Eurographics Workshop on 3D Object Retrieval (2012)
M. Spagnuolo, M. Bronstein, A. Bronstein, and A. Ferreira (Editors)

Key-component Detection on 3D Meshes using Local Features

Ivan Sipiran[1] and Benjamin Bustos[1]

[1]KDW-PRISMA Research Group
Department of Computer Science, University of Chile

Abstract

In this paper, we present a method to detect stable components on 3D meshes. A component is a region on the mesh which contains discriminative local features. Our goal is to represent a 3D mesh with a set of regions, which we called key-components, that characterize the represented object and therefore, they could be used for effective matching and recognition. As key-components are features in coarse scales, they are less sensitive to mesh deformations such as noise. In addition, the number of key-components is low compared to other local representations such as keypoints, allowing us to use them in efficient subsequent tasks. An desirable characteristic of a decomposition is that the components should be repeatable regardless shape transformations. We show in the experiments that the key-components are repeatable under several transformations using the SHREC'2010 feature detection benchmark.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms , languages, and systems

1. Introduction

Three-dimensional information is becoming a useful resource in computer vision applications. An important aspect of this kind of information is that it can represent an object in a more approximated way than using other media. In addition, with the recent introduction of low-cost 3D sensors such as Kinect, we can now have access to three-dimensional information in real-world applications. Thus, the integration of 3D data with visual information could be used in order to improve the effectiveness of high-level tasks.

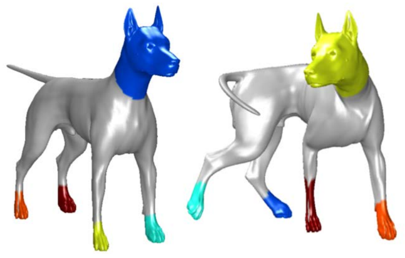

Figure 1: *Key-components detected on 3D meshes under a deformable transformation.*

It is clear that 3D data requires its own processing and analysis methods. Similarly to images, there is a need for basic tasks that provide a background for high-level tasks. Obviously, many problems arise due to the lack of a regular topology in 3D representations. In addition, the possible transformations that may occur differ from those present in images (for instance non-rigid transformation, topology changes, tessellations, among others). Therefore, three-dimensional data requires special attention as its related problems are not trivial.

A basic and important task is to find interesting structures in representations such as 3D point clouds or meshes. Many proposals have been presented to detect interest points (also called keypoints) on 3D data. Regarding meshes, an interest point is a point on the mesh with a local outstanding structure. As such, the keypoints represent interesting information at fine scales and thus, they could be sensitive to noise and other transformations. Therefore, it is required to find larger and interesting structures to overcome the problems at fine scales.

In this paper, we propose an algorithm to detect features at a coarse level on meshes. Our motivation is that larger structures are more resilient to local changes, while allowing us to reduce the amount of information to represent 3D

DOI: 10.2312/3DOR/3DOR12/025-032

meshes in retrieval and recognition tasks. The idea is to decompose a 3D mesh in a set of components, which should be consistently found in meshes regardless the transformation applied. In addition, the number of components should be much less than the number of keypoints, so using the components in subsequent tasks would be efficient.

We introduce the term *key-component* as a region on a 3D mesh where there are a lot of discriminative local features (see Fig. 1). Key-components will be useful to the extent that they are repeatable and robust against several transformations. Our method differs from mesh segmentation methods as it computes a non-complete decomposition of a mesh while is aware of the local features present in the components.

The main contribution of this paper is three-fold. First, we use a clustering algorithm in the mesh geodesic space in order to determine clusters of keypoints. These clusters are the starting point to compute the key-components. Second, we introduce a region growing algorithm which computes a key-component from a cluster and extracts the corresponding region on the mesh. Finally, we show a comprehensive evaluation of our approach in different scenarios. For the evaluation, we use a standard feature detection benchmark which contains shapes with several transformations.

The rest of the paper is organized as follows. Section 2 presents the related works regarding mesh decomposition and local features. Section 3 describes the local features detection and our algorithm for detecting the key-components. Section 4 shows the evaluation and discussion of the obtained results using the SHREC'10 feature detection and description benchmark. Finally, Section 5 concludes the paper.

2. Related Work

Mesh decomposition is a important analysis tool with applications in computer vision and graphics. The idea is to partition a given mesh in components or regions which can be used in applications. Although there are a lot of approaches for mesh segmentation, we are interested in those methods driven by local features. For a comprehensive study about mesh segmentation techniques, we recommend the survey by Shamir [Sha08].

One of the earliest techniques for feature-driven mesh decomposition was presented by Mortara et al. [MPS*03]. This method decomposes a triangular mesh based on a characterization of a vertex using its local curvature. It analyzes the evolution of the curve formed by the intersection of the mesh with a set of spheres with increasing radii. The number of connected components of the curve and the local properties (curvature and length ratio) define a classification for each vertex, which is used to group vertices with similar features. Differently, Huang et al. [HWAG09] proposed to decompose a shape based on a modal analysis. Taking the eigen-decomposition of the Hessian of a energy function defined on the mesh, it is possible to define the set of typical deformations of a mesh. Therefore, this method is able to estimate the part that tend to be rigid and subsequently segment them.

Local features have also been used for mesh segmentation. Agathos et al. [APPS09] propose a mesh segmentation method based on interest points. Given a mesh, the algorithm computes a protrusion function for each vertex, which is defined as the sum of geodesic distances to all vertex on the mesh. Thus, a vertex is selected as interest point if the value of its protrusion function is greater than the mean of geodesic distances between each pair of vertices. The interest points are grouped in order to avoid regions with many interest points. Each interest point is used as seed for computing the mesh segments. Similarly, Katz et al. [KLT05] computed a 3D embedding for a shape and subsequently, the convex hull of the embedding was calculated. The vertices of the convex hull were considered as keypoints, over which the method computed a set of core components.

On the other hand, Hu and Hua [HH09] proposed to find keypoints using the eigen-decomposition of the Laplace-Beltrami operator of a shape. Each keypoint has a scale which is used to define a local patch, so mesh is represented as a set of local patches product of the keypoint-based decomposition. After describing each local patch with its Laplace-Beltrami spectrum, they are used in a matching algorithm. On the other hand, Toldo et al. [TCF09] applied a segmentation based on local properties of the mesh, specifically a shape index computed from the principal curvature values. Each segment is described with a histogram of local properties. Finally, a bag of features approach is used for describe the entire shape in order to be used in shape retrieval. Differently, Shapira et al. [SSS*10] performed a hierarchical segmentation using a shape diameter function (SDF). Subsequently, each segment is described using several local features such as a normalized histogram of SDF, shape distribution signatures and conformal geometry signatures. The signatures were used in matching and retrieval.

Recently, a common approach in the shape analysis community is to extend methods from image processing and computer vision. For instance, Digne et al. [DMAMS10] extend the maximally stable extremal regions (MSER) to shape decomposition. The method used the concept of vertex-weighted component trees applied to meshes. To accomplish this goal, it was necessary to use the mean curvature as function defined over the mesh. Similarly, Litman et al. [LBB11] also used the MSER framework to detect stable components on meshes. The authors proposed an approach based on diffusion geometry. The algorithm considers the shape as a graph and associates weights to vertices and edges according to the evaluation of a local property (the heat kernel) between vertices and edges.

More recently, Fang et al. [FSKR11] introduced the perceptually consistent mesh segmentation. The authors proposed a vertex signature (the heat-mapping) as the average of the heat kernel evaluated on each point of the mesh. Then, a segmentation process is driven from the points with highest value of heat-mapping.

3. Key-component Detection

Our method consists of three steps: keypoint detection, clustering in the geodesic space, and key-component extraction. Our proposal is based on the detection of interest points, which can be effectively used for detecting stable components on meshes. In the literature, there are many techniques to detect keypoints, with different approaches and advantages. In this work, we use the Harris 3D method [SB11], which has proven to be effective and efficient in various scenarios.

In order to maintain the paper self-contained, we begin our description with a brief introduction to the Harris 3D method, and then we will describe how the keypoints are used to detect the mesh components.

3.1. Keypoints detection

Given a 3D mesh, we need to find interest points on it. In general terms, an interest point is a point on the mesh surface with a neighborhood geometrically unusual. For instance, many approaches link this definition with the curvature measured on the vertices of the mesh. So, points on nearly planar regions would not be considered as interesting. A keypoint detection method is robust if it works according with the previous criterion. However, it also needs to be insensitive to noise, tessellations, and missing data (holes or range data). The Harris 3D method is an extension of the well-known operator in computer vision.

There are several reasons to choose the Harris 3D method:

- It is effective. Recent reports have shown high repeatability values against several transformations [BBB*10, DCG11].
- It is efficient. An adequate implementation of this method can process meshes with 50,000 vertices in a fraction of a second.
- It is easy to implement. The method only requires simple operations over local mesh patches.

3.2. Clustering in the Geodesic Space

Key-components are those regions on the mesh in which there is a high concentration of local features. One way to measure the concentration is using the geodesic distances between the keypoints, and therefore grouping them according to their closeness in terms of this kind of distance. Let $S = \{s_1, s_2, \dots, s_n\}$ be the set of keypoints previously detected, our goal is to find partitions $S_i \subset S$, $i = 1 \dots m$ in order to fulfill the following properties:

1. $d_{geod}(x,y) \leq T_o$, $\forall x,y \in S_i$.
2. $d_{geod}(x,y) \geq T_p$, $\forall x \in S_i$ and $\forall y \in S_j$, $i \neq j$.
3. $\bigcup_{i=1}^{m} S_i \subseteq S$.
4. $S_i \bigcap S_j = \emptyset$, $i \neq j$.

Property 1 suggests that elements in a subset S_i share approximately the same location on the mesh (threshold T_o controls the proximity permitted). Property 2 states that two subsets S_i and S_j cannot be very close to each other (threshold T_p controls how far two subset should be). Property 3 considers a non-complete partitioning of the initial set S. Obviously, there can be keypoints which do not fulfill the two first properties. This is because some interest points could be isolated, and therefore they would not belong to any component. Moreover, isolated keypoints could have been selected due to noise. It is clear that, in order to detect consistent components on meshes, we need to discard isolated keypoints. Finally, property 4 defines a disjoint partition of the set S.

In practice, we need to consider a clustering process regarding the geodesic distances between keypoints. In order to accomplish this goal, our method computes a set $P \in \mathbb{R}^2$, in which euclidean distances between elements in P preserve the geodesic distances between elements in S. That is, we need to find the set P such that

$$P = \underset{p_1,\dots,p_n}{\operatorname{argmin}} \sum_{i<j} (\|p_i - p_j\| - d_{geod}(s_i, s_j)) \tag{1}$$

where each $p_i \in \mathbb{R}^2$ corresponds to the keypoint $s_i \in S$.

This problem is commonly called Multidimensional Scaling [BG05] and it is used to embed points in one space into another (generally for better visualization). The optimization problem in the Eq. 1 can be solved with an iterative method which takes a random sampling in the destination space as starting set P. The method used in this work was the SMACOF algorithm. In addition, for approximating the geodesic distances, we used the Dijkstra algorithm considering the mesh as a graph. Figure 2 shows the resulting set of 2D points applied on a set of keypoints. Note how the resulting points represent the distribution of keypoints on the mesh.

Next, we apply a clustering algorithm over the set P in order to define the partitioning of S. We proposed a clustering algorithm derived from Leow and Li [LL04] (See algorithm 1). In the algorithm 1, in addition to the properties 1 and 2, we also introduce a constraint regarding the number of elements that a cluster may have. So our algorithm ensures partitions with a minimum number of elements. Partitions with a few elements are discarded because they could generate components with low significance in the mesh. Figure 3 shows the groups of keypoints found using our algorithm.

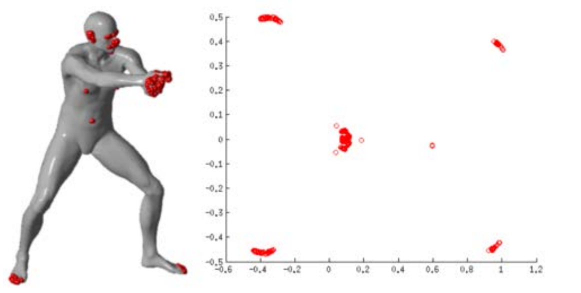

Figure 2: *Left: Shape with keypoints. Right: Multi-dimensional scaling of the keypoints.*

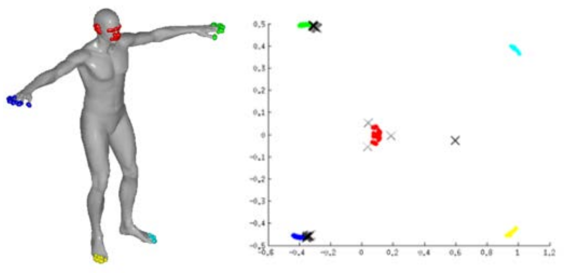

Figure 3: *Left: Shape with cluster of keypoints. Right: multi-dimensional scaling of the keypoints. Points represented as crosses do not belong to any cluster.*

Algorithm 1 Adaptive Clustering

```
Require: Set of points $P$
Require: Inter-cluster distance $T_p$
Require: Intra-cluster distance $T_o$
Require: Minimum number of elements per cluster $Nm$
Require: Number of iterations $Iter$
Ensure: Set of clusters $C = \{C_1, \dots, C_m\}$

1: Let $C$ a set of clusters
2: $C \leftarrow \emptyset$
3: for $j \leftarrow 1$ to $Iter$ do
4:     for each $p \in P$ do
5:         $mindist \leftarrow min_{i\in[1,|C|]} \|p - centroid(C_i)\|$
6:         if $mindist > T_p$ then
7:             $C_{new} = \{p\}$
8:             $C \leftarrow C \cup C_{new}$
9:             $K \leftarrow K - \{p\}$
10:        else if $mindist \leq T_o$ then
11:            $C_i \leftarrow C_i \cup \{p\}$
12:            $P \leftarrow P - \{p\}$
13:        end if
14:    end for
15:    for $i \leftarrow 1$ to $|C|$ do
16:        if $|C_i| \geq Nm$ then
17:            Update centroid for $C_i$
18:        else
19:            $P \leftarrow P \cup C_i$
20:        end if
21:    end for
22: end for
23: Return $C$
```

3.3. Key-component Extraction

The starting point to extract mesh components is the set of clusters previously computed. Each cluster will generate a component comprising the region of the mesh where the keypoints are located. Now, we need a criterion to decide how large this region will be. In addition, the selected region should be large enough to include all the keypoints in the cluster.

We start by defining the geodesic center of each cluster. The idea is to determine the point on the mesh which is the center of the distribution of a cluster. This point could be used as the center of the region to be extracted as component. We can take advantage of the transformed set of points P in order to accomplish this goal. The geodesic center of a cluster is a point on the mesh whose mapped version in $\mathbb{R}^2$ is near to the centroid of the cluster of the transformed points. To solve this, we choose the closer point to the centroid in $\mathbb{R}^2$ as the geodesic center. Note that the selected point is only an approximation of the real geodesic center, as our method is selecting a keypoint (finding the real geodesic center is a hard task as we would have had to map every point on the mesh into the $2D$ space, which is impossible in practical terms). Formally, let P_i be the set of 2D points corresponding to the set S_i of keypoints. The geodesic center of S_i is defined as follows:

$$c_i = \{s_j \in S_i | p_j = \underset{p\in P_i}{\operatorname{argmin}} \|p - centroid(P_i)\|\} \quad (2)$$

where $p_j \in \mathbb{R}^2$ corresponds to $s_j \in S$.

Now, we need to define a size for the component. To do that, our method computes the smallest sphere containing every keypoint in a cluster. This is a classic problem in computational geometry and it can be efficiently solved using linear programming. The output of this tasks is a pair (o_i, r_i) representing the center and the radius of the sphere enclosing the keypoint in the cluster S_i.

Once the geodesic center c_i and the sphere (o_i, r_i) have been computed, we propose a region growing algorithm on the mesh. Our initial seed is the vertex c_i and the constraint for the growing step is imposed by the sphere. The algorithm 2 details this procedure. It is worth noting that we introduce a scaling factor $\sigma > 1$ for the radius r_i. Thus, we ensure a connectivity between the keypoints in S_i. A value greater than 1 would guarantee to find a connected component lying inside the sphere with radius $\sigma \times r_i$.

Algorithm 2 Key-component Extraction

```
Require: Vertex set $V$
Require: Geodesic center $c_i$
Require: Cluster of keypoints $S_i$
Require: Sphere $(o_i, r_i)$
Require: Scaling radius factor $\sigma$
Ensure: Vertex set $V_R$
Ensure: Face set $F_R$

1: Let $V_R$ be an empty vertex set
2: Let $F_R$ be an empty face set
3: Let $waiting$ be the set of remaining keypoints
4: Let $visited$ be a vertex queue
5: $visited.enqueue(c_i)$
6: $waiting \leftarrow S_i$
7: while $waiting \neq \emptyset$ and $visited \neq \emptyset$ do
8:     $v \leftarrow visited.dequeue()$
9:     if $v$ is not marked then
10:        $V_R \leftarrow V_R \cup \{v\}$
11:        Mark $v$
12:        $waiting \leftarrow waiting - \{v\}$
13:        for each $w \in v.adjacentVertices()$ do
14:            if $w$ is not marked then
15:                if $\|w - o_i\| < \sigma \times r_i$ then
16:                    $visited.enqueue(w)$
17:                end if
18:            end if
19:        end for
20:        $F_R \leftarrow F_R \cup v.adjacentFaces()$
21:    end if
22: end while
23: Unmark vertices
24: Return $F_V$ and $F_R$
```

