

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Visualization techniques for 3D facial comparison

MASTER'S THESIS

Bc. Katarína Furmanová

Brno, Spring 2015

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. Katarína Furmanová

Advisor: RNDr. Igor Chalás

Acknowledgement

I would like to thank my supervisor RNDr. Igor Chalás for his guidance. My thanks also goes to Bc. Zuzana Ferková for her help and feedback and to anthropologists from Laboratory of Morphology and Forensic Anthropology at Masaryk University for their expertise and advice.

Abstract

Many areas where facial analysis is used – such as criminal identification or authorization software – are nowadays thanks to technological advances quickly moving from 2D image representation to 3D. However, with higher dimensionality of the data, also the challenges for its visualization increase. How to visualize more than one facial surface without facing occlusions or losing track of data adherence? How to encode the measurements and visualize them to best convey their meaning? How to easily identify correlations between data? My aim in this work is to deal with some of these challenges and extend the visualization possibilities of FIDENTIS Analyst application – application for analysis of 3D facial data. I present three different visualization techniques, each targeted on different issue, in order to provide complex visualization toolbox which could be used not only to visualize the results of the analysis but aid the process itself.

Keywords

Comparative visualization, Facial analysis, Visual analysis, Nested surfaces, Rendering pipeline, Cross section, Heat plot, OpenGL, GLSL, Java 2D, FIDENTIS

Contents

Introduction	1
1 Analysis of current state	3
1.1 <i>The FIDENTIS project</i>	3
1.1.1 Pair comparison	4
1.1.2 Comparison with database	5
1.1.3 Batch processing	6
1.2 <i>Shortcomings of currently used visualizations</i>	7
2 Related work	9
2.1 <i>Order Independent Transparency</i>	9
2.2 <i>Opacity modulations</i>	10
2.3 <i>Cross-sections and contours</i>	11
2.4 <i>Color coding and textures</i>	12
2.5 <i>Interactive visual analysis</i>	13
3 Selected visualization techniques	15
3.1 <i>Surface superimposition</i>	15
3.1.1 Transparency	15
3.1.2 Fog simulation	16
3.1.3 Shadow-casting curvature glyphs	19
3.1.4 Intersection contours	19
3.2 <i>Cross sections</i>	20
3.3 <i>Plots</i>	22
3.3.1 Numerical results	22
3.3.2 Auxiliary results	24
4 Implementation	26
4.1 <i>Surface superimposition</i>	26
4.1.1 Data preprocessing	27
4.1.2 Depth map creation	29
4.1.3 Shading	31
4.1.4 Rendering	34
4.1.5 Contouring	37
4.2 <i>Cross sections</i>	37
4.3 <i>Plots</i>	39
5 Results and evaluation	42
5.1 <i>Surface superimposition</i>	42
5.2 <i>Cross sections</i>	46

5.3	<i>Plots</i>	46
5.4	<i>Summary</i>	47
	Conclusion	49
	Bibliography	51
	Appendices	55
A	<i>Questionnaire</i>	56
B	<i>Questionnaire results</i>	69
C	<i>User guide</i>	73
D	<i>Thesis Archive in IS MU</i>	78

*"A picture shows me at a glance what it takes
dozens of pages in a book to expound."*

– Ivan Turgenev

Introduction

Visualization plays important role in analyzing, exploring and presenting data. But what exactly is a visualization? There are many definitions. One says it is the use of computer-generated, interactive, visual representations of data to amplify cognition [7]. Another says it is a transformation of symbolic into geometric [22]. From scientific point of view it is a relatively newly recognized field, although one may argue, that visualization is as old as humanity, or at least as old as paintings in the Chauvet Cave. Nevertheless, most definitions agree upon one thing – the purpose of visualization is to provide a user a better insight into data.

Even in the short time that visualization has been around as a recognized scientific field, it has evolved immensely, no doubt driven by enormous quantities of new data that increase in volume every year. These data need to be processed and analyzed effectively, otherwise they may lose their value. However, visualizing large amounts of data is often challenging, especially for multivariate data spanning across several dimensions.

In my work I am going to explore the field of 3D facial data comparison and analysis. Facial analysis is used in many areas – such as criminal identification, authorization software and plastic surgery as well as medical and anthropological research. Until recently, many of these areas worked only with 2D image representation. However, as technology evolves, 3D imaging becomes more and more common. New dimension means new data, but it also poses new challenges for visualization of these data. How to visualize more than one facial surface without facing occlusions or losing track of data adherence? How to encode the measurements and visualize them to best convey their meaning? How to easily identify correlations between data?

My aim in this work is to overcome some of these challenges in order to extend the visualization possibilities of FIDENTIS Analyst application, which is a part of interdisciplinary collaborative project FIDENTIS. The project is focused on analysis of 3D facial data and strives to provide a compact toolbox for this purposes.

In the first chapter I am going to introduce the FIDENTIS project in closer detail. I will describe its aim and parts dedicated to com-

parison in order to analyze the needs of the application in terms of visualization.

The second chapter is dedicated to research of related work in the field of visualization and visual analysis of 3D surfaces. I will examine several techniques in order to pick or design the best solution for my case of study.

In third chapter I am going to present the chosen techniques and in fourth chapter I will describe their implementation.

In the final fifth chapter I will present the results of my work and evaluate the contribution of my work based on the user study among scientist from the field of facial analysis.

1 Analysis of current state

The aim of my work is to extend the currently used visualization techniques in FIDENTIS Analyst application, which is a part of FIDENTIS project¹ dedicated to facial analysis and research. To provide background for my work I will describe the aim of the project and the current state of the application. I will also analyze the used visualizations in order to identify the space for improvement and I will present the results of discussion with anthropologists on this topic.

1.1 The FIDENTIS project

FIDENTIS is an interdisciplinary collaboration project between Laboratory of Morphology and Forensic Anthropology at Faculty of Science, Masaryk University, and Human Computer Interaction Laboratory at Faculty of Informatics, Masaryk University, targeting research of human face with primary focus on collection and analysis of 3D data. The project consists of two parts – FIDENTIS Database and FIDENTIS Analyst.

The FIDENTIS 3D Face Database, originally published in [20], represents large dataset of 3D facial scans acquired via Vectra M1, H1 or XT imaging systems. In order to prepare data for research, the scans are pre-processed (removal of artifacts, reconstruction from more scans, etc.) by scientists before inclusion in database. The database also contains basic demographic and morphometric data. It currently consists of approximately 2,000 specimen. During my work I have used data from this database to test my results.

FIDENTIS Analyst is an application developed for morphological analyses of 3D facial models. It contains several features designed for forensic purposes, such as 3D facial composite construction, automated landmark localization and analysis of facial morphological variation. To meet various demands of facial analysis, three different modes of model comparison were developed – Pair compar-

1. More information about the FIDENTIS project can be found at <http://fidentis.cz/>.

ison, Comparison with database and Batch processing. Moreover, two comparative techniques were included in each mode – landmark based variations of Procrustes analysis [17, 15] and surface based variations of nearest neighbor search [9].

Each of the modes serves different purpose and the comparative techniques used in the application work with different data structures. Proper visual representation of comparison results is therefore crucial for accurate understanding and interpretation. In following sections I will discuss the modes of application as well as the currently used visualizations to analyze their purpose and space for possible improvement.

1.1.1 Pair comparison

The Pair comparison mode serves for comparison of two models marked as primary and secondary. One of the goals of this method is to determine whether the two models depict the same person. Another use case is analysis of differences between the two models, e.g. for analysis of facial changes during aging process or facial variations in siblings. For these purposes it is important that the final visualization shows global differences, such as size and overall shape, as well as local differences, such as varying width of the mouth or shape of the nose.

As for the used visualization, the superimposition principle was applied here for both landmark-based and surface-based comparison. The comparison process runs in two steps – firstly the secondary model is aligned to the primary model, then the differences are computed. In case of landmark-based comparison, the models are aligned using only landmark configurations, whereas the surface-based comparison applies Iterative Closest Point algorithm [4] to entire meshes to register the models. After the registration, the models are drawn on top of each other with optional transparency, so the user can verify the alignment. However, rendering transparent intersecting objects via fixed OpenGL pipeline results in undesired artifacts, as can be seen in Figure 1.1 (a). On the other hand, rendering them with full opacity hides important details.

The registration is followed by comparison itself. Here the resulting visualization differs based on the comparison method selected.

For landmark-based comparison the two landmark configurations are overlaid and the landmarks are connected to approximate facial shape – Figure 1.1 (b). Distance enhancement between corresponding landmarks was implemented to provide means to analyze even small local differences. Surface-based comparison computes distance from each vertex of secondary model to the nearest vertex of primary model. These distances are then mapped on secondary model in a form of color map – Figure 1.1 (c). This illustrates the differences between models nicely, however since the result is mapped on the secondary model, the shape of primary model is not obvious.

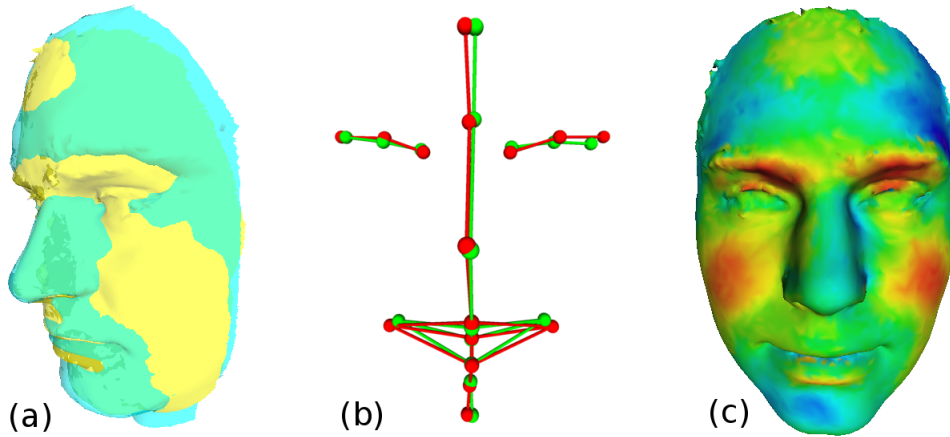


Figure 1.1: Pair comparison. (a) Superimposed models after registration with visible artifacts caused by improperly handled transparency. (b) Landmark configurations after Procrustes analysis. (c) Color map showing results of surface-based comparison.

1.1.2 Comparison with database

The Comparison with database mode of application serves to identify the closest match to given model (referred to as primary model) in specified dataset, e.g. comparing a model of suspect with database of criminals. It is also used for analyzing the difference between the primary model and the dataset. The same work-flow as in Pair comparison is followed here – first registration, then comparison, however, there is no visualization for verifying the registration of more

than two models. The landmark-based comparison is displayed as a mean landmark configuration computed from dataset, with vectors displaying deviation of primary model from this mean – Figure 1.2 (a). The visualization of surface-based comparison is again color map, however, this time it is mapped on average model computed from dataset and it shows the minimal distance from the vertices of average model to the vertices of primary model.

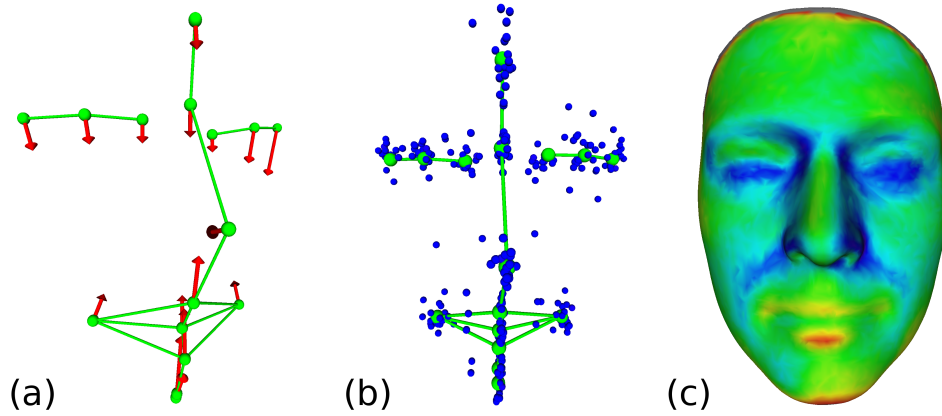


Figure 1.2: (a) Visualization of landmark-based Comparison with database. (b) Visualization of landmark-based Batch processing. (c) Color map on average model showing results of surface-based Batch processing.

1.1.3 Batch processing

The Batch processing is a mode designed for research on large sets of models. The example of typical use case of this mode is analysis of facial variations of given population. The work-flow consists of same steps as in previous modes. For landmark-based comparison all landmark configurations are displayed with mean configuration highlighted – Figure 1.2 (b). As for surface-based comparison, average model is computed and for each vertex of the average model distances to the nearest vertices of each model in dataset are computed. The final per-vertex value is then calculated from these distances using one of the following techniques: Root Mean Square, Arithmetic

Mean, Geometric Mean, Variance, Minimal Distance, Maximal Distance or 75 Percentile. The resulting values are again displayed as color map on the average model – Figure 1.2 (c).

1.2 Shortcomings of currently used visualizations

In preparation for my work I met several times with anthropologists to discuss the biggest drawbacks of currently used visualizations, as well as the areas they would like to visualize and explore in greater detail. There are several tasks they need to perform within the application, where various visualization techniques would be beneficial: aligning the models or verifying the results of automatic alignment, analyzing shape of the models, analyzing local variability and shape differences of models, comparing models – pairwise or within a set, analyzing variability of entire dataset, etc.

When discussing how visualizations could aid these tasks and what the currently used ones are missing, three most notable points concerning especially surface-based comparison came up:

- **Lack of shape information.** The currently used color map mapped on one model is suitable for large datasets where average model is used as basis, however the shapes of the models in dataset are lost to the observer. Therefore, at least in case of pair comparison it would be beneficial to preserve and illustrate the shape of both models.
- **Lack of local information.** When processing larger sets of data, the color map is computed on global level – the final color at one vertex of average mesh is always dependent on the scale computed from all the values (scale from minimal to maximal detected value). There is no possibility to limit the presented information only to specific area and display it on local scale.
- **Limited view of data.** There is basically only one type of visualization for each comparative method – color map for surface-based methods and landmark visualizations for landmark-based methods. There isn't any analysis tool or additional visualization for numerical data as is typical for software dedicated

1. ANALYSIS OF CURRENT STATE

to data analysis. One must process the numerical data in an additional software, if one wishes to analyze the relationship between data in larger dataset more closely.

2 Related work

Tools dedicated to facial analysis and comparison typically contain only the visualization techniques similar to these already implemented in FIDENTIS Analyst application – color maps and landmark-based visualizations. Therefore, based on the analysis of drawbacks of these techniques and the typical use cases of the application I conducted a research in other areas of 3D data visualization, comparative visualization as well as visualization and visual analysis of cohort study data that could be applicable in these use cases. According to [16], there are three main approaches to visual data comparison: juxtaposition, superposition and explicit encoding. Juxtaposition is a method, which positions the objects next to each other and therefore is not suitable for comparing larger sets of data. The superposition approach is used to place objects one on top of the other and presenting them in the same time and space. This approach is also unusable for displaying larger sets of 3D objects, but is applicable in case of their subsets or cross-sections. The explicit encoding depicts the relationships between objects by encoding them visually and is therefore most suitable for large datasets. In following sections I will discuss several approaches that use both superimposition and explicit encoding principles.

2.1 Order Independent Transparency

When using superimposition principle on 3D surfaces, transparency plays an important role, since full opacity often covers important details. Unfortunately fixed OpenGL pipeline offers very little control over transparency and requires sorting the geometry in order to achieve correct results. With complex and intersecting objects this is difficult or downright impossible. Therefore several approaches to achieve so called Order Independent Transparency were introduced – Depth Peeling, Dual Depth Peeling, Approximative techniques or Concurrent Linked Lists.

Depth Peeling [13] is technique that works by front-to-back peeling of geometry layers. It is iterative method that requires several passes over geometry. In each pass a depth-map for the front layer is

created so in next iteration anything with lower or equal depth can be "peeled". The algorithm blends the colors acquired in each layer with image acquired from previous layers.

Dual Depth Peeling [2] is an extension of original Depth Peeling algorithm which allows simultaneous peeling of two layers. Since default OpenGL depth buffer doesn't allow testing needed for this, custom depth buffer is necessary for this method and the default buffer is turned off.

Approximative techniques, such as ones presented in [24, 2, 23], try to avoid the necessity of sorting fragments by adjusting the compositing operator of alpha blending so that it is order independent.

All of the aforementioned techniques are limited in some way, either requiring multiple passes of geometry or providing only approximative results. They are also difficult to extend and modify in order to achieve context dependent results. A solution to this drawbacks provide Concurrent or Per Pixel Linked Lists (PPLL) that are constructed on GPU [30]. For each pixel a list of fragments is created, each fragment pointing to its neighbors along viewing ray. After the creation of such lists, the task of computing the result by blending in correct order is trivial.

