Visage Technologies
FACE TRACKING & ANALYSIS

VisageTracker Configuration Manual

version 8.5

Visage Technologies AB

www.visagetechnologies.com

Contents

1. Introduction

This manual is meant for users who wish to take advantage of advanced functionalities that can be obtained from the tracker using custom configuration files.

The tracker is fully configurable through an extensive set of parameters in easily manageable configuration files. Each configuration file fully defines the tracker operation, in effect customizing the tracker for a particular application.

The configuration file is loaded every time the new tracking session is started by calling the VisageTracker::track() function. It is possible to change the configuration file between tracking sessions using VisageTracker::setTrackerConfigurationFile().

Furthermore, the configuration files in the same format are also used for facial features detection though in this case only a subset of configuration parameters is used. At the moment, Face Detector.cfg is used and it is not possible to change configuration name for facial features detection.

1.1. Standard configuration files

visage|SDK comes with several standard configuration files aimed at common usage scenarios.
Table 1. provides an overview of all available configurations.

Table 1. Standard configuration files

Configuration file name	**Overview**
Head Tracker.cfg	Optimized for high performance head pose tracking.
Facial Features Tracker - High.cfg	Facial features tracker optimized for real time operation from camera or video files on high performance mobile devices such as iPhone6 as well as all desktop/laptop computers. Tracks head pose, mouth, eyebrows and eye motion.
Facial Features Tracker - Low.cfg	Facial features tracker optimized for real time operation from camera or video files on low performance mobile devices such as iPhone4S. Tracks head pose, mouth, eyebrows and eye motion.
Face Detector.cfg	Used in face detection.

2. Customizing the tracker

Information in this chapter allows users to create own application-specific tracker configurations.

2.1. Configuration parameters

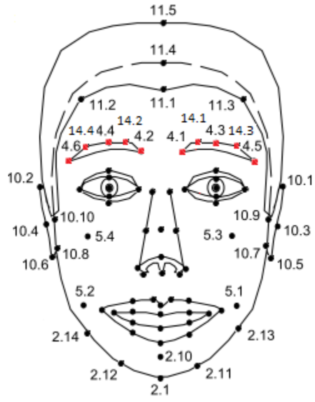
The following table provides the detailed description of parameters defined in the configuration file and their usage. Some parameters are available only on specific platform marked in table as "WIN" for Windows, "IOS" for iOS , "AND" for Android, "MAC" for macOS, "LIN" for Linux and "HTML5" for HTML5. Furthermore, the labels "TRACKER" and "DETECTOR" in the table indicate whether the parameter influences VisageTracker or VisageFeaturesDetector.

Table 2. Configuration parameters

Parameter name	Description
Parameters controlling tracker initialization and recovery	
min_face_scale [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER]	This value controls the lower limit for face scale search range used during initialization and recovery. It is defined as decimal fraction [0.0 - 1.0] of the input image size, where image size is defined as smaller of the image's width and height. For example, if min_face_scale is set to 0.1 and image dimensions are 800x600, smallest face that will be searched for will be 0.1 x min (800, 600) = 60px.
max_face_scale [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER]	This value controls the upper limit of face scale search range used during initialization and recovery. It is defined as decimal fraction [0.0 - 1.0] of the input image size, where image size is defined as smaller of the image's width and height. For example, if max_face_scale is set to 0.8 and image dimensions are 800x600, largest face that will be searched for will be 0.8 x min (800, 600) = 480px.
face_detector_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	This value controls the face detector sensitivity (TPR) for VisageFeaturesDetector detections and VisageTracker initializations. Valid values for this parameter are from 0 to 1. Setting the parameter to 1 will ensure maximal achievable true positive rate, but it will result with large amounts of false positive detections. Setting it closer to 0 will ensure lower amounts of false positives, but also lower number of true positive detections.
recovery_timeout [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER]	This value is used when the tracker loses the face and cannot detect any face in the frame. This value tells the tracker how long it should wait before considering that the current user is gone and initializing the full re-initialization procedure. If the face is detected before this time elapses, the tracker considers that it is the same person and recovers, i.e. continues tracking it using the previous settings. The time is expressed in milliseconds.
Parameters controlling the smoothing filter	
smoothing_factors [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER]	The tracker can apply a smoothing filter to the tracking results to reduce the inevitable tracking noise. Smoothing is preformed using multiple filters which range from the strongest filter (maximal smoothing, longest delay) to weakest (highest response, less delay). An adaptive combination of filters is used, maximizing stability when the face is still while reducing delay when the face moves. Still, smoothing inevitably introduces some delay so it should be used sparingly. Smoothing factors will affect the weight that is given to each filter, with higher values giving higher weight to the strongest filter. Values can range between 0 and 10. The value 0 provides minimal smoothing and highest response (lowest delay). The value 10 provides maximal smoothing and lowest response (longest delay). Negative value disables smoothing completely for specific group. Our recommended range for all groups is from 0.5 to 2.0.

