

User Manual & Algorithm Description Document

“skullanalyzer” v1.0

Andreas Bertsatos*, 26.10.2019

Department of Animal and Human Physiology,
National and Kapodistrian University of Athens,
Panepistimioupolis 157 01, Athens, Greece

*email: abertsatos@biol.uoa.gr

Introduction

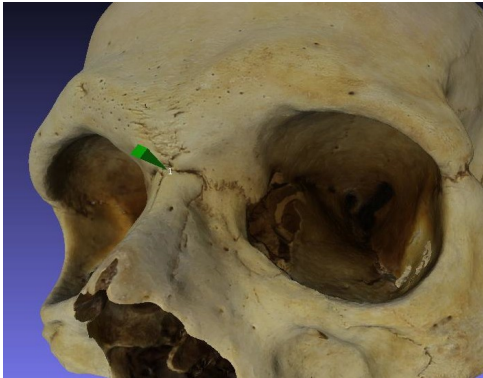
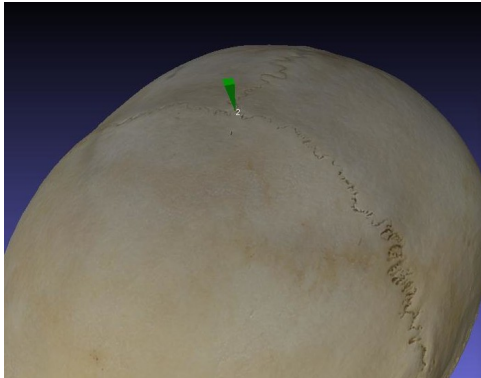
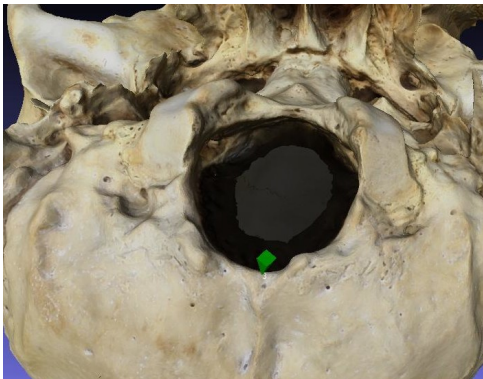
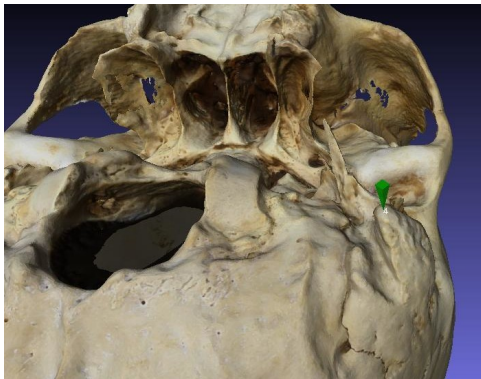
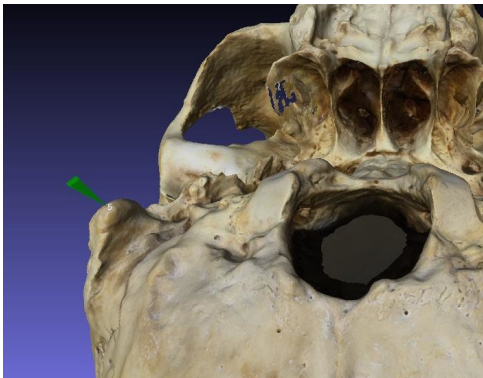
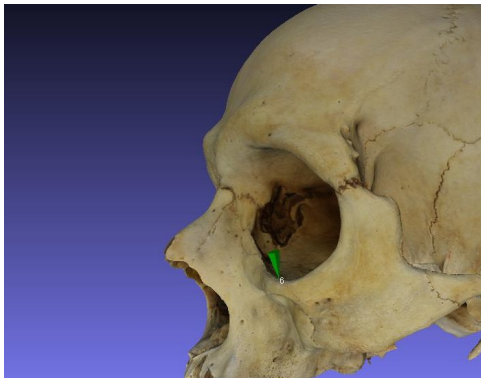

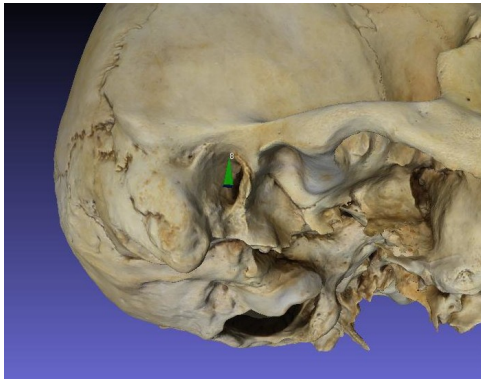
The present document serves both as a user manual as well as an algorithm description document for the **skullanalyzer (v1.0)** computer program. The **skullanalyzer** loads a 3D triangular mesh cranial model along with a number of user-defined landmarks and extracts a set of cranial geometric features, which can be used for further analyses. In order to minimize observer error, the landmarks most sensitive to induce bias into the extracted features are optimized consistently for their correct location on the cranial 3D model. The first part of the document, the user manual, describes how to download, compile, and use the **skullanalyzer** along with the requirements of the cranial 3D models and landmark acquisition for its correct operation. The second part of the document concerns the algorithm description and particularly focuses on the geometric properties of the extracted features as well as the optimization of the *nasion* and *mastoidale* landmarks' positioning on the cranial surface.

About the “skullanalyzer”

The **skullanalyzer** is an open source command line program published under the GNU General Public License v3.0. The source code can be found at <https://github.com/pr0m1th3as/skullanalyzer> along with a precompiled binary for Linux 64-bit operating systems as well as a supplementary GNU Octave function (`plot_features.m`), which can be used within the [GNU Octave](#) programming environment for visualizing the extracted cranial geometric features.

The **skullanalyzer** requires two input files, one Alias Wavefront file (`model_name.obj`) containing a triangular mesh of the cranial model to be analyzed and a Meshlab PickedPoints (`model_name.pp`) sidecar file, which contains the 3D coordinates of a few user-defined landmarks that correspond to the 3D model described in the OBJ file. These landmarks can be easily captured and saved in appropriate format with the open source [Meshlab](#) software and should follow the naming convention shown in Table 1. A prepared testing cranial sample can be download from [here](#).

Table 1: User-defined landmarks and appropriate naming conventions in Meshlab PickedPoint file

Landmark	Naming convention	Landmark	Naming convention
<i>nasion</i>	name=1	<i>bregma</i>	name=2
			
<i>opisthion</i>	name=3	<i>left mastoidale</i>	name=4
			
<i>right mastoidale</i>	name=5	<i>left orbitale</i>	name=6
			
<i>left porion</i>	name=7	<i>right porion</i>	name=8
			

It should be noted that despite the cranial model shown in Table 1 contains texture information, the **skullanalyzer** does not require texture information or any other information that can be embedded in the OBJ file, such as vertex normals, except for the vertex coordinates and their associated faces. Of course, the **skullanalyzer** can handle any OBJ files that meet the Wavefront Alias standard as long as it contains a pure triangular mesh model. One important aspect of the 3D models contained in the OBJ file is that the triangular mesh needs to be relatively dense in order for the program to properly handle all necessary geometric computations. Although there is not a precise limit for minimum area of each face, it is strongly advised that the analyzed 3D mesh should contain at least 100,000 faces. Furthermore, the 3D model should preferably contain only the ectocranial surface of the skull and it should be watertight across the entire cranial vault and at the two mastoid processes if these are to be analyzed.

User-defined landmarks captured with Meshlab's pickpoint tool are illustrated in Figure 1 below. However, in case of existing landmark dataset for a given 3D model in a raw table format or preference of different software for obtaining landmark positions from a given 3D mesh, the user can utilize the GNU Octave function (`write_MeshlabPoints.m`), which is freely available at <https://github.com/pr0m1th3as/long-bone-diaphyseal-CSG-Toolkit>, to save the required list of landmarks along with their respective coordinates to the appropriate Meshlab PickedPoints format.

