Visage Technologies
FACE TRACKING & ANIMATION

VisageTracker Configuration Manual

version 5.0

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, MAC, LIN	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 - Asymmetric.cfg	WIN, MAC, LIN	Fully automatic facial features tracker optimised for real time operation from camera or video files. Tracks head pose, mouth , eyebrows and eye motion. This configuration has expanded set of action units to support tracking of asymmetric face movement
Facial Features Tracker.cfg	WIN, IOS, AND, MAC; LIN	Fully automatic facial features tracker optimised for real time operation from camera or video files. Tracks head pose, mouth, eyebrows and eye motion
Off-line Facial Features Tracker (semi - automatic setup).cfg	WIN, MAC, LIN	A facial features tracker optimised for getting improved tracking results from offline tracking in video files. Requires manual adjustment during initialisation (only once for each video file or person) Tracks head pose, mouth, eyebrows and eye motion.
Facial Features Tracker - Manual Face Detection.cfg	WIN, MAC, LIN	A facial features tracker optimised for real time operation from camera or video files. Manual picking of eyes, nose and mouth corners required at initialisation or re - initialisation. This configuration is intended for special cases where the

		face can not be automatically detected in order to initalise the tracking. Such special cases currently include thermal videos and sometimes faces wearing markers or simply videos of very low quality. It is important that the face is resonably front-facing and expression neutral at the moment of initalisation. Tracks head pose, mouth, eyebrows and eye motion
FFT - HighPerformance.cfg	IOS, AND	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
FFT - MidPerformance.cfg	IOS, AND	Fully automatic facial features tracker optimised for real time operation from camera or video files on mid performance mobile devices such as iPhone4S. Tracks head pose, mouth, eyebrows and eye motion
FFT - LowPerformance.cfg	IOS, AND	Fully automatic facial features tracker optimised for real time operation from camera or video files on low performance mobile devices such as iPhone4. Tracks head pose, mouth, eyebrows and eye motion
HT - HighPerformance.cfg	IOS, AND	Fully automatic tracker optimised for high performance head pose tracking from camera or video files on high performance mobile devices such as iPhone5 (eyebrow/mouth movements are tracked just to support head pose so they are not precise and should not be used) .
HT - MidPerformance.cfg	IOS, AND	Fully automatic tracker optimised for high performance head pose tracking from camera or video files on mid performance mobile devices such as iPhone4S (eyebrow/mouth movements are tracked just to support head pose so they are not precise and should not be used) .
HT - LowPerformance.cfg	IOS, AND	Fully automatic tracker optimised for high performance head pose tracking from camera or video files on low performance mobile devices such as iPhone4 (eyebrow/mouth movements are tracked just to support head pose so they are not precise and should not be used) .

Facial_Features_Tracker_-_Web.cfg	HTML5	Fully automatic facial features tracker optimised for real time operation from camera or video file. Tracks head pose, mouth, eyebrows and eye motion

* "WIN" for Windows, "IOS" for iOS, "AND" for Android, "MAC" for MAC OS X and "HTML5" for HTML5

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). If only one camera is present the value should be 0.
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_flip [WIN, IOS, AND, MAC, LIN]	If set to 1, camera image is flipped vertically. 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.
automatic_initialisation	If 0, tracker uses semi-automatic initialization procedure where the user is