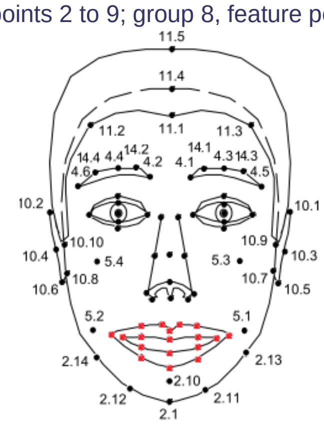
Smoothing is applied only on the detected feature points (2D points) but it also affects the 3D data indirectly.
Smoothing factors are set separately for the following groups of tracking results, one factor value for each group:

Eyebrows:
Applies smoothing to parameters that represent eyebrow movement.
The following members of FaceData::featurePoints2D are directly affected by this factor: group 4, feature points 1 to 6; group 14, feature points 1 to 4.



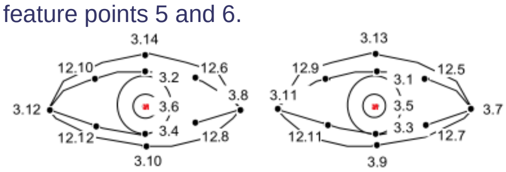
Feature points affected by eyebrows smoothing factor marked red

Mouth:
Applies smoothing to parameters that represent mouth movement.
The following members of FaceData::featurePoints2D are directly affected by this factor: group 2, feature points 2 to 9; group 8, feature points 1 to 10.



Feature points affected by mouth smoothing factor marked red

Pupils:
Applies smoothing to parameters that represent pupil movement (indirectly affects the responsiveness of gaze direction estimation).
The following members of FaceData::featurePoints2D are directly affected by this factor: group 3, feature points 5 and 6.



Feature points affected by pupils smoothing factors marked red

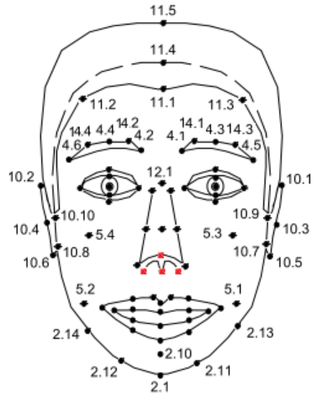
Eyelids:
Applies smoothing to parameters that represent eyelid region movement (indirectly affects responsiveness of eye closure estimation).
The following members of FaceData::featurePoints2D are directly affected by this factor: group 3, all feature points except 5 and 6 (pupils); group 12, feature points 5 to 12.

Feature points affected by eyelids smoothing factor marked red

Nose:

Applies smoothing to parameters that represent nose movement.

The following members of FaceData::featurePoints2D are directly affected by this factor: group 9, feature points 3 to 5 and feature point 15.

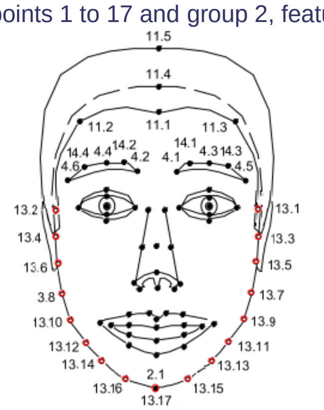


Feature points affected by nose smoothing factor marked red

Visible face contour and chin:

Applies smoothing to parameters that represent contour of the face and chin.

The following members of FaceData::featurePoints2D are directly affected by this factor: group 13, feature points 1 to 17 and group 2, feature point 1.

