VisageTracker Configuration Manual

version 8.0

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 customising the tracker for a particular application.

The configuration file is loaded every time the new tracking session is started by calling trackFromVideo(), trackFromCam() and trackFromRawImages(). It is possible to change the configuration file between tracking sessions using setTrackerConfigurationFile().

Furthermore, the configuration files in the same format and used for facial fetures detection though in this case only a subset of configuration parameters in used.

1.1. Standard configuration files

visage|SDK comes with several standard configuration files aimed at common usage scenarios such as head tracking, facial features tracking or off-line configurations allowing minor manual interventions to gain better accuracy. The set of configuration files is different in each version of the SDK due to performance issues on different platforms. Table 1. provides an overview of all available configurations.

Table 1. Standard configuration files

Configuration file name	Available in *	Overview
Head Tracker.cfg	WIN, IOS, AND, MAC, HTML5, LIN, FLASH	Fully automatic tracker optimised for high performance head pose tracking from camera or video files (eyebrow/mouth movements are tracked just to support head pose so they are not precise and should not be used)
Facial Features Tracker - High.cfg	WIN, IOS, AND, MAC, HTML5, LIN, FLASH	Fully automatic facial features tracker optimised for real time operation from camera or video files on high performance mobile devices such as . iPhone5. Tracks head pose, mouth, eyebrows and eye motion
Facial Features Tracker - Low.cfg	WIN, IOS, AND, MAC, HTML5, LIN, FLASH	Fully automatic facial features tracker optimised 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

* "WIN" for Windows, "AND" for Android, "MAC" for MAC OS X and “LIN” for Linux and RedHat

2. Customizing the tracker

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

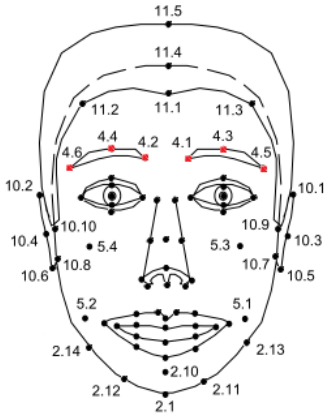
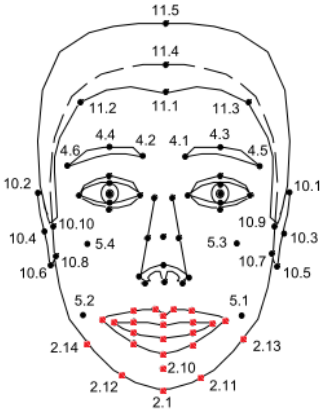
2.1. Configuration parameters

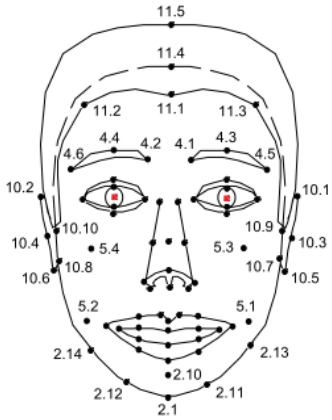
The following table provides the detailed description of parameters defined in the configuration file and their usage. Some parameters are available only on specific platform marked in table as "WIN" for Windows, "IOS" for iOS , "AND" for Android, "MAC" for MAC OS X, "LIN" for Linux and "HTML5" for HTML5.

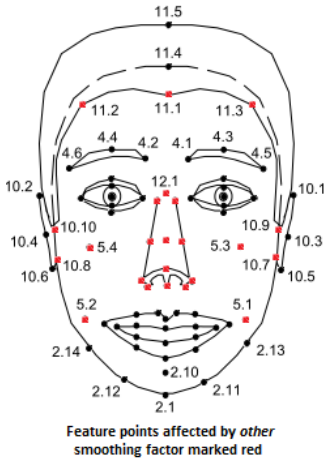
Table 2. Configuration parameters

Parameter name	Description
display_width [WIN, MAC, LIN]	Width of the display window. Input video image is resized to this size for display purposes. Affects only the display during tracking but not tracking results, though it may affect performance, which has an indirect effect on results. Height is calculated from the size of the input video, preserving the aspect ratio.
camera_input [WIN, MAC, LIN]	Camera input system. Default is 0. Alternative setting is 1, to be used if the camera does not function properly with the default setting.
camera_device [WIN, IOS, AND, MAC, LIN]	Camera device number. Used for selecting a camera when multiple cameras are available (i.e. front or back camera where available). The mapping of physical cameras to camera device numbers is hardware-dependent and should be determined experimentally for a specific computer or device.
camera_width [WIN, IOS, AND, MAC, LIN]	Requested camera horizontal resolution. The tracker will try to initialise the camera to work in this resolution; if it fails, the default camera resolution will be used.
camera_height [WIN, IOS, AND, MAC, LIN]	Requested camera vertical resolution. Used in the same way as the horizontal resolution.
camera_frame_rate [WIN, IOS, AND, MAC, LIN]	Requested camera frame rate. The tracker will try to initialise the camera to work with this frame rate; if it fails, the default camera frame rate will be used.
camera_mirror [WIN, IOS, AND, MAC, LIN]	If set to 1, camera image is flipped horizontally so that the user has the impression like in front of a mirror. Image is flipped before entering the tracker, so tracking is performed on the flipped image. This means that tracking results will also be flipped.
camera_settings_dialog [WIN]	Enable or disable camera settings dialog box. If set to 1, the settings dialog box will open at the beginning of tracking.
camera_auto_settings [WIN]	Enable or disable automatic camera settings (1 = enable, 0 = disable). In the current release this feature is experimental and may not work well with all cameras, so it is disabled by default.
video_file_sync [WIN, IOS, AND, MAC, LIN]	Synchronisation of video playback from file. If set to 0, all video frames are processed and displayed so the effective video playback speed depends on the available processing power - on a slow computer playback will be slower than real time, while on a fast computer it may be faster. If the flag is set to 1 playback is synchronised by skipping video frames or introducing delay as needed, so the video file is played at its normal speed. This may deteriorate tracking results on slower computers because video frames are skipped.
init_yaw_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	This value is used during initialisation. It controls the amount of yaw (left-right rotation of the head) allowed at initialisation; the tracker waits until the head pose is within this limit before it initialises and starts tracking. It is expressed in meters, as the deviation of the nose tip position from the imaginary line drawn between the eyes perpendicular to left eye - right eye connecting line. The value of 0 means

Parameter name	Description
	the tracker will require perfectly frontal head pose before it starts (it is not recommended to set it to 0 because the tracker may then never start); higher values relax the requirements.
init_roll_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	This value is used during initialisation. It controls the amount of roll (tilt of the head) allowed at initialisation; the tracker waits until the head pose is within this limit before it initialises and starts tracking. It is expressed in degrees. The value of 0 means the tracker will require perfectly frontal head pose before it starts (it is not recommended to set it to 0 because the tracker may then never start); higher values relax the requirements.
init_velocity_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	This value is used during initialisation. It controls the speed of head movement allowed at initialisation; the tracker waits until the head stabilises below this speed limit before it initialises and starts tracking. It is expressed in meters per second. The value of 0 means the tracker will require perfectly still head before it starts (it is not recommended to set it to 0 because the tracker may then never start); higher values relax the requirements.
init_timeout [WIN, IOS, AND, MAC, HTML5, LIN]	This value is used during initialisation. It controls the time allowed at initialisation, in milliseconds. If the desired head pose was not found during this time, the tracker chooses the best available image seen during this time. The timing starts from the moment when face is detected.
init_timeout_enable [WIN, IOS, AND, MAC, HTML5, LIN]	This value is used during initialisation. It enables or disables the initialization timeout mechanism; when it is disabled, the parameter init_timeout (see above) is ignored and initialization continues until the desired head pose is reached. The setting is separate for camera, video file and raw image input modes and determined by the first, second and third bit of the value, respectively. Thus value 1 means that the timeout mechanism is enabled when tracking from camera; 2 means it is enabled when tracking from video file; 4 means it is enabled when using the raw image interface and 0 means it is always disabled; combinations are allowed, e.g. 6 enables timeout in video and raw image input modes.
init_display_status [WIN]	This value is used during initialisation. It enables or disables the initialization status display. When enabled, the initialization status is displayed interactively on the screen during initialization in order to help the user to position the head. The setting is separate for camera, video file and raw image input modes and determined by the first, second and third bit of the value, respectively. Thus value 1 means that the display is enabled when tracking from camera; 2 means it is enabled when tracking from video file; 4 means it is enabled when using the raw image interface and 0 means it is always disabled; combinations are allowed, e.g. 6 enables display in video and raw image input modes.
recovery_timeout [WIN, IOS, AND, MAC, HTML5, LIN]	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]	The tracker can apply a smoothing filter to the tracking results to reduce the inevitable tracking noise. An adaptive smoothing filter 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. The value 0 provides maximal smoothing and lowest response (longest delay). Higher values provide less smoothing but higher response. The value of -1 disables smoothing completely for specific group. Smoothing factors are set separately for the following groups of tracking results, one factor value for each group: **Translation**: Applies smoothing to face translation values.

