Gridification of facial feature vertices extraction from 3D head

Achraf Othman[1], Heithem Abbes[1], and Mohamed Jemni[1]
[1]*Research Lab. of Technologies of Information and Communication, University of Tunise*
5 Av. Taha Hussein, BP56 Bab Mnara Tunis, Tunisia
{achraf.othman, oussama.elghoul}@utic.rnu.tn, mohamed.jemni@fst.rnu.tn

***Abstract*— the extraction of Facial features vertices from 3D faces is a beforehand task for retrieval, recognition, classification and tracking processes. Facial regions are detected from the cloud of points using the ratio of geodesic to Euclidean distances between pairs of vertices that combine neighbor coordinates and position information of the vertex. Many facial feature points among those specified by the MPEG-4 Standard are extracted automatically inside the facial region based on the topology of 3D face. The extracted feature points can be used to define the visual aspect of the face and to provide an MPEG-4 compliant face model. In addition, the displacement of these points allows the MPEG-4 decoder to deform the shape and to ensure the animation of the face. 3D faces are composed of vertices and surfaces. Since the number of vertices and surfaces is important, the sequential computation time is great. Therefore, we present a gridification of a novel algorithm for extracting facial feature vertices from 3D mesh. This allowed running important experimentations based on large inputs data to optimize the 3D processing chain. Experimentations on EumedGrid platform show the contribution of the gridification in improving obtained results.**

***Index Terms*—Image processing, computer vision, segmentation, edge and feature detection.**

I. INTRODUCTION

In recent years, a concurrent interest in facial animation had been increased. In fact, thanks to the technological growth in hardware development, especially that of graphics card and grid computing, it become possible to synthesize facial expressions in short time. Indeed, if it is easy to make a virtual character credible in the eyes of the user, the task is more difficult in respect of an avatar, the human eye having trained to see real human and collecting directly the smallest especially level defects of facial animation. Several works has been completed with the aim to simulate or clone the transformations of facial animation. Each system developed has these own characteristics and its own techniques of animation. But all these techniques were based essentially on the topology of the mesh which constituted the human head. Insufficient resources and computing speed were the main constraints of real time. In addition to the problems of processing, the deformations, applied on a model, were usually modeled with hands which required a lot of human resources. The reusability of key or meta-model was important and interesting, contrary to a system based on interpolation like Free-Form Deformation (FFD) [1]. Several systems with parameters had emerged whose main were FACS [2] and MPEG-4 Facial Animation Standard [3]. The choice of a system returns to choose the key points allocated on the 3D model. Each system had its own feature points which were applied on different faces or heads. Nowadays, FACS system is employed in the field of the analysis of facial expressions and MPEG-4 standard is oriented for applications in networks. Despite the advantage of parametric systems, we note that the selection of feature points still manual and until now there are no approaches for full automatic extraction of these points from 3D head. Our work is located in the first phase of the animation facial which consists to design a system able to extract automatically feature points from a 3D mesh based on the ratio of geodesic and Euclidean distances. Due to the important time of processing, we will run algorithms on EumedGrid Infrastructure [4]; which require a parallel version of algorithms written en MPI [5]. A point concerning the design of virtual models and their mesh topologies will not be discussed here. In our implementations, we apply directly on the mesh generated automatically by free open source tools or a data from benchmark. Our works can be used in too many applications.

The paper is organized as follows. Section 2 is a review of several previous works of facial feature vertices extraction from 3D head. Survey of segmentation techniques is defined in Section 3. In Section 4, we talk about the 3D face geometry. Section 5 describes the data used for experiments. The approach and different steps are presented in Section 6. Final conclusion is drawn in Section 7.

II. REVIEW OF PREVIOUS WORKS

Extracting feature points from 3D objects is a topical problem that attracts many researchers. Features are considered as important parts of 3D objects. Feature extraction from 3D objects is a beforehand task for analysis, segmentation, classification, retrieval, recognition, tracking, and compression. Several works have been developed aiming to resolve these problem. Below, we quote some

algorithms designed to detect automatically the most prominent facial features from 2D images [6]:

- D'Orazio [7] proposed a fast eye detector algorithm that uses iris geometrical information for determining in the whole image the region candidate to contain an eye, and then the symmetry for selecting the couple of eyes. This work was limited on extracting eye position on 2D images. And, there isn't an extension to determine others parts of the face. Through different experimentation, the results shown that the algorithm is robust.

- The work of Jacquine and others [8] was to track faces and features points (nose, eye and mouth). The authors proposed two automatic algorithms which respectively performed the detection of head outlines and identify nose-eye-mouth regions. Experimental was applied only on 3 faces. The head location algorithm performed robustly in normal situation. The tracking of feature point was performed robustly. No experimental results were given.

- Yilmaz et al. [9] proposed a new approach for detecting facial features for images and video streams. They assign a confidence number to combinations of feature candidates given the edge map of the face. Feature candidates are determined using probability distribution of color space of skin, eyes and eyebrows. Authors conducted experiments on both still images and eleven video sequences including two CNN interviews. There work had not been applied on 3D faces and no experimental results were given.