Parameter name	Description
[WIN]	expected to manually fine-tune the setting of the face shape mask in the initial video frame. Otherwise, tracker operation is fully automatic, with one possible mode: 1 - fast initialisation.
manual_face_detection [WIN]	When set to 1, the tracker disables the automatic detection of the face during the tracking initialization and instead it presents the image to the user and asks the user to click on the eyes, nose tip and mouth corners. This is used for videos in which automatic face detection does not work, so tracking can not start; manually picking the face allows the tracker to initialise and then it continues tracking. For example, this is the case with thermal videos and sometimes with videos where the face wears some painted markers. The function for manual feature picking is implemented through the TrackerGUIInterface abstract interface class, and specifically through the virtual function TrackerGUIInterface::selectMainFacialFeatures; it is provided with full source code and developers can modify or replace it with different picking methods, or even with their own automatic detection method. If this option is set to 0, automatic face detection is used. **Important notes:** (1) The first video frame must contain a face in reasonably front-facing pose and neutral expression, otherwise the tracker will not initialise properly. (2) The manual face detection is also called every time when tracking is lost during the video. In videos of low quality, or with lots of motion or occlusions, this may occur very often.
init_yaw_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	This value is used during automatic initialisation or, in manual initialization mode while the tracker initially searches for the best frame on which to perform initialization. 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 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 automatic initialisation or, in manual initialization mode while the tracker initially searches for the best frame on which to perform initialization. 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 automatic initialisation or, in manual initialization mode while the tracker initially searches for the best frame on which to perform initialization. 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 automatic initialisation or, in manual initialization mode while the tracker initially searches for the best frame on which to perform initialization. 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,	This value is used during automatic initialisation or, in manual initialization mode while the tracker initially searches for the best frame on which to perform initialization. It enables or disables the initialization timeout mechanism; when it is

Parameter name	**Description**
LIN]	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 automatic initialization or, in manual initialization mode while the tracker initially searches for the best frame on which to perform initialization. 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 only in automatic initialization mode. It 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. The value is ignored in manual initialization mode.
Parameters controlling the display	
display_video [WIN]	Controls the display of input video during tracking. Display is on if this value is set to 1, off if 0.
display_model_texture [WIN]	Controls the display of the textured 3D model used in tracking. Display is on if this value is set to 1, off if 0.
display_tri_mask [WIN]	Controls the display of the triangle mask used to select the points to track. Display is on if this value is set to 1, off if 0. The triangle mask is determined by the parameters feature_points_tri_use_num and feature_points_tri_use_list.
display_model_wireframe [WIN]	Controls the display of the 3D model used in tracking. Display is on if this value is set to 1, off if 0.
display_results [WIN]	Controls the visualization of tracking results including visualization of facial features, head pose, eye locations or any combination of these three options. The parameters can be set to the following values: 0 = no results display; 1 = display tracked feature points; 2 = display head pose; 4 = display eye locations. Furthermore, these values can be added together (e.g. 6 = display head pose and eye locations).
display_track_points [WIN]	Controls the display of track points. The track points are automatically chosen suitable points on the face that are actually tracked by matching an image patch around each point from one video frame to the next. The match quality is evaluated and visualized as the color of the point. Bright green means good match and green-brown colors indicate lower match quality, red being the worst. Display is on if this value is set to 1, off if 0.
Parameter controlling the smoothing filter 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,	

Parameter name	**Description**
smoothing inevitably introduces some delay so it should be used sparingly.	
smoothing_factors [WIN, IOS, AND, MAC, HTML5, LIN]	The value 0 provides maximal smoothing and lowest response. 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. 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 TrackingData::featurePoints3D - group 4 TrackingData::featurePoints3DRelative - group 4 TrackingData::faceAnimationParameters - group 4 **Mouth:** Applies smoothing to parameters that represent mouth/lips movement. The following members of TrackingData are affected by this factor: TrackingData::featurePoints2D - group 2, feature points 2 to 9 TrackingData::featurePoints3D - group 2, feature points 2 to 9 TrackingData::featurePoints3DRelative - group 2, feature points 2 to 9 TrackingData::faceAnimationParameters - groups 2 (excluding jaw and chin) 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

Parameter name	Description
	Eye closure: Applies smoothing to parameters that represent eye closure. The following members of TrackingData are affected by this factor: TrackingData::featurePoints2D - group 3, excluding feature points 5 and 6 TrackingData::featurePoints3D - group 3, excluding feature points 5 and 6 TrackingData::featurePoints3DRelative - group 3, excluding feature points 5 and 6 TrackingData::faceAnimationParameters - group 3, excluding eyeballs **Other:** Applies smoothing to parameters that represent other data. The following members of TrackingData are affected by this factor: TrackingData::featurePoints2D - group 2, feature points 1 and 10 to 14, groups 5, 6, 9, 10, 11 TrackingData::featurePoints3D - group 2, feature points 1 and 10 to 14, groups 5, 6, 9, 10, 11 TrackingData::featurePoints3DRelative - group 2, feature points 1 and 10 to 14, groups 5, 6, 7, 9, 10, 11 TrackingData::faceAnimationParameters - group 2, only jaw and chin, groups 5, 6, 9, 10
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.
tex_size [WIN, IOS, AND, MAC, HTML5, LIN]	Size of the rectangular texture image used for storing the facial texture captured while initialising the tracker. This texture is later used in tracking operation. **Note:** it is required that this value must be a power of 2.
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.

