] describes two algorithms that use this analogy of flooding the object sections, flooding convex decomposition and watershed decomposition. The flooding convex decomposition segments the objects based on his convexity. This algorithm uses the analogy of the watershed that is being filled until it overflows, and then it restarts on another watershed, each one of them representing, in the end, a segment. For that, it uses a dual graph. Initially it starts in a graph node and goes traversing the graph, agglomerating the faces while these form a convex segment of the object. If two faces cease to form a convex segment after agglomerating, the cycle stops and restarts on another point of the mesh. The watershed decomposition analogy is like following a drop “downhill”. The algorithm starts by looking and labeling the minimal areas and then does the same for the plateaus areas. After the labeling, a loop is executed, which descends the plateaus until it finds a labeled area. This approach uses surface criterion, such as convexity, making it a surface based approach.

2.2.6 *Explicit Boundary Extraction*

This technique extracts the contours of the segmentation boundaries, that is, the lines that separate each segment. There are many ways of extracting these contours and they are based on several features of the object, like for instance the concavity. After this extraction, the algorithms that use this technique are able to determine the segments boundaries. According to Lee et al. the human being usually segments an object based on its concavity [18]. Figure 2.4 shows that the algorithm has four main steps. The first step assigns to each vertices of the mesh its respective feature vector that in this case represents the concavity value on that point. After those values have been assigned, the contour curves in those point are extracted from the object. In the second stage a contour is chosen based on its length and on its centrality in the mesh. To create the segment it is necessary to close the contour curve, for this, the algorithm finds the shortest path between each extreme point of the curve, as can be seen in figure 2.4 on the phase Feature Contour Completion. To finalize the resulted segments are tests to for meaningfulness. So, this technique is classified as part based.

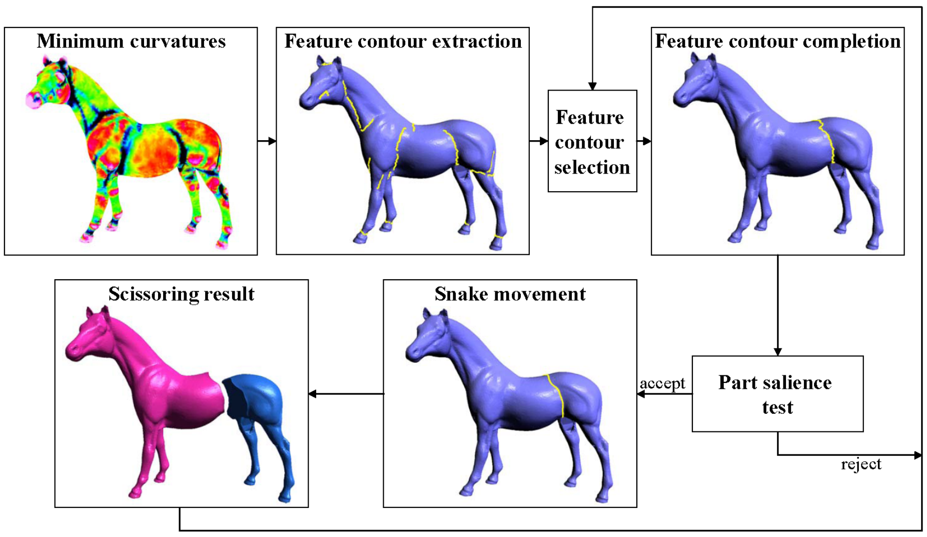


Figure 2.4: Steps of *Explicit Boundary Extraction* algorithm. (Image taken from [18])

2.2.7 *Random Walks*

In [17] Lai et al. proposes two similar algorithms, an iterative and an automatic that uses this approach on 3D object segmentation. In general the algorithm distributes seeds over the mesh, and then to each adjacent edge of each face is attributed a set of probabilities. These correspond to the probability of a random walk passing an edge of a non-seed face to reach one of the seeds. Each segment is represented by the face. To build them each face is assigned to the segment whose probability of reaching the seed that belongs to the respective segment is bigger.

The difference between both methods is on the seed selection. In the iterative method the seeds are selected by the user and then distributed through the mesh where each seed corresponds to a final segment. In the case of the automatic method, the seeds are automatically distributed on the mesh distribution is based on the salience of the object and on the regions that contain important features to the segmentation. Initially to find these areas it does the remeshing and then determines the seeds. This classifies the approach as surface based.

2.2.8 *Symmetry Based* Segmentation

Many objects are symmetric. Simari proposed an algorithm that segments the object based on its symmetries [29]. Initially the algorithm searches the maximum of symmetric parts of the object. For that, it uses the *robust M-estimation* using an *iteratively reweighted least squares* (IRLS) algorithm [13] then the symmetric sides are devised using the planar symmetry. The object has symmetry relatively to a plane, that is, by intersecting a plane with the object, both sides that are separated by the plane are symmetric. After the intersection one of the sides is discarded and the other is divided again using another plane and repeats successively until there is no more symmetry in the object. By this process a hierarchical tree is generated, which is called folding tree and represents the decomposition of the object. The root is the entire object, the children are the parts segmented by the planes, that is, each edge has the respective symmetric side and so on until it reaches the leafs. One of the stop criteria is when the number of faces on the mesh is lower than a given threshold. Another stop criteria is when the area of the mesh is lower than a given percentage of the object, and still another one is when the number of recursive folds exceeds a maximum. Because this approach uses the symmetry to segment an object, it is classified as surface based.

2.2.9 *Convexity Based* Segmentation

For some authors the main feature of an object is its convexity, that is, if we segment an object in the convex zones it creates sections with meaning to the human being. An example already referred in 2.2.1 uses the convexity as a growing criteria, as it agglomerates faces until the convexity property is violated. In a similar way the *Hierarchical Cluste*r approach described in 3.3.2 uses the convexity to calculate the clustering costs [6]. This approach is labeled as surface base since it uses the convexity to segment.

Another way to use the convexity to segment an object is by calculating and assigning its convexity value for each point of the mesh. Then a cyclic graph is built, whose nodes represent points of the mesh with similar convexity values and the edges represent the connections between them. After having the graph, the object is segmented by crossing a plane in that points [19]. In this case the approach is classified as part based since it uses the convexity of the object to produce meaningful segments.

2.2.10 *Feature Points and Core Extraction*

Katz et al. proposed an algorithm that uses *Feature Points and Core Extraction* [15] to produce a meaningful segmentation. The segmentation can be pose variant, that is, the same object with different poses can generate different segmentation results. For instance if an object that represents a human body is sitting and the same human representation is standing up, the results of the segmentation can differ. To overcome this problem and the problem of noise sensitive, the algorithm initially applies a mesh coarsing and then it finds a pose invariant representation of the object (Figure 2.5. a). To accomplish that, it uses a Multi-dimensional Scaling (MDS) [9]. After having this abstraction, the feature points of the mesh are extracted (Figure 2.5. b) using the geodesic distances and the convex hull. To get the core (Figure2.5.c) a spherical mirror is used around of the pose invariant representation and similarly are detected the remaining segments of the object where each segment corresponds to one of the feature points previously calculated. To finalize, and as in the first step a mesh coarsing was applied, it is necessary to revert this process, that is, to refine the mesh and to present the segmented object in the original position (Figure 2.5. d).

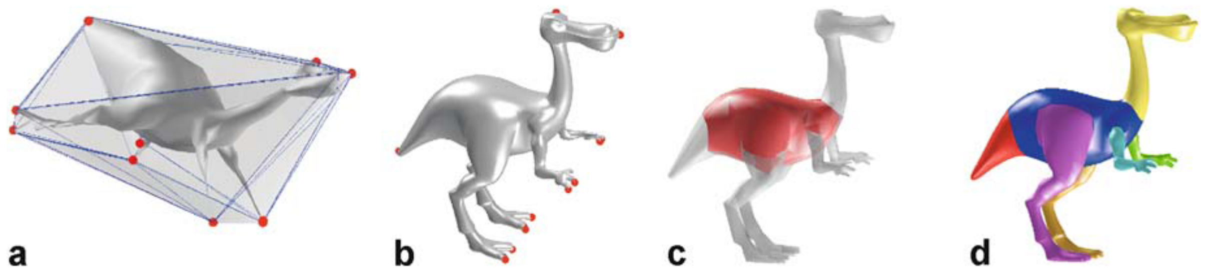


Figure 2.5: Steps of *Feature Points and Core Extraction* algorithm. a) MSD shape; b) Feature points; c) Core; d) Final Result. (Image taken from [15] ©Springer-Verlag 2005)

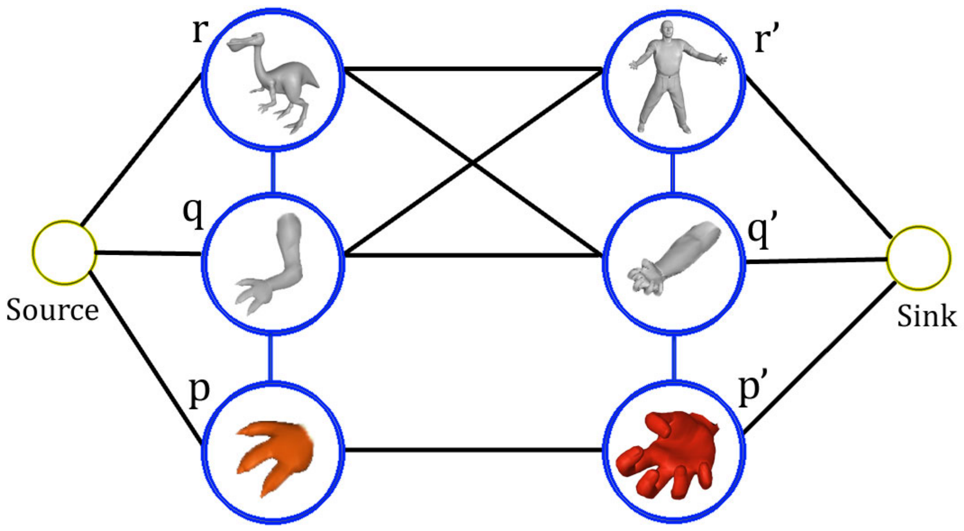


Figure 2.6: Example of a graph used to measure the similarity between two segments. (Image taken from [27] ©Springer Sience+Business Media, LLC 2009)

2.2.11 *Segmentation using Shape-Diameter Function (SDF)*

The *Shape Diameter Function* (SDF's)[3] values are calculated and used by Shapira et al. in [28] to detect to which segment a given point belongs. The points that have similar *SDF's* are grouped using an iterative clustering, which generates the object segments. The second step of the algorithm uses the *K-Way* technique to smooth the segments boundaries. In [25] and [27] the *SDF* is used to identify similarities between objects. The algorithm previously described is initially used to segment the object. After having the segmentation, in the next set it is going to calculate again the *SDF* value for each segment and then successively until it reaches a maximum of segmentation. In result of the recursion a hierarchical tree is generated where the root node represent the entire object and the levels below represents the recursive segmentations. The algorithm concept is that besides calculating the distances to find the similar segments, the proposed algorithm also takes into account the segment context, for instance, if it is a hand in a human body representation. To get the object context, the algorithm uses a graph that has the distances between the segment and the root node, that is, the entire object. If we want to compare two segments of two different objects using this algorithm, we use a graph (Figure 2.6) that contains the two different objects on the top of the graph and the nodes are the segments of both objects, that were founded during the algorithm until it reaches the final segments that are going to be compared. This graph is a flow graph, so, two extra nodes are added, a source node that is connected to the nodes of the first object and the node in the opposite side that is the sink and is connected to the nodes of the second object. The nodes of each object are connected between each other, as can be seen in Figure 2.6, where the p and p' nodes are connected between each other. Then the nodes in the upper hierarchy (r, q, r', q') are connected between each ones. Then the maximum flow through the graph is calculated , this value representing the distance between two segments and the similarity between these segments, based on its context. This approach is classified as part based because it is used in a shape retrieval application which need meaningful segments.

[3] *The Shape Diameter Function* is an example of descriptors presented by Shamir et al. at [28]. For each point on the mesh it calculates the diameter of the object on the neighbor of the point. To calculate this value, a cone is drawn, the vertice is the point and the base is on the opposite direction of the normal of the point, it has an angle and the height is the diameter of the object on that point. That is, several radius are drawn inside the cone, they reach the other side of the object. The average of the values of these radius is calculated and saved on a shape descriptor

2.2.12 Summary of Segmentation Approaches

For several years segmentation approaches were proposed, each with particular characteristics and purposes. The approaches referred in this section are the most commonly used and relevant for this master thesis.

The first approach described is the *Region Growing*, which starts by visiting a seed (this is a face of the mesh), then agglomerates the adjacent faces by traversing the mesh and it stops when it reaches a stop condition, like the convexity, and restarts the process on a new non visited seed. In a similar way the next approach also agglomerates faces,like in the *Clustering* approach but instead of creating patch by patch, it creates all patches simultaneously. This is constituted by two different types. First the *Hierarchical Clustering* where a hierarchical tree is formed, in which the leafs are the mesh triangles and the root is the entire object. This tree is built bottom to top and it clusters adjacent faces with the minimum merging cost that can be calculated on different ways, like fitting the resultant cluster to a primitive. Then in the *Iterative Clustering*, given a number of clusters, several results of segmentation are generated, then it chooses which is the best segmentation. It starts by creating a representative group of clusters and then each adjacent face is clustered until it reaches a radius r.

Some solutions use the Skeleton based segmentation where the first stage is to generate the object skeleton using different techniques in which one of them is the *Reeb Graph*. Then, it uses the information of the skeleton, like where there is an abrupt change on the object shape, to create the segmentation. Another approach that uses the object shape to segment is the *Geometry and structure-based* segmentation; they are the *Taylor* and *Plumber*. These have the goal of detecting tubular features by blowing bubbles that starts on a seed that is predefined and stops blowing when it finds an abrupt change on the object shape like a for example a bifurcation.

The *Watershed based* approach simulates the analogy of leaking water to a basin and the stop condition is when the basin is full of water and a separation line is created in the junctions point. In the end, the filled basins represent the segmentation results. As the name itself indicates, the *Explicit Boundary Extraction* extracts initially the contours of the segmentation boundaries, which can be extracted in different ways based on the object features like the concavity.

Many of these approaches merge adjacent faces. Another example is the *Random Walks* approach which also uses initially seeds that are distributed by the mesh and then for each adjacent face it calculates the probability of a random walk passing by it when reaching a seed.

One feature that is also important to human beings is the symmetry, so an approach was proposed by Simary in [29], that uses this feature to decompose the objects. To accomplish that, it uses planar symmetry, then it ignores one of the sides and repeats the process on the other side to create a hierarchical tree. Another feature that has showed to be increasingly important is the convexity, so the solutions that are *Convexity based* segment the object based on this feature and generates convex segments.

In an attempt to overcome the pose variant limitation a new approach arises called *Feature Points and Core Extraction*. This initially creates a pose invariant representation of the object, and then extracts the feature points and the core. After finding the core it is necessary to extract the rest of the segments which is done by matching each part to the feature points. In the end, it reverses the initial process so the object can come back to the same shape.

A recent approach is the use of *SDF* for segmenting objects. In summary this values give the diameter of an object in a neighbor of a point and then is used to merge the points that have the same or close diameter values using a histogram.

All of these referred approaches segment 3D objects individually, instead of segmenting various objects simultaneously. The latter is a recent concept and it is used in the solution proposed on this document. Next we present a study containing various existing approaches on this concept.

2.3 Segmentation of 3D objects from collections

The approaches previously described segment only one object, and do not take into account other similar objects. This is a recent concept that has few studies and not much work done on the topic. However, one of the most prestigious investigation groups in the 3D shapes analysis area, the group of *Thomas Funkhouser of Princeton* has presented an algorithm [12] that uses the information of similar objects. The first step of the algorithm is to build a graph whose nodes represent the mesh faces, the edges represent the edges of the mesh that connects adjacent faces (Figure 2.7 – Green Lines) of the same object and they also represent the correspondence between the faces of different meshes (Figure 2.7 – Reed Lines). This correspondence between the different objects makes the correspondent parts of the mesh aggregate in the same segment. In the next step the algorithm is going to execute a hierarchical clustering of the graph, where the adjacent faces of the same model and the correspondent faces of different models are going to probably belong to the same segment. The result of this approach is a meaningful segmentation as can be seen in Figure 2.7 where the back of the chairs are separated from the legs and from the sitting area and even each leg is separated.

Another and different approach uses a different concept, which was proposed by Kalogerakis et al. in [14]. This new approach was developed in order to segment and label 3D mesh parts simultaneously, producing a meaningful segmentation. The solution uses a collection of labeled training meshes that are previously decomposed and labeled by users. Then, with this group labeled and with all features identified, it segments based on this data one object at a time. The main limitation that they refer that is been overcomed is that do not need manual parameters but instead it needs a trained collection of meshes. For this solution the goal was to segment and label an object while for the solution presented on this document the main goal is to segment a collection of objects automatically based on geometric similarity of the elements of the same collection.

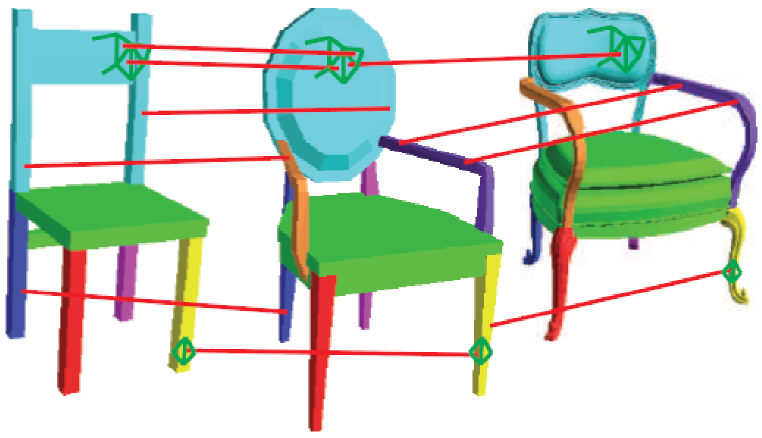


Figure 2.7: Representation of the graph in a set of 3D objects. Green Lines – represent the edges that connect the adjacent faces of the same object. Red Lines – represent the correspondence between the faces of different meshes. (Image taken from [12])

2.4 Comparative Analysis

The segmentation approaches have similar goals but different purposes, since they are application dependent, that is, they depend on the application that will use the segmentation, since collision detection and animation to 3D object indexing and retrieval and on the object type like for instance CAD. So, to evaluate and compare segmentation approaches is not an easy task to do. There already published several comparative studies that use different comparison metrics. Some of these studies are very detailed, the most relevant are [1], [26], [5] and [3], they demonstrate that compare techniques is a challenge. Because the segmentation usually is a stage of different applications, there is not a heightened concern as long as the final result is the expected result, which generates several limitations on the segmentation approaches. To define which ones are the most significant, it was made a survey of the disadvantages and the advantages of the studied approaches. This analysis will provide the necessary information to create a new solution that can overcome the limitations getting a better result. The tables 2.1 is a resume of the survey.

2.4.1 Limitations of 3D object Segmentation Approaches

During the process of analyzing the approaches some limitations were found in order to develop a new improved solution that will overcome these limitations.

On this section are described the most relevant limitations of each approach that are important to this project. One of the most common limitations is the inputs provided by the users need in order to decompose the objects, the approaches need information provided by the user. Also another common limitation is the over-segmentation, this means that the result of the segmentation has more segments that it should have.

Some approaches results do not detect all the segments that compose the object, this limitation is called, the ability of detecting all segments also it happens that after the decomposition, other approaches do not generate smooth boundaries, that is, the boundaries are serrated. Another limitation given the object shape is the pose invariance, like for instance human body representations, where it can be sitting or standing up, after the results of segmentation are different.

Because one of the goals of this proposed solution is to segment a collection of objects, a study on this particular subject was made, where segmenting object by object, that is, having an individual segmentation is a limitation because on the case of existing a collection of objects, the segmentation of these objects takes a long time.

Inputs provided by the users

Most of the 3D object segmentation approaches needs some specific information provided by the users. For instance, in the approaches *Region Growing and Random Walks* the user needs to choose the object seeds that are going to form the most relevant segments. In a similar way, in the *Taylor* and *Plumber* the user needs to indicate which the points represents the center of the sphere, which in the end will correspond to the final representation of the object segments. In the case of the *Iterative Clustering* and segmentation on 3D object sets [12] it is provided the final number of segments resulting of the segmentation. In the previously mentioned approaches the interaction with the user is done in the beginning of the algorithm. In the *Hierarchical Clustering* it is done in the end, that is, after having built the hierarchy of segments, the level of the hierarchy that is meaningfully to the user is chosen by him. The same happens with the *Watershed-Based* approach were it can be necessary, at the end, for the users to complete the boundaries or choose the segments that can be ambiguous.

Over-Segmentation

This limitation is common on the several approaches that use Surface-Based segmentation. One example of over-segmentation is the *Region Grow* approach described in [32] where one of the agglomeration criteria used is the convexity of the object, if the object has many small and convex areas, all of them will be identified has different segments this also happens on the Watershed-Based approach where is also used the convexity to decompose an object.

In a similar way, the *Hierarchical Clustering* and *Random Walks* approaches can produce over segmented results.

Ability of detecting all segments

Many times, the objects have less evident details or features, so the object becomes as realistic as possible. In these cases some of these details or features are not detected in the segmentation. This slip happens on *Taylor* and *Plumber* approaches while, for instance, it is segmenting an object that represents a human body with a very short neck or short fingers, the algorithm may not detect these tubular zones. Another example is a object that represents a fat human body, in this case, the legs join sooner that it should, so, after segmenting, the legs would be shorter in relation to the arms length [22]. Other features that may not be found are the fingers of a closed fist. In a similar way, the solutions *Skeleton based* does not detect the fingers on a closed fist and does not detect also structures with reduced size[30]. This failure on detecting segments happens in a similar way on the solution present by Katz et al. [15], whos approach denominated, *Feature Points and Core Extraction.* In this case the problem is the impossibility of detecting some boundaries, that is, only detects one segment when it should detect two. The example shown is a lion whose extremity of the tail forms a closed loop with the back, so when is going to segment, the algorithm considers that the tail and the back forms only one segment. The solutions that use the *SDF's* values have the same limitation as to the parts detections, because it bases on the width of the objects. In the case the object not having cylindrical zones, or with very reduced width, they will not be detected by the algorithm.

Generating smooth boundaries

Another common limitation is the existence of boundaries that are not smooth when finishes segmenting. This happens because usually the objects are represented by a polygonal mesh. This problem appears in solutions that use the *Watershed based* and *Feature Points and Core Extraction* approaches. To overcome this limitation, many of these solutions have a post processing stage that smooths the segments boundaries.

Individual Segmentation

In the case of having a collection of objects and the need of segment all of the objects in that collection, by doing it object by object is slower than segmenting all the objects simultaneously. Since the goal the work proposed on this document is to segment a collection of objects simultaneously, it is necessary to understand what solutions have some concern in this field. This is a recent concern, only in the last two years were presented solutions that overcame this limitation, as in [12]. Besides that, this solution still has a big limitation because it depends on the alignment and similarity of the objects of the collection to segment the object.

Pose Invariant

In computer graphics it is important to create realistic scenarios and scenes, so for that, are created several objects that may move on the scenario causing several different poses of the object. So, this may cause different segmentation for the same object but in different poses. The solutions that uses the *Region Growing* or the *Hierarchical Clustering* or the *Iterative Clustering* or the *Skeleton Based* [30] or the *Explicit Boundary Extraction* or the segmentation on 3D objects sets in [12] this is one of the main limitation.

2.4.2 Advantages

As the limitations, some advantages were highlighted during the study of the approaches. Some of these advantages are an attempt to overcome the referred limitations.

To overcome the previous limitations most of the approaches use preprocessing or post-processing stage, with the first they try to overcome the pose invariance it finds on the beginning a pose invariance representation of the object. With the post processing, they try to overcome the over segmentation or also by defining some limitations like the number of patches on the beginning of the execution of the approach.

Few of these advantages do not correspond to a limitation. One, is the extra information about the objects, that is, some solutions can give extra information about a segment of the object or of the object, like for instance the thickness. Also, another type of information that can be provided by the approach is the topological information that gives the perception of distances between the object parts. The hierarchy is another advantage but it does not give information about the object, it gives to the user the possibility of choosing which segmentation is more meaningful to him.

The main goal of some of the identified common advantages is to overcome some of the referred limitations. Some of the most important advantages are, pose invariance, Extra and topological information of the object , hierarchy and over-segmentation.

Pose Invariant

As it was referred in section 2.4.1 some segmentation approaches presented depends on the object pose. But some of them try to overcome this problem, in order to do that, most solutions use a preprocessing stage where they try to find a pose invariant representation of the object. Solutions that use the *Feature Point and Core Extraction* use this kind of techniques to overcome this limitation. The *Geometry and structure-based* segmentation or the *SDF* approaches are based on the cylindrical shapes or in the width of the segments so, on the solutions that use these approaches this limitation does not occur.

Extra information about the object

As the *Geometry and structure-based* segmentation and the *SDF* approaches are based on the cylindrical shapes or in the width of the segments, this information can be provided. It also can then be used by the application where the segmentation is built-in. In these approaches the main information obtained is the object width. Another kind of information that can be extracted of the object from the segmentation is the semantic of each segment in the case of the object to be a representation of the human body. In the solution presented by Mortara et al. in [22] it is used the *Plumber* approach that detects the body parts. Initially it is found the tubular zones of the object, then it is compared the distances between each segments founded and at last it is assigned to each a semantic, so it is possible to get the width of the tubular segments.

Topological Information

In the case of *Skeleton Based Segmentation* the biggest advantage is the fact that besides of using the geometric information of the object, it also uses the topological information, this generates a better segmentation because it uses the spatial relation of the object. For instance, on solution proposed by Julien Tierny at [30] where it first builds the enhanced topological skeleton using this, it first delimits the core of the object then identifies the junction areas and finally this topological information not only is used to segment the object but also to label it.

Hierarchy

Some solutions studied, generate a hierarchy of segments that has several segmentation levels, these provides to the user an option to choose the most meaningful result.