- Lai et al. [10] tried to extract facial feature points from 2D images under different lighting condition. The main difficulties were high detection accuracy, low computational time and nonlinear lamination. To solve these problems, authors proposed to make use of skin color, lip color and also the face boundary information. For experiments, they 743 images from the Omron database. The detection accuracy was around 86% and the computational time 7s.

- According to Mu and others [11], their work addressed a method of extracting human facial features from the head-and-shoulder images used in videophone communications included motion information in their feature detection scheme. The algorithm was called spatial eye-mouth finder (EMF) based on temporal and spatial domains. Only 2D images were tested and results were satisfactory.

- Lam and others approach's [12] could detect 15 feature points at different perspective variation. Rotation of the face could be estimated using geometrical measurements. Experimental results shown that the overall recognition rates are over 84 percent.

For 3D mesh:

- Xu [13] proposed a method to solve a specific problem, i.e., locating the nose tip by one hierarchical filtering scheme combining local features. The algorithm extracts local surface features and local statistical features of each point. The method is based on computing the Effective Energy (EE) for each point. Then, the Included Angle Curve IAC is defined to estimate the nose ridge. This work was limited on locating only the nose tip position. No detailed experimental results were given.

- Shalini [14] proposed 3D face recognition algorithm based on PCA. For this algorithm, a subsection of each face range image of size 354 pixels, enclosing the main facial features was employed. The recognition algorithm employed geodesic and Euclidean distance between two faces. The data set contained 1128 head models of 105 subjects.

- Jiang [15] introduced a new algorithm of calculating the displacements of face animation vertices. It was based on Facial Animation Parameters (FAP) of MPEG-4 Standard. According to authors, feature points (Facial Definition Parameters) was identified manually. Experimental results were applied on a limited number of 3D faces.

In literature, there exist others works and algorithms. We focus only on several approaches related to our work. In fact, existing works, through various algorithms, lead us to use geometry rules like Euclidean and geodesic distance. Another, there aren't approaches to extract feature points from 3D head. So from the results of 2D faces and thanks to the segmentation of the 3D mesh, we introduce a novel approach for extracting feature vertices from 3D head. Mesh segmentation techniques will be discussed in the next section towards introduce the new approach of segmentation 3D faces.

III. MESH SEGMENTATION TECHNIQUES

Segmentation of 3D surface meshes has become an important step for many fundamental problems in computer graphics like Digital Shape Reconstruction or Recognition. It can be used not only for provide semantic information but also for the exploitation of high level semantics from 3D Data. Usually, a 3D output is acquiring by different 3D editor tools. The generated output is a set of 3D points which present the topology of the object. The topological structure, generally, is a triangulated, and it is ready for any processing. While many 3D mesh segmentation algorithms, methods and techniques has developed according to a specific problem over the last several years. These different methodologies can be classified into the following categories [16] [17]:

(i) Region growing: is simple region-based mesh segmentation. In this technique, the segmentation examines neighbours vertices of initial vertices and determines whether the vertex should be added to the region.

(ii) Watershed-based: In this technique, regions are segmented into water fills or catchment basins. The watershed algorithm derives its name from the manner in which regions are segmented into catchment basins.

(iii) Reeb graphs: It is a topological approach for 3D indexing that preserves mesh proprieties. It is composed by a set of vertex that represents critical points called minima or maxima depending to their position.

(iv) Model-based: Starting from a repetitive 3D Model or shape, this technique can seek for a probabilistic model towards explaining the variation of this shape.
(v) Skeleton-based: This technique is adapted for 3D Mesh animation. The segmentation areas of the object are not based on the geometry of the shape but also on anatomical information.
(vi) Clustering: or k-means algorithm is an approach that composes the mesh on k partitions or clusters in which each cluster belongs to the score with the nearest average.
(vii) Spectral analysis: A common feature of the low-poly mesh. It uses selected vertices of the affinity matrix to obtain data representations that can be more easily clustered.
(viii) Critical points-based: This method uses defined critical points in the mesh. These points are salient feature and used to aid the segmentation process.
(ix) Multiscale Shape Descriptors: Based on shape analysis and mesh classification, this method converts vertices to a signal and apply some signal processing techniques (Fourier Transform as example) toward 3D reconstruction or volume visualisation.
(x) Markov Random Fields: Pichleret. al. [18] used Markov Random Fields to acquire the segmentation of a range view scan.
(xi) Direct segmentation: Is a technique that segment the mesh into smaller regions belonging on sharp or different sort of edges.
Mesh segmentation is an important step to our work. It can divide 3D faces into meaningful areas such nose, ears and front of the face. Many techniques was described in this section, we interested in methods based on digital geometry due to the face topology. The next section will be a description of the 3D face.

