Summary of the used method and technical manual of the application

1 Summary of the used method

In this section, we present a summary of the proposed method for the feature curve extraction on triangle meshes. This method is separated into two basic steps. At the first step, we estimate the saliency of each vertex using spectral analysis. The magnitude of the estimated saliency identifies if a vertex is feature or not. Based on the geometry, we can say that the feature vertices represent the edges of a feature curve (both crests and valleys) or corners. At the second step, we estimate the mean curvature of the extracted features and we use it to classify the different feature curves (if exist). Additionally, we use the information related to the mean curvature and the saliency of each feature curve in order to find similarities with feature curves of other models.

1.1 Notations and Basic Definitions of 3D Meshes

We assume that a triangle mesh $\mathcal{M}$ consists of n vertices $\mathbf{v}$ and n_f faces f. Each i vertex is represented by Cartesian coordinates, denoted by $\mathbf{v}_i = [x_i,\ y_i,\ z_i]^T,\ \forall\ i = 1, \cdots, n$. Each f_j face constitutes a triangle that can be represented by its centroid $\mathbf{c}_j = (\mathbf{v}_{j1} + \mathbf{v}_{j2} + \mathbf{v}_{j3})/3$ and its outward unit normal $\mathbf{n}_{c_i} = \frac{(\mathbf{v}_{j2}-\mathbf{v}_{j1})\times(\mathbf{v}_{j3}-\mathbf{v}_{i1})}{\|(\mathbf{v}_{j2}-\mathbf{v}_{j1})\times(\mathbf{v}_{j3}-\mathbf{v}_{j1})\|}$, where $\mathbf{v}_{j1}$, $\mathbf{v}_{j2}$ and $\mathbf{v}_{j3}$ are the position of the vertices that define face $f_j = \{\mathbf{v}_{j1}\ \mathbf{v}_{j2}\ \mathbf{v}_{j3}\}, \quad \forall\ j = 1, \cdots, n_f$. The first-ring area of a vertex $\mathbf{v}_i$ is defined as the neighborhood $\mathcal{N}_i$ in which the vertex $\mathbf{v}_i$ is connected to other vertices by only one edge (i.e., topological degree equal to 1).

1.2 Spectral Analysis for Estimating the Saliency of Each Vertex (Feature Vertex Extraction)

In order to identify the vertices, which represent features, we firstly follow a spectral analysis process for estimating the saliency of each vertex. Then, we use the magnitude of the saliency to classify the vertices into features (big values) and non-features (small values). A description of this spectral analysis is presented below:

For each vertex $\mathbf{v}_i$ of the mesh, we create a patch of $k+1$ vertices $\mathcal{P}_i = \{\mathbf{v}_i, \mathbf{v}_{i_1}, \mathbf{v}_{i_2},\ \cdots,\ \mathbf{v}_{i_k}\}$ (including the vertex $\mathbf{v}_i$) that consists of the k geometrical nearest vertices of the vertex $\mathbf{v}_i$ based on their coordinates. For this process, we use the k nearest neighbors (k-nn) algorithm (typically, setting $k = 15$). The

geometrical information of these patches (e.g., k nearest vertices sorted in increased order based on their relative distance with the reference vertex) is used to create n matrices $\mathbf{N}_i \in \mathbb{R}^{(k+1)\times 3}$, each one for the n vertices of the mesh. Finally, the matrix $\mathbf{N}_i$ consists of the $k+1$ normals of the corresponding vertices:

$$\mathbf{N}_i = [\mathbf{n}_i,\ \mathbf{n}_{i_1},\ \mathbf{n}_{i_2},\ \cdots,\ \mathbf{n}_{i_k}]^T\ \forall\ i = 1, \cdots, n \tag{1}$$

where the normal $\mathbf{n}_i$ of a vertex $\mathbf{v}_i$ is defined as:

$$\mathbf{n}_i = \frac{\sum_{j \in \mathcal{N}_i} \mathbf{n}_{cj}}{|\mathcal{N}_i|}\ \forall\ i = 1, \cdots, n \tag{2}$$