In the *Hierarchical Clustering* in section 2.2.2 all solutions that use this approach generate a hierarchical tree that has several segmentation levels and produces different segmentation results depending on the level. The trees are also used to transverse and during the process create segments, meaning that not only the levels produce the segmentation.

Over Segmentation

This is one of the most common limitations of the approaches. The *Watershed-Based* and *Random Walks* approaches tries to overcame this limitation by using a post processing stage that agglomerates the smaller segments based on rules or by initially predefining a maximum number of segments. The *Hierarchical Clustering* overcomes this limitation by initially setting a maximum threshold of deepness introduced by the user or even during the execution the user can chose the deepness more meaningful to him.

Approaches		Articles	Limitations	Advantages	Part Based	Surface Based
Region Growing		[32]	User Interaction Over-Segmentation		X	
		[31]	User Interaction	Segments and label simultaneously	X	
Clustering	*Hierachical*	[11]	User Interaction Over-Segmentation	Generates a hierarchical segmentation Generates smooth boundaries		X
		[4]	User Interaction Over-Segmentation	Generates a hierarchical segmentation Generates smooth boundaries		X
		[6]	User Interaction Over-Segmentation	Generates a hierarchical segmentation Generates smooth boundaries		X
		[10]	User Interaction Over-Segmentation	Generates a hierarchical segmentation Generates smooth boundaries		X
	Iterative	[2]	User Interaction			X
		[16]	User Interaction	Generates a hierarchical segmentation	X	
Skeleton Based		[28]	Failed detection of some segments	Analogue skeletons on the analogue parts of different objects Huge applicability in animation and distortion Pose Invariant		X
		[30]	Failed detection of some segments	Uses topological information Huge applicability in animation and distortion	X	
Geometry and Structure-Based		[21]	User Interaction Failed detection of some segments	Give extra information about the object (width) Pose Invariant		X
		[22]	Failed detection of some segments	Give extra information about the object (width) Detects human body parts Pose Invariant		X
Whatershed Based		[20]	User Interaction Over-Segmentation Generates boundaries that are not smooth or not closed	Pose Invariant		X
		[32]	User Interaction Over-Segmentation			X
Explicit Boundary Extraction		[18]	User Interaction	Generates Smooth and closed boundaries	X	
Random Walks		[17]	User Interaction			X
Symmetry Based		[29]	Over-Segmentation	Generates Smooth boundaries		X
Convexity Based		[6]	User Interaction	Generates a hierarchical segmentation Generates smooth boundaries		X
		[19]	Over-Segmentation		X	
		[32]	User Interaction par Over-Segmentation		X	
Feature Points and Core Extraction		[15]	User Interaction Failed detection of some segments Generates boundaries that are not well defined	Avoids over-segmentation Pose Invariant	X	
SDF		[25]	Failed detection of some segments	Give extra information about the object (width) Pose Invariant	X	
		[27]	Failed detection of some segments	Give extra information about the object (width) Pose Invariant	X	
		[28]	Failed detection of some segments	Give extra information about the object (width) Pose Invariant	X	
Segmentation of 3D objects from collections		[12]	User Interaction Depends on the object alignment	Segments simultaneously an entire collection of objects	X	
		[14]	The needing of using a trained set of objects	Does not need inputs from the users	X	

Table 2.1: Table with the resume of the related work

2.5 Discussion

Several segmentation approaches have emerged as decomposition of 3D objects as is used in many different applications in distinct domains that need different segmentation results. This also makes the task of evaluating the approaches hard to perform, since there is an extensive range of approaches. Some surveys on this subject were already performed by researchers. On these surveys, the authors classify the segmentation approaches as part based or surface based, depending on the segmentation accomplished. Also on this chapter we have described some of the most common limitations and advantages and emphasized the most relevant to this work.

As all approaches are different there are no better approaches than others, each has limitations and advantages, even though, with these studies and the evolution of these approaches, some tries to overcome some limitations that are present in previous solutions. For instance, the *Feature Points and Core Extraction* try to overcome the problem of pose invariance using a pose invariance representation of the object. This limitation is present on *Region Growing* and *Clustering* among others. Many of the approaches that are Surface Based can generate an over segmentation because it uses the surface to segment the object. In the *Region Growing* approach is used the convexity to segment, and if there are many small convex patches, this will be identified, to overcome this problem it uses a post-processing stage that removes the extra segments by merging them to others or by initially predefining a maximum number of segments. The Hierarchical Clustering approaches can avoid producing over segmentation when users choose the results. A very common limitation, almost in all approaches is the needing of user interaction. Some need to predefine the seeds like in *Region Growing* and *Iterative Clustering* approaches. Others to select the level of the hierarchy generated by the segmentation approach, these are all the approaches that produces a hierarchical segmentation. The approach that overcomes this limitation is the one proposed by Kalogerakis et al. in [14] by using a collection of labeled training meshes. As refereed, it was emphasize the most relevant to this work. The individual segmentation of objects can be consider a limitation if we consider segmenting a collection of objects. If the segmentation decomposes object by object as it is in the major approaches, then it can take more time to decompose the entire collection than if segmentation was simultaneously.

In order to overcome some limitations highlighted in this chapter, as the need for user inputs, the over-segmentation and the individual segmentation, a different solution is proposed on the next chapter. This is an evolution of the original solution. The main differences between them are highlighted and the new architecture and decisions are also described in the next chapter.

Chapter 3

Collection-aware Segmentation

We present in this document the adaption to object segmentation of an algorithm which initial purpose was to extract object parts for indexing. The *Original Collection Aware Segmentation* (*CaS*) algorithm, proposed by Alfredo Ferreira et al. in [10], was embedded on an indexing and retrieval solution. The adapted algorithm isolates segmentation stage from the rest of indexing and retrieval solution and improves it.

3.1 Approach Overview

In this section we present an overview of the *Adapted CaS* approach. However to understand this adapted technique it is necessary to first have a look at the *Original CaS approach.* One of the main goals was to improve the algorithm aiming only at purpose of the segmentation, that is, returning only the segments that are important to the segmentation, instead of the purpose of the *Original CaS* that was to return all the segments resulting from the segmentation. Because of these and other goals, the architecture of both approaches are different, so this section will also highlight these differences.

3.1.1 *Original CaS* for Retrieval

The original version has two stages as illustrated in Figure 3.1. The first is the initialization where the foundations of the segmentation are computed and loaded. The second stage is the iteration were the objects of the collection will be automatically segmented.

It starts by generating for each object of the Model Collection the respective Hierarchical Segmented Mesh (HSM) using the *Hierarchical Fitting Primitives* (*HFP*, [4]) approach that consists on merging clusters and then fitting them to primitives to create the final clusters (detailed at section 2.2.2). Next it saves the HSM on the HSM Set and computes the object signature, Spherical Harmonics (SHA). This is a type of descriptor that represents an entire 3D object with a number. Finally it adds the object with the signature into the shape pool.

In the next step, the iteration stage, decomposability of each object on the shape pool is verified. In fact, the iteration stage is the iteration of the shape pool. This data structure will store the segment-signature pairs found during the decomposition and that are going to be used, in particular the signatures, to label a segment as decomposable or not. To be decomposable, a segment has to be considered unique, it cannot have a predefined number of x elements similar to it. This similarity is computed using the differences between the segments signatures in the Shape Pool and the segment to be decomposed. If the object is decomposable, it is decomposed by going to the respective HSM on the HSM set and getting the child nodes of the decomposable segment, then it computes the signatures of these new segments and

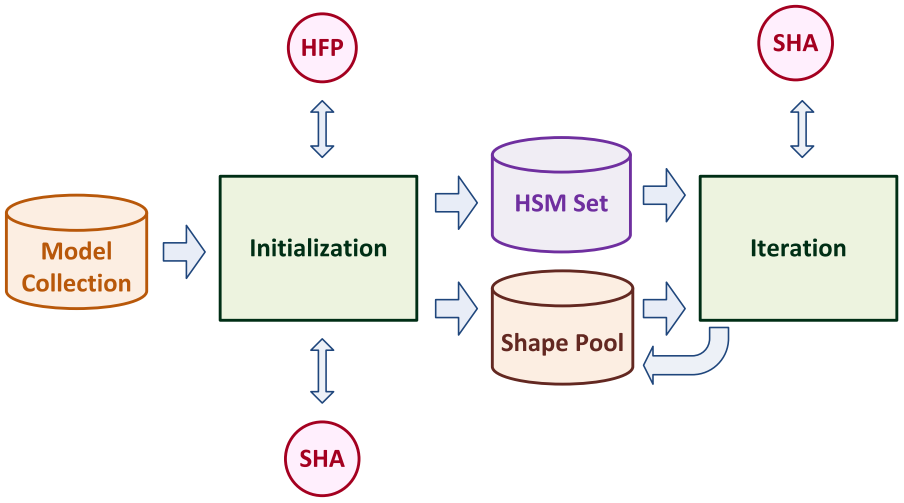


Figure 3.1: Overall architecture of the Original CaS, with the two stages (Initialization and Iteration).

adds the pair on the Shape Pool. If the object is not decomposable, it passes to the next element on the shape pool. This execution generates a second hierarchical tree that is built top to bottom and whose elements are the segment-signature pairs that are saved on the signature pool. Each iteration visits the next level of the HSM trees.

The iteration stage finishes when there are no more decomposable segments on the shape pool and the algorithm ends by returning this data structure which contains the segments that are going to be used to index the collection in the retrieval framework.

3.1.2 *Adapted CaS* for Object Decomposition

As the main goal of this dissertation is to adapt the *Original CaS* to perform segmentation in collections of 3D objects, the architecture was changed. After presenting the overview of the *riginal CaS* we now present an overview of the *Adapted CaS*, represented on Figure 3.2. Comparing both architectures in Figures 3.1 and 3.2 it is possible to observe the main differences.

One of the architectural difference is that the shape pool was substituted by the signature pool. In the *Original CaS* the main goal was to return all the segments resulting of the various decomposition levels for the retrieval framework, so it needed to save the segments in the shape pool. Now, the goal of the *Adapted CaS* is to decompose collections of 3D objects, but instead of having the signature pool with all the segments and respective information like the signatures, the signature pool just contains the signatures of all segments and is only used to label a segment as decomposable or not, improving algorithm efficiency.

Without the signature pool, we no longer create the second hierarchical tree, which was built during the iteration stage of the *Original CaS* to help guide the iteration by informing the next segment to be processed. Instead, the *Adapted CaS* uses two lists, the Non Decomposable List (NDL) and the Non Segmented List (NSL).

Another main difference is that to start the segmentation, first, in the initialization stage, all roots of the HSM's are saved in the NDL so they can be processed in the first iteration.

In the end, this approach returns the Decomposed Models Collection which contains all the decomposed objects that result from the Adapted CaS approach.

This section was only a brief overview of the *Adapted CaS* and main differences between both approaches. The next sections explain in more detail the data structures used, the stages of the approach and in the end how all of these are combined.

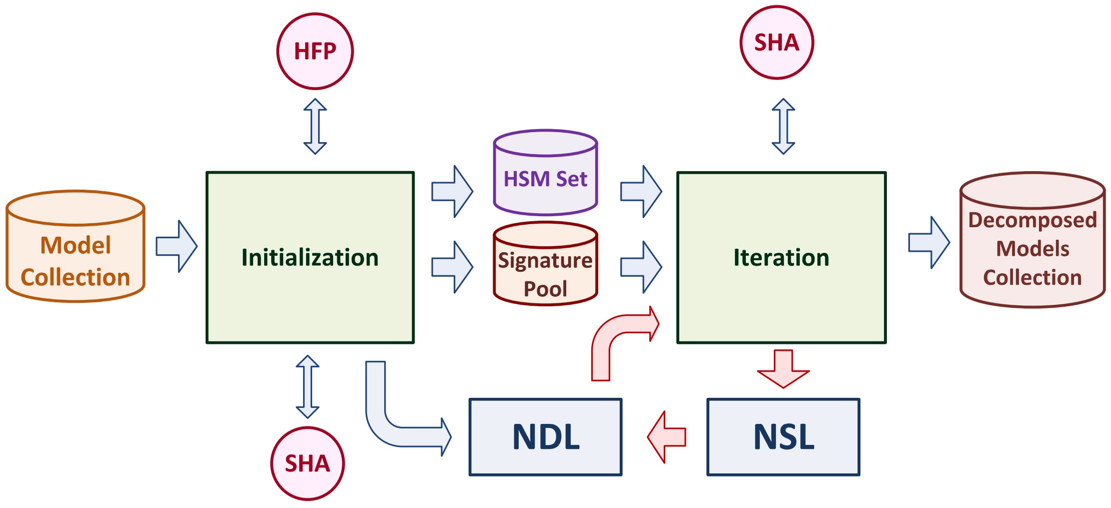


Figure 3.2: Overall architecture of *Adapted CaS*, with the two stages (Initialization and Iteration).

3.2 Data Structures

During the execution of the *CaS* approaches data structures are used to save segments and to help the iteration stage. Some of these structures are built in the initialization stage and others in the iteration stage. These data structures are also one of the main difference between original and the *Adapted CaS* approaches. The first used two hierarchical trees, one created on the initialization stage using the *HFP* and other built on the iteration stage. The *Adapted CaS* approach uses just one hierarchical tree and two lists of segments (NDL and NSL), being that both the tree and NDL are built on the initialization stage while NSL is built on the iteration stage.

3.2.1 Hierarchical Tree

The segmentation process of the *CaS* algorithm is based on a hierarchy of segments. The *original CaS* presented by Alfredo Ferreira et al. in [10] used two binaries trees. The Adapted version presented on this document uses only one binary tree.

In the first presented approach, the Hierarchical Segmented Mesh (HSM) is generated during the initial phase of the technique, using the *Hierarchical Fitting Primitives* (*HFP*, [4]) algorithm. This generates a hierarchical tree (Figure 3.3) that is build bottom to top. It starts on the end of the leafs of the tree, where each represents a cluster that is a triangle of the mesh, then it merges pairs of clusters forming new clusters that are then the parents of these pairs of clusters. The segments that are merged, are the ones that have lower cost of merging. This cost is calculated by fitting the cluster to each primitives, a plane, cylinder and cone, forming a new cluster. The process repeats until it reaches the top of the tree, where the entire object is represented.

In the *Original CaS* approach, this tree supports the iteration phase. It is used to build the second tree, since it contains the information of the next segments that are going to be processed. That is, the

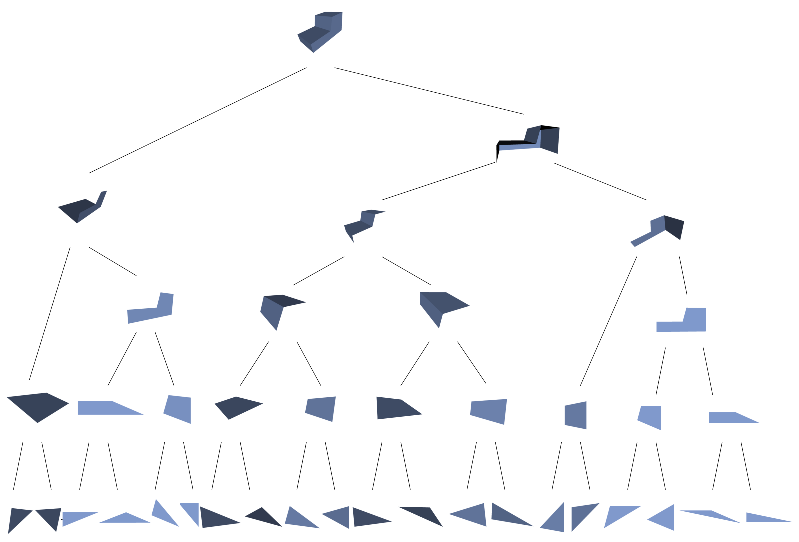


Figure 3.3: HSM tree produced by the *HFP* algorithm in the initialization stage.

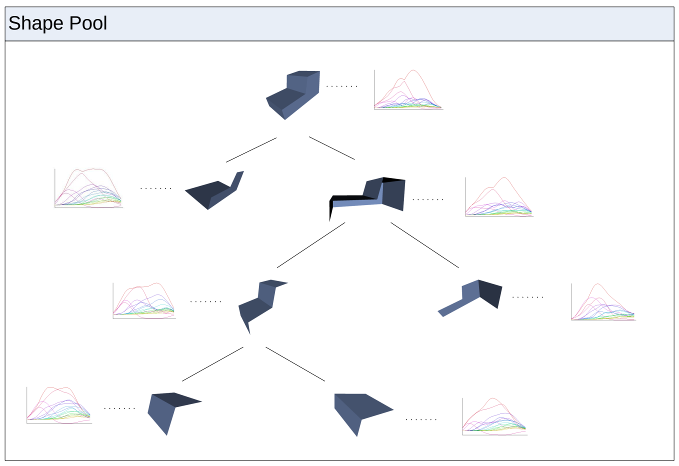


Figure 3.4: Tree produced by the *Original CaS* algorithm in the iteration stage.

HSM tree is built in the initialization phase and contains the hierarchy of segments since the object that is located in the root, to the leafs that contains the mesh triangles. The second tree is generated using the previous HSM to obtain the information of the next segments to be considered in the next iteration. This tree is built in the opposite direction of the first tree, top to down, starting on the root that represents the initial object, and transversing until it reaches the nodes that cannot be decomposed. This tree is actually saved on the shape pool (Figure 3.4) and it is formed by segments that have the information of the child and parent nodes, making it necessary to access many times the shape pool during the iteration phase. In the end of the iteration the shape pool containing all the segments discovered during the iteration is returned and then used on the next stage of the retrieval framework.

One of the architectural changes was to substitute the shape pool by the signature pool. So, the *Adapted CaS* just uses the HSM tree that is the same used in the *Original CaS* and is previously described. Then, in the initialization stage the roots are saved in the NDL and in the iteration stage the HSMs are transversed from the root to the leafs and the nodes that are important for the segmentation are provisionally saved on the NSL.

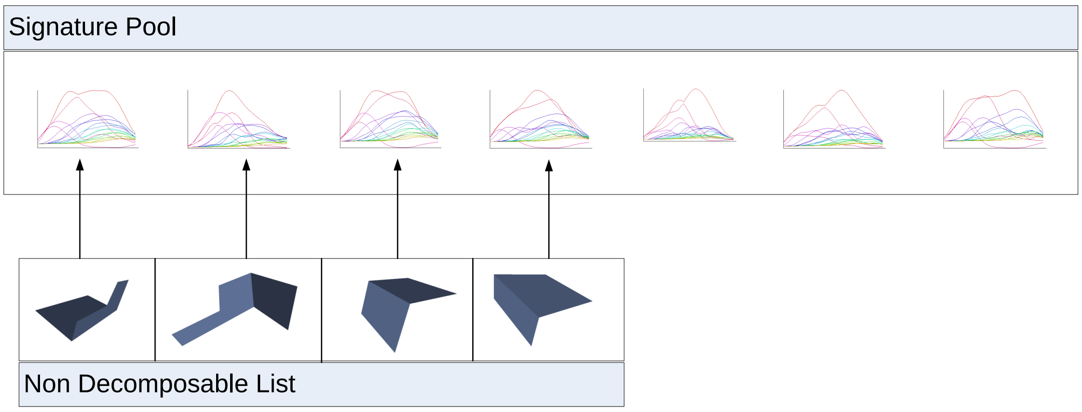


Figure 3.5: Representation of the Signature Pool and the NDL used on the *Adapted CaS* at the end of the execution filled with the final segments of the decomposition.

3.2.2 Signature (SHA)

The signature are descriptors which, in this work, are described by the Spherical Harmonics which are a numerical representation of a 3D object. It represents a 3D object as a point in a multidimensional space. In this particular case it makes the comparison of two objects easier to do since it is only necessary to compare two values, calculating the difference between both values using Euclidean distance [1].

If the difference is low, then they are close in the space, means that they are similar otherwise meaning that they are different.

3.2.3 Signature Pool

There is no longer a shape pool, so it was necessary to have a set to save the signatures as the segments need to be compared to each other. The Shape Pool was substituted by a signature Pool making the comparison faster and easier to do, thus it is not necessary to access the shape pool and build a tree inside the shape pool. Instead it just iterates the list producing a faster approach.

Each segment on NDL has a pointer to the respective signature that is saved on the signature pool. Whenever it is necessary to label a segment, it just iterates the signature pool and compares the object signature with all the signatures. In Figure 3.5 these two data structures and their links are represented. The signature pool has all the signatures found during the process of decomposition, while the NDL represented on the figure has just the important segments of the segmentation process to clarify the operation.

This figure shows that although the object is eliminated from the NDL, its signature remains in the signature pool so the next segments to be labeled are compared to it.

3.2.4 Non Decomposable List

As the new algorithm only uses one tree it is necessary for the algorithm to know which segment is going to be processed. For that, it is used the Non Decomposable List (NDL). This list contains the segments [2] that are going to be processed and also the segments that have been processed but are not decomposable.

In the beginning, during the initialization stage, this list is filled with the objects of the collection that are going to be segmented. Then, the iteration stage is the iteration of this list itself. It starts on

[1]Euclidean distance measures the distance between two points in the n-dimensional space, and is given by the Pythagorean formula. **Euclidean distance:** $d_2(x, y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$

[2]Segment is a node of the HSM produced by the *HFP* approach.

the first element of the list, and then proceeds until it reaches the end of the list. This iteration proceeds by verifying if each segment on this list is decomposable, if so, it decomposes the segment, removes it from the NDL and passes to the next element, otherwise it just passes to the next element. When it reaches the end, it verifies if the NSL is not empty, if not it appends the segments of the NSL to the end of the NDL and restarts a new iteration. In the other hand, if the NSL is empty, it ends the algorithm and returns the entire NDL. The purpose of this list is also to understand which are the segments that belong to the final decomposition.

3.2.5 Non Segmented List

Each iteration visits a level of the HSM. The first iteration visits the first node of the HSM (the entire object). In the second iteration, the child nodes of the first segment are visited in the case of being decomposable, and so on, until there are no more decomposable segments. So, just using the NDL was not enough because each iteration would not be executed level by level. Also on each iteration the segments that were labeled as non decomposable need to be processed with the new segments.

That is why we create the Non Segmented List (NSL). It contains the segments to be processed, found in each iteration. On each iteration in which a segment is decomposable, it goes to the HSM to get the child nodes of this segment, then it adds them to the NSL and removes the parent node from the NDL.

At the end of the iteration, when it reaches the end of the NDL, it verifies if this NSL is empty. If not, it appends all the segments of the NSL to the NDL, but before adding each segment, it computes the signature, saves them in the signature pool and erases the NDL. If the NSL is empty, then it means that there are no more segments to be decomposed, so the algorithm ends.

3.3 Segmentation Procedure

The *CaS* approach is constituted by two stages. The initialization stage where the data structures and all bases for the segmentation are built, and the iteration stage where the decomposition is really executed.

3.3.1 Initialization

In this stage all the segmentation foundations are built. Following the fluxogram in the Figure 3.6, the initialization starts on the first object of the collection and repeats the process for all the objects in the collection. Each step of the iteration stage is explained in the next paragraphs.

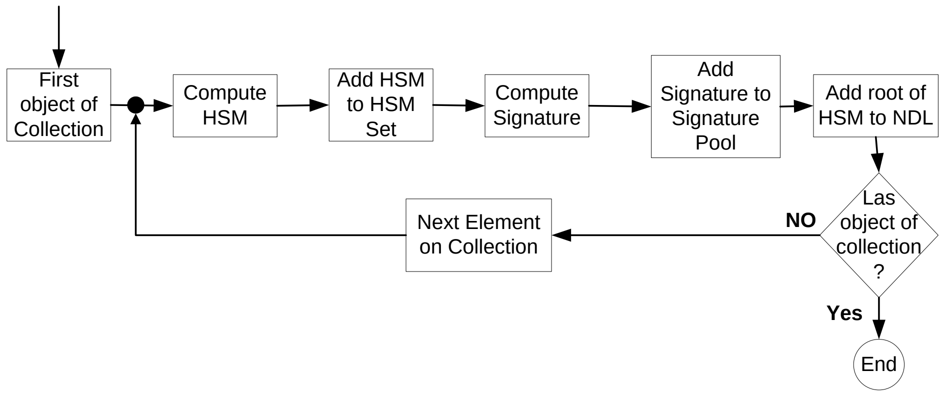

First object of Collection
Compute HSM
Add HSM to HSM Set
Compute Signature
Add Signature to Signature Pool
Add root of HSM to NDL
Las object of collection ?
NO
Next Element on Collection
Yes
End

Figure 3.6: Initialization Fluxogram.

Compute HSM

The iteration starts by computing the Hierarchical Segmented Mesh using the Hierarchical Fitting primitives. This structure is used to guide the iteration, it has the next segments that are going to be precessed.

Add HSM to HSM Set

Then, it adds each HSM tree in the HSM Set where all the HSMs are saved . This is accessed when an object is decomposed and it is necessary to get the child nodes of the HSM.

Compute Signatures

In this step the signatures are computed. These are used in the iteration stage to verify if the object is decomposable or not.

Add Signature to Signature Pool

Each computed signature is saved in the signature pool, so when it is necessary to verify if the object is decomposable, it iterates the Signature pool and compares the signature of the segment that is being processed with the signatures of the segments that were already found, and whose signatures are already saved in the signature pool.

Add root of HSM to NDL

To finalize, it saves the root of the HSM on the NDL, so the iteration can start by segmenting the initial objects of the collection.

When it reaches the last element in the collection, the iteration ends and it passes to the iteration with these structures computed and loaded.

3.3.2 Iteration

The iteration stage is where the decomposition is actually performed and in this approach it corresponds to the iteration of the NDL itself. It starts on the first element of the NDL and procedes until it reaches the end of list. In the fluxogram represented in Figure 3.7 are the basic steps of this stage.

Is Decomposable? (step a)

It starts by verifying if the segment is decomposable. As this technique is based on the uniqueness of an object, the object has to be considered singular to be decomposable. This step is explained in more detail in section 3.4. If is decomposable, it decomposes the object by passing to the next step. If is not decomposable and has not reached the last element of the NDL, it just passes to the next element of the NDL.

Add each child node of HSM to NSL (step b)

In this step it accesses the respective HSM saved in the HSM Set and gets the child nodes which are then saved on the NSL.

Remove decomposed element of NDL (step b)

Because this approach does not need to maintain all the segments found during the decomposition, the object that is decomposed is eliminated from the NDL so it cannot be processes again.

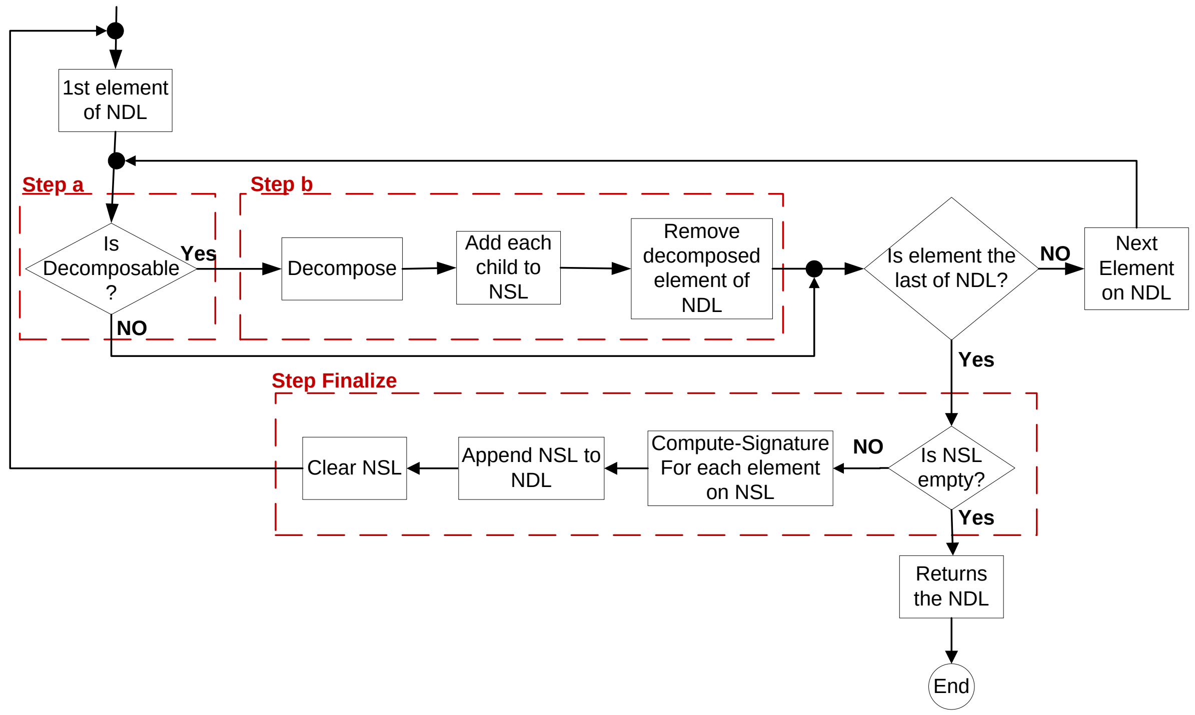


Figure 3.7: Iteration Fluxogram.

Is element the last of NDL?

Then it verifies if it has reached the end of the NDL, which is the end of the iteration. If it has not, it passes to the next element of the NDL, otherwise it passes to the next step.

Is NSL empty? (step finalize)

The NSL contains the segments to be processed on the next iteration, so it is necessary to verify if it is empty. If the NSL is empty the iteration stage ends. If not, it is going to finalize the iteration.

Compute signature for each element on NSL (step finalize)

When a segment is decomposed, the signatures of the child nodes are not computed since the approach is executed level by level of the HSM. The signatures of each segment found on each iteration are computed on this step, meaning that they are only computed before starting a new iteration, the iteration where these segments are going to be processed.

Append NSL to NDL (step finalize)

After computing all signatures of the NSL, it appends all elements of the NSL to the NDL, so these elements can be processed on the next iteration.

Clear NSL (step finalize)

In the end of the step finalize it is necessary to clean the NSL so the new segments found on the next iteration can be saved.

When the step finalize ends, a new iteration starts in the first element of the NDL. This process repeats until the NSL is empty which means that there are no more segments to decompose, so it ends by returning the NDL that contains all important segments that are going to form the segmented object.

Observing the pseudo code in section A and analyzing the execution of the approach, more specifically the iteration stage, it is possible to evaluate the algorithm complexity. Being N the number of elements on the NDL, in the best case the complexity is $f(N) = \Omega(N)$, because all the objects of the collection are not decomposable, so, it just iterates the NDL once. In the worst case it reaches the maximum deepness predefined, so the complexity is $O(2^{(depth-1)}.N)$. The complexity of the algorithm is $O(N)$ because $O(3N) = O(2N) = O(N)$, so it does not depend on the number of times that the collection it is going to be decomposed, but on the NDL size. These results are proved on the section 4.3.1 where a test to the time spent on the execution of the technique was tested.

The following schemes represented in the figures 3.8, 3.9 and 3.10 represent an example of the execution of the iteration stage for a better understanding of the proposed algorithm. Although the figures represent the execution of the iteration stage for just one object that is going to be segmented, in reality the iteration segments the entire collection by traversing each CaS tree level by level.

Starting at image 3.8, we can see that the process begins with the entire object. First it verifies if the segment is decomposable or not (Iteration 3.8 Step 1. a). It is decomposable, so it goes to the respective HSM to get the child nodes (left, and right), saves the child nodes on the NSL and deletes the decomposed segment of the NDL (figure 3.8, iteration 1, step b).

Because on this example it has only one element, by removing it from the NDL, the NDL becomes empty, so it has reached the last element of the NDL. The next step is to verify if the NSL is empty (Figure 3.8, iteration 1, step finalize), in this case is not, so the next step is to first compute the signatures of each element of NSL, adding them to the signature pool, then appends the segments of NSL to the end of the NDL, removes all elements of the NSL and restarts the iteration, passing to iteration 2. We can see in the image that only one node of the tree has been visited in the first iteration (figure 3.8, CaSTree on Iteration1). The CaSTree represents the traverse of the HSM, showing which nodes of the tree were visited during the execution of each iteration.

The new iteration starts similarity to the previous, by visiting the first element of the NDL (Figure 3.8 Iteration 2 Step 1. a) and verifying if it is decomposable. In the figure it considers that the object is not decomposable, so it passes to the next object on the NDL. But the second element is decomposable, so it decomposes it, adds the child nodes on the NSL and repeats the process previously described. In this second iteration, it has visited three nodes of the HSM (figure 3.8, CaSTree on Iteration2).

The figures show that the process repeats (figures 3.8, 3.9 and 3.10) until it reaches the last element of the NDL and confirms that the NSL is empty (figure 3.10 Iteration 4 Step Finalization a) meaning that there are no segments to append to the NDL, thus that are no more decomposable segments and that it has reached the final decomposition. In the end, all the important nodes for the decomposition are returned and the entire object is built with the segments resulting of the segmentation and saved in the Segmented Models Collection.

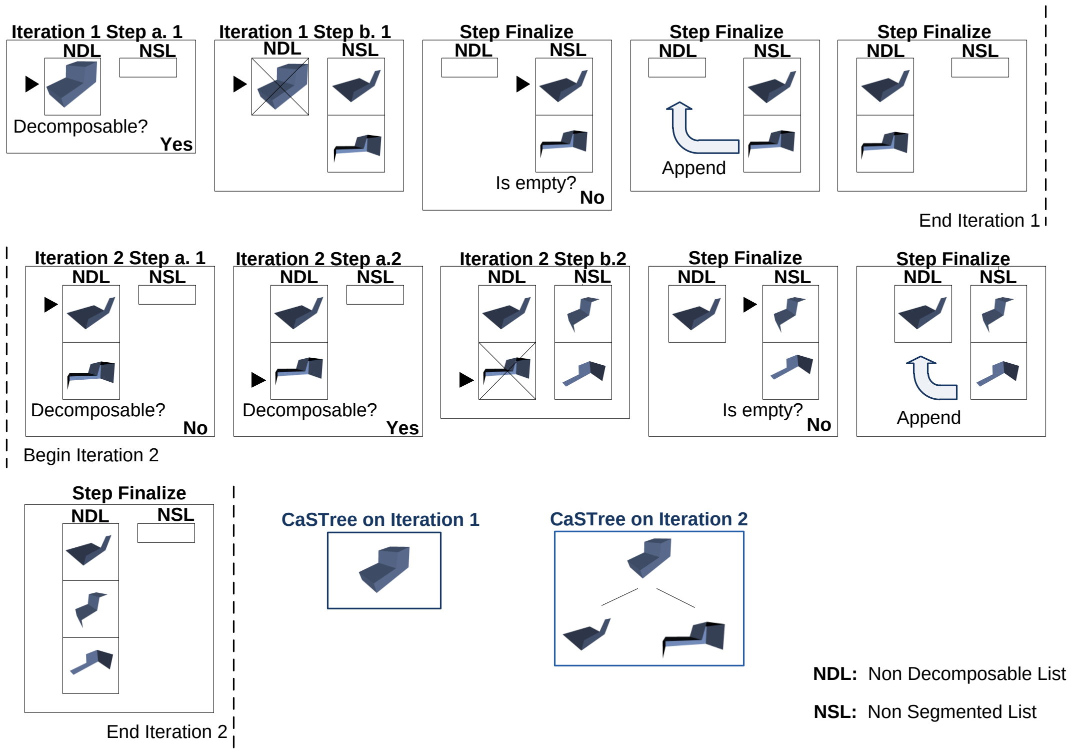


Figure 3.8: Example of an execution of the iteration stage, representing iteration 1 and 2.

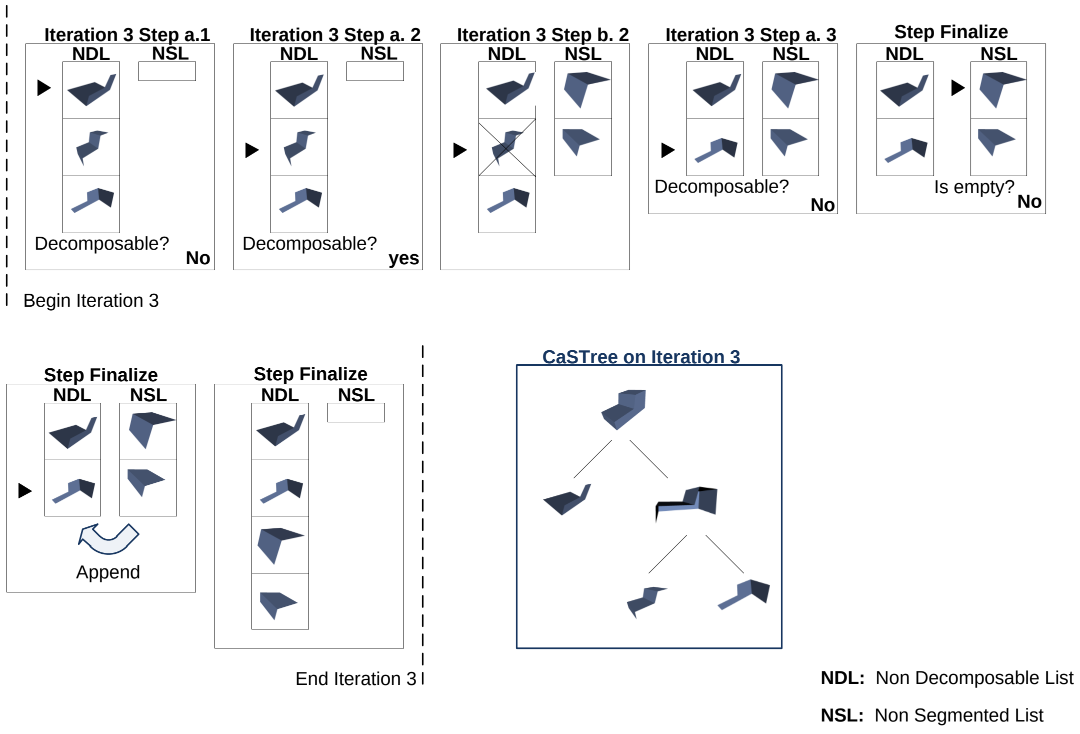


Figure 3.9: Example of an execution of the iteration stage, representing iteration 3.

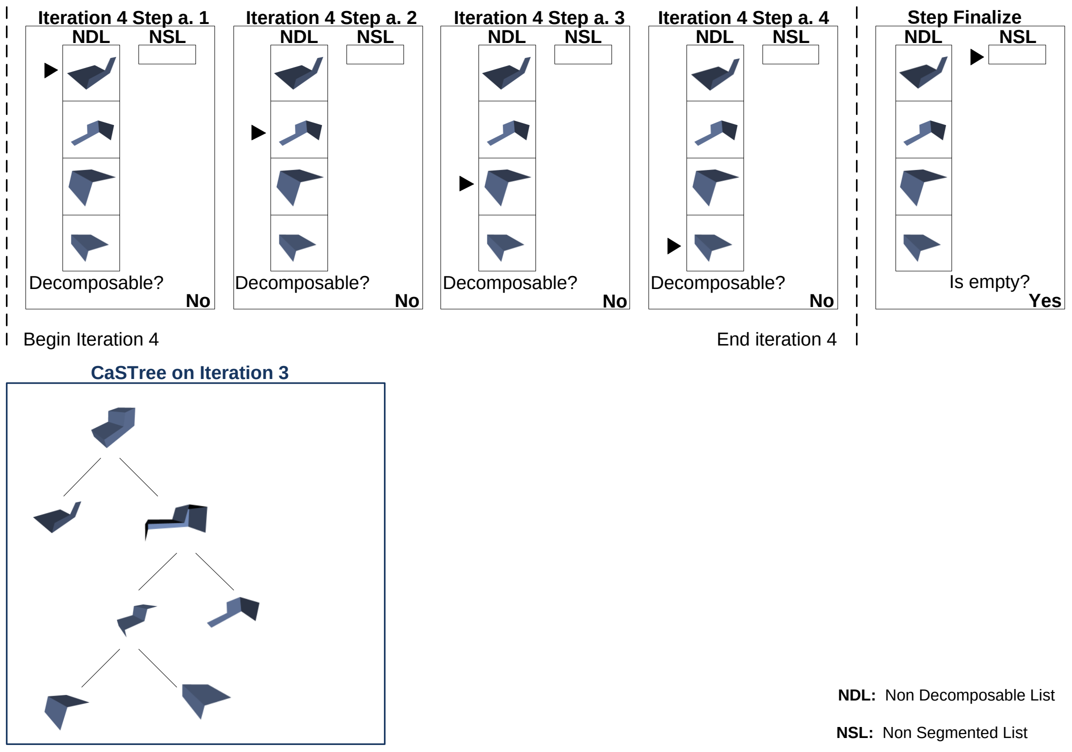


Figure 3.10: Example of an execution of the iteration stage, representing iteration 4.

3.4 *Adapted CaS/HFP* Decomposer

After understanding how each component and stage of the technique works, this section describes the interaction between them and how the entire approach works. The figure 3.11 represents the overall flux-ogram, showing both stages, the initialization stage where the inputs of the iteration stage are computed and saved, and the iteration stage where the segmentation is actually executed.

The *CaS/HFP* Decomposer starts by receiving the entire collection of objects on the initialization stage. On this stage for each object it computes the respective HSM using the *HFP* which generates a binary tree built bottom to top. It starts on the leafs which represent the triangles of the mesh, then for each pair of neighbors it calculates the merging cost. This cost is calculated by fitting a primitive (cylinder, cone, sphere and plane) to the resulting cluster, then it compares the values and clusters the pairs that have lower cost. This new cluster generated from the clustering is saved as a parent node of these clusters, in the tree. It repeats the process until it reaches the top of the tree where the entire object is saved. This tree will be traversed during the iteration stage and contains the segments that are going to be important for the final result. After computing these trees, the initialization stage saves them on the HSM Set.

Because the decomposition algorithm is based on the singularity of an object, the second step of the initialization stage is to compute the signatures of each object. These are shape descriptors, a numerical representation of the object in a multidimensional space. Using these representations makes the comparison between segments easier and this is used to label a segment as decomposable or not. When each signatures finishes being computed, it saves them on the signature pool.

The initialization stage ends by adding all the roots of the HSM on the NDL. This list contains all the segments that are going to be processed and all the segments that were labeled as not decomposable.

Observing figure 3.11, it is possible to visualize that after adding the HSM nodes to the NDL the initialization stage ends, so it passes to the next stage, the iteration stage. This will use all the structures previously built.

During the execution, the iteration stage traverses the HSMs, level by level, that is, each new iteration corresponds to a level on the hierarchy. In order to support the segmentation, the iteration stage uses the NDL. In practice, each iteration of the stage is an iteration to the NDL starting by visiting the first element of the NDL and verifying if is decomposable or not.

Like was previously referred, the *CaS/HFP* segments the objects based on the collection where they belong. The figure 3.12 represents the steps for labeling a segment as decomposable or not. It first verifies if has reached the minimal triangle count by calculating the minimal number of triangles between both child nodes and compares this minimal with a previously defined value, the minimum triangle count threshold, which in this case is three. If the minimal is below the minimum triangle count threshold the segment is decomposable, otherwise the segment is labeled as non decomposable. This step prevents from reaching the triangles of the mesh.

In the case of the result being no, it passes to the next step that is the execution of the K-Within Range (K-WR). This algorithm is used to verify if a segment is singular. In order to accomplish that verification, two thresholds were previously defined, the similarity and similar count thresholds. The first threshold is used to verify if two objects are similar or not. To be similar, the distance between two objects on the multidimensional space has to be less them the similar threshold, which is computed using the segments signatures and the difference between them. Then, the second threshold is used when comparing a segment with all the segments that are on the shape pool, as it cannot have more similar segments than the similar count threshold. With this result it is going to verify if the segment is singular. To be singular the number of similar segments has to be above the similar count, so if it is the case, then the segment is labeled as decomposable. If not, is labeled as not decomposable.

If the segment is decomposable, then it is going to decompose it. For that, it goes to the respective HSM and get its child nodes, then inserts both nodes in the NSL, which contains the segments that are going to be processed on the next iteration. Finally, it removes the processed element from the NDL passing to the next element on the NDL that also will be processed. In case of the segment being not decomposable it passes to the next element on the NDL.

An iteration ends when it reaches the end of the NDL, although the iteration stage may have ended or not. This depends on the NSL. If the NSL is not empty it means that there are new segments to be processed. So, it appends all the elements on the NDL, removes them from the NSL and then restarts a new iteration by vising the first element on the NDL again.

The iteration stage and the entire technique ends when it has reached the end of the NDL and the NSL is empty, meaning that there are no more segments to be decomposed. So, the technique ends by returning the entire collection of decomposed objects.

The execution of the *Manual Segmentation Test using HFP*, detailed on section 4.2, consisted on asking the users to segment a collection of ten objects randomly chosen and then interpret the results to understand how human segment the object. One of the complains of the users was that some objects were over segment. This happened because the *HFP* approach was used. In the same section the limitations found during the test are also detailed.

After executing the *Adapted CaS* approach with different similarity and similar count thresholds we noticed that this problem still happens on this technique, so, to overcome this limitation we introduced a new feature. The introduction of the Geons to verify if an object is decomposable or not, gives rise to a different technique, the *Geons-augmented CaS*. The next section explains this technique and the main differences between both.

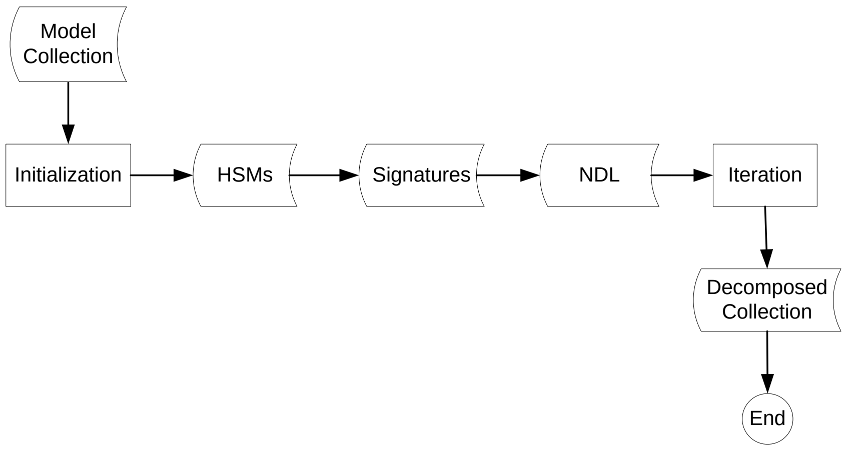


Figure 3.11: High level fluxogram of *Adapted CaS* approach.

3.5 *Geons-augmented CaS/HFP* Decomposer

The Geons are simple 3D objects, like cylinders, cones, cubes. The theory proposed by Bieldman on [8] called "Recognition by Components Theory" defends that just like English words are constituted by a number of phonetics. Complex objects can also be composed by these simpler 3D objects, that is, by segmenting these complex objects we get these simpler objects (Figure 3.13). Another property of the Geons is that using different arrangements of the same Geons can produce different objects. Geons are also view-invariance, so they do not depend on the point of view of the camera, which has been contradicted by recent studies where they defend that the 3D object recognition can be viewpoint dependent. Geons are also stable or resistive to visual noise. They can have other objects in front, an occlusion, and are still recognizable. The last feature of these objects is discriminability, meaning, that they can be distinguished from the others on different view points.

In Figure 3.14 the architecture of the *Geons-augmented CaS* is represented. The main difference between this approach and the *Adapted CaS* is the collection of Geons. This is used to verify if an object is decomposable or not. Because *Geons-augmented CaS* approach uses the *HFP* which, in turn, uses primitives to calculate the cost of merging clusters, and also because some of the Geons are the same as these primitives, then using these objects can help to stop the segmentation when it finds a Geon. Image 3.15 represents the decomposition fluxogram, which is similar to the figure 3.12 and description of section 3.4, being the main difference that after the first comparison, if the minimal triangle count of

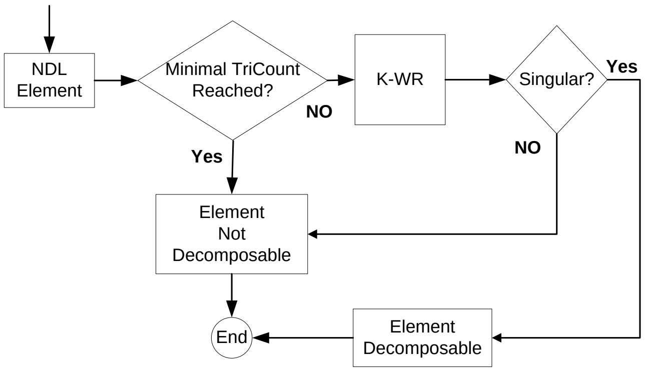


Figure 3.12: Fluxogram for segment decomposability identification of *Adapted CaS*.

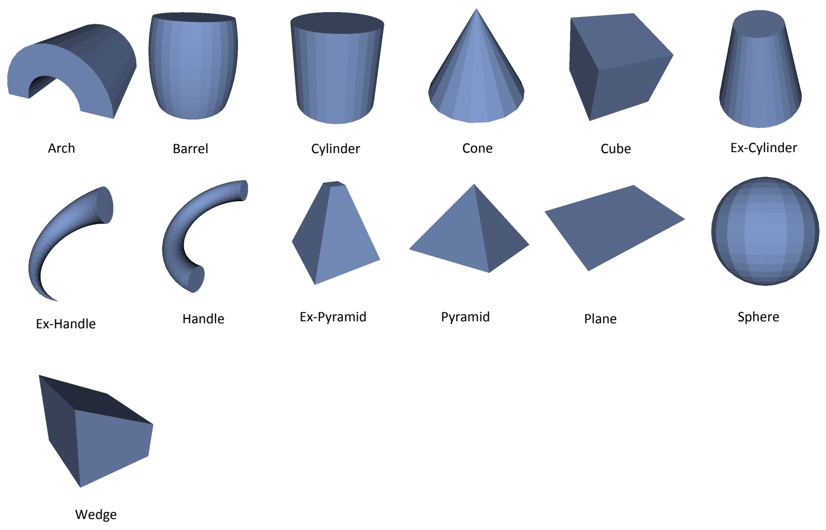


Figure 3.13: Example of group of Geons used on the approach.

the child nodes is above the minimum triangle count predefined, then it compares the segmented with the entire collection of Geons using the distance between both signatures. Then, if it is not similar, it proceeds as the previous description, that is, it executes the K-WR. Otherwise, the segment is labeled as not decomposable.

In order to compare the segments to the Geons it is necessary to predefine the best similar values between them. So, the best values for the similarity thresholds between the segments and the Geons were studied. To compare objects we use, as was already referred, the Spherical Harmonics (SHA), which one of its characteristics is the rotation invariance that is, two objects are detected as similar even if they are in a different position in space. But this are not scale invariant, that is, it takes into account the scale when comparing to objects, using the difference between signatures. So, it was necessary not only to discover the best similarity threshold for the usual Geons, but also to perform a profound study about the different scales and different smooth variations on the Geons shape to really overcome this problem of scale invariance.

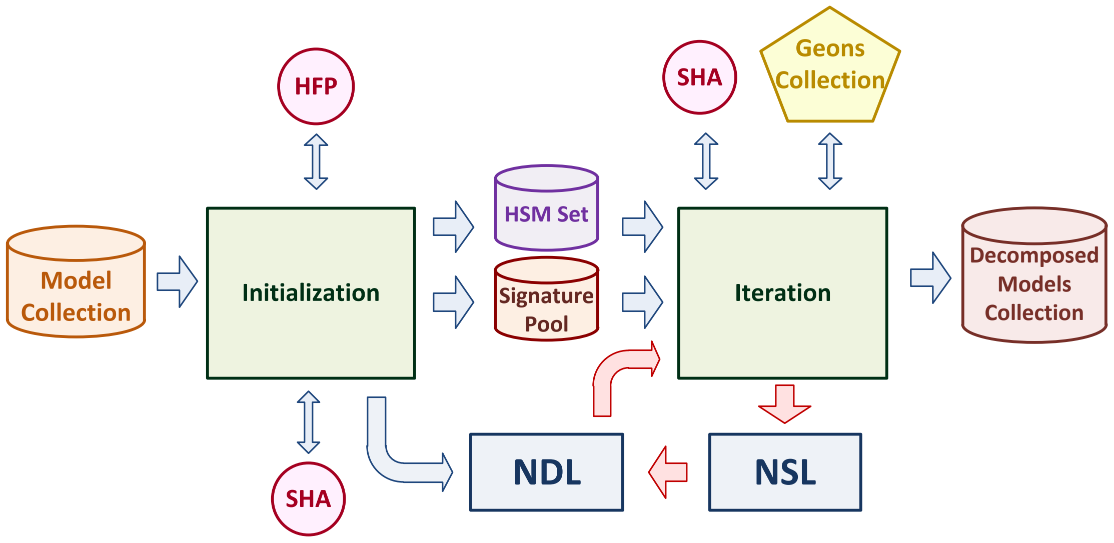


Figure 3.14: Overall architecture of the *Geons-augmented CaS*, with the two stages (Initialization and Iteration).

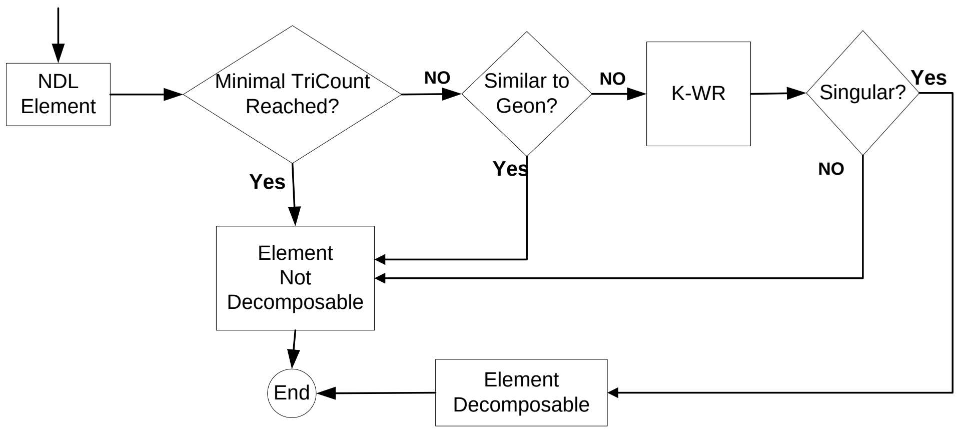


Figure 3.15: Decomposition fluxogram of *Geons-augmented CaS*.

On this stage of the thesis a study was performed on the cylinder. For that, several different types of cylinders were used. Figure 3.16 are represents the different variations of cylinders, some of which are closed both on the top as on the bottom, some have a hole from one side (top) to the other (bottom) but still has some thickness on the sides. And others are totally open, without a base or a top. Also the non scale invariance is shown on image 3.16, for instance, looking at cylinder 2 and cylinder 3 it is obvious that the shape is the same but the scale is different. As this study was only about the cylinders, it is necessary to perform a similar future study with the remaining Geons.

As one of the primitives used on the *HFP* is the plane, this primitive was added to the group of Geons. But still this is one of the main problems of using *HFP*. Using planes makes the results of segmentation as a plane instead of a volume as it should be, meaning that the Geons like the cube do not work on this approach. Thus, a good path to follow in the future is to use another approach instead of *HFP* and make a deeper study of these thresholds.

After performing a test to evaluate the similarity threshold between the objects and the Geons the final values of the similarity threshold that are predefined on the approach are between 0.10 and 0.35.

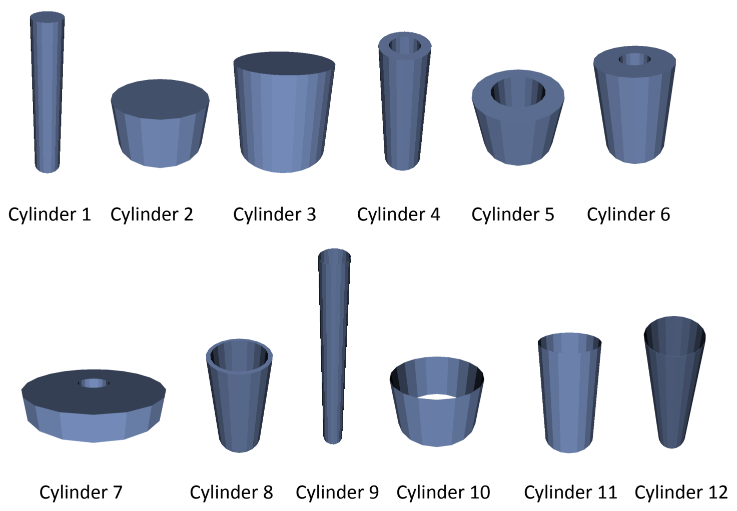


Figure 3.16: Set of cylinders used to stop from over segmenting,

Feature	*Original CaS*	*Adapted CaS*	*Geons-augmentd CaS*
Uses two hierarchical trees	X		
Uses one tree and two lists		X	X
Uses a Shape Pool	X		
Uses a Signature Pool		X	X
Uses Geons			X
Receives as input the thresholds	X		
Outputs segments	X		
Outputs the collection of decomposed objects			X

Table 3.1: Table with the main differences between *Original*, *Adapted* and *Geons-augmented CaS algorithms*

3.6 Summary

On this section we described two different segmentation approaches that were based on the original approach called Collection Aware Segmentation. These approaches were compared and their main differences were highlighted.

The first approach to be developed was the *Adapted CaS* approach that consists on segmenting a collection of objects automatically based on the geometric similarities of elements in the collection, producing meaningful results to the user.

After executing the *Manual Segmentation Test Using HFP*, some problems where detect, like the over segmentation of some objects. To overcome this limitation, a new characteristic was added to the technique, which was the use of Geons to label a segment as decomposable or not.

Based on the classification in Section this approach is classified as Part Based because it produces meaningful segments using a Surface Based approach, the *HFP*.

The main differences between the three approaches are shown in Table 3.1. The first difference is in the data structures used. The *Original CaS* used two hierarchical tress and a signature pool to save all the segments during the decomposition. However, both *Adapted* and *Geons-augmented CaS* approaches use only one tree, two lists, and a signature pool. With these changes the two approaches have become faster than the *Original CaS*.

The next depicted feature was Geons which is only used on the *Geons-agumented CaS* that was used to solve the over segmentation limitation of the *Adapted CaS*.

As one of the goals of the *Adapted CaS* was to have an automatic segmentation, the *Original CaS* is the only approach that needs to receive as input the similarity, similar count and depth thresholds. On the opposite, for *Adapted CaS* and *Geons-augmented CaS* approaches, these are predefined.

The *Original CaS* was integrated into a 3D object retrieval framework as a stage, so it returned the entire shape pool with all the segments discovered during the execution of the approach. The purpose of this thesis was also to prove that it was possible to isolate the *Original CaS* from the framework and produce different approaches that segment a collection of objects. Two approaches were implemented, the *Adapted CaS* and *Geons-augmented CaS* approaches, both of which return the entire collection of decomposed objects.

In the following chapter we discuss the evaluation of our work and the main experimental results.

Chapter 4

Experimental Evaluation

After completing the algorithm implementation, it was necessary to verify the efficiency and effectiveness of the algorithm, refine the approach and finally, test the segmentation results quality. Several tests were performed.

The *Manual Segmentation Test using HFP*, was used to understand the human interpretation of an object, more precisely, how humans segment 3D objects. With this information it was possible to refine the approach.

Performance tests were used to evaluate the approach efficacy and efficiency. Time and memory consuming of the algorithm were compared with the *Original CaS* approach values.

The *User Evaluation of CaS/HFP* test propose was to evaluate the quality of the approach and demonstrate that the *Geons-augmented CaS* results improved comparatively to the *Original CaS* approach.

All experiments were performed on a computer equipped with an Intel Core^{TM}2 Duo P8600 @ 2.40GHz, 4 GB of memory, and a Windows 7 operating system.

4.1 Prototypes

Tests execution requires different prototypes depending on the purpose of each test. Two different types of prototypes were used:

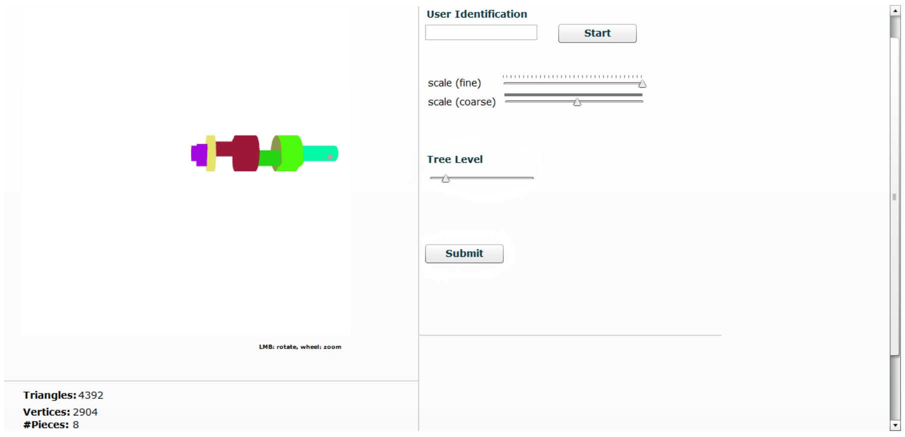
User tests were done using user interface based prototype, for understanding how human decompose 3D objects and to evaluate *Geons-augmented CaS* results quality and were used to interact with the users.

Performance tests where done using command-line based prototypes calling different *CaS* implementations to compare.

4.1.1 User Interfaces for evaluation

As some tests require user feedback, it was used two different tools from Shade WorkBench application. Shade WorkBench is a web application which allows an easy addition of new techniques for shape description, similarity computation, object segmentation, retrieval evaluation and best view computation. This application was developed by Tiago Lourenço in his master thesis entitled ”ShaDe Workbench: 3D Shape Descriptor Workbench” (which has no published articles yet).

The Manual segmentation prototype was used for users decompose each object of a collection while the Comparison prototype was used for users compare pairs of segmentation results choosing witch segmentation results produces the most meaningful segmentation



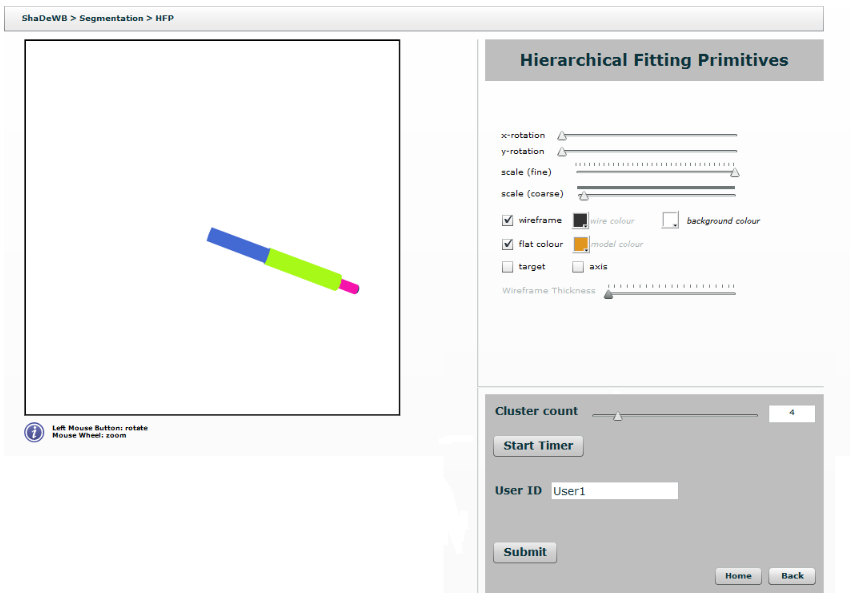
Triangles: 4392
Vertices: 2904
#Pieces: 8

Figure 4.1: Shade Workbench manual segmentation tool interface with a segmented object with a level choice.

Manual Segmentation

In order to perform the *Manual Segmentation Test Using HFP* was developed a prototype (Figure 4.1) where it was possible for users manually segment an object. The users by changing the tree level, chooses which the results represents a segmentation more meaningful to him. At the end, the application saves a XML document with the tree level chosen, the respective number of segments and the time it took for segment each object.

For the *CaS Evaluation With Users Test* it was necessary to repeat the *Manual Segmentation Test Using HFP* but using a different collection of objects. The manual tool was again used but with a variation (Figure 4.2). Instead of changing the tree levels, users changed the number of segments produced by the results. The objective of this variation was to compare the *Geons-augmented CaS* results with manual segmentation and *Original CaS* results.



Cluster count 4
User ID User1

Figure 4.2: Shade Workbench manual segmentation tool interface with a segmented object with four clusters.

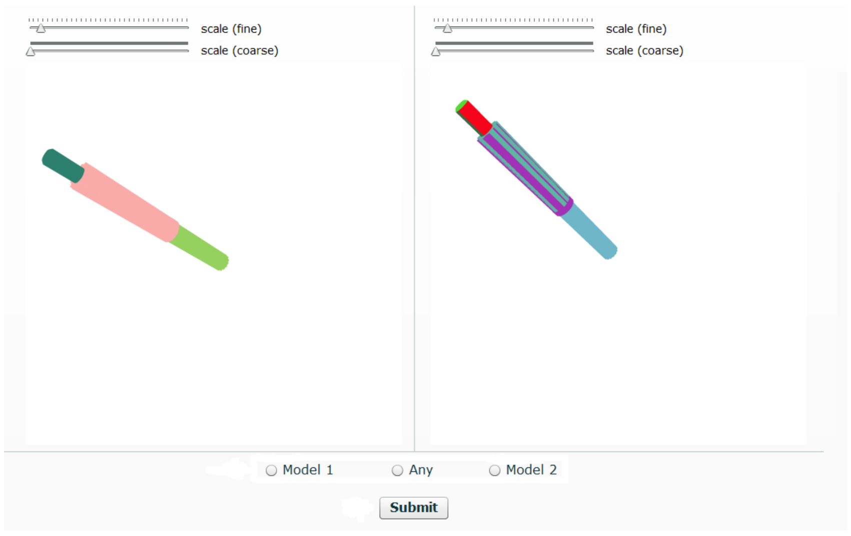


Figure 4.3: ShadeWorkbench comparison tool interface with two different segmentation results.

Comparison

This prototype (Figure 4.3) was used to demonstrate the quality of the system at the level of the human interpretation. Users were asked to compare pairs of segmentation results and choose which one is more meaningful or both.

4.1.2 *CaS* Prototypes

After defining each *CaS* architectures, each correspondent prototypes were implemented using the c++ language, producing a command-line based application.

It receives an input on the command line and the output is saved on a specific folder. These prototypes were used to prove the approach efficiency by using the time, memory used values and also by comparing all *CaS* approaches (the *Original*, the *Adapted* and the *Geon-augmented CaS*).

Original CaS

The original approach produced a command-line based prototype, and was used to compare its results with the results of the *Adapted* and *Geons-augmented CaS* approaches.

To execute the time and memory test, the *Original CaS* prototype needs important information. Special key words are used to assign the inputs to the information. Like the important thresholds, depth, similar count and similarity. The directories, where the collection of objects to be decomposed is, and where and name of the XML document where the printable results are saved. Also the extension of the objects (STL or OFF files) that belong to the collection and the format of the segments resulting from decomposition (VRML or DAE).

The decomposition results, that is, the elements of the shape pool are saved on the same directory as the executable.

These input's are not always the same, for more detail, it can be used the help command to explore all the options of the application.

Adapted CaS

The *Adapted CaS* approach receives a set of parameters and returns the collection of segmented objects.

In the *Adapted CaS* prototype the decomposed collection are saved on the directory given by the user as input. It was also added a parameter in case of having to save the segments resulting from the several levels of decomposition.

This prototype also receives the same type of objects and save the decomposed objects as the previously described prototype.

Geons-augmented CaS

This technique is an extension of the *Adapted CaS*, where the comparison between segments and Geons was added. So, it proceeds exactly as the previous. The main difference between the *Adapted CaS* prototype and the *Geons-Augmented* is the produced results.

After having performed all the tests, the final prototype does not need to receive the thresholds, as the study was to define these values.

This prototype also receives the same type of objects and save the decomposed objects as the previously described prototype

4.2 *Manual Segmentation Test Using HFP*

This test was performed in order to understand the human perception of an object, specifically, how does a person segment a given 3D object. The two-stage test began with a questionnaire. It had the purpose of identifying the characteristics and prior related experience of the test users. Following the questionnaire, the users were instructed on how the test would proceed and given the opportunity to try the tool. On the second stage of the test, the users were asked to segment an object using the Shade WB segmentation tool that uses the *HFP*.

4.2.1 Sample group

The sample group consisted of twenty users. Each one was identified by a number and was asked to fill in a questionnaire that gathered the following data about the user: age, gender, school degree, computer experience and, finally, 3D modeling experience.

With this information it was possible to conclude that the sample consisted of users with age between twenty one and thirty four years old. Fifteen were male and five were female. Eighteen elements of the group have an academic degree, with the remaining two having the twelfth grade. The sample is also constituted by a group of experienced computer users, with nine having some 3D modeling experience, unlike the rest of the elements.

4.2.2 Test Set

The *CaS* technique was tested using the Engineering Shape Benchmark. This ESB is a collection of three-dimensional objects of engineering. Most of these objects are not very familiar to the users, so they allow an instant interpretation, instead of biasing the results with a preconceived interpretation that the users might have in mind. For this test, each user was asked to segment the same ten objects from this collection (4.4). These were randomly chosen from the ESB. This collection is detailed on the appendix (B).

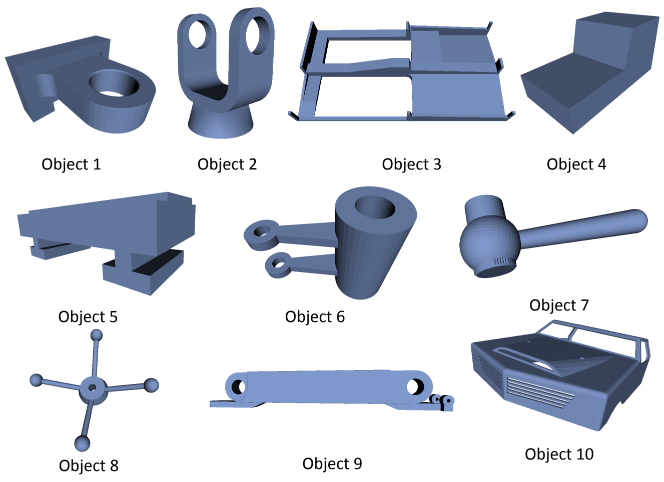


Figure 4.4: Collection of objects used in the test

4.2.3 Description

As the main goal of this test was to understand the human perception, the user was asked to manually segment a collection of objects. For that purpose, the prototype described in section was provided to the user..

The *HFP* is an approach that uses hierarchical clustering, that is, it generates a hierarchical binary tree by clustering the triangle meshes. It starts on the bottom of the tree, where each leaf represents a triangle of the mesh and then it clusters the triangles that have lower cost to merge, building the tree from bottom to top. The cost is calculated by fitting the resultant cluster to a primitive (plane, cylinder and cone).

As this algorithm produces a hierarchical tree, it is necessary for the user to choose which level produces a more meaningful result. So, the only task the users had was to select different levels of the tree using the scroll, observe the different results and then choose the result which had, in their opinion, the most meaningful segmentation.

Before the actual test, each user was given a test object, so that he could try the application and clearly be familiarized with the way it worked. The user had five minutes to explore the segmentation tool by choosing different levels, increasing and decreasing the object size and rotating the object.

Then, the user was shown each element from a collection of ten objects (image 4.4). The objects appeared in a random order, for each user. After the user having observed each result and chosen the one that produced a more meaningful segmentation, the choice was saved for analysis. The collected data consisted of the level of the hierarchy and respective number of segments, as well as the time it took to segment each object, later used to understand the complexity of each object and the difficulty felt while choosing a level. The test protocol can be found in the appendix E.1.

4.2.4 Results

With all the test sessions successfully completed, it was necessary to analyze the results. The data analyzed in this section pertains to the number of segments of the chosen segmentations and to the time users spent to select each result.

Number of Segments

This section analyzes the results, comparing the different choices and returning the number of segments of the most chosen segmentation of each object. In order to understand and analyze the conclusions it

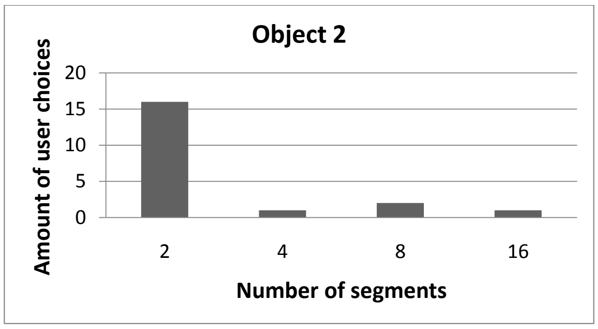


Figure 4.5: Results of user choices for object2

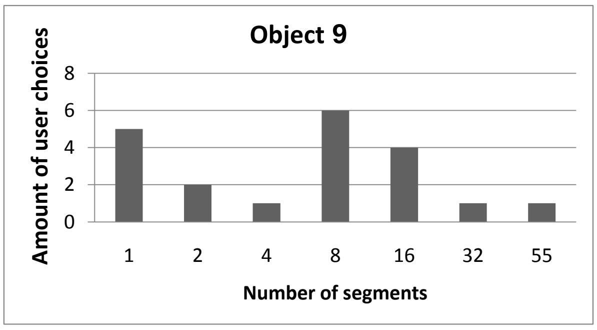


Figure 4.6: Results of user choices for object9

was necessary to complete the study using the tables and conclusions that are on the appendix C.

As in the test the users chose tree levels instead of the number of segments, and also because the tool uses a binary hierarchical tree, the number of segments varies even if the tree level is the same for different objects.

The Table C.2 of the Appendix C.2 was used to identify the number of segments for each of the most chosen segmentations. This showed not to be enough and it was necessary to understand not only which number of segments had more choices, but also how many choices it got and if it is the only choice. This kind of analysis was supported by bar charts that represent the amount of users choices per object (at Appendix C.3).

The sample group clearly agreed on some objects. That is, the number of segments that each user chose is really close to the choices of most of the other users. A situation like the one represented in image 4.5 of object 2 where most of the users chose the result with two segments, because these are simple objects and the *HFP* generated simple results that were satisfactory to the user.

Other objects did not appear to have a segmentation that pleased the users. Object 8 is an example. In spite of being a simple object, its results of the segmentation did not match the user expectations. As we can see in image 4.4, object8 is constituted by cylinders and spheres, so the user expected to have a segmentation in which each of these constituents belonged to a different segment and also the piece at the center. This did not happen because the algorithm uses a hierarchical binary tree so, when it finds a cylinder and a sphere, the rest is still attached to the body center and, when fiddling a second pair, the previous one is again decomposed, causing an over segmentation (Figure 4.7). Still, users prefer to either have over segmentation but with all constituents separated or have the entire object as one segment.

For object 9 it is not so easy to come to a conclusion, the result, as it can be seen in image 4.6, is very ambiguous, because it has seven different numbers of segments in which the users are divided between having just one or two segments and eight or sixteen, representing a wide range of values. This seems to have happened because it was hard for the user to understand the object and also because the segmentation results were not the expected, so some found that the better choice is the entire, not segmented, object.

Time

The table C.3 of appendix C.2 represents the time it took each user to segment each object of the collection. The two last columns of the table represent the total amount of time it took to each user segment an object and the average time it took the user to segment each object of the collection. The last line has the average of the time it took to segment each object. With this information we can conclude that, in average, each user took 10:36 minutes to segment the entire collection.

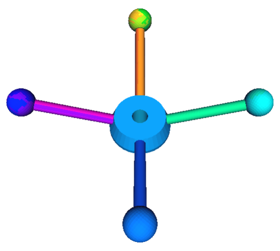


Figure 4.7: Representation of object 8 with a segmentation result of a choice of level 6

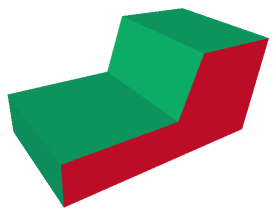


Figure 4.8: Representation of object 4 with a segmentation result of a choice of level 4

As it is shown in table C.3,the object that took more time was the object 9, with an average of one minute and thirty four seconds. This means that it can be considered a difficult object to segment and the results presented were not the expected. This can also be observed in the previous section (4.2.4) where there are many different choices for the number of segments. On the contrary, the object that took less time to segment was the object 7 meaning that the users chose the segmentation that they consider more meaningful faster. This happened partially because is an object with a familiar shape, similar to a hammer.

4.2.5 *HFP* limitations

With this test it was possible to understand some limitations of the used algorithm. Because the *HFP* generates a hierarchical tree, the first limitation that occurs is that some segments are over-segmented while others are not. For instance, when it is traversing the object 8, HSM first founds a stick with a ball and the rest of the object is another segment. When we choose a level like 6, the stick and spheres initially discovered are over segmented but one of the sticks and one of the spheres still belong to the same segment (Figure 4.7), causing some conflicts for the user.

Another limitation is the fact that the segmentation uses planes to fit, so instead of segmenting by volume, it segments by face. A good example is object 4 because it is very rectangular and makes each face belong to a different segment (image 4.8).

4.2.6 Manual Segmentation VS *Adapted CaS/HFP*

As we are aiming for an automatic and meaningful segmentation, it is necessary to define the similarity and the similar count thresholds. These variables of the *Original CaS* approach needed to be inserted by the user. On the *Adapted CaS* these are predefined so the same values are used for each execution of this approach. After the execution of Manual Segmentation Test Using *HFP* and the analysis of its results (section 4.2.4), the same collection used in the manual segmentation was segmented using the approach with different threshold arrangements. Then it was possible to compare the user results with the *Adapted CaS* results, in order to choose the test which produces the most meaningful results.

These two thresholds are used to label a segment as decomposable or not decomposable. A decomposable object has to be considered singular, that is, it cannot have more than a given number of objects (similar count threshold) similar to him (similarity threshold). To verify if two objects are similar we use the difference between them. If that calculus is above the similarity threshold, it means that the objects are similar.

To perform the comparison between the results of the user and both *CaS* approaches, the *CaS* prototypes were executed using different threshold values. Several other indicators were used for the comparison: the difference between the numbers of clusters chosen by the user in the *Manual Segmentation Test Using HFP*, the number produced by the *CaS* approaches and the percentages of human choices for each number of segments. But, after trying several thresholds, it was obvious that just using this information was not enough because, sometimes, for the same number of segments the result is very different. So, it was necessary to introduce a classification to understand which values produced the most meaningful results. This was created by observing and comparing the results.

The results were classified from worst (*) to better (****). The worst (*) results are very different from the ones chosen by the user. Other results were not so easy to classify, because some have the same number of clusters but do not actually have the same clusters. Others have a different number of clusters but are similar, while others, having a different number of clusters, still represent a meaningful segmentation. To differentiate this objects three more classification levels were created. The medium (**) classification is used when the number of cluster is near but the segmentation is worst and that are wrong, that is that are not meaningful for the user. For an object to be classified as good/similar (***), it needs to have the same number of segments and the same result. The better (****) classification exists because it was necessary to distinguish the objects where the result produced by the *CaS* approach was better than the results produced by the user. This happened because the users felt and stated that it would be more meaningful to segment the object in some other way.

The best course of action would be to execute another test, where a group of users should classify the segmentation using this scale and identify which value produces the best segmentation. But because this is a master degree thesis, the time is very limited. Also, a test of that kind requires lots of effort from the users so, it was not possible to execute it. We expect that these values will be validated in a future works.

With these results was possible to conclude that, even changing the thresholds, some objects still have the problem of over segmentation. To to solve this problem, a different technique was proposed, using Geons to label the segments. The same comparison was performed, using the new results and its conclusions are described in the next section.

4.2.7 Manual Segmentation VS *Geon-augmented CaS/HFP*

After the previous evaluation, one of the main problems found was that some of the results still over segmented, using the *HFP*, even with different threshold values. So, to overcome this limitation, a new feature was added to the approach. It consists of using Geons to verify if an object is decomposable or not. Geons is a psychology theory of object recognition proposed by Biederman's in [8]. There are simple 3-dimensional forms such as spheres, cubes, cylinders, cones or wedges. This theory consists of the human interpretation of objects. While interpreting an object, a human being can detect and segment it using those simple forms. This means that the objects are constituted by these simple and familiar forms.

A limitation was found when the Geons are parallelepiped, because the *HFP* uses planes to fit, producing a plane segmentation instead of a volumetric segmentation. So a future work would be to use another hierarchical segmentation algorithm and do a refinement to the similarity threshold values between the segments and the Geons.

To understand which values to use to consider a segment similar to a geon, segments similar to the geon were compared. It was detected which difference values where enough to consider them similar. These values vary from 0.10 to 0.35, depending on the Geon.

The evaluation process performed on this section is the same as the last one, that is, the number of clusters of the *Geons-augmented CaS/HFP* was compared with the results of the manual segmentation, using the same evaluation criteria. After analyzing and comparing all the results, the test that presented the better results was test 5, with 0.58 of similarity and 1 of similarity count. This test has the higher percentage of user choices, 5 objects with the same number of segments as chosen by the users and also 2 objects with the best classification. It is not the test with the biggest number of best classification, but it has 5 good/similar objects, making it, overall, the best of the tests.

4.3 *Performance Tests*

These tests will evaluate the performance of the algorithm, that is, the time it takes to segment a collection of objects and how much memory it consumes.

4.3.1 Execution Time

As this approach is destined for the end-user and as one of the main goals was to improve the time taken to decompose a collection of objects, it is necessary to test the time spent on decomposing the collection.

The approach proposed in this document has some time consuming tasks, like the object signatures calculations and the distances between them to define the singularity of each segment. That said it was necessary to calculate the time it takes to execute each task. For that, we used the *Geons-augmented CaS* prototype (Section 4.1.2) that executes the *Geons-augmented CaS* algorithm and saves the time results to a XML document.

As shown in Figure 4.9 the time is very close to the linear line. It does not superpose the linear line because of external factors that have influenced some periods of time, such has internet related connections. With these results it was also possible to prove the predicted complexity of the algorithm, as estimated in section 3.3.2.

In order to prove the *Geons-augmented CaS* approach improvement relatively to the *Original CaS*, a comparison between the time spent by both approaches was performed, executing the *Original CaS* prototype (4.1.2).

This consisted of executing five tests for each approach and then comparing the results. From the first test to the fifth the collections used were constituted by 60 objects, 210, 410, 610 and 853 respectively. These used the same objects as the Manual Segmentation, extracted from the ESB.

The first observation is that both algorithms have a linear growing. But comparing them in more detail (Figure 4.10) reveals that the new algorithm has lower time spent and this difference increases along with the growing of the collection used on the tests, as shown in figure. With the time values, it was possible to calculate the percentage of the difference between the times of both approaches. The first test, with 60 objects, has a difference of approximately 20%. The third test has approximately 28% and finally the last, with 853 objects, has approximately 30%, meaning that, when the collections of objects increase, the difference between both approaches also increases, proving the improvement of the approaches in terms of execution time, which was one of the main goals.

To compare the execution time in more detail, in each step of the approach the the figures 4.11 and 4.12 are used. With this comparison it is easier to understand where the time was spent in each approach. On both of them, most of the time was spent computing the descriptors and decomposing. This happens because the computation of the descriptors, the SHA signatures, is, as predicted, a time consuming step. The same happens with the decomposition, because it embodies the signature computation, where the differences between the signatures are calculated and the segments are labeled and then decomposed. In the *Original CaS*, these steps take more time than in the Geon-augmented, because of the improvements. The use of two lists instead of a tree, as well as the Geons, decreases the number of segments, so there are fewer signatures to compute and compare and also less segments to decompose.

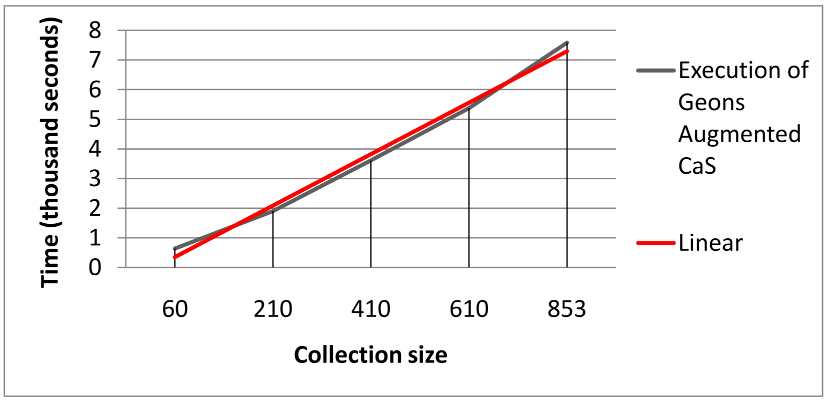


Figure 4.9: Time line of the *Geons-augmented CaS* approach execution of the various tests

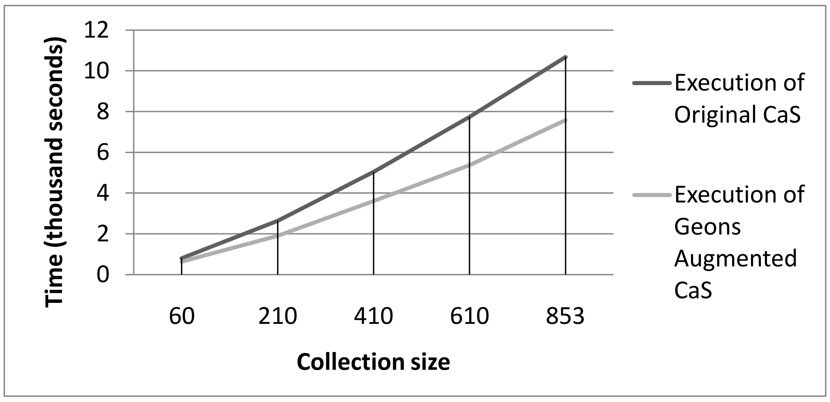


Figure 4.10: Graphic with the results of both approaches time test

As shown in figures 4.11 and 4.12, the time spent computing the similarities is bigger in the *Geons-augmented CaS* than in the *Original CaS*, because it also compares the Geons with each segment, so it can be decomposable or not.

After this time analysis and comparison it is possible to conclude that this new approach is an improvement. This is important, because the users do not want to wait a long time. In spite of that, as we are dealing with a large number of objects, it still takes a long time to decompose the entire collection, but if the user submits only one object of the collection, the execution is faster.

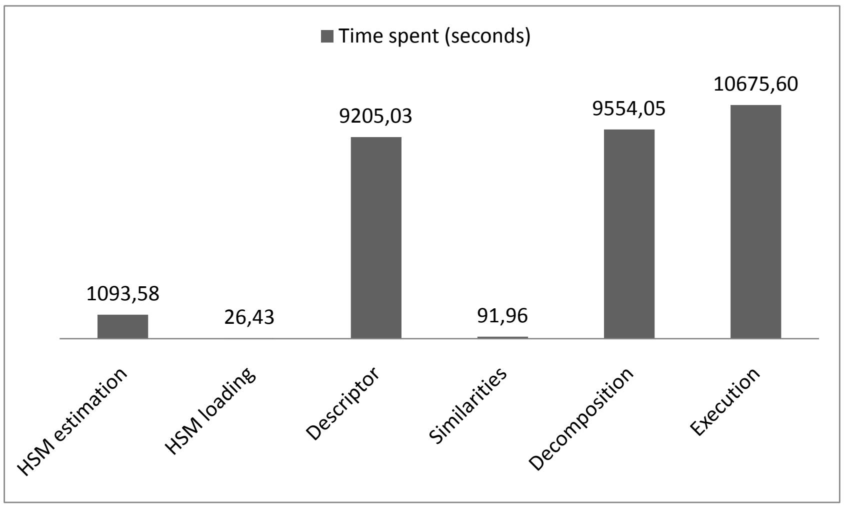


Figure 4.11: Graphic with the results of the *Original CaS* approach time tests of the relevant steps

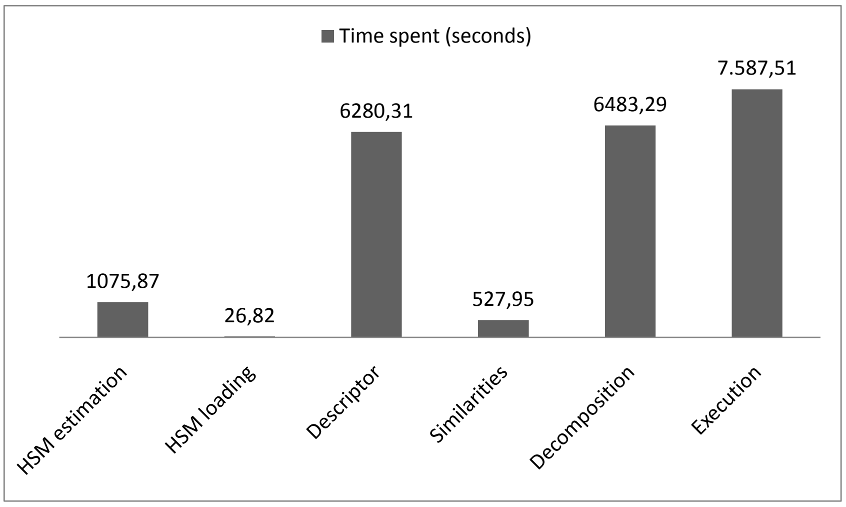


Figure 4.12: Graphic with the results of the *Geons-augmented CaS* approach time tests of the relevant steps

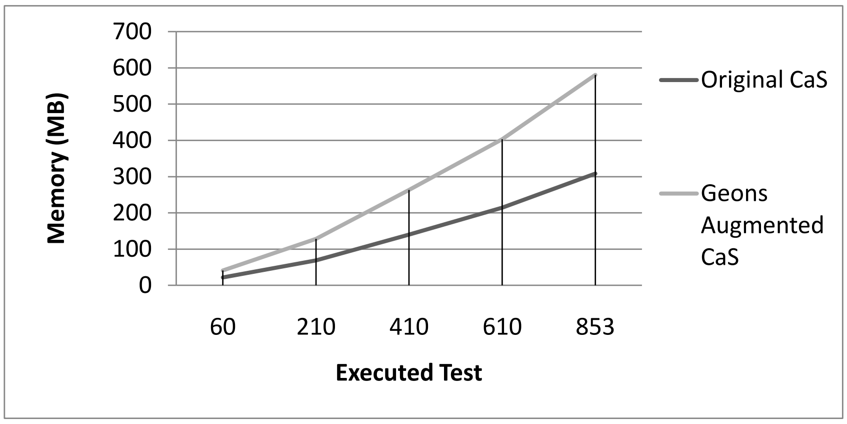


Figure 4.13: Graphic with the results of both approaches memory tests

4.3.2 Memory requirements

Another factor that was taken into account is the fact that the algorithm applied to a collection of objects. The collection used to test the approach is composed by objects of the Engineering Shape Benchmark (ESB), proposed by the PRECISE group. This benchmark is composed of 3D engineering objects. So, it was necessary to study the memory requirements for a good execution of the algorithm. For the execution of this test, both the original and the *Geons-augmented CaS* prototypes were used.

Comparing the results of both techniques shown on figure 4.13, the *Geon-augmented CaS* approach uses more memory than the *Original CaS*. This happens because more lists are used, besides the Non Decomposed List (NDL) and the Non Segmented List (NSL), another list is also used, where the signatures of the Geons are saved, so each segment can be compared to them.

The memory is nowadays a cheap resource which has increased over the years. So, having spending more memory to get a faster technique, with better results to the users is a good trade-off between these two important measures.

4.4 *CaS Evaluation With Users Test*

After having refined the application it was necessary to evaluate the segmentation quality, that is, if the technique produces meaningful segments to the user and better results than the other approaches, validating our approach according to the human perception.

The test to evaluate the quality of the technique consisted of mainly two stages. First, the same test previously described in section 4.2 was executed, with a different collection of ten objects. The second stage consisted of asking the user to compare the result of the *Original CaS* with the *Geons-augmented CaS* and his previous choices with the results of the *Geons-augmented CaS*, selecting which one he preferred using the 4.1.1 prototype.

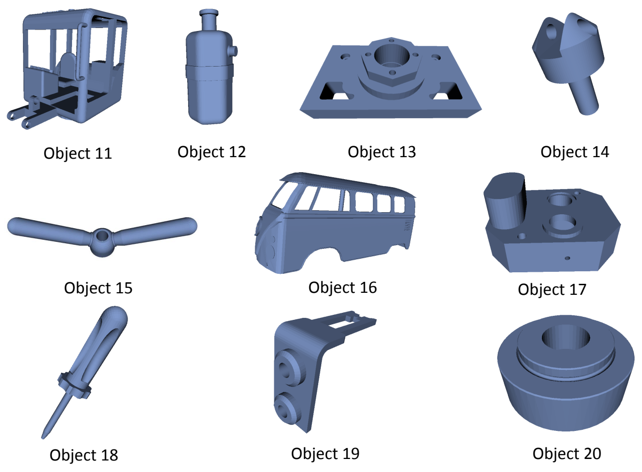


Figure 4.14: Collection of objects used in the test

4.4.1 Test Set

Like the test presented in Section 4.2, a collection of ten different objects belonging to the Engineering Shape Benchmark was used. These objects were also extracted from the ESB and were randomly chosen. The Figure 4.14 represents the entire collection that was used to execute this test. This collection is more detailed in appendix B.

4.4.2 Sample group

This test was performed with a sample group of twenty users, different from the one that participated in the manual segmentation test 4.2. In order to understand the users characteristics, each one was asked to answer a five question questionnaire, which inquired about the age, gender, school degree, computer experience and 3D modeling experience.

Based on this questionnaire we observed that the sample group consisted of users with ages between twenty one and thirty. Four are female and sixteen are male. The majority have an academic degree: just one only has the 12th grade. All the users are computer experts, with six having no experience with 3D modeling, while the remaining fourteen stated to have some experience working with these tools.

4.4.3 Description

As the segmentation is part based, that is, the produced results have to be meaningful to the human being, this test consisted of evaluating the quality of the results produced by the approach. For that, the users were asked to compare the segmentation results of *CaS* with the ones previously produced by them.

In order to evaluate this approach with the accuracy, it was necessary to request the users to segment a new collection of ten objects. In an attempt not to bias the results, the test of Section 4.2 was performed with a new collection of objects The Shade WB tool was also used, but with a slight change: instead of segmenting the objects by level, the users could segment by the number of segments. After executing the *Manual Segmentation Test Using HFP*, we observed that the users required the possibility of adding just new segments one by one, because that would be easier to compare with the results of the *CaS* algorithm because it produces a different number of segments, not only depending on the tree levels.

After segmenting the objects, the users had to compare the results between the two *CaS* approaches, the *Original* and the *Geons-augmented*. To perform this test a different Shade WB tool was used: the comparison tool. The user was presented the same object but with different segmentation results that were generated using the original and *Geons-agumented CaS* prototypes (Section 4.1.2). The user had to compare both results and then choose the most meaningful to him. In some situations, both results were equal or very similar and meaningful for the user, so there was an option to choose both alternatives.

Aiming to prove the quality of the segmentation results, a similar test was performed but using the *Geons-augmented CaS* results, which were compared with the ones chosen by the user on the first stage of the test. With these results it was possible to prove that the goal of having a meaningful segmentation for the user was achieved. The protocol of the test is available in Appendix E.1, where the test steps are detailed.

4.4.4 Results

Because each human being has individual features, evaluating the quality of the segmentation proved to be a hard task to execute. It is not possible to please all the users with the results produced by the approach. However we try to produce the most meaningful results to the user and, with that, prove the quality of the solution. So, we will focus on the second stage results, the user results comparison. This chapter starts with a brief analysis of the first stage, in particular regarding the chosen number of segments.

Number of Segments

In appendix E.2 is a table with the final results of the user choices, as well as a summary, containing the number of segments that had more choices for each object.

The segmentation results of some objects generated consensus, with the users choosing the same number of segments almost all of the times. Others generated very different numbers of segments. This depended on the complexity of the object and also on the results of the *HFP* segmentation. These are not the most important conclusions to take into account on this test, because the results of the manual segmentation were only used for the next section that has the purpose of comparing them with the ones produced by the Geons-augmented approach.

Original CaS **versus** ***Geons-augmented CaS***

As improving the *CaS* approach was one of the goals, it was necessary to actually verify the improvements by comparing both of the produced results. So, the user was asked to compare both results and choose which one is more meaningful to him. The results of both approaches were produced using the same threshold values. They are: the similar count with 1, the similarity with 0.58 and the depth equal to 20. These values were found in Section 4.2.7 on manual segmentation versus *Geons-augmented CaS/HFP*, where the results of the first segmentation made by users were compared with the results produced by the *Geons-augmented CaS*. A classification was assigned to each of the segmentation results and the percentage of choices was used, so it could be possible to identify the threshold values that produce the best results.

Because the *Geons-augmented CaS* is derived from the original algorithm, some segmentation results are the same. So, as shown in Figure 4.15 the any option was chosen one hundred and one times. The remaining objects proved to have better segmentation results using the *Geons-augmented CaS*, as the *Geons-augmended CaS* approach has ninety seven choices. The only exception is the object twelve, which caused some disagreement between the testers. There is not much difference between the two approaches: just one segment. The *Original CaS* has four, while the *Geons-augmented* has three. Counting the results, three users preferred the *Original CaS* result.

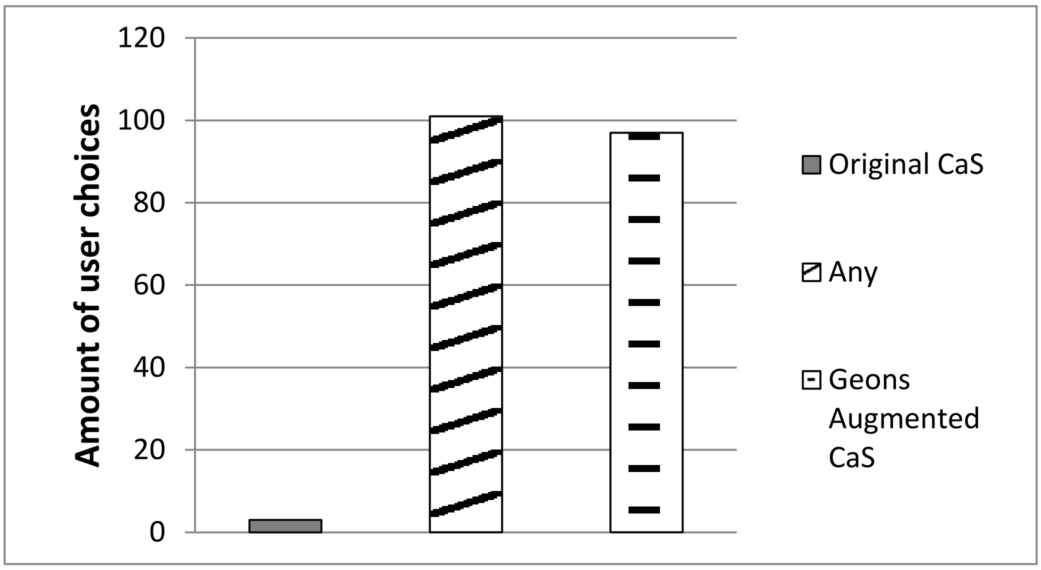


Figure 4.15: Results of comparing *Original CaS* with *Geons-augmented CaS*

As shown in Figure 4.16 the segmentation result of object 18 produced by the *Original CaS* approach has three more segments than the results shown in Figure 4.17, which represents the segmentation result using the *Geons-augmented CaS* approach. These three more segments produce an over-segmentation for the user, so this is one of the objects in which all users have chosen the *Geons-augmented CaS* approach.

With this information it is proven that the *Geons-augmented CaS* approach, according to the users, produces better results than the *Original CaS* approach.

User Segmentation versus *Geons-augmented CaS*

The aim of this test is to prove the segmentation quality, given the fact that it focuses on the human being interpretation. But as has been proved over time, the human being as an individual produces many different interpretations of the world and in consequence, of the objects themselves. The evaluation of the results proved to be a challenge but it an effort was made to found the best way to analyze and prove the quality of the results.

As described in section 4.4.3, the users were provided a tool to compare the results of the *CaS* approach with their own. With this comparison it was possible to understand if the results produced by the approach were worst, the same, or better than the ones chosen by the user.

By analyzing the results on Figure 4.18 it is possible to conclude that, for three objects, it is clear that the users preferred their results instead of the *CaS* results. This happens on the more complex objects where the result produced by *CaS* can be considered under segmented compared to the users choices. Two objects of the collection have really a close number of choices between Geons-augmented *CaS* and

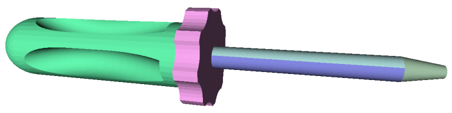


Figure 4.16: Object 18 with a segmentation result of the execution of the *Original CaS* approach.

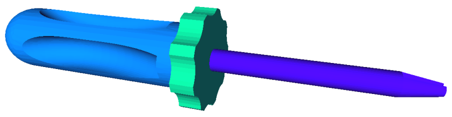


Figure 4.17: Object 18 with a segmentation result of the execution of the *Geons-augmented CaS* approach.

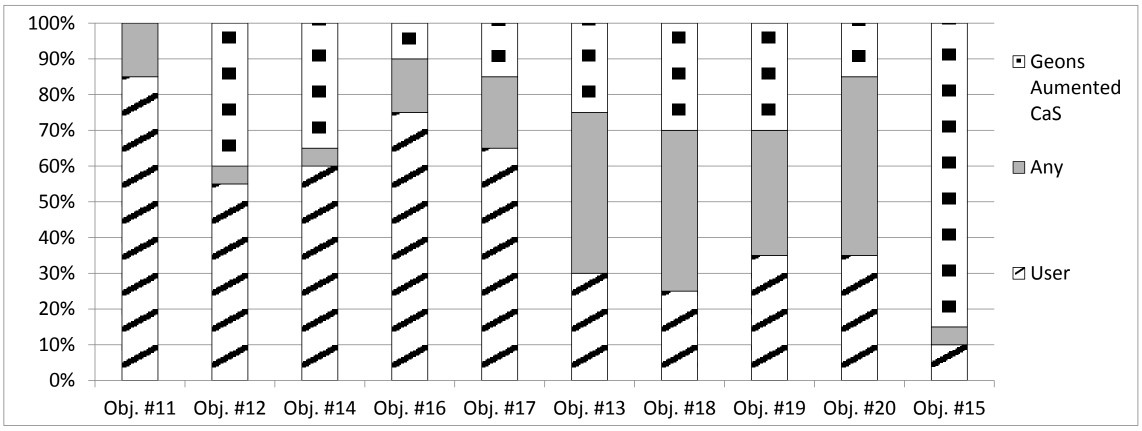


Figure 4.18: Results of comparing *Geons-augmented CaS* with users segmentation

the results produced by the user. The any approach option won on three objects, meaning that either *Geons-augmented CaS* has produced a result equivalent to what the users were expecting.

For one object we need to add the total of both and *Geon-augmented CaS* choices to get a bigger amount of votes than the user result, as the choices were equally distributed by the three possible options. With this result is possible to conclude that, for that object in particular, the users were not in consensus. Overall, the results are still good. The best possible case happened with object 15: the users preferred the Geon-augmented result to their own segmentation.

As shown in Figure 4.19, the segmentation result of object 18 produced by the manual segmentation has the same result as the one shown in Figure 4.20, which represents the segmentation result of object 18 using the *Geons-augmented CaS* approach. This object in particular had nine users choosing both results, with five preferring their results and six choosing the *Geons-augmented CaS* approach.

This analysis allowed us to conclude that, apart from a few objects where the segmentation is not the best for the user, the results for majority of the collection have been chosen by the user as being similar or better than the one he chose, proving the quality of the results.

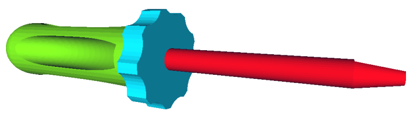


Figure 4.19: Object 18 with a segmentation result of the execution of the manual segmentation.