IV. The face geometry

A graphical representation of a face in the 3D world is required to render a virtual head. A 3D mesh is a list of vertices (or nodes or points) and a list of surfaces (or polygons) connecting the vertices. The mesh is commonly used to represent the shape. The relative position of the face feature vertices, described in the previous section, differs from person to person. The geometry of the face model is usually defined by an artist-designed face wire-frame [19]. Laser scanners can acquire a precise 3D realistic face shape with texture. The problem with this modeling technique is that the obtained face model topology is arbitrary and does not fit the typical motion characteristics of human faces. Furthermore, these models are too dense to be animated [20]. Different research groups dealing with face animation and coding have therefore designed own 3D mesh faces that can handle the anatomical face deformations. The face mesh model is composed of a finite set of vertices $\{V\} = \{V_1, V_2, \cdots, V_n\} \in \mathbb{R}^3$. The model is located around the 3D proprietary axes (x, y and z) such that the origin lies inside the face volume. The finite set of vertices composes a finite set of surfaces $\{S\} = \{S_1, S_2, \cdots, S_n\} \in \mathbb{R}^3$ for triangulated surfaces.

V. Data

Three dimensional head models for the study and experiments were acquired from the Open Source tool MakeHuman™ [21]. It is a tool for modeling 3D humanoid characters. It generates both the shape and texture. The exported mesh is regular and not arbitrary like scanned mesh. The data set exported contains too many face models of various gender, age and ethnic. The data are stored into two matrixes: vertices and surfaces. The rows of the first matrix represent the vertices and the columns are their coefficient in 3D space. The rows of the second matrix represent the surface and the columns denote the index of vertices that compose the surface. For triangulated mesh, the number of columns in the second matrix is 3.

VI. Contributions and structure

A. Problematic

The extraction of data from 3D mesh (matrix of vertices and surfaces) is quickly emerged as a preliminary step for post-processing. These extracted data are abstract and they indicate neither the rotation nor the scale of face. After that, we build a three Cartesian axis in the head. Generally, the face has a common topology at the level of the ears, eye and the nose that are the areas more detailed at the level of mesh size, in other words the areas dense in number of points and surfaces. This area has an important number of vertices. So, we talk about density distribution of the mesh. The distances between vertices in this area is minimal. Since we can consider the mesh as a graph related, where vertices are the summits and the Euclidean distance between a pair of vertices is the arc. The distance between two vertices, which are not direct neighbors, is the minimal distance passing through their neighbors. This distance is called Geodesic distance. This treatment can lead us to determine a mark of the face such as:

- The two vertices of ears are the first Cartesian axis.
- The middle of the two vertices of ears and the nose tip is the second Cartesian axis.
- The perpendicular to precedent two axes is the third Cartesian axis.

B. Ascertainment

Aiming to determine automatically feature vertices from the mesh of the face, we have decided to study and analyze its topology. In a first step, we noticed that the distribution of vertices is not fair. The density of vertices should be distributed according to the surface curvature. Some face parts like nose, lips, eyes or ears comprises the largest density of vertices because it needs more geodesic details. The first idea we had, concerning the extraction of significant face parts, is to compute areas containing the largest number of vertices. In others terms, apply one of the

clustering algorithms to determine these areas should be an interesting way. However, this technique shows a malfunction to the faces scanned or some processed faces, due to the regular distribution of the vertices densities in all the face curvature.

The second ascertainment that we noticed is about the curvature of the mesh. In fact, the face has a particular geodesic surface in which the curve is more intense in the neighborhood of the eyes, less intense in the neighborhood of the nose and variable in the ears surfaces. The intensity of the curvature is defined as the bending of the curve or as the rate of the direction's change of its tangent vector. This characteristic encourages the use of a Watershed-based approach.

The third ascertainment is based on an anatomical aspect; it confirms that the human face is symmetric. The symmetry does not allow to do the segmentation or to compute feature points; however it can be very helpful to refine the obtained results.

Finally we remark that there exists a very interesting property of the mesh that we can exploit to determine significant parts of the face. Indeed, the rate of the geodesic distance between two vertices divided by its Euclidian distance increase in the ears, the nose, the lips and the eyes surfaces (show Fig 1).

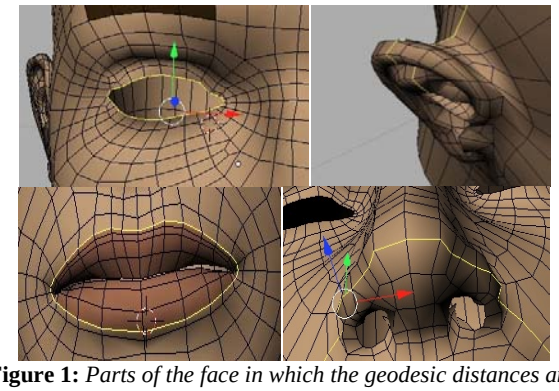

Figure 1: *Parts of the face in which the geodesic distances are larger than the Euclidian distances.*

Based on these characteristics we have built our approach, which is described in the next section.

C. Approach

The approach that we propose is mainly based on the last property cited in the previous paragraph. We propose to compute the vertices that are close by Euclidean distance and far away by geodesic distance. After determining the significant zones of the mesh we propose to apply some geometric properties like the symmetry to compute feature vertices.