2.2 Opacity modulations

However, even after solving the problem of correct transparency rendering, the interpretation and understanding of resulting image containing transparent surfaces is often difficult task. Busking et al. [6] try to solve these issues by introducing a layered rendering pipeline for image-based rendering of intersecting surfaces. In their work they describe several techniques that enhance the understandability of transparently rendered surfaces and distances between them using glyphs, shadow-casting or fog simulation – see Figure 2.1 for examples of their results.

Other techniques try to improve the understandability of transparent surfaces purely by modifying transparency values. Here belong angle-based transparency [18] – setting the transparency to the angle between surface normal and viewing direction – or normal variation transparency [5]. Another technique introduced in [8] uses

geodesic fragment neighbors for transparency modulation and contour rendering. For comparison of these techniques see Figure 2.2.

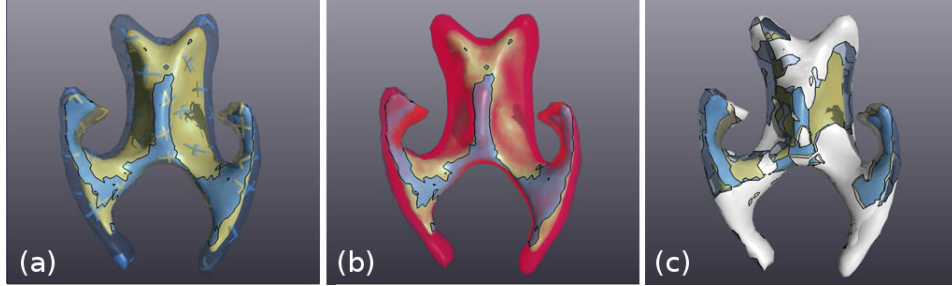


Figure 2.1: Visualization of intersecting surfaces introduced in [6]. (a) Shadow-casting glyphs. (b) Fog simulation. (c) Relevance filtering.

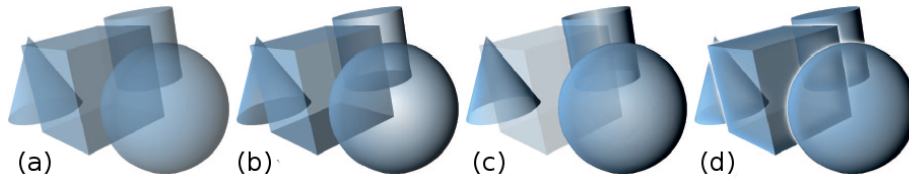


Figure 2.2: Transparency modulations according to [8]. (a) Constant transparency. (b) Angle-based transparency. (c) Normal variation transparency. (d) Geodesic fragment neighbors-based transparency.

2.3 Cross-sections and contours

When using superimposition or juxtaposition principles, displaying big sets of 3D data all at once is ill-advised, because it often causes occlusions and visual clutter, which defeats the primary purpose of good visualization. A possible solution to this is cross-sectional view, an approach widely used in medicinal visualization for volumetric data, e.g. CT images, where a slice along given plane is projected into 2D space – see Figure 2.3 for example.

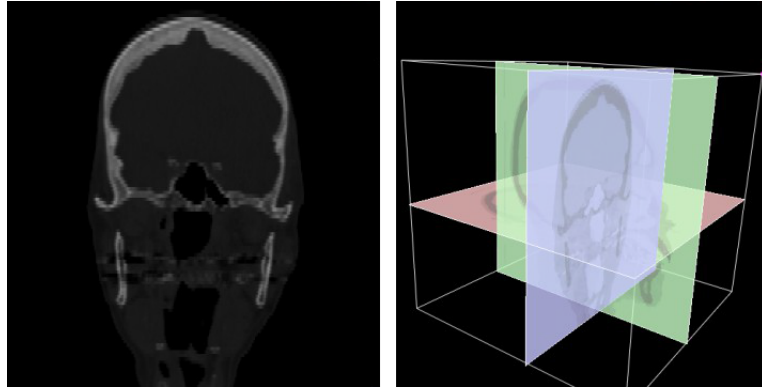


Figure 2.3: Example of cross-sectional visualization in YaDiV software [14].

A similar approach to this is contouring of specific object features followed by projection of contour into 2D space. This is often used when monitoring temporal changes of given feature – such as cell size or width of molecular tunnel. Contours are also inseparable part of geospatial data visualization.

2.4 Color coding and textures

According to [3] color is one of the seven basic visual variables. It is therefore not surprising, that it is often used for explicit encoding of information, especially in case of 3D surfaces, where some of the other variables (size, shape, etc.) may not be applicable. I have described in previous chapter how color encoding is used in FIDEN-TIS Analyst application. Naturally, there are many other applications which use this principle for encoding data – e.g. YMCA [26] and CloudCompare [11], both of which are dedicated to mesh comparison.

Another example of color encoding are heat plots and dense pixel displays. These techniques are typical for displaying multivariate data, allowing to display large amounts of data at the same time. In combination with interactive options such as thresholding, filtering and data reorganization, they are very effective for discovering data relationships.

Concerning the enhancement of perception of 3D shapes, a common practice is applying textures. Similarly to color encoding, stroke textures are frequently used to highlight the surface features such as curvature [10, 19] – see Figure 2.4. Instead of strokes, various glyphs marking spots of significance can be mapped on the objects as well, often in combination with color mapping or opacity modulations.



Figure 2.4: Example of texture indicating principal directions of curvature and color map indicating the curvature values [10].

2.5 Interactive visual analysis

The visualization techniques I have described are often embedded in large systems as features supporting the numerical or statistical analysis and results. The growth of acquired data in the past years, brought on by constantly evolving technology, increases the demand for effective and efficient data processing tools. In these terms, visual analysis – an interactive approach combining computational power, numerical analysis and visualization – has proven to be suitable technique, especially for multivariate, high-dimensional data. The problem with such data, often obtained e.g. by medical cohort studies, is that the data is often defined on domains that only partially overlap. Still, many toolboxes and applications for visual analysis of scientific data have been developed, each tailored to the specific needs of given area.

Example of such toolbox directly related to my area of focus is aforementioned YMCA – Your Mesh Comparison Application [26].

The original goal of the application was comparison and analysis of different point-cloud reconstruction algorithms – detection of problematic parts and comparison of performance. However the provided tools are applicable in many areas where mesh comparison is necessary. As is typical for visual analysis tools, it provides several different views of data. In this case, the authors try to overcome the issues of comparing multiple 3D meshes by combining 3D mesh representation – with color encoding and opacity modulations for displaying mesh variance and detected variance hot-spots – and 2D view using parallel coordinates for analysis of performance of algorithms on different datasets in terms of global error value – see Figure 2.5. The application also contains tools for detail analysis – so called magic lens tool, which allows user to analyze the error values in selected areas – and data summarization tools, which serve for quick data categorization.

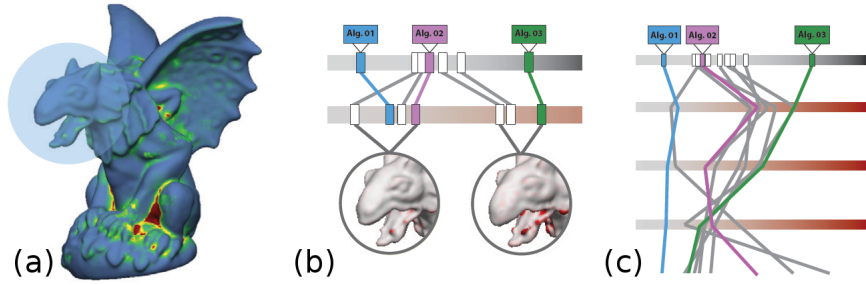


Figure 2.5: Example of tools provided in YMCA [26]. (a) Color map. (b) Detailed view. (c) Parallel coordinates for comparison of algorithms.

Another example of interactive visual analysis tool presented in [1] focuses on the problem of analysis of heterogeneous cohort-study data. The authors present a Data-Cube-Based model for representation and aggregation of data defined across different, only partially matching domains. This model serves as basis for interactive selection, matching and filtering of the data in their prototype application, which again uses various techniques of data representation (scatter plots, histograms, atlas view – modified 3D view, etc.) to provide detailed view as well as relevant summarization.

3 Selected visualization techniques

After conducting a research of related work that I have introduced in previous chapter, I presented the researched techniques to the anthropologists and together we selected several methods that could serve as a basis for my work. These methods were selected to provide a solution to the three main drawbacks of the currently employed visualizations in FIDENTIS Analyst application – the lack of shape information, the lack of local information and the limited view of data.

3.1 Surface superimposition

The first set of the techniques which were selected addresses the issue of preserving the shape information when comparing two models. The two main demands for this visualization were that it should *preserve the shape of both models* and *indicate the differences between models*. The easiest solution to shape preservation would be to use juxtaposition – to display the models next to each other. However, it is difficult to judge the correlation between models just from seeing them rendered side by side. On the other, hand placing them on top of each other causes occlusions and increases the visual complexity. Despite this, I have decided to use superimposition principle with several visual enhancements based on the work of Busking et al. [6] – see Figure 3.1 for example of proposed visualizations.

3.1.1 Transparency

The problem of occlusions can be solved by using transparency. Unfortunately, simply decreasing the opacity of superimposed models tends to make the final image incomprehensible, especially when the models are intersecting and their surfaces are very close to each other. One of the popular techniques for improving the understandability of transparent surfaces is modulation of transparency values based on the placement of the surface in respect to other surfaces. A variation of this technique suitable for cases like Pair comparison – when we are dealing with two nested models – is splitting the surfaces into inner parts – parts of one model enclosed within

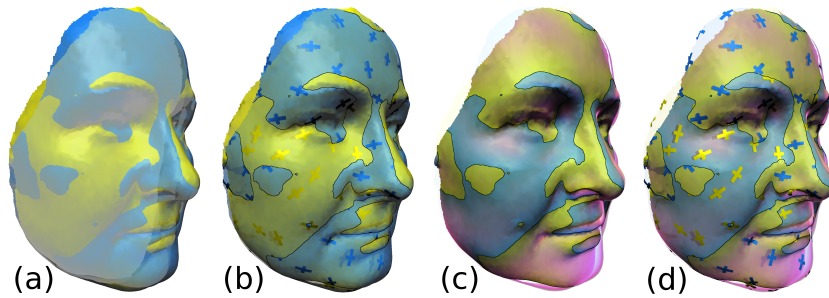


Figure 3.1: Overview of visualization techniques. (a) Models rendered with 50% opacity. (b) Opaque inner surface, transparent outer surface with shadow casting glyphs and intersection contours. (c) Simulation of fog between surfaces. (d) Combination of (b) and (c).

the other model – and outer parts and making the inner parts completely opaque.

Since I am working with facial models which are usually not closed, it is impossible to define something as lying inside or outside of such model. The terms *inner* and *outer* may thus be a bit confusing. To clarify the terminology, I will call the surface that is closest to the camera the *outer surface* and the surface further from camera the *inner surface*. This classification of surfaces takes place in image space, which allows easy handling of special cases such as the one highlighted in Figure 3.2 – the solution for such cases will be discussed in chapter 4.

The technique of turning the inner surfaces opaque reduces the complexity of final image, as this way the observer, for the most part, sees only two layers of surfaces nearest to the camera – only one of which is transparent. Therefore the user should have no problem with determining the adherence of surfaces to the models. Well-chosen contrasting colors of the models and interactive adjustment of opacity values should also help with image interpretation.

3.1.2 Fog simulation

The modulation of transparency helps the user with classification of order of the surfaces. Unfortunately, it is not very helpful for judging the distance between these surfaces. As a visual clue for this task

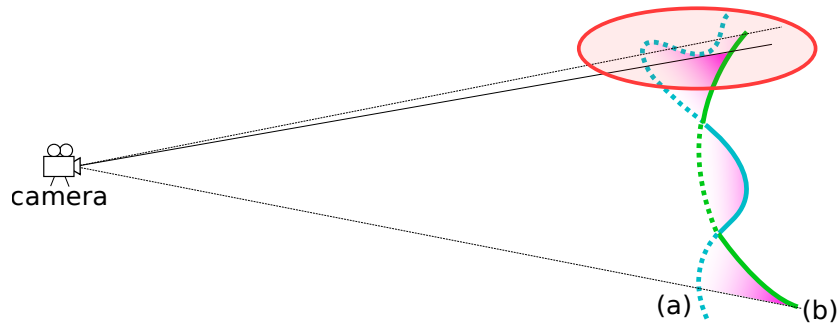


Figure 3.2: Surface transparency and fog simulation scheme. The dotted parts of models (a) and (b) are considered *outer* and are rendered transparently, while the *inner* parts (solid line) are rendered opaque. The intensity of fog (pink color) depends on the distance between surfaces along the viewing direction. The area highlighted in red shows special case when the second surface along viewing ray belongs to the same model as the first one, and thus is classified as the *outer* as well.

I chose two techniques. First of them is fog simulation. The aim of this technique is to simulate a partially transparent volume – fog – of color different to the colors of the models and filling the space between the two surfaces. In reality, the presence of such a volume would clue the observer in on the distance thanks to the accumulation of opacity. Supposing the outer surface is nearly completely transparent, in places where surfaces are close to each other the inner surface would be still visible quite well. However, with growing distance between surfaces the opacity of the fog would accumulate, so in places with large enough surface distances, the inner surface would be completely occluded by the fog – see Figure 3.2. It should be noted here, that the term distance in this case denotes the distance along viewing ray – the result is therefore view dependent.

The downside of the real case scenario is that the outer layer needs to be nearly completely transparent, otherwise the mixture of three colors (two models and fog) would be confusing, or, if the outer surface was close to opaque, the fog would be hardly visible, which would defeat the entire purpose of this technique. To deal with this issue, I have devised three different fog simulation methods based

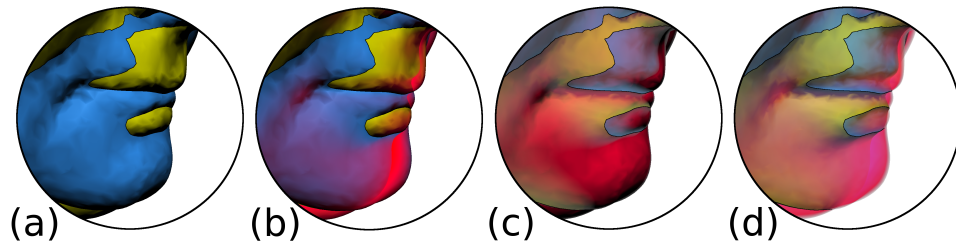


Figure 3.3: Fog simulation techniques. (a) Models rendered with full opacity. (b) Color overlay – notice the illusion that the blue surface lies behind the red one. (c) Transparency mapping on outer surface. (d) Color mapping on inner surface.

on the real case scenario (and thus also view dependent). Figure 3.3 shows the comparison of these methods.

- **Color overlay.** The first method modifies the color of outer surface based on the distance between surfaces – the distance serves as ratio between original color of the surface and the color of the fog. This yields similar result as if the final image was overlaid with new layer in color of fog with opacity based on the surface distances. It solves the issue of the transparency necessity for outer surface, on the other hand it creates misleading illusions about surface adherence to models.
- **Transparency mapping on outer surface.** With this method the entire outer surface is colored to the color of the fog. The distance is then mapped on the opacity values of outer surface – the bigger the distance, the higher the opacity. This method yields nice visual results, but the downside of the method is that the interpretability of surface adherence to models is reduced by coloring the entire outer layer to one color.
- **Color mapping on inner surface.** The last method modifies the color of the surface much the same as the first method – mixing the color of model and the fog based on distance – only this time the color is mapped on the inner layers. It simulates the real case scenario, which may be the most natural for users, but it also has the same problem as the real case scenario – the outer layer needs to be transparent.

3.1.3 Shadow-casting curvature glyphs

It has been found in several studies [12, 21] that shadows aid the human perception of depth and shape. Therefore, as the second method dedicated to improving the interpretability of distances between surfaces I have chosen shadow-casting glyphs, a method based on works [19, 28]. These glyphs are mapped on the outer parts of the surfaces and cast shadows on the inner surfaces. The light source position is fixed in respect to models, so when the users rotate the scene, they can explore the shadows from various angles. The placement of the glyphs on individual models is chosen in such a way that they would not intersect, but would be evenly distributed across the surface.

The color and shape of glyphs were chosen to provide additional information. The color of the glyphs matches the color of the surface so they are only visible when the outer surface is not fully opaque – this is done intentionally to help with classification of surface adherence to models. As for the shape, directional strokes are often employed, with length, orientation and density used to convey properties of surface such as curvature. However, these are more suitable for single surface shape analysis. The primary aim of my work is to illustrate the distances between surfaces, therefore I have opted for the shape presented in [28], a *plus* sign of constant size elongated in one direction. The elongated part of the glyph indicates the direction of maximal principal curvature at the center of the glyph. The perpendicular direction indicated by the shorter part of the glyph is then direction of minimal principal curvature.