The screenshot shows the 'Form' dialog box in Meshlab. It has a 'Mode' section with three radio buttons: 'Pick Point' (selected), 'Move Point', and 'Select Point'. There are buttons for 'Load Points From File' and 'Save'. Below this is a table with 5 columns: 'Point Name', 'X', 'Y', 'Z', and 'active'. The table contains 8 rows of data, all with checked boxes in the 'active' column. Below the table are buttons for 'Rename Point', 'Remove Point', 'Clear Point', 'Remove All Points', and 'Undo last move'. There is a 'Template Controls' section with a checkbox 'Save this as your default template', buttons for 'Save', 'Load', 'Clear', and 'Add Point', and a label 'Template Name: No Template Loaded'. At the bottom is a 'Normal Options' section with a checked checkbox 'Show Normal?', a label 'Draw as a:', and two radio buttons: 'Pin' (selected) and 'Line'.

Point Name	X	Y	Z	active
1	47.2995	14.1996	53.623	<input checked="" type="checkbox"/>
2	-2.08849	-73.9684	1.81574	<input checked="" type="checkbox"/>
3	-23.101	64.955	-42.1586	<input checked="" type="checkbox"/>
4	-51.7873	65.9968	6.29839	<input checked="" type="checkbox"/>
5	29.5555	61.2756	-56.8551	<input checked="" type="checkbox"/>
6	14.6742	42.3007	68.2542	<input checked="" type="checkbox"/>
7	-50.1029	40.2579	17.3237	<input checked="" type="checkbox"/>
8	38.0338	33.9293	-48.9844	<input checked="" type="checkbox"/>

Figure 1: All necessary landmarks captured with Meshlab's pickpoint tool

Downloading and compiling the “skullanalyzer”

Downloading and compiling **skullanalyzer** from source is a straightforward procedure on Ubuntu and can be easily achieved through the terminal. The only requirement is having a recent version of the G++ compiler. If you don't have one or not sure, open up a terminal and enter:

```
$ sudo apt install build-essential
```

Downloading source code directly from github repository and compiling the program can be accomplished with the following two commands:

```
$ wget https://github.com/pr0m1th3as/skullanalyzer/raw/master/skullanalyzer.cpp
```

```
$ g++ skullanalyzer.cpp -o skullanalyzer
```

To confirm that **skullanalyzer** has been compiled successfully, you can test the produced binary by entering:

```
$ ./skullanalyzer --help
```

which should print a compact help file on the terminal, or enter:

```
$ ./skullanalyzer --citation
```

to see how you can cite it if useful in academic work :)

If compiling from source is not an option, the user can download a precompiled binary, which has been compiled on Ubuntu 18.04LTS with g++ version 7.4, directly from github and make it executable by entering:

```
$ wget https://github.com/pr0m1th3as/skullanalyzer/raw/master/skullanalyzer
```

```
$ chmod +x skullanalyzer
```

```
$ ./skullanalyzer --help
```

Ubuntu users willing to run the **skullanalyzer** program from any folder in their terminal can move the binary into the `home/user/.local/bin` folder in their home directory. This way, **skullanalyzer** can be called from any directory containing the necessary `.obj` and `.pp` files without being present in the same directory. The extracted geometric features (saved as csv files) will still be saved in the same directory as the input files. Note that in such case, the `./` preceding the program's name should be omitted. Users of other Linux distributions can take the same steps perhaps with slight variation according to the specifics of their distribution, whereas Windows users should be able to use the **skullanalyzer** precompiled binary through a bash emulator like [Cygwin](#) or compile from source with a tool like [MinGW](#) to produce a native Windows executable.

Using the “skullanalyzer”

For the rest of the document, we are assuming that the program binary is placed in a folder present in the system’s PATH variable (such as the `home/user/.local/bin` folder as explained earlier), so it can be called from any directory, which contains the 3D model (`ABH073-Cranium.obj`) and landmark coordinates (`ABH073-Cranium.pp`) files, as follows:

```
$ skullanalyzer ABH073-Cranium.obj ABH073-Cranium.pp
```

Resulting in the following output to the terminal.

```
Loading Vertices and Faces from ABH073-Cranium.obj file... OK
Mesh contains 249919 vertices and 500000 faces.
Reading landmarks from ABH073-Cranium.pp file... OK
Defining anatomical position by Frankfurt plane.
Optimizing landmark coordinates for nasion
User coordinates are: x=47.2995 y=14.1996 z=53.623
Optimal coordinates are: x=47.2884 y=13.9098 z=53.6361
Slicing along sagittal plane at nasion:
Calculating intersection points... OK
  2587 faces were intersected.
Nasion - Opisthocranion horizontal length is 164.268
Raster grid size is 299 rows by 366 columns at a resolution of 0.5mm/pixel.
Extracting Nasion - Bregma midsagittal curve:
  235 polyline 2D coordinates calculated.
Number of harmonics up to 99.99% cumulative power spectral density: 11
Extracting occipital bone height map image:
  18204 faces were retained from a 36.6789mm diameter circular projection area.
  Height Map Image centered on 90 by 90 pixels of 1mm^2 square area each.
Extracting supraorbital ridge height map image:
  2659 faces were retained from a 49.1581mm x 21.9174mm rectangular projection area.
  Height Map Image horizontally aligned on 60 by 50 pixels of 1mm^2 square area each.
Optimizing landmark coordinates for mastoidale left
User coordinates are: x=-51.7873 y=65.9968 z=6.29839
Optimal coordinates are: x=-52.0828 y=66.049 z=5.73321
Extracting left mastoid process lateral height map image:
  9148 faces were retained from a 51.5099mm x 27.439mm rectangular projection area.
  Height Map Image extracted on 100 by 80 pixels of 0.5mm^2 square area each.
Extracting left mastoid process inferior height map image:
  11603 faces were retained from a 30mm x 30mm rectangular projection area.
  Height Map Image extracted on 60 by 60 pixels of 0.5mm^2 square area each.
Optimizing landmark coordinates for mastoidale right
User coordinates are: x=29.5555 y=61.2756 z=-56.8551
Optimal coordinates are: x=28.8134 y=61.5484 z=-56.613
Extracting right mastoid process lateral height map image:
  10830 faces were retained from a 57.6372mm x 28.8312mm rectangular projection area.
  Height Map Image extracted on 100 by 80 pixels of 0.5mm^2 square area each.
Extracting right mastoid process inferior height map image:
  11745 faces were retained from a 30mm x 30mm rectangular projection area.
  Height Map Image extracted on 60 by 60 pixels of 0.5mm^2 square area each.
```

As shown above, the **skullanalyzer** loads both files and goes through all necessary calculations to extract the required geometric features from the cranial 3D mesh while optimizing for *nasion* and *mastoidale* landmarks along the way and provide certain feedback regarding its calculations. However, when called with only 2 input arguments (i.e. the input files) no results are exported into relevant output files, which can be utilized in further analyses. For this purpose, the program can be invoked with a number of parameters set as an extra string argument preceding the input files, e.g.:

```
$ skullanalyzer -peos ABH073-Cranium.obj ABH073-Cranium.pp
```

The program can take up to 4 discrete and independent parameters bundled in a single string argument beginning with a dash (-) as shown with previous command line example. The four letters, “p”, “e”, “o”, “s”, can be set in any random order after the initial (-) character. The following table lists the functionality of each parameter as well as the default behaviour. Note that any other characters are ignored by the **skullanalyzer**.

Table 2: User parameters for controlling the program’s output

Character	Functionality when set in parameter string	Default operation when missing
p	Save the updated list of landmarks including the optimized points into a new separate .pp file.	Does not export updated landmark coordinates.
e	Export the geometric features into the relevant .csv files. This includes the 2D polyline of the <i>nasion-bregma</i> segment as well as all available HMIs.	Does not export the .csv files of any geometric feature.
o	Save all calculated features into GNU Octave text data format. This includes both the EFDs of the <i>nasion-bregma</i> segment as well as all available HMIs.	Does not export the .mat file with all calculated results.
s	Silence output to terminal. Suitable for batch processing several samples through a bash script or other programs.	All requested operations are reported in the terminal.