Parameter name	Description
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 on Action Units and to the section on the 3D Model.
Parameters controlling the initialisation of tracked feature points The tracked feature points (also called shortly "track points" elsewhere in this documentation) are the automatically chosen points on the face that are tracked by matching an image patch around each point from one video frame to the next.	
ROI_num [WIN, IOS, AND, MAC, HTML5, LIN]	Number of Regions Of Interest (ROI) in which track points are tracked. Each ROI is defined by a set of triangles of the face model. ROIs are numbered from 0 to (ROI_num - 1). Parameters dealing with initialising and tracking the points can be set individually for each ROI. For the first ROI (ROI 0) all parameters must be set, and these settings are considered default. For other ROIs, some of the parameters may be omitted and in such case the default value (from ROI 0) is used.
feature_points_num[i] [WIN, IOS, AND, MAC, HTML5, LIN]	Maximum number of points to be used within the ROI number i. It must be set for each ROI. The tracker may use less points if it can not find enough suitable points (suitable points are the ones with distinguishable features, allowing them to be tracked). Using more points may result in better tracking but slows down the tracker.
feature_points_tri_use_num[i] [WIN, IOS, AND, MAC, HTML5, LIN]	Number of triangles forming the mask for ROI number i. It must be set for each ROI. This is the size of the list in feature_points_tri_use_list[i].
feature_points_tri_use_list[i] [WIN, IOS, AND, MAC, HTML5, LIN]	The list of triangles forming the mask for ROI number i. It must be set for each ROI. These are the triangles within the face model in which the tracker will search for good points to track at the time of tracker initialisation. The list consists of feature_points_tri_use_num[i] integer numbers that are indices of triangles in the 3D model used in tracking. The mask can be visualised using the display_tri_mask parameter.
bound_rect_modifiers[i] [WIN, IOS, AND, MAC, HTML5, LIN]	Bounding rectangle modifiers for the mask for ROI number i, four float values (optional). If set, these values modify the mask defined by feature_points_tri_use_list[i] by modifying its bounding rectangle. The four values move the left, right, upper and lower edge of the bounding rectangle. Values are expressed as percentage of the original rectangle width or height. Positive values augment the rectangle, negative values shrink it. Shrinking the rectangle simply clips the mask. Augmenting it extends the mask to the new edges.
max_init_image_width [WIN, IOS, AND, MAC, HTML5, LIN]	Maximal width of the work image used in the initialisation stage. If the input video image is wider than this value, it is resized to this width for initialisation. It is recommended to keep this value bigger or equal to the actual video image width, i.e. use the maximal image size during initialization.
feature_points_min_distance[i] [WIN, IOS, AND, MAC, HTML5, LIN]	Minimum required distance between points for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default. Smaller value will make it easier for the tracker to find more points, but points that are too close are useless for tracking. The value is expressed as a percentage of the width of the work image used for initialisation, see the parameter max_init_image_width above.
feature_point_block_size[i]	The size of the pixel block used when choosing good points to track for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is

