Migration guide

Visage Technologies strives to *minimize* changes in API and configuration files when releasing new versions of the SDK. The inevitable changes are listed here, together with specific instructions for developers who have the existing applications built with older versions.

For each *visage|SDK* release, the incremental changes in relation to the previous release are listed. To apply the changes correctly, apply them in order from the older version to the newer version **without** skipping intermediary versions.

Contents

---

visage|SDK 9.0b1

General

VisageFaceAnalyser now offers multi-frame analysis functionality which performs estimations on the selected high-quality frames and returns an averaged, smoothed value. This functionality is available via the *VisageFaceAnalyser::analyseStream()* function.

Introducing smaller, faster and more accurate age and gender estimation models which completely replace the old models that will no longer be distributed.

API changes

C++ API:

VisageSDK namespace

Introducing new FaceData class member for getting the face bounding box:

- `FaceData::faceBoundingBox`

C# API:

VisageCSWrapper namespace

Introducing new FaceData class member for getting the face bounding box:

- `FaceData.FaceBoundingBox`

C++ API:

VisageFaceAnalyser API was refactored to optimise usage and to include newly introduced multi-frame analysis feature.

Old face analysis functions were removed:

- `VisageFaceAnalyser::estimateAge()`
- `VisageFaceAnalyser::estimateGender()`
- `VisageFaceAnalyser::estimateEmotion()`

and replaced with new ones:

- `VFAReturnCode VisageFaceAnalyser::analyseImage(const VsImage* image, const FaceData& faceData, const int options, AnalysisData& results)`
- `VFAReturnCode VisageFaceAnalyser::analyseStream(const VsImage* frame, const FaceData& faceData, const int options, AnalysisData& results, const int faceIndex = 0)`

- `FaceData::faceBoundingBox`

- `void VisageFaceAnalyser::resetStreamAnalysis(const int faceIndex = -1)`

Three separate calls to old analysis estimation functions can now be performed by a single call to *analyseImage()* or *analyseStream()* where parameter *options* specifies which analysis operations to perform.

For more details on the updated API please see VisageFaceAnalyser API documentation.

Introducing new *AnalysisData* structure which contains face analysis results (age, gender and emotions) returned by the *VisageFaceAnalyser::analyseImage()* and *VisageFaceAnalyser::analyseStream()* functions.

C# API:

Due to native *VisageFaceAnalyser API* changes, VisageCSWrapper's VisageFaceAnalyser API was updated accordingly.
New *AnalysisData* structure which contains face analysis results (age, gender and emotions) was added as well.

C++ API:

Prototype of VisageFeaturesDetector constructor and the following method has been changed:

- `VisageFeaturesDetector()`

  to:

  `VisageFeaturesDetector(const char* detectorConfigFile)`

- `bool VisageFeaturesDetector::Initialize(const char* path)`

  to:

  `bool VisageFeaturesDetector::Initialize()`

For more details please see VisageFeaturesDetector API documentation.

C# API:

Prototype of VisageFeaturesDetector constructor has been changed:

- `VisageFeaturesDetector()`

  to:

  `VisageFeaturesDetector(string detectorConfigFile)`

Following method has been removed:

- `bool VisageFeaturesDetector::Initialize(String path)`

For more details please see VisageFeaturesDetector API documentation.

C++ API:

Introducing new functions for getting and setting specific data paths required by tracking algorithm:

- `const string& getVftDataPath(void) const`
- `const string& getVfdDataPath(void) const`
- `const string& getPrDataPath(void) const`
- `const string& getErDataPath(void) const`
- `bool setVftDataPath(string vft_data_path)`
- `bool setVfdDataPath(string vfd_data_path)`
- `bool setPrDataPath(string pr_data_path)`
- `bool setErDataPath(string er_data_path)`

Removed functions for getting and setting the obsolete data path no longer required by tracking algorithm:

- `const string& getBdtsDataPath() const`
  `bool setBdtsDataPath(string bdts_data_path)`