Then, we estimate the covariance matrix $\mathbf{R}_i$ for each matrix $\mathbf{N}_i$, according to:

$$\mathbf{R}_i = \mathbf{N}_i^T \mathbf{N}_i \in \mathbb{R}^{3\times 3}\ \forall\ i = 1, \cdots, n \tag{3}$$

and we decompose it:

$$\text{eig}(\mathbf{R}_i) = \mathbf{U}_i \mathbf{\Lambda}_i\ \forall\ i = 1, \cdots, n \tag{4}$$

where $\mathbf{U}_i \in \mathbb{R}^{3\times 3}$ denotes the matrix whose columns are the corresponding right eigenvectors and $\mathbf{\Lambda}_i = \text{diag}(\lambda_{i1}, \lambda_{i2}, \lambda_{i3})$ denotes a diagonal matrix with the corresponding eigenvalues $\lambda_{ij},\ \forall\ j = 1, \cdots, 3$, so that:

$$\mathbf{R}_i * \mathbf{U}_i = \mathbf{U}_i * \mathbf{\Lambda}_i \tag{5}$$

The saliency s_i of a vertex $\mathbf{v}_i$ is defined as the value given by the inverse norm 2 of the corresponding eigenvalues:

$$s_i = \frac{1}{\sqrt{\lambda_{i1}^2 + \lambda_{i2}^2 + \lambda_{i3}^2}}\ \forall\ i = 1, \cdots, n \tag{6}$$

We also normalized the values in order to be in the range of [0-1], according to:

$$\bar{s}_i = \frac{s_i - \min(s_i)}{\max(s_i) - \min(s_i)}\ \forall\ i = 1, \cdots, n \tag{7}$$

We assume that a small value of saliency means that the vertex lies in a flat area while a big value means that the vertex lies in an edge or corner. Observing the Eq. (6), we can see that a large value of $\sqrt{\lambda_{i1}^2 + \lambda_{i2}^2 + \lambda_{i3}^2}$ corresponds to small saliency. This observation can be explained under the aspect of spectral analysis, having used the steps of Eqs. (1)-(4). The normal of a vertex, lying in a flat area, is represented by one dominant eigenvector (Fig. 1), the corresponding eigenvalue of which has a very big value $\lambda_1 \gg \lambda_2 \cong \lambda_3$. On the other hand, the normal of a vertex lying in a corner is represented by three eigenvectors, the corresponding eigenvalues of which have small but almost equal values $\lambda_1 \cong \lambda_2 \cong \lambda_3$. For the identification of which vertices represent features, we use the k-means algorithm for separating the normalized values of the saliency into 5 different classes. We assume that the first two classes consist of non-features vertices, while the three next classes consist of features. In Fig. 2, we present the model "14.ply" in 5 different colors. Each color represents one of the 5 classes. The most salient vertices are those with the highest value, represented with red color, while vertices in flat areas are represented with blue color.

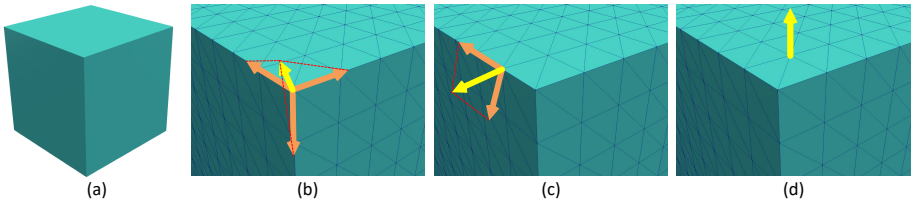

Figure 1: (a) Cube model, (b) corner feature (3 main eigenvectors need to describe the vertex normal), (c) edge feature (2 main eigenvectors need to describe the vertex normal), (d) flat area feature (1 main eigenvector needs to describe the vertex normal).

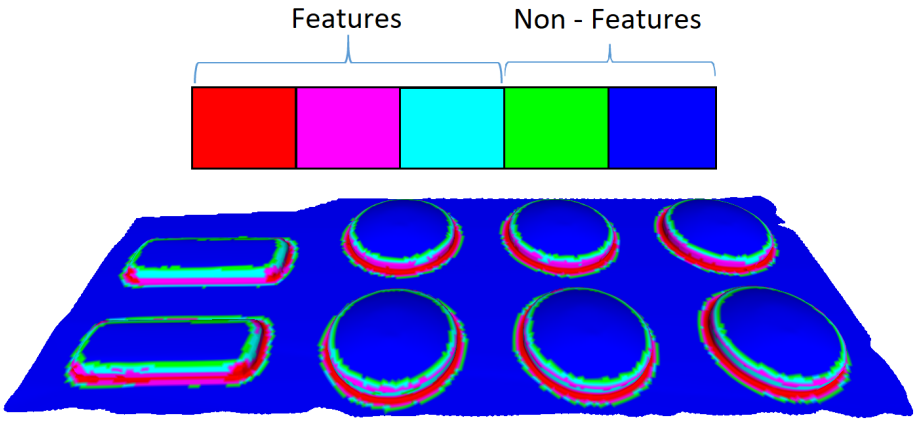

Figure 2: Colored vertices classified in different classes based on their saliency.

Finally, in Fig. 3 we present the features extraction (red vertices) based on the grouping of the three high salient classes.

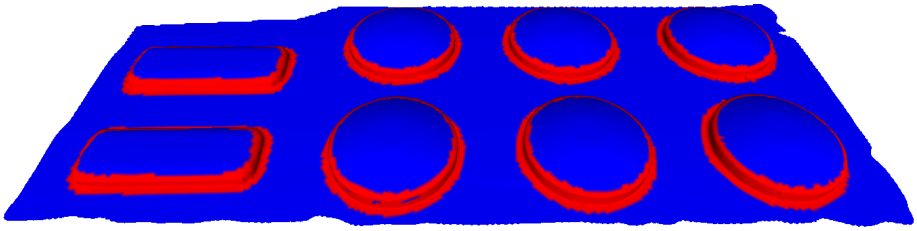

Figure 3: Corresponding .obj output with features representation of the model "14.ply" with connectivity.

To note here that the execution time of the algorithm depends on: (i) the size of the mesh and (ii) the size of the patches, but generally, it is very fast.

1.3 Feature Curve Identification Based on Mean Curvature

Once we have estimated the features of a mesh, we use the mean curvature m_c for identifying the feature curves based on the clustering of the m_c values. In Fig. 4, we present examples of feature curves of different models. The initial number of the feature curves is unknown for each model hence we evaluate the optimal number of clusters, in a range of [1-5], using the Calinski-Harabasz clustering evaluation criterion, and then we use the k-means algorithm for the clustering.

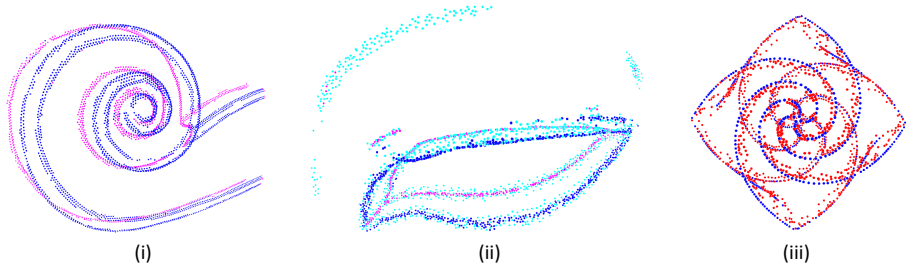

Figure 4: Feature curves of different models: (i) 7.ply, (ii) 12.ply, (iii) 13.ply.

1.4 Feature Curve Similarities Between Different Models

The normalized mean curvature $\bar{m}_c$ and salient values $\bar{s}$ are additionally used in order to compare feature curves of different models and find possible similarities between them. More specifically, we estimate the vectors $\dot{s} \in \mathbb{R}^{10\times1}$ and $\dot{m} \in \mathbb{R}^{10\times1}$ which consist of the values of the corresponding histograms, according to:

$$\begin{aligned} \dot{s} &= \text{histogram}(\bar{s}) \\ \dot{m} &= \text{histogram}(\bar{m}_c) \end{aligned} \tag{8}$$

and we create the vector $q \in \mathbb{R}^{20\times1}$ by concatenating the vectors $\dot{s}$ and $\dot{m}$:

$$q = [\dot{s}\ \dot{m}] \tag{9}$$

To investigate the similarity between two feature curves A and B, we estimate the correlation coefficient using the vectors q_A and q_B, according to:

$$r = \frac{\sum_{i=1}^{20}(q_{A_i} - \bar{q}_A)(q_{B_i} - \bar{q}_B)}{\sqrt{(\sum_{i=1}^{20}(q_{A_i} - \bar{q}_A))(\sum_{i=1}^{20}(q_{B_i} - \bar{q}_B))}} \tag{10}$$

where $\bar{q}_A$ and $\bar{q}_B$ indicate the mean values. In Fig. 5, we present a table 15×15 showing the similarities between all models, following the aforementioned pipeline. A value equal to 0 means that the feature curves of two different models are identical.