So, the previous command would save all results and calculations to the appropriate files without producing any output to the terminal, whereas the following command would save exactly the same files in the working directory but also print all feedback regarding its processing steps as shown bellow. Note that the order of characters in the parameter string is irrelevant. Also note that the working directory (where **skullanalyzer** is called from) must always be the directory where the 3D model (.obj) and landmark coordinates (.pp) files are present.

```
$ skullanalyzer -poe ABH073-Cranium.obj ABH073-Cranium.pp
```

```

Loading Vertices and Faces from ABH073-Cranium.obj file... OK
Mesh contains 249919 vertices and 500000 faces.
Reading landmarks from ABH073-Cranium.pp file... OK
Defining anatomical position by Frankfurt plane.
Optimizing landmark coordinates for nasion
User coordinates are: x=47.2995 y=14.1996 z=53.623
Optimal coordinates are: x=47.2884 y=13.9098 z=53.6361
Slicing along sagittal plane at nasion:
Calculating intersection points... OK
    2587 faces were intersected.
Nasion - Opisthocranion horizontal length is 164.268
Writing to ABH073-Cranium_a.pp file... OK
Raster grid size is 299 rows by 366 columns at a resolution of 0.5mm/pixel.
Extracting Nasion - Bregma midsagittal curve:
    235 polyline 2D coordinates calculated.
    Writing to ABH073-Cranium_NasionBregmaSegment.csv file... OK
Number of harmonics up to 99.99% cumulative power spectral density: 11
Extracting occipital bone height map image:
    18204 faces were retained from a 36.6789mm diameter circular projection area.
    Height Map Image centered on 90 by 90 pixels of 1mm^2 square area each.
    Writing to ABH073-Cranium_occipitalHeightMapImage.csv file... OK
Extracting supraorbital ridge height map image:
    2659 faces were retained from a 49.1581mm x 21.9174mm rectangular projection area.
    Height Map Image horizontally aligned on 60 by 50 pixels of 1mm^2 square area each.
    Writing to ABH073-Cranium_supraorbitalHeightMapImage.csv file... OK

```

```

Optimizing landmark coordinates for mastoidale left
User coordinates are: x=-51.7873 y=65.9968 z=6.29839
Optimal coordinates are: x=-52.0828 y=66.049 z=5.73321
Writing to ABH073-Cranium_a.pp file... OK
Extracting left mastoid process lateral height map image:
  9148 faces were retained from a 51.5099mm x 27.439mm rectangular projection area.
  Height Map Image extracted on 100 by 80 pixels of 0.5mm^2 square area each.
  Writing to ABH073-Cranium_mastoidLeftLateralHeightMapImage.csv file... OK
Extracting left mastoid process inferior height map image:
  11603 faces were retained from a 30mm x 30mm rectangular projection area.
  Height Map Image extracted on 60 by 60 pixels of 0.5mm^2 square area each.
  Writing to ABH073-Cranium_mastoidLeftInferiorHeightMapImage.csv file... OK
Optimizing landmark coordinates for mastoidale right
User coordinates are: x=29.5555 y=61.2756 z=-56.8551
Optimal coordinates are: x=28.8134 y=61.5484 z=-56.613
Writing to ABH073-Cranium_a.pp file... OK
Extracting right mastoid process lateral height map image:
  10830 faces were retained from a 57.6372mm x 28.8312mm rectangular projection area.
  Height Map Image extracted on 100 by 80 pixels of 0.5mm^2 square area each.
  Writing to ABH073-Cranium_mastoidRightLateralHeightMapImage.csv file... OK
Extracting right mastoid process inferior height map image:
  11745 faces were retained from a 30mm x 30mm rectangular projection area.
  Height Map Image extracted on 60 by 60 pixels of 0.5mm^2 square area each.
  Writing to ABH073-Cranium_mastoidRightInferiorHeightMapImage.csv file... OK
Data saved to ABH073-Cranium.mat

```

As shown in the terminal output above, the **skullanalyzer** saves each file with an intuitive naming convention by keeping the 3D model's or landmark coordinates' filenames as a baseline. It should be noted that it is not necessary for the two input files to share the same filename as in the present example. All extracted geometric features saved in .csv files acquire their baseline filename (i.e. "ABH073-Cranium") from the 3D model's .obj filename and then a descriptive filename (i.e. *_NasionBregmaSegment*) is appended before the file's extension (.csv). On the contrary, the updated Meshlab PickedPoint file (i.e. "ABH073-Cranium_a.pp") acquires its baseline filename from the landmark coordinates' input file and ("_a") is appended before the file's extension. For example, if the landmark coordinates' input file was named "points.pp" the updated landmark coordinates' output file would have been renamed to "points_a.pp". Finally, the GNU octave text data format file shares the same filename with the 3D model but with a ".mat" extension.

The **skullanalyzer** requires at least 5 landmarks for determining the correct orientation of the cranium model into its anatomical (Frankfurt) position. These are the *nasion*, *opisthion*, *left orbitale*, *left porion* and *right porion* landmarks. Depending on which other landmarks are present into the landmark coordinates' input file, the program will extract a different number of geometric features accordingly. If the minimum number of landmarks is present, then only the *supraorbital ridge* and *occipital protuberance* HMIs will be extracted. Note that the *supraorbital ridge* HMI is only extracted if there is a protruding surface of the *supraorbital ridge* with respect to the *nasion* landmark after it is optimally located. If *bregma* is given, then the *nasion-bregma* segment will be extracted and its EFDs calculated. If left and/or right *mastoidale* landmarks are given, then they are also optimally located and the lateral and inferior HMIs are extracted for each *mastoid process* respectively.

The following table summarizes the possible combinations of present landmark coordinates in the input file and the resulting extracted geometric features. Note, that the same combinations apply to the fields of structures being present in the saved GNU octave text data format file.

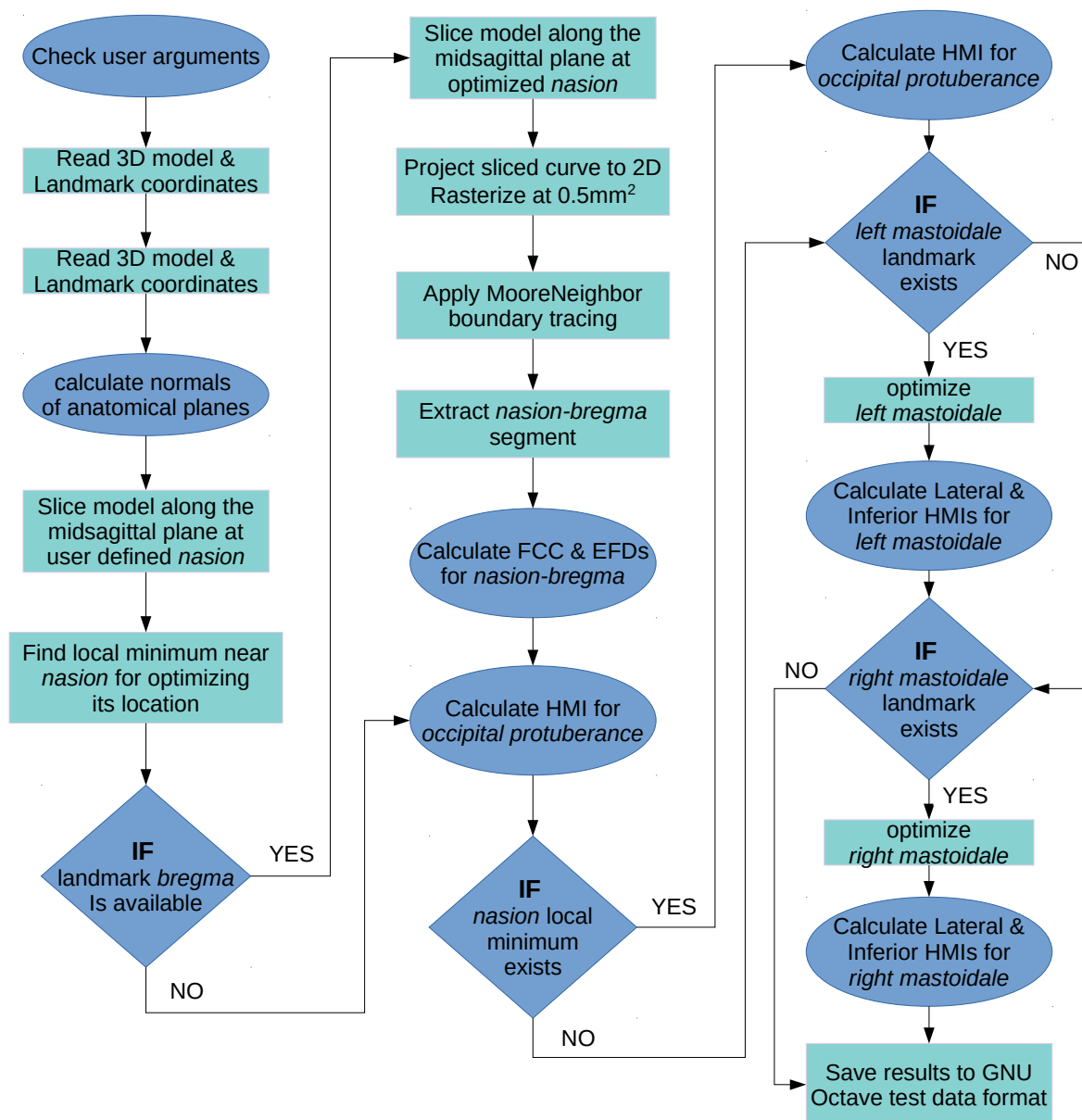
Table 3: Combinations of landmarks and extracted features