The input of the system is a mesh stored in two matrixes one contains the vertices list and second describes faces. The output is a vector containing the index of feature vertices. Previously, we described the problem of rotation and scale. To resolve this problem, we try to compute the high density distribution in the 3D head. The high density distribution contains vertices that are too close. To compute these distributions:

- Computing matrix of neighbours for each vertices.
- Computing matrix of Euclidean distances for each pair of vertices.
- Computing matrix of geodesic distances for each pair of vertices.
- Extracting key point for the local three Cartesian axes in the mesh.
- Computing feature vertices through geometric rules.

Experimentations was been run on EumedGrid platform [4]. We wrote a job description file based on Job Description Language (JDL). The input sandbox is the main program and 3 files; matrix of vertices, matrix of surfaces and configuration file. The output sandbox contains as a result 3 principals' files; z_clusters.app, z_fp.app and z_repere.app which:

- z_clusters.app: include head clusters vertices.
- z_fp.app: encloses coordinate of facial feature vertices.
- z_repere.app: contains 4 vertices of the local three Cartesian axes.

D. Neighbors matrix

The matrix of neighbors is computed from matrixes of vertices and surfaces. It will be used for computing the shortest path between all pairs of vertices. We note that a pair of vertices is neighbors if they are included into two surfaces at the same time. The value in a cell in the matrix is 1 is the vertex i and the vertex j are neighbors, and 0 else. The matrix will be noted M_{neig}. The algorithm for computing M_{neig} is shown in the next figure. The complexity of the algorithm is $O(n^2 * m^2)$ which n is the number of vertices and m is the number of surfaces.

```
Data: n,m,M_surf
Result: M_neig
begin
    for i ← 1 to n do
        for j ← 1 to n do
            M_neig[i, j] ← 0;
    for i ← 1 to n do
        for j ← 1 to n do
            if i ≠ j then
                for k ← 1 to m do
                    if i ∈ M_surf[k] and j ∈ M_surf[k]
                    then
                        for l ← k + 1 to m do
                            if i ∈ M_surf[l] and
                            j ∈ M_surf[l] then
                                M_neig[i, j] ← 1;
```

Figure 2: *Sequential Algorithm for computing Neighbors matrix*

For the distributed version, we send two matrixes (vertices

and surfaces) to all nodes. Each node computes a part of Neighbor-Matrix. After that, root will gather all matrixes parts and saves the result.

# vertices	# surfaces	N1	N2	N3	N4
550	1022	1.06	1.03	0.99	0.99
725	1363	1.17	1.02	1.05	1.03
794	760	2.45	2.54	1.75	2.06
900	1704	5.06	4.30	3.89	4.27
1073	2045	5.37	5.23	5.21	5.18
1760	1727	17.49	13.31	15.61	13.13

Tab 1: *Details of execution time of Neighbor-Matrix Processing*

Figure 3 and Tab 1 show details of execution time of Neighbor-Matrix.

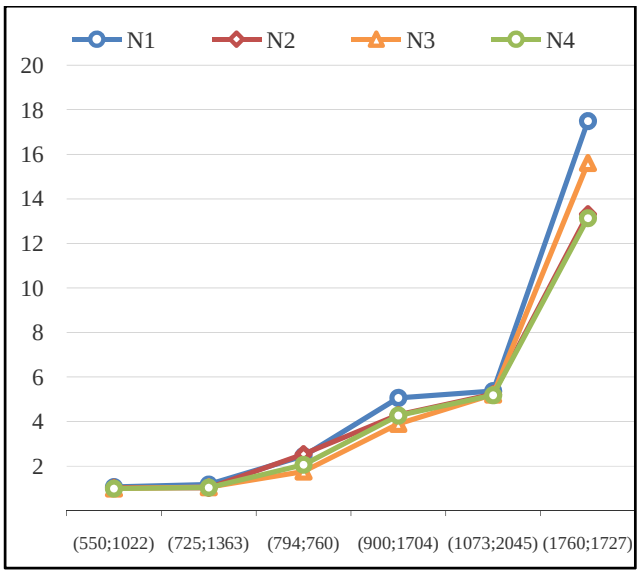

Figure 3: *Sequential Algorithm for computing Neighbors matrix*

E. Euclidean distance matrix

In digital geometry with a three Cartesian axis, the Euclidean distance noteddist between two vertices V_i and V_j is computed as follow:

$$dist(V_i, V_j) = \sqrt{(P_{jx} - P_{ix})^2 + (P_{jy} - P_{iy})^2 + (P_{jz} - P_{iz})^2}$$

The Euclidean distance matrix noted M_{eucl} is computed between two vertices V_i and V_j described in figure 3.

```
Data: n, M_points
Result: M_eucl
begin
    for i ← 1 to n do
        for j ← 1 to n do
            M_eucl[i, j] ← dist(M_points[i], M_points[j]);
```

Figure 4: *Algorithm for computing Euclidean distance matrix*

For Euclidean distance-Matrix, we apply the same procedure of Neighbor-Matrix. Table 2 and figure 5 show details of execution time of Neighbor-Matrix.