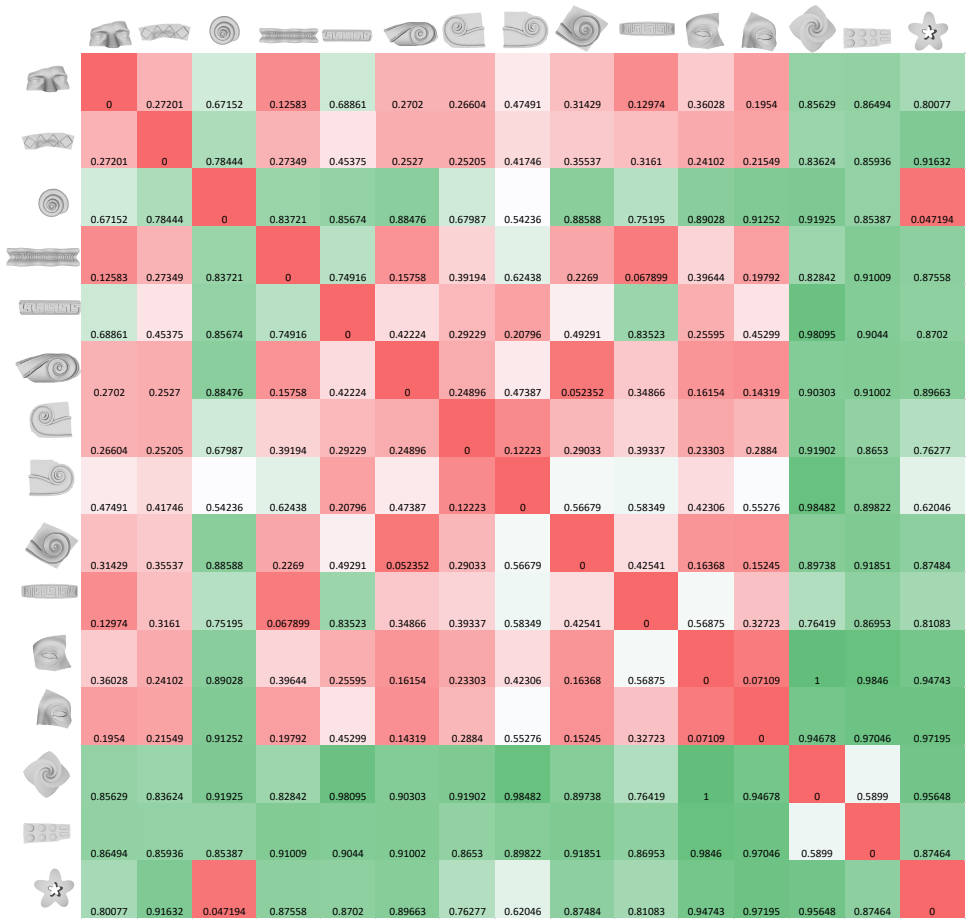

Figure 5: Correlation matrix with values ranging from 0 to 1 showing the similarity of feature curves between different models. The models are ordered by the increased number which indicates their name.

Inspecting the numerical results of Fig. 5, we can see that the models: 11 and12, 7 and 8, 6 and 9, 4 and 10 are highly related to each other, which is obviously true. On the other hand, models 3 and 15 are also highly related to each other however without an apparent similarity. Models 13 and 14 are totally unrelated to any other model.

To note here that the results may differ a little by run to run since we use k-means for the clustering which does not give exactly the same values in each execution, however, this does not negatively affect the quality of the results.

2 Technical Manual of the Application

In this section, we present and explain every functionality of the application for helping the first time user. Fig. 6 shows the general view of the application when a user opens it for the first time.

