A Novel Approach for 3D Head Segmentation and Facial Feature Points Extraction

Achraf Othman and Oussama El Ghoul, *Research Lab. LaTICE, University of Tunis, Tunisia*

***Abstract*—This work presents a novel approach for 3D head segmentation as well as for extraction of facial features vertices. These tasks are the preliminary task for retrieval, recognition, classification and tracking processes. 3D head regions are detected from a cloud of points using the ratio of geodesic to Euclidean distances between pairs of vertices that combine neighbor coordinates and position information of the point. In this paper, we propose a new algorithm to extract automatically feature points inside the facial region based on the topology of 3D face. Furthermore, in order to validate our approach, we have run experimentations on several types of full 3D head that differ in number of points and surfaces, gender and ethnic. Also, to approve its robustness, we apply our approach on a benchmark of scanned faces containing 67 models.**

***Index Terms*—Image processing, Computer vision, Segmentation, Edge and feature detection.**

I. Introduction

In recent years, a concurrent interest in facial animation has been increased. In fact, thanks to the technological growth in hardware development, especially that of graphics card, it becomes 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, as the human eye, trained to see real human, can detect directly the smallest level defects of facial animation. Several works have been achieved with the aim to simulate or clone the transformations of facial animation. Each system developed its 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. In addition to the problems of processing, the deformations, applied on a model, were usually modelled manually which required skilled 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 whereas MPEG-4 Standard is oriented for applications in networks. Despite the advantage of parametric systems, we note that the selection of feature points is 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. This goal must be pass by a beforehand task called 3D head segmentation. 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 3D scanned face from benchmark [4]. The benchmark GavabDB contains 427 of 3D facial surface images corresponding to 61 individuals (45 male and 16 female), and there are 7 different images per each person. The whole set of individuals are Caucasian and most of them are aged between 18 to 40 years old. Each element is given by a mesh of connected 3D points of the facial surface without texture. The database provides systematic variations with respect to the pose and the facial expression. The paper is organized as follows. Section 2 is a review of previous works of facial feature vertices extraction from 3D head. A survey of segmentation techniques is presented in Section 3. In Section 4, we describe the 3D face geometry. Section 5 presents the data used for experiments. The approach and the different steps of treatment are presented in Section 6. Final conclusion is drawn in Section 7.

II. The face geometry

A graphical representation of a face in the 3D world is required to render a virtual head. The head can be a 3D mesh which is a list of vertices (or nodes or points) and a list of surfaces (or closed polygons chain) 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 [5]. Laser scanners can acquire a precise 3D realistic face shape with texture. The problem with this modelling 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 [6]. Different research groups dealing with face animation and coding have therefore designed their own 3D mesh faces that can handle the anatomical face deformations. The face mesh model is composed of a finite set of vertices $\{V\} = \{V_1, V_2, \ldots, V_n\} \in \mathbb{R}^3$. 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 $\{S\} = \{S_1, S_2, \ldots, S_n\} \in \mathbb{R}^3$ for triangulated surfaces.

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 to 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. Generally, the topological structure is a triangulated, and it is ready for any processing. Many 3D mesh segmentation algorithms, methods, and techniques have been developed according to a specific problem over the last several years. These different methodologies can be classified into the following categories [7], [8]:

- Region growing: is a simple region-based mesh segmentation. In this technique, the segmentation examines neighbors vertices of initial vertices and determines whether the vertex should be added to the region. The process is iterated on using general data clustering algorithm. The disadvantage of this technique is consuming and noise or variation of intensity may result holes and over-segmentation.
- Watershed-based: regions are segmented into water fills. The watershed algorithm derives its name from the manner in which regions are segmented into catchment basins. Different approaches can be employed to use the watershed principle: local minima of the gradient or marker position defined by the programmer or determined automatically with morphological operators.
- Markov Random Fields [9]: authors used Markov Random Fields to acquire the segmentation of a range view scan.
- 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. It defines the connectivity of its level sets.
- Model-based: starting from a repetitive 3D Model or shape, this technique can seek for a probabilistic model towards explaining the variation of the shape. This task involves registration of the training examples to a common pose or samples.
- 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. It can be readily used to produce shape segmentations which are natural, robust, and have a multi-scale property.
- 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. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The problem is computationally difficult and NP-hard in execution time.
- Critical points-based: this method uses defined critical points in the mesh. These points are salient feature and used to aid the segmentation process. The approach is based on three key ideas: the transformation of the mesh vertices into a pose invariant representation, the robust extraction of prominent feature points, and the extraction of the core component of the mesh.
- Multi-scale Shape Descriptors: based on shape analysis and mesh classification, this method converts vertices to a signal and applies some signal processing techniques toward 3D reconstruction or volume visualization.
- Direct segmentation: is a technique that segments the mesh into smaller regions belonging on sharp or different sort of edges. Generally, this approach is based on hierarchical mesh decomposition. Each node in the hierarchy tree is associated with a mesh of a particular patch and the root is associated with the whole input object.