Parameter name	Description
[WIN, IOS, AND, MAC, HTML5, LIN]	used by default. Smaller value will make it easier for the tracker to find more points. The value is expressed as a percentage of the width of the work image used for initialisation (see the parameter max_init_image_width above). If the resulting absolute size is less than 3 pixels, it is set to 3 pixels, which is the minimal allowed value. Furthermore, if the resulting value is even, it is increased by one to make it odd because an odd number of pixels is required.
init_angle[i] [WIN, IOS, AND, MAC, HTML5, LIN]	Initialisation angle for ROI number i, in degrees. The initialisation of track points for the ROI is triggered when the horizontal head rotation angle exceeds this angle. If set to 0, initialisation is triggered immediately. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default.
feature_points_coords[i] [WIN, IOS, AND, MAC, HTML5, LIN]	The list of preset feature point grouped by four values for each feature point in a particular ROI: triangle index and three barycentric coordinates. If parameter is set, all points in a ROI must be listed. If parameter is not set, points will be determined automatically. Parameters controlling the process of matching the tracked feature points from one video frame to the next.
max_work_eye_dist [WIN, IOS, AND, MAC, HTML5, LIN]	Maximum eye distance in pixels in work image. If eye distance is larger than this value in input image, the work image is scaled accordingly. Setting it to a smaller value, e.g. 50, may be useful for improving tracking performance (speed) because tracking will be faster in smaller images. Note that the work image cannot be larger than half of work buffer resolution. See work_buffer_width and work_buffer_height parameter.
work_buffer_width [WIN, IOS, AND, MAC, HTML5, LIN]	Width of the buffer used for the work image. The input image is scaled accordingly to max_work_eye_dist parameter and copied into this buffer. Because of padding required by the algorithm the width of the work image cannot exceed half of the work buffer width. This parameter mainly affects memory requirements for e.g. if work_buffer_width and work_buffer_height are set to 1024, they will require 3MB of memory. If this parameter is set too low it can also affect tracking quality and performance because it implicitly sets maximum work image width.
work_buffer_height [WIN, IOS, AND, MAC, HTML5, LIN]	Height of the buffer used for the work image. The input image is scaled accordingly to max_work_eye_dist parameter and copied into this buffer. Because of padding required by the algorithm the height of the work image cannot exceed half of the work buffer height. This parameter mainly affects memory requirements for e.g. if work_buffer_width and work_buffer_height are set to 1024, they will require 3MB of memory. If this parameter is set too low it can also affect tracking quality and performance because it implicitly sets maximum work image height.
fast_image_resize [WIN, IOS, AND, MAC, HTML5, LIN]	If set to 1, images are resized from input to work resolution using a faster but less accurate method.
search_range[i] [WIN, IOS, AND, MAC, HTML5, LIN]	The range, in x and y directions, within which each tracked feature point is matched between one frame and the next. It sets the value for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default. The value is expressed as a percentage of the width of the work image, see the parameter max_work_image_width above. If set to 0 the search step is skipped and only the refinement step is performed, though this is not recommended.
refine_range[i] [WIN, IOS, AND, MAC, HTML5, LIN]	The range, in x and y directions, within which feature points are looked for during the refinement step. The refinement step is performed in each video frame, after the base search for the tracked feature point has been performed. This step refines the match using the texture information captured during the tracker initialisation. It sets the value for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default. The value is expressed as a percentage of the width of the work image, see the parameter