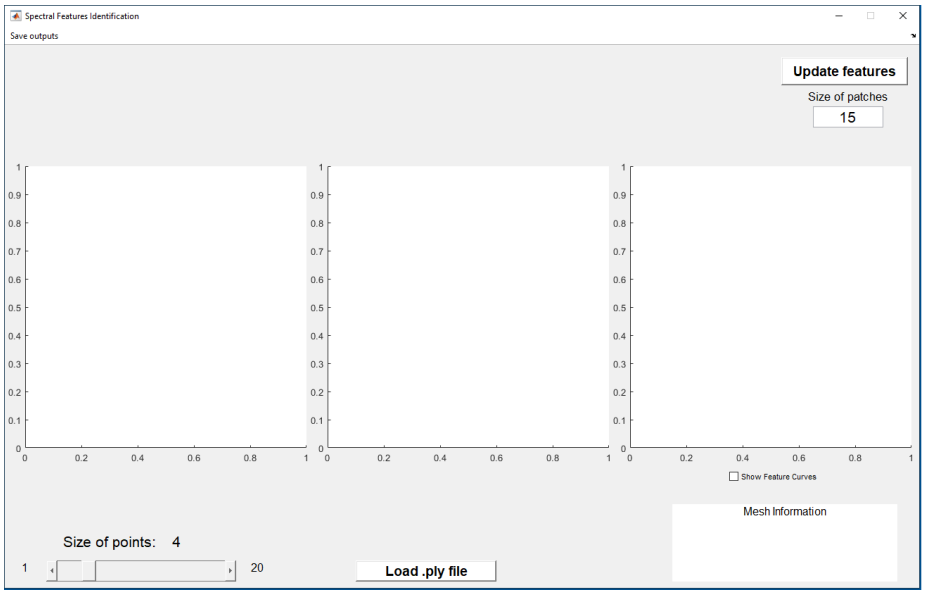

Figure 6: General view of the application when a user opens it for the first time.

Firstly, the user has to load a .ply file. The “Load .ply file” button (Fig. 7 highlighted in red) is used for this purpose. When it is pushed, a new window opens for searching the preferable file.

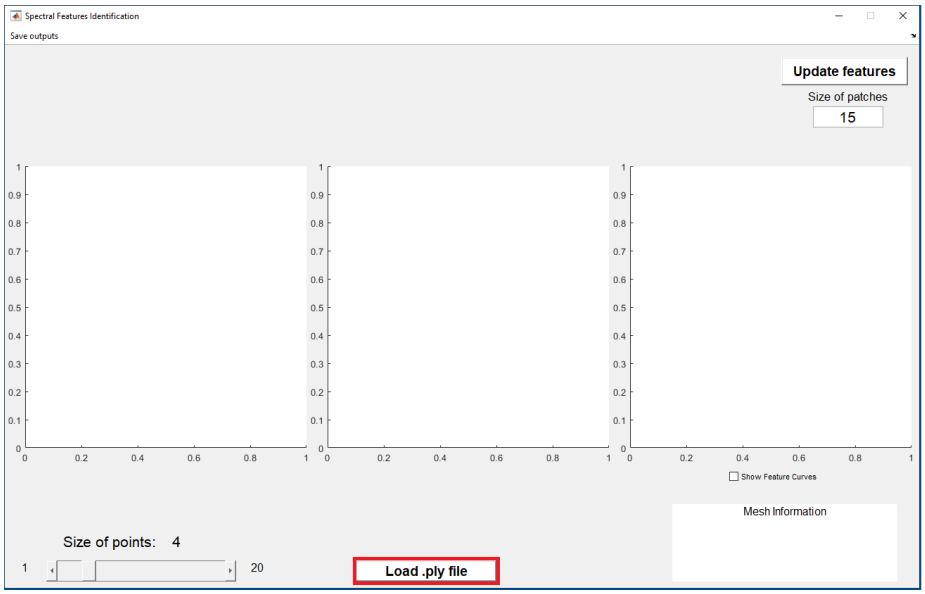

Figure 7: The "Load .ply file" button for loading a new .ply file.

When a .ply file has been selected, two point clouds are presented in the corresponding axes areas, as presented in Fig. 8.

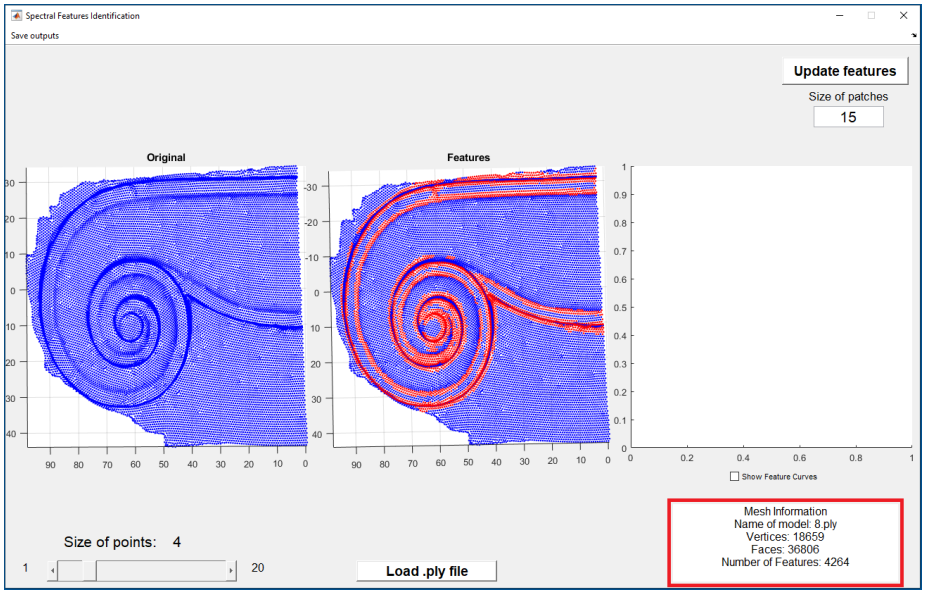

Figure 8: Displaying on the selected model and features identification.

