Recognition of Important Features of Triangulated Human Head Models

Kateřina Kubásková
Supervised by: Ivana Kolingerová

Department of Computer Science and Engineering
University of West Bohemia
Pilsen / Czech Republic

Abstract

Feature detection is often used in geoinformatics or computer graphics. A lot of feature detection methods have been developed. The goal of this work is to find methods suitable to detect features, implement and test them on the triangulated model of human head which is used to create an identikit.

Keywords: feature, curvature, thresholding, triangular model, morphological operators, MLS approximation

1 Introduction

The features of the model usually refer to a region of a part with some interesting geometric or topological properties. The detection of features is automatic and simple for human brain, but not for the computer. There have been developed many methods for recognition of features and they are used in various fields; CAD, NPR, computer vision, computer graphics, geoinformatics or cartography.

The aim of this work is to find methods suitable to detect features, implement and test them on a triangulated model of human head used to create a 3D identikit - a 3D portrait enabling to identify a person. The proposed method should be able to detect the features automatically or manually. Next step is to detect important points - the control points (the corners of eyes, lips, etc.). It is important to detect them, because they are used to deform a human head model or add a texture (see Figure 1). These points have been found manually and it was very time-consuming.

The rest of the paper is organized as follows. Section 2 gives an overview of the needed theory and existing methods. Section 3 presents the proposed methods to detect features on models of human head. Section 4 analyses results and experiments and Section 5 offers conclusions.

2 Theory and state of the art

In this section we briefly describe necessary background needed for the methods that can be used to recognize important features.

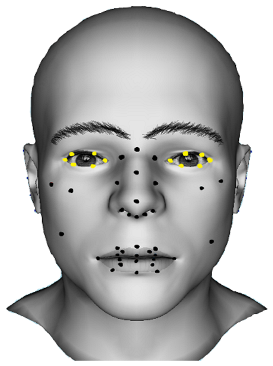

Figure 1: Identikit of a man seen together with control points.

2.1 Differential geometry

At the beginning it is necessary to define a regular surface. Let $D \subset \mathbb{R}^2$ denote an open subset. A smooth vector function $r : D \to \mathbb{R}^3$ of two variables is called parametrization for the surface $S \subset \mathbb{E}^3$ consisting of all points P with $\vec{OP} = r(u,v)$ with $(u,v) \in D$ if

1. r is one-to-one map,
2. the partial derivatives $r_u(u,v) = [x_u(u,v), y_u(u,v), z_u(u,v)]$ and $r_v(u,v) = [x_v(u,v), y_v(u,v), z_v(u,v)]$ are linearly independent at every point $(u,v) \in D$. A subset S that has a parametrization r as above is called a **regular surface** [12].

Next we define the **tangent plane** to the surface $S \subset \mathbb{E}^3$ at P as a plane that contains all tangent vectors to curves on S in a point P. The tangent plane is spanned by the tangent vectors to two parameter curves: $\rho(s,t) = r(u,v) + sr_u(u,v) + tr_v(u,v)$.

A vector $n \in \mathbb{R}^3$ is called a **normal vector** to S at P if n is perpendicular to all tangent vectors $v \in \rho(s,t)$. Let n be a unit normal vector, then
$n(P) = n(u,v) = \frac{r_u(u,v) \times r_v(u,v)}{|r_u(u,v) \times r_v(u,v)|}$ [12].

An important property that characterizes a surface is curvature because it measures a local bending. The **normal curvature** ${}^n\kappa(i)$ is defined as the curvature of the curve that belongs both to the surface S and to the normal plane containing both n and unit tangent vector of the curve. The normal curvature can be computed as ${}^{n}\kappa(i) = \ddot{r} \cdot n$, where $\ddot{r}$ is the second derivative of a vector function r parametrized by the arc length. It can be negative, positive or zero [10].

At every point we define two types of directions - principal and asymptotic. Asymptotic directions are the directions with normal curvatures equal to zero. Principal directions are orthogonal directions with maximal or minimal normal curvatures. These two **principal curvatures** are called maximal curvature κ_{max} and minimal curvature κ_{min}.

The principal curvatures determine the range of the curvature of the surface in local neighborhood of a point. We can use the principal direction to compute other curvatures. The **Gaussian curvature** at a point $P \in S$ is obtained by the relation in Equation 1.

$$K(P) = \kappa_{max}(P) \cdot \kappa_{min}(P). \quad (1)$$

The average of principal curvatures is called the **mean curvature** (Equation 2).

$$H(P) = \frac{1}{2}(\kappa_{max}(P) + \kappa_{min}(P)). \quad (2)$$

More information can be found in [1, 10, 12].