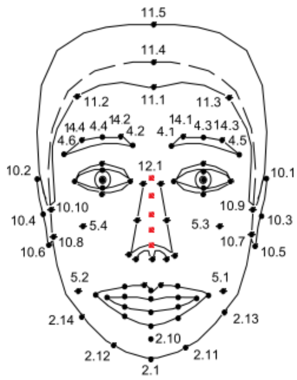


Feature points affected by face contour and chin smoothing factor marked red

Visible nose contour:

Applies smoothing to parameters that represent contour of the nose.

The following members of FaceData::featurePoints2D are directly affected by this factor: group 14, feature points 21 to 25.



Feature points affected by nose contour smoothing factor marked red

Screen space gaze:

Applies smoothing to parameters that represent screen space gaze position.

The following members of FaceData::gazeData are directly affected by this factor:

	x, y.
Data parameters and paths	
bdts_data_path [WIN, IOS, AND, MAC, LIN] [TRACKER]	Path to the folder containing the detector data files (*.bdf) It is **relative** to the location of the configuration file. In the current distribution these files are contained in the folder Samples/data/bdtsdata. **NOTE:** For HTML5 bdts_data_path is "bdtsdata" and it cannot be changed.
camera_focus [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Focal length of a pinhole camera model used as approximation for the camera used to capture the video in which tracking is performed. The value is defined as distance from the camera (pinhole) to an imaginary projection plane where the smaller dimension of the projection plane is defined as 2, and the other dimension is defined by the input image aspect ratio. Thus, for example, for a landscape input image with aspect ratio of 1.33 the imaginary projection plane has height 2 and width 2.66. See section 2.2.2. "Estimating the camera focus" for further details.
Parameters related to the 3D face fitting and 3D models used Because the tracker and detector yield only 2D points, visage|SDK uses 3D facial models to estimate the 3D information such as head pose, 3D facial points, Action Units or full 3D facial mesh. Depending on application requirements, up to three different models may be used: one for head pose estimation, one for Action Units estimation and one for 3D mesh fitting. For performance/data size/memory footprint reasons, it is recommended to use only the models corresponding to the functionality required by the application - for example, if the application requires only 3D head pose but not the 3D mesh nor action units, use only one model and disable others (note that it is possible to disable Action Units and 3D model fitting within the application – see VisageTracker::setTrackerConfiguration). Furthermore, models can be customized or completely replaced by custom-built ones if so required by specific applications - see section 2.3 for details. The following parameters are used to specify which models are used.	
enable_fitting [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Controls if 3D face fitting will be performed or not. Setting this parameter to 1 enables 3D face fitting, while setting it to 0 disables it. Default value is 1. **NOTE**: When 3D face fitting is disabled, parts of the tracking output will cease to be avaiable in FaceData. For more details, please see FaceData documention.
pose_fitting_model [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	File name of the 3D model used to estimate the 3D head pose (returned in FaceData::faceTranslation and FaceData::faceRotation). This model is required for the functioning of the tracker and should not be disabled; it is recommended to use the default one as set in the tracking configurations shipped in visage|SDK. The file name may contain a path, and it must be relative to the location of the configuration file. In face detector configurations, it is possible to disable this model by setting this parameter to "none" or simply removing it from the configuration; this will yield a small gain in data size, memory footprint and performance. **NOTE:** HTML5 version does not support relative paths. Provide only name of model file (e.g. jk_300.wfm).
pose_fitting_fdp [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Name of the MPEG-4 feature Points Definition (FDP) file corresponding to the 3D model file specified by the pose_fitting_model parameter. The file name may contain a path, and it must be relative to the location of the configuration file. For more details, please refer to the section on the 3D Model. **NOTE:** HTML5 version does not support relative paths. Provide only name of model file (e.g. jk_300.fdp).
pose_fitting_au_use [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Indicates which Action Units from the 3D model file specified by the pose_fitting_model parameter are actually active in tracking; the ones set to 1 are active and the ones set to 0 are not used. The comment line after the numbers is included for easier identification of Action Units.
pose_fitting_su_use [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Indicates which Shape Units from the 3D model file specified by the pose_fitting_model parameter are actually active in tracking; the ones set to 1 are active and the ones set to 0 are not used.
pose_fitting_pose_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for rotation (3 values) and translation (3 values) for the 3D model file specified by the pose_fitting_model parameter. A higher value results in faster reaction of the tracker but also more sensitivity to noise. The comment line after the numbers is included for easier identification of the pose parameters.

pose_fitting_au_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for Action Units (one for each AU) for the 3D model file specified by the pose_fitting_model parameter. A higher value results in faster reaction of the tracker but also more sensitivity to noise. The comment line after the numbers is included for easier identification of Action Units. For further details please refer to the section on Action Units and to the section on the 3D Model.
pose_fitting_su_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for Shape Units for the 3D model file specified by the pose_fitting_model parameter. A higher value results in faster reaction of the tracker but also more sensitivity to noise.
au_fitting_model [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	File name of the 3D model used to estimate the Action Units (returned in FaceData::faceTranslation and FaceData::faceRotation); for more details on Action Units, their customization, and the 3D models in general, please refer to the section on the 3D Model. If Action Units are not required by an application, it is recommended to disable this function by setting this parameter to "none" or simply removing it from the configuration; this will yield a small gain in data size, memory footprint and performance. **NOTE:** HTML5 version does not support relative paths. Provide only name of model file (e.g. candide3.wfm).
au_fitting_fdp [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Name of the MPEG-4 feature Points Definition (FDP) file corresponding to the 3D model file specified by the au_fitting_model parameter. The file name may contain a path, and it must be relative to the location of the configuration file. For more details, please refer to the section on the 3D Model. **NOTE:** HTML5 version does not support relative paths. Provide only name of model file. (e.g. candide3.fdp).
au_fitting_au_use [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Indicates which Action Units from the 3D model file specified by the au_fitting_model parameter are actually active in tracking; the ones set to 1 are active and the ones set to 0 are not used. The comment line after the numbers is included for easier identification of Action Units. For further details please refer to the section on Action Units and to the section on the 3D Model.
au_fitting_su_use [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Indicates which Shape Units from the 3D model file specified by the au_fitting_model parameter are actually active in tracking; the ones set to 1 are active and the ones set to 0 are not used.
au_fitting_au_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for Action Units (one for each AU) for the 3D model file specified by the au_fitting_model parameter. A higher value results in faster reaction of the tracker but also more sensitivity to noise. The comment line after the numbers is included for easier identification of Action Units. For further details please refer to the section on Action Units and to the section on the 3D Model.
au_fitting_su_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for Shape Units for the 3D model file specified by the au_fitting_model parameter . A higher value results in faster reaction of the tracker but also more sensitivity to noise.
au_names [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Contains list of action units names. Exclusive to the au_fitting_model.
mesh_fitting_model [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	File name of the 3D model used to fit a fine 3D mesh to the face (returned in FaceData::faceTranslation and FaceData::faceRotation); for more details on 3D models and their customization, please refer to the section on the 3D Model. If an application does not require the fine 3D facial mesh, it is recommended to disable this function by setting this parameter to "none" or simply removing it from the configuration; this will yield a small gain in data size, memory footprint and performance. **NOTE:** HTML5 version does not support relative paths. Provide only name of model file. (e.g. candide3.wfm).
mesh_fitting_fdp [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Name of the MPEG-4 feature Points Definition (FDP) file corresponding to the 3D model file specified by the mesh_fitting_model parameter. The file name may contain a path, and it must be relative to the location of the configuration file. For more details, please refer to the section on the 3D Model.