Landmarks	Extracted features in csv files	Extracted features in GNU Octave's data structures and respective fields
<i>nasion, opisthion, left orbitale, left porion, right porion</i>	<code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code>	FCC.NasionBregma = 0 EFD.NasionBregma = [0 0 0 0] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60]
<i>nasion, opisthion, bregma, left orbitale, left porion, right porion</i>	<code>_NasionBregmaSegment.csv</code> <code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code>	FCC.NasionBregma = [1:n] EFD.NasionBregma = [n,4] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60]
<i>nasion, opisthion, bregma, left mastoidale, left orbitale, left porion, right porion</i>	<code>_NasionBregmaSegment.csv</code> <code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code> <code>_mastoidLeftLateralHeightMapImage.csv</code> <code>_mastoidLeftInferiorHeightMapImage.csv</code>	FCC.NasionBregma = [1:n] EFD.NasionBregma = [n,4] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60] HMI.LeftMastoidLateral = [80,100] HMI.LeftMastoidInferior = [60,60]
<i>nasion, opisthion, bregma, right mastoidale, left orbitale, left porion, right porion</i>	<code>_NasionBregmaSegment.csv</code> <code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code> <code>_mastoidRightLateralHeightMapImage.csv</code> <code>_mastoidRightInferiorHeightMapImage.csv</code>	FCC.NasionBregma = [1:n] EFD.NasionBregma = [n,4] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60] HMI.RightastoidLateral = [80,100] HMI.RightMastoidInferior = [60,60]
<i>nasion, opisthion, bregma, left mastoidale, right mastoidale, left orbitale, left porion, right porion</i>	<code>_NasionBregmaSegment.csv</code> <code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code> <code>_mastoidLeftLateralHeightMapImage.csv</code> <code>_mastoidLeftInferiorHeightMapImage.csv</code> <code>_mastoidRightLateralHeightMapImage.csv</code> <code>_mastoidRightInferiorHeightMapImage.csv</code>	FCC.NasionBregma = [1:n] EFD.NasionBregma = [n,4] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60] HMI.LeftMastoidLateral = [80,100] HMI.LeftMastoidInferior = [60,60] HMI.RightastoidLateral = [80,100] HMI.RightMastoidInferior = [60,60]
<i>nasion, opisthion, left mastoidale, left orbitale, left porion, right porion</i>	<code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code> <code>_mastoidLeftLateralHeightMapImage.csv</code> <code>_mastoidLeftInferiorHeightMapImage.csv</code>	FCC.NasionBregma = 0 EFD.NasionBregma = [0 0 0 0] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60] HMI.LeftMastoidLateral = [80,100] HMI.LeftMastoidInferior = [60,60]
<i>nasion, opisthion, right mastoidale, left orbitale, left porion, right porion</i>	<code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code> <code>_mastoidRightLateralHeightMapImage.csv</code> <code>_mastoidRightInferiorHeightMapImage.csv</code>	FCC.NasionBregma = 0 EFD.NasionBregma = [0 0 0 0] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60] HMI.RightastoidLateral = [80,100] HMI.RightMastoidInferior = [60,60]
<i>nasion, opisthion, left mastoidale, right mastoidale, left orbitale, left porion, right porion</i>	<code>_occipitalHeightMapImage.csv</code> <code>_supraorbitalHeightMapImage.csv</code> <code>_mastoidLeftLateralHeightMapImage.csv</code> <code>_mastoidLeftInferiorHeightMapImage.csv</code> <code>_mastoidRightLateralHeightMapImage.csv</code> <code>_mastoidRightInferiorHeightMapImage.csv</code>	FCC.NasionBregma = 0 EFD.NasionBregma = [0 0 0 0] HMI.OccipitalProtuberance = [90,90] HMI.SupraorbitalRidge = [50,60] HMI.LeftMastoidLateral = [80,100] HMI.LeftMastoidInferior = [60,60] HMI.RightastoidLateral = [80,100] HMI.RightMastoidInferior = [60,60]

All testing presented in this document has been performed with a 3D model of a cranial sample from the Athens modern reference skeletal collection, which has been digitized by 3D photogrammetry and contains approximately 250,000 vertices. This 3D model sample (ABH073-Cranium.obj) and its sidecar landmark coordinates' file (ABH073-Cranium.pp) bundled together with all extracted features and updated landmark coordinates saved in respective files as shown in Table 3 are freely available as a testing dataset from [zenodo](https://zenodo.org/) repository.

Algorithm Description

The following flowchart illustrates the base components of the algorithm implemented in the **skullanalyzer** for extracting the geometric features from a 3D model.



Optimizing the locations of landmarks

The **skullanalyzer** automatically optimizes the locations of the *nasion*, left *mastoidale* and right *mastoidale* landmarks as a means of minimizing the effect of the observer error in landmark acquisition on the accuracy of the extracted geometric features. It should be noted that different optimization algorithms are implemented for the *nasion* landmark and the *mastoidale* landmarks. Nominally, the *nasion* is defined as the suture between the frontal and nasal bones on the midsagittal plane, which is a distinctly depressed area directly between the eye orbits. This location most often coincides with the most posterior point on the aforementioned depression. Hence, the optimization algorithm for the *nasion* landmark calculates the midsagittal cross section based on the user-defined *nasion* and then finds the local minimum (i.e. the most deep point on this local depression) on the midsagittal curve, since accurate positioning of *nasion* can be problematic in models without a texture component such as those derived from CT scans. A drawback of this optimization approach is that the optimized position is partially dependent on that the user-defined *nasion* defines the midsagittal plane, so the optimization algorithm actually operates along a 2D plane instead of the 3D space that defines the overall local morphology. As a result, the optimized coordinates can exhibit a slight variation between different iterations of user-defined *nasion* landmarks as shown in Table 4. Although the embedded figures depict the optimized *nasion* in the “same” position on the cranial surface, the actual 3D coordinates vary by an average of 0.5mm. This margin of optimization error may be safely considered acceptable, since it produces negligible variation in the extracted *nasion-bregma* segment and the subsequently calculated EFDs as shown in Table 5. However, the user is advised to always inspect the cranial surface for accurately locating the *nasion* landmark on the midline of the skull, which can be easily evaluated even on non-textured models.

The optimization of the *mastoidale* landmark is much more straightforward, since by definition it is the craniometric point at the lowest point of the mastoid process. Since the **skullanalyzer** predetermines the anatomical position of the cranium under analysis, optimizing the left and right *mastoidale* landmarks always results to the same optimized coordinates as shown in Table 6 and Table 7 respectively. The only requirement for accurate optimization is that the user-defined points must be within 8mm radius from their optimal position, which is by far a large margin for easily picking the two *mastoidale* landmarks within the acceptable range allowing the **skullanalyzer** to find their optimal positions. The optimization algorithm implemented for the *mastoidale* landmarks practically eliminates any user induced error in the respective geometric features calculations. However, the user is advised to always take into consideration the maximum radius limitation to

avoid any shortcomings, since the optimization algorithm will only look for the most inferior point coordinates within a radius of 8mm from the user-defined *mastoidale* landmark.