2.2 Methods for feature recognition

There are several different representations for surfaces used in computer graphics but triangular meshes seem to be prevalent. Our head models are represented by 3D triangular meshes defined by a graph $G = (V, E, F)$, where V is a set of vertices of the mesh, E is a set of edges, F is a set of triangles. Due to popularity of triangular meshes representation there are many studies on feature detection on them.

There are some classical methods which somehow evaluate (classify) the vertices or edges and threshold them as belonging to a feature or not. Hubeli et al. in [3] describe some classification operators such as second order difference or best fit polynomial. Another classification operators can be the disrete Laplace-Beltrami operator from [5].

A range of robust methods is available to detect features directly from the meshes. Rössl et al. [4] compute curvature and use morphological operators to produce feature lines. Ohtake et al. [7] have developed a technique that is based on computing curvature tensors and their derivatives at each vertex by means of the projection and global approximation of an implicit surface. Their method achieves good results, but the detection process is time-consuming. Yoshizawa et al. [8] used local polynomial fitting of triangulated meshes to estimate curvature tensors and their derivatives and it reduces the computation time. Another reduction of time-complexity of estimation of the curvatures and their derivatives are addressed in [9] by applying the modified moving-least-squares (MLS) approximation directly to the mesh.

Kim et al. [11] have developed a technique based on voting tensor theory, which can handle n-dimensional triangular mesh. Karlíček in [2] tested and compared methods suitable for various classes of geometric objects, including triangulated head models.

3 Proposed method

In the previous section we described some methods that are used for detection of features. We selected and combined those methods that can handle sharp angles between neighbouring triangles and triangulated approximation of smooth surfaces and also work in optimal time-complexity. Our proposed method consists of the following parts:

- Vertex evaluation - for detection of features
- Evaluation thresholding
- Detection of important areas
- Detection of control points

3.1 Vertex evaluation

We experimented with three methods. The first one uses discrete curvature and was chosen because of good experience reported for head models in [2]. The second one uses a modified MLS approximation and the last one is Laplace-Beltrami operator.

3.1.1 Discrete curvature

The triangular model is a piecewise linear function and it is not possible to determine the derivation. We use approximation equations for curvature described in [10].

To compute the discrete curvature, a "mixed area" for each vertex, denoted A_{mixed}, is needed. It is based on Voronoi region defined for non-obtuse triangle. The area of the Voronoi region can be computed as

$$A_{Voronoi} = \frac{1}{8} \sum_{j \in N_i} (cot\alpha_{ij} + cot\beta_{ij})||v_i - v_j||^2,$$

where N_i is the set of 1-ring neighbour vertices of vertex v_i, α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles sharing the edge (v_i, v_j) (see Figure 2a). This expression for the Voronoi area does not hold in case of obtuse angles. If there is an obtuse triangle among the 1-ring neighbours, the Voronoi region either extends beyond the 1-ring or is truncated compared to our area computation.

The mixed area A_{mixed} is computed as follows: for each non-obtuse triangle we use the Voronoi area, for each obtuse triangle we use the midpoint of the edge opposite to the obtuse angle and connect the midpoint to the centers of the adjacent edges. The mixed area is in Figure 2b.

Now when the mixed area is explained, we can express the curvature. Mean curvature is computed by Equation 3:

$$H(v_i) = \frac{1}{4A_{mixed}} \sum_{j \in N_i} (cot\alpha_{ij} + cot\beta_{ij})||v_i - v_j||^2. \quad (3)$$

Gaussian curvature is given by Equation 4:

$$K(v_i) = (2\pi - \sum_{j=1}^{\#f} \theta_j)/A_{mixed}, \quad (4)$$

where θ_j is the angle of the j-th face at the vertex v_i, and $\#f$ denotes the number of faces around this vertex. This operator will return zero for any flat surface.

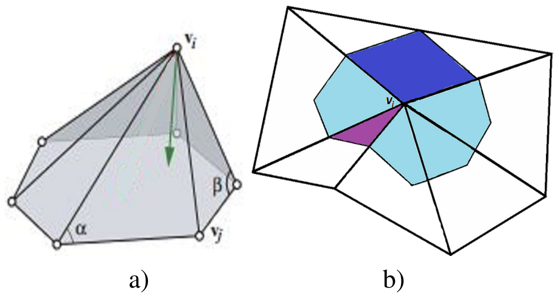

Figure 2: a) 1-ring neighbourhood of vertex v_i. The angles opposite to the edge (v_i, v_j) are α and β. Cotangent (green) Laplacian vectors for vertex v_i [5]. b) Mixed area at vertex v_i. Light blue are the areas in non-obtuse triangles, dark blue is in a triangle with an obtuse angle at the vertex v_i and the purple area in the triangle with an obtuse angle at one of the remaining vertices [2].

We have seen in Section 2.1 that the mean and Gaussian curvatures are easy to express in terms of the two principal curvatures ${}^n\kappa_{min}$ and ${}^n\kappa_{max}$. Therefore we can define the discrete principal curvatures by Equation 5.

$$\begin{aligned} {}^nk_{max}(v_i) = &\quad H(v_i) + \sqrt{H^2(v_i) - K(v_i)}, \\ {}^nk_{min}(v_i) = &\quad H(v_i) - \sqrt{H^2(v_i) - K(v_i)}. \end{aligned} \quad (5)$$

We must make sure that H^2 is always larger than K to avoid numerical problems. If it is not the case than set $\sqrt{H^2(v_i) - K(v_i)}$ to zero.

3.1.2 MLS approximation

In this section we use modified MLS approximation described in [9].

Given a vertex v_i, we first find a local reference plane H. We use a tangent plane orthogonal to a vertex normal at vertex v_i as a local reference plane. Estimation of the normal vector $n = (n_1, n_2, n_3)^T$ at each vertex is done by averaging the normals of a 1-neighbourhood of triangles and the vertex itself. The tangent plane in general form is obtained

$$H : x \cdot n_1 + y \cdot n_2 + z \cdot n_3 = d.$$

Let N_i^k be k-neighbourhood of vertices at vertex v_i, and let $X = \{x_i\}_{i \in N_i^k}$ be the orthogonal projections of the vertices N_i^k to H, represented in a specific orthonormal coordinate system defined on H, so that the origin is v_i. The size of the k-neighbourhood depends on the user. But 1 and 2-neighbourhood generates a poor MLS surface in which it is difficult to locate the principal directions. Adding more neighbors leads to an increased time-complexity. In our experiments we usually use 3-neighbourhood.

Next, we define a local approximation to the surface as the third-degree polynomial p that minimizes the weighted least-squares error given by

$$\mathscr{E} = \sum_{v_j \in \mathbf{N}_i^k} (p(x_j) - f_j)^2 \theta(\|v_i - v_j\|),$$

$$\theta(\|v_i - v_j\|) = e^{-(\|v_i - v_j\|^2/h^2)},$$

where the function θ is Gaussian non-negative weighting. The parameter h is the average of the lengths of the 1-neighborhood edges of v_i and $f_j = n^T v_j - d$ are the heights of the vertices v_j over H.

The error function can be rewritten as

$$\mathscr{E} = \sum_{v_j \in \mathbf{N}_i^k} (b(x_j)^T c - f_j)^2 \theta(\|v_i - v_j\|).$$

where $p(x) = b(x)^T c$, $b(x)$ is the base vector of the polynomial and c is a vector of unknown coefficients. Then we put the partial derivations of the error function equal to zero to find the coefficients of the polynomial and get a linear system of equations given by Equation 6.

$$Ac = d, \quad (6)$$

where $A = \sum_{v_i \in \mathbf{N}_i^k} 2b(x_j)b(x_j)^T, \theta(\|v_i - v_j\|)$ and $d = \sum_{v_i \in \mathbf{N}_i^k} b(x_j) f_j \theta(\|v_i - v_j\|)$. The solution of the linear system is $c = A^{-1}d$.

Now we can estimate the principal curvatures at each vertex v_i. We convert the MLS polynomial $z = p(x_i)$ into the implicit surface $F = z - p(x_i)$.

For each vertex v_i we can estimate the unit normal vector at v_i as $n = \nabla F/(|\nabla F|)$. Next we can estimate the principal curvatures κ in the associated principal directions $t = (t_1, t_2, t_3)$ as follows

$$\kappa = \frac{F_{ij} t_i t_j}{|\nabla F|},$$

where F_{ij} denotes the second partial derivatives of F. Directions t_{max} and t_{min} are given by eigenvectors corresponding to the two non-zero eigenvalues of $\nabla \mathbf{n}$. The matrix $\nabla \mathbf{n}$ is given by Equation 7.

$$\nabla \boldsymbol{n} = \begin{bmatrix} \frac{\partial n_1}{\partial x} & \frac{\partial n_1}{\partial y} & \frac{\partial n_1}{\partial z} \\ \frac{\partial n_2}{\partial x} & \frac{\partial n_2}{\partial y} & \frac{\partial n_2}{\partial z} \\ \frac{\partial n_3}{\partial x} & \frac{\partial n_3}{\partial y} & \frac{\partial n_3}{\partial z} \end{bmatrix}. \quad (7)$$

Mean and Gaussian curvature can be then estimated using principal curvatures by expressions in Section 2.1. Unlike the method in Section 3.1.1, now we can get also negative values.