	NOTE: HTML5 version does not support relative paths. Provide only name of model file. (e.g. candide3.fdp).
mesh_fitting_au_use [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Indicates which Action Units from the 3D model file specified by the mesh_fitting_model parameter are actually active in tracking; the ones set to 1 are active and the ones set to 0 are not used. The comment line after the numbers is included for easier identification of Action Units.
mesh_fitting_su_use [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Indicates which Shape Units from the 3D model file specified by the mesh_fitting_model parameter are actually active in tracking; the ones set to 1 are active and the ones set to 0 are not used.
mesh_fitting_au_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for Action Units (one for each AU) for the 3D model file specified by the mesh_fitting_model parameter. A higher value results in faster reaction of the tracker but also more sensitivity to noise. The comment line after the numbers is included for easier identification of Action Units. For further details please refer to the section on Action Units and to the section on the 3D Model.
mesh_fitting_su_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Sensitivity values for Shape Units for the 3D model file specified by the mesh_fitting_model parameter. A higher value results in faster reaction of the tracker but also more sensitivity to noise.
Parameter controlling the processing of eyes.	
process_eyes [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Bit-flag parameter that controls gaze vector calculation and pupil points refinement. If the parameter is set to 0, both functionalities will be disabled. First bit controls the gaze calculations and second bit controls the pupil point refinement, so setting the parameter to 1 enables the gaze calculations, setting it to 2 enables the pupil refinement and setting it to 3 enables both functionalities. Both functionalities are enabled by default (process_eyes 3).
Precision/performance trade-off parameters Other than the parameters listed here, there are a few more parameters that affect performance – please see section 2.2.1.	
lbf_stage_modifier [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER]	Reduces total number of stages by this value. Default value is 0. There are maximum 5 stages. For example, setting the value to 1 will skip the final stage during detection, 2 will skip 2 final stages, etc. Increasing the value of this parameter increases performance but reduces feature points precision.
lbf_nperturb [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Number of perturbations for each feature points detection. Default value is 4. Reducing the number of perturbation increases performance but reduces feature points precision.
lbf_nperturb_threads [TRACKER]	Number of threads used for perturbations. Default value is 1. If lbf_nperturb parameter is set higher than 1, on multi-core processors, increasing the number of threads can increase performance, since each thread (each perturbation) may execute on a separate core.
Limits (min, max) on tracker outputs. When any of the results goes out of the specified range, full or partial re-initialisation is initiated.	
rotation_limit [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Limit values for the rotations around the x, y and z axis.
translation_limit [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Limit values for the translations in x, y, and z directions.
action_unit_limit [WIN, IOS, AND, MAC, HTML5, LIN] [TRACKER, DETECTOR]	Limit values for action units. Please refer to the section on Action Units for further details regarding Action Units.

2.2. General configuration and setup guidelines

These general guidelines may help to obtain optimal tracking results:

- Determine *camera_focus* parameter (see section 2.2.2).
- The room and the face should be well lit. User can experiment with different types of lighting (indirect daylight is usually the best, neon lights the worst).
- User should disable automatic adjustment of the camera settings by the driver like gain, exposure, white balance and similar and set them manually, if possible, depending on the camera used and lighting conditions.

2.2.1. Optimizing tracking performance

This section summarizes the configuration parameters that most affect the tracking performance.

Table 3. Parameters effect on performance

PARAMETERS	EFFECT ON PERFORMANCE
lbf_stage_modifier	Increasing this parameter improves performance but may reduce accuracy.
lbf_nperturb	Increasing this parameter decreases performance but also reduces tracking noise.
process_eyes	Disabling this increases performance, but reduces pupil points detection accuracy and disables gaze vector calculation.
au_model	Disabling this increases performance.
mesh_model	Disabling this increases performance.

A detailed explanation of the parameters can be found in the section 2.1.

Other than these parameters, the resolution of input image also affects performance.

2.2.2. Estimating the camera focus

The *camera_focus* parameter can be estimated by using the tool in the following way:

1. Print the provided chessboard pattern (chessboard.png) on a sheet of paper.
2. Fix the sheet of paper with chessboard pattern from the previous step on a flat surface.
3. Take 10 to 20 images of the chessboard pattern from different angles and distances with the camera that is to be calibrated taking care that the whole chessboard pattern is visible without minding the background.
4. Run CameraCalibration tool and select all the images taken in the previous step.
5. After calibration is done the tool will output camera focal length which can be input as camera focus parameter in tracker configuration file.

2.2.3. Configuration and data files

Other than the configuration files (.cfg), the tracker requires several other data files some of them also user-customizable, these files are defined in the configuration file.

The following example shows one possible file structure for a tracking application on Windows and relevant path settings in config file.

File structure:

(...)\TrackerApp\Resources\Facial Features Tracker – High.cfg
(...)\TrackerApp\Resources\Facial Features Tracker - Low.cfg
(...)\TrackerApp\Resources\candide3.wfm

```
(...)\TrackerApp\Resources\candide3.fdp
(...)\TrackerApp\Resources\jk_300.wfm
(...)\TrackerApp\Resources\jk_300.fdp
(...)\TrackerApp\Resources\bdtsdata\FF\ff.dat
(...)\TrackerApp\Resources\bdtsdata\LBF\vfadata\
(...)\TrackerApp\Resources\bdtsdata\NN\
```

Config file settings:

```
...
au_fitting_model candide3.wfm
au_fitting_fdp candide3.fdp
bdts_data_path bdtsdata
...
```

Tracker initialized with:

```
// assumes that the current working folder is (...)\TrackerApp
tracker = new VisageSDK::VisageTracker("Facial Features Tracker - High .cfg");
```

Similar folder structures are possible on other operating systems.