Many techniques were described in this section, We are interested, in particular, in methods based on digital geometry due to the face or full head topology. Mesh segmentation is an important step to our work and constitutes its first step. It can divide 3D faces into meaningful areas such nose, ears and front of the face. In some case, it is not possible to detect only the required clusters due to the noise or others factors (especially in scanned faces); we therefore involve some algorithms or techniques for labelling. After clustering, feature points extraction serves to facial animation. Next section will be an overview of different existing systems.

IV. Feature points extraction

Extracting feature points from 3D objects is a typical problem that attracts many researchers. Before, researchers focused on 2D face. They integrated edge detection operator and others techniques. Nowadays, many researches are interested in 3D faces. Features are considered as important parts of 3D objects. Feature extraction from 3D objects is a beforehand task for analysis, segmentation, animation, compression, classification, retrieval, recognition and tracking. Several works have been developed aiming to resolve these problems. Below, we quote some algorithms designed to detect automatically the most prominent facial features from 2D faces [10]:

- D'orazio et al. [11] 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 is no extension to determine others parts of the face. Through different experimentations, the results showed that the algorithm is robust.
- In order to track faces and features points: nose, eye and mouth [12], the authors proposed two automatic algorithms which respectively perform the detection of head outlines and identify nose-eye-mouth regions. Experimentation was applied only on 3 faces. The head location algorithm performed robustly in normal situation. The tracking of feature point was performed robustly, however, no experimental results were given.
- Yilmaz et al. [13] 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. This work has not been applied on 3D faces and no experimental results were given.

- Lai et al. [14] 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, the authors proposed to use the skin color, lip color and also the face boundary information. For experiments, they used 743 images from the Omron database. The detection accuracy was around 86%.
- According to [15], their work addressed a method for extracting human facial features from the head-and-shoulder images used in videophone communications and included motion information in their feature detection scheme. The algorithm, called spatial eye-mouth finder (EMF), is based on temporal and spatial domains. Only 2D images were tested and results were satisfactory.
- Lam et al. [16] approachs could detect 15 feature points at different perspective variation. Rotation of the face could be estimated using geometrical measurements. Experimental results showed that overall recognition rates were over 84%.

For 3D mesh, algorithms are based on the observation that feature points can be characterized by local as well as global conditions, in terms of their distances (geodesic, Euclidean or others).

- Xu et al. [17] proposed a method to solve a specific problem: 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 for each point. Then, the Included Angle Curve is defined to estimate the nose ridge. This work was limited on locating only the nose tip position. No detailed results were given.
- Shalini et al. [18] 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.
- Dalong et al. [19] introduced a new algorithm of calculating the displacements of face animation vertices. It was based on Facial Animation Parameters (FAP). According to the authors, Facial Definition Parameter (FDP) was identified manually by a programmer. Experimental results were applied on a limited number of 3D scanned faces.

In the 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. Furthermore, there are no approaches to extract feature points from 3D head. 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 introducing the new approach of segmentation of 3D faces/head.

V. Contribution and structure

A. Problematic

The extraction of data from 3D mesh (matrix of vertices and surfaces) has quickly emerged as a preliminary step for post-processing. These extracted data are abstract and they indicate neither the rotation (orientation) nor the scale of face. Generally, the face has a common topology at the level of the ears, eyes and the nose. Thes areas are more detailed at the level of mesh size. In other words, these areas are dense in number of points and surfaces. These areas have an important number of vertices. So, we talk about density distribution of the mesh. The distances between vertices in these areas are 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 2 vertices of ears are the X Cartesian axis.
- The middle of the two vertices of ears and the nose tip is the second Cartesian axis (Z axe).
- The perpendicular to precedent two axes is the third Cartesian axis (Y axe).

B. Ascertainement

Aiming to determine automatically feature vertices from the mesh of the face, we studied and analyzed 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 solution we proposed, concerning the extraction of significant face parts, is to compute areas containing the largest number of vertices. Applying 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 directions 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 (See figure 1). Based on these characteristics, we have built our approach, which is described in the next section.