C# API:

VisageConfiguration class contains new public attributes:

- `VisageConfiguration.VftDataPath`
- `VisageConfiguration.VfdDataPath`
- `VisageConfiguration.PrDataPath`
- `VisageConfiguration.ErDataPath`

Removed public attribute:

- `VisageConfiguration.BdtsDataPath`

Data folder changes:

Data files	Old location	New location
Face analysis data files	bdtsdata/LBF/vfadata	vfa
Face recognition data files	bdtsdata/NN/fr.*	vfr
Tracking algorithm data files	bdtsdata/FF/vnn	vft/ff
	bdtsdata/NN/vnn	vft/fa
Tracking features data files	bdtsdata/NN/pr.*	vft/pr
	bdtsdata/NN/*.lbf	vft/er
3D models data files	*root*	vft/fm

Data files changes:

Removed

- Old tracking algorithm data (*vft/fa*) - **landmarks.bin** and **init_shape.bin**
- Old gender estimation data (*vfa/gd*) - **gd.lbf**
- Old data path parameter - **bdts_data_path** parameter

Added

- New tracking algorithm data (*vft/fa*) - **aux_file.bin**
- New gender estimation data (*vfa/gd*) - **gd.vino.bin** and **gd.vino.xml**
- New data path parameters:
  - **vft_data_path** - path to the folder tracking algorithm data files required by tracker
  - **vfd_data_path** - path to the folder containing algorithm data file required by tracker and detector
  - **pr_data_path** - path to the folder containing containing pupils' refinement data files
  - **er_data_path** - path to the folder containing ears' refinement data files

Configuration files using the removed

bdts_data_path

parameter should be updated to use new data paths instead.

Modified

- Age estimation data (*vfa/ad*) - **ae.vino.bin** and **ae.vino.xml**

Projects using older versions of these files should be updated to contain the newest data files from the *data* folder.

visage|SDK 8.8

General

The default legacy algorithm that could be allowed via the *use_vnn* configuration parameter (value 0) has been removed. The VNN algorithm is now the default and only tracking and detection algorithm.

The algorithm can be configured to improve feature points precision and robustness, over tracking speed (performance) by setting the *refine_landmarks* parameter to 1 in the configuration file. Otherwise, when performance is preferred over feature points precision, the recommendation is to disable *refine_landmarks* parameter by setting it to 0 in the configuration file.

Face tracking and detection algorithms are enhanced so that they can track and detect faces wearing protective masks of various colors and patterns.

Landmark detection changes:

Visible contour landmarks from **group 13** (13.1-13.17) are **no longer available** (not detected or estimated). Instead, contour landmarks are now provided only in the form of a physical contour within **group 15** (15.1-15.17). For more details please see FDP documentation.

Changing the detection of contour points from visible to physical results in improved stability and accuracy of the 3D head-pose estimation.

API changes

Introducing new C++ and C# API for getting SDK version:

C++	const char* getVisageSDKVersion(void)
	Introducing new function in *VisageSDK namespace* for getting SDK version. Prototype is declared in the following headers; *VisageTracker.h, VisageFeaturesDetector.h, VisageFaceRecognition.h, VisageFaceAnalyser.h* and *VisageGazeTracker.h*.
C#	*VisageSDKVersion* class with method: `String^ Get(void)`
	Introducing *VisageSDKVersion* class in *VisageCSWrapper namespace* with method for getting SDK version

Parameter ***frame** is now mandatory for the following method:

```
VisageLivenessBlink::update(const FaceData *faceData, VsImage *frame)
```

Introducing new FaceData class member for getting the rotation of the face from the camera viewpoint:

C++	`FaceData::faceRotationApparent`
C#	`FaceData::faceRotationApparent`

Changes in configuration file

Parameters	Files
use_vnn	Facial Features Tracker - Ultra.cfg
lbf_stage_modifier	Facial Features Tracker - Low.cfg
lbf_nperturb	

lbf_nperturb_threads	

For detailed description of these changes, consult VisageTracker Configuration Manual, paragraph *2.1.Configuration parameters*.