3.1.4 Intersection contours

The last enhancement technique I have chosen is contouring of intersections. The motivation for this is that, with transparency and glyphs added to the surface, sometimes the intersection are not very prominent in the image, or are not visible at all – see Figure 3.4 for example. On the other hand bump edges and occlusions may be misinterpreted as intersections in places where there are none. Explicitly marking the contours of surface intersections is therefore highly beneficial, especially for comparative studies of models.

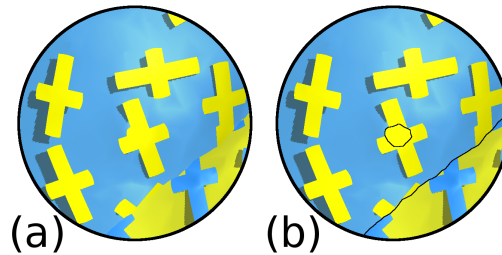


Figure 3.4: Example of case when the small intersection is hidden due to glyph placement. (a) Case without intersection contours. (b) Case with intersection contours.

3.2 Cross sections

I have described a set of methods for visualization of two intersecting facial surfaces. However comparing two models covers but a small part of the tasks anthropologists perform with the FIDENTIS Analyst application. One to many and Batch comparison modes are designed for processing large datasets where displaying all models at once is impossible. Nevertheless, the information about the shape variation, especially local shape variation displayed on local scale, is desirable. But how to deal with the complexity of the data?

Projection, or rather reduction of the 3D data into 2D space is a popular approach, as 2D images can be in some cases easier to interpret than 3D. The cross section method I proposed for this intent is inspired by the technique typically used for visualization of volumetric data. A slicing plane is taken and intersection of 3D data with this plane is computed and then displayed in 2D. In my case I assumed that all the models in dataset are aligned and that one facial model will be selected as primary – this may be the primary face in One to many comparison mode or average face in Batch processing. This model serves as a base for visualization and computation. It is displayed in 3D space along with slicing plane – see Figure 3.5 (a). The users can move and rotate the slicing plane in the space of primary model to get the desired intersection. The intersections of the plane with every model in dataset are then computed, the intersection with primary model (primary intersection) is sampled and the variability at each sample point is determined.

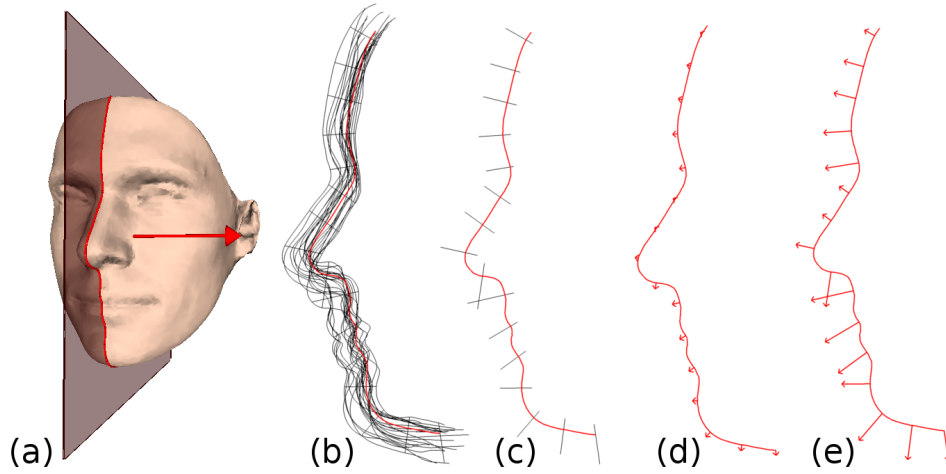


Figure 3.5: Cross sections. (a) Reference picture of average face with the plane specifying cross-sectional slice. (b) Red – intersection of plane with the average face. Black – intersections with all faces in dataset. (c) Distance span in the direction of normal to the primary intersection curve at sampling points. (d) Vectors indicating average distance in the normal direction from average face to all faces in dataset. (e) Same as (d) with enhanced vector sizes.

There are three main options of what user can display, each option representing the variability of shapes in a different way.

- **Intersections with all faces.** Figure 3.5 (b). When the intersections with all faces from dataset are displayed, it is possible to observe how well the models are aligned, especially with interactive manipulation of slicing plane, when user can move across the model and see the local changes. Unfortunately, with increasing number of models the interpretability of final image with all the intersections displayed decreases.
- **Distance span.** Figure 3.5 (c). Distance span indicates the interval of distances from given sampling point to the intersection curves of each model from dataset. The distance is computed in the direction of normal to the primary intersection. It

is displayed as line segment with endpoints indicating maximal distances in positive and negative direction of normal at given point.

- **Average distance.** Figure 3.5 (d,e). Average distance from given sampling point to the intersection curves of each model from dataset, again in the normal direction. This indicates the variability in dataset at given slice. The shorter the vectors are the lower is the variability. Naturally, the user can scale the vectors to better analyze the differences.

3.3 Plots

The last visualization method I have decided to use is dedicated to displaying numerical results computed during the comparison of larger datasets – One to many or Batch comparison. A typical ways of displaying large sets of numerical data are tables and various plots and charts. Somewhere in between them lies a technique called heat plot or heat map. The principle of this method lies in displaying the data in a table-like manner, but instead of numerical values, the data are encoded as colors of the table cells. By filtering and reordering of the data correlations may be discovered more easily as with studying the numerical values themselves, as it is usually more intuitive for users to compare colors than numbers.

I have created two versions of heat maps, one for numerical results of Batch processing and one for auxiliary results of either Batch or One to many comparison.

3.3.1 Numerical results

The numerical results in Batch processing consist of table of $n \times n$ values representing the measurement between each pair of models in the dataset consisting of n faces. These measurements can represent maximal or minimal distance between the two faces, variance, geometric mean, etc.

To display the values I have created a heat plot of $n \times n$ cells, each cell representing one measurement – Figure 3.6. The exact numerical value can be displayed by hovering over the cell. The users can filter

3. SELECTED VISUALIZATION TECHNIQUES

the lowest or/and highest values on interactive scale. There is also an option of reordering the values in the plot according to rows and columns. These options help with easy identification of extrema in the data and observation of relationships of data in dataset.

As an additional feature a histogram of values may be displayed. The color scale corresponding to one used in heat plot is used for drawing the histogram to help with matching the intervals of the scale with data in the plot. The histogram illustrates the distribution and variability of the values in the set.

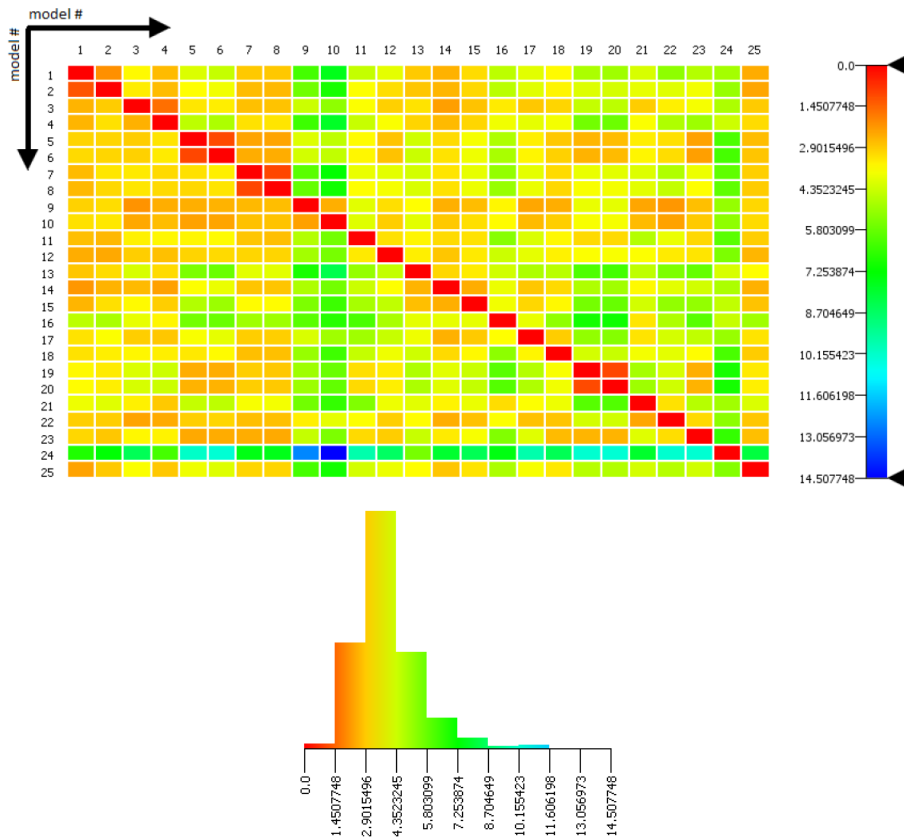


Figure 3.6: Heat plot for visualization of numerical results and the accompanying histogram.

3.3.2 Auxiliary results

Auxiliary results are detailed results computed for each model in the processed dataset. For each vertex of given model they contain the distances to the closest points on each model in dataset. So the auxiliary results for model M from the dataset consisting of n models would contain *number of vertices of $M \times n$* values.

Since the models typically contain thousands of vertices it is rather impractical to display separate cell for each value. Therefore I adjusted the heat plot to display each value as a vertical line segment of 1px width – Figure 3.7. One row of the heat plot then depicts distances from vertices of model of which the auxiliary results are displayed (primary model) to the nearest vertices of one model in dataset. A vertical slice at position x then represents the distances from x th vertex of the primary model to the nearest vertex of each model from dataset. In simpler terms, one row corresponds to one model, and one column correspond to one vertex of primary model.

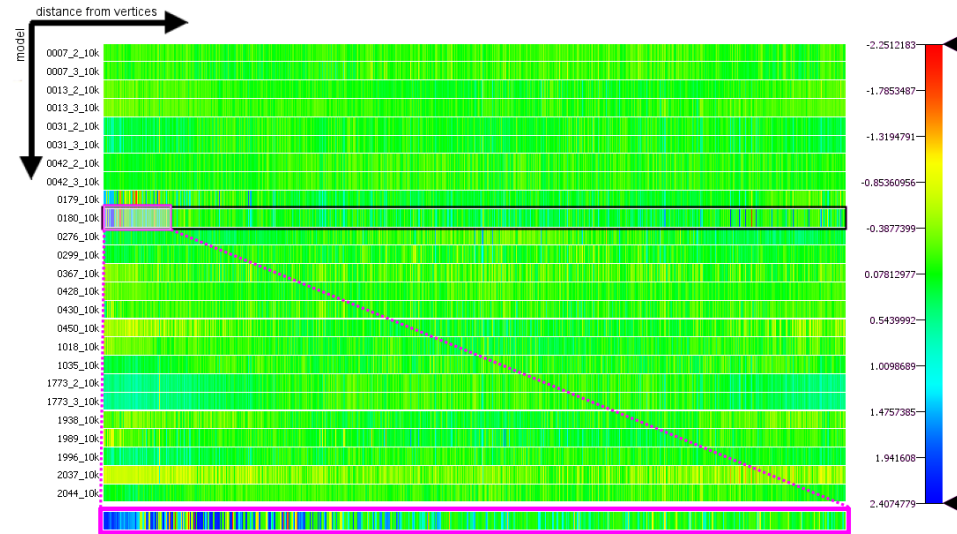


Figure 3.7: Heat plot for visualization of auxiliary results of average face computed from dataset of 25 models. Each row depicts distances of vertices of average face from one model. The pink highlight shows zooming window and the detailed view of individual values.

3. SELECTED VISUALIZATION TECHNIQUES

When scaling the cell width to 1px is not enough to fit the plot into dedicated space, the neighboring entries in vertex dimension are averaged and displayed as one value. Since this is the typical case, the zooming window that can be moved along the selected row has been added. The individual values of the area covered by the window are then displayed under the heat plot. Again, the values can be filtered by interactive scale and the histogram of values can be displayed here as well, either for entire set of results or just for the selected row.

The intent of this visualization is to provide means for analysis of variation in the dataset and for comparison of models in dataset in closer detail. As entries in each row at given horizontal position correspond to same vertex or vertices of primary face, it is possible to observe how models in the dataset differ from primary model at given point.

4 Implementation

In previous chapter I have presented several methods I have chosen for visualization of facial data to meet the needs of scientists. In this chapter I am going to describe the process of implementation and techniques used for it.

4.1 Surface superimposition

For improvement of shape understandability, when comparing two models, I introduced method based on superimposition principle and transparency. Unfortunately, achieving proper, so called Order Independent Transparency for intersecting geometry is problematic, because all fragments corresponding to one pixel need to be sorted and blended in correct order. As I stated in section 2.1 of chapter 2, there are several common methods for solving this well know problem.

The Depth Peeling and approximative techniques are limited to the extensions and modification. However, for the designed visualization techniques I needed access to information such as neighboring fragments or depth distance between fragments, which is not attainable by these techniques. Therefore as a basis for the implementation I chose Per Pixel Linked Lists constructed on GPU [30]. I implemented an algorithm for Order Independent Transparency presented in [27] and modified it, creating a rendering pipeline presented in Figure 4.1.

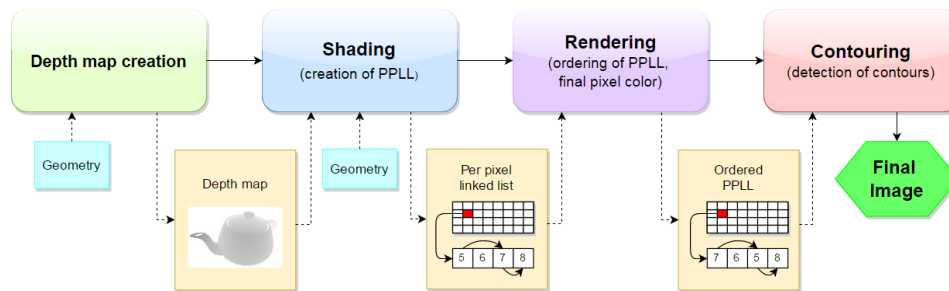


Figure 4.1: Rendering pipeline.

The pipeline consists of four basic steps, two of which are optional:

- **Depth map creation (optional).** Creation of depth map necessary for shadow-casting glyphs, glyph placement.
- **Shading.** Shading of fragments, creation of linked lists, glyph placement.
- **Rendering.** Ordering of the linked lists, color and opacity modulations for visualization, computation of final color per pixel.
- **Contouring (optional).** Detection and rendering of the intersection contours.

The following sections describe the whole process of rendering in greater detail. All of the described methods were implemented in Java using JOGL – the OpenGL wrapper library for Java – and GLSL – the OpenGL Shading Language.

4.1.1 Data preprocessing

Before the models can be passed to rendering pipeline they need to be preprocessed. For the visualization of shadow-casting curvature glyphs the glyph placement and orientation need to be computed.

There are two typical approaches to placement of glyphs on surfaces – either points of some significance can be selected (e.g. points with highest curvature value) or the glyphs can be evenly distributed over surface. Since the visualization technique should primarily express the differences between models on entire model surfaces, even distribution of glyphs is more suitable solution. The even distribution is approximated by algorithm which constructs a valued graph over the model mesh with each vertex storing its neighbors and distances to them. The algorithm iteratively selects a vertex as glyph center and marks the vertices in neighborhood that don't satisfy given distance condition along the graph edges (are closer to the glyph than glyph radius) as used, so they cannot be selected in following iterations.

At the selected vertices the principal curvature directions need to be determined in order to rotate the glyphs to indicate these directions. For this I used the normal averaging method described in [19].

Firstly, for each glyph center vertex an orthogonal coordinate system $(\vec{u}, \vec{v}, \vec{n})$ must be selected. Here, vector $\vec{n}$ is the normal of the given vertex. The other two vectors can be arbitrary as long as the orthogonality is preserved. All the vectors must be of unit length. From this system the second fundamental tensor $\mathbf{II}$ can be defined:

$$A = \begin{pmatrix} \frac{\partial \vec{n}}{\partial u} \cdot \vec{u} & \frac{\partial \vec{n}}{\partial v} \cdot \vec{u} \\ \frac{\partial \vec{n}}{\partial u} \cdot \vec{v} & \frac{\partial \vec{n}}{\partial v} \cdot \vec{v} \end{pmatrix}. \quad (4.1)$$

The components $\frac{\partial \vec{n}}{\partial \vec{a}} \cdot \vec{b}$ indicate the rate at which the surface normal tips in the direction of $\vec{a}$ when we move in the direction of $\vec{b}$. This tensor is defined and averaged for the close neighborhood of the vertex at which the curvature is to be computed. The directions of principal curvature can be then found by rotation of the orthogonal system so that the terms $\frac{\partial \vec{n}}{\partial v} \cdot \vec{u}$ and $\frac{\partial \vec{n}}{\partial u} \cdot \vec{v}$ in the tensor A "disappear" – see Figure 4.2. This is done by diagonalizing A to obtain:

$$D = \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} w_{uu} & w_{vu} \\ w_{uv} & w_{vv} \end{pmatrix}, \quad (4.2)$$

where $|\kappa_1| > |\kappa_2|$ and $A = PDP^{-1}$. The columns of P are the eigenvectors of A , while κ_1, κ_2 are the eigenvalues.