Parameter name	Description
	max_work_image_width above. If set to 0, refinement step is skipped, though this is not recommended.
patch_size[i] [WIN, IOS, AND, MAC, HTML5, LIN]	Size of the rectangular image patch to use when searching for a feature. The actual size to use, in pixels, is obtained by dividing with the current distance of the face from the camera in order to always cover roughly the same surface, but the minimal actual size is kept at 3x3 pixels. It sets the value for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default. The initial value is expressed as a percentage of the width of the work image, see the parameter max_work_image_width above.
match_method[i] [WIN, IOS, AND, MAC, HTML5, LIN]	A value, between 0 and 5, choosing between the available match methods. Available methods are the ones used by OpenCV's cvMatchTemplate() function. 0 = CV_TM_SQDIFF, 1 = CV_TM_SQDIFF_NORMED, 2 =CV_TM_CCORR, 3 = CV_TM_CCORR_NORMED, 4 = CV_TM_CCOEFF, 5 =CV_TM_CCOEFF_NORMED. It is recommended to keep the default value 5. It sets the value for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default.
bad_match_threshold[i] [WIN, IOS, AND, MAC, HTML5, LIN]	The threshold value, between 0 and 1, used to determine a bad match for a tracked feature point and use a recovery method to try finding the point again. It sets the value for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default.
global_bad_match_threshold [WIN, IOS, AND, MAC, HTML5, LIN]	The threshold value, between 0 and 1, used to determine a global bad match for all tracked feature points. If the average match quality of all track points falls below this value, the tracker will re-initialise.
visibility_check [WIN, IOS, AND, MAC, HTML5, LIN]	This parameter controls the visibility check on feature points. Visibility check affects behavior at larger head rotation angles so if head will not rotate more than approx. 40 degrees it is recommended to set it to 0. Values are: 0 = no visibility check; 1 = normals checking mode; 2 = color coding check mode; 3 =both modes combined.
sensitivity_falloff[i] [WIN, IOS, AND, MAC, HTML5, LIN]	A parameter that controls how fast the tracker sensitivity to a particular track point decreases when the match quality for that point decreases. Higher values result in sharper falloff, i.e. by increasing this value the tracker increasingly ignores points with bad match values. It sets the value for ROI number i. If this parameter is not set for a particular ROI, the value from ROI 0 is used by default.
recovery_frames [WIN, IOS, AND, MAC, HTML5, LIN]	Number of frames after re-initialising the tracker in which tracking process is modified to let the tracker recover from previous failure (matching done based on texture information only and global_bad_match_threshold is ignored).
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.
eye_closing_au [WIN, IOS, AND, MAC, HTML5, LIN]	Index of action unit controlling eye closure. If set, closure of both eyes is controlled by this action unit unless it is overridden by leye_closing_au and reye_closing_au.
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.

Parameter name	Description
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.
Parameter controlling points detection.	
bdts_points_use [WIN, IOS, AND, MAC, HTML5, LIN]	Enables or disables detection of specific feature point, as follows: 0 - disable, 1 - enable.
bdts_points_angle [WIN, IOS, AND, MAC, HTML5]	Detection activation angle for specific feature point in degrees, as follows: 0 - disables activation angle, otherwise - activation angle in degrees.
bdts_trees_factor [WIN, IOS, AND, MAC, HTML5]	Reduces number of trees used in detection by set factor for each bdts point. Reducing number of trees increases performance but reduces tracking precision.
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 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,	Maximum angle deviation from 90° measured as the maximum angle cosine that appears in the contour.

Parameter name	Description
LIN]	
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. Recommended setting is the maximum values supported by the camera, though high values can cause slowdowns and low frame rate.
- Set *display_width* and *max_init_image_width* parameters to the same value as *camera_width* parameter.
- Change *max_work_image_width* parameter to control accuracy / performance ratio.
- 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 and the number of points detected, as well as points tracked with the template matching algorithm. 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. Also, head rotation is estimated from all the points thus turning off some seemingly inconsequential points like points around the mouth, might negatively impact head rotation accuracy.

The parameters can be categorized in the following way:

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

2. Parameters that effect number of points tracked/detected either by template matching or localized point detection
    a. *bdts_points_use*

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

b. *bdts_trees_factor*
c. *process_eyes*
d. *ROI_num*
    i. *feature_points_num*
    ii. *search_range*
    iii. *patch_size*

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

Work image resolution

Increasing the value of *max_work_eye_dist* will result in the tracker performing slower but may increase the tracker accuracy. Larger values for the maximum eye distance, in most cases increase the accuracy because the face and the work image will be larger and as a consequence hold more information.

camera_width and *camera_height* parameters are only relevant in case of tracking from the camera. Smaller values will result in a smaller resolution work image. See the general guidelines in section 0.

Template matching or localized point detection

The tracker should be most accurate when all points of the localized point detection are turned on. Parameter *bdts_points_use* controls which points are active. For the list of all points see the MPEG-4 specification document, however not all points can be used in the tracker - refer to the configuration file to see which points are available.

bdts_tree_factor is also related to the localized point detection. Increasing the value will reduce number of trees used for each detected point resulting in a better performance but usually poorer quality.