3.1.3 Laplace-Beltrami operator

The last method for classification of vertices is the Laplace-Beltrami operator [5, 6]. The discrete Laplace-Beltrami operator for a triangular mesh at the vertex v_i is defined in Equation 8.

$$\delta_i = \sum_{(v_i,v_j)\in\mathbf{E}} \omega_{ij}(v_j - v_i), \tag{8}$$

where $\sum_{(v_i,v_j)\in\mathbf{E}} \omega_{ij} = 1$ and the choice of weights defines the character of δ_i.

We choose the cotangent weights $\omega_{ij} = \frac{cot\alpha + cot\beta}{2}$, where α and β are the angles opposite to the edge (v_i, v_j) (see Figure 2a). The cotangent Laplacian is zero on planar 1-rings because of geometry-dependence. The cotangent Laplacian vector can be seen in Figure 2.

The cotangent Laplace-Beltrami operator is dependent on the size of triangles, therefore, for evaluation of vertices we have to use a modified cotangent Laplacian, which is defined in Equation 9.

$$\delta_\mathbf{i} = \frac{1}{d_i} \sum_{(v_i,v_j)\in\mathbf{E}} \omega_{ij}(v_j - v_i), \tag{9}$$

where $d_i = \frac{S_i}{3}$ and S_i is the area of the adjacent triangles at vertex v_i. The discrete Laplace-Beltrami operator is a vector, therefore, we use the size of the vector for evaluation of vertices.

It is also possible to use truncating of evaluation. A percentage of vertices with the highest evaluation are selected and they get a new value, the highest evaluation without already selected vertices.

3.2 Thresholding

After we have classified vertices using the described methods, we apply a standard thresholding to evaluate the vertices. The user specifies the threshold parameter as minimal weight that a vertex must have to be included into the subset of feature vertices. In case of the curvature computed using MLS approximate we have also negative weights; so in negative thresholding the user specifies a maximal weight.

We normalize the values of evaluation using the Equation 10.

$$w(v_i) = \frac{w(v_i)}{|w_{max}|} \cdot 100 \quad [\%], \tag{10}$$

where $w(v_i)$ is either the curvature from Section 3.1.1 or 3.1.2 or the size of cotangent Laplacian from Section 3.1.3 and $|w_{max}|$ is the absolute value of maximum evaluation of all vertices.

In agreement with recommendation in [2] we can also apply morphological operators on feature vertices to achieve better results. Morphological operators deal only with binary values. So we use a thresholding operation to determine the feature vector Ω:

$$\Omega_i = \begin{cases} 1 & for\, w_i \in [a,b] \\ 0 & \text{otherwise,} \end{cases}$$

where w_i is evaluation at vertex v_i and a, b are thresholding parameters. A set $\mathscr{F}_s$ of significant vertices can be expressed as $\mathscr{F}_s := \{v_j \in \mathbf{F} \,|\, \Omega_j = 1\}$.

The morphological operators for triangulated meshes are defined as follows.

Dilation

Let $\mathscr{F}_s \subseteq \{1, \cdots, N\}$. The dilation of $\mathscr{F}_s$ by k-neighbourhood $\mathbf{N}^k$ is defined as.

$$\text{dilate}^k(\mathscr{F}_s) := \{v_j \,|\, \exists v_i \in \mathscr{F}_s : v_j \in \mathbf{N}_i^k\}.$$

The dilation operator adds vertices to the feature. It can therefore be used to fill ”holes” in the features.

Erosion

Let $\mathscr{F}_s \subseteq \{1, \cdots, N\}$. The erosion of $\mathscr{F}_s$ by k-neighbourhood $\mathbf{N}^k$ is defined as

$$\text{erode}^k(\mathscr{F}_s) := \{v_j \,|\, \mathbf{N}_j^k \subseteq \mathscr{F}_s\}.$$

The erosion operator reverses the effect of dilation. It cuts off undesired branches.

Opening

The opening operator is defined as

$$\text{open}^k(\mathscr{F}_s) = \text{dilate}^k(\text{erode}^k(\mathscr{F}_s)).$$

The opening operator applies the erosion and after it applies the dilation. This application removes undesired artifacts, but the size of the feature is not preserved.

Closing

The closing operator is defined

$$\text{close}^k(\mathscr{F}_s) = \text{erode}^k(\text{dilate}^k(\mathscr{F}_s)).$$

In closing $\mathscr{F}$ the feature is first grown and shrinked afterwards. It fills holes in the inner region of the feature and fills bays along the boundary.

3.3 Detection of important areas

Our aim is mainly to detect features automatically. In this case, we have to find a suitable threshold that gives the best result. One minimal threshold does not achieve a good result on the whole model. Due to this we decided to detect important areas, which include the entire area of eyes, ears, nose and lips. Then we detect features and by applying different thresholds in different areas we get better results.

We based the detection of important areas on the simplest from the presented vertex evaluation approaches, i.e. the mean curvature from Section 3.1.1. The minimal threshold is selected as an average of evaluations. In Figure 3 we can see the resulting features. Then morphological operators are applied: closing operator, the opening operator, another closing operator and a dilation. The results of this algorithm are dependent on the choice of k-neighbourhood and the number of vertices N.

From experimental results we chose the following k-neighborhood:

- $N \leq 13000 \Rightarrow k = 5$,
- $13000 \leq N \leq 20000 \Rightarrow k = 10$,
- $N > 20000 \Rightarrow k = 15$.

If an automated processing is not required, the user may select whether the last operator, the dilation, is applied and chooses its k-neighborhood.

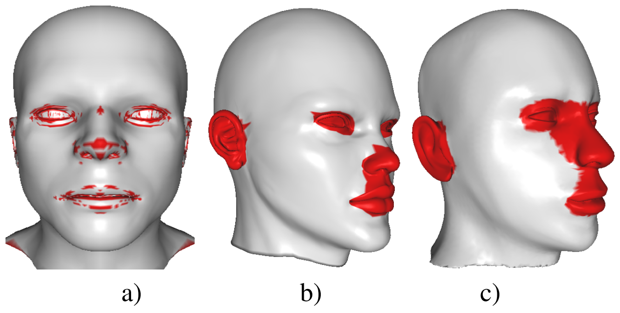

Figure 3: Detection of important areas: a) Features after evaluation by mean curvature and thresholding by average of evaluations. Detection of important areas with b) 15, c) 10 - neighbourhood.

Now the important areas on the head are detected. The area of ears is already separated but the rest is so far connected which is not what the user needs.

To separate these areas, two methods can be applied. The first method uses a simple distance based selection. The boundary between eyes and nose is in the half of distance between the point with maximal x-coordinate and the peak of the nose (maximal y-coordinate). The boundary between the nose and the lips is in the two thirds of the distance between the peak of the nose and the point with the minimal z-coordinate (see Figure 4).

The second method is based on the first, but the detection of the nose is modified. The area of the nose contains all vertices in the marked region in Figure 5. This region is formed by the nose root (the lowest point on the line of the nose), the boundary between the eyes and the nose and the width of the nose.

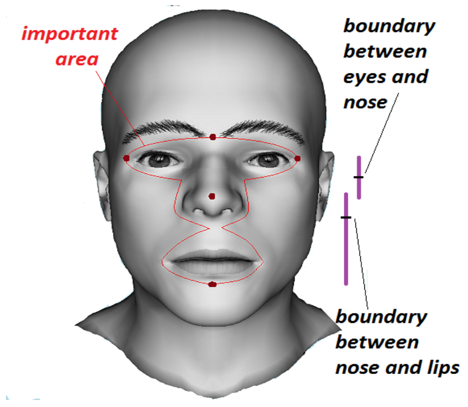
important area
boundary between eyes and nose
boundary between nose and lips

Figure 4: First method of separation of areas.

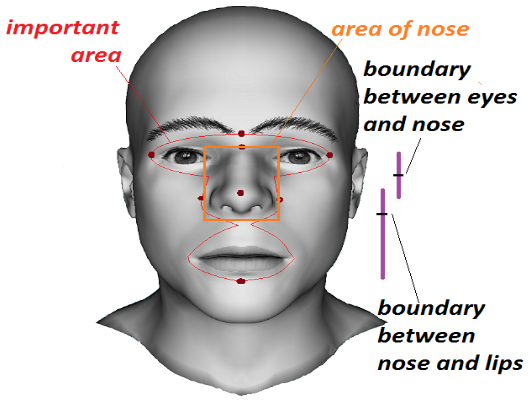
important area
area of nose
boundary between eyes and nose
boundary between nose and lips

Figure 5: Second method of separation of areas.

3.4 Detection of control points

Next goal is to try to find the control points - the most important points of the feature. By the feature detection we get the border of the lips, eyes and ears, then detection of the control points is not difficult as described further.

Eyes On the eyes we need to detect the corners and 8 points. The corners of an eye have extreme x-coordinates (see Figure 6).