# vertices	# surfaces	N1	N2	N3	N4
550	1022	0.03	0.02	0.02	0.02
725	1363	0.06	0.04	0.04	0.05
794	760	0.03	0.03	0.04	0.03
900	1704	0.07	0.08	0.07	0.07
1073	2045	0.07	0.07	0.07	0.06
1760	1727	0.16	0.19	0.30	0.25

Tab 2: *Details of execution time of Euclidean distances-Matrix Processing*

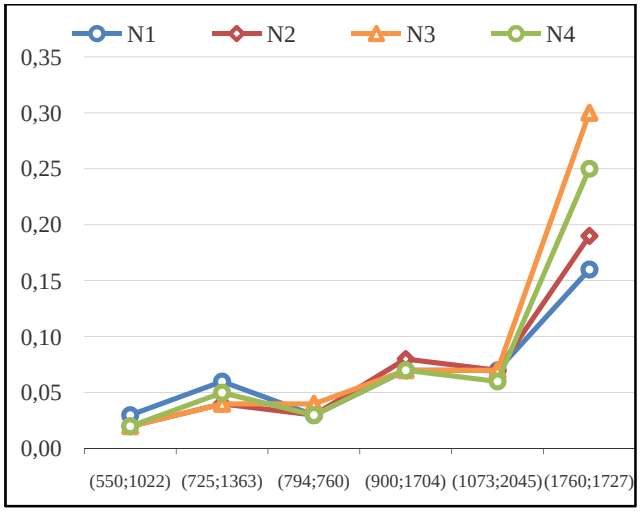

Figure 5: *Sequential Algorithm for computing Euclidean distance matrix*

F. Geodesic distance matrix

The geodesic distance between two vertices is the shortest path from the source vertex to the target vertex through the neighbors. To compute the entire shortest path between all pair of vertices, we employ the algorithm Floyd-Warshall [22] which run in (n^3) ; n is the total number of vertex in the mesh. In this step we consider the mesh as a graph $G = (V, S)$. The weight between two vertices is the Euclidean distance. The Floyd-Warshall algorithm is designed to solve all pair's shortest paths problems for graphs with negative cost edges –in our case, the cost edges are always positives due to the value of Euclidean distance. In words, the algorithm maintains a matrix $d_{ij} \in M_{floyd}$ such that at iteration k ; d_{ij} is the shortest path from i to j using nodes $1, 2, \cdots, k$ as intermediate nodes. After the algorithm terminates, assuming that no negative cost cycle is present, the shortest path from nodes i to j is d_{ij}. The main operation in the algorithm is:

$$d_{ik} = \min(d_{ik}; d_{ij} + d_{jk}).$$

After *k* iteration, d_{ij} is the shortest path distance from *i* to j involving a subset of nodes in {1,2,…,k } as intermediate nodes. The pseudo code of the algorithm is given below.

```
Data: n, M_floyd
Result: M_floyd
begin
    for i ← 1 to n do
        for j ← 1 to n do
            for k ← 1 to n do
                if M_floyd[i,j] ≥ (M_floyd[i,k] + M_floyd[k,j]) then
                    M_floyd[i,j] = M_floyd[i,k] + M_floyd[k,j];
```

Figure 6: *Pseudo-code of the Floyd-Warshall Algorithm*

# vertices	# surfaces	N1	N2	N3	N4
794	760	0.69	0.43	0.40	0.42
550	1022	1.61	1.53	1.29	1.44
725	1363	1.26	1.27	1.25	1.28
900	1704	3.17	2.65	2.33	2.65
1073	2045	3.29	3.27	3.25	3.27
1760	1727	18.09	17.35	19.35	21.23

Tab 3: *Details of execution time of geodesic Matrix Processing*

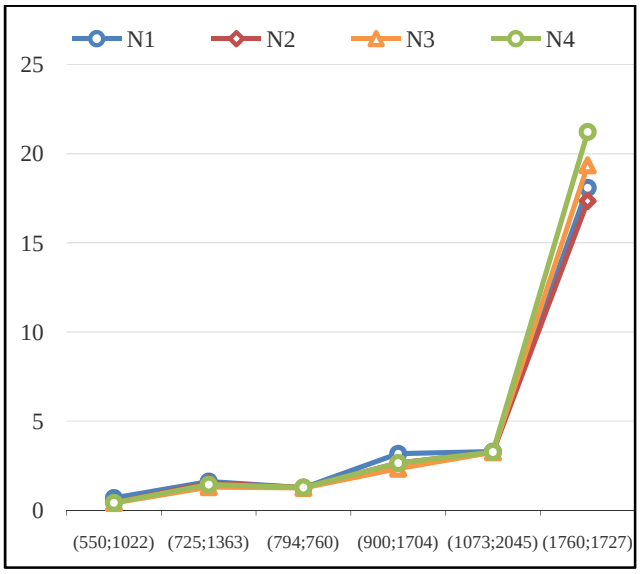

Figure 7: *Sequential Algorithm for computing Neighbors matrix*

For Euclidean distance-Matrix, we apply the same procedure of Neighbor-Matrix. Table 3 and figure 7 show details of execution time of Neighbor-Matrix.