Parameter name	Description
	The following members of TrackingData are affected by this factor: TrackingData::faceTranslation TrackingData::faceAnimationParameters - only global translation of the face that is stored in the global translation Body Animation Parameters (BAPs) **Rotation:** Applies smoothing to face rotation values. The following members of TrackingData are affected by this factor: TrackingData::faceRotation TrackingData::faceAnimationParameters - only head rotation stored in FAP 7 group as specified by MPEG-4 specification **Action units:** Applies smoothing to facial action units. The following members of TrackingData are affected by this factor: TrackingData::actionUnits **Eyebrows:** Applies smoothing to parameters that represent eyebrow movement. The following members of TrackingData are affected by this factor: TrackingData::featurePoints2D - group 4, feature points 1 to 6 TrackingData::featurePoints3D - group 4, feature points 1 to 6 TrackingData::featurePoints3DRelative - group 4, feature points 1 to 6 TrackingData::faceAnimationParameters - group 4  Feature points affected by *eyebrows* smoothing factor marked red **Mouth and chin:** Applies smoothing to parameters that represent mouth and chin movement. The following members of TrackingData are affected by this factor: TrackingData::featurePoints2D - group 2, feature points 1 to 14; group 8, feature points 1 to 10 TrackingData::featurePoints3D - group 2, feature points 1 to 14; group 8, feature points 1 to 10 TrackingData::featurePoints3DRelative - group 2, feature points 1 to 14; group 8, feature points 1 to 10  Feature points affected by *mouth and chin* smoothing factor marked red

Parameter name	Description
	TrackingData::faceAnimationParameters - groups 2 and 8 **Gaze:** Applies smoothing to parameters that represent gaze direction. The following members of TrackingData are affected by this factor: TrackingData::gazeDirection TrackingData::gazeDirectionGlobal TrackingData::featurePoints2D - group 3, feature points 5 and 6 TrackingData::featurePoints3D - group 3, feature points 5 and 6 TrackingData::featurePoints3DRelative - group 3, feature points 5 and 6 TrackingData::faceAnimationParameters - group 3, eyeballs only  Feature points affected by *gaze* smoothing factor marked red **Eye closure:** Applies smoothing to parameters that represent eye closure. The following members of TrackingData are affected by this factor: TrackingData::featurePoints2D - group 3, all feature points except 5 and 6; group 12, feature points 5 to 12 TrackingData::featurePoints3D - group 3, all feature points except 5 and 6; group 12, feature points 5 to 12 TrackingData::featurePoints3DRelative - group 3, all feature points except 5 and 6; group 12, feature points 5 to 12 TrackingData::faceAnimationParameters - group 3, excluding eyeballs  Feature points affected by *eye closure* smoothing factor marked red **Other:** Applies smoothing to parameters that represent other data. The following members of TrackingData are affected by this factor:

Parameter name	Description
	TrackingData::featurePoints2D - group 5, all feature points; group 9, feature points 1 to 7 and 12 to 15; group 10, feature points 7 to 10; group 11, feature points 1 to 3; group 12, feature point 1 TrackingData::featurePoints3D - group 5, all feature points; group 9, feature points 1 to 7 and 12 to 15; group 10, feature points 7 to 10; group 11, feature points 1 to 3; group 12, feature point 1 TrackingData::featurePoints3DRelative - group 5, all feature points; group 7, feature point 1; group 9, feature points 1 to 7 and 12 to 15; group 10, feature points 7 to 10; group 11, feature points 1 to 3; group 12, feature point 1 TrackingData::faceAnimationParameters - groups 5, 6, 9, 10  Feature points affected by *other* smoothing factor marked red
Data parameters and paths	
model_filename [WIN, IOS, AND, MAC, HTML5, LIN]	Name of the 3D model file used in the tracking process. 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. (i.e candide3.wfm).
fdp_filename [WIN, IOS, AND, MAC, HTML5, LIN]	Name of the MPEG-4 feature Points Definition (FDP) file corresponding to the 3D model file used in the tracking process. 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. (i.e candide3.fdp).
bdts_data_path [WIN, IOS, AND, MAC, LIN]	Path to the folder containing the detector data files (*.bdf) It is be **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]	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 Extended Kalman Filter (EKF)	
au_use [WIN, IOS, AND, MAC, HTML5, LIN]	Indicates which Action Units from the 3D model file 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_names [WIN, IOS, AND, MAC, HTML5, LIN]	Contains list of action units names.
ekf_sensitivity [WIN, IOS, AND, MAC, HTML5, LIN]	Sensitivity values for rotation (3 values), translation (3 values) and Action Units (one for each AU). 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