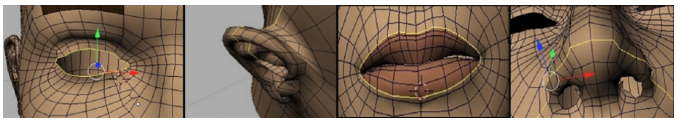

Fig. 1. Parts of the face in which the geodesic distances are larger than the Euclidian distances (Yellow lines).

C. Our 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, we propose:

- Computing matrix of neighbors 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.
- Extracting feature vertices using geometric metrics.

To validate our approach, we conducted experimentations on GavabDB benchmark and full 3D head. For each model, we created 3 files: matrix of vertices, matrix of surfaces and configuration file. The configuration file contains the number of iterations for segmentation step and the number of vertices that will be extracted. For all models, we set the value of iteration to two. The number of extracted points is the total number of vertices divided by four. The output sandbox contains as a result 3 principals files: z_clusters.app, z_fp.app and z_repere.app which:

- z_clusters.app: includes head clusters vertices. The number of extracted vertices is equal to the value stored in the configuration file.
- z_fp.app: encloses three dimensional coordinate of facial feature vertices.
- z_repere.app: contains 4 vertices of the new local three Cartesian axes.

D. Neighbours' 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 two vertices are neighbors if they are included in two surfaces at the same time. The value in a cell (i, j) in the matrix is 1 if 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 $O(n^2 \times m^2)$ which n is the number of vertices and m is the number of surfaces. In the beginning, all cells of the matrix contain the value 0. Then, we iterate n times, which n is the number of vertices. For each iteration of n, we iterate m times, which m is the number of surfaces. If we have two vertices in the same surface m, we store in the cell the value 1 else 0. Table I show the average execution time of Neighbor-Matrix in sequential mode and in parallel version of the algorithm. We made use the parallel version (in MPI) because the execution time in sequential version was very important. Thanks to EumedGrid Infrastructure [20] that allows us to reduce the execution time and run the algorithm in the others models in short time. For the parallel version, we used 4 nodes. For execution, we wrote a job description file that includes two matrices and the configuration file. We apply the same technique to rest of the algorithms: Computing the Euclidean and geodesic distance.

TABLE I
DETAILS OF EXECUTION TIME OF NEIGHBOURS-MATRIX PROCESSING (SEQUENTIAL AND PARALLEL (//) VERSIONS)

		Neighbours algorithm		Geodesic algorithm	
Vertices	Surfaces	Seq.time	//.time	Seq.time	//.time
550	1022	63.156	0.990	2.365	0.420
725	1363	96.486	1.032	8.965	1.449
794	760	102.368	2.069	6.352	1.288
900	1704	269.385	4.271	13.659	2.650
1073	2045	389.890	5.185	15.797	3.271
1760	1727	632.154	13.133	163.802	21.235
2179	2148	3760.578	126.365	204.890	23.598
2228	4368	5866.765	192.483	282.485	23.658
4929	9495	86724.047	2538.980	3070.906	56.645
6292	6152	70484.922	2045.362	642.968	31.456

E. Euclidean distance matrix

In digital geometry, with a three Cartesian axis, the Euclidean distance noted $dist$ 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. We iterate n times to browse all pairs of vertices. And for each vertex, we compute this Euclidean distance ($dist$ function in the algorithm).

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 [21] which run in $O(n^3)$, which 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 pairs 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 other 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 $1, 2, \ldots, k$ 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 iterations, d_{ij} is the shortest path distance from i to j involving a subset of nodes in $1, 2, \ldots, k$ as intermediate nodes. For geodesic distance-Matrix, we apply the same procedure of Neighbor-Matrix. Table I shows details of execution time of Neighbor-Matrix (sequential and parallel version). Results in parallel version are encouraging. Also, the execution time depends on the number of points and the number of surfaces at the same time. For example, the execution time for the model having 4929 vertices is greater that the model having 6292 vertices as in the first model, the number of surfaces is more important than the second model. It is a compromise between vertices, surfaces and execution time. After computing neighbors, Euclidean distance and geodesic distance matrices, we try to locate the face, to determine the head pose and the scale. This step requires segmentation task. Segmentation algorithm is based on the ratio of Euclidean distance and geodesic distance. We iterate k times to extract vertices that compose clusters. Figure 2 shows the experimental result in a scanned face from GravabDB benchmark. In figure 2, we remark that the edge is extracted as a cluster. This is due to the short distance between vertices in the edge of the face and to topology of the face. Our main goal is to extract feature points to animate a 3D head. So, we are interested only in full 3D head. We run our algorithms in scanned face just to approve robustness for noisy mesh in some cases. As a perspective, we may work to improve our technique.