Table 4: Optimized *nasion* iterations with different user defined coordinates

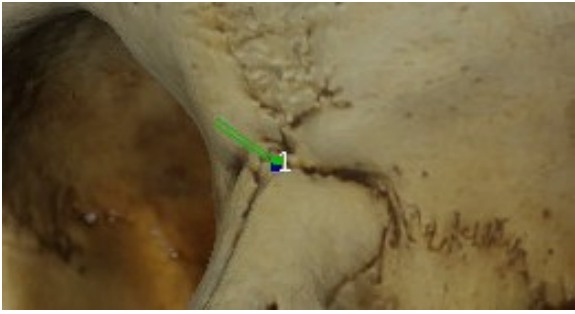

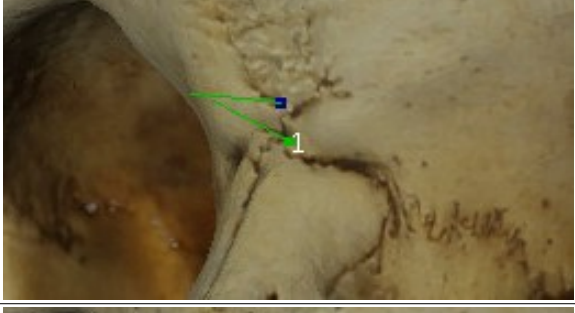

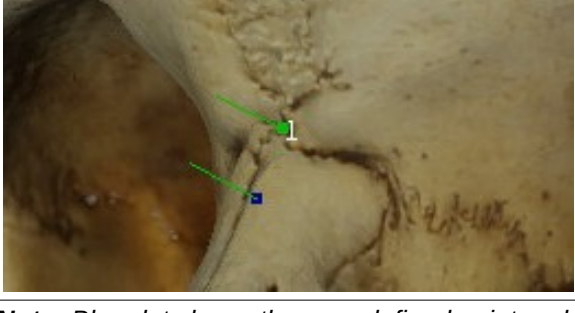
iteration	skullanalyzer terminal output	View in MeshLab
1	User coordinates are: x=47.2995 y=14.1996 z=53.623 Optimal coordinates are: x=47.2884 y=13.9098 z=53.6361	
2	User coordinates are: x=47.7112 y=16.1132 z=54.1253 Optimal coordinates are: x=46.9836 y=13.179 z=53.4614	
3	User coordinates are: x=47.1697 y=11.5387 z=54.589 Optimal coordinates are: x=46.6905 y=13.4176 z=53.7855	
4	User coordinates are: x=47.6101 y=15.4702 z=54.0303 Optimal coordinates are: x=46.996 y=13.1877 z=53.4518	
5	User coordinates are: x=47.9655 y=16.9616 z=54.1944 Optimal coordinates are: x=47.0849 y=13.142 z=53.4183	
<p>Note: The optimized nasion points can vary depending on the initial user-defined alignment points.</p>		<p>Note: Blue dot shows the user-defined point and green dot shows the optimized location of the left nasion landmark.</p>

Table 5: Optimized *nasion-bregma* segments and resulted EFDs per iteration

iteration	<i>nasion-bregma</i> segment	Calculates EFDs			
1		-49.159	0	105.5	0
		65.219	-1.63E-13	-72.042	8.77E-14
		-16.295	9.76E-14	-14.504	7.47E-15
		5.4359	-8.30E-14	-5.6362	-7.45E-15
		-2.7125	3.03E-14	-2.41	-6.94E-15
		2.2255	-4.56E-14	-2.5855	1.53E-14
		-1.6793	2.52E-14	-0.78335	-1.30E-14
		0.27917	-1.93E-14	-1.4981	1.22E-14
		-1.4789	2.10E-14	-0.64613	-6.51E-15
		-0.035447	-1.82E-14	-0.92736	5.16E-15
		-0.94211	1.54E-14	-0.28116	-9.04E-15
		0.066239	-1.39E-14	-0.57783	7.73E-15
		2		-48.56	0
65.228	-1.57E-13			-70.967	7.74E-14
-15.901	1.11E-13			-14.468	1.88E-14
5.5702	-4.64E-14			-5.7129	-2.19E-14
-2.6226	2.15E-14			-2.3897	-9.25E-15
2.3193	-3.73E-14			-2.5431	1.04E-14
-1.4699	2.14E-14			-0.78517	-1.31E-14
0.43493	-1.93E-14			-1.5156	1.56E-14
-1.4121	2.11E-14			-0.64264	-4.55E-15
-0.0079031	-1.44E-14			-0.92834	9.73E-15
-0.90067	1.69E-14			-0.32344	-8.16E-15
0.064687	-1.12E-14			-0.62079	1.15E-14
3				-48.465	0
		65.332	-1.54E-13	-71.532	9.09E-14
		-16.057	9.99E-14	-14.448	1.53E-14
		5.3782	-5.77E-14	-5.5415	-4.56E-15
		-2.6092	2.65E-14	-2.3643	-1.01E-14
		2.3289	-4.15E-14	-2.6472	1.61E-14
		-1.5874	2.29E-14	-0.77705	-1.13E-14
		0.31434	-2.21E-14	-1.4414	1.13E-14
		-1.4209	2.06E-14	-0.6557	-3.86E-15
		0.011275	-1.94E-14	-0.96528	7.88E-15
		-0.94812	1.90E-14	-0.29846	-3.09E-15
		0.018646	-1.51E-14	-0.56953	5.75E-15
		4		-48.56	0
65.228	-1.58E-13			-70.969	7.13E-14
-15.899	1.13E-13			-14.462	1.81E-14
5.5674	-4.50E-14			-5.7198	-1.81E-14
-2.622	1.99E-14			-2.3883	-8.87E-15
2.3225	-3.68E-14			-2.5355	1.02E-14
-1.4757	2.21E-14			-0.799	-1.11E-14
0.44005	-1.94E-14			-1.5033	1.52E-14
-1.4136	2.07E-14			-0.64626	-4.65E-15
-0.01061	-1.51E-14			-0.93487	8.51E-15
-0.89589	1.67E-14			-0.31189	-8.53E-15
0.060885	-1.11E-14			-0.62997	1.18E-14
5				-48.597	0
		65.302	-1.57E-13	-71.037	1.61E-13
		-15.905	1.22E-13	-14.425	1.72E-14
		5.4804	-7.06E-14	-5.6376	-1.11E-15
		-2.581	2.87E-14	-2.3831	-3.82E-15
		2.3514	-3.56E-14	-2.5855	1.38E-14
		-1.4953	2.02E-14	-0.77302	-8.68E-15
		0.4189	-1.61E-14	-1.4713	9.24E-15
		-1.378	1.70E-14	-0.65406	-8.36E-15
		0.0038365	-1.73E-14	-0.95484	1.21E-14
		-0.92374	2.02E-14	-0.30248	-9.95E-15
		0.056692	-2.33E-14	-0.59701	5.90E-15
		Note: The resulted polyline segments are nearly identical between different iterations.		Note: Variation in the calculated EFDs is insignificant as compared to EFD differences between different individuals.	

Table 6: Optimized left *mastoidale* iterations with different user defined coordinates

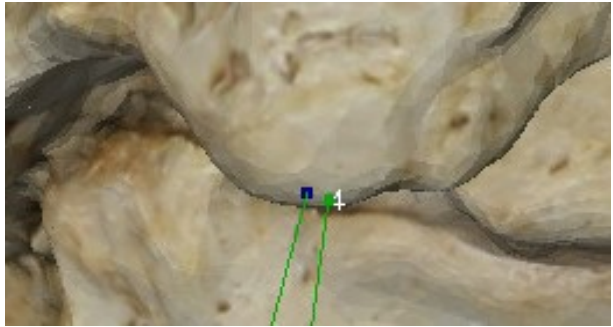
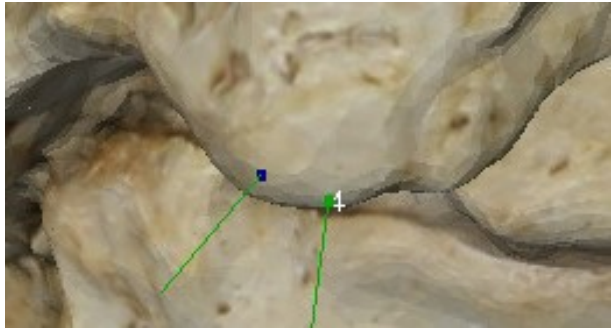








iteration	GNU Octave's console	View in MeshLab
1	User coordinates are: $x=-51.7873$ $y=65.9968$ $z=6.29839$ Optimal coordinates are: $x=-52.0828$ $y=66.049$ $z=5.73321$	
2	User coordinates are: $x=-51.0715$ $y=65.703$ $z=7.46316$ Optimal coordinates are: $x=-52.0828$ $y=66.049$ $z=5.73321$	
3	User coordinates are: $x=-52.267$ $y=65.6986$ $z=7.38393$ Optimal coordinates are: $x=-52.0828$ $y=66.049$ $z=5.73321$	
4	User coordinates are: $x=-50.8507$ $y=65.7805$ $z=6.80932$ Optimal coordinates are: $x=-52.0828$ $y=66.049$ $z=5.73321$	
5	User coordinates are: $x=-53.372$ $y=65.6583$ $z=6.76208$ Optimal coordinates are: $x=-52.0828$ $y=66.049$ $z=5.73321$	
<p>Note: The optimized left mastoidale coordinates are always the same regardless of how accurately the user may initially pick the landmark.</p>		<p>Note: Blue dot shows the user defined point and green dot shows the optimized location of the left mastoidale landmark.</p>