Parameter name	Description
	on Action Units and to the section on the 3D Model.
Parameter controlling the processing of eyes.	
process_eyes [WIN, IOS, AND, MAC, HTML5, LIN]	Switches the processing of eye rotation and closure on or off, as follows: 0 - eye rotation off, 1 - eye rotation and closure on **Note:** if process_eyes is set to 1, AUs controlling eye rotation and closure (AU13, AU 23, AU16 and AU17 in asymmetric configuration and AU13, AU16 and AU17 in symmetric configuration) must also be enabled by setting them to 1 in the au_use parameter.
leye_closing_au [WIN, IOS, AND, MAC, HTML5, LIN]	Index of action unit controlling eye closure of left eye. If set closure of left eye is controlled by this action unit. If this is set then reye_closing_au must also be set and eye_closing_au is ignored.
reye_closing_au [WIN, IOS, AND, MAC, HTML5, LIN]	Index of action unit controlling eye closure of right eye. If set closure of right eye is controlled by this action unit. If this is set then leye_closing_au must also be set and eye_closing_au is ignored.
eye_h_rotation_au [WIN, IOS, AND, MAC, HTML5, LIN]	Index of action unit controlling horizontal rotation of both eyes.
eye_v_rotation au [WIN, IOS, AND, MAC, HTML5, LIN]	Index of action unit controlling vertical rotation of both eyes.
eye_points_coords [WIN, IOS, AND, MAC, HTML5, LIN]	Array of vertices corresponding to pupil of left and right eye consisting of two elements, one for each eye. Elements of the array consist of 4 values where first value is index of triangle which contains pupil vertex while following three values are barycentric coordinates of vertex in selected triangle. First element of the array corresponds to fdp point 3.5 while second element corresponds to fdp point 3.6.
Precision/performance trade-off parameters	
lbf_stage_modifier [WIN, IOS, AND, MAC, HTML5, LIN]	Reduces total number of stages by this value. Default value is 0. There is 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 tracking precision.
lbf_nperturb [WIN, IOS, AND, MAC, HTML5, LIN]	Number of perturbations for each feature points detection. Default value is 6. Reducing the number of perturbation increases performance but reduces tracking precision.
tracking_preproc [WIN, IOS, AND, MAC, HTML5, LIN]	Flag that indicates whether additional preprocessing of input image will be used for tracking algorithms. If set to 1, it increases accuracy, but decreases performance.
high_prec_addition [WIN, IOS, AND, MAC, HTML5, LIN]	Flag that indicates whether high precision addition will be used in internal implementation. If set to 1, it increases accuracy, but decreases performance.
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]	Limit values for the rotations around the x, y and z axis.
translation_limit [WIN, IOS, AND, MAC, HTML5, LIN]	Limit values for the translations in x, y, and z directions.
action_unit_limit [WIN, IOS, AND, MAC, HTML5, LIN]	Limit values for action units. Please refer to the section on Action Units for further details regarding Action Units.
Parameters controlling the strip detection	

Parameter name	Description
The tracker can detect credit card magnetic strips in the input image to provide measurement estimations. These parameters control the detection thresholds and search bounds.	
detect_strip_area_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	Minimal area the strip can occupy. Used for filtering small contours that appear from the noise in the image.
detect_strip_angle_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	Maximum angle deviation from 90° measured as the maximum angle cosine that appears in the contour.
detect_strip_ratio_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	Maximum error of the strip width/height ratio measured in percentage of the perfect ratio. See perfect ratio parameter below.
detect_strip_perfect_ratio [WIN, IOS, AND, MAC, HTML5, LIN]	The strips real width/height ratio.
detect_strip_roi_y_offset [WIN, IOS, AND, MAC, HTML5, LIN]	Search region Y offset measured from the center point between the eyes as the percentage of the eye distance.
detect_strip_roi_width [WIN, IOS, AND, MAC, HTML5, LIN]	Search region width as the percentage of the eye distance.
detect_strip_roi_height [WIN, IOS, AND, MAC, HTML5, LIN]	Search region height as the percentage of the eye distance.