In the left area, the original mesh, without any feature identification, is presented while in the right area the same 3D model is presented but its features are highlighted in red color. Additionally, the "Mesh Information" box area is filled with information related to the mesh. More specifically:

- The name of the loaded model (e.g., 8.ply)
- The number of vertices that the model has (e.g., 18659)
- The number of faces that the model has (e.g., 36806)
- The number of vertices that have been identified as features (e.g., 4264)

The feature identification is a very fast procedure so it runs automatically when a new model is loaded. On the other hand, the feature curve identification is not so fast, especially for large models, so it is disabled by default. However, the user can easily enable it by checking the "Show Feature Curves" checkbox (Fig. 9 box highlighted in red color).

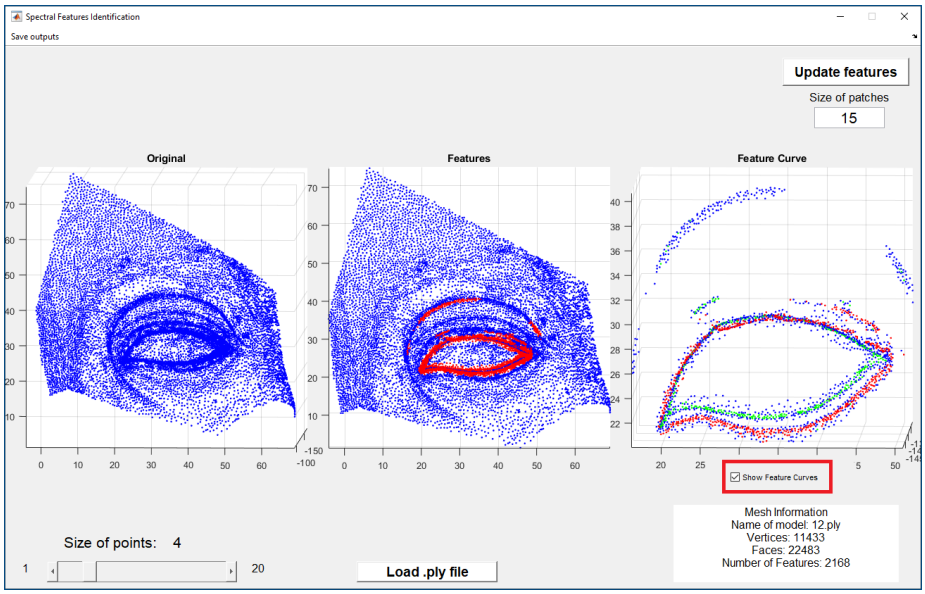

Figure 9: Enabling the feature curve identification functionality.

To note here that when a model is chosen, the application automatically estimates the features of a model using the default pre-defined value of patches size equal to 15. However, this value is only optional and the user can change it for better results. Specifically, the user can change the number, which indicates the size of patches, using the editable area, as presented in Fig. 10 (highlighted in red color), and then pressing the "Update features" button. The ideal patch size depends on the geometric characteristics (small or large-scale features) of each model. Instinctively, we could say that the larger the patch size, the more the number of the identified feature vertices. Despite the fact that it is generally true, it is not perfectly correct since larger patch size means the identification of large-scale features (low-spacial frequencies features) while smaller patch size means the identification of small-scale features.