2.3. The 3D models used in tracking

As explained in section 2.1., tracker and detector can use up to three different 3D model files for estimating 3D information by fitting the 3D face model to detected/tracked 2D feature points in the image. The 3D models are written in a simple, documented text file format so they can be fully configured or custom models can be used for any specific requirements.

This section briefly describes the default models shipped with visage|SDK, and specifies the file formats used to enable customization.

2.3.1. The Candide model

This model was previously used to evaluate action units and shape units (au_fitting_model in section 2.1.) and estimate face rotation and translation. This model is no longer used but is kept for legacy purposes. Instead, a more detailed/accurate model is used – see section 2.3.2 The jk_300 model.

The model is defined in the file candide3.wfm, consists of 157 vertices forming 228 faces. An alternative model, candide3-ClosedMouth.wfm is available for special purposes, when closed mouth is required.

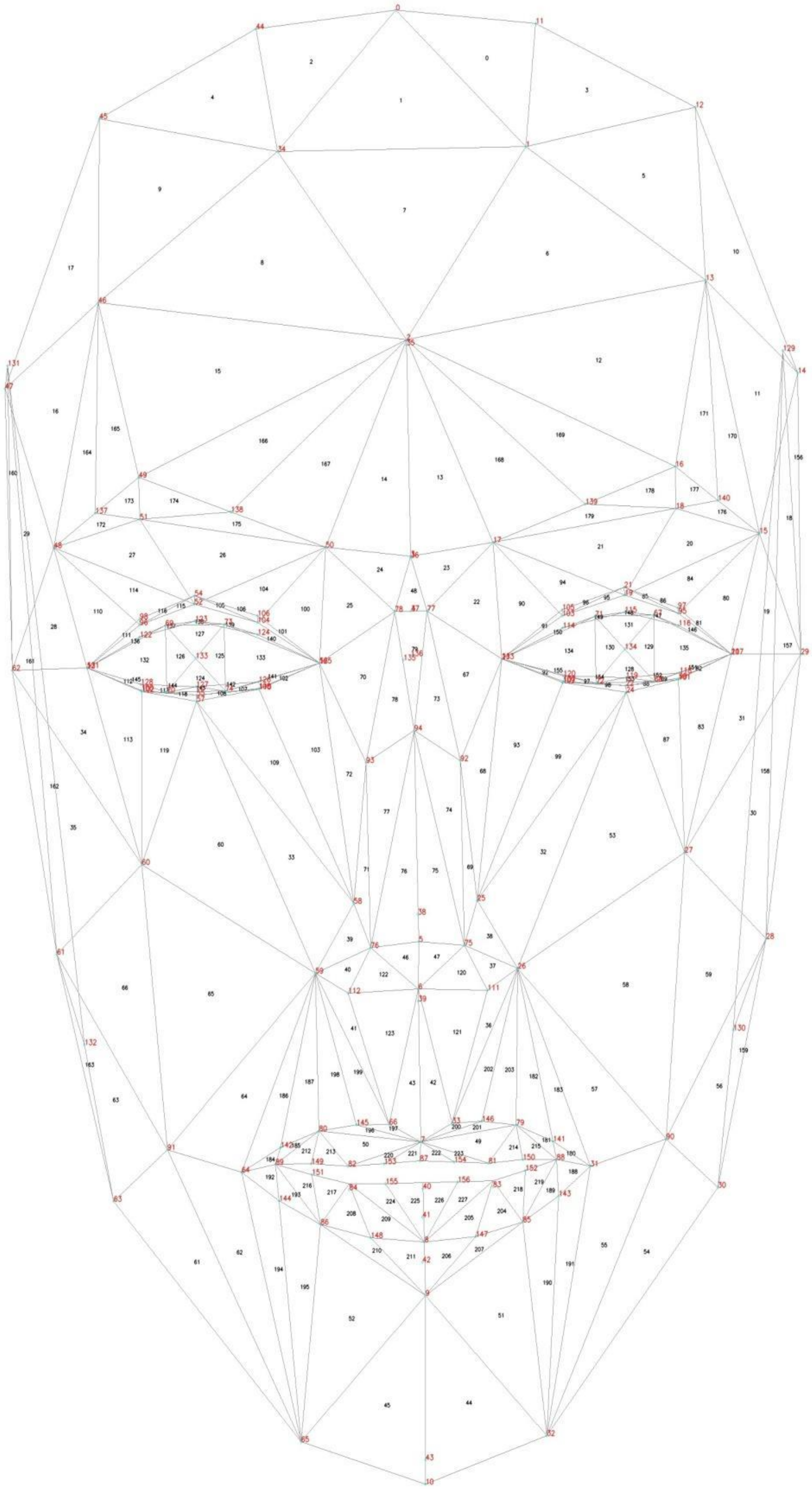


Figure 1. Candide model

2.3.2. The jk_300 model

This model is currently used to estimate pose, evaluate actions units and shape units (au_fitting_model in section 2.1.) and provide fine mesh of the face. The model consists of 357 vertices and 640 triangles.