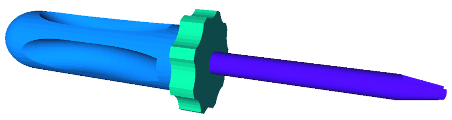


Figure 4.20: Object 18 with a segmentation result of the execution of the *Geons-augmented CaS* approach

4.5 Summary

In this chapter, two different types of tests were performed: tests with users and *Performance tests.* To execute these tests, different prototypes were used, in order to accomplish their goals.

The first to be executed was the *Manual Segmentation Test Using HFP*, which goal was to understand how humans decompose 3D objects. It consisted of asking the users to segment a collection of ten randomly chosen objects. To execute this task, the Shade Workbench tool for manual segmentation was used. With the results of this test, it was possible to conclude that, when the objects are more familiar, the segmentation chosen by user tends to be the same. When they are more complex the users tend to be in disagreement. It was also possible to highlight some limitations of the *HFP* algorithm, like the over-segmentation when the users try to isolate a feature of the objects, others becoming over-segmented.

After having identified the number of segments with more choices for each object, a comparison between the user segmentation and the *Adapted CaS* approach was performed. Then, as the results still were over-segmented, the approach was again improved, generating the *Geons-augmented CaS* approach. The comparison was once again performed, but this time between the user choices and the *Geons-augmented CaS* approach and defining the similarity and similar count threshold values.

To test the efficacy and effectiveness two tests were performed. In both the *Original CaS* prototype and the *Geons-augmented CaS* prototype were used. The time test was used to prove that the complexity of the algorithm, while also comparing the time tests of the *Original CaS*] and the *Geons-augmented CaS* approaches. This comparison proved the linear complexity and the improvement of the latter approach. On the other hand, the *Geons-augmented CaS* requires more memory, because it uses two lists instead of a tree. Nevertheless, memory is a cheap resource so it is a good trade off to have a faster technique.

The last test was again executed and the users were required to compare pairs of segmentation results and chose the one with the most meaningful segmentation. First, the users had to compare the results from the manual segmentation with the *Geons-augmented CaS* results and then the *Original CaS* results also with the ones produced by the *Geons-augmented CaS* approach. With these results it was possible to conclude that, comparatively to the *Original CaS*, the *Geons-augmented CaS* produces equal or better results. The second comparison allowed us to conclude that half of the results produced by the *Geons-augmented CaS* represented a meaningful segmentation for the users.

Chapter 5

Conclusion and Future Work

The *CaS* segmentation algorithm was initially proposed and integrated into a 3D object retrieval framework by Alfredo Ferreira et al. [10]. One of the main goals of this thesis was to isolate the *CaS* algorithm from the rest of the application and prove that it produces meaningful segmentation results to the user and that it could be improved comparatively to the original algorithm, where it uses the segments similarity in order to organize them.

To achieve these goals the *Adapted CaS* was developed. This algorithm segments a collection of objects using their geometric features in order to produce the collection of decomposed objects.

The *Adapted CaS* starts by creating the foundations, a hierarchical decomposition (*HSM*), the *Spherical Harmonics* and two main lists. Then, it decomposes by traversing the *HSM* and iterating the lists which contain the segments that are going to be decomposed or that are not decomposable. To be decomposable a segment has to be considered singular, meaning that it must have less than a predefined number of similar segments. *Spherical harmonics* and the difference between them are used to evaluate the segmented singularity.

After the execution of the test to understand how human beings segment objects, new changes were made and the *Geons-augmented* algorithm was developed. It consists on using *Geons* to decompose a segment. These *Geons* are simple objects, for instance, a cone or a cylinder, with particular characteristics. The main difference of the *Geons-augmented CaS* is that to be decomposable, a segment must not only have to be unique, but also cannot be similar to any of the *Geons*. *Spherical Harmonics* are also used for comparing each segment to all the *Geons*.

In the case of segmenting an object collection, the approaches that segment object by object can take a long time. So, one contribution of this work is the decomposition of an object collection by decomposing multiple objects simultaneously.

Another limitation that is very common in these approaches is the need for user interaction. We have overcome this limitation by predefining the thresholds that produce the most meaningful segmentation to the user.

With the evolution of the approach to the *Geons-augmented CaS*, we have overcome the over-segmentation limitation that was present on the *Adapted CaS*.

However the *Geons-augmented CaS* algorithm still has some limitations. The algorithm uses the *HFP*, so it has inherited some of the *HFP* limitations. The main limitation is specifically the plane primitive. It produces a plane segmentation result instead of a volume segmentation as the user would expect.

Different tests were performed in order to test different features of the technique. In *Manual Segmentation Test Using HFP* users were asked to manually segment a collection of objects to study how human beings segment objects.

These results were also used to study which thresholds produce better results, creating an automatic segmentation where there is no need to interact with users and to refine the approach.

Performance Tests were also executed to prove the efficacy and effectiveness of the algorithm. On these tests, the execution time and memory requirements were evaluated. These values were also compared with the *Original CaS* and the *Geons-augmented CaS* approaches. We concluded that even though the *Geons-augmented* uses more memory, the time spent on segmenting a collection is less then on the *Original CaS* approach, thus improving the overall performance of the approach.

Finally, to prove the quality of the results, a test was performed with users, the *CaS Evaluation With Users Test*, where it was required to choose between two results the one which represented the most meaningful segmentation. The users had to compare the results produced by the *Original CaS* and the *Geons-augmented CaS* which proved the improvement of the results produced by the *Geons-augmented* comparatively to the *Original CaS*. Then, the users also had to compare the results that they had produced on the *Manual Segmentation Test Using HFP* with the results produced by the *Geons-augmented*, in order to prove the quality of the results produced by the approach. With these results it was possible to conclude that for most of the objects, the approach produces a meaningful segmentation

Taking into account the referred limitation of using *HFP*, it is necessary on a future work to study the possibility of using other hierarchical based segmentation approaches different from *HFP*.

It also seems promising to experiment other descriptors different from the *Spherical Harmonics*. Also, as one of the characteristics of these descriptors is that they are not scale invariant, it is necessary to perform a deeper study on the *Geons*, like the one performed to the cylinder, using a range of the same *Geon* but with some variances on the shape.

In the future we believe that the proposed approach proves to be a robust, stable and scalable solution to the decomposition of 3D object collections.

Bibliography

[1] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 3D mesh segmentation methodologies for cad applications. *Computer-Aided Design and Applications*, 4(6):827–841, 2007.

[2] I. Atmosukarto and L. G. Shapiro. Global 3d mesh segmentation using local operators. In *Eurographics 2008 Workshop on 3D Object Retrieval*, 2008.

[3] M. Attene, S. Biasotti, M. Mortara, G. Patan, M. Spagnuolo, and B. Falcidieno. Computer graphics in italy: Computational methods for understanding 3d shapes. *Comput. Graph.*, 30(3):323–333, 2006.

[4] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.*, 22(3):181–193, 2006.

[5] M. Attene, S. Katz., M. Mortara, G. Patane, M. Spagnuolo, and A. Tal. Mesh segmentation - a comparative study. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, page 7, Washington, DC, USA, 2006. IEEE Computer Society.

[6] M. Attene, M. Mortara, M. Spagnuolo, and B. Falcidieno. Hierarchical convex approximation of 3d shapes for fast region selection. *Comput. Graph. Forum*, 27(5):1323–1332, 2008.

[7] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theor. Comput. Sci.*, 392(1-3):5–22, 2008.

[8] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[9] T. F. Cox, M. A. A. Cox, and T. F. Cox. *Multidimensional Scaling, Second Edition.* Chapman & Hall/CRC, September 2000.

[10] A. Ferreira, S. Marini, M. Attene, M. Fonseca, M. Spagnuolo, J. Jorge, and B. Falcidieno. Thesaurus-based 3d object retrieval with part-in-whole matching. *International Journal of Computer Vision.*

[11] M. Garland, A. Willmott, and P. 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.

[12] A. Golovinskiy and T. Funkhouser. Consistent segmentation of 3D models. *Computers and Graphics (Shape Modeling International 09)*, 33(3):262–269, June 2009.

[13] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions (Wiley Series in Probability and Statistics).* Wiley-Interscience, New York, revised edition, April 2005.

[14] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics*, 29(3), 2010.

[15] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[16] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 954–961, New York, NY, USA, 2003. ACM.

[17] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 183–191, New York, NY, USA, 2008. ACM.

[18] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Mesh scissoring with minima rule and part salience. *Comput. Aided Geom. Des.*, 22(5):444–465, 2005.

[19] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 2, New York, NY, USA, 2004. ACM.

[20] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[21] M. Mortara, G. Patan, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2003.

[22] M. Mortara, G. Patanè, and M. Spagnuolo. From geometric to semantic human body models. *Computers & Graphics*, 30(2):185–196, 2006.

[23] I. Pratikakis. *Watershed-driven image segmentation.* PhD thesis, Vrije Universiteit Brussel (VUB), 1998.

[24] G. Reeb. Sur les points singuliers dúne forme de pfaff completement integrable ou dúne fonction numrique. *Comptes Rendus des séances de lÁcadémie des sciences*, 222:847–849, 1946.

[25] S. Shalom, L. Shapira, A. Shamir, and D. Cohen-Or. Part analogies in sets of objects. In *Proceedings of Eurographics Symposium on 3D Object Retrieval*, pages 33–40, 2008.

[26] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[27] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3d objects. *International Journal of Computer Vision*, 2009.

[28] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonization using the shape diameter function. *The Visual Computer*, 24(4):249–259, April 2008.

[29] P. Simari, E. Kalogerakis, and K. Singh. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 111–119, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[30] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Topology driven 3d mesh hierarchical segmentation. *Shape Modeling and Applications, International Conference on*, 0:215–220, 2007.

[31] Y. Zhang, J. Paik, A. Koschan, and M. A. Abidi. A simple and efficient algorithm for part decomposition of 3d triangulated models based on curvature analysis. In *in Proceedings of the International Conference on Image Processing, III*, pages 273–276, 2002.

[32] E. Zuckerberger. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, October 2002.

Appendix A

Geons-augmented CaS Iterator Pseudo code

To complement the explanation of the iteration stage, this appendix presents a pseudocode of how that stage works, the *CaS*-Iterator. This pseudocode is the groundwork for the complexity conclusions.

```
CaS-Iterator($\mathcal{NDL}$)
 1  while $iterationCount <= depthThreshold$
 2     do
 3        for each $element$ in $\mathcal{NDL}$
 4           do
 5              el is the element of NDL
 6              if $Decomposable(el)$
 7                then
 8                     Add $\langle CaSTree_{el,child_{left}} \rangle$ to $NSL$
 9                     Add $\langle CaSTree_{el,child_{right}} \rangle$ to $NSL$
10                     Remove $\langle elem \rangle$ of $NDL$
11
12        if Is-Empty($NSL$)
13          then
14               return NDL;
15
16        for each $element$ in $\mathcal{NSL}$
17           do
18              elNSL is the element of NSL
19              Compute-Signature($elNSL$)
20              Add $\langle elNSL->signature \rangle$ to $SignaturePool$
21
22        Append $NSL$ to $NDL$
23        Clean $NSL$
24        iterationCount++;
```

It starts with a cycle that ends in the case of iterationCount being equal or smaller than a predefined depthThreshold (Line 1). Then, it enters a new cycle by iterating the NDL (Line 3). Next it verifies if the object is decomposable or not (Line 6). In the case of being decomposable, it decomposes the object by adding the respective children from the HSM to the NSL (Line 8 - Line 9) and removing the decomposed element from the NDL (Line 10). When the iteration of the NDL stops, because it reached the last element, it verifies if the NSL is empty (Line 12), if it is not, it passes to the next cycle that is going to iterate the NSL (Line 16) to append each element of the NSL to the NDL. So, it starts by calculating each signature of the elements of the NSL (Line 19), adds each signature to the Signature Pool (line 2). Next it appends all elements of the NSL to the NDL (Line 22) and finally cleans the NSL (Line 23). When the iteration finishes its count is increased and a new iteration starts. The algorithm finishes when the NSL is empty (Line 14).

Appendix B

Test Set

In order to execute the tests, two collections of objects were created. These are formed by ten objects that were randomly extracted from the Engineering Shape Benchmark (ESB).

These collections were used on the *Manual Segmentation Test Using HFP* and on the *CaS Evaluation With Users Test*. The collection shown on figure B.1 was used on the first test while the collection of figure B.2 was used on the second test. Both collections have ten objects because if we had used fewer objects, it would be not enough to give sufficient information to reach a conclusion. On the other hand, if the collection was bigger it would take too much time for the user to manually segment the entire collection, taking into account that segmenting manually an object can take an average of one minute per object.

Tests of performance also used the collection represented on figure of figure B.1 with more 50 objects for the first test, the second test had more 200 objects while the third test had more 400, the fourth had more 600 and finally the last test had all the objects of the ESB which is a total of 853 objects.

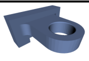
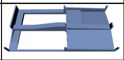
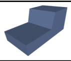
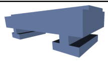
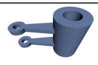
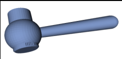
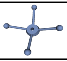
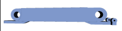
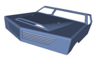
	Name	Image
Object1	advgr01	
Object2	aries129a	
Object3	ball_return	
Object4	BLUCO_412158_II	
Object5	CLAMP_SLIDER	
Object6	cpe1b1	
Object7	ganter_gm_99_2	
Object8	gantergn_213-60-b15	
Object9	link2	
Object10	tom_body	

Figure B.1: Collection of objects used in the *Manual Segmentation Test Using HFP*

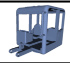
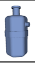
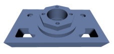
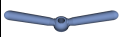
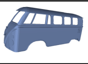
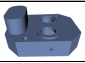
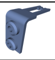
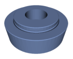
	Name	Image
Object11	cab_base	
Object12	COLLECTOR_40932	
Object13	demo08	
Object14	disk	
Object15	ganter99_8-32-m16	
Object16	leftroof	
Object17	part06	
Object18	ph_driver_prt	
Object19	schmersal_st_14_b5	
Object20	SUPPORTCYLINDER10	

Figure B.2: Collection of objects used in the *CaS Evaluation With Users Test*

Appendix C

Manual Segmentation Test Using HFP Results

This chapter of the appendix contains the tables and graphics of the *Manual Segmentation Test Using HFP* results as well as a resume of the obtained results. It starts with the protocol to the test execution.

C.1 Protocol for User Study

Introduction

For the past years, the human interpretation of the surrounding world has been subject of study in different areas including computer graphics, which has proven to be a challenge since the human being is an individual with his own characteristics. This test will more specifically, help to understand how humans decompose 3D objects.

This study emerges in the scope of the Collection-aware Segmentation project which is a master degree project in Information Systems and Computer Engineering. The main goal is to simultaneously segment a collection of objects based on their similarity, while producing meaningful results to user.

As the goal is to understand how human beings decompose 3D objects, this test will consist on one main step where the users will be asked to segment a collection of objects.

Methodology

To understand how people segment 3D objects, a test with users will be performed. The test consists on one main step that will be executed on a module of the Shade WB. This provides the segmentation functionality using *HFP*.

The test session consists on a total of two steps. These are identified on the next table (Table C.1) along with the respective anticipated duration time. After the table there is a brief explication of each one of these steps.

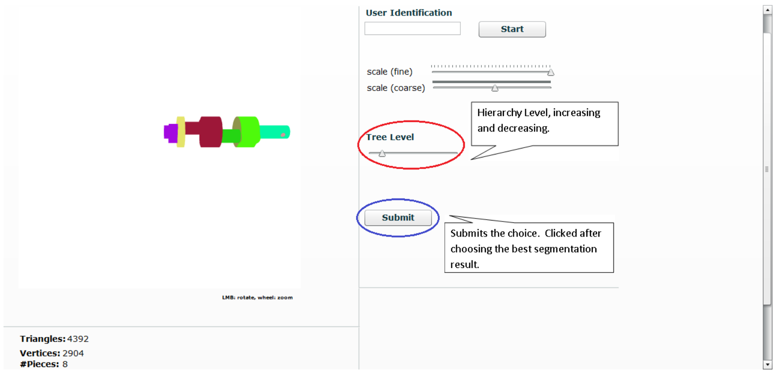
1	Filling in a questionnaire, explanation of the experience and of how the manual segmentation application works.	5 min
2	Segmentation of a collection of objects.	15 min
	Total time:	20 min

Table C.1: Table with the steps and respective estimated time

1. Filling in a questionnaire, explanation of the experience and of how the manual segmentation application works

In the beginning users will be asked to fill in a small questionnaire. This will help to understand some of the users' characteristics and if they have already used any modeling tool.

A user gide is given to the users, to guide them through the test. Then, the purpose and the procedure of the session is explained and finally the first application that he is going to use is presented to him,



Hierarchy Level, increasing and decreasing.

Submits the choice. Clicked after choosing the best segmentation result.

Triangles: 4392
Vertices: 2904
#Pieces: 8

Figure C.1: Shade Workbench manual segmentation tool interface with a segmented object with four clusters.

so he can explore it and learn how it works. More specifically, it will be asked to segment an object by choosing different levels of the Hierarchical Segmented Mesh.

To execute this task the user resorts, was previously refered, to the Shade Workbench tool that is represented on Figure C.1. Initially the entire object is presented to the user, without any segmentation, that is, it has just one cluster. Then, after having observed the object shape, by rotating and changing the dimensions, the user has to change the tree level that produces a segmentation result more meaningful to him. This number is changed using the scroll indicated with the red arrow, to save his choice, the user has to click on the submit button indicated with the blue arrow.

2. Segmentation of a collection of objects

After finishing the previous stage, the user already knows how the application works. So, in this stage he will be asked to segment a collection of objects.

In more detail, a set of objects us randomly presented to the user, one object at a time. For each object the user has to choose which segmentation makes more sense, that is, the user has to try out several levels of the hierarchy of the HSM and then choose the level that has the number of clusters which produces a segmentation that he thinks to be the more meaningful to him. This option is then registered by the application for a later evaluation.

Collection of objects to test

The users will segment a collection of ten objects (Figure C.2) that were randomly chosen from the Engineering Shape Benchmark (ESB), which is a benchmark with more than eight hundred objects. This collection of ten objects are randomly presented to the user, thus mitigating the influence of the chosen sequence on the segmentation results.

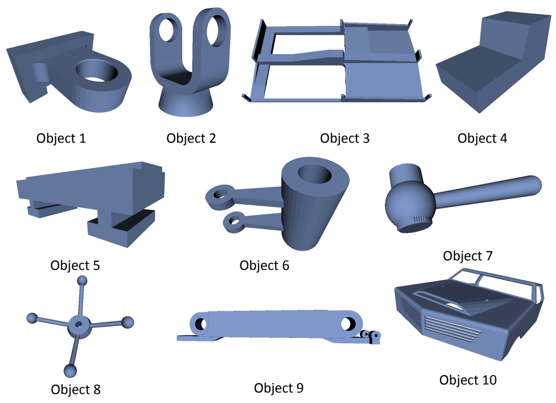


Figure C.2: Collection of 3D objects to segment.

Questionnaire

I appreciate in advance for your cooperation in filling in this questionnaire.

1. What is your age?
   (a) Between 15 and 20 years
   (b) Between 21 and 34 years
   (c) Between 35 and 55 years
   (d) More than 55 years
2. What is your gender?
   (a) Feminine
   (b) Masculine
3. What are your qualifications?
   (a) 9th degree or less
   (b) Between 9th degree 12th
   (c) 12th degree or equivalent
   (d) Academic degree
4. How do you consider yourself on the use of computers?
   (a) Inexperienced user
   (b) Regular user
   (c) Experienced user
5. Do you have any experience in applications for three-dimensional modulation?
   (a) None
   (b) Some
   (c) Many

I would like to thank you for your participation.

C.2 Detailed results using tables

In order to interpret the *Manual Segmentation Test Using HFP*, the results were registered in tables. The Table C.2 has the number of segments that each user chose and on the last line is used the mode, to calculate which number of segments has more user choices. However, what happens is that some objects can have the same number or very close amount of choices for different number of segments.

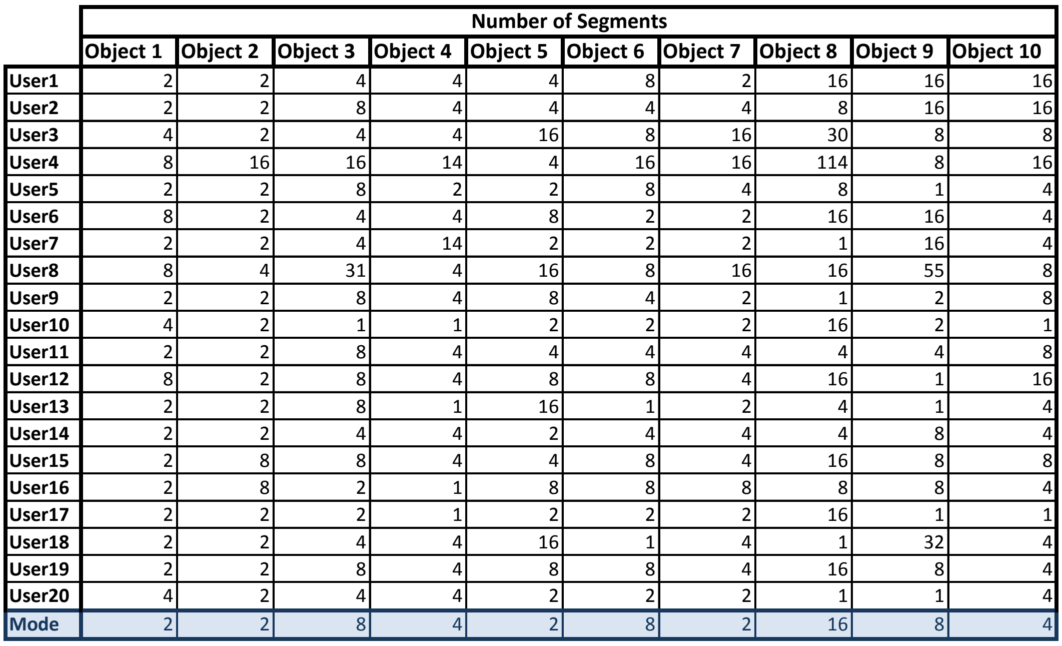
On table C.3 is the time that each user took to segment each object, and to segment the entire collection. The tables also presents the average time taken for each object and for the entire collection. With these values we can conclude that the users took on average, 10 minutes to segment the entire collection of objects and also that it took an average of one minute to segment each object. Some objects took more time than others because of their complexity and because of the results produced by the *HFP*.

C.3 Detailed results using bar charts

In order to have a better interpretation of the results, we used these bar charts, because just the previous table was not enough, due to the fact that an object can have the same number of user choices for different numbers of segments.

For some objects it was easier to conclude which is the segment number with the most meaning to the user just by observing these charts. For objects 1 and 2 for example, most of the users have chosen the result with 2 segments (Figure C.3 and Figure C.4), then for object 4 the majority of the users prefers the result with 4 segments and finally for object 8 the result with 16 segments proved to be the most meaningful.

But for other objects it was not so easy to come to a conclusion because users have chosen a wide range of number of segments. For these objects, we used pie charts in order to understand how many and which are the most meaningful results to the user. These are represented on the next section.