Setting the *process_eyes* parameter to 0 will turn off six detections (2 pupil, upper and lower eyelid for both eyes) which will most likely result in improved performance but features like gaze tracking and eye closure will be disabled.

Number of template matching feature points is controlled by *ROI_num* and *feature_points_num* parameters. Larger values for *search_range* and *patch_size* might improve tracker accuracy in terms of more precise and faster lateral and vertical tracking but because a larger area needs to be processed, performance drops.

2.2.2. Estimating the camera focus

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. Recommendations regarding manual face detection

The tracker normally relies on automatic detection of the face and main facial features in order to initialise and start tracking. However, in certain kinds of videos the automatic detection does not work, and thus tracking never starts. For this purpose, the "*manual_face_detection*" configuration parameter allows to replace the automatic face/main features detection with a manual one, allowing the tracker to start and successfully track the face in the video. Examples of such videos include but are not limited to:

- thermal videos
- videos of people wearing special markers on the face
- videos of low quality

The tracker configuration to be used in this case is "Facial Features Tracker - Manual Face Detection.cfg".

In the current implementation (the FaceTracker2 sample project on Windows only), the image is presented to the user and the user is asked to click on the eyes, nose tip and mouth corners. The function for manual feature picking is implemented through the TrackerGUIInterface abstract interface class, and specifically through the virtual function TrackerGUIInterface::selectMainFacialFeatures; it is provided with full source code and developers can modify or replace it with different picking methods, or even with their own automatic detection method.

More precision can be obtained by modifying the "Facial Features Tracker - Manual Face Detection.cfg" configuration file and setting the "automatic_initialisation" parameter to 0. With this change, the user will also be asked to fine-tune the 3D face model over the face after initial picking of features. This fine-tuning of the model can provide improved tracking results.

If still better quality is desired, another possibility is to use the configuration "Off-line Facial Features Tracker (semi-automatic setup).cfg" and modify it by setting "manual_face_detection" to 1. The off-line configuration uses an extended set of facial Action Units and it is set to process every frame of video so it may provide better quality results. However, it should be noted that this configuration works best on good quality videos, while on low-quality videos it may actually provide inferior results to other configurations.

2.2.4. 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\Head Tracker.cfg

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

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

(...)\TrackerApp\Resources\bdtsdata\ puploc.bdf

(...)\TrackerApp\Resources\bdtsdata\ puploc-new.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp32.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp34.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp38.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp42.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp44.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp46.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp82.bdf

(...)\TrackerApp\Resources\bdtsdata\ lp84-v2.bdf

...

Config file:

...

model_filename candide3.wfm

fdp_filename candide3.fdp

detector_data_path bdtsdata

...

Tracker initialized with:

tracker = new VisageSDK::VisageTracker2(NULL, NULL, "Head Tracker.cfg");   //This assumes that current working folder is (...)\TrackerAp

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. The off-line tracking configuration uses another version of the model (candide3-exp.wfm), with extended action units (see section 2.3 Action Units).