Facial Features Tracker - High.cfg to **Facial Features Tracker.cfg**
Facial Features Tracker - High - With Ears.cfg to **Facial Features Tracker - With Ears.cfg**

refine_landmarks parameter

Parameter value	Effect
0	landmarks refinement is disabled
1	landmarks refinement is enabled

For detailed description of these changes, consult VisageTracker Configuration Manual, paragraph *2.1.Configuration parameters*.

Configuration file	Effect
Facial Features Tracker.cfg	*refine_landmarks* parameter set to 1 (enabled)
Head Tracker.cfg	*refine_landmarks* parameter set to 0 (disabled)

For detailed description of these changes, consult VisageTracker Configuration Manual, paragraph *2.1.Configuration parameters*.

If you want to update your existing configuration files, it is recommended to copy the parameters values from Facial Features Tracker.cfg configuration file supplied in this package.

Data files changes

As a consequence of improving algorithms, removing legacy algorithm, there are certain changes in the data files.

API	Status	Location	Files/Folders
`VisageTracker VisageFeaturesDetector`	REMOVED	*bdtsdata/NN/*	**fa.lbf**, **fc.lbf**, **is.bin**, **model.vino.bin**, **model.vino.xml**
`VisageTracker VisageFeaturesDetector`	REMOVED	*bdtsdata/FF/*	**ff.dat**
`VisageTracker VisageFeaturesDetector`	REMOVED	*bdtsdata/LBF/*	**lv**
`VisageTracker VisageFeaturesDetector`	REMOVED	*bdtsdata/NN/vnn*	**std_dev_image.bin**, **mean_image.bin**
`VisageTracker VisageFeaturesDetector`	MODIFIED	*bdtsdata/NN/vnn*	**bdtsdata/NN/vnn**

Projects using older versions of these files should be updated to contain the newest data files from the *bdtsdata* folder.

Sample changes

Samples displaying feature points are updated to draw physical contour points.

visage|SDK 8.7

General

The in-house developed runner is **no longer available** and is being replaced by OpenVINO™ toolkit, which is now the only and default neural network runner.

OpenVINO™ toolkit significantly improves the performance of visage|SDK algorithms.

It is implemented in the following libraries:

- OVPlugin.dll,
- inference_engine.dll,
- MKLDNNPlugin.dll,
- mkl_tiny_tbb.dll,
- tbb.dll
- libmmd.dll,
- svml_dispmd.dll

These libraries are dependencies of libVisageVision64.dll and should now be included in projects along with other visage|SDK libraries. Additionally, OpenVINO™ toolkit requires its own set of data files with extensions ***.vino.bin*** and ***.vino.xml*** provided in *Samples/data/bdtsdata*.

OpenVINO is a trademark of Intel Corporation or its subsidiaries.

The old face recognition model is being replaced by a smaller, faster and more accurate face recognition model.

Introducing a more accurate and robust face detection model. The new model is used in face tracking and face detection when *use_vnn* configuration parameter is set to 1. Otherwise, the previous face detection model will be used.

For more information about using VNN detection algorithm, please consult VisageTracker Configuration Manual, paragraph *2.1.Configuration parameters* and `VisageFeaturesDetector` documentation.

VNN algorithm can now be configured for better performance, at the cost of feature points precision. This mode is enabled by setting *use_vnn* parameter in configuration file to 1.

Recommended for usage when performance is preferred over feature points precision, e.g. when interested in fast performing head pose (head rotation and translation) estimation.

API changes

Added support for multiple image format for `extractDescriptor()` and `addDescriptor()` methods.

`extractDescriptor()`	In addition to RGB and grayscale, now accepts RGBA input image
`addDescriptor()`	In addition to RGB, now accepts RGBA and grayscale input image

Changes in configuration file

use_vnn configuration parameter added to `VisageFeaturesDetector` configuration file `FaceDetