Vis Comput (2013) 29:1319–1332
DOI 10.1007/s00371-013-0870-9

ORIGINAL ARTICLE

Key-components: detection of salient regions on 3D meshes

Ivan Sipiran · Benjamin Bustos

Published online: 29 August 2013

Abstract In this paper, we present a method to detect stable components on 3D meshes. A component is a salient 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. A 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 and robust under several transformations using the SHREC'2010 feature detection benchmark. In addition, we discover the connection between the theory of saliency of visual parts from the cognitive science and the results obtained with our technique.

Keywords 3D features · Mesh decomposition

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.

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 points of interest (also called keypoints) on 3D data. Regarding meshes, a keypoint 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 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 applied transformation. In addition, the number of components should be

I. Sipiran (✉) · B. Bustos
PRISMA Research Group, Department of Computer Science, University of Chile, Santiago, Chile
e-mail: isipiran@dcc.uchile.cl

B. Bustos
e-mail: bebustos@dcc.uchile.cl

Fig. 1 Key-components detected on 3D meshes using our method

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). In such way, key-components correspond to regions with high protrusion and they are therefore distinctive for the object. Additionally, the size of the salient region is determined by a clustering algorithm used to find agglomerations of keypoints in a sense of geodesic closeness. Moreover, key-components will be useful to the extent that they are repeatable and robust against several transformations.

We believe that the detection of robust components is the next step in the search of reliable local structures for many tasks such as matching, retrieval, segmentation, and so on. The use of finer local structures (keypoints) has also proven to be effective for these application. Nevertheless, we are still facing the problem of the robustness of local features against perturbations. In our opinion, the components may provide robustness against transformations and they allow us to deal with some problems still present in the use of keypoints. On the other hand, we believe that the detection of key-components is a challenging problem and its utility has been shown recently in shape matching [23]. In our paper, we present a method to detect key-components and present enough evidence of their robustness.

Our method is inspired by the cognitive theory of saliency of visual parts [11]. This theory studied the important role of object parts in high-level vision tasks. In addition, it exposed the characteristics of parts in order to be considered as salient. To this respect, the theory formulated the existence of three key aspects for parts: the relative size, the protrusion and the strength of the boundary. We present a procedure to detect salient parts or regions on 3D meshes trying to cover the aforementioned aspects. More specifically, our method selects regions with agglomerations of keypoints as key-components, so they are expected to have a high protrusion. Additionally, our results confirm the fact that the size is important to define robust salient regions.

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. Even more, if we would like to establish a comparison with image processing tasks, we would say that mesh segmentation is related to image segmentation while our method is more similar to image saliency detection [10]. In other words, we are interested in detecting portions of the mesh which are distinctive enough and robust and repeatable against transformations rather than decomposing the whole mesh.

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. Furthermore, we compare our method with a variation of the MSER components detection technique [20] which make use of the diffusion geometry to detect overlapping components on meshes.

A preliminary version of our method has been presented in a conference paper [28]. In this extended version, we improve the presentation of our method, giving detailed explanations for its implementation and maintaining it self-contained. Furthermore, we made a comprehensive experimentation regarding the parameters involved in the detection of key-components. We present a sensitivity analysis to determine the best parameter configuration. Finally, we provide a comparison of our method with an existing method and show that the key-components outperform the state of the art.

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. Also, we propose a variation of a state-

of-the-art method in order to compare it with our method. Finally, Sect. 5 concludes the paper.

2 Related work

Mesh decomposition is an 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 [24].

One of the earliest techniques for feature-driven mesh decomposition was presented by Mortara et al. [22]. 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. [13] proposed to decompose a shape based on a modal analysis. Taking the eigen-decomposition of the Hessian of an 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 parts that tend to be rigid and subsequently segment them.

Local features have also been used for mesh segmentation. Agathos et al. [1] propose a mesh segmentation method based on points of interest. 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 point of interest if the value of its protrusion function is greater than the mean of geodesic distances between each pair of vertices. The points of interest are grouped in order to avoid regions with many points of interest. Each point of interest is used as seed for computing the mesh segments. Similarly, Katz et al. [14] 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.

Also, Gal and Cohen-Or [9] proposed to represent a 3D object as a set of salient geometric features. Their scheme entirely relies on curvature information over the shape's surface. This method starts by computing the curvature on a set of sampled points. Next, points are sorted according to their curvature values. The algorithm takes points with high curvature and performs a grouping of neighbor points until a good quadratic fitting surface can be found that approximates better the neighborhood. Subsequently, a region-growing algorithm clusters the mesh by adding points to the initial segments according to an empirical measure which involves area, curvature and similarity. The final clusters are called salient geometric features which are used in shape matching.

On the other hand, Hu and Hua [12] 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 a 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. [32] 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. [25] 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.

More recently, the decomposition of meshes from spectral functions defined on the surface has been introduced. The idea of these techniques is to take advantage of intrinsic information to define a function on vertices, edges or faces. Subsequently, the defined function is used by a grouping algorithm which provides a segmentation. For instance, the Heat Mapping approach [8] defines a vertex signature which can be interpreted as the average temperature on the surface by applying heat on a vertex. Then, a segmentation process using the k-means algorithm is driven from points with the highest value of heat affinity. Similarly, the Center-Shift method [31] proposes the evaluation of the biharmonic kernel in a point as a vertex function. Next, the algorithm finds a set of termination points which are vertices with maximal weighted mean of the defined function evaluated in local neighborhoods. Finally, a segment refinement is applied to provide the final segmentation.

Likewise, Skraba et al. [29] proposed to assign to each vertex its heat kernel signature evaluated in a fixed time. With these values, the technique applies a persistence-based clustering. This clustering considers to track regions associated with local maxima of the function. On the other hand, Aubry et al. [2] computed an n-dimensional function for each vertex. Each vertex was represented by its wave kernel signature. Next, every descriptor from a training set of shapes is collected in a large n-dimensional point cloud. This points are clustered by a Gaussian mixture algorithm where points in the same cluster correspond to the same

mesh segment. This clusters are used for assigning a label to each vertex of a new shape.

In the same direction, the extension of methods from image processing and computer vision has been studied. For instance, Digne et al. [6] 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. [20] 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. A very interesting work that also uses concepts from image processing is the Variational Mesh Decomposition [33]. It is based on the well-studied Mumford–Shah functional which is common in image segmentation literature. This technique proposes a convex version of the functional which is applied on a face-based multichannel function on the surface. The function is defined from the eigenvectors of a Laplacian matrix defined on dihedral angles for edges.

Saliency on meshes Previous approaches for detecting saliency on 3D meshes are mainly focused on defining a saliency function on surface points [15, 17, 18, 26]. Nevertheless, in these approaches, it is not clear the connection between the point saliency and the determination of a region with boundary. In addition, there is no robustness evaluation of the proposed techniques against transformations. This is important in order to guarantee effectiveness in further processes and applications such as shape recognition and retrieval. Our paper attempts to cover these two aspects by presenting a technique that detects robust salient regions and making an extensive evaluation.

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 points of interest, 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 [27], 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 points of interest on it. In general terms, an point of interest 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 and it has proven to be an alternative to cope with the aforementioned problems.

Briefly, the idea is to represent the neighborhood around a vertex as an oriented image patch, and subsequently to apply the Harris operator in an effective way. First, the algorithm tries to find an adaptive neighborhood around a vertex using an algorithm robust to geometric changes. Second, the orientation of the local patch is found by applying PCA over the local neighborhood. This step converts the oriented image patch into a canonical representation invariant to orientation. Third, a quadratic surface is fitted to the transformed local patch. Fourth, the method computes the smoothed derivatives of the fitted surface by convolving it with Gaussian functions. Fifth, using the derivatives, the Harris response is calculated using an autocorrelation function. Finally, the responses are used to determine the final set of points of interest selecting those vertices with the higher responses.

The Harris 3D method works around the responses computed for each vertex. The Harris response characterizes the protrusion of a vertex with respect to its neighborhood. In addition, the use of Gaussian functions to compute the derivatives decreases the effect of local changes in the overall computation. For example, Fig. 2 plots the Harris response for some shapes. The selection of keypoints is based on the Harris response.

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

- It is effective. Recent reports have shown high repeatability values against several transformations [5, 7].
- 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

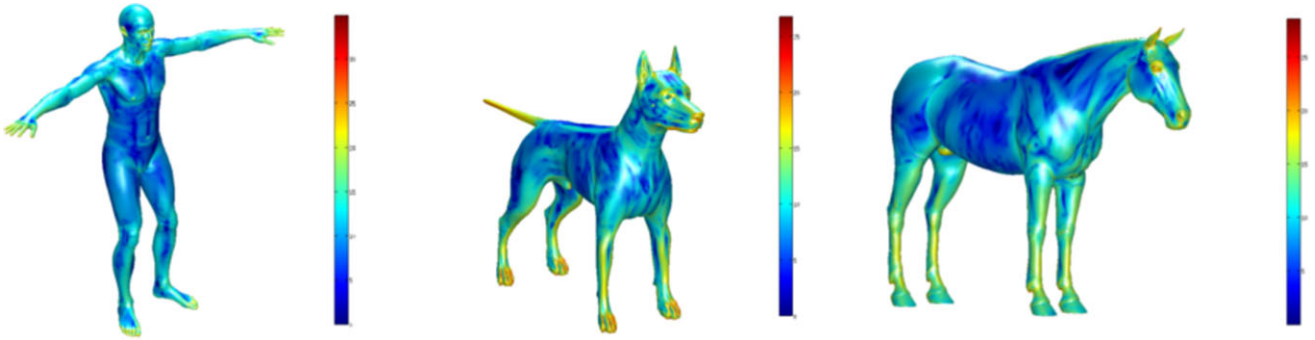

Fig. 2 Harris 3D operator plotted as a saliency value for each vertex. Note how the high values are present in discriminative regions of the meshes

between the keypoints, and therefore grouping them according to their closeness in terms of this kind of distance. Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of keypoints previously detected, our goal is to find partitions $S_i \subset S$, $i = 1, \ldots, m$ over the set of keypoints S in order to fulfill the following properties:

1. $d_{\text{geod}}(x, y) \leq R, \forall x, y \in S_i$.
2. $d_{\text{geod}}(x, y) \geq T, \forall x \in S_i$ and $\forall y \in S_j, i \neq j$.
3. $|S_i| \geq N, \forall i$.
4. $\bigcup_{i=1}^{m} S_i \subseteq S$.

Property 1 suggests that elements in a subset S_i share approximately the same location on the mesh (threshold R controls the proximity permitted). Property 2 states that two subsets S_i and S_j cannot be very close to each other (threshold T controls how far two subset should be). Property 3 establishes that each partition should contain a minimum number of keypoints to be considered as a valid partition. Property 4 considers a non-complete partitioning of the initial set S. Obviously, there may be keypoints which meet the two first properties, but not the third. This is because some points of interest could be isolated, and therefore they would not belong to any partition. 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 5 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 approximately preserve the geodesic distances between elements in S. That is, we need to find the set P such that

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

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

This problem is commonly called Multidimensional Scaling [3] and it is used to embed points in one space into another (generally for better visualization). The optimization problem in Eq. (1) is a minimum-distortion problem and 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 [3]. In addition, for approximating the geodesic distances, we used the fast marching method [16]. Figure 3 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.

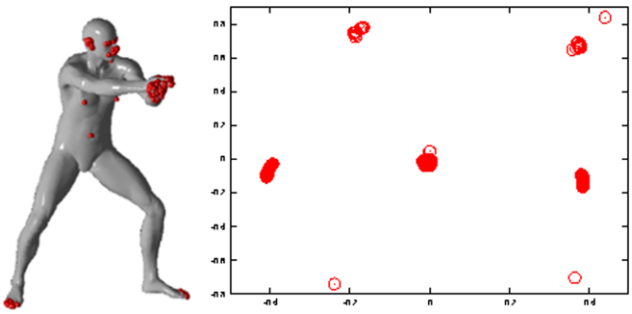

Fig. 3 *Left*: Shape with keypoints. *Right*: multidimensional scaling of the keypoints

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 [19] (see Algorithm 1). The algorithm iterates over two steps: assignment and update. The assignment step (lines 4–18) performs in point-wise manner. Firstly, the distance to the closest cluster is obtained. If the distance is greater than T (the inter-cluster threshold), we create a new cluster (according to the property 2). Otherwise, if the distance is not greater than R (the intra-cluster threshold), the point p is inserted in the corresponding cluster (according to property 1). After the assignment step, each point belongs to a cluster. Subsequently, the update step (lines 19–26) computes the new centroids for clusters that meet the property 3. Otherwise, clusters with a few points are removed, and their points are inserted back in P to be further processed. Note that the algorithm could

Algorithm 1 Adaptive clustering

Require: Set of points P
Require: Inter-cluster distance T
Require: Intra-cluster distance R
Require: Minimum number of elements per cluster N
Require: Number of iterations *Iter*
Ensure: Set of clusters $C = \{C_1, \ldots, C_m\}$

```
Let C a set of clusters
C ← ∅
for j ← 1 to Iter do
    for each p ∈ P do
        if C = ∅ then
            d = 2T
        else
            C* = arg min_{C_i ∈ C} dist(p, C_i)
            d = distance from p to C*
        end if
        if d > T then
            C_new = {p}
            C ← C ∪ C_new
            P ← P − {p}
        else if d ≤ R then
            C* ← C* ∪ {p}
            P ← P − {p}
        end if
    end for
    for i ← 1 to |C| do
        if |C*| ≥ N then
            Update centroid for C*
        else
            P ← P ∪ C*
            C ← C\{C*}
        end if
    end for
end for
Return C
```

converge before the last iteration, however, we opt for using a number of iterations as stop criterion. In all experiments presented in Sect. 4, we set $Iter = 10$. This value was set empirically from the observation that, on average, the clustering algorithm converges in 6–8 iterations.

Figure 4 shows the groups of keypoints found using our algorithm.

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.

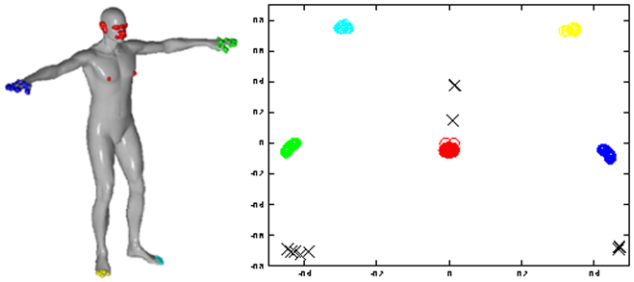

Fig. 4 *Left*: Shape with cluster of keypoints. *Right*: multidimensional scaling of the keypoints. Points represented as *crosses* do not belong to any cluster

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 close 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 C_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:

$$\hat{c}_i = \left\{ s_j \in S_i | c_j = \underset{c \in C_i}{\operatorname{argmin}} \left\| c - centroid(C_i) \right\| \right\} \tag{2}$$

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 [21]. 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 $\hat{c}_i$ 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 $\hat{c}_i$ and the constraint for the growing step is imposed by the sphere. Algorithm 2 details this procedure. Briefly, the region-growing algorithm starts from the geodesic center $\hat{c}_i$ and inserts the neighbor vertices into the queue. Each time a vertex is extracted from the queue, the algorithm verifies if the vertex is a keypoint. If so, the keypoint is deleted from the remaining set. The algorithm finishes when the remaining set is empty, which

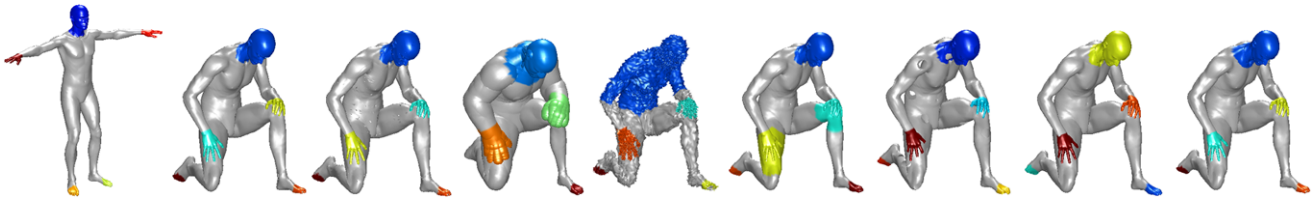

Fig. 5 Key-components detected on shapes with several transformations. From left to right: null shape, isometry, microholes, local scale, noise, topology, holes, sampling, and shot noise. Color are arbitrary

means that a component has been extracted and it contains the complete set of input keypoints.

It is worth noting that we introduce a scaling factor $\sigma > 1$ for the radius r_i (Algorithm 2, line 15). 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 $\sigma \times r_i$. That is, a suitable choice for σ should be in the interval $(1, 2]$. We avoid a value of one in our choice because the patch containing the cluster of keypoints could contain more than one connected component. Indeed, the larger the σ value, the higher the probability that the extracted region is a connected component. On the other hand, the σ value cannot be extremely large because it could affect the characterization of the cluster of keypoints. In all our experiments, we use $\sigma = 1.5$ which allow us to always obtain one connected component in the extraction without compromising the characterization of the keypoint clusters.

Figure 5 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 [5]. 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. Also, the models were normalized so the surface area is 1. This facilitated the configuration of the parameters of clustering.

Algorithm 2 Key-component extraction

Require: Vertex set V
Require: Geodesic center $\hat{c}_i$
Require: Cluster of keypoints S_i
Require: Sphere (o_i, r_i)
Require: Scaling radius factor σ
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: $visited.enqueue(\hat{c}_i)$
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

4.2 Evaluation criterion

To evaluate our approach, we use the methodology previously used in Litman et al. [20] to determine the repeatability of a decomposition. 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

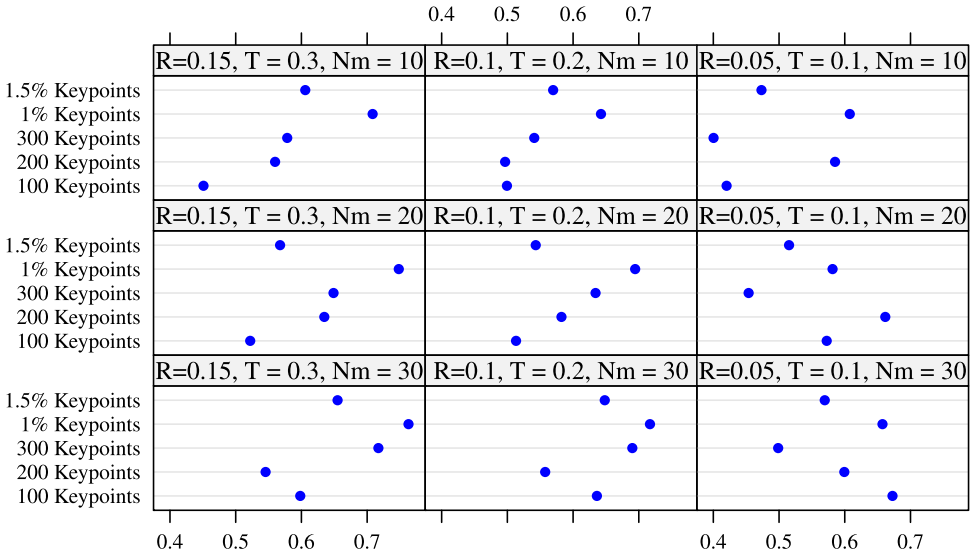

Fig. 6 This plot shows the average repeatability at overlap 0.8 for different parameter configurations. *Columns* represent region sizes: large (*left*), medium (*middle*), and small (*right*). *Rows* represent the minimum number of keypoints allowed in a cluster: $N = 10$ (*top*), $N = 20$ (*middle*), and $N = 30$ (*bottom*). Each block contains the repeatability for five different number of keypoints: three fixed configurations (100, 200, and 300) and two depending on the number of vertices (1 % and 1.5 %). Each block has the same scale

$Y_1, \ldots, Y_m$, 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

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

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

$$O\left(X_i, X'_j\right) = \frac{A(X_i \cap X'_j)}{A(X_i \cup X'_j)}. \tag{4}$$

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 Effect of key ingredients

Our method relies on two aspects which are important in the final resulting key-components. First, the number of keypoints could determine the protrusion of the regions and therefore their distinctiveness. Second, the clustering parameters could reveal the importance of the size in the repeatability of key-components. For these reasons, this section is devoted to study the effect of these two aspects in order to find a good parameter configuration.

Regarding the number of keypoints, we test five configurations: three with a fixed number of keypoints (100, 200, and 300) and two with a number that depends on the number of vertices (1 % and 1.5 %). Furthermore, we consider three configurations for the clustering which correspond to small ($R = 0.05, T = 0.1$), medium ($R = 0.1, T = 0.2$), and large regions ($R = 0.15, T = 0.3$). In addition, we evaluate the minimum number of keypoints needed for a key-component ($N = 10, 20, 30$). It is worth mentioning that values for clustering are empirical, mainly guided by the fact that meshes are normalized to area one. Figure 6 shows a plot with the results using all possible configuration as mentioned before. This plot shows the repeatability at overlap 0.8 as an average for every shape in the dataset (including transformations).

There are two aspects which deserve attention. First, there is a notorious predominance for the use of 1 % of the number of vertices as keypoints. This responds to the need of balancing the trade-off between quality and quantity of local features. In other words, much more keypoints could contain noisy information, and therefore it could degrade the quality of key-components. In counterpart, much less keypoints could not determine robust regions. This follows from the fact that regions are determined by groups of keypoints and obviously less keypoints could not tend to cluster. A particular case is shown for small regions and many keypoints per region ($R = 0.05, T = 0.1, N = 20$ and $R = 0.05, T = 0.1, N = 30$). In these cases, the use of few keypoints (200 and 100, respectively) shows the best repeatability. We believe that these two configurations exhibit a particular behavior because a few keypoints are grouped in few small regions. As these small regions correspond to the presence of many keypoints, they are distinctive and hence repeatable (although below the repeatability of large regions).

Second, the highest repeatability values correspond to large regions. That is, large regions are more stable to transformations and the probability that large key-components come from perturbed features is low. Moreover, among the different configurations for large regions, there is a trend regarding the expected number of keypoints per region. The

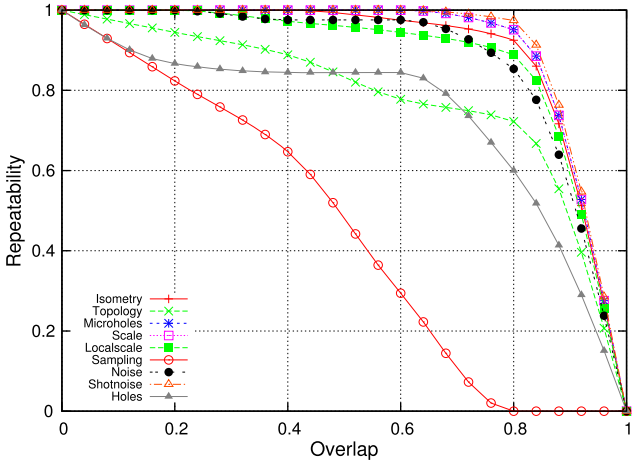

Fig. 7 Overlap vs. repeatability plot for the KC-1 variant

Table 1 Average overlap values for variant KC-1

Transform.	Strength				
	1	≤2	≤3	≤4	≤5
Isometry	0.94	0.95	0.85	0.93	0.94
Topology	0.75	0.70	0.84	0.77	0.74
Micro holes	0.93	0.95	0.95	0.95	0.93
Scale	0.95	0.95	0.92	0.93	0.95
Local scale	0.95	0.87	0.93	0.88	0.93
Sampling	0.62	0.38	0.09	0.02	0.00
Noise	0.92	0.91	0.93	0.94	0.86
Shot noise	0.93	0.95	0.96	0.96	0.94
Holes	0.78	0.69	0.77	0.86	0.79
Average	0.86	0.82	0.80	0.80	0.79

greater the number of keypoints allowed to belong to a region, the greater the repeatability of the key-components. This result encourages us to believe that key-components can correspond to regions with high protrusion which are distinctive and repeatable at the same time. This finding is consistent with the theory of saliency of visual parts in terms that key-components are distinctive. Furthermore, there is a visible relation between robustness and repeatability, and the relative size of regions.

To provide a closer look on the behavior of our algorithm against transformations, we chose the three configurations with the highest average repeatability at overlap 0.8 (in addition, we named each configuration to facilitate reading):

- **KC-1:** # keypoints = 1 % number of vertices, $R = 0.15$, $T = 0.3$, $N = 30$.
- **KC-2:** # keypoints = 1 % number of vertices, $R = 0.15$, $T = 0.3$, $N = 20$.
- **KC-3:** # keypoints = 1 % number of vertices, $R = 0.1$, $T = 0.2$, $N = 30$.

Variant KC-1 determines large regions with a high number of keypoints. Figure 7 plots the repeatability of key-components at several overlap values. Most transformations (shot noise, scale, microholes, isometry, local scale, and noise) obtained a high repeatability (greater than 80 %) at overlap 0.8. Even more, four transformations (shot noise, microholes, scale, and isometry) have a repeatability greater than 90 %. This indicates that this variant is very robust for these transformations regardless the transformation strength. Differently, topology, holes and sampling transformations obtained a repeatability below 80 %. With respect to the topology transformation, we believe that large key-components degrades the performance because they tend to include the introduced topological noise while the region is extracted. The larger the region, the more likely to merge two parts of the mesh that are not connected in the original mesh. In addition, our approach heavily depends on the computation of geodesic distances which are distorted with this transformation.

Also, a special case is the sampling transformation whose repeatability value drops to zero. In effect, the downsampling of meshes is not well handled by our approach due to the dependency of the number of vertices in the keypoint detection stage. In this case, the three variants use a number of keypoints which depends on the number of vertices (specifically 1 %). Obviously, when a shape is downsampled, the number of keypoints is also reduced. This fact could affect the entire process of key-components extraction, which depends on the number of keypoints and the distribution of them over the surface. In spite of this, we believe that the KC-1 variant of our technique is robust and it provides distinctive and repeatable key-components.

As well, Table 1 presents the average overlap for each transformations and its strengths. These results support those obtained in Fig. 7. The same four transformations with repeatability greater than 90 % in overlap 0.8 have high overlap values in most of the strengths.

For the variant KC-2, Fig. 8 and Table 2 show the results. The difference between KC-1 and KC-2 is minimal. Moreover, there are four transformations (microholes, scale, local scale, and noise) where overlap values are very similar (see Table 1 and Table 2).

With respect to the variant KC-3, almost all transformations present a drop in their repeatability values at overlap 0.8 with respect to the previous variants (see Fig. 9). Nevertheless, it is worth noting that the repeatability for the topology transformation rose above 90 %. It also can be seen in Table 3, where the overlap values for all the topology strengths were improved with respect to KC-1 and KC-2. We believe that this phenomenon correspond to the fact that medium-size regions are more stable to the topological noise. Furthermore, the overlap values for the highest strengths in local scale and shot noise were also improved.

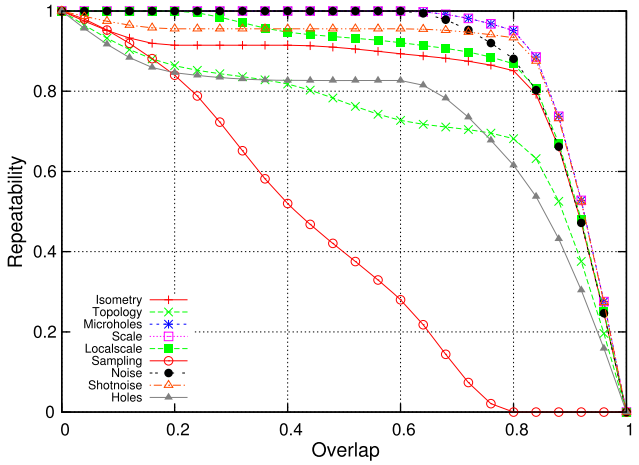

Fig. 8 Overlap vs. repeatability plot for the KC-2 variant

Table 2 Average overlap values for variant KC-2

Transform.	Strength				
	1	≤2	≤3	≤4	≤5
Isometry	0.85	0.95	0.78	0.87	0.87
Topology	0.74	0.64	0.78	0.71	0.67
Micro holes	0.93	0.95	0.95	0.95	0.93
Scale	0.95	0.95	0.92	0.93	0.95
Local scale	0.95	0.87	0.93	0.88	0.87
Sampling	0.58	0.41	0.24	0.02	0.00
Noise	0.92	0.91	0.93	0.94	0.91
Shot noise	0.85	0.87	0.95	0.95	0.93
Holes	0.68	0.75	0.70	0.81	0.74
Average	0.83	0.81	0.80	0.78	0.76

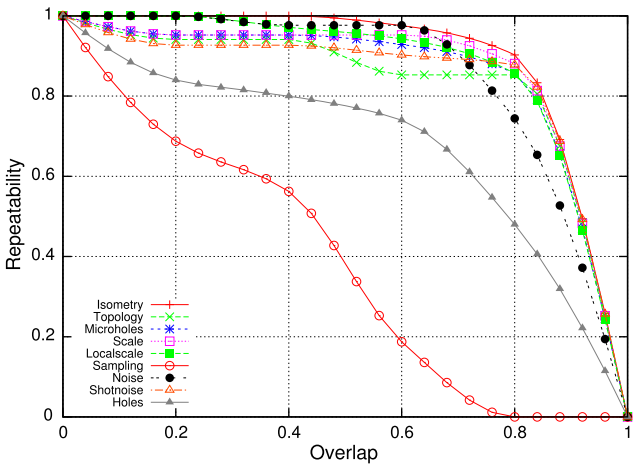

Fig. 9 Overlap vs. repeatability plot for the KC-3 variant

In order to compare the three variants KC-1, KC-2 and KC-3, Table 4 shows the winner configuration for each