Table 7: Optimized right *mastoidale* iterations with different user defined coordinates

iteration	GNU Octave's console	View in MeshLab
1	User coordinates are: $x=29.5555$ $y=61.2756$ $z=-56.8551$ Optimal coordinates are: $x=28.8134$ $y=61.5484$ $z=-56.613$	
2	User coordinates are: $x=30.5673$ $y=60.8209$ $z=-56.0152$ Optimal coordinates are: $x=28.8134$ $y=61.5484$ $z=-56.613$	
3	User coordinates are: $x=29.0715$ $y=61.0765$ $z=-55.2264$ Optimal coordinates are: $x=28.8134$ $y=61.5484$ $z=-56.613$	
4	User coordinates are: $x=30.4065$ $y=60.9039$ $z=-55.6063$ Optimal coordinates are: $x=28.8134$ $y=61.5484$ $z=-56.613$	
5	User coordinates are: $x=28.9993$ $y=60.8051$ $z=-54.7697$ Optimal coordinates are: $x=28.8134$ $y=61.5484$ $z=-56.613$	
<p>Note: The optimized right mastoidale coordinates are always the same regardless of how accurately the user may initially pick the landmark.</p>		<p>Note: Blue dot shows the user defined point and green dot shows the optimized location of the right mastoidale landmark.</p>

Elliptic Fourier Descriptors of the *nasion* – *bregma* segment

The *nasion-bregma* segment is the 2D planar projection of the ectocranial mesh sliced along the midsagittal plane, which passes through the optimized *nasion* landmark. The projection produces a right hand side view of the skull, as shown in Figure 2, with the horizontal axis aligned with the cranial dorsoventral (anteroposterior) axis and the vertical axis aligned with the craniocaudal axis. The origin of the 2D projection plane coincides with the projection of the *nasion* landmark. The segment starts at the optimized *nasion* and ends at the point along the sliced mesh, which is closer to *bregma*. It should be noted that since the *nasion* optimization algorithm is not completely rigid but introduces some small error as previously stated, this means that the starting point of the *nasion-bregma* segment does not necessarily start at the deepest point of the local depression. This may result in minor artifacts near the origin of the 2D plot as illustrated in the example shown below. However, this minute variation may be regarded as non-significant as explained earlier. The extracted 2D segment is saved in the respective *_NasionBregmaSegment.csv file as a two column matrix, where the columns refer to x (horizontal) and y (vertical) dimensions of the 2D projection and each row at a different projected point of the sliced segment.

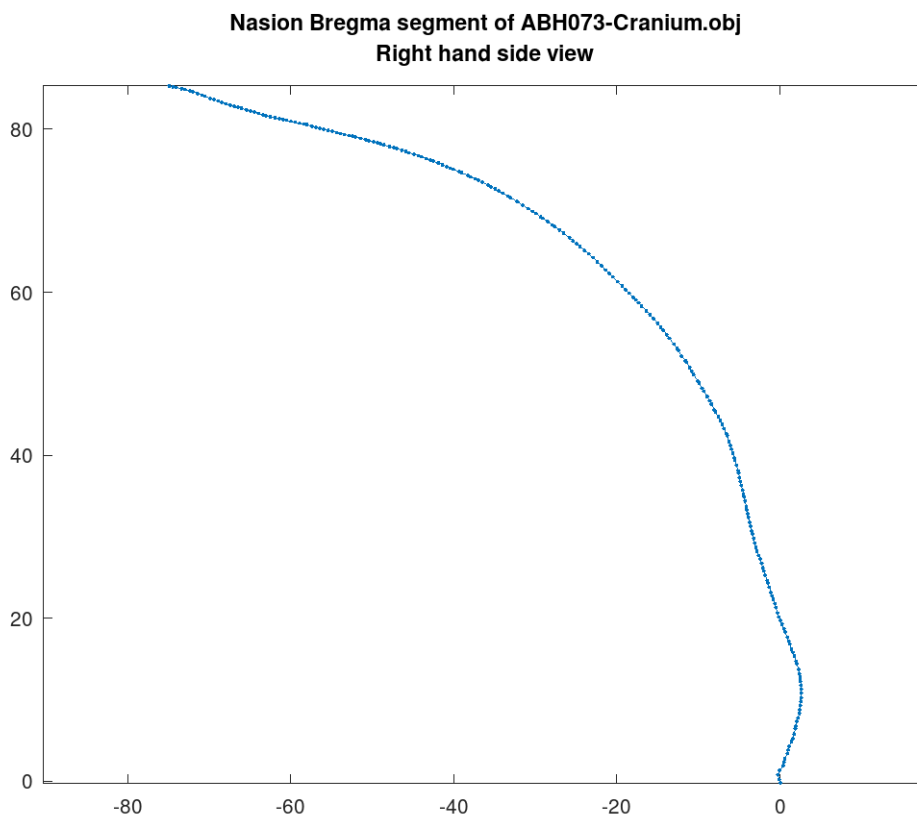


Figure 2: *nasion-bregma* segment plotted in GNU Octave

The calculation of the EFDs relies on the prior calculation of Freeman Chain Code (FCC) that describes the particular extracted segment. The FCC is derived from the raster image of the 2D

projection, which is rasterized at a resolution of 0.5mm. This further lowers the impact of the error of the *nasion* optimization algorithm. Finally, the FCC of the *nasion-bregma* segment is calculated from *nasion* to *bregma* and back to *nasion* to produce a closed contour, which facilitates the calculation of the EFDs. The number of harmonics of the EFDs are calculated up to their 99.99% cumulative power spectral density. This enables an accurate representation of the 2D *nasion-bregma* segment without adding many higher harmonics, which mostly represent quantization noise rather than actual shape variation. Furthermore, although a full range Fourier expansion is applied on the previously calculated closed contour FCC, the resulted EFDs correspond to a half range Fourier expansion applied on an open line. Hence, the **b** and **d** coefficients of the EFDs are zero, as it is demonstrated in Table 5. Note that these values are nearly zero as a result of floating-numbers operations, but they are practically zero and should not be considered for any further analysis.

Height Map Images: orientation and rasterization details

Occipital Protuberance HMI

The HMI of the *occipital protuberance* area is the negative height map of a circular projection disk, whose diameter is determined by the user-defined *opisthion* landmark and the *pseudo-opistocranion* landmark, which is automatically calculated by the **skullanalyzer** as the most distant point from the optimized *nasion* along the dorsoventral axis at the back of the skull. The projection produces an inferoposterior view of the skull, as shown in Figure 3. The HMI's horizontal axis is aligned with the cranial frontal (left-to-right) axis and its vertical axis is aligned with the vector defined by *pseudo-opisthion* (top) and *opisthion* (bottom) landmarks.