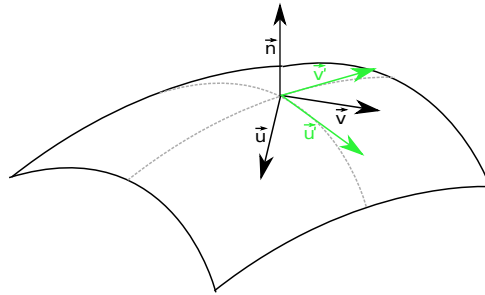


Figure 4.2: Curvature. Orthogonal system before rotation (black) and after rotation (green) indicating principal curvature directions.

The principal curvature directions are given as:

$$\begin{aligned}\vec{u}' &= w_{uu}\vec{u} + w_{uv}\vec{v}, \\ \vec{v}' &= w_{vu}\vec{u} + w_{vv}\vec{v}.\end{aligned}\tag{4.3}$$

The vector $\vec{u}'$ indicates the direction of maximal principal curvature, while vector $\vec{v}'$ indicates the direction of minimal principal curvature. The values of the curvature are defined by κ_1 and κ_2 respectively.

4.1.2 Depth map creation

As soon as the computation of glyph centers and orientation is completed, the models can be passed to rendering pipeline. If the option of shadow-casting glyphs is enabled, the shadow placement must be computed. For this I chose basic shadow mapping approach presented in [29], based on testing the visibility of fragment from the light position.

The geometry is rendered from the position of light in order to get depth of fragments nearest to the light. These depth values are stored in depth map – a texture which is passed along the rendering pipeline to the point where the geometry is rendered from normal camera position. Then the depth value in the light space is computed for each fragment and it is compared with corresponding value in depth map. If the value in depth map is lower, the fragment lies in shadow. See Figure 4.3 for example.

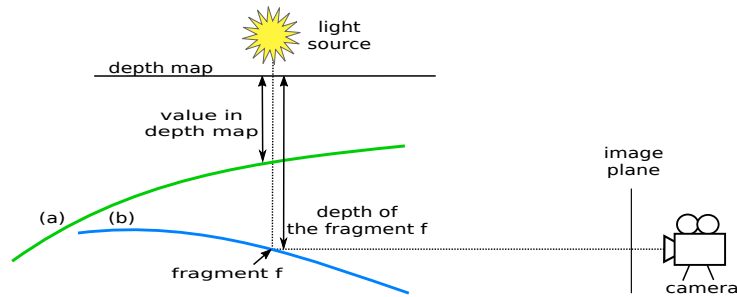


Figure 4.3: Shadow mapping. At the location of fragment f , the surface (b) is shadowed by the surface (a).

In order to create the shadows of the glyphs, it must be tested at the time of rendering from light position whether the rendered fragment is part of the glyph or not.

My first idea for determining this was to generate texture coordinates for the vertices around glyph center, based on the vertex normal, curvature directions and distance from the vertex and map the glyph texture on the model. However, the algorithm based on this idea proved to be inefficient – the texture coordinates computation taking up to several minutes. Therefore, I used the idea presented in [19] – constructing a 3D representation of glyph around glyph center and testing whether fragment lies inside it – see Figure 4.4 (a).

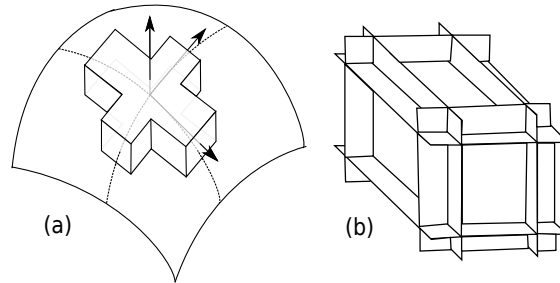


Figure 4.4: (a) 3D representation of glyph placed on the surface. (b) Cuboid defining planes.

The 3D representation of glyph consists of two crossing cuboids. Each cuboid can be defined by 6 planes (see Figure 4.4 (b)) and each of these planes is defined by its distance from glyph center and either by normal of the glyph center, or principal curvature direction vector. Therefore for each rendered triangle the closest glyph centers for its vertices along with corresponding normals and curvature vectors are passed to the fragment shader. These vectors and values are computed in object space. In order to operate in the same space, untransformed positions of vertices in object space need to be passed from vertex to fragment shader to obtain the fragment position in object space.

Once the cuboid planes and the fragment position are defined in the same space, the test whether fragment lies inside the 3D glyph is a simple matter of testing the distances from fragment position to the planes.

If the rendered fragment lies inside the glyph, it is processed and its depth value may be written to the depth map, otherwise the fragment is discarded.

4.1.3 Shading

Once the depth map is created, the geometry is rendered from the camera position. In this phase shading of fragments and creation of Per Pixel Linked List takes place.

The fragments are shaded using Phong shading model [25]. It is a shading model based on the interpolation of vertex attributes in fragment shader, thus achieving visually smooth results. It is therefore commonly used model for per pixel lighting. Figure 4.5 shows the lighting scheme.

For each fragment the color computation consists of three parts – ambient, diffuse and specular – which are summed to get the final color. The ambient component (equation 4.4) serves as rough approximation of indirect light reflections. It therefore doesn't depend on the positions, only on colors of the material and light. The diffuse component (equation 4.5) is computed according to Lambert's law, which says the amount of reflected light is proportional to the cosine of the angle (a dot product) between normal vector of the surface and light direction vector. The specular component (equation 4.6) specifies the direct reflections of light. The value depends on the cosine of the angle between reflection direction and eye (camera) direction from the surface.

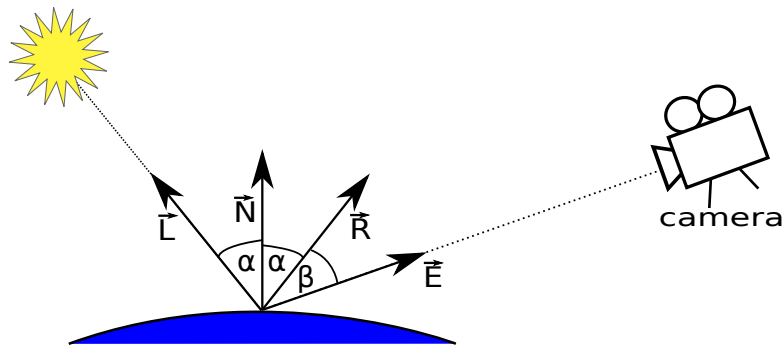


Figure 4.5: Lighting scheme.

$$Color_{amb} = Light_{amb} \cdot Material_{amb} \quad (4.4)$$

$$Color_{diff} = Light_{diff} \cdot Material_{diff} \cdot \cos(\alpha) \quad (4.5)$$

$$Color_{spec} = Light_{spec} \cdot Material_{spec} \cdot \cos(\beta)^s \quad (4.6)$$

$s = shininesses$

When the fragment is shaded, it is added to a fragment list. To be able to use the list for Order Independent Transparency, several things need to be set up.

Firstly, there needs to be a way to access the first fragment list entry for each pixel. For this purpose a texture with the size of maximal expected image size is created. This texture is called Head Pointer texture and at each position it stores index of the last processed fragment corresponding to the given pixel position.

The shading of fragments and list building is computed on GPU, therefore several fragments will be processed in parallel. To precede concurrent modifications of the list, an Atomic Counter needs to be created. This counter keeps track of number of entries in the list, as it is increased every time new entry is to be added.

Finally, the list itself is implemented as *RGB32UI* image buffer. This means that each fragment entry needs to be of *uvec4* format – a four component vector of *uint* – unsigned integer.

A first component of each fragment entry will be the pointer to the index of next fragment at given pixel. As index of *uint* format, this is a straightforward operation.

The second component should store the color of the fragment. However, the color is of *vec4* format – four component float vector. Fortunately, the function *packUnorm4x8* defined in GLSL allows to pack four floating point values into 8-bit integers resulting in 32-bit unsigned integer. In this format the color can be added as the second component of the fragment entry.

Third component consists of fragment depth. The depth is the float value, so it needs to be transformed to *uint* using *floatBitsToUint* function.

The fourth component is unused in original Order Independent Transparency implementation. However, for the implementation of the visualization techniques such as fogging, I needed additional in-

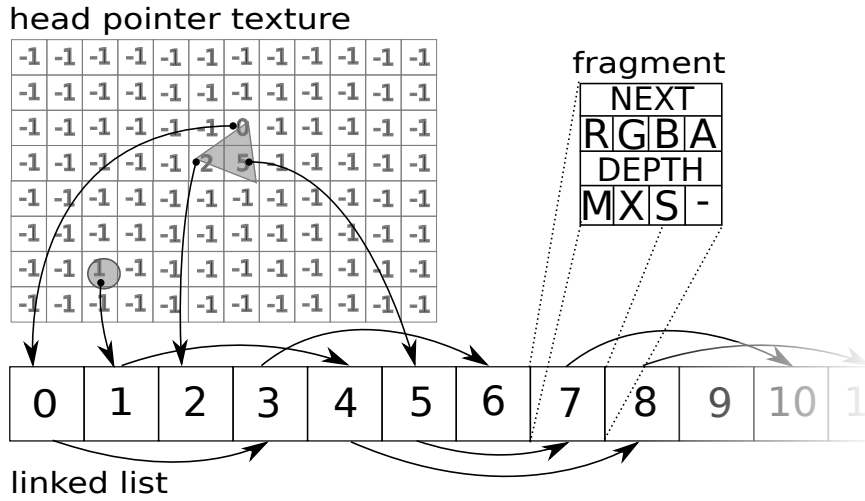


Figure 4.6: Scheme of the linked list and structure of the fragment. Next – the pointer to the next fragment at given pixel, RGBA – color of the fragment, Depth – depth of the fragment, M – index of model to which the fragment belongs, X – fragment is/isn't part of the glyph, S – fragment is shadowed, – unused

formation about fragment. I therefore used this space to store index of model to which the fragment belongs (passed to fragment shader as uniform), information about whether the fragment is part of the glyph (computed as described in previous section) and information about whether fragment lies in shadow. These three components with one additional unused zero component are represented as float values and packed using the *packUnorm4x8* function again.

The complete scheme of linked list can be found in Figure 4.6.

Once everything is set up, the process of storing a fragment in the list looks as follows:

1. Atomic Counter storing the number of fragments in the list is incremented. The new value is stored in *index* variable. This is an atomic operation.
2. The value in Head Pointer texture corresponding to current pixel position is exchanged, the new value being *index*. The old value is stored as *old_index*. This is an atomic operation as well.

3. The fragment is shaded and prepared for storing in the list as *uvec4* – the first component will be *old_index*, second packed color, third depth of the fragment and fourth the additional information for rendering.
4. The fragment information is added to the list at the position of *index*.

4.1.4 Rendering

When the list of fragments is complete, it is passed to the next step in rendering pipeline, where the final color for each pixel is computed.

Prior to the color computation, the values in the Depth buffer after rendering models are analyzed in order to get their span – this aids the later computations. Then the Color and Depth buffer are cleared and a rectangle covering the viewport is rendered, so that the new color can be computed for each pixel of viewport.

In fragment shader corresponding to this part of rendering pipeline, the list of fragments adherent to the pixel processed at given time needs to be extracted from the list of all fragments – a reversal process to the list building. The index in Head Pointer texture at given pixel is looked up and an entry from the fragment list at this index is copied to the temporary list created in the shader. The first component of the copied entry points to the next fragment entry in the list corresponding to same pixel, so the next entry is retrieved and copied to the temporary list as well. The process is repeated as long as the first component of the newly retrieved entry points to valid index.

Once the per pixel list is retrieved, it is sorted according to depth value of each fragment (the third component of each fragment entry), so the nearest fragment is the first in the list. After this, the computation of color, based on the selected visualization options (passed to shader as uniform variables), may take place.

If the option of making the inner surface opaque (see section 3.1.1 of chapter 3) is turned on, the alpha values of fragments need to be adjusted based on the inner-outer classification. The alpha value of first fragment is not modified – it is user adjustable – and the fragment is automatically considered outer. For the second fragment the

test must be performed to decide whether it belongs to the same model as the first one, or is part of another model. If it is part of another model, the second fragment is turned opaque (alpha value set to 1), labeled inner and the fragments behind it may be discarded, since they will not be visible. If the second fragment belongs to the same model as the first one, it is usually the case of protuberant surface such as nose. In this case it is desirable to keep the alpha values unchanged, consider the fragment outer and proceed to process third fragment as if it was the first one. The process is repeated until one fragment is declared opaque, or the list of fragments ends. It should be noted that it will always be the even (second, fourth, etc.) fragment that will be declared opaque – see Figure 4.7.

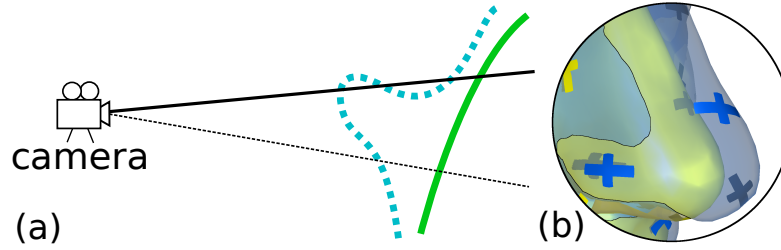


Figure 4.7: Example of case when the viewing ray intersect the surface belonging to same model several times. (a) Schematic view. (b) Rendered result.

When the surfaces are classified as inner or outer, the solution for shadow-casting glyphs option is trivial. With each fragment the information about whether it is part of the glyph and whether it is shadowed is stored. So for fragments that were declared outer in previous step it is tested whether they are part of the glyph and if true, they are turned opaque. For the fragments that were declared inner it is tested whether they are shadowed, and if true, their brightness and saturation is decreased.

As for the fog, it is always function of depth difference between two fragments – outer and inner, or, in special cases such as in Figure 4.7, first and second, third and fourth fragment, etc. The z_{ratio} which is used as depth parameter in computations is defined as following:

$$z_{ratio} = k \cdot \frac{|z_a - z_b|}{|z_{max} - z_{min}|}, \quad (4.7)$$

where z_{max} and z_{min} are the maximum and minimum depth values detected when rendering geometry, z_a and z_b are the depth values of two fragments between which the fog is to be simulated and k is a parameter for fog density.

There are three different fog implementations (see section 3.1.2 of chapter 3 for description):

- **Color overlay.** The first method highlights the distances by modifying the color of outer fragment – the fragment nearer to the camera. It interpolates between fog color and surface color, using z_{ratio} as interpolation parameter:

$$Color_{new} = Color_{fragment} \cdot (1 - z_{ratio}) + Color_{fog} \cdot z_{ratio} \quad (4.8)$$

- **Transparency mapping on outer surface.** With this method each outer fragment is colored to the color of the fog by changing the hue component of the HSV fragment color representation to hue of the fog. The saturation and value are thus preserved. The distance is then mapped on the alpha values (opacity) of outer fragment:

$$Alpha_{new} = 0,2 \cdot Alpha_{fragment} + z_{ratio} \quad (4.9)$$

- **Color mapping on inner surface.** The last method modifies the color of inner fragments in the same way as the first method modifies the color of outer fragments – see equation 4.8.

After the aforementioned modifications to fragments are done, the final per pixel color may be computed. The variable $Color_{final}$ is first initialized to the color of the background. Then the sorted per pixel list of fragments is traversed, starting with the fragment furthest from camera and with each fragment the final color is modified in following way:

$$Color_{final} = Color_{final} \cdot (1 - Alpha_{fragment}) + Color_{fragment} \cdot Alpha_{fragment} \quad (4.10)$$

At the end of the computation the pointers in original list of fragments are modified to represent the sorted version of per pixel lists.

4.1.5 Contouring

After the Rendering phase, the final image is nearly complete. The only remaining part is highlighting the model intersections. If this option is selected, one additional layer – again a rectangle covering viewport – is rendered over the image result from previous steps. In this layer only the pixels detected at the intersection points will be modified, the rest of them will be discarded and thus not interfering with the already rendered image.

To compute the intersection, the sorted list of fragments acquired in previous step is passed to the fragment shader. There, for the pixel corresponding to processed fragment the neighborhood of 3×3 pixels is searched. For each pixel within the neighborhood the first two fragments closest to the camera are looked up in the sorted list of fragments. Two conditions must be satisfied in order for the fragment to be declared as intersection point:

1. At least one of the first fragments in the neighborhood must belong to different model then the first fragment at the current pixel position.
2. The depth difference between first two fragments at given pixels must be lower than experimentally chosen threshold, to prevent false detections of intersections.

If the fragment is identified as intersection point, it will be rendered, otherwise it will be discarded.