2.2. General configuration and setup guidelines

These general guidelines may help to obtain optimal tracking results:

- Set *camera_width* and *camera_height* to those supported by the actual camera.
- Determine *camera_focus* parameter (see Estimating the camera focus for more information).
- 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 tracker accuracy vs. performance

This section summarizes which configuration parameters most effect accuracy vs. performance tradeoff, their general effect, and a general guideline for reaching the desired result on a given platform.

A general rule to increase the accuracy of the tracker is to increase the work image resolution. However, if the tracker FPS is too low (< 15 FPS) accuracy will also drop so a balance needs to be found. This depends on the machine configuration and the use case scenario.

The parameters can be categorized in the following way:

1. Parameters that effect work image resolution
    a. *camera_width*[1]
    b. *camera_height*

2. Parameters that effect quality of points tracked/detected by localized point detection
    a. *lbf_stage_modifier*
    b. lbf_nperturb
    c. *process_eyes*

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

2.2.2. Estimating the camera focus

1**camera_width* and *camera_height* are only relevant when tracking from camera

The *camera_focus* parameter can be roughly estimated in the following way:

1. Start the application and tracking from camera.
2. Take an object of known length (e.g. rope, stick or ruler) and place it perpendicular to the camera so that its length fills the smaller dimension of the camera image (e.g. height for landscape image).
3. Measure the distance from the object to the camera.
4. Calculate the camera focus value by dividing the measured distance with the length of the object and multiplying it with 2.

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\

(...)\TrackerApp\Resources\Facial Features Tracker – High.cfg

(...)\TrackerApp\Resources\Facial Features Tracker - Low.cfg

(...)\TrackerApp\Resources\candide3.wfm

(...)\TrackerApp\Resources\candide3.fdp

(...)\TrackerApp\Resources\bdtsdata\LBF\fa.lbf

(...)\TrackerApp\Resources\bdtsdata\LBF\fc.lbf

(...)\TrackerApp\Resources\bdtsdata\LBF\vfadata\

(...)\TrackerApp\Resources\bdtsdata\LBF\ye\

Config file:

...

model_filename candide3.wfm

fdp_filename candide3.fdp

bdts_data_path bdtsdata

...

Tracker initialized with:

tracker = new VisageSDK::VisageTracker2("Facial Features Tracker - High .cfg"); //This assumes that current working folder is (...)\TrackerApp

Similar folder structures are possible on other operating systems.

2.3. The 3D model used in tracking

The crucial part of the tracking process is the 3D model that is fitted to the face in each frame of the video. The current model is defined in the file candide3.wfm, consists of 133 vertices forming 190 faces. An alternative model, candide3-ClosedMouth.wfm is available for special purposes, when closed mouth is required.