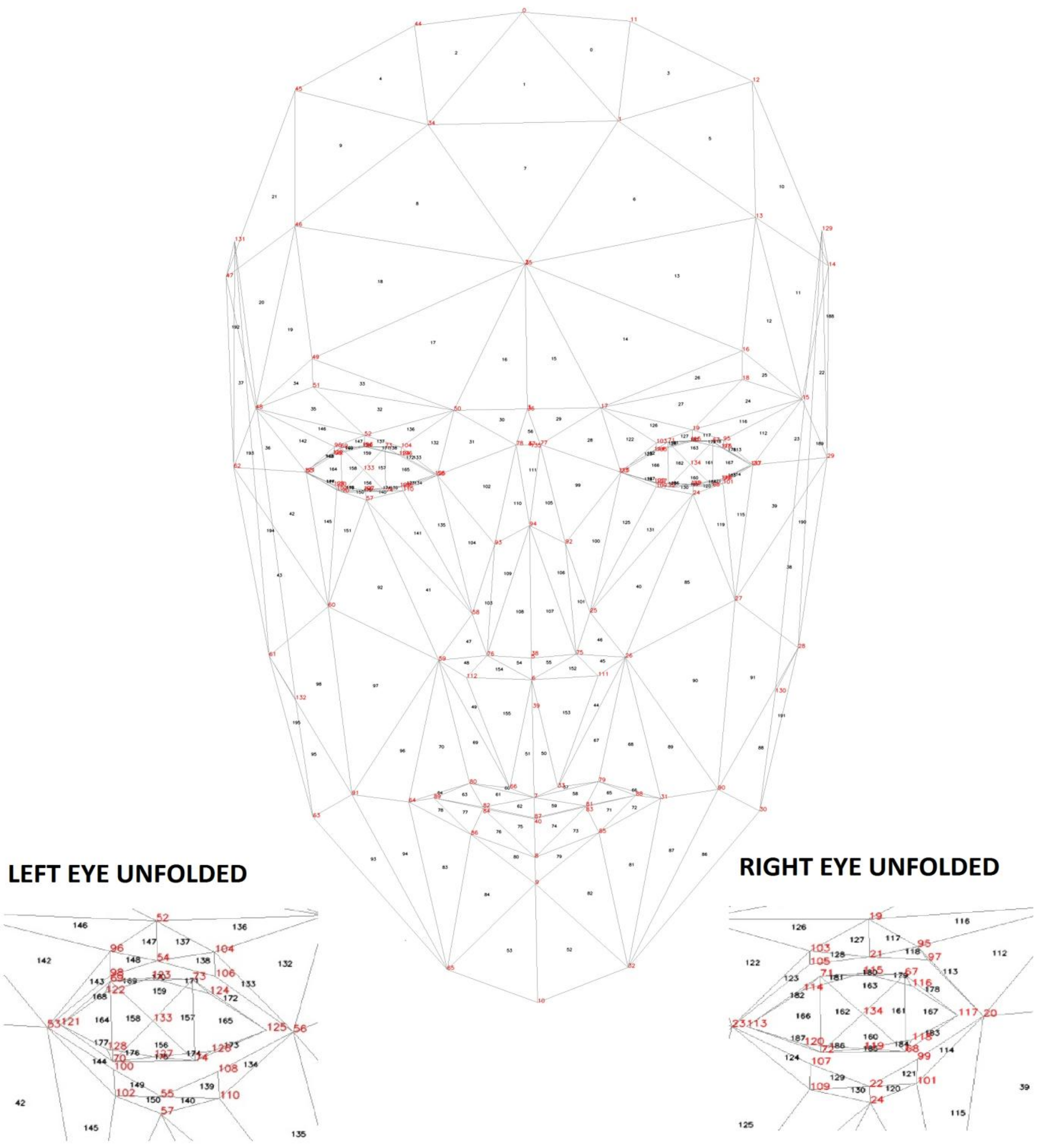

Figure 1. Candide model

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
Facial Features Tracker.cfg, Head Tracker.cfg, Facial Features Tracker - Manual Face Detection.cfg, FFT - HighPerformance.cfg, FFT - MidPerformance.cfg, FFT - LowPerformance.cfg, HT - HighPerformance.cfg, HT - MidPerformance.cfg, HT - LowPerfomance.cfg Facial_Features_Tracker_-_Web.cfg	Candide3.wfm	AU1: Nose wrinkler AU2: Jaw z-push (NOT ACTIVE) AU3: Jaw x-push (NOT ACTIVE) AU4: Jaw drop AU5: Lower lip drop (NOT ACTIVE) AU6: Upper lip drop (AU10) AU7: Lip stretcher (AU20) AU8: Lip corner depressor (AU13/15) AU9:Lip presser (AU23/24) (NOT ACTIVE) AU10: Outer brows rasier AU11: Inner brows raiser AU12: Brow lowerer AU13: Eyes 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
Offline Facial Features Tracker (semi automatic setup).cfg, Facial Features Tracker - Asymmetric.cfg	Candide-exp.wfm	AU1: Nose wrinkler AU2: Jaw z-push 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