Occipital Protuberance height map of ABH073-Cranium.obj
Posterior view

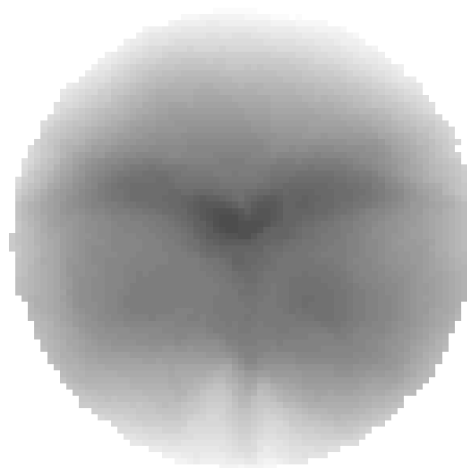


Figure 3: occipital protuberance HMI plotted in GNU Octave

The *occipital protuberance* HMI is an 8-bit gray scale image with a fixed size of 90x90 pixels. Each pixel covers a 1mm² square area upon which the circular projection disk is centered. The 8-bit projected height resolution of the *occipital protuberance* HMI is set at 0.1mm and projected heights in excess of 25.5mm are set to zero. The extracted HMI is saved in the respective *_occipitalHeightMapImage.csv file as a 90x90 (rows, columns) matrix containing values from 0 to 255. By negative height map it is meant that the faces further away from the projection disk are assigned with lower values. Hence, the more pronounced the occipital protuberance is, the darker it will appear on the HMI plot.

Supraorbital Ridge HMI

The HMI of the *supraorbital ridge* area is the negative height map of a rectangular projection area, which is located at the optimized *nasion* landmark with coronal orientation. The projected cranial mesh concerns only the faces that lie anterosuperiorly of the optimized *nasion* landmark. This projection produces an anterior view of the skull, as shown in Figure 4. The HMI's horizontal axis is aligned with the cranial frontal (right-to-left) axis and its vertical axis is aligned with the craniocaudal axis. Note that the left side of the HMI corresponds to the right hand side of the skull.

Supraorbital Ridge height map of ABH073-Cranium.obj
Anterior view

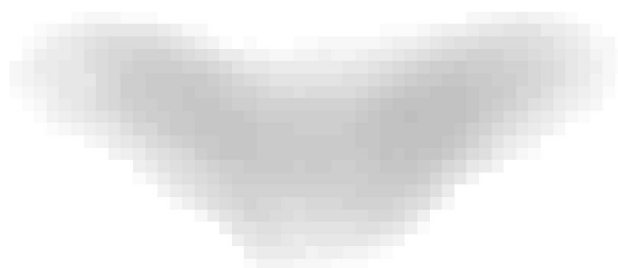


Figure 4: supraorbital ridge HMI plotted in GNU Octave

The projected faces are horizontally centered on a rectangular projection plate of 60mm width and 50mm height with the optimized *nasion* landmark positioned at the midpoint of its base (horizontal) edge. The *supraorbital ridge* HMI is an 8-bit gray scale image with a fixed size of 60x50 pixels, each of which cover a 1mm² square area. The 8-bit projected height resolution of the

supraorbital ridge HMI is set at 0.05mm and projected heights in excess of 12.75mm are set to zero. The extracted HMI is saved in the respective *_supraorbitalHeightMapImage.csv file as a 50x60 (rows, columns) matrix containing values from 0 to 255. Once again, the negative height map means that the more pronounced the supraorbital ridge is, the darker it will appear on the HMI plot.

Left Mastoid Process Lateral HMI

The left *mastoid process* lateral HMI is the negative height map of a rectangular projection area, which is located laterally of the left *mastoidale* landmark displaced by 20mm and parallel to the sagittal plane. The projected cranial mesh concerns only the faces that lie inferiorly and posteriorly of the left *porion* landmark displaced anteriorly by 3mm and laterally of the left *mastoidale* landmark displaced medially by 5mm. This projection produces a lateral view of the skull from the left hand side, as shown in Figure 5. The HMI's horizontal axis is aligned with the cranial dorsoventral axis and its vertical axis is aligned with the craniocaudal axis. Note that the left side of the HMI corresponds to the anterior side of the skull.

Left Mastoid Process height map of ABH073-Cranium.obj
Lateral view

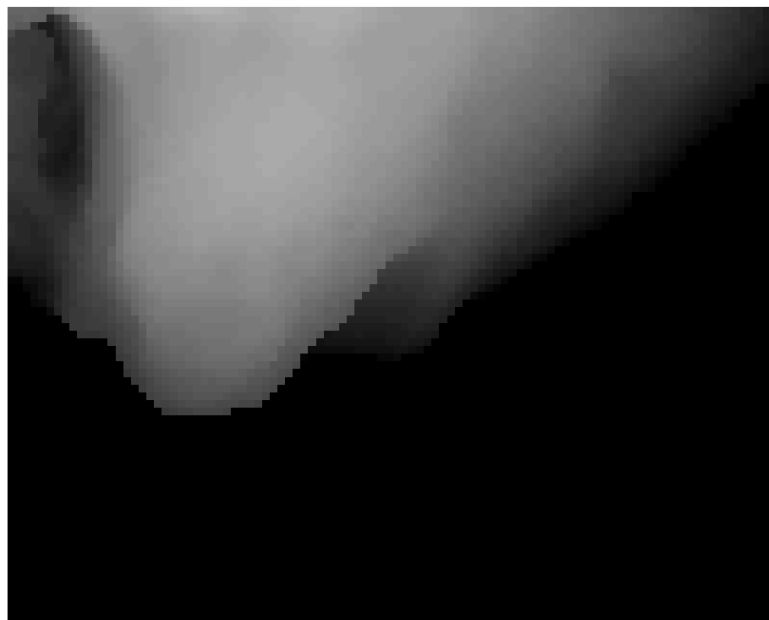


Figure 5: left mastoid process lateral HMI plotted in GNU Octave

The projected faces are top-left aligned on a rectangular projection plate of 50mm width and 40mm height. The left *mastoid process* lateral HMI is an 8-bit gray scale image with fixed dimensions of 100x80 pixels and a spatial resolution of 0.5mm. The 8-bit projected height resolution of the left *mastoid process* lateral HMI is set at 0.1mm and projected heights in excess of 25.5mm are set to zero. The extracted HMI is saved in the respective

*_mastoidLeftLateralHeightMapImage.csv file as a 80x100 (rows, columns) matrix containing values from 0 to 255.

Left Mastoid Process Inferior HMI

The left *mastoid process* inferior HMI is the negative height map of a square projection area, which is located inferiorly of the left *porion* landmark displaced by 40mm and parallel to the transverse plane. The projected cranial mesh concerns only the faces that lie inferiorly of the left *porion* landmark and are bounded by a square area of 30mm x 30mm with its center defined by the left *mastoidale* landmark and extruded along the craniocaudal axis. This projection produces an inferior view of the skull centrally aligned below the left *mastoidale*, as shown in Figure 6. The HMI's horizontal axis is aligned with the cranial frontal axis and its vertical axis is aligned with the dorsoventral axis. Note that the left side of the HMI corresponds to the medial side of the skull, whereas the top faces anteriorly.

Left Mastoid Process height map of ABH073-Cranium.obj
Inferior view

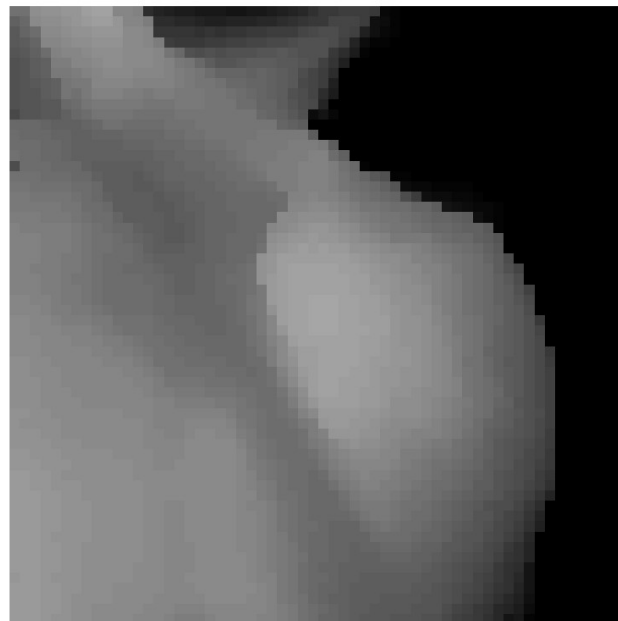


Figure 6: left mastoid process inferior HMI plotted in GNU Octave

The projected faces are centered on a square projection plate of 30mm width and 30mm height. The left *mastoid process* inferior HMI is an 8-bit gray scale image with a fixed size of 60x60 pixels and a spatial resolution of 0.5mm. The 8-bit projected height resolution of the left *mastoid process* HMI is set at 0.15mm and projected heights in excess of 38.25mm are set to zero. The extracted HMI is saved in the respective *_mastoidLeftInferiorHeightMapImage.csv file as a 60x60 (rows, columns) matrix containing values from 0 to 255.