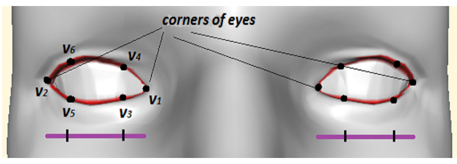

Figure 6: Detection of control points on eyes. Features (red) and control points (black).

Lips At first we detect the corners of lips (v_1, v_2) as extreme x-coordinates. Using these two points we detect other 12 points. Some of them are on the border of lips and some of them in the middle of the lips (see Figure 7).

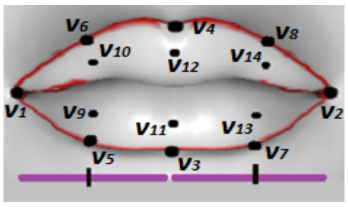

Figure 7: Detection of control points on lips. Features (red) and control points (black).

Ear We detect 5 points on the ears (see Figure 8a) First two points (v_1, v_2) have extreme z-coordinates. Other two points (v_3, v_4) lie in the first quarter of the distance from v_1 or v_2 and has the minimal y-coordinate. The last point v_5 lies in the middle of the ear.

Nose The detection on the nose is the most difficult task. Some of the points have to be detected outside of the set of features. The tip v_1 of the nose is found by the maximal y-coordinate, the nose root v_2 have minimal y-coordinate on the line of nose. The point v_3 lies in the middle between v_1 and v_2. Other 5 points are detected in the features using the already known points, details see [1]. The detection can be seen in Figure 8b).

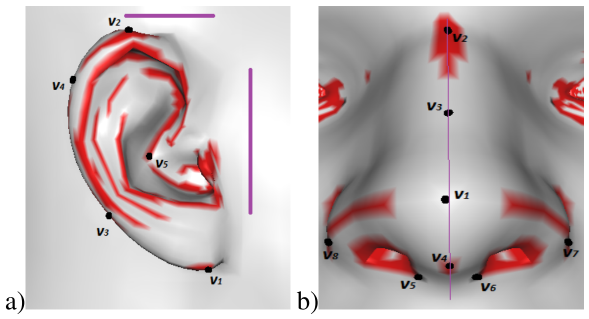

Figure 8: Detection of control points: a) on ears, b) on nose. Features (red) and control points (black).

4 Experiments and results

In this section we show results of the proposed methods. The method was implemented in Microsoft Visual Studio 2010 using the programming language C# and .Net Framework version 4. The program, where the method from Section 3.1.1 and morphological operators were implemented, was taken from [2]. We added other methods described in Section 3.1.2 and 3.1.3 and algorithms for automatic detection, detection of important areas and detection of control points.

We tested our proposed methods on 11 models. Each method works on various areas of the head differently. The choice of thresholds is important. First, we evaluate the results of curvature computation described in Section 3.1.1, then our implemented methods.

4.1 Discrete mean and maximal curvature

As minimal and Gaussian curvature do not achieve good results we present only results of mean and maximal curvature (see Figure 9).

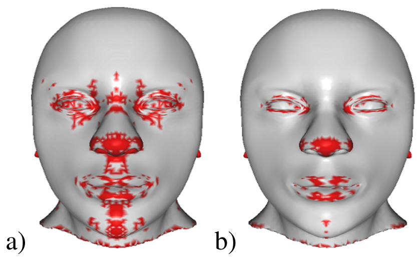

Figure 9: Detection of features using: a) minimal curvature from Section 3.1.1, b) Gaussian curvature from Section 3.1.2.

Mean curvature provides very good results on the area of eyes and ears. We get the border of eyes and ears. On the lips and nose the results of mean curvature are not very satisfactory. On a few models we get the border of the lips, on the nose we get only features around nostrils (see Figure 10 a).

Results of maximal curvature are similar to the mean curvature. On the eyes and ears we have also good results and even on the lips are the results quite good. With nose we have also problems and have only features around nostrils. The results can be seen in Figure 10 b).

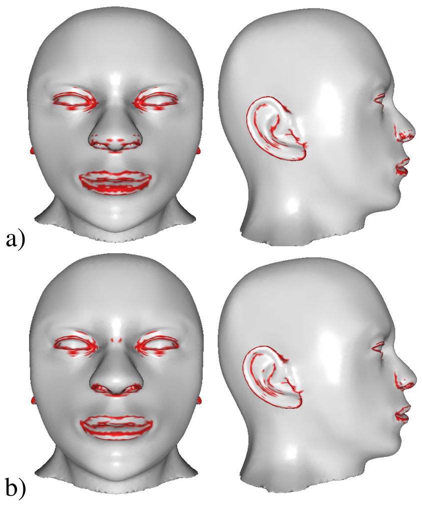

Figure 10: Detection of features: a) mean curvature, b) maximal curvature.