Briefly, the region growing algorithm starts from the geodesic center c_i and inserts the neighbor vertices into the queue. Each time a vertex is extracted from the queue, the algorithm verifies if the vertices is a keypoint. If so, the keypoint is deleted from the remaining set. The algorithm finishes when the remaining set is empty, which means that a component has been extracted and it contains the complete set of input keypoints.

Figure 4 shows the components detected in several shapes.

4. Experiments and Results

In this section, we describe the dataset, the evaluation criterion used to assess our method, and the experimental results.

4.1. Dataset

In order to evaluate the proposed method, we used the SHREC'10 feature detection and description benchmark [BBB*10]. This dataset is composed by three shapes (null shapes) and a set of shapes obtained by applying a set of transformations on the null shapes. Shapes have approximately 10,000 to 50,000 vertices and they were represented as triangular meshes. The set of transformations applied on the null shapes are isometry, micro-holes and big holes, topology, noise and shot noise, global and local scale, and downsampling. Each transformation was applied in five levels, so the total number of shapes in the dataset is 138.

In addition to the shapes, the dataset contains a ground-truth specifying the vertex-to-vertex correspondences between the transformed and the null shapes. Our method was not evaluated on meshes with big holes because it was not possible to compute the geodesic distances in that models. Also, the remaining models were normalized so the surface area is 1. This facilitated the configuration of the parameters of clustering.

4.2. Evaluation Criterion

Our goal is to determine if the mesh components are consistent between a null shape and a transformed shape. Given a null shape X and a transformed mesh Y, the components are represented as $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$, respectively. Using the ground-truth, we compute the corresponding component to each component Y_j in X, which is denoted as X'_j. Then, the component repeatability between X and Y is defined as

$$R(X,Y) = \sum_{j=1}^{m} \max_{1 \leq i \leq n} O(X_i, X'_j) \tag{3}$$

where the overlap between two components is defined as an area ratio

$$O(X_i, X'_j) = \frac{A(X_i \bigcap X'_j)}{A(X_i \bigcup X'_j)}. \tag{4}$$

In addition, we define the repeatability in overlap o as the percentage of components in the entire collection that have overlap greater than o with their corresponding null shape. Clearly, totally coincident components give a repeatability of 1.

4.3. Results

In this section, we present the results obtained with our approach. For the keypoint detection, we select the 500 keypoints with the higher Harris response in each shape. In all our experiments, the scaling factor σ was set to 1.75. Furthermore, the parameter $Iter$ in the clustering algorithm was

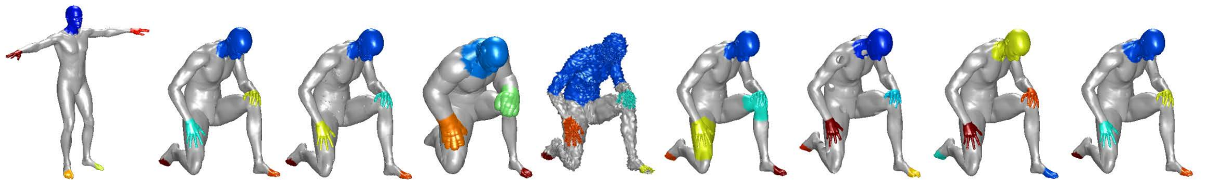

Figure 4: *Key-components detected on shapes with several transformations. From left to right: null shape, isometry, microholes, local scale, noise, topology, holes, sampling, and shot-noise.*

Variant	T_o	T_p	Nm
KC-1	0.1	0.2	30
KC-2	0.05	0.1	30
KC-3	0.1	0.2	10
KC-4	0.05	0.1	10

Table 1: *Clustering parameters for each experiment configuration.*

	Variant			
Transform.	**KC-1**	**KC-2**	**KC-3**	**KC-4**
Isometry	0.84	0.81	0.62	0.55
Topology	0.48	0.62	0.56	0.40
Micro holes	0.73	0.80	0.55	0.50
Scale	0.81	0.79	0.69	0.50
Local scale	0.82	0.73	0.62	0.43
Sampling	0.00	0.00	0.04	0.03
Noise	0.80	0.71	0.74	0.49
Shot noise	0.75	0.75	0.71	0.45
Holes	0.39	0.26	0.42	0.29
Average	0.62	0.61	0.55	0.40

Table 2: *Repeatability values at overlap 0.8 for each variant.*

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.58	0.92	0.92	0.79	0.82
Topology	0.83	0.71	0.78	0.53	0.63
Micro holes	0.76	0.88	0.85	0.84	0.85
Scale	0.87	0.88	0.85	0.92	0.76
Local scale	0.94	0.91	0.92	0.79	0.79
Sampling	0.56	0.33	0.20	0.14	0.00
Noise	0.89	0.82	0.77	0.79	0.79
Shot noise	0.80	0.77	0.78	0.78	0.85
Holes	0.56	0.60	0.60	0.50	0.49
Average	0.76	0.76	0.74	0.68	0.67

Table 3: *Repeatability values for variant KC-1.*

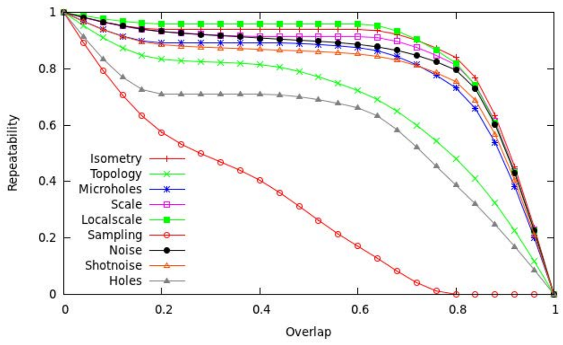

Figure 5: *Overlap vs. Repeatability plot for the KC-1 variant.*

set to 10. Furthermore, we used four different settings for the clustering algorithm. Table 1 shows the parameters used in each configuration. In addition, we use the repeatability at overlap 0.8 to compare the variants (see Table 2).

For the variant KC-1, Fig. 5 and Table 3 show the results. Interestingly, almost all transformations maintain a high repeatability (greater than 80%) at overlap values < 0.8. It is not the case for sampling, holes, and topology transformation. In the sampling transformation, the problem is that we always select 500 keypoints. In addition, as the number of vertices is decreased, the keypoints are distributed over the entire shape and they do not tend to cluster. Differently, in the topology transformation, its repeatability systematically decreases because the clustering relies on geodesic distances. Therefore, as the topological changes alter the geodesic distances on the mesh, the clustering and the subsequent key-component extraction is affected. It is worth noting that our method detects repeatable components in presence of difficult transformations such as noise, shot-noise and micro-holes.

Table 3 shows the repeatability values regarding the five levels of transformations. Note that our method is resilient to high levels of transformations. The only transformation which is very sensitive to the level of transformation is sampling due to the aforementioned reasons. This behavior can be seen in all the variants.

For the variant KC-2, Fig. 6 and Table 4 shows the results. Note that almost all transformations have a perfect repeata-

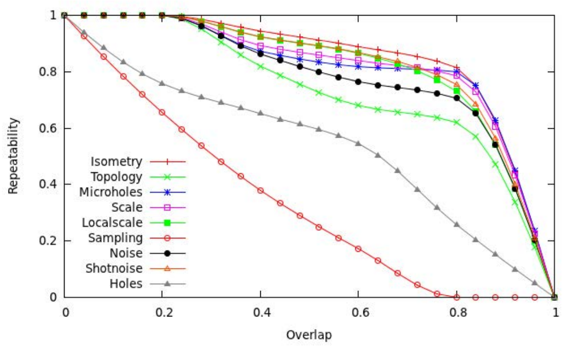

Figure 6: *Overlap vs. Repeatability plot for the KC-2 variant.*

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.66	0.84	0.85	0.86	0.90
Topology	0.77	0.76	0.74	0.71	0.74
Micro holes	0.77	0.77	0.74	0.76	0.76
Scale	0.87	0.82	0.84	0.78	0.79
Local scale	0.82	0.81	0.77	0.74	0.76
Sampling	0.60	0.49	0.24	0.13	0.02
Noise	0.86	0.79	0.84	0.85	0.84
Shot noise	0.85	0.85	0.83	0.86	0.74
Holes	0.68	0.71	0.71	0.66	0.65
Average	0.76	0.76	0.73	0.71	0.69

Table 5: *Repeatability values for variant KC-3.*

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.58	0.83	0.86	0.95	0.85
Topology	0.74	0.76	0.75	0.79	0.86
Micro holes	0.89	0.85	0.90	0.89	0.90
Scale	0.91	0.92	0.93	0.81	0.83
Local scale	0.84	0.87	0.82	0.93	0.75
Sampling	0.53	0.31	0.21	0.03	0.00
Noise	0.87	0.83	0.71	0.73	0.72
Shot noise	0.84	0.93	0.85	0.89	0.73
Holes	0.50	0.58	0.48	0.44	0.41
Average	0.75	0.76	0.72	0.72	0.67

Table 4: *Repeatability values for variant KC-2.*

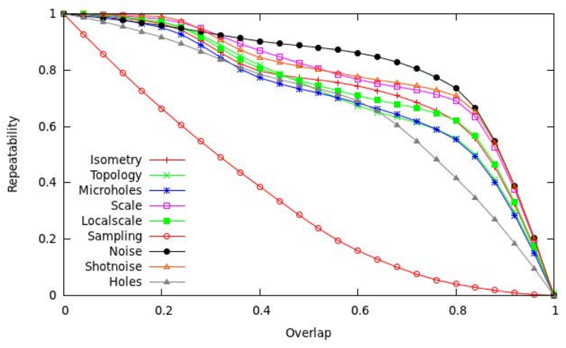

Figure 7: *Overlap vs. Repeatability plot for the KC-3 variant.*

bility at overlap < 0.2. It means that every component in the dataset overlaps a component in the null shapes. However, the repeatability decreases at higher overlap values, falling below 80% at overlap 0.8 in every transformation.

Variants KC-1 and KC-2 compute components with a high number of keypoints. They differ in the size of the detected components. The clustering parameters T_o and T_p define the extent in which the keypoints are grouped, so KC-1 detects larger components than KC-2. It is interesting to note that larger components are more repeatable and therefore, they could be more robust to mesh transformations.

For KC-3, Fig 7 and Table 5 show the results. In this case, the repeatability values at overlap 0.8 are below 80% in every transformation. In addition, there are some transformations (topology, micro-holes, local scale, and isometry) with repeatability below 60% at overlap 0.8.

For the variant KC-4, Fig. 8 and Table 6 show the results. Differently to the other variants, KC-4 shows low repeatability values at overlap 0.8. In every transformation, the repeatability is below 60% at overlap 0.8. Compared to KC-1 and KC-2, KC-3 and KC-4 compute components with a low number of keypoints, so it seems to harm the performance of these variants. In fact, many components could be detected with a few keypoints, and there is no guarantee that these keypoints are stable. Therefore, the components could also be unstable.

Note that KC-1 presents the best results. The parameters used for KC-1 allow us to detect large components with high number of keypoints compared to the other three variants. Obviously, larger components are more stable to transformations. In addition, the higher the number of keypoints, the higher the probability of having an interesting component. Clearly, variants KC-3 and KC-4 show a poor performance due to the low number of keypoints in each component.

5. Conclusions

We have presented a method to detect components on 3D meshes, which contain a high concentration of local features. The key-components are suitable for matching and recognition tasks due to their high repeatability obtained in our experiments using a standard benchmark. Interestingly, the proposed method detects consistent components under several transformations such as noise, local scale, holes, and non-rigid transformations. In our opinion, key-components

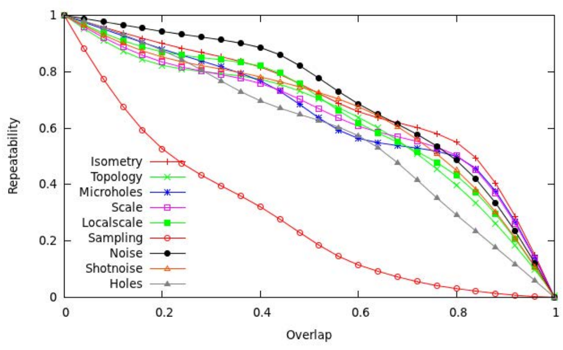

Figure 8: *Overlap vs. Repeatability plot for the KC-4 variant.*

	Strength				
Transform.	**1**	**≤2**	**≤3**	**≤4**	**≤5**
Isometry	0.53	0.78	0.80	0.81	0.73
Topology	0.67	0.66	0.69	0.71	0.73
Micro holes	0.74	0.72	0.75	0.73	0.74
Scale	0.72	0.72	0.73	0.73	0.75
Local scale	0.73	0.74	0.70	0.80	0.69
Sampling	0.50	0.37	0.19	0.10	0.00
Noise	0.75	0.74	0.74	0.79	0.75
Shot noise	0.72	0.74	0.70	0.73	0.73
Holes	0.60	0.62	0.64	0.57	0.58
Average	0.66	0.68	0.66	0.66	0.63

Table 6: *Repeatability values for variant KC-4.*

represent an alternative to fine scale features. On the one hand, we showed that key-components are stable to local transformations. On the other hand, the number of key-components is obviously much less than the number of key-points, so matching algorithms using local features could benefit from our approach.

In the future, we plan to improve the key-component extraction step into a completely automatic process. In addition, description algorithms are needed in order to effectively use the components in subsequent tasks.

Acknowledgments

This project has been partially funded by CONICYT (Chile) through the Doctoral Scholarship, and FONDECYT (Chile) Project 1110111.

References

[APPS09] Agathos A., Pratikakis I., Perantonis S., Sapidis N. S.: Protrusion-oriented 3D mesh segmentation. *The Visual Computer 26*, 1 (Aug. 2009), 63–81. 2

[BBB*10] Bronstein A. M., Bronstein M. M., Bustos B., Castellani U., Crisani M., Falcidieno B., Guibas L. J., Kokkinos I., Murino V., Sipiran I., Ovsjanikov M., Patanè G., Spagnuolo M., Sun J.: SHREC 2010: Robust feature detection and description benchmark. In *Proc. Workshop on 3D Object Retrieval (3DOR'10)* (2010), Eurographics Association. 3, 5

[BG05] Borg I., Groenen P.: *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005. 3

[DCG11] Dutagaci H., Cheung C., Godil A.: Evaluation of 3D interest point detection techniques. In *Proc. Eurographics 2011 Workshop on 3D Object Retrieval (3DOR'11)* (2011), Eurographics Association. 3

[DMAMS10] Digne J., Morel J.-M., Audfray N., Mehdi-Souzani C.: The level set tree on meshes. In *Proc. of the Fifth Int. Symposium on 3D Data Processing, Visualization and Transmission* (Paris, France, May 2010). 2

[FSKR11] Fang Y., Sun M., Kim M., Ramani K.: Heat-Mapping: A robust approach toward perceptually consistent mesh segmentation. *IEEE Computer Vision and Pattern Recognition* (2011). 2

[HH09] Hu J., Hua J.: Salient spectral geometric features for shape matching and retrieval. *Vis. Comput. 25*, 5-7 (2009), 667–675. 2

[HWAG09] Huang Q.-X., Wicke M., Adams B., Guibas L.: Shape decomposition using modal analysis. *Computer Graphics Forum 28*, 2 (2009), 407–416. 2

[KLT05] Katz S., Leifman G., Tal A.: Mesh segmentation using feature point and core extraction. *Visual Computer 21*, 8 (2005), 649–658. 2

[LBB11] Litman R., Bronstein A. M., Bronstein M. M.: Diffusion-geometric maximally stable component detection in deformable shapes. *Computers & Graphics 35*, 3 (2011), 549–560. Shape Modeling International (SMI) Conference 2011. 2

[LL04] Leow W. K., Li R.: The analysis and applications of adaptive-binning color histograms. *Comput. Vis. Image Underst. 94* (April 2004), 67–91. 4

[MPS*03] Mortara M., Patan G., Spagnuolo M., Falcidieno B., Rossignac J.: Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica 38* (October 2003), 227–248. 2

[SB11] Sipiran I., Bustos B.: Harris 3D: a robust extension of the harris operator for interest point detection on 3D meshes. *The Visual Computer 27* (2011), 963–976. 3

[Sha08] Shamir A.: A survey on mesh segmentation techniques. *Comput. Graph. Forum 27*, 6 (2008), 1539–1556. 2

[SSS*10] Shapira L., Shalom S., Shamir A., Cohen-Or D., Zhang H.: Contextual Part Analogies in 3D Objects. *Int. Journal of Computer Vision 89*, 2-3 (2010), 309–326. 2

[TCF09] Toldo R., Castellani U., Fusiello A.: Visual vocabulary signature for 3D object retrieval and partial matching. In *Proc. Workshop on 3D Object Retr. (3DOR)* (2009), Eurographics Association, pp. 21–28. 2