Table 3 Average overlap values for variant KC-3

Transform.	Strength				
	1	≤2	≤3	≤4	≤5
Isometry	0.91	0.94	0.88	0.94	0.94
Topology	0.87	0.84	0.86	0.79	0.78
Micro holes	0.86	0.92	0.93	0.92	0.86
Scale	0.92	0.93	0.86	0.86	0.92
Local scale	0.95	0.87	0.87	0.91	0.94
Sampling	0.56	0.23	0.07	0.02	0.00
Noise	0.90	0.90	0.92	0.92	0.86
Shot noise	0.86	0.89	0.90	0.89	0.95
Holes	0.72	0.65	0.77	0.83	0.77
Average	0.84	0.80	0.79	0.79	0.78

Table 4 Comparison of the three evaluated variants: KC-1, KC-2, and KC-3

Transform.	Strength				
	1	≤2	≤3	≤4	≤5
Isometry	KC-1	KC-1	KC-3	KC-3	KC-1
Topology	KC-3	KC-3	KC-3	KC-3	KC-3
Micro holes	KC-1	KC-1	KC-1	KC-1	KC-1
Scale	KC-1	KC-1	KC-1	KC-1	KC-1
Local scale	KC-1	KC-1	KC-1	KC-3	KC-3
Sampling	KC-1	KC-2	KC-2	KC-1	KC-1
Noise	KC-1	KC-1	KC-1	KC-1	KC-2
Shot noise	KC-1	KC-1	KC-1	KC-1	KC-3
Holes	KC-1	KC-2	KC-1	KC-1	KC-1
Average	KC-1	KC-1	KC-1	KC-1	KC-1

transformation and their strengths in terms of overlap values. There is a clear predominance of KC-1 in most of the transformations, although KC-3 get the best results for topology and the highest level of local scale. Nevertheless, we believe that KC-1 is the best variant which confirms the robustness and repeatability of large key-components.

4.4 Comparison with other methods

So far, there are no methods that explicitly detect regions on 3D surfaces under the motivation of finding robust components based on local keypoints. Nevertheless, one method that has more to do with ours is that proposed by Litman et al. [4, 20]. This method proposed to detect stable components in deformable meshes using a diffusion geometry approach. The approach of Litman et al. decomposes a mesh into a set of (possibly overlapping) components using an approach similar to the MSER detection in computer vision. In addition, a component could be part of a larger component

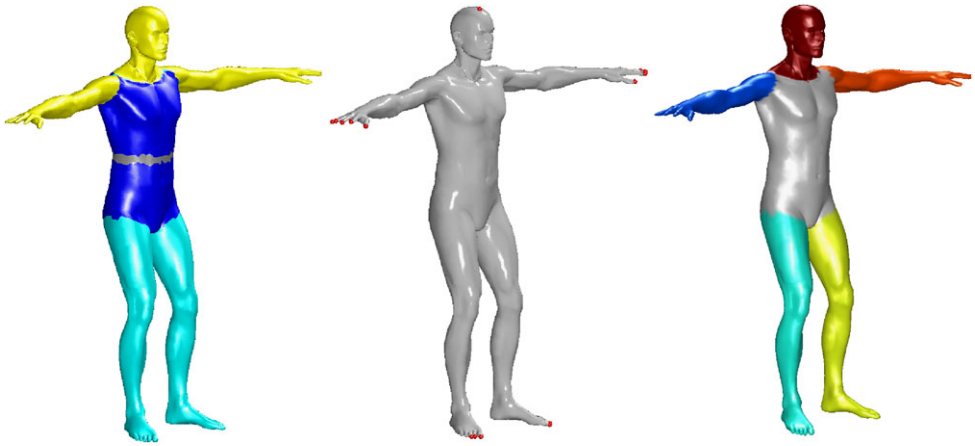

Fig. 10 The three stages of the MSER key-components detection. At *left*, components detected with the approach of Litman et al. In the *middle*, HKS keypoints detected. At *right*, final MSER key-components detected

Fig. 11 MSER 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. Colors are arbitrary

which entirely contains the first one. We propose a variation to detect key-components based on the components provided by the original method. We call this variant *MSER key-components*.

The algorithm we implement is simple and it is described as follows (Fig. 10 shows the three stages of our implementation):

- *Detecting MSER's.* We use the original implementation from Litman et al. [20] to detect a set of initial components.
- *Detecting keypoints.* For each mesh, we computed keypoint based on the heat kernel of a vertex. Our implementation follows the method proposed by Sun et al. [30]. Briefly, we evaluated the heat kernel signature for a vertex $HKS_t(x, x)$ in $t = 0.1 \times surface\_area$. Next, we selected a vertex x as keypoint if $HKS_t(x, x) > HKS_t(y, y), \forall y \in N_2(x)$, where $N_2(.)$ is the 2-ring neighborhood of a vertex.
- *Selecting the MSER key-components.* We could have chosen the set of components which contains the detected keypoints. However, since components can overlap, we need to apply one more constraint. In this case, we selected the components with smaller area which cover the entire set of keypoints. Figure 11 shows the MSER key-components detected using the proposed variant.

We experimented with the original MSER components and the results are shown in Fig. 12 and Table 5. Additionally, Fig. 13 and Table 6 show the obtained results for the MSER key-component method. Surprisingly, the variant obtained better results with respect to average overlap and repeatability. Therefore, we now compare our proposed method with the aforementioned variant. Note the improvement in holes and sampling transformation of MSER key-components with respect to our method. That is, while our method obtained null repeatability at overlap 0.8, MSER key-components obtained almost 0.6. Despite of the results obtained for these two transformations, our method outperforms MSER key-components in the rest of transformations.