4.2 Cross sections

The technique I have designed for visualization of local shape variability shows the variability in a set of models at the place of their intersection with 3D plane. It was again implemented using JOGL library.

The base of this technique lies in computing the intersection of facial model with 3D plane. Since the meshes used in FIDENTIS Analyst application are triangulated, this problem reduces to computing an intersection of triangle with plane and this in turn reduces

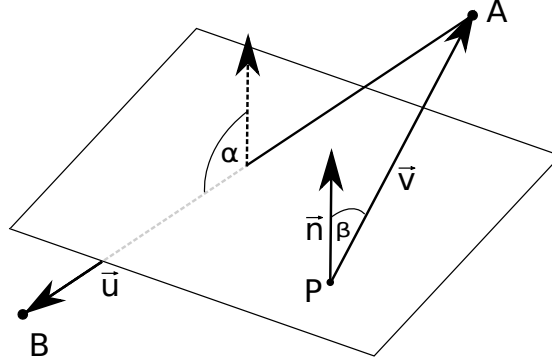


Figure 4.8: Line-plane intersection scheme.

to intersection of line segment (triangle edge) with the plane. For a plane given by point P and normal $\vec{n}$ and the line segment given by two points A and B – see Figure 4.8, it can be solved using following equations:

$$\vec{u} = B - A \quad \vec{v} = A - P \quad (4.11)$$

$$D = \cos(\alpha) = \vec{n} \cdot \vec{u} \quad N = -\cos(\beta) = -\vec{n} \cdot \vec{v} \quad (4.12)$$

If $\cos(\alpha)$ is equal to zero, the segment is parallel to the plane. In that case if $\cos(\beta)$ is equal to zero as well, the segment lies in the plane, otherwise there is no intersection. If segment is not parallel to the plane, the intersection point of line (given by the segment) with the plane must be computed and tested whether it lies within segment. The line corresponding to the segment AB may be represented as:

$$l = A + t \cdot \vec{u}, \quad (4.13)$$

and the t parameter for line-plane intersection point is given as:

$$t = \frac{N}{D}. \quad (4.14)$$

If $t \in < 0, 1 >$, then the segment intersects with the plane.

The described test is done for each edge of each triangle. The detected plane-triangle intersections – points and line segments are stored in a list. When each triangle is processed, the intersecting line segments are connected at the points with same or very close coordinates to create polyline paths – see Figure 4.9 (a). Depending on the

model and plane orientation, there may be one or more intersection paths per model.

These paths are then rotated to XY plane to be displayed in 2D view. As all the paths lie in same plane, the rotation corresponds to rotation of the plane normal to z-axis. The angle between the normal and z-axis can be computed from their dot product, while the rotation axis is given by their cross product:

$$\cos(\theta) = \vec{n} \cdot \vec{z} \quad \vec{u} = \vec{n} \times \vec{z} \quad (4.15)$$

The rotation of each point to the XY plane is then done by following matrix for rotation around arbitrary axis u :

$$R = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z \sin\theta & u_x u_z(1-\cos\theta) + u_y \sin\theta \\ u_y u_x(1-\cos\theta) + u_z \sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x \sin\theta \\ u_z u_x(1-\cos\theta) - u_y \sin\theta & u_z u_y(1-\cos\theta) + u_x \sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \quad (4.16)$$

Once in 2D, the intersection paths belonging to the model marked as primary (further referred to as primary paths) are sampled at equal distances and at each sample point a normal to the path is computed – a trivial task, since the path consist of line segments. For each sampling point a ray given by the point and the normal to the primary path at that point is defined – Figure 4.9 (b). Then the intersections between these rays and paths of models from dataset are computed – if there is more than one intersection for the model, the closest one is taken – Figure 4.9 (c).

Finally, between each sampling point and intersections adherent to it (intersections of the line passing through the sampling point) a vector is computed. The vectors adherent to one sampling point are summed, the final vector providing information about variability at given sampling point – Figure 4.9 (d).

4.3 Plots

The last visualization technique I chose, consists of 2D plots for numerical data. The implementation of this technique was done using Java 2D API.

For displaying the numerical results of Batch processing, the table-like plot is drawn. Numerical value is mapped on each cell of the ta-

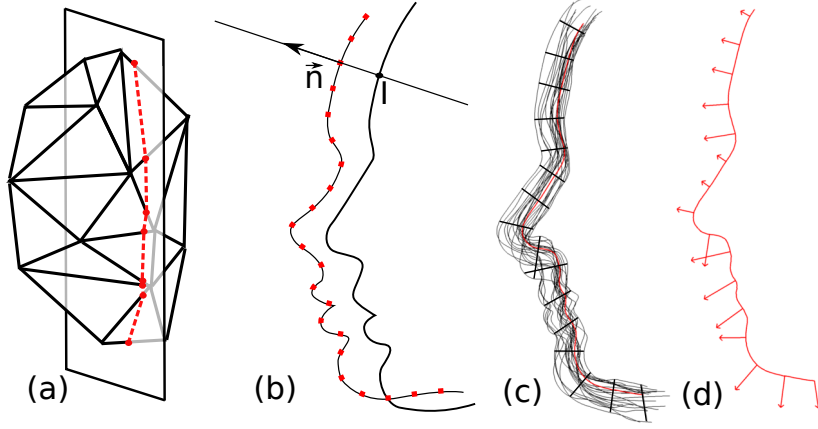


Figure 4.9: (a) Example of detected intersection path on triangular mesh. (b) Example of sampled intersection path. The point I on secondary model illustrates its intersection with the ray given by sampling point and normal to the primary path at that point. (c) Intersections with entire dataset. (d) Vectors indicating variability (scaled for illustration).

ble in a form of color. For computing of colors I operate in HSV color space, mapping the values onto hue component.

To compute the color which should represent given value, firstly the minimal and maximal values in the numerical results must be found. These values are then mapped onto boundary color values, which are by default red (hue = 0) for minimum and blue (hue = 240) for maximum. The hue color component is then computed according to following equation:

$$h = h_{min} + \frac{(v - v_{min})}{(v_{max} - v_{min})} \cdot (h_{max} - h_{min}), \quad (4.17)$$

where h denotes hue and v denotes numerical values.

The same color picking principle is also applied when displaying auxiliary results. The only additional calculations that needs to be done concern clustering the neighboring values to fit the whole dataset into plot area – as there may be thousands of entries per model – and reversal process for zooming and displaying the original values.

Both plots are drawn using Java 2D functions for drawing lines and rectangles and all interactivity must be achieved by tracking the mouse motion and computing its position in relation to interactive objects.

5 Results and evaluation

I chose and implemented three different approaches of visualization for 3D facial data. For scenarios where two models are processed and compared, I chose superposition principle with several adjustments and cues to improve the shape and distance perceptibility – intersection contours highlight, surface opacity modulation, shadow-casting glyphs indicating surface curvature, and simulation of fog based on the distance between surfaces.

Second method I designed is based on cross-sectional slices. Its purpose is to visualize local variability and shape. This technique transfers the local data from 3D to 2D view, which reduces visual complexity and allows the observer to focus on important details.

The third and final tool I created employs heat maps for visualization of numerical data exported from the FIDENTIS Analyst application. The purpose of this tool is to reduce the need for further data post-processing in other applications and provide yet another outlook on data.

Examples of this techniques may be found in attachment A as a part of evaluation questionnaire.

I created a standalone application in which all the described visualization techniques are implemented, compatible with data used in FIDENTIS Analyst application and ready to be integrated there as well.

To evaluate my visualization techniques I conducted a user study among anthropologists working on FIDENTIS project. The techniques were evaluated by four scientists working in a field of facial morphometry and analysis. The users were first introduced to the visualization principles and the test application, in which they could try the visualizations. They were then asked to fill out a questionnaire in which they evaluated the selected techniques. The questionnaire can be found in full extent in attachment A.

5.1 Surface superimposition

The questionnaire consisted of four parts, the first one concerning evaluation of visualizations for pair comparison based on surface su-

perimposition method. In this part eight selected combinations of techniques were presented to scientists (see Figure 5.1), including the visualization technique currently employed in FIDENTIS Analyst application.

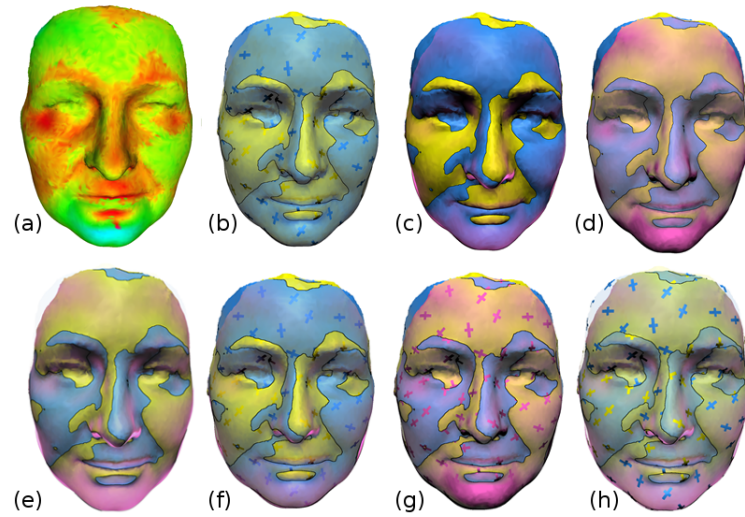


Figure 5.1: Visualizations presented for evaluation.

For each of these techniques the users were asked to rank the following questions:

1. How well does this visualization convey shapes of individual models?
2. How well does this visualization convey differences between models?

The scale from *very well* through *well*, *neutral*, *poorly* to *very poorly* was provided for evaluation. The averaged results of their evaluation can be seen in Figure 5.2. The detailed answers are included in attachment B.

It can be seen from these results, that while the visualization currently employed in the application – visualization (a) – ranks high on the second question (imparting differences), it ranks low on conveying shapes in comparison with other methods. In fact, out of the

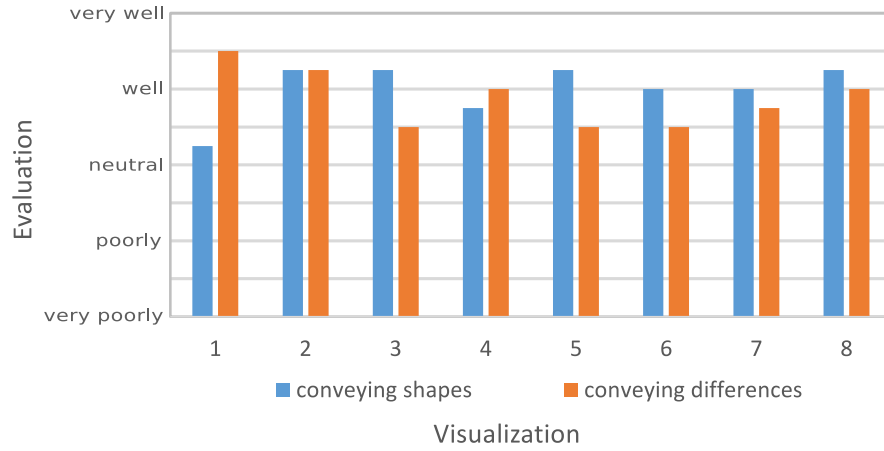


Figure 5.2: Averaged evaluation of surface superimposition visualization techniques.

presented methods, this one ranked highest for the second question, but lowest for the first one.

Concerning the newly proposed methods, evaluated as the best suited for conveying the shapes of models were: visualization with shadow-casting glyphs (b), color overlay (c), color mapping on inner surface (e) and combination of shadow-casting glyphs and color mapping on inner surface (h). It is notable that for conveying the shapes all new methods were ranked higher than the one currently used.

For conveying the differences between models, the best ranked of the new methods was visualization with shadow-casting glyphs (b), closely followed by transparency mapping on outer surface (d) and combination of shadow-casting glyphs and color mapping on inner surface (h).

The visualizations (b) – shadow-casting glyphs and (h) – combination of shadow-casting glyphs and color mapping on inner surface rank high in both questions, balancing both – need for illustration of the shape and imparting the differences between models.

During the evaluation some remarks were made concerning the visualizations. The scientists appreciated that when using glyphs,

the shape of outer surfaces is indicated even when the surface is almost transparent. However, it was suggested that the light position should not be fixed in respect to orientation of the models in order to achieve moving shadows when the models are rotated. This idea was prompted by occurrence of shadow distortion on surfaces that were almost parallel to light direction.

Regarding the fog simulation, users found it less straightforward and understandable than the glyphs. The main concern was the view dependent intensity of fog. Due to the fact that the intensity is based on distance between surfaces in viewing direction, with model rotation the intensity may change at certain locations. It was suggested that the technique should be made view independent.

It was further noted that the intersection contours are very beneficial for intersection detection and suggested that option of highlighting them should be added also to the currently employed color map visualization.

The scientists further appreciated the interactivity of the application, the option of combining and adjusting the techniques. For example, using these interactive adjustments they designed a visualization of only the volume between two surfaces by setting both surfaces transparent and adding Color overlay fog simulation – Figure 5.3. I did not considered such an option prior to the evaluation, but they found the visualization contributory to their work.



Figure 5.3: Visualization of volume between surfaces.

5.2 Cross sections

The second part of questionnaire was dedicated to evaluation of local variability visualization based on cross sections. The scientists were asked to evaluate this technique by three questions:

1. How well does this visualization convey variability in a set of models at specific (sampling) points? (again on a scale from *very well* to *very poorly*)
2. Does this visualization convey local variability better or worse than color map on surface? (*better, I don't know, worse*)
3. Is this visualization contributory to variability perception and analysis? (*yes, I don't know, no*)

In the first question all respondents answered *very well* or *well*, proclaiming the visualization fairly demonstrative for local variability.

In the second question, the scientists deemed this visualization better than the one used in FIDENTIS Analyst application in terms of conveying local variability, appreciating possibility to focus on one place and scale the visual results.

All respondents marked the technique as contributory to variability perception and analysis in third question.

The scientists liked the option of showing cross sections of entire dataset versus vectors indicating variability. There was only one remark – it was suggested to add the option of displaying absolute variability values (as opposed to currently used relative, which take into account orientation of vectors).

5.3 Plots

In third part of the questionnaire I asked for the evaluation of heat plots I created for numerical and auxiliary results. For both parts I prepared two questions:

1. How well does this visualization convey variability in a set of models? (on a scale from *very well* to *very poorly*)

2. Is this visualization contributory to analysis of facial dataset?
(yes, I don't know, no)

For visualization of numerical results, the respondents claimed the visualization convey variability in a set of models mostly *very well*. All respondents marked this visualization as contributory to the analysis, especially appreciating the color-coded histogram.

As for visualization of auxiliary results, the scientists found it less demonstrative of variability, marking it *well* or *neutral* in first question and only two respondents found it contributory to the analysis of dataset. It was remarked that when performing facial analysis the scientists are most interested in facial location with greatest variability. Therefore it was suggested to improve this visualization by displaying the location of selected vertices on facial model.

5.4 Summary

In the last part of the questionnaire I asked the respondents to choose the best suited visualization method for specific tasks to see if and where my methods were usable. Based on the discussions with scientists prior and during the development of my work, I have listed the tasks they typically perform with FIDENTIS Analyst application:

- Verifying the alignment of models.
- Analyzing the shape of models.
- Comparing two models.
- Analyzing local variability of models.
- Analyzing a set of models (comparison, variability, etc.).

For each of these tasks the scientist were asked to choose up to three techniques they found suitable for it.

For *verifying the alignment of models* the two selected visualizations were surface superimposition method with shadow-casting glyphs and cross-sectional slices with option of showing slices of all models in dataset. It was also suggested that for verification of alignment of

two models the superimposition method with fully opaque models and only intersection contours (no glyphs) would be suitable, since glyphs increase image complexity, which was considered unnecessary for this task.

In order to *analyze the shape of models* the scientist selected as most suitable again the cross-sectional slices and shadow-casting glyphs visualization.

For *comparing two models*, the color map on model was found as most suitable by two respondents. Their other choices consisted of combinations of fog simulations and glyphs, but not showing any significant preference for one particular method.

For the task of *analyzing local variability* the preferred methods were color map on model, cross-sectional slices and heat plot for numerical results.

Finally, in case of *analyzing a set of models* all scientists uniquely settled on heat plot for numerical results and some of them also found cross-sectional slices beneficial for the task.

In conclusion, the most useful visualizations were found to be color map on model (currently employed in application), shadow-casting glyphs without fog, cross-sectional slices and heat plot for numerical results. The fog simulations were often found confusing, with scientists stating that although visually nice-looking, the interpretation "takes some getting used to". As for visualization of auxiliary results, the lack of information about location of displayed values made it less preferable to other techniques.

Conclusion

The aim of my work was to design and implement visualization methods for facial comparison in order to extend the visualization techniques currently provided in FIDENTIS Analyst application – an application for facial analysis used by anthropology scientists.