Right Mastoid Process Lateral HMI

The right *mastoid process* lateral HMI is the negative height map of a rectangular projection area, which is located laterally of the right *mastoidale* landmark displaced by 20mm and parallel to the sagittal plane. The projected cranial mesh concerns only the faces that lie inferiorly and posteriorly of the right *porion* landmark displaced anteriorly by 3mm and laterally of the right *mastoidale* landmark displaced medially by 5mm. This projection produces a lateral view of the skull from the right hand side, as shown in Figure 7. The HMI's horizontal axis is aligned with the cranial dorsoventral axis and its vertical axis is aligned with the craniocaudal axis. Note that the right side of the HMI corresponds to the anterior side of the skull.

Right Mastoid Process height map of ABH073-Cranium.obj
Lateral view

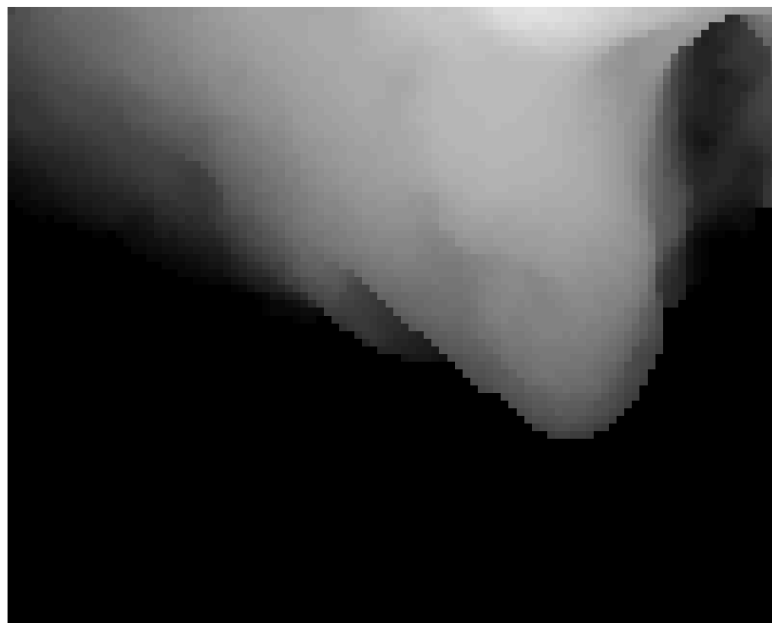


Figure 7: right mastoid process lateral HMI plotted in GNU Octave

The projected faces are top-right aligned on a rectangular projection plate of 50mm width and 40mm height. The right *mastoid process* lateral HMI is an 8-bit gray scale image with fixed dimensions of 100x80 pixels and a spatial resolution of 0.5mm. The 8-bit projected height resolution of the right *mastoid process* lateral HMI is set at 0.1mm and projected heights in excess of 25.5mm are set to zero. The extracted HMI is saved in the respective *_mastoidRightLateralHeightMapImage.csv file as a 80x100 (rows, columns) matrix containing values from 0 to 255.

Right Mastoid Process Inferior HMI

The right *mastoid process* inferior HMI is the negative height map of a square projection area, which is located inferiorly of the right *porion* landmark displaced by 40mm and parallel to the transverse plane. The projected cranial mesh concerns only the faces that lie inferiorly of the right *porion* landmark and are bounded by a square area of 30mm x 30mm with its center defined by the right *mastoidale* landmark and extruded along the craniocaudal axis. This projection produces an inferior view of the skull centrally aligned below the right *mastoidale*, as shown in Figure 8. The HMI's horizontal axis is aligned with the cranial frontal axis and its vertical axis is aligned with the dorsoventral axis. Note that the right side of the HMI corresponds to the medial side of the skull, whereas the top faces anteriorly.

Right Mastoid Process height map of ABH073-Cranium.obj
Inferior view

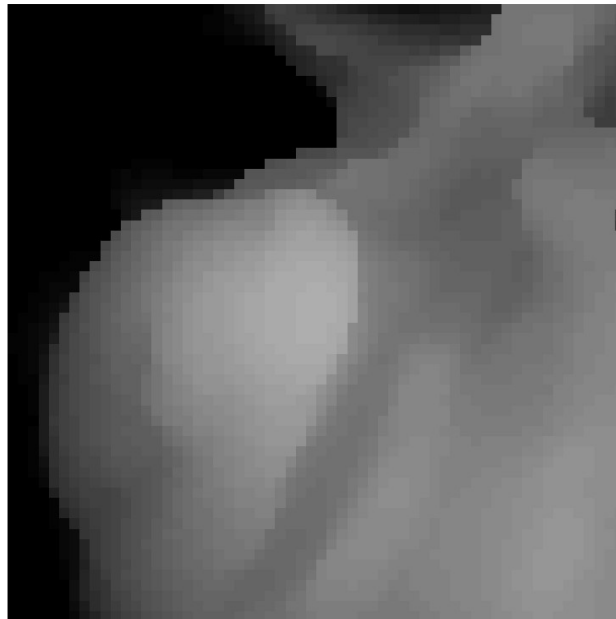


Figure 8: right mastoid process inferior HMI plotted in GNU Octave

The projected faces are centered on a square projection plate of 30mm width and 30mm height. The right *mastoid process* inferior HMI is an 8-bit gray scale image with a fixed size of 60x60 pixels and a spatial resolution of 0.5mm. The 8-bit projected height resolution of the right *mastoid process* HMI is set at 0.15mm and projected heights in excess of 38.25mm are set to zero. The extracted HMI is saved in the respective *_mastoidRightInferiorHeightMapImage.csv file as a 60x60 (rows, columns) matrix containing values from 0 to 255.

Limitations of the “skullanalyzer” program

As mentioned earlier the **skullanalyzer** has certain limitations regarding the properties of the cranial 3D mesh models that it can handle properly. These concern the amount of faces and the absence of holes from particular areas of the ectocranial surface. Although a completely watertight mesh is the most safe approach to avoid segmentation errors, it is not mandatory. For example, the cranial model used in the present manual is not watertight (i.e. *foramen magnum*). Furthermore, the **skullanalyzer** has been implemented in such manner that it can handle cranial meshes that may also contain endocranial geometry, such as a double layer configuration or a folded surface, which can be the case with 3D models produced from CT scans etc. This is achieved by utilizing the Moore Neighbor tracing algorithm to automatically detect the outer surface of the skull along the midsagittal curve. This algorithm poses a limitation in that the midsagittal curve must be rasterized in order to apply the Moore Neighbor tracing, which further implies that any small holes or sparse faces on the ectocranial surface can be problematic. Furthermore, the rasterization and Moore Neighbor tracing are also responsible for the small error in the *nasion* optimization. These limitations have been partially compensated with particular techniques such as resampling, which have been implemented in the **skullanalyzer**. However, the user is strongly advised to always check the quality and appropriateness of the 3D mesh prior to analysis and use the supplementary GNU Octave function to visualize the extracted geometric features prior to utilizing the HMIs or the .mat file for any further analysis.

Known Bugs

A known issue concerns the HMI calculation of the inferior view of the *mastoid processes*, which often results in miscalculation of the grayscale value of a couple of vertical border pixels of the extracted HMI. Despite all efforts to debug the underlying implemented algorithm of the HMI extraction, which is fairly complex, this miscalculation persists every now and then. This erratic behaviour **explicitly** appears on the inferior view of the *mastoid processes*, is always limited to the medial border of the projected height map and affects a very small number of pixels if any. However, this bug does not interfere with the overall view of mastoid process and hence it may be regarded as non-significant for all practical purposes such as image pattern recognition.

Acknowledgements

The author would like to thank Nefeli Garoufi and Dr. Maria-Eleni Chovalopoulou for revising the present manual and offering valuable suggestions to improve its readability.