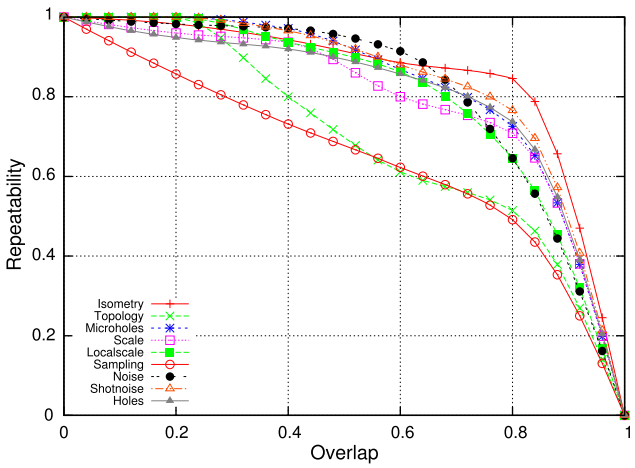

Fig. 12 Overlap vs. repeatability for the MSER method

Table 5 Average overlap values for the MSER approach

Transform.	Strength				
	1	≤2	≤3	≤4	≤5
Isometry	0.79	0.87	0.87	0.88	0.87
Topology	0.75	0.74	0.65	0.54	0.57
Micro holes	0.79	0.80	0.79	0.79	0.77
Scale	0.81	0.87	0.75	0.63	0.52
Local scale	0.75	0.72	0.74	0.65	0.56
Sampling	0.80	0.80	0.80	0.62	0.14
Noise	0.79	0.75	0.76	0.77	0.76
Shot noise	0.81	0.81	0.80	0.81	0.82
Holes	0.79	0.79	0.75	0.75	0.73
Average	0.79	0.79	0.77	0.71	0.64

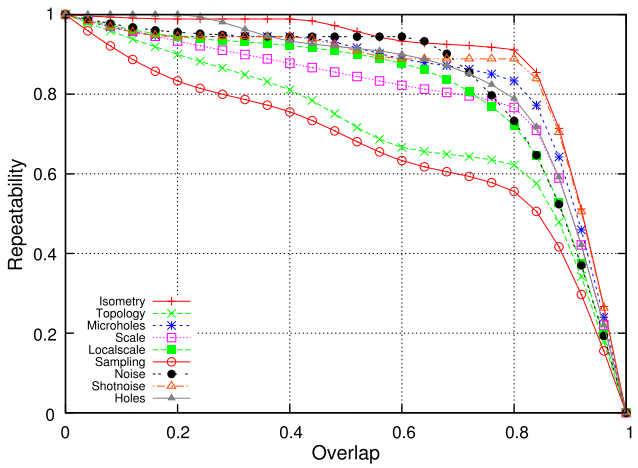

Fig. 13 Overlap vs. repeatability for the MSER key-component method

Moreover, this can also be seen if we compare our best configuration and the MSER key-components with respect to the overlap values (see Tables 1 and 6).

An aspect that also deserves attention is the need of preprocessing in the MSER key-components. The original method for detecting the initial components depends on the definition of edge or vertex weights. These weights are computed using diffusion geometry. This procedure requires to compute a similarity matrix in a vertex-wise manner, and subsequently compute an eigen-decomposition for that matrix. The problem with this approach is that if we have large models, the similarity matrix is very large and the eigen-problem could be unmanageable. The method proposed by Litman et al. suggested to simplify a model prior to the components detection step. In contrast, our method does not need that requirement and yet it is efficient. In this respect, the main advantage of our method is that once the keypoints have been detected, the subsequent tasks only work over these reduced number of information. We believe that the

Table 6 Average overlap values for the MSER key-components approach

Transform.	Strength				
	1	≤2	≤3	≤4	≤5
Isometry	0.83	0.88	0.90	0.87	0.86
Topology	0.77	0.72	0.67	0.52	0.55
Micro holes	0.82	0.81	0.80	0.79	0.78
Scale	0.76	0.84	0.80	0.67	0.58
Local scale	0.77	0.72	0.71	0.64	0.58
Sampling	0.83	0.83	0.80	0.62	0.12
Noise	0.79	0.79	0.83	0.83	0.79
Shot noise	0.83	0.83	0.80	0.80	0.80
Holes	0.79	0.77	0.66	0.63	0.62
Average	0.80	0.80	0.78	0.71	0.63

fact of relying on the initial keypoint detection step allows us to maintain the whole process efficient. This ability to deal with large meshes can be useful in applications where simplification is not an option since one could lose important details.

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 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 keypoints, so matching algorithms using local features could benefit from our approach.

Also, our experiments showed evidence of the connection between our method and the theory of saliency of visual parts proposed in the cognitive science. First, key-components correspond to regions with high protrusion since they are found from agglomerations of robust and distinctive local features. Moreover, the proposed clustering is responsible for ensuring that local features that belong to some perturbation (noise, holes, etc.) are not considered in the detection of salient regions. Second, there is an intimate relationship between the size of the key-components and the robustness against mesh transformations. Experiments showed that large salient regions (each with a large agglomeration of keypoints) are more repeatable. These two

aspects are consistent with the aforementioned theory, so our method can be thought of as a method embodying this theory.

Acknowledgements This project has been partially funded by CONICYT (Chile) through the Doctoral Scholarship, and FONDECYT (Chile) Project 1110111. We would like to thank Roee Litman for gently providing us the implementation of MSER components for our evaluation. Also, we thank Michael Bronstein for his extremely useful help with the SHREC'2010 feature detection and description benchmark.

References

1. Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N.S.: Protrusion-oriented 3D mesh segmentation. Vis. Comput. **26**(1), 63–81 (2009)
2. Aubry, M., Schlickewei, U., Cremers, D.: Pose-consistent 3D shape segmentation based on a quantum mechanical feature descriptor. In: Mester, R., Felsberg, M. (eds.) DAGM-Symposium. Lecture Notes in Computer Science, vol. 6835, pp. 122–131. Springer, Berlin (2011)
3. Borg, I., Groenen, P.: Modern Multidimensional Scaling: Theory and Applications. Springer, Berlin (2005)
4. Boyer, E., Bronstein, A., Bronstein, M., Bustos, B., Darom, T., Horaud, R., Hotz, I., Keller, Y., Keustermans, J., Kovnatsky, A., Litman, R., Reininghaus, J., Sipiran, I., Smeets, D., Suetens, P., Vandermeulen, D., Zaharescu, A., Zobel, V.: SHREC 2011: robust feature detection and description benchmark. In: Proc. Eurographics 2011 Workshop on 3D Object Retrieval (3DOR'11), pp. 71–78. Eurographics Association, Aire-la-Ville (2011)
5. 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), Eurographics Association, Aire-la-Ville (2010)
6. 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 (2010)
7. Dutagaci, H., Cheung, C., Godil, A.: Evaluation of 3D interest point detection techniques via human-generated ground truth. Vis. Comput. **28**, 901–917 (2012)
8. Fang, Y., Sun, M., Kim, M., Ramani, K.: Heat-mapping: a robust approach toward perceptually consistent mesh segmentation. In: IEEE Computer Vision and Pattern Recognition, pp. 2145–2152 (2011)
9. Gal, R., Cohen-Or, D.: Salient geometric features for partial shape matching and similarity. ACM Trans. Graph. **25**(1), 130–150 (2006)
10. Goferman, S., Zelnik-Manor, L., Tal, A.: Context-aware saliency detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 1915–1926 (2012)
11. Hoffman, D.D., Singh, M.: Salience of visual parts. Cognition **63**(1), 29–78 (1997)
12. Hu, J., Hua, J.: Salient spectral geometric features for shape matching and retrieval. Vis. Comput. **25**(5–7), 667–675 (2009)
13. Huang, Q.X., Wicke, M., Adams, B., Guibas, L.: Shape decomposition using modal analysis. Comput. Graph. Forum **28**(2), 407–416 (2009)
14. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. Vis. Comput. **21**(8), 649–658 (2005)
15. Kim, Y., Varshney, A., Jacobs, D.W., Guimbretière, F.: Mesh saliency and human eye fixations. ACM Trans. Appl. Percept. **7**(2), 12:1–12:13 (2010)
16. Kimmel, R., Sethian, J.A.: Computing geodesic paths on manifolds. Proc. Natl. Acad. Sci. USA **95**, 8431–8435 (1998)
17. Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. ACM Trans. Graph. **24**(3), 659–666 (2005)
18. Leifman, G., Shtrom, E., Tal, A.: Surface regions of interest for viewpoint selection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 414–421 (2012)
19. Leow, W.K., Li, R.: The analysis and applications of adaptive-binning color histograms. Comput. Vis. Image Underst. **94**, 67–91 (2004)
20. Litman, R., Bronstein, A.M., Bronstein, M.M.: Diffusion-geometric maximally stable component detection in deformable shapes. Comput. Graph. **35**(3), 549–560 (2011). Shape Modeling International (SMI) Conference 2011
21. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. In: Proceedings of the Eighth Annual Symposium on Computational Geometry, SCG '92, pp. 1–8. ACM, New York (1992)
22. Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica **38**, 227–248 (2003)
23. Pokrass, J., Bronstein, A.M., Bronstein, M.M., Sprechmann, P., Sapiro, G.: Sparse modeling of intrinsic correspondences. Comput. Graph. Forum **32**(2pt4), 459–468 (2013). doi:10.1111/cgf.12066
24. Shamir, A.: A survey on mesh segmentation techniques. Comput. Graph. Forum **27**(6), 1539–1556 (2008)
25. Shapira, L., Shalom, S., Shamir, A., Cohen-Or, D., Zhang, H.: Contextual part analogies in 3D objects. Int. J. Comput. Vis. **89**(2–3), 309–326 (2010)
26. Shilane, P., Funkhouser, T.: Distinctive regions of 3D surfaces. ACM Trans. Graph. **26**(2) (2007)
27. Sipiran, I., Bustos, B.: Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. Vis. Comput. **27**, 963–976 (2011)
28. Sipiran, I., Bustos, B.: Key-component detection on 3D meshes using local features. In: 3DOR, pp. 25–32 (2012)
29. Skraba, P., Ovsjanikov, M., Chazal, F., Guibas, L.: Persistence-based segmentation of deformable shapes. In: CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment (2010)
30. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Proceedings of the Symposium on Geometry Processing, SGP '09, pp. 1383–1392. Eurographics Association, Aire-la-Ville (2009)
31. Sun, M., Fang, Y., Ramani, K.: Center-shift: an approach towards automatic robust mesh segmentation (ARMS). In: CVPR, pp. 630–637. IEEE Press, New York (2012)
32. Toldo, R., Castellani, U., Fusiello, A.: Visual vocabulary signature for 3D object retrieval and partial matching. In: Proc. Workshop on 3D Object Retr. (3DOR), pp. 21–28. Eurographics Association, Aire-la-Ville (2009)
33. Zhang, J., Zheng, J., Wu, C., Cai, J.: Variational mesh decomposition. ACM Trans. Graph. **31**(3), 21:1–21:14 (2012)

Ivan Sipiran is a Ph.D. candidate at the Department of Computer Science, University of Chile. He is a research assistant of the PRISMA Research Group. His research interests include 3D object retrieval, geometry processing and computer vision.

Benjamin Bustos is an Assistant Professor at the Department of Computer Science, University of Chile. He is head of the PRISMA Research Group. He leads research projects in the domains of multimedia retrieval, multimedia databases, video copy detection, sketch-based image retrieval, and image processing. His research interests include similarity search, multimedia information retrieval, 3D object retrieval, (non)-metric indexing, and pattern recognition. Benjamin Bustos obtained doctoral degree in natural sciences from the University of Konstanz, Germany, in 2006.