In order to do so, I first analyzed the visualizations currently provided in the application and discussed with scientists the needs of the applications, the areas they would like to explore in greater detail and their expectations for the new visualizations. With the help of scientists I put together the list of issues I then attempted to solve when designing the visualizations.

Based on this list I conducted a research of related work in area of 3D data visualization, comparative visualization and visual analysis of cohort study data. I looked for principles that could be applicable for facial data and provide solution to the issues at hand.

I then presented the results of my research to anthropologists. Together we selected several techniques that served as bases for my work. I adjusted them to meet the needs of input data and better address the problems. I chose and implemented three different approaches of visualization.

For scenarios where two models are processed and compared, I chose superposition principle – placing the two models on top of each other – and implemented several adjustments and cues to improve the shape and distance perceptibility. Namely I implemented intersection contours detection and highlight, surface opacity modulation based on its position in regards to viewer, shadow-casting glyphs indicating surface curvature and simulation of fog based on the distance between surfaces.

Another method I designed in order to visualize local variability and shape is based on cross-sectional slices. This technique transfers the local data from 3D to 2D view, which reduces visual complexity and allows the observer to focus on important details.

The third and final tool I created employs heat maps for visualization of numerical data exported from the application. The purpose of this tool is to reduce the need for further data post-processing in other applications and provide yet another outlook on data.

After implementing these techniques I conducted a user study with scientists working in the area of facial analysis to evaluate the contribution of my visualizations to the area. The results showed the scientists found most of the presented visualization techniques contributory, three of them – surface superimposition method with shadow-casting glyphs, cross-sectional slices and heat plot for numerical data – particularly so, marking them as visualization of choice for several tasks they usually perform during their work.

The standalone application with the visualization tools I created is already used by the scientists. In the nearest future I am going to integrate the visualization techniques into FIDENTIS Analyst application, where they could aid the process of facial analysis as well as communicate the results. I would also like to implement the suggestions for improvement and extensions made by scientists during evaluation process – mainly the view independent fog simulation and the auxiliary results heat plot extension for displaying the location of the data on the model.

Bibliography

- [1] ANGELELLI, P., OELTZE, S., HAASZ, J., TURKAY, C., HODNELAND, E., LUNDERVOLD, A., LUNDERVOLD, A. J., PREIM, B., AND HAUSER, H. Interactive visual analysis of heterogeneous cohort-study data. *IEEE computer graphics and applications* 34, 5 (2014), 70.
- [2] BAVOIL, L., AND MYERS, K. Order independent transparency with dual depth peeling. *NVIDIA OpenGL SDK* (2008), 1–12.
- [3] BERTIN, J. Matrix theory of graphics. *Information Design Journal* 10, 1 (2000), 5–19.
- [4] BESL, P. J., AND MCKAY, N. D. Method for registration of 3-D shapes. In *Robotics-DL tentative* (1992), International Society for Optics and Photonics, pp. 586–606.
- [5] BORN, S., WIEBEL, A., FRIEDRICH, J., SCHEUERMANN, G., AND BARTZ, D. Illustrative stream surfaces. *IEEE transactions on visualization and computer graphics* 16, 6 (2009), 1329–1338.
- [6] BUSKING, S., BOTH, C. P., FERRARINI, L., MILLES, J., AND POST, F. H. Image-based rendering of intersecting surfaces for dynamic comparative visualization. *The visual computer* 27, 5 (2011), 347–363.
- [7] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B. *Readings in Information Visualization: Using Vision to Think*. Interactive Technologies Series. Morgan Kaufmann Publishers, 1999.
- [8] CARNECKY, R., FUCHS, R., MEHL, S., JANG, Y., AND PEIKERT, R. Smart transparency for illustrative visualization of complex flow surfaces. *Visualization and Computer Graphics, IEEE Transactions on* 19, 5 (2013), 838–851.
- [9] CHALÁS, I., FERKOVÁ, Z., URBANOVÁ, P., KOZLÍKOVÁ, B., KOTULANOVÁ, Z., AND SOCHOR, J. Surface-based visualization on human face variation. In *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*

- (Postfach, Germany, 2014), I. Viola, K. Buhler, and T. Ropinski, Eds., Eurographics Association.
- [10] DIEWALD, U., PREUSSER, T., AND RUMPF, M. Anisotropic diffusion in vector field visualization on euclidean domains and surfaces. *Visualization and Computer Graphics, IEEE Transactions on* 6, 2 (2000), 139–149.
- [11] EDF R&D, TELECOM PARISTECH. Cloudcompare (version 2.6.1). <http://www.cloudcompare.org/>, 2015. Accessed: 2015-04-27.
- [12] ERENS, R. G. F., KAPPERS, A. M. L., AND KOENDERINK, J. J. Perception of local shape from shading. *Perception & Psychophysics* 54, 2 (1993), 145–156.
- [13] EVERITT, C. Interactive order-independent transparency. *White paper, NVIDIA* 2, 6 (2001), 7.
- [14] FRIESE, K. I., BLANKE, P., AND WOLTER, F. E. YaDiV – an open platform for 3D visualization and 3D segmentation of medical data. *The visual computer* 27, 2 (2011), 129–139.
- [15] GALVÁNEK, M., FURMANOVÁ, K., CHALÁS, I., AND SOCHOR, J. Automated facial landmark detection, comparison and visualization. In *31st Proceedings of Spring Conference on Computer Graphics* (Bratislava, Slovakia, 2015), J. Jorge, L. P. Santos, and R. Ďurikovič, Eds., Comenius University, pp. 21–28.
- [16] GLEICHER, M., ALBERS, D., WALKER, R., JUSUFI, I., HANSEN, C. D., AND ROBERTS, J. C. Visual comparison for information visualization. *Information Visualization* 10, 4 (2011), 289–309.
- [17] GOWER, J. C. Generalized procrustes analysis. *Psychometrika* 40, 1 (1975), 33–51.
- [18] HUMMEL, M., GARTH, C., HAMANN, B., HAGEN, H., AND JOY, K. I. Iris: Illustrative rendering for integral surfaces. *Visualization and Computer Graphics, IEEE Transactions on* 16, 6 (2010), 1319–1328.

- [19] INTERRANTE, V., FUCHS, H., AND PIZER, S. M. Conveying the 3D shape of smoothly curving transparent surfaces via texture. *Visualization and Computer Graphics, IEEE Transactions on* 3, 2 (1997), 98–117.
- [20] KOTULANOVÁ, Z., CHALÁS, I., AND URBANOVÁ, P. 3D Virtual Model Database of Human Faces: Applications in anthropology and forensic sciences. In *Mikulov Anthropology Meeting. The Dolní Věstonice Studies 20* (Brno, 2014), S. Sázelová, A. Hupková, and T. Mořkovský, Eds., Academy of Sciences of the Czech Republic, Institute of Archaeology in Brno; Masaryk University, Department of Anthropology at Faculty of Science, pp. 177–180.
- [21] MAMASSIAN, P., KNILL, D. C., AND KERSTEN, D. The perception of cast shadows. *Trends in cognitive sciences* 2, 8 (1998), 288–295.
- [22] MCCORMICK, B. H., DEFANTI, T. A., AND BROWN, M. D. Visualization in Scientific Computing. *Computer Graphics* 21, 6 (1987).
- [23] MCGUIRE, M., AND BAVOIL, L. Weighted Blended Order-Independent Transparency. *Journal of Computer Graphics Techniques (JCGT)* 2, 2 (2013).
- [24] MESHKIN, H. Sort-Independent Alpha Blending. Perpetual Entertainment. GDC Session, Mar 2007.
- [25] PHONG, B. T. Illumination for computer generated pictures. *Communications of the ACM* 18, 6 (1975), 311–317.
- [26] SCHMIDT, J., PREINER, R., AUZINGER, T., WIMMER, M., GRÖLLER, M. E., AND BRUCKNER, S. YMCA - Your Mesh Comparison Application. In *IEEE VIS 2014* (2014), VIS, IEEE Computer Society.
- [27] SHREINER, D., SELLERS, G., KESSENICH, J. M., AND LICEA-KANE, B. M. *OpenGL programming guide: The Official guide to learning OpenGL, version 4.3*. Addison-Wesley, 2013.

BIBLIOGRAPHY

- [28] WEIGLE, C., AND TAYLOR, R. M. Visualizing intersecting surfaces with nested-surface techniques. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 503–510.
- [29] WILLIAMS, L. Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics* (1978), vol. 12, ACM, pp. 270–274.
- [30] YANG, J. C., HENSLEY, J., GRÜN, H., AND THIBIEROZ, N. Real-Time Concurrent Linked List Construction on the GPU. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1297–1304.

Appendices

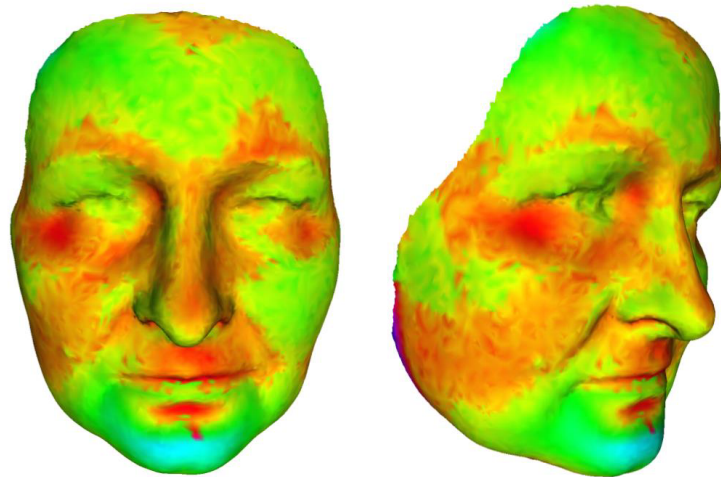
A Questionnaire

PART I - Visualization of intersecting surfaces

This part of the questionnaire is dedicated to visualization of two intersecting facial surfaces with aim to convey the shapes and differences of the shape.

1. Color map - visualization currently used in application

Distances or variability between two or more faces are mapped on one model in a form of color.



Minimal distance (negative)

Maximal distance (positive)

How well does this visualization convey shapes of individual models?

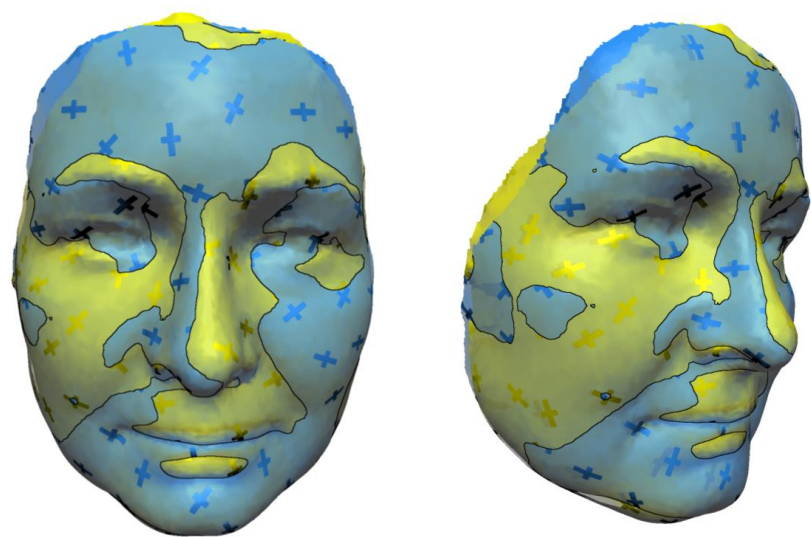
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

2. Transparency + glyphs

Two models are rendered over each other, intersections of surfaces are contoured and outer layer (layer closer to viewer) is rendered transparently with opaque shadow-casting glyphs. The orientation of glyphs indicates principal curvature directions of surface at given points.



How well does this visualization convey shapes of individual models?

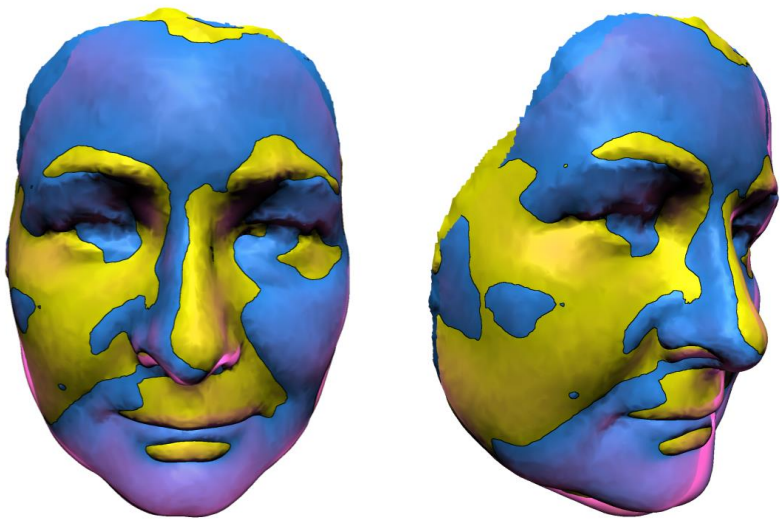
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

3. Overlay fog

Two models are rendered over each other with adjustable transparency. Additional layer in color different to colors of the models is rendered over the image indicating the distances between surfaces - the bigger the distance between surface, the more opaque the layer is.



How well does this visualization convey shapes of individual models?

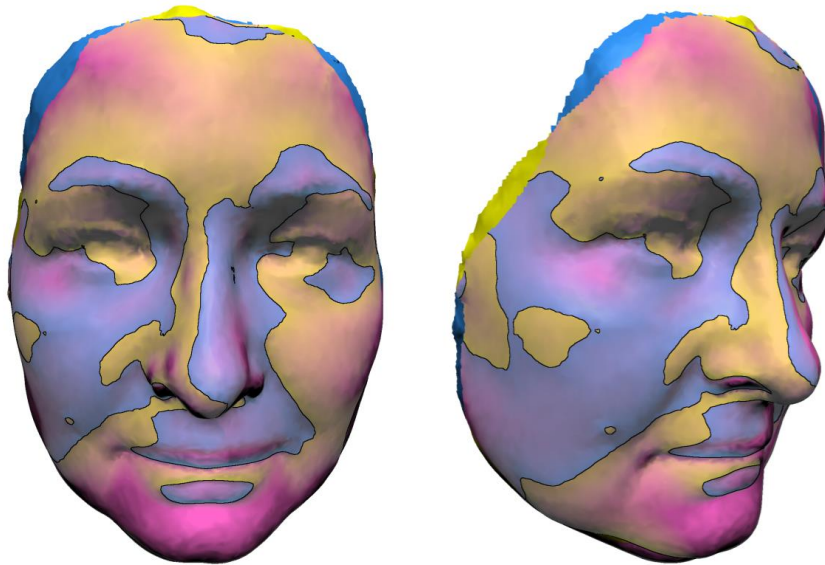
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

4. Outer surface fog

The entire outer layer (layer closer to viewer) of intersecting models is colored to color different to colors of the models. The distances between surfaces are mapped to transparency - the bigger the distance between surfaces, the more opaque the color is.



How well does this visualization convey shapes of individual models?

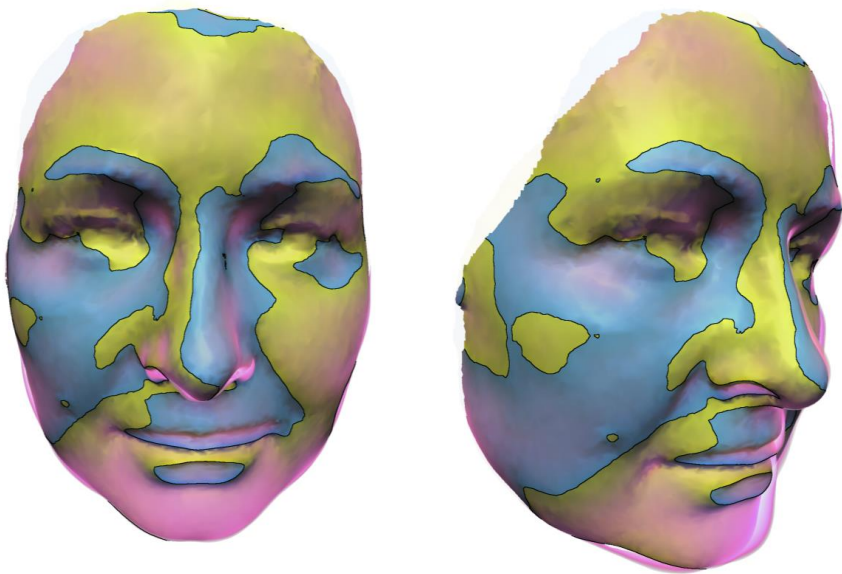
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

5. Inner surface fog

The distance between model surfaces is mapped onto inner layers (layer further from viewer) in a form of color. Outer layers are rendered at least partially transparently in order for the inner surfaces to be visible.



How well does this visualization convey shapes of individual models?

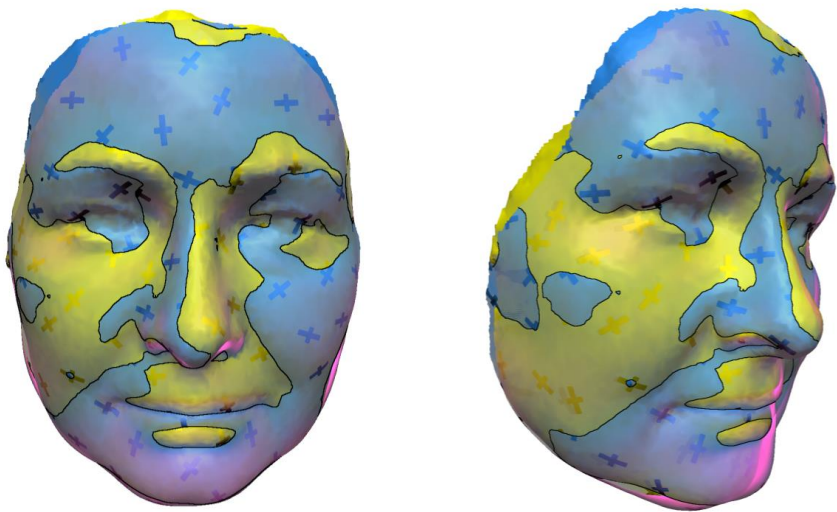
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

6. Overlay fog + glyphs

Two models are rendered over each other with adjustable transparency. Additional layer in color different to colors of the models is rendered over the image indicating the distances between surfaces - the bigger the distance between surface, the more opaque the layer is. In addition the shadow-casting curvature glyphs are mapped on outer surface.



How well does this visualization convey shapes of individual models?

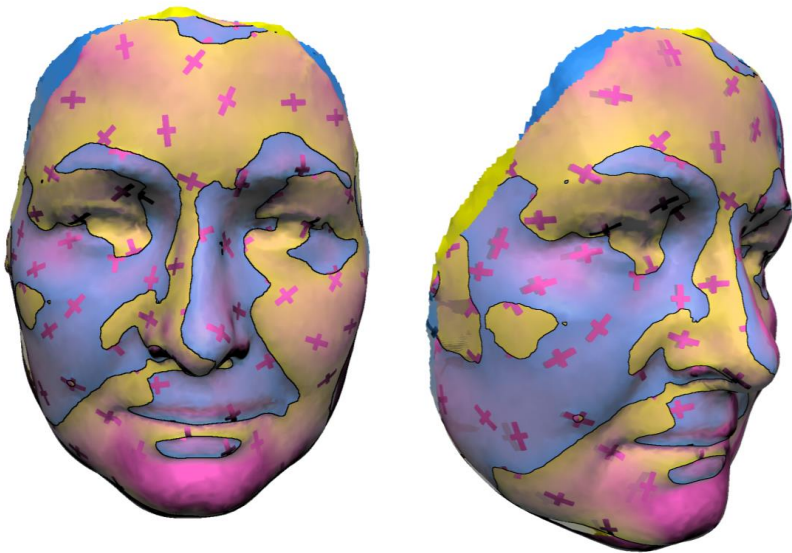
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

7. Outer surface fog + glyphs

The entire outer layer (layer closer to viewer) of intersecting models is colored to color different to colors of the models. The distances between surfaces are mapped to transparency - the bigger the distance between surfaces, the less transparent the color is. In addition the shadow-casting curvature glyphs are mapped on outer surface.



How well does this visualization convey shapes of individual models?

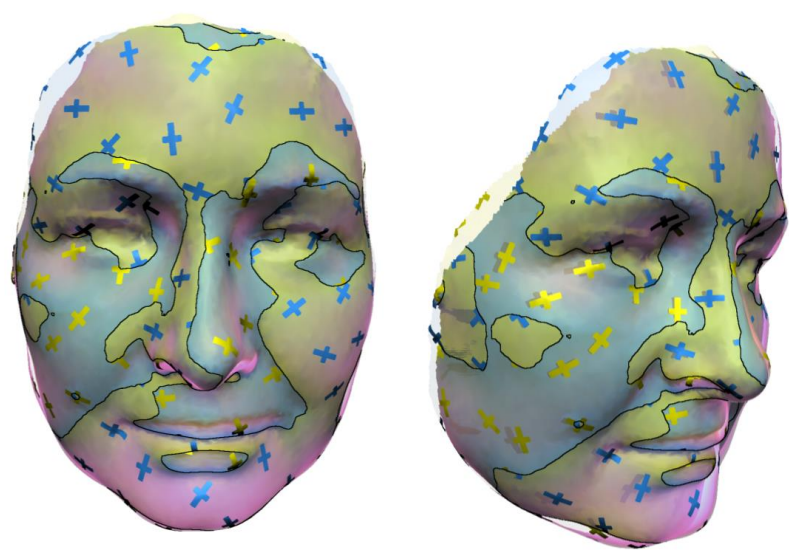
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

8. Inner surface fog + glyphs

The distance between model surfaces is mapped onto inner layers (layer further from viewer). Outer layers are rendered at least partially transparently. In addition the shadow-casting curvature glyphs are mapped on outer surface.



How well does this visualization convey shapes of individual models?

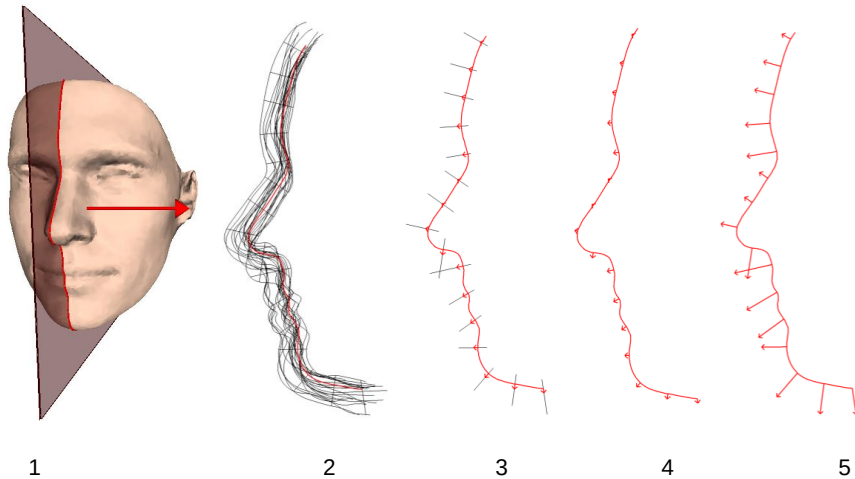
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

How well does this visualization convey differences between models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

PART II - Visualization of surface cross-sections

This part of the questionnaire is dedicated to visualization of two or more intersecting surfaces via cross-sectional slices.



1 - reference picture of average face with the plane specifying cross-sectional slice.

2 - red - intersection of plane with the average face, black - intersections with all faces in dataset

3 - black - variability at sampling points in the direction of normal to the intersection with average face, red - vectors indicating average distance in this direction from average face to all faces in dataset

4 - same as (3) - red

5 - same as 4 with enhanced vector sizes

How well does this visualization convey variability in a set of models at specific (sampling) points?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

Does this visualization convey local variability better or worse than color map on surface (see #1)?

better	I don't know	worse
--------	--------------	-------

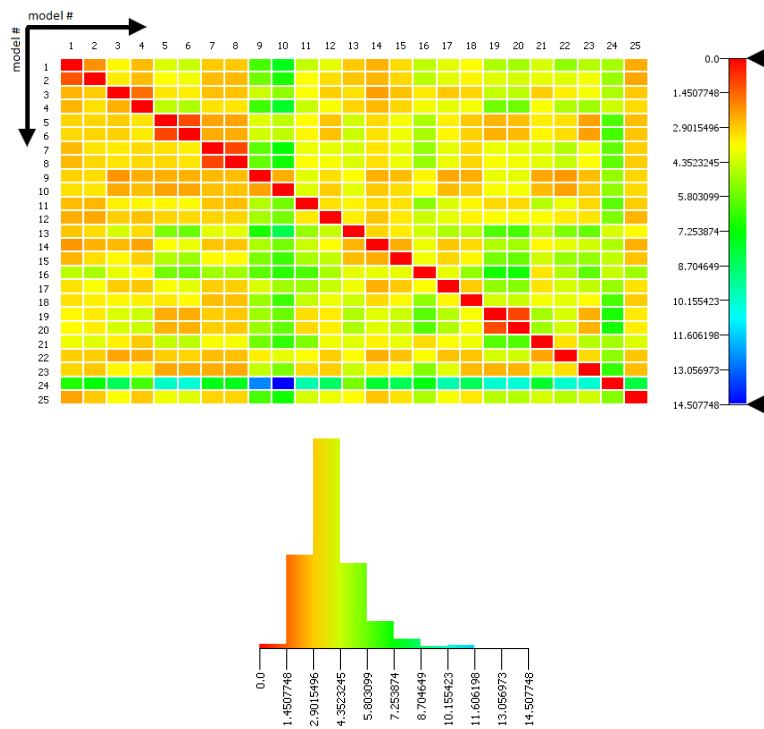
Is this visualization contributory to variability perception and analysis?

yes	I don't know	no
-----	--------------	----

PART III - Visualization of numerical results for whole dataset

1. Numerical results

Pairwise numerical results for set of facial models mapped on color in interactive heat plot. The visualization allows selection, sorting and filtering of values. Complementary histogram shows distribution of values in given set.



How well does this visualization convey variability in a set of models?

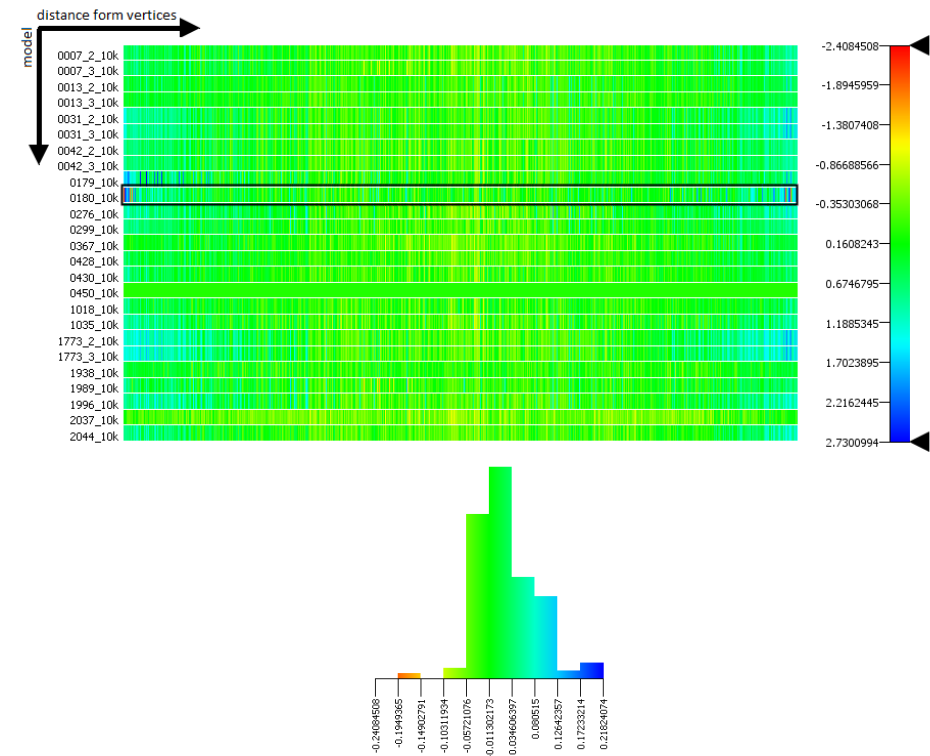
very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

Is this visualization contributory to analysis of facial dataset?

yes	I don't know	no
-----	--------------	----

2. Auxiliary results

Distances from one model (e.g. average model) to each model in set. Each vertical line in the heat plot represents distance from one vertex of selected model to the nearest vertex in another model in a set. Complementary histogram shows distribution of values either in entire set or in respect to one selected face.



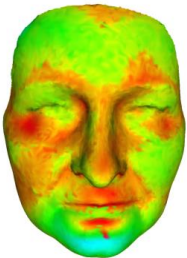
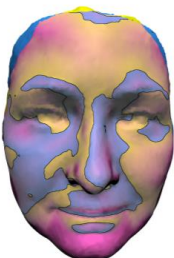
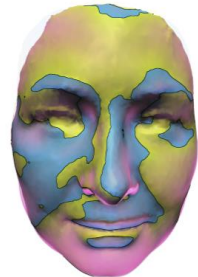
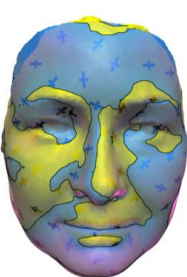
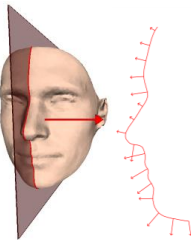
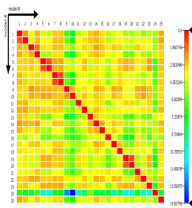
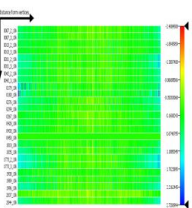
How well does this visualization convey variability in a set of models?

very well	well	neutral	poorly	very poorly
-----------	------	---------	--------	-------------

Is this visualization contributory to analysis of facial dataset?

yes	I don't know	no
-----	--------------	----

Summary

			
1.	2.	3.	4.
			
5.	6.	7.	8.
			none
9.	10.	11.	12.

Of the afore mentioned visualizations, select a visualization which would be in your opinion best suited for:

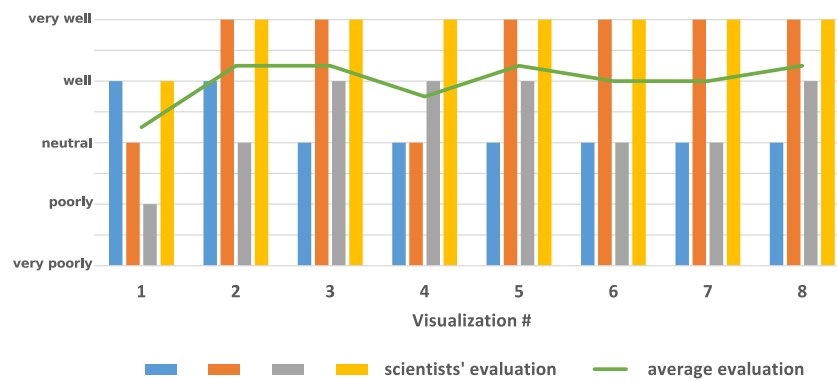
1. verifying the alignment of the models:
2. analyzing shape of the models:
3. comparing two models:
4. analyzing local variability of models:
5. analyzing a set of models (comparison, variability, etc.):

Comments & suggestions:

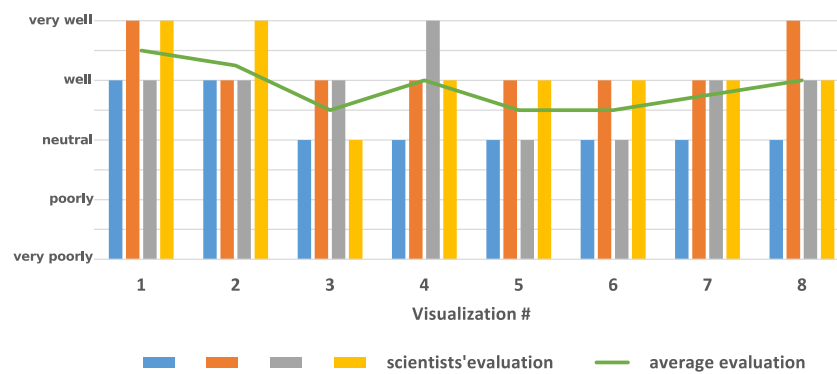
B Questionnaire results

Part I – Visualization of intersecting surfaces

- How well does the visualization convey shapes of individual models?



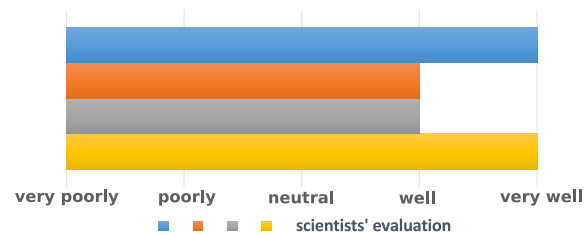
- How well does the visualization convey differences between models?



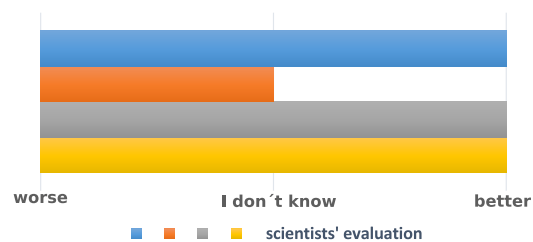
For visualizations corresponding to Visualization # please refer to page 67.