4.2 Mean and maximal curvature computed using MLS

Let us recall that evaluation computed by this method have also negative values, therefore, the color of features in figures is different than before (blue for negative and red for positive values).

The result of mean and maximal curvature are very satisfactory on the area of eyes and ears. On the lips the maximal curvature is better than the mean curvature. The area of the nose is complicated. Using maximal curvature one can detect the bridge of nose on some models. On other models again only the nostrils are detected. The results can be seen in Figure 11.

4.3 Cotangent Laplace-Beltrami operator

The definition of cotangent Laplace-Beltrami operator is very similar to the mean curvature in 3.1.1. Therefore, the results of these methods are not too different (see Figure 12).

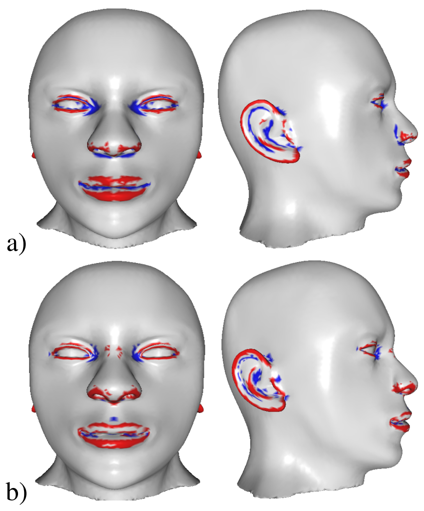

Figure 11: Detection of features: a) mean curvature computed using MLS, b) maximal curvature computed using MLS.

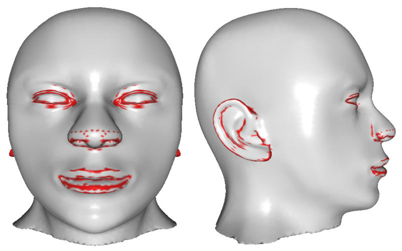

Figure 12: Features using Laplace-Beltrami operator.

4.4 Detection of important areas

The detection of important areas works on all models very well. In case that the important areas are too small for the user's needs, dilation can be applied (see Figure 13).

Next, we show the result of selection of important areas. We can see the difference of two methods in Figure 14. Using the first method, we do not get all necessary vertices in the area of nose (see Figure 14b). The second method fixes this failure and we get the whole area of nose in all models (see Figure 14c,d). It is possible to apply dilation or erosion on each important area if the user is not satisfied with the result achieved so far.

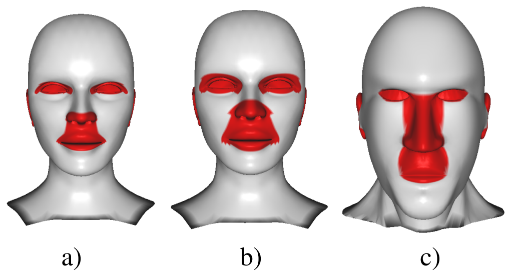

Figure 13: Detection of important areas: a,c) the dilation is not used, b) dilation in 3-neighborhood.

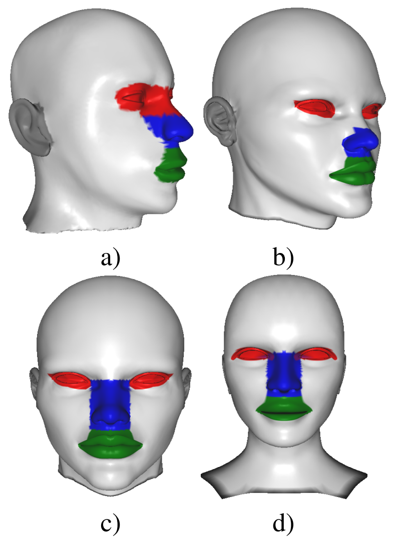

Figure 14: Result of selection of important areas: a,b) the first method, c,d) the second method.

4.5 Automatic detection

Our aim is the automatic detection of features, but for this we need to know a suitable configuration of parameters. It is impossible to determine an optimal threshold that will work on all heads. Therefore, for each area we chose the method that generally works well on this area for most tested models.

Our proposed automatic detection is following:

- Eyes: Maximal curvature from MLS approximate
- Ears: Mean curvature from MLS approximate
- Lips: Mean curvature with 5% values truncated
- Nose: Maximal curvature

The results of automatic detection of features are shown in Figure 15. The automatic detection is very good in 75 percent of models.

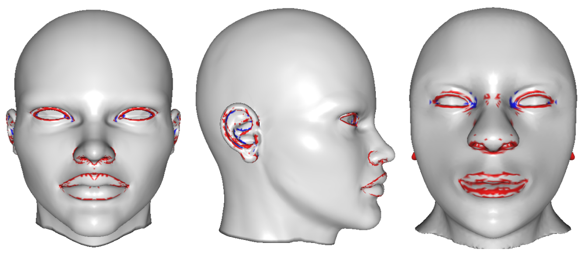
Figure 15: Automatic detection

4.6 Detection of control points

The last group are results of the detection of control points. Automatic detection of control points after the automatic detection of features is shown in Figure 16. These results are very good.

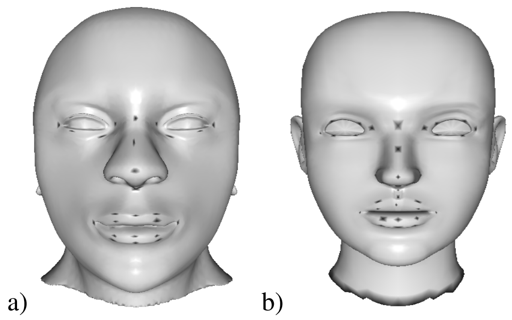

Figure 16: Detection of control points.

5 Conclusion

In this paper we have presented methods to detect important features on triangulated human head models. The aim of this work was to propose automatic detection and detection with manuall choice of parameters.

We used three methods. We detected important areas using the method described in Section 3.1.1. To classification of vertices we then used Laplace-Beltrami operator (Section 3.1.3) and estimation curvature computed using MLS approximate (Section 3.1.2). Then we get the features by applying thresholding and we could use morphological operator to modify the features. We can also detect the features on each important area. In automatic detection we used methods which have the best results on the important area and the minimal threshold parameter is determined by an average of evaluation on this area.

The proposed method was tested on several models used for 3D identikits. The automatic method achieves a good result on most models. The worst results are in the area of the nose. Future development should consider other methods working well automatically also in the area of the nose. Detection of control points has satisfactory results, but there are based on results of the features.

6 Acknowledgement

I would like to thank Ivana Kolingerová for her support and excellent leadership in this project and to Petr Martínek for his help and consultations.

References

[1] K. Kubásková, *Feature recognition on triangulated models of human head*. Pilsen, 2015. Bachelor thesis, Faculty of Applied Sciences, University of West Bohemia (In Czech).

[2] L. Karlíček, *Feature recognition on triangulated models*. Pilsen, 2014. Master thesis, Faculty of Applied Sciences, University of West Bohemia (In Czech).

[3] A. Hubeli, K. Meyer, M. Gross, *Mesh Edge Detection*. CS Technical Report. ETH: Institute of Scientific Computing, vcarsko, 2000.

[4] Ch. Rössl, L. Kobbelt, H.-P. Seidel, *Extraction of Feature Lines on Triangulated Surface Using Morphological Operators*. Smart Graphics, AAAI Technical Report SS-00-04, 2000.

[5] A. Nealen, T. Igarishi, O. Sorkine, M. Alexa, *Laplacian Mesh Optimization*. GRAPHITE '06, p. 381-389, ISBN:1-59593-564-9, 2006.

[6] O. Sorkine, *Laplacian Mesh Processing*. The Eurographics Association, p.53-70, 2005.

[7] Y. Ohtake, E. Belyaev, H.-P. Seidel, *Ridge-Valley Lines on Meshes via Implicit Surface Fitting*. Journal ACM Transactions on Graphics, 2004, vol. 23, no. 3, p. 609-612.

[8] S. Yoshizawa, A. Belyaev, H.-P. Seidel, *Fast and Robust Detection of Crest Lines on Meshes*. ACM Symposium on Solid and Physical Modeling, 2005, p.227-232, ISBN: 1-59593-015-9.

[9] S.-K. Kim, Ch.-H. Kim, *Finding ridges and valleys in a discrete surface using a modified MLS approximation*. Computer-Aided Design, 2006, vol. 38, no. 2, p. 173-180.

[10] M. Meyer, M. Desbrun, P. Schrder, A. H. Barr, *Discrete Differential Geometry Operators for Triangulated 2-Manifolds*. Visualization and Mathematics III, 2003, p. 35-57, ISBN: 978-3-662-05105-4.

[11] H.S. Kim, H.K. CHoi, K.H. Lee, *Feature Detection of Triangular Meshes Based on Tensor Voting Theory*. Computed Aided Design, 2009, vol. 41, no. 1, p. 47-58.

[12] M. Raussen, *Elementary Differential Geometry: Curves and Surfaces*. Department of mathematical sciences, 2008, Aalborg University, Denmark.