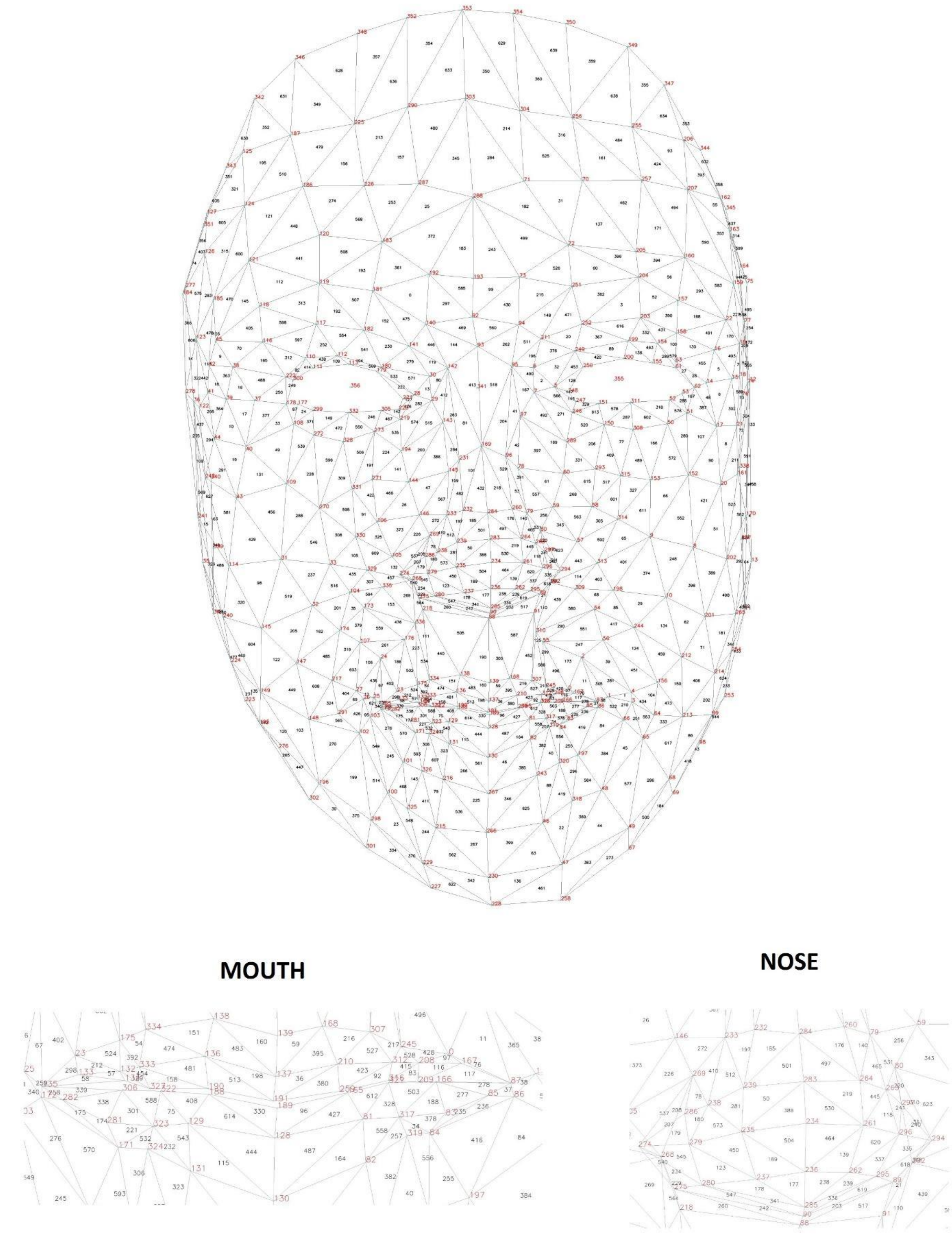


Figure 2. jk_300 model

2.3.3. File formats for 3D models

It is possible to modify this file or to configure the tracker to use a different 3D model file. The 3D model has several Action Units defined for animating the model, and a number of Shape Units for deforming the initial model shape.

The 3D models are written in plain text wfm file format, specified as follows (lines beginning with # are comments):

```
# VERTEX LIST:
[vertex count]
[x y z] (vertex coordinates)
…
[x y z] (vertex coordinates)

# TEXCOORD LIST:
[texcoord count]
[u v] (normalized texture coordinates)
…
[u v] (normalized texture coordinates)

# FACE LIST:
[face count]
[i1 i2 i3] (vertex indices making a face)
…
[i1 i2 i3] (vertex indices making a face)

# ANIMATION UNITS LIST:
[action units count]

# action unit description
[number of affected vertices]
[vertex_index x_offset y_offset z_offset]
…
[vertex_index x_offset y_offset z_offset]

…

# action unit description
[number of affected vertices]
[vertex_index x_offset y_offset z_offset]
…
[vertex_index x_offset y_offset z_offset]

# SHAPE UNITS LIST:
[shape units count]

# shape unit description
[number of affected vertices]
[vertex_index x_offset y_offset z_offset]
…
[vertex_index x_offset y_offset z_offset]

…

# shape unit description
[number of affected vertices]
[vertex_index x_offset y_offset z_offset]
…
[vertex_index x_offset y_offset z_offset]

# END OF FILE
```

Related to the 3D model file is the FDP file. This simple file contains the correspondences between the standard MPEG-4 Facial Feature Points with some non-standard extensions and the vertices of the face model. For details regarding the MPEG-4 Feature Points, including a schematic view of all feature point numbers, see the MPEG-4 Face and Body Animation Introduction document, available in visage|SDK package.