G. *Key vertices extraction*

Key vertices noted P_i are the vertices that compose the local three Cartesian axes in the mesh. These vertices are:

- P_1: The left ear.
- P_2: The right ear.
- P_3: Themidpoint between $P_1$and P_2.
- P_4: The nose tip.

The four vertices will be extracted from the high density distribution in the mesh. In 3D head, the high density distributions are the areas of ears, eyes and frontal face (mouth and nose). Some properties for this distribution that two vertices are too close based on Euclidean distance and too far based on geodesic distance. So, if the ratio on Euclidean distance and geodesic distance for a set of vertices is near zero. For all pair of vertices we compute the ratio and we cluster the different distribution collected from the matrix of ratio. The results are shown below.

The results (figure 8) show that the high density distributions are located in the ears and frontal face (composed by eyes, nose and mouth). Now, after extracting three density distributions from 3D head, we can extract $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ from the two cluster that are too far and equidistant to the rest clusters. $P_3(x_3, y_3, z_3)$ is determined as fellow:

$$P_3\left(x_3 = \frac{x_1 + x_2}{2}, y_3 = \frac{y_1 + y_2}{2}, z_3 = \frac{z_1 + z_2}{2}\right)$$

The vertex $P_4(x_4, y_4, z_4)$ is the equidistant vertex from P_1 and P_2. So we have: $\mathrm{dist}(P_1, P_4) = \mathrm{dist}(P_2, P_4)$. In some case, if we haven't a symmetric mesh, we cannot find the coordinates of P_4. We change the formula of P_4 as:

$$\mathrm{dist}(P_1, P_4) = \mathrm{dist}(P_2, P_4) \pm \varepsilon \,/\, \varepsilon \in \mathbb{R}$$

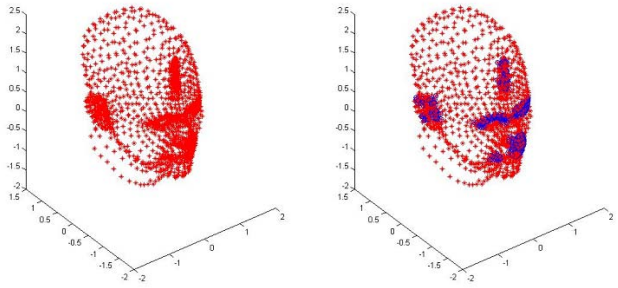

Figure 8: *Experimental results show the high density distribution in 3D Head after computing the ratio of Euclidean and geodesic distance*

H. *Feature vertices extraction*

After computing the local three Cartesian axes in the 3D mesh. We can extract now the nose tip. This vertex will be noted $P_{9.3}$ –same notation of MPEG-4 SNHC:

- Building a plane P passing fromP_3. The normal is $\overrightarrow{P_1P_2}$.
- Projection all the vertices near to P: $dist(P, P_i) \leq \varepsilon$.
- Draw a line Δ through the too far points in P.
- Compute the Euclidean distance from the projected points and Δ.
- The point $P_{9.3}$ is the far point to Δ and near to P_4.

The plane P is computed from $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ and a normal $\vec{n} = \overrightarrow{P_1P_2} = \begin{pmatrix} x_n = x_2 - x_1 \\ y_n = y_2 - y_1 \\ z_n = z_2 - z_1 \end{pmatrix}$ through a point $I(x_i, y_i, z_i)$. The equation of the plane P is:

$$P(x, y, z) = ax + by + cz + d.$$

We have:

$$P(x,y,z) = xx_n + yy_n + zz_n - (x_i x_n + y_i y_n + z_i z_n)$$

Which:

$$a = x_n; b = y_n; c = z_n; d = -(x_i x_n + y_i y_n + z_i z_n)$$

After, we find all the vertices $A(x_a, y_a, z_a)$ neighbors in P, we have:

$$dist(A,P) = \frac{|ax_a + by_a + cz_a + d|}{\sqrt{a^2 + b^2 + c^2}}$$

The projected point for A noted A′ is determined as fellow: Be $A(x_a, y_a, z_a)$ and his project $A'(x'_a, y'_a, z'_a)$, we have $\overrightarrow{P_1P_2} \parallel \overrightarrow{AA'}$, there exist a scalar k such as $\overrightarrow{AA'} = k.\overrightarrow{P_1P_2}$.