	Number of Segments									
	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6	Object 7	Object 8	Object 9	Object 10
User1	2	2	4	4	4	8	2	16	16	16
User2	2	2	8	4	4	4	4	8	16	16
User3	4	2	4	4	16	8	16	30	8	8
User4	8	16	16	14	4	16	16	114	8	16
User5	2	2	8	2	2	8	4	8	1	4
User6	8	2	4	4	8	2	2	16	16	4
User7	2	2	4	14	2	2	2	1	16	4
User8	8	4	31	4	16	8	16	16	55	8
User9	2	2	8	4	8	4	2	1	2	8
User10	4	2	1	1	2	2	2	16	2	1
User11	2	2	8	4	4	4	4	4	4	8
User12	8	2	8	4	8	8	4	16	1	16
User13	2	2	8	1	16	1	2	4	1	4
User14	2	2	4	4	2	4	4	4	8	4
User15	2	8	8	4	4	8	4	16	8	8
User16	2	8	2	1	8	8	8	8	8	4
User17	2	2	2	1	2	2	2	16	1	1
User18	2	2	4	4	16	1	4	1	32	4
User19	2	2	8	4	8	8	4	16	8	4
User20	4	2	4	4	2	2	2	1	1	4
Mode	2	2	8	4	2	8	2	16	8	4

Table C.2: Table with the number of segments chosen

	Time											
	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6	Object 7	Object 8	Object 9	Object 10	Total Sum	Average
User1	00:36	00:34	01:00	00:37	00:35	00:29	00:35	00:49	00:55	00:54	07:04	00:42
User2	01:02	00:38	00:58	00:54	01:03	01:00	00:48	00:45	00:54	00:59	09:01	00:54
User3	00:27	00:35	00:55	00:50	00:47	00:31	00:42	00:51	00:37	00:41	06:56	00:41
User4	01:23	01:17	01:13	01:11	01:35	01:03	00:33	01:26	02:56	01:52	14:29	01:26
User5	00:31	00:28	01:35	00:46	00:27	00:33	00:43	01:21	00:41	00:59	08:04	00:48
User6	01:10	00:19	00:54	00:29	00:31	00:49	00:44	01:16	01:32	01:13	08:57	00:53
User7	00:39	00:30	01:13	01:00	01:04	00:40	00:56	00:54	01:21	01:00	09:17	00:55
User8	01:05	02:03	01:24	00:53	01:00	01:11	00:47	01:17	01:46	00:39	12:05	01:12
User9	00:53	00:40	01:26	00:51	00:42	01:15	00:42	02:17	01:51	01:00	11:37	01:09
User10	00:20	00:25	00:45	01:15	00:18	00:36	00:14	00:46	01:28	00:09	06:16	00:37
User11	00:39	00:34	01:02	00:29	01:18	00:17	00:49	01:17	01:46	00:49	09:00	00:54
User12	01:22	01:31	01:15	00:54	02:07	02:08	01:11	01:18	02:47	01:27	16:00	01:36
User13	00:40	00:53	00:55	01:00	00:57	01:30	00:53	01:05	01:21	00:47	10:01	01:00
User14	00:58	01:26	02:16	01:44	01:18	01:22	00:35	02:54	02:01	01:29	16:03	01:36
User15	00:28	00:48	00:30	00:43	00:32	00:55	00:32	00:39	01:09	00:47	07:03	00:42
User16	01:03	01:18	00:42	00:40	00:43	01:07	00:40	01:09	01:31	01:36	10:29	01:02
User17	00:22	00:52	01:55	00:24	01:47	00:41	00:52	01:57	01:35	01:55	12:20	01:14
User18	00:33	00:42	02:05	02:05	01:22	01:54	00:31	01:31	02:06	01:16	14:05	01:24
User19	00:50	01:20	01:44	01:07	01:33	01:53	00:49	00:43	01:10	00:50	11:59	01:11
User20	00:36	00:20	01:13	01:29	00:42	01:07	00:45	01:41	01:59	01:37	11:29	01:08
Average	00:46	00:51	01:15	00:58	01:01	01:03	00:43	01:17	01:34	01:05	10:36	

Table C.3: Table with the times it took for each user to segment an object

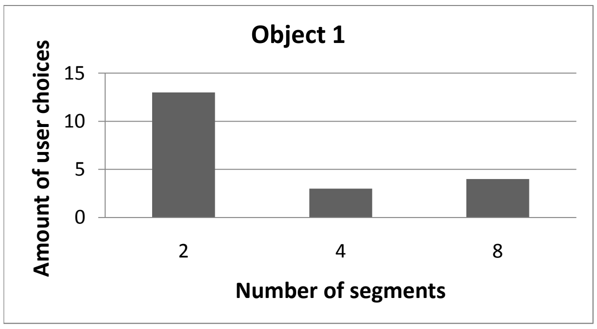


Figure C.3: Results of user choices for object1

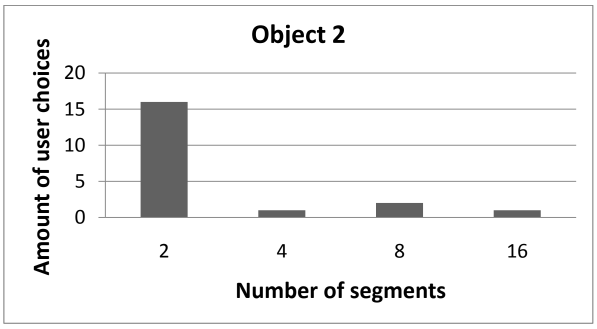


Figure C.4: Results of user choices for object2

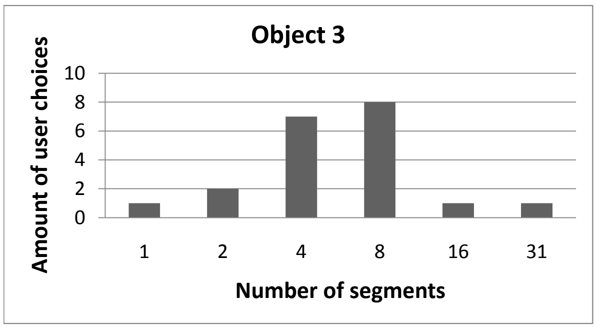


Figure C.5: Results of user choices for object3

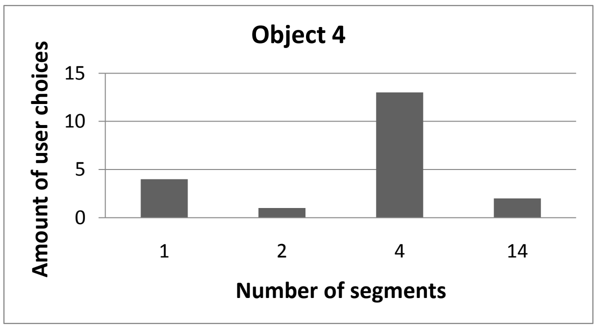


Figure C.6: Results of user choices for object4

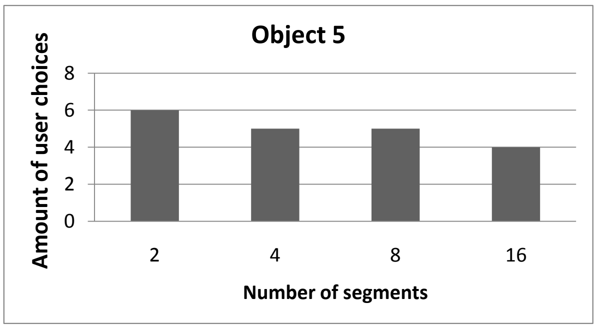


Figure C.7: Results of user choices for object5

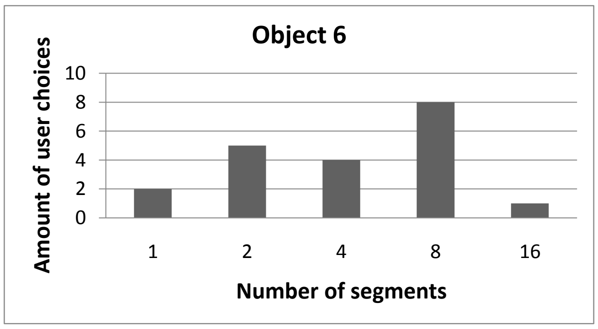


Figure C.8: Results of user choices for object6

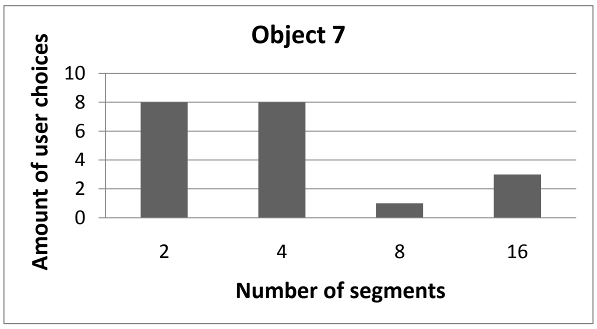


Figure C.9: Results of user choices for object7

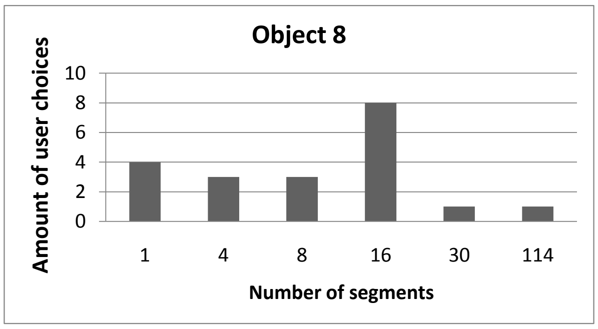


Figure C.10: Results of user choices for object8

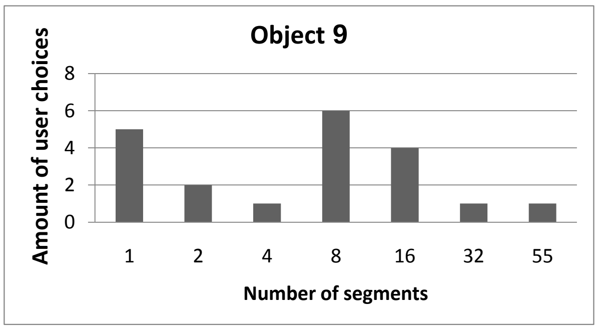


Figure C.11: Results of user choices for object9

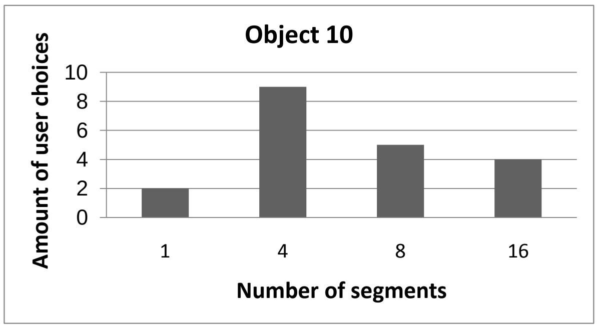


Figure C.12: Results of user choices for object10

C.4 Detailed results using pie charts with merged clusters

As it was previously described, for some objects it was not so easy to conclude which number of segments produced the most meaningful results to the user. So, on this section, we represented the pie charts that were used to reach a conclusion as well as the values extracted from them.

Although it is necessary to choose more than one segment, it is easy to chose which ones, because it only required to use two segment numbers in order to obtain the biggest range of choices, starting with object 3 where 75% of the users have chosen the results with 4 and 8 segments. The same happens to object 7 where the choices for the segment number are 2 and 4, both of these choices have the same number of user choices, making it a total of 80% of choices (figure C.16). For object 9 it is not so easy to come to a conclusion, it can be seen in Figure C.17 it is very ambiguous, because it has five users

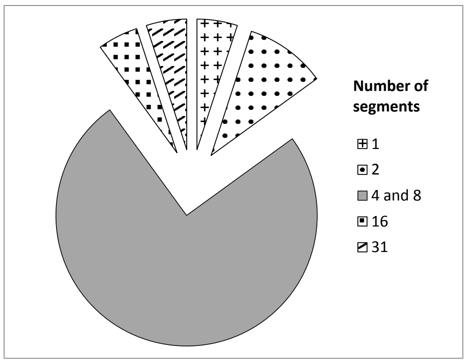


Figure C.13: Pie chart of the number of segments choices fusion of object 3

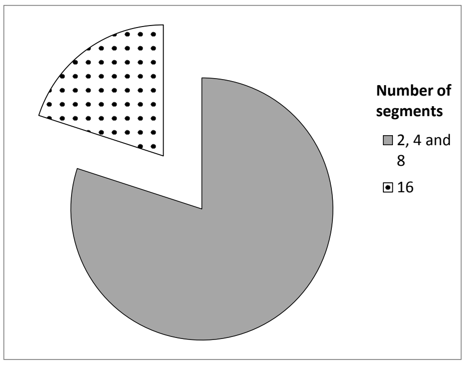


Figure C.14: Pie chart of the number of segments choices fusion of object 5

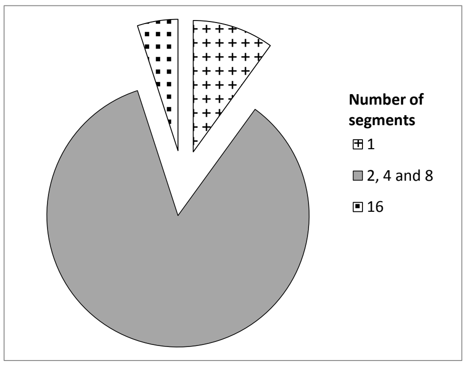


Figure C.15: Pie chart of the number of segments choices fusion of object 6

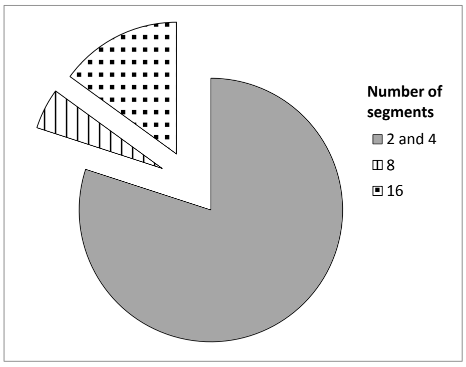


Figure C.16: Pie chart of the number of segments choices fusion of object 7

that have chosen one segment and six that have chosen eight. But by merging the number of segments 8 and 16 we get 50%, half of the choices. The users for object 10 have chosen only four different segment numbers, but 70% have chosen level 4 and 8 (figure C.18), the rest are on opposite sides, that is, 10% preferred to have the entire object and 20% decided to have a detailed segmentation.

The remaining objects proved to be harder to interpret and to solve this problem, we merged three numbers of segments. These have the number levels with very similar amounts of choices, like object 5 that has from four to five users in each segment number, so, to not use all these values, the only segment number that was excluded was the 16 because is very distant from the rest. This makes a total of 80% of user choices (Figure C.14). The object 10 is less ambiguous but if we merge the segments 2 and 4 we get nine users choosing these number of segments which is more choices then the result with 8, so to get a better result were chose the number of segments 2, 4 and 8 with a total of 85% (Figure C.15).

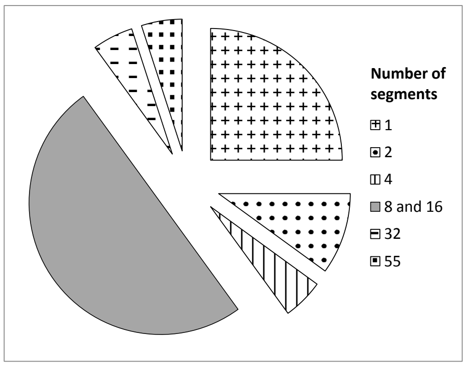


Figure C.17: Pie chart of the number of segments choices fusion of object 9

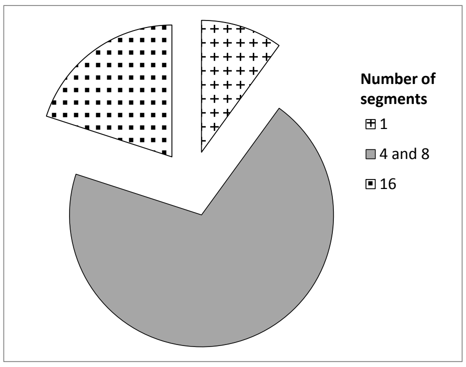


Figure C.18: Pie chart of the number of segments choices fusion of object 10

Appendix D

Comparative Results

For the purpose of reaching the goals of producing an automatic segmentation with results that are meaningful to the users it we executed the *Manual Segmentation Test Using HFP* which consisted on asking the users to segment a collection of objects. The results of this test were used to improve and refine the approach by comparing them to the results produced by *CaS* algorithm using different thresholds, with the purpose of defining these values.

The table D.1 has the values of the similarity and similar count thresholds that were used on each test. These values were found by first defining a minimum (0.3) and a maximum (1) and the mean of these (0.65). Then each interval ([0.3, 0.65] and [0.65, 1]) was also divided also in two by calculating a new mean for each of the two intervals (0,48 and 0.825), and so on, to see where it gets a segmentation result closest to the one selected by the user. The interval [0.3, 0.48] was immediately excluded because most of the results were over segmented. Then we also calculated the mean between the 0.48 and 0.65, getting the value of 0.60. This value and the 0.65 produced good results so we also tested 0.58 and 0.59. For the interval [0.65, 1] the opposite happened, that is, the objects are under segmented and most of them with the same number of clusters, even if we increase the similar-count threshold. In these cases the number of segments increase, but all objects have almost the same number of segments, so, these are not good values to use.

For comparing the results, we used the difference between number of segments and the percentage of user choices which is all represented in Table D.2.

	Similarity	Similar Count
Test1	0.3	1
Test2	0.3	2
Test3	0.48	1
Test4	0.56	1
Test5	0.58	1
Test6	0.59	1
Test7	0.6	1
Test8	0.65	1
Test9	0.65	2
Test10	0.65	3
Test11	0.7	1
Test12	0.738	1
Test13	0.738	2
Test14	0.825	1
Test15	0.825	2
Test16	0.825	3
Test17	1	1
Test18	1	2
Test19	1	3

Table D.1: Similarity and Similar count thresholds for each test

	Nº Cluster											
	1	2	4	8	14	16	30	31	32	52	55	114
obj1		65%	15%	20%								
obj2		80%	5%	10%		5%						
obj3	5%	10%	35%	40%		5%		5%				
obj4	20%	5%	65%		10%							
obj5		30%	25%	25%		20%						
obj6	10%	25%	20%	40%		5%						
obj7		40%	40%	5%		15%						
obj8	20%		15%	15%		40%	5%					5%
obj9	25%	10%	5%	30%		20%			5%		5%	
obj10	10%		45%	25%		20%						

Table D.2: Table of the user choices percentages for each object with each number of segments.

D.1 *Adapted CaS*

Aiming to define the values of the similar count and similarity thresholds, we compared different results of *CaS* approaches, depending on these values, with the number of segments that were previously chosen in the *Manual Segmentation Test Using HFP*. For that we used the *Adapted CaS* prototype that was executed with different parameters. To have a wider range of different values nineteen tests were executed. In the next tables are the results of these tests. Next to the number of the test is first, the similarity threshold used and then the value of the similar count threshold used.

To compare the results of the *Adapted CaS* approach with the results of the manual segmentation created by the users we used the difference between the number of segments, the percentage of choices and a classification.

With this comparison it was possible to notice that some objects were still over segmented relatively to the results produced by the user, and also, with this kind of objects, it was possible to observe that these segments that belonged to the over segmented objects were similar to primitives (cylinders, cones, spheres). The most obvious object where this happened is object 8, even having less segments than the result chosen by the user, it presents results where the spheres and cones are over-segmented. The tables show that this object just has a classification of one or two worst classification. So, to solve this limitation we added the Geons to label a segment as decomposable or not and a new study was performed.

	User	Test 1 (0.3 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	20	18	*
Obj2	2	25	23	*
Obj3	4/ 8	18	14/ 10	*
Obj4	4	4	0	****
Obj5	2/ 4/ 8	18	16/ 14/ 10	*
Obj6	2/ 4 / 8	21	19/ 17/ 13	*
Obj7	2/ 4	32	30/ 28	*
Obj8	16	41	25	*
Obj9	8/ 16	25	17/ 9	*
Obj10	4/ 8	31	27/ 23	*

Table D.3: Experimental results of test 1

	User	Test 2 (0.3 / 2)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	29	27	*
Obj2	2	32	30	*
Obj3	4/ 8	34	30/ 26	*
Obj4	4	4	0	****
Obj5	2/ 4/ 8	21	19/ 17/ 13	*
Obj6	2/ 4 / 8	29	27/ 25/ 21	*
Obj7	2/ 4	50	48/ 46	*
Obj8	16	52	36	*
Obj9	8/ 16	36	28/ 20	*
Obj10	4/ 8	48	44/ 40	*

Table D.4: Experimental results of test 2

	User	Test 3 (0.48 - 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	6	4	*
Obj2	2	7	5	*
Obj3	4/ 8	6	2/ -2	***
Obj4	4	3	-1	**
Obj5	2/ 4/ 8	8	6/ 2/ 0	***
Obj6	2/ 4/ 8	6	4/ 2/ -2	*
Obj7	2/ 4	13	11/ 9	*
Obj8	16	13	-3	**
Obj9	8/ 16	8	0/ -8	**
Obj10	4/ 8	9	5/ 1	***

Table D.5: Experimental results of test 3

	User	Test 4 (0.56 - 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	4	2	**
Obj3	4/ 8	4	0/ - 8	**
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	3	1/ -1/ -5	***
Obj6	2/ 4 / 8	5	3/ 1/ -3	*
Obj7	2/ 4	8	6/ 4	*
Obj8	16	8	-8	**
Obj9	8/ 16	6	-2/ -10	*
Obj10	4/ 8	5	1/ -3	***

Table D.6: Experimental results of test 4

	User	Test 5 (0.58 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	4	0/ -4	**
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	*
Obj7	2/ 4	7	5 / 3	**
Obj8	16	6	-10	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4 / 8	4	0/ -4	***

Table D.7: Experimental results of test 5

	User	Test 6 (0.59 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	4	0/ -4	**
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4 / 8	4	2/ 0/ -4	*
Obj7	2/ 4	5	3/ 1	***
Obj8	16	6	-10	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	3	-1/ -5	**

Table D.8: Experimental results of test 6

	User	Test 7 (0.6 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	3	-1 / -5	***
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/-2/ -6	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	*
Obj7	2/4	5	3/ 1	***
Obj8	16	6	-10	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	3	-1/ -5	**

Table D.9: Experimental results of test 7

	User	Test 8 (0.65 - 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	3	1/ -5	***
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4 / 8	3	1/ -1/ -5	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	5	-11	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	2	-2/ -6	*

Table D.10: Experimental results of test 8

	User	Test 9 (0.65 - 2)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	4	2	**
Obj2	2	5	3	*
Obj3	4/ 8	5	1/ -3	***
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	*
Obj6	2/ 4/ 8	6	4/ 2/ -2	*
Obj7	2/ 4	9	7/ 5	*
Obj8	16	8	-8	**
Obj9	8/ 16	8	0/ -8	**
Obj10	4/ 8	6	2/ -2	***

Table D.11: Experimental results of test 9

	User	Test 10 (0.65 / 3)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	5	3	*
Obj2	2	7	5	*
Obj3	4/ 8	10	6/ 2	*
Obj4	4	4	0	****
Obj5	2/ 4/ 8	7	5/ 3/ -1	*
Obj6	2/ 4 / 8	7	5/ 3/ -1	***
Obj7	2/ 4	8	6/ 4	*
Obj8	16	6	-10	**
Obj9	8/ 16	10	2/ -6	**
Obj10	4/ 8	7	3/ -1	***

Table D.12: Experimental results of test 10

	User	Test 11 (0.7/ 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	**
Obj2	2	2	0	*
Obj3	4/ 8	2	-2/ -4	***
Obj4	4	2	-2	***
Obj5	2/ 4/ 8	2	0/ -2/ -4	*
Obj6	2/ 4/ 8	2	0/ -2/ -4	*
Obj7	2/ 4	3	1/ -1	**
Obj8	16	3	-11	*
Obj9	8/ 16	2	-6/ -14	**
Obj10	4/ 8	2	-2/ -6	***

Table D.13: Experimental results of test 11

	User	Test 12 (0.738 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -4	**
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/ -2/ -4	***
Obj6	2/ 4 / 8	2	0/ -2/ -4	***
Obj7	2/ 4	2	0/ -2	***
Obj8	16	2	-14	*
Obj9	8/ 16	3		*
Obj10	4/ 8	2	-2/ -6	***

Table D.14: Experimental results of test 12

	User	Test 13 (0.738 / 2)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	4	2	**
Obj2	2	4	2	*
Obj3	4/ 8	5	1/ -3	***
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	*
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	5	3/ 1	**
Obj8	16	5	-11	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.15: Experimental results of test 13

	User	Test 14 (0.825 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -4	**
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/ -2/ -4	***
Obj6	2/ 4 / 8	2	0/ -2/ -4	***
Obj7	2/ 4	2	0/ -2	***
Obj8	16	2	-14	*
Obj9	8/ 16	2	-6/ -14	*
Obj10	4/ 8	2	-2/ -6	*

Table D.16: Experimental results of test 14

	User	Test 15 (0.825 / 2)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	4	2	**
Obj2	2	4	2	*
Obj3	4/ 8	4	0/ -4	***
Obj4	4	3	-1	**
Obj5	2/ 4/ 8	4	2/ 0/ -4	*
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	5	-11	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.17: Experimental results of test 15

	User	Test 16 (0.825 / 3)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	4	2	**
Obj2	2	4	2	*
Obj3	4/ 8	5	1/ -3	***
Obj4	4	3	-1	**
Obj5	2/ 4/ 8	4	2/ 0/ -4	*
Obj6	2/ 4 / 8	4	2/ 0/ -4	***
Obj7	2/ 4	5	3/ -1	**
Obj8	16	4	-12	*
Obj9	8/ 16	7	-1/ -9	**
Obj10	4/ 8	5	1/ -3	***

Table D.18: Experimental results of test 16

	User	Test 17 (1 / 1)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -4	**
Obj4	4	2	-2	*
Obj5	2/ 4/ 8	2	0/ -2/ -4	***
Obj6	2/ 4/ 8	2	0/ -2/ -4	***
Obj7	2/ 4	2	0/ -2	***
Obj8	16	2	-14	*
Obj9	8/ 16	2	-6/ -14	*
Obj10	4/ 8	2	-2/ -6	*

Table D.19: Experimental results of test 17

	User	Test 18 (1 / 2)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	4	2	**
Obj2	2	4	2	*
Obj3	4/ 8	4	0/ -4	***
Obj4	4	3	-1	**
Obj5	2/ 4/ 8	4	2/ 0/ -4	*
Obj6	2/ 4 / 8	4	2/ 0/ -4	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	4	-12	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.20: Experimental results of test 18

	User	Test 19 (1 / 3)		
	Nº Clusts	Nº Clusts	Dif	Quality
Obj1	2	4	2	**
Obj2	2	4	2	*
Obj3	4/ 8	4	0/ -4	***
Obj4	4	3	-1	**
Obj5	2/ 4/ 8	4	2/ 0/ -4	*
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	4	-12	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.21: Experimental results of test 19

	Choices %									
	Test1	Teste2	Teste3	Teste4	Teste5	Teste6	Teste7	Teste8	Teste9	Teste10
obj1	0	0	0	65	65	65	65	65	15	0
obj2	0	0	0	5	80	80	80	80	0	0
obj3	0	0	0	35	35	35	0	0	0	0
obj4	65	65	0	5	5	5	5	5	0	65
obj5	0	0	25	0	30	30	30	30	25	0
obj6	0	0	0	0	20	20	20	0	0	0
obj7	0	0	0	5	0	0	0	40	0	5
obj8	0	0	0	15	0	0	0	0	15	0
obj9	0	0	30	0	5	5	5	5	30	0
obj10	0	0	0	0	45	0	0	10	0	0
	65	65	55	130	285	240	205	235	85	70

Table D.22: The percentage according to user choices for the comparison test

	Choices %								
	Teste11	Teste12	Teste13	Teste14	Teste15	Teste16	Teste17	Teste18	Teste19
obj1	65	65	15	65	15	15	65	15	15
obj2	80	80	5	80	5	5	80	5	5
obj3	10	10	0	10	35	0	10	35	35
obj4	5	5	0	5	0	0	5	0	0
obj5	30	30	25	30	25	25	30	25	25
obj6	25	25	20	25	20	20	25	20	20
obj7	0	40	0	40	40	0	40	40	40
obj8	0	0	0	0	0	15	0	15	15
obj9	5	0	5	10	5	0	10	5	5
obj10	45	0	15	0	45	0	0	45	45
	265	255	85	265	190	80	265	205	205

Table D.23: The percentage according to user choices for the comparison test

D.2 *Geons-augmented CaS*

After improving the technique by adding the Geons to segment an object we needed to choose again which thresholds values produced the most meaningful segmentation. So, we repeated the execution *CaS* prototype, only now, with the *Geons-augmented* version (section 3.5) and also compared the results with the manual segmentation. With this comparison is possible to prove that it has reached it goal, overcome the over segmentation limitation, noticing that now for object 8 some tests have a classification of better results.