The FDP file format consists of one line of text for each feature point, in the following format:

<group>.<index><x><y><z><mesh_index>.<vertex_index>.

The information used by the tracker is the MPEG-4 group and index, and the corresponding vertex index - the index of the feature point's vertex in the 3D model.

2.4. Action Units

The action units returned by the tracker, and referred to in the configuration parameters documentation, are defined in the 3D face model file (see previous section). Action Units can be modified by the user by editing or replacing the 3D face model file specified by the au_fitting_model configuration parameter.

Furthermore, the tracker configuration file defines the names for action units (see au_names parameter). These names are returned as tracking results together with action unit values - see documentation of VisageSDK::FaceData structure for further details. The actual actions units used in the standard configurations are shown in Table 4.

Possible use of action units include facial animation, or facial analysis; for example, it would be possible to define FACS action units in order to obtain automatic FACS scoring.

Table 4. Actions units used by standard configurations

Action Units	
AU1: Nose wrinkler AU2: Jaw z-push AU3: Jaw x-push AU4: Jaw drop AU5: Lower lip drop AU6: Upper lip raiser (AU10) AU7: Lip stretcher left (AU20) AU8: Lip corner depressor (AU13/15) AU9: Lip presser (AU23/24) AU10: Left outer brow raiser AU11: Left inner brows raiser AU12: Left brow lowerer	AU13: Left eye closed (AU42/43/44/45) AU14: Lid tightener (AU7) (NOT ACTIVE) AU15: Upper lid raiser (AU5) (NOT ACTIVE) AU16: Rotate eyes left (NOT ACTIVE) AU17: Rotate eyes down (NOT ACTIVE) AU18: Lower lip x-push AU19: Lip stretcher right AU20: Right outer brow raiser AU21: Right inner brow raiser AU22: Right brow lowerer AU23: Right eye closed