G. Key vertices extraction

Key vertices play an important role in head pose orientation. 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 P1and P_2 and 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 (figure 2) 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 $P_1(x1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ from the two clusters that are too far and equidistant to the rest clusters. The vertex $P_4(x_4, y_4, z_4)$ is the equidistant vertex from P_1 and P_2. So we have: $dist(P_1, P_4) = dist(P_2, P_4)$. In some case, if we have not a symmetric mesh, we cannot find the coordinates of P_4. We change the formula of P_4 as: $dist(P_1, P_4) = dist(P_2, P_4) \pm \varepsilon / \varepsilon \in \mathbb{R}$. Experimental results on several scanned face and 3D head are the same for each model. Our algorithms are invariant to the pose, orientation, scale and translation. And this is approved mathematically by the ratio of the geodesic and Euclidean distance. Starting from this local 3D Cartesian axis, we compute head pose, scale and translation vector. After that, we apply the required transformations to align the head in the center according to Standard MPEG-4.

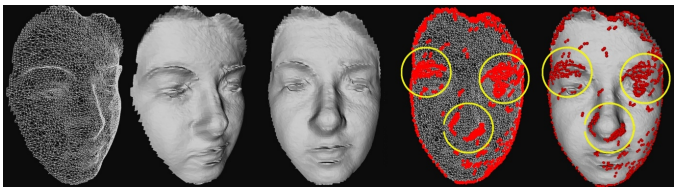

Fig. 2. Extracted cluster in red applied to 3D scanned face. Yellow circles present the high density area in the face. Due to the noisy of the scanned mesh, boundaries of the face were considered also as a cluster (Vertices count=6292; Surface count=6152)

H. Feature vertices extraction

Extracting facial feature points such as eyes, mouth and nose plays an important role in many applications such facial animation, and this is our main goal. The proposed methods are based on the geometrical metrics of 3D mesh. At the beginning, we may extract the first feature point that allows us to compute the rest. The algorithm is based on the observation that feature points can be characterized by local as well as global conditions, in terms of their Euclidean and geodesic distances. We can extract now the nose tip. This vertex will be noted $P_{9.3}$ same notation of MPEG-4 SNHC as follow:

1) Building a plane P passing from P_3. The vector of P is noted $\vec{P_1P_2}$.
2) Projection of all the vertices near to P where $dist(P, P_i) \leq \varepsilon$. ε is a small value aiming to reduce the error rate of computing.
3) Draw a line Δ through the too far points in P.
4) Compute the Euclidean distance from projected points to Δ plane.
5) The point $P_{9.3}$ is the far point to Δ and the nearest vertex 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 through a point $I(x_i, y_i, z_i)$. The normal vector n is defined as follow:

$\vec{n} = \vec{P_1P_2} = \begin{pmatrix} x_n = x_2 - x_1 \\ y_n = y_2 - y_1 \\ z_n = z_2 - z_1 \end{pmatrix}$. The equation of the plane P in Three dimension Cartesian Axis is (where a, b, c and d are constants): $P(x, y, z) = ax + by + cz + d$. 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: $a = x_n$, $b = y_n$, $c = z_n$ and $d = -(x_i x_n + y_i y_n + z_i z_n)$. After, we find all vertices $A(x_{a'}, y_{a'}, z_{a'})$ neighbors in P, we have:

$$dist(A, P) = \frac{|ax_a + by_a + cz_a + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (1)$$

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 $\vec{P_1P_2} \| \vec{AA'}$, there exists a scalar k such as $AA' = k.P_1P_2$. We have $A' \in P$ means that $aa'_x + bb'_y + cc'_z + d = 0$. 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 extremumminima 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 3 shows the different steps from a source 3D head passing from the step of clustering, extracting the three local Cartesian axes and the final feature vertices. Experimental results were run on 13 full 3D head. The obtained results are very promising and that allow us to develop a full facial animation process in the future.

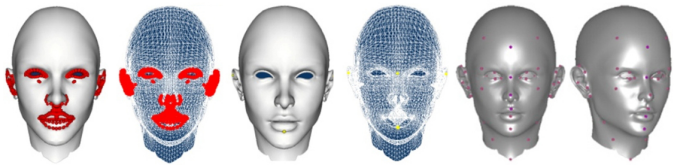

Fig. 3. Red areas present the extracted cluster; yellow points are the local Cartesian axis; pink points are extracted vertices

VI. Discussion and Conclusion

In this paper, a novel and practical facial feature vertices system has been described, which combines several algorithms of feature detection with robustness to different mesh (gender, ethnic, scale, translation, pose head, number of vertices, number of surfaces). The proposed technique is based on 3D face/head segmentation computed from the ratio of Euclidean and geodesic distance. 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. Experiment results testified the feasibility and validity of our method based on introducing geometrical metrics. There are several interesting directions in which our work can be extended, that address the main limitation of our work: reducing the execution time and robustness to the noisy model. The hidden goal of facial feature extraction is imitating human visual perception. Since this methods can neither be formalized nor measured mathematically. An empirical basis for research should be provided.