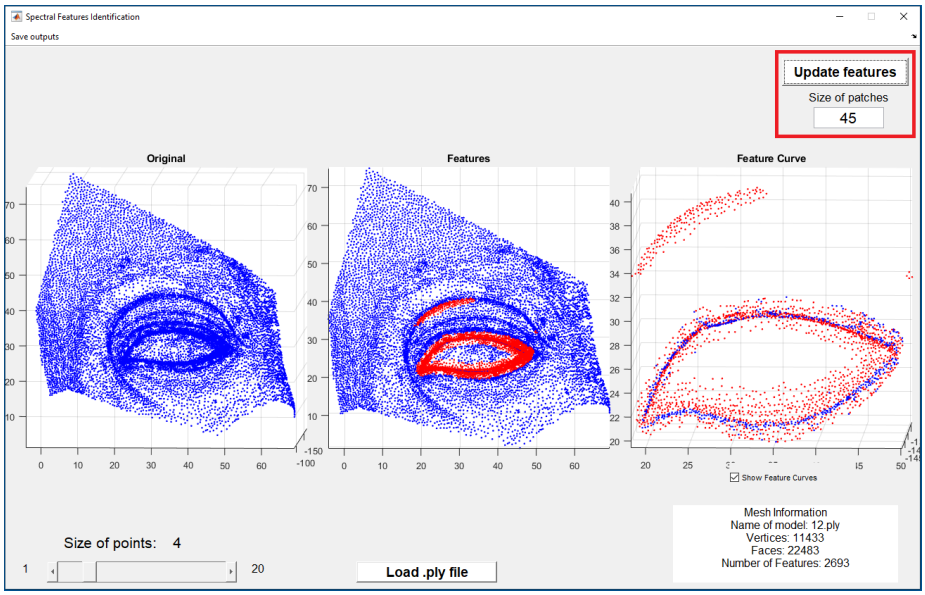

Figure 10: Features identification of model "12.ply" using a patches size equal to 45.

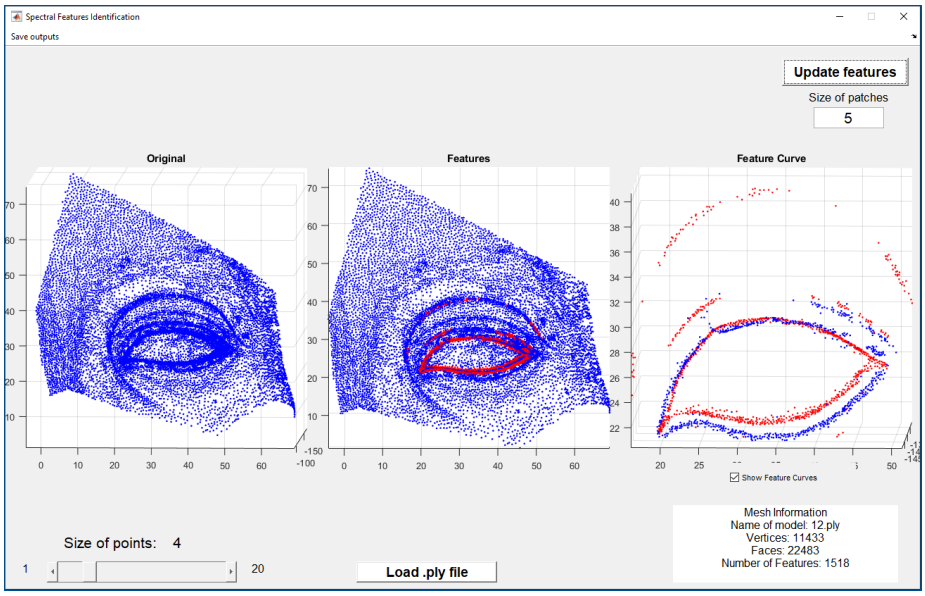

Figure 11: Features identification of model "12.ply" using a patches size equal to 5.

In Figs. 10 and 11, we present two examples in which a larger (e.g., 45) and a smaller (e.g., 5) patch size area is selected for the same model. It is obvious that a different number of features are identified in each case, as also shown in the information box. To note here that in all of our experiments, we use the

same patch size, equal to 15, for any model and we do not search for ideal patch size per each model individually.

Some models are denser than others. For this reason, we have added a slider which defines the size of the points, as presented in Fig. 12 (highlighted in red box).