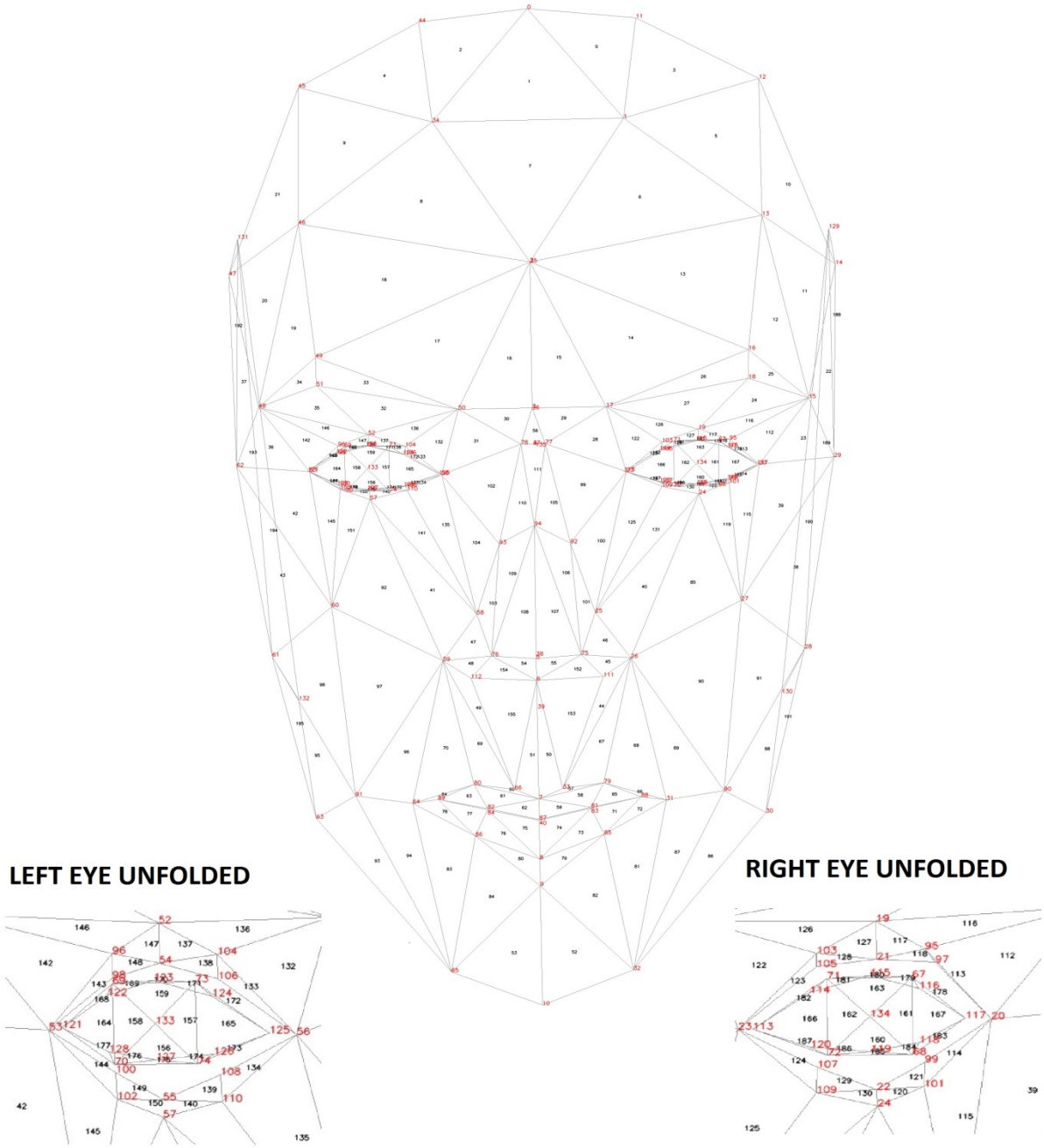


Figure 1. Candide model

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

The model is written in a simple text format including comments, so it should be easy to understand if it is desired to change it, for example in order to use a more detailed model, or modify the action units.

Related to the 3D model file is the FDP file. This simple file contains the correspondences between the standard MPEG-4 Facial Feature Points 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.

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.

During tracking, the 3D model can be obtained through the getTrackingData() function and the TrackingData structure.

The actual vertex and triangle numbers of the default 3D face model (candide3.wfm) are illustrated in the figure 1.

2.4. Action Units

The action units used by the tracker, and referred to in the configuration parameters documentation, are defined in the 3D face model file (see previous section). Their syntax is simple: for each AU there is a list of vertices it moves and the displacement for each vertex. Action Units can be modified by the user by editing the 3D face model file.

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 TrackingData structure for further details. The actual actions units used in the standard configurations are shown in table 3.

Table 3 Actions units used by standard configurations

Configurations	3D model file	Action Units
Face Detector.cfg Facial Features Tracker – High.cfg Facial Features Tracker – Low.cfg Head Tracker.cfg	Candide3.wfm	AU1: Nose wrinkler AU2: Jaw z-push (NOT ACTIVE) AU3: Jaw x-push AU4: Jaw drop AU5: Lower lip drop AU6: Upper lip drop (AU10) AU7: Lip stretcher left (AU20) AU8: Lip corner depressor (AU13/15) AU9:Lip presser (AU23/24) (NOT ACTIVE) AU10: Left outer brow rasier 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 rasier (AU5) (NOT ACTIVE) AU16: Rotate eyes left AU17: Rotate eyes down 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