	User	Test 1 (0.3 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	16	14	*
Obj2	2	21	19	*
Obj3	4/ 8	18	14/ 10	*
Obj4	4	4	0	***
Obj5	2/ 4/ 8	13	11/ 9/ 5	*
Obj6	2/ 4 / 8	13	11/ 9/ 5	*
Obj7	2/ 4	16	14/ 12	*
Obj8	16	7	-9	****
Obj9	8/ 16	23	15/ 7	*
Obj10	4/ 8	27	23/ 19	*

Table D.24: Experimental results of test 1

	User	Test 2 (0.3 / 2)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	22	20	*
Obj2	2	25	23	*
Obj3	4/ 8	34	30/ 26	*
Obj4	4	4	0	***
Obj5	2/ 4/ 8	15	13/ 11/ 7	*
Obj6	2/ 4 / 8	16	14/ 12/ 8	*
Obj7	2/ 4	21	19/ 17	*
Obj8	16	7	-9	****
Obj9	8/ 16	34	26/ 18	*
Obj10	4/ 8	37	33/ 29	*

Table D.25: Experimental results of test 2

	User	Test 3 (0.48 - 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	6	4	*
Obj2	2	7	5	*
Obj3	4/ 8	7	3/ -1	**
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	6	4/ 2/ -2	**
Obj6	2/ 4/ 8	5	3/ 1/ -3	**
Obj7	2/ 4	9	7/ 5	***
Obj8	16	7	9	****
Obj9	8/ 16	8	0/ -8	***
Obj10	4/ 8	9	5/ 1	***

Table D.26: Experimental results of test 3

	User	Test 4 (0.56 - 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	4	2	*
Obj3	4/ 8	4	0/ -4	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	3	1/ -1/ -5	****
Obj6	2/ 4 / 8	5	3/ 1/ -3	*
Obj7	2/ 4	7	5/ 3	****
Obj8	16	5	11	****
Obj9	8/ 16	4	-4/ -12	**
Obj10	4/ 8	7	3/ -1	***

Table D.27: Experimental results of test 4

	User	Test 5 (0.58 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	4	0/ -4	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	3	1/ -1/ -5	***
Obj7	2/ 4	5	3/ 1	****
Obj8	16	5	11	****
Obj9	8/ 16	4	-4/ -12	**
Obj10	4 / 8	4	0/ -4	***

Table D.28: Experimental results of test 5

	User	Test 6 (0.59 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	4	0/ -4	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	3	1/ -1/ -5	***
Obj7	2/ 4	5	3/ 1	****
Obj8	16	5	11	****
Obj9	8/ 16	4	-4/ -12	**
Obj10	4 / 8	3	1/ -5	**

Table D.29: Experimental results of test 6

	User	Test 7 (0.6 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	3	1/ -5	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	3	1/ -1/ -5	***
Obj7	2/4	5	3/ 1	****
Obj8	16	5	11	****
Obj9	8/ 16	4	-4/ -12	**
Obj10	4/ 8	3	1/ -5	**

Table D.30: Experimental results of test 7

	User	Test 8 (0.65 - 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	3	1/ -5	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	2	0/ -2/ -6	***
Obj7	2/4	3	1/ -1	**
Obj8	16	4	12	**
Obj9	8/ 16	4	-4/ -12	**
Obj10	4/ 8	2	2/ -6	**

Table D.31: Experimental results of test 8

	User	Test 9 (0.65 - 2)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	4	2	*
Obj2	2	5	3	*
Obj3	4/ 8	5	1/ -3	***
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	***
Obj6	2/ 4/ 8	5	3/ 1/ -3	**
Obj7	2/ 4	7	5/ 3	**
Obj8	16	6	10	****
Obj9	8/ 16	7	-1/ -9	***
Obj10	4/ 8	6	2/ -2	***

Table D.32: Experimental results of test 9

	User	Test 10 (0.65 / 3)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	5	3	**
Obj2	2	7	5	*
Obj3	4/ 8	10	6/ 2	*
Obj4	4	4	0	****
Obj5	2/ 4/ 8	7	5/ 3/ -1	****
Obj6	2/ 4/ 8	7	5/ 3/ -1	***
Obj7	2/ 4	8	6/ 4	**
Obj8	16	6	10	****
Obj9	8/ 16	10	2/ -6	***
Obj10	4/ 8	7	3/ -1	**

Table D.33: Experimental results of test 10

	User	Test 11 (0.7/ 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -6	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	2	0/ -2/ -6	***
Obj7	2/ 4	3	1/ -1	***
Obj8	16	3	-13	**
Obj9	8/ 16	3	-5/ -13	*
Obj10	4/ 8	2	-2/ -6	**

Table D.34: Experimental results of test 11

	User	Test 12 (0.738 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -6	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	2	0/ -2/ -6	***
Obj7	2/ 4	2	0/ -2	***
Obj8	16	2	-14	*
Obj9	8/ 16	3	-5/ -13	*
Obj10	4/ 8	2	-2/ -6	**

Table D.35: Experimental results of test 12

	User	Test 13 (0.738 / 2)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	4	-2	*
Obj2	2	4	-2	*
Obj3	4/ 8	5	1/ -3	***
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	5	3/ 1	**
Obj8	16	4	-12	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.36: Experimental results of test 13

	User	Test 14 (0.825 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -6	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	2	0/ -2/ -6	***
Obj7	2/ 4	2	0/ -2	***
Obj8	16	2	-14	*
Obj9	8/ 16	2	-6/ -14	*
Obj10	4/ 8	2	-2/ -6	**

Table D.37: Experimental results of test 14

	User	Test 15 (0.825 / 2)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	4	-2	*
Obj2	2	4	-2	*
Obj3	4/ 8	4	0/ -4	**
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	4	-12	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.38: Experimental results of test 15

	User	Test 16 (0.825 / 3)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	4	-2	*
Obj2	2	4	-2	*
Obj3	4/ 8	5	1/ -3	***
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	5	3/ 1	****
Obj8	16	4	-12	*
Obj9	8/ 16	7	-1/ -9	***
Obj10	4/ 8	5	1/ -3	**

Table D.39: Experimental results of test 16

	User	Test 17 (1 / 1)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	2	0	***
Obj2	2	2	0	***
Obj3	4/ 8	2	-2/ -6	**
Obj4	4	2	-2	**
Obj5	2/ 4/ 8	2	0/ -2/ -6	***
Obj6	2/ 4/ 8	2	0/ -2/ -6	***
Obj7	2/ 4	2	0/ -2	***
Obj8	16	2	-14	*
Obj9	8/ 16	2	-6/ -14	*
Obj10	4/ 8	2	-2/ -6	**

Table D.40: Experimental results of test 17

	User	Test 18 (1 / 2)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	4	-2	*
Obj2	2	4	-2	*
Obj3	4/ 8	4	0/ -4	**
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	4	-12	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.41: Experimental results of test 18

	User	Test 19 (1 / 3)		
	Nº Clust	Nº Clust	Dif	Quality
Obj1	2	4	-2	*
Obj2	2	4	-2	*
Obj3	4/ 8	4	0/ -4	**
Obj4	4	3	-1	***
Obj5	2/ 4/ 8	4	2/ 0/ -4	***
Obj6	2/ 4/ 8	4	2/ 0/ -4	***
Obj7	2/ 4	4	2/ 0	***
Obj8	16	4	-12	*
Obj9	8/ 16	4	-4/ -12	*
Obj10	4/ 8	4	0/ -4	***

Table D.42: Experimental results of test 19

	Choices %									
	Test1	Teste2	Teste3	Teste4	Teste5	Teste6	Teste7	Teste8	Teste9	Teste10
obj1	0	0	0	65	65	65	65	65	15	0
obj2	0	0	0	5	80	80	80	80	0	0
obj3	0	0	0	35	35	35	0	0	0	0
obj4	65	65	0	5	5	5	5	5	0	65
obj5	0	0	0	0	30	30	30	30	25	0
obj6	0	0	0	0	0	0	0	25	0	0
obj7	0	0	0	0	0	0	0	0	0	0
obj8	0	0	0	0	0	0	0	15	0	0
obj9	0	0	30	5	5	5	5	5	0	0
obj10	0	0	0	0	45	0	0	0	0	0
	65	65	30	115	265	220	185	225	40	65

Table D.43: The percentage according to user choices for the comparison test

	Choices %								
	Teste11	Teste12	Teste13	Teste14	Teste15	Teste16	Teste17	Teste18	Teste19
obj1	65	65	15	65	15	15	65	15	15
obj2	80	80	5	80	5	5	80	5	5
obj3	10	10	0	10	35	0	10	35	35
obj4	5	5	0	5	0	0	5	0	0
obj5	30	30	25	30	25	25	30	25	25
obj6	25	25	25	25	20	20	25	20	20
obj7	0	40	0	40	40	0	40	40	40
obj8	0	0	15	0	15	15	0	15	15
obj9	0	0	5	10	5	0	10	5	5
obj10	0	0	45	0	45	0	0	45	45
	215	255	135	265	205	80	265	205	205

Table D.44: The percentage according to user choices for the comparison test

Appendix E

CaS Evaluation With Users Test

As we aimed for producing meaningful segmentation results, it was necessary to prove the quality of these results. This chapter of the appendix has the tables and graphics of the *CaS Evaluation With Users Test* results as well as a summary of the obtained results.

E.1 Protocol for User Study

Introduction

Nowadays, many computer graphics applications use the segmentation of 3D objects, specially the segmentation which has in consideration the human perception. So, one of the most important aims of these segmentation approaches is to have a meaningful segmentation to the user. In order to achieve this goal, for the proposed approach this test will be performed and its purpose is to prove quality of the results produced by this technique.

This study emerges in the scope of the Collection Aware Segmentation project which is a master degree project in Information Systems and Computer Engineering. The main goal is to segment a collection of objects simultaneously based on the segments similarity.

To evaluate the results, this test will consist of two main steps, one where the users will segment a collection of objects, and a second where the users will compare both results of the *Original CaS* and *Geons-augmented CaS* approach and compare the *Geons-augmented CaS* approach results with the results of the previous step.

Methodology

To prove the quality of the produced results, a test with user will be performed, which will be consisted by two main steps with different tasks. Both will be executed on a different module of the Shade WB for each. On the first step the Shade WB tools used provides the segmentation functionality using *HFP*. The second module used for the second main step shows two objects but with different segmentation results and three options for the user to choose: the left, right or any result.

The test session consists on a total of four steps. These are identified on the next table (E.1) along with the respective anticipated duration time. After the table there is a brief explication of each one of these steps.

1	Filling in a questionnaire, explanation of the experience and how the manual segmentation application works.	5 min
2	Segmentation of a collection of objects.	15 min
3	Explanation of how the comparison application works.	3 min
4	Choosing the best segmentation result.	15 min
	Total time:	38 min

Table E.1: Table with the steps and respective estimated time

1. Filling in a questionnaire, explanation of the experience and how the manual segmentation application works

In the beginning the users will be asked to fill in a small questionnaire. This will help to understand some of the users characteristics and if they use any tool of modeling.

A user guide is given to the users, to guide through the tes, then, the purpose and the procedure of the session will be explained to the user and finally the first application that he is going to use is presented to the user, so he can explore it and learn how it works. More specifically, it will be asked to segment an object by choosing different numbers of clusters.

To execute this task the users resorts, as was previously referred, to the Shade Workbench tool that is represented on E.1. Initially the entire object is presented to the user, without any segmentation, that is, it has just one cluster. Then, after having observed the object shape, by rotating and changing the dimensions, the user has to change the number of segments, in the example on Figure E.1, is an object with four clusters. This number is changed using the scroll indicated with the red arrow and, in order to save his choice, the user has to click on the submit button indicated with the blue arrow.

2. Segmentation of a collection of objects

After finishing the previous stage, the user already knows how the application works. So, in this stage he will be asked to segment a collection of objects.

In more detail, a set of objects us randomly presented to the user, one object at a time. For each object the user has to choose which segmentation makes more sense, that is, the user has to try out several levels of the hierarchy of the HSM and then choose the level that has the number of clusters which produces a segmentation that he thinks to be the more meaningful to him. This option is then registered by the application for a later evaluation.

This step is executed so that the results can be used on the next step. Because the approach was tuned to another collection of ten objects, we want to prove that this approach still presents meaningful results to other collections of objects.

3. Explanation of how the comparison application works

As we are aiming to prove the quality of the *Geons-augmented CaS* results with this test. To execute this step of the test it is going to be presented to the user another tool of the Shade Workbench, the comparison tool, so that he can explore it and learn how it works.

In this step to the user is shown the same object with two different segmentation results, as shown on Figure E.2, then the user has to chose an option by highlighting a blue rectangle. The user can choose

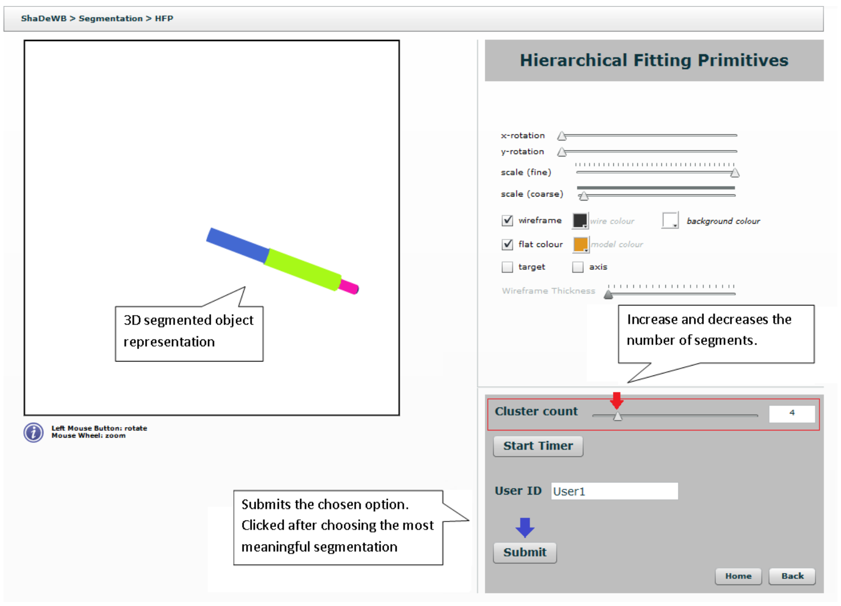

ShaDeWB > Segmentation > HFP
Hierarchical Fitting Primitives
x-rotation
y-rotation
scale (fine)
scale (coarse)
wireframe
wire colour
background colour
flat colour
model colour
target
axis
Wireframe Thickness
3D segmented object representation
Increase and decreases the number of segments.
Left Mouse Button: rotate
Mouse Wheel: zoom
Cluster count
4
Start Timer
User ID
User1
Submits the chosen option. Clicked after choosing the most meaningful segmentation
Submit
Home
Back

Figure E.1: Shade Workbench manual segmentation tool interface with a segmented object with four clusters.

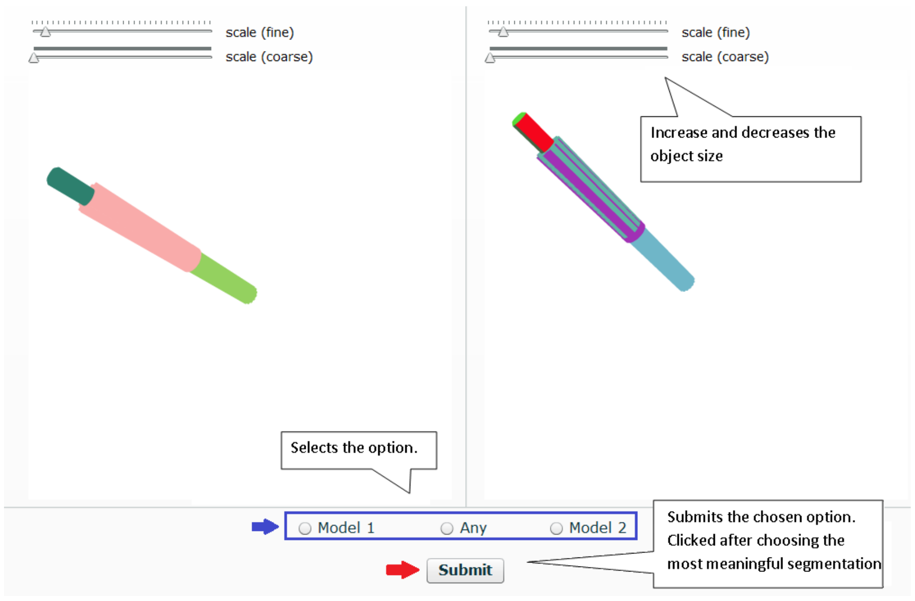

scale (fine)
scale (coarse)
scale (fine)
scale (coarse)
Increase and decreases the object size
Selects the option.
Model 1
Any
Model 2
Submit
Submits the chosen option. Clicked after choosing the most meaningful segmentation

Figure E.2: Shade Workbench comparison tool interface with two different segmentation results.

the Model 1 option if he prefers the left result, otherwise he must chose the Model 2 option. In the case of both techniques being the same or very close in the way that the user thinks both are meaningful results, then he should chose the any choice. Then, to save his option, he clicks on the Submit button indicated with the red arrow.

4. Choosing the best segmentation result.

As this task it to evaluate the quality of the *Geons-augmented CaS* results and as the user already understood the task and how the prototype works, on this step the user is going to compare pairs of segmentation results.

To prove the approach improvement it is necessary to compare the results from the *Original CaS* with the *Geons-augmented CaS* and also the results produced by this approach with the results produced by users on the previous step of manual segmentation. For that, the same collection of ten objects will be used. First they were segmented using the *Original CaS* approach and then segmented by the *Geons-augmented CaS* and then they were segmented by the users the *Manual Segmentation Test Using HFP*.

During the test this objects with different results will appear randomly to the user so there will not be influence on the results, Even the side on which each result appears will be ramdomized, meaning that one object may present the *Original CaS* result on the left, and the *Geons-augmented CaS* result on the right, and on the next, this order can randomly be the same or inverted.

Collection of objects to test

The users will segment a collection of ten objects that were randomly chosen from the Engineering Shape Benchmark (ESB) which is a benchmark with more than eight hundred objects. This collection of ten objects (Figure E.3) will appear randomly to the user so the sequence cannot induce the segmentation results.

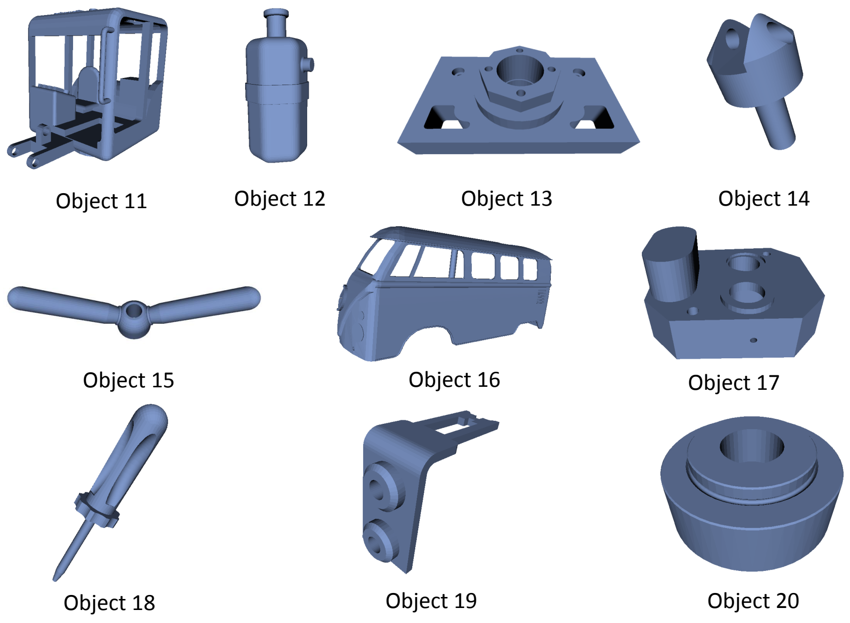


Figure E.3: Collection of 3D objects to segment.

Questionnaire

I appreciate in advance for your cooperation in filling in this questionnaire.

1. What is your age?
   (a) Between 15 and 20 years
   (b) Between 21 and 34 years
   (c) Between 35 and 55 years
   (d) More than 55 years
2. What is your gender?
   (a) Feminine
   (b) Masculine
3. What are your qualifications?
   (a) 9th degree or less
   (b) Between 9th degree 12th
   (c) 12th degree or equivalent
   (d) Academic degree
4. How do you consider yourself on the use of computers?
   (a) Inexperienced user
   (b) Regular user
   (c) Experienced user
5. Do you have any experience in applications for three-dimensional modulation?
   (a) None
   (b) Some
   (c) Many

I would like to thank you for your participation.

E.2 Detailed results using tables

On Table E.2 are the results of the number of segments that each user has chosen for each object. In order to obtain the number of segments with more choices, we used the mode. As these conclusions are not the most important we will not focus on them but on the results that are on the next section.

	Segment Number									
	Object 11	Object 12	Object 13	Object 14	Object 15	Object 16	Object 17	Object 18	Object 19	Object 20
User1	6	2	2	2	4	4	3	3	2	2
User2	5	4	5	4	4	4	5	3	4	4
User3	5	1	3	3	1	1	6	5	1	6
User4	4	2	2	2	4	4	5	4	6	4
User5	4	4	2	2	2	2	3	2	4	3
User6	5	2	3	2	4	2	3	2	2	4
User7	8	2	3	2	2	1	2	2	2	3
User8	7	6	2	2	4	4	10	4	2	4
User9	8	2	2	2	1	4	2	3	2	2
User10	6	7	7	3	2	5	4	2	3	5
User11	4	4	5	4	4	4	5	2	5	4
User12	6	2	2	3	4	4	4	3	3	4
User13	16	9	6	2	4	7	7	2	11	5
User14	10	8	7	4	5	8	9	5	15	8
User15	5	4	3	4	4	3	5	5	5	4
User16	14	9	6	4	4	6	10	4	27	8
User17	10	9	5	4	6	7	3	4	4	5
User18	4	4	2	2	2	5	4	4	2	3
User19	4	2	2	4	4	3	3	3	2	6
User20	5	2	3	3	4	4	3	3	4	2
Mode	5	2	2	2	4	4	3	3	2	4

Table E.2: Table with the number of segments chosen

E.3 Manual Segmentation VS *Geons-augmented CaS*

This test consisted on proving the quality of the approach relatively to the results of the decomposed objects produced by the *Geons-augmented CaS* approach. To prove this we use the results of image E.4 where it is possible to observe that for some objects it is clear that the users preferred their results instead of the *CaS* results and for other objects the *Geons-augmented CaS* approach seems to produce better results.

The objects that had results worst than the chosen by users are object 11, where most of the users, seventeen have chosen their segmentation. Another object which has more choices of the user option is the object 16 with a total of fifteen users choosing the manual segmentation result, besides that, the five remaining users chose any and *CaS* option instead of theirs. Finally, the last object that users have chosen is object 17 with thirteen choosing their results, although a total of seven users chose any option or *CaS* approach.

Two of the collection objects have really close numbers on the *Geons-augmented CaS* option and the user result. That is, object 12, for instance, has eleven choices of user results and nine as a total of any and *CaS* results, meaning that almost half of the sample think that the *CaS* approach produces a meaningful result. Object 14 has twelve choices of the user results and eight for the any and *CaS* options.

For objects 13 and 18 the option with more choices is the any, it was chosen by nine users. Also for object 20 the any option had more choices, with a total of ten. That is, half of the users have produced the same result or the results are similar to the ones by the approach, so it is equally meaningful on the three objects.

The object 19 has the most misapprehend results because it has choices from one to twenty seven segments, despite that fact, it has the same number of choices for the results produced by the user choices as well as the results produced by *CaS*, that is, seven choices, but if we add the any choice with the *CaS*, we get a total of thirteen choices.

On the other hand, we have some objects where the *CaS* results overcome the users results. The most significant object and where this preference obvious is the object 15 where just one user chose his segmentation result in opposition to the rest of the group that has chosen any or *CaS* option.

With this analysis it is possible to conclude that the users where satisfied with the overall results for the entire collection, so even though some objects show results which are not the most meaningful to the user, the majority of the objects have meaningful results to the user.

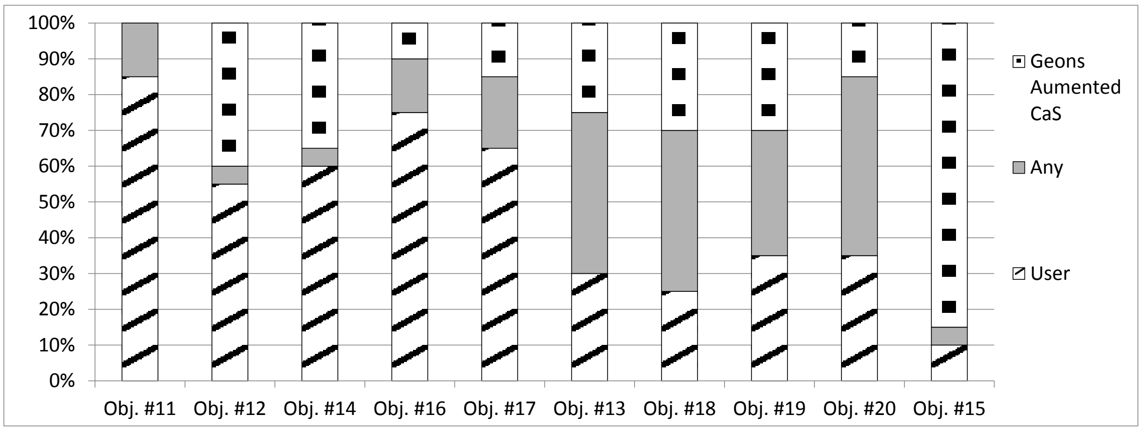


Figure E.4: Results of comparing *Geons-augmented CaS* with users segmentation