Be $\begin{pmatrix} x'_a - x_a = k.x_{\overrightarrow{P_1P_2}} \\ y'_a - y_a = k.y_{\overrightarrow{P_1P_2}} \\ z'_a - z_a = k.z_{\overrightarrow{P_1P_2}} \end{pmatrix}$, in addition $A' \in P$ mean that:

$$aa'_x + bb + cc'_x + d = 0$$

In three Cartesian axes, the coordinates of A′ equal:

$$\begin{cases} x'_a = \dfrac{(b^2 + c^2)x_a - aby_a - acz_a - da}{a^2 + b^2 + c^2} \\ y'_a = \dfrac{-abx_a + (a^2 + c^2) - acz_a - da}{a^2 + b^2 + c^2} \\ z'_a = \dfrac{-acx_a - bcy_a + (a^2 + b^2)z_a - dc}{a^2 + b^2 + c^2} \end{cases}$$

The point $P_{9.3}$checks the following properties:

- The distant point to the line Δ.
- The nearest point to P_4.

To find the rest of vertices we use the algorithm peaks and troughs to locate the local extremum–minima and maxima; in 3D mesh. Extrema are the largest value (maxima) or smallest value (minima), that a function takes in a point either within a given neighborhood.

We apply the algorithm in 3D head from a plane P in the middle of the head (through the nose tip), the mid-plane between P and the left ear and the mid-plane between P and the right ear. The results are the feature vertices from 3D Head. Figure 8 shows the different steps from a source 3D head passing from the step of clustering, extracting the three local Cartesian axes (in yellow) and the final feature vertices. Another result is shown in figure 9.

VII. Conclusion

A novel and practical facial feature vertices system has been described, which combines several algorithms of feature detection with robustness to different mesh. The procedure is fully automatic, while user modification is also allowed if necessary. The results can be used in a system for automatic facial animation based on MPEG-4 SNHC.

Acknowledgments

We would like to thank Oussama El Ghoul and EumedGrid Support for theirs fruitful remarks and help.

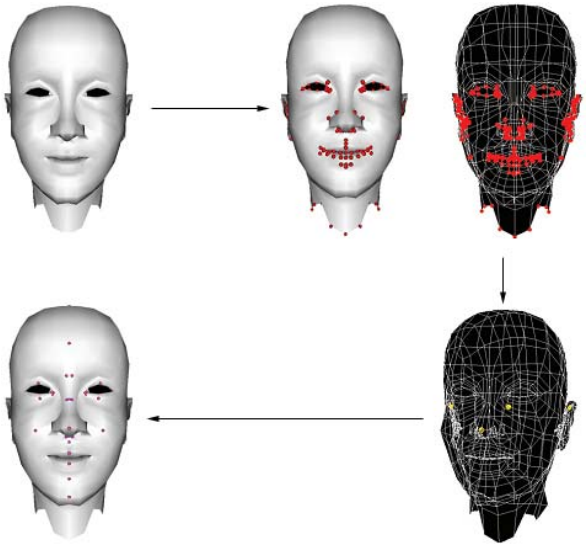

Figure 9: *Different steps to extract feature vertices from 3D head*

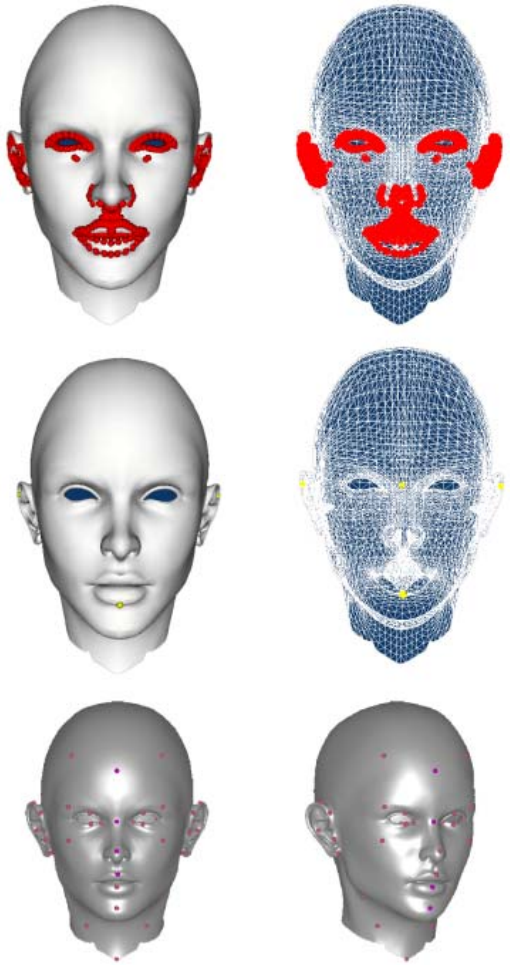

Figure 10: *Facial features vertices applied in 3D head*

REFERENCES

[1] Thomas W. Sederberg, and Scott R. Parry, "Free-Form Deformation of Solid Geometric Models," *SIGGRAPH'86*, vol. 20, no. 4, pp. 151-160, August 1986.

[2] Paul Ekman, and Wallace V. Friesen, "Facial Action Coding System, " *Consulting Psychologists Press*, 1978.

[3] Igor S. Pandzic, and Robert Forchheimer, "MPEG-4 facial animation: the standard, implementation and applications," ISBN. 9780470844656, J. Wiley, 2002.