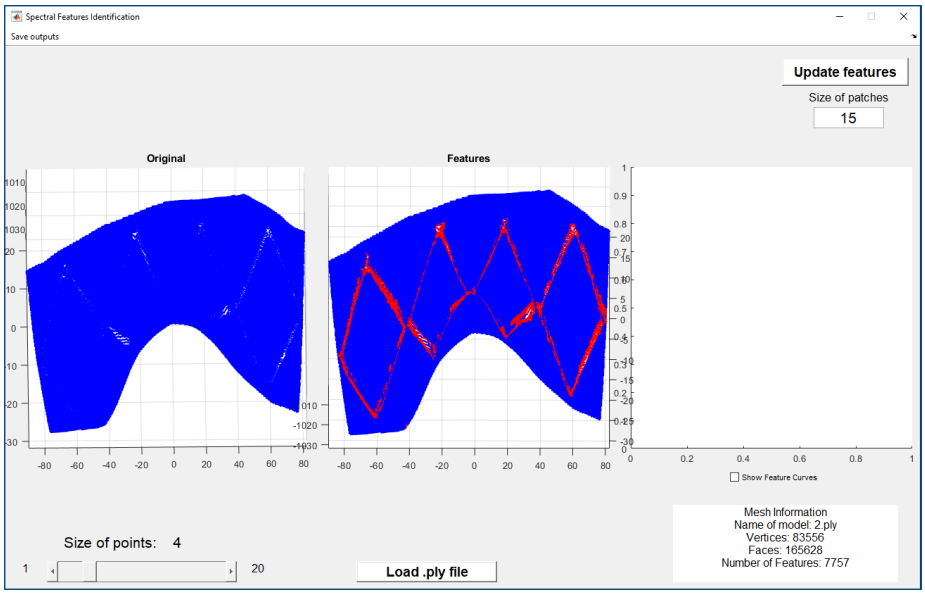

Figure 12: Displaying point clouds with point size equal to 4.

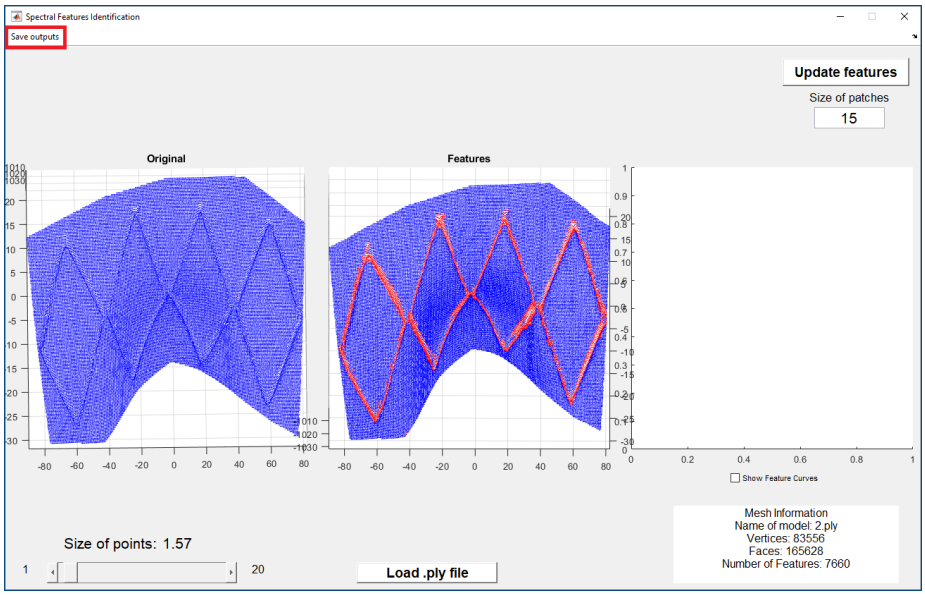

Figure 13: Displaying point clouds with point size equal to 1.57.

Finally, in the menu area there is a tab "save outputs", in Fig. 13 (highlighted in red box). We provide three standard different types of output files:

- a .txt file containing the indexes of the feature points, named **modelnumber_.txt** (e.g., "1_txt").
- a .obj file highlighting the feature points in red color, named **modelnumber_features.obj** (e.g., "1_features.obj").
- a .obj file showing in different colors, different categories of features based on their saliency, named **modelnumber_saliency_features.obj** (e.g., "1_saliency_features.obj").

Additionally, if the user enables the "feature curve identification" functionality, the application provides three extra type of output files:

- k .txt files (each one for the k identified feature curves), containing the indexes of the corresponding points of each feature curve, named **modelnumber_featline.txt** (e.g., "1_1.txt").
- a .obj file showing the features curves in different colors, named **modelnumber_feature_curves.obj** (e.g., "1_feature_curves.obj").
- a .csv file containing the 20 coefficients of the histograms, named **modelnumber_coefficient.csv** (e.g., "1_coefficient.csv").

In order to estimate the correlation between feature curves of different models, we use the Matlab script "feature_curve_correlation.m" using as input the previous extracted .csv files of every model. The output file of this script is:

- a .csv file named "similarity_matrix_of_feature_curves.csv".