Acknowledgment

Special thanks to Dr. Heithem Abbes for his fruitful remarks and help. We would like to thank EumedGrid Support that allows us to run our scripts in their infrastructure.

References

[1] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 151–160, 1986.

[2] P. Ekman, W. V. Friesen, and J. C. Hager, *Facial Action Coding System*. Consulting Psychologists Press, 1978.

[3] I. S. Pandzic and R. Forchheimer, *MPEG-4 facial animation: the standard, implementation and applications*. John Wiley and Sons, 2002, vol. 13, no. 5.

[4] A. Moreno and A. Sanchez, "Gavabdb: A 3d face database," *Proc 2nd COST Workshop on Biometrics on the Internet Fundamentals Advances and Applications*, p. 7582, 2004.

[5] B. Fleming and D. Dobbs, *Animating facial features and expressions*, ser. Charles River Media graphics. Charles River Media, 1999, vol. 1.

[6] I.-C. Lin, J.-S. Yeh, and M. Ouhyoung, "Extracting realistic 3d facial animation parameters from multiview video clips," *IEEE Comput. Graph. Appl.*, vol. 22, no. 6, pp. 72–80, November 2002.

[7] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3d mesh segmentation methodologies for cad applications," *Computer Aided Design And Applications*, vol. 4, no. 6, pp. 827–841, 2007.

[8] A. Shamir, "A survey on mesh segmentation techniques," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1539–1556, 2008.

[9] A. Pichler, R. B. Fisher, and M. Vincze, "Decomposition of range images using markov random fields," *Proceedings of International Conference on Image Processing*, pp. 1205–1208, 2004.

[10] I. Ravyse and H. Sahli, "Facial analysis and synthesis scheme," *Advanced Concepts for Intelligent Vision Systems Proceedings*, vol. 4179, pp. 810–820, 2006.

[11] T. DOrazio, M. Leo, G. Cicirelli, and A. Distante, "An algorithm for real time eye detection in face images," *Proceedings of the 17th International Conference on Pattern Recognition 2004 ICPR 2004*, pp. 278–281, 2004.

[12] A. Jacquin, A. Eleftheriadis, T. B. Laboratories, and M. Hill, "Automatic location tracking of faces and facial features in video sequences," *Computer*, no. June, 1995.

[13] A. Yilmaz and M. Shah, "Automatic feature detection and pose recovery for faces," *Proceedings of the Fifth Asian Conference on Computer Vision*, no. January, pp. 1–6, 2002.

[14] P. C. Yuen, S. Lao, and M. Kawade, "Robust facial feature point detection under nonlinear illuminations," *Proceedings IEEE ICCV Workshop on Recognition Analysis and Tracking of Faces and Gestures in RealTime Systems*, pp. 168–174, 2001.

[15] L. Mu and Forchheimer, "Automatic extraction of human facial features," *Signal Processing Image Communication*, vol. 8, no. 4, pp. 309–326, 1996.

[16] K.-M. Lam and H. 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, 1998.

[17] C. Xu, T. Tan, Y. Wang, and L. Quan, "Combining local features for robust nose location in 3d facial data," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1487–1494, 2006.

[18] S. Gupta, M. K. Markey, J. K. Aggarwal, and A. C. Bovik, "Three dimensional face recognition based on geodesic and euclidean distances," *Proceedings of SPIE*, vol. 6499, pp. 64 990D–64 990D–11, 2007.

[19] J. Dalong, L. Zhiguo, W. Zhaoqi, and G. Wen, *Animating 3D Facial Models with MPEG-4 FaceDefTables*. IEEE Computer Society, 2002, p. 395.

[20] A. Giuseppe, B. Roberto, K. Kostas, R. Federico, T. Federica, and V. Kevin, "Grid infrastructures as catalysts for development on escience: experiences in the mediterranean," *Bio-algorithms and Med-systems journal*, pp. 23–25, 2007.

[21] T. M. Chan, *More algorithms for all-pairs shortest paths in weighted graphs*. ACM Press, 2007, pp. 590–598.