[4] Giuseppe Andronico, Roberto Barbera, Kostas Koumantros, Federico Ruggieri, Federica Tanlongo, and Kevin Vella, "Grid infrastructures as catalysts for development on eScience: experiences in the Mediterranean," *Bio-algorithms and Med-systems journal*, vol. 3, no. 5, pp. 23-25, 2007.

[5] William Gropp, Ewing Lusk, and Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, MIT Press, 1999.

[6] Ilse Ravyse, "Facial Analysis and Synthesis,"Vrije Universiteit Brussel, Ph.d 2006.

[7] Tiziana D'Orazio, Marco Leo, Grazia Cicirelli, and Arcangelo Distante, "An algorithm for real time eye detection in face images," *17th International Conference on Pattern Recognition (ICPR'04)*, vol. 3, pp. 278–281, August 2004.

[8] Arnaud Jacquin, Alexandros Eleftheriadis, and Ros Eleftheriadis, "Automatic Location Tracking of Faces and Facial Features in Video Sequences," *International Workshop on Automatic Face and Gesture Recognition*, June 1995.

[9] Alper Yilmaz and ubarak A. Shah, "Automatic feature detection and pose recovery for faces," in *Asian Conference on Computer Vision ACCV2002*, Melbourne, Australia, January Jan 2002, pp. 23-25.

[10] Jian Huang Lai, Pong C Yuen, WenSheng Chen, Shihong Lao, and Masato Kawade, "Robust facial feature point detection under nonlinear illuminations," in *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, Vancouver Canada, August 2001, p. 0168.

[11] Fenghao Mu, Haibo Li, and Robert Forchheimer, "Automatic extraction of human facial features," *Signal processing: Image communication*, vol. 8, no. 4, pp. 309–326, May 1996.

[12] Kin-Man Lam and Hong Yan, "An analytic-to-holistic approach for face recognition based on a single frontal view,"*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 673–686, July 1998.

[13] Chenghua Xu, Tieniu Tan, Yunhong Wang, and Long Quan, "Combining local features for robust nose location in 3D facial data," *Pattern Recognition Letters*, vol. 27, no. 13, October 2006.

[14] Gupta Shalini, Mia K. Markey, Jake Aggarwal, and Alan C. Bovik, "Three dimensional face recognition based on geodesic and euclidean distances," *Proceedings of the SPIE*, vol. 6499, pp. 64990D-64990D-11, 2007.

[15] Jiang Dalong, Li Zhiguo, Wang Zhaoqi, and Gao Wen, "Animating 3D facial models with MPEG-4 FaceDefTables," *SS '02 Proceedings of the 35th Annual Simulation Symposium* , 2002.

[16] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, Nikolaos Sapidis, and Philip Azariadis, "3D Mesh Segmentation Methodologies for CAD applications," *Computer-Aided Design & Applications*, vol. 4, no. 6, pp. 827-841, 2007.

[17] Ariel Shamir, "A Survey on Mesh Segmentation Techniques," Computer Graphics Forum, vol. 27, pp. 1539-1556, 2008.

[18] Andreas Pichler, Robert B. Fisher, and Markus Vincze, "Decomposition of range images using Markov Random Fields," *International Conference on Image Processing ICIP 2004*, vol. 2, pp. 1205-1208, October 2004.

[19] Bill Fleming and Darris Dobbs, *Animating Facial Features and Expressions*.: Charles River Media, 1999.

[20] I-Chen Lin, "Reliable extraction of realistic 3d facial animation parameters from mirror-reflected multi-view video clips," National Taiwan University, Taiwan, Ph.D dissertation 2003.

[21] MakeHuman team. (2011) MakeHuman. [Online]. http://www.makehuman.org

[22] M. Chan Timothy, "More Algorithms for All-Pairs Shortest Paths in Weighted Graphs," *Proceeding of STOC'07*, September 2007.

Achraf Othman is currently a PhD student under the supervision of Prof. Mohamed Jemni. He received in August 2010 the Master degree on Computer Science from Tunis College of Sciences and Techniques (ESSTT), University of Tunis in Tunisia. His research interests are in the areas of Sign Language Processing. His current topics of interests include Grid Computing, Computer graphics and Accessibility of ICT to persons with disabilities.

Heithem Abbes received his Ph.D. in Computer Science at National School of Computer Sciences in collaboration with the University of Paris 13 in 2010. He obtained hi Master Degree in Computer Science in 2005, and his Engineer Diploma in 2003 from the faculty of Sciences of Tunis. His research is focused on Grid Computing and Peer-to-Peer systems.

Mohamed Jemni is a Professor of ICT and Educational Technologies at the University of Tunis, Tunisia. He is the Head of the Laboratory Research of Technologies of Information and Communication (UTIC). Since August 2008, he is the General chair of the Computing Center El Khawarizmi, the internet services provider for the sector of the higher education and scientific research. His Research Projects Involvement are tools and environments of e-learning, Accessibility of ICT to Persons with Disabilities and Parallel & Grid Computing.