Part II – Visualization of surface cross sections

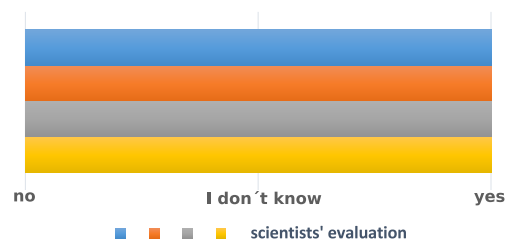
- How well does this visualization convey variability in a set of models at specific (sampling) points?



- Does this visualization convey local variability better or worse than color map on surface?

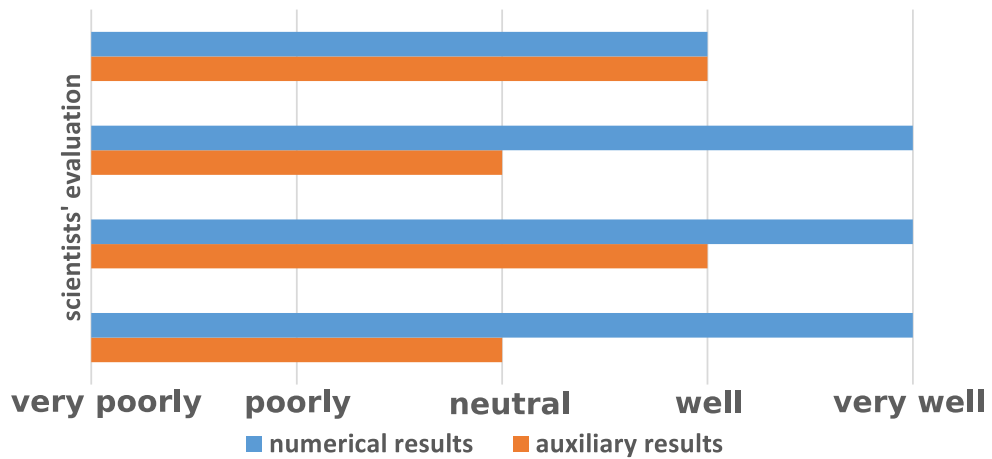


- Is this visualization contributory to variability perception and analysis?

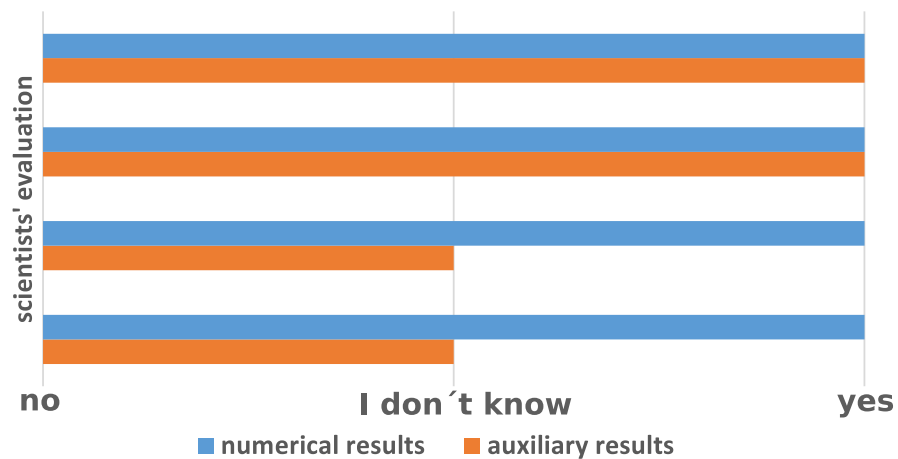


Part III – Visualization of numerical results for whole dataset

- How well does the visualization convey variability in a set of models?

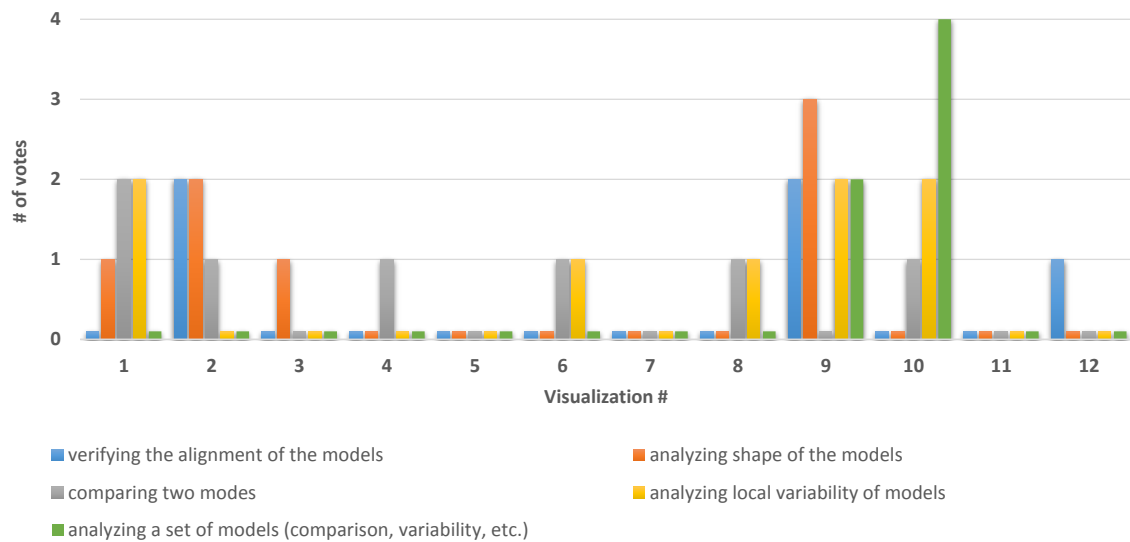


- Is the visualization contributory to analysis of facial dataset?



Summary

- Select a visualization which would be in your opinion best suited for:
 1. Verifying the alignment of the models.
 2. Analyzing shape of the models.
 3. Comparing two models.
 4. Analyzing local variability of models.
 5. Analyzing a set of models (comparison, variability, etc.).



For visualizations corresponding to Visualization # please refer to page 67.

C User guide

To present the results of my work I created an executable standalone application, where the visualization methods described in this thesis are implemented.

Main menu

The main window of the application is split into two parts – Editor window (on the left) and Properties window (on the right). After the start of the application, main menu is displayed in the Editor window. The menu consists of three buttons – *Surfaces*, *Slices* and *Plots* – see Figure 6.1. *Surfaces* button leads to surface superimposition visualization techniques, *Slices* leads to visualizations based on cross-sectional slices and *Plots* leads to heat plot visualizations for numerical and auxiliary results.

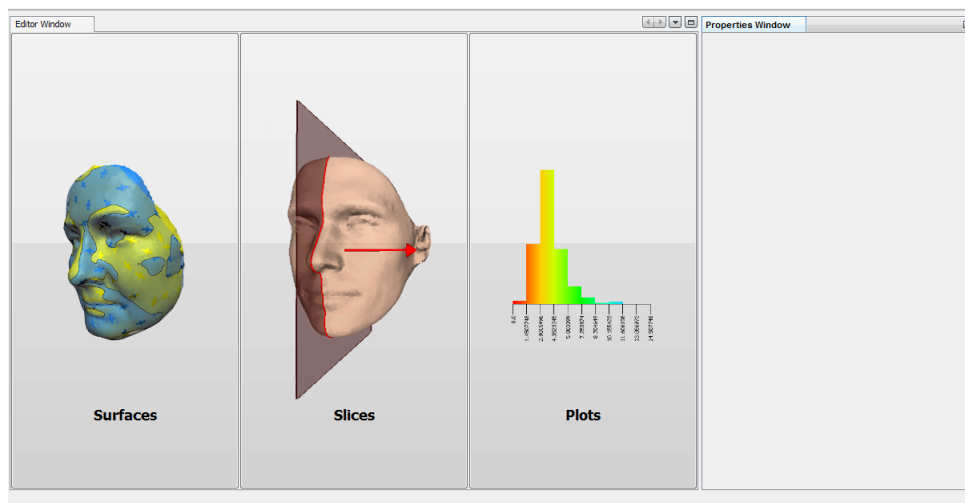


Figure 6.1: Main menu of the application.

Surfaces

In *Surfaces* mode (Figure 6.2), the Editor window serves for displaying two nested 3D models. The models can be rotated and zoomed

either by embedded buttons or by mouse – left mouse button press & drag for rotation, right mouse button press & drag for movement and wheel for zooming.

The two models can be loaded using *Load...* buttons in Properties window. Using other controls in this window visualization properties, such as colors or transparency can be adjusted. Controls for turning on and off the visualization features – glyphs, fog simulation and intersection contours – are also situated here.

The displayed image can be exported into high resolution PNG file by *Export visual results...* button, which opens new window for name and location selection. This functionality can be found in other parts of the application as well.

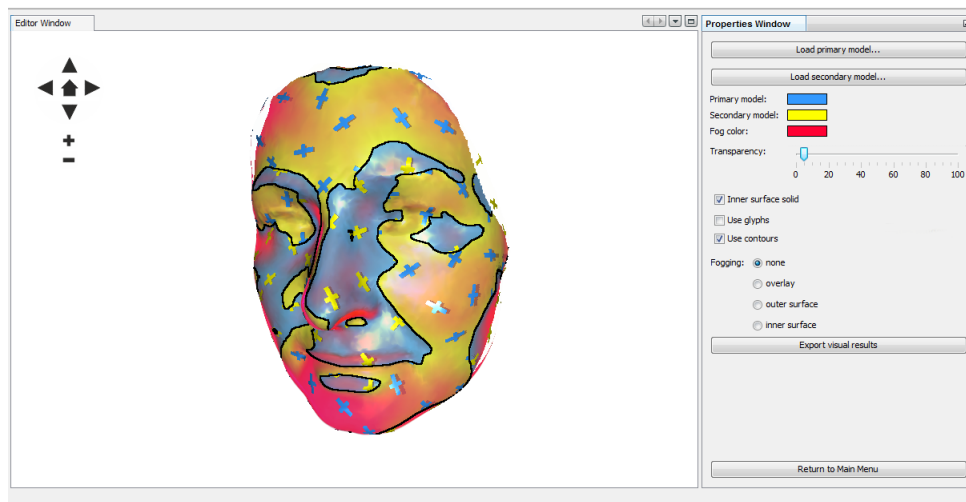


Figure 6.2: Surface superimposition visualization in the application.

Slices

In *Slices* mode (Figure 6.3), the Editor window is split into two parts. In the left part of the window the average model with intersection plane is displayed. The plane can be adjusted by dragging the arrow gizmo – left mouse button for rotation of the plane, right mouse button for adjusting the position.

Right part of the Editor window is dedicated to 2D view of model intersections with the plane. This view can again be zoomed (mouse wheel) and moved (left mouse button).

The buttons for loading average (primary) model and the rest of the models from dataset are situated in Properties window, along with controls for plane position and visualization features. The users can quickly set plane orientation along one of the world axis or manually enter normal of the plane.

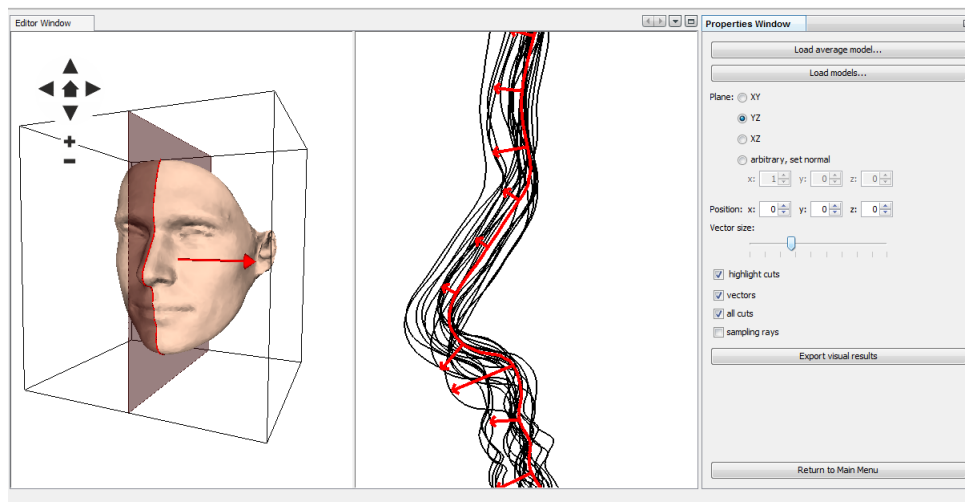


Figure 6.3: Visualization based on cross sections in the application.

Plots

The heat plot visualization differs based on the loaded data. In both cases the heat plot is situated in Editor window and the values can be filtered using interactive scale displayed on the right side of the plot.

Additionally, in case of numerical results, the heat plot values can be sorted by invoking pop-up menu over the plot entry (right mouse button click) and selecting one of the options – see Figure 6.4. The exact values of heat plot cells can be displayed by hovering over the cell. The histogram, which can be displayed in separate window, shows the distribution of values in the dataset.

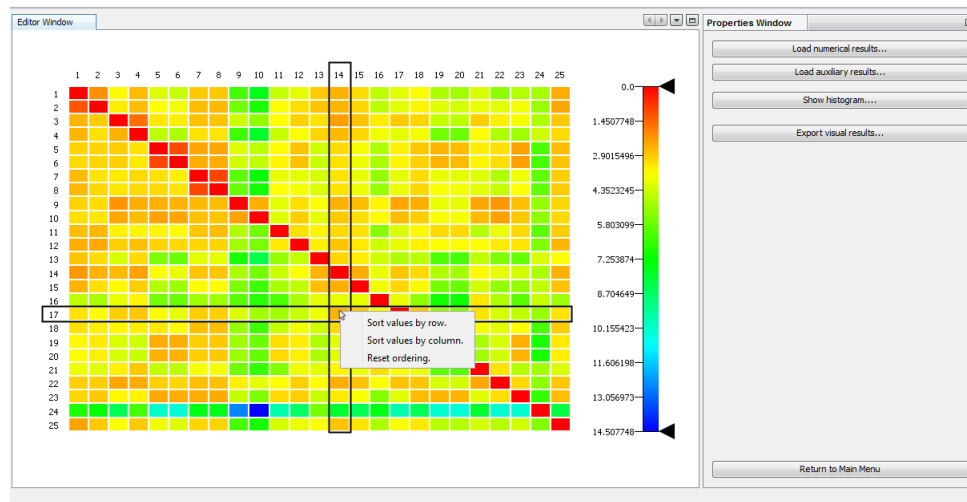


Figure 6.4: Visualization for numerical results in the application.

In case of auxiliary results, the zoomed area can be adjusted by dragging the zooming window (purple rectangle) along the selected row of heat plot – see Figure 6.5. Concerning the histogram, it shows the distribution of values in selected row or in entire dataset if nothing is selected.

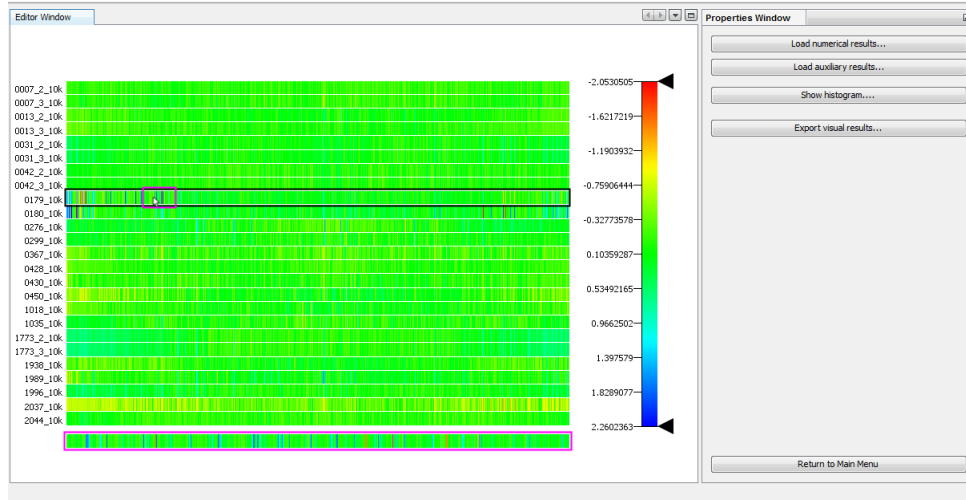


Figure 6.5: Visualization for auxiliary results in the application.

D Thesis Archive in IS MU

This appendix contains a list of files that accompany this text in the thesis archive in the Masaryk University Information System¹.

- Text of the this thesis in PDF.
- LaTeX source files and figures used in this thesis.
- Executable application containing the visualization techniques described in this work.
- Source files of the application.
- Test data for the application.

1. The archive is accessible at http://is.muni.cz/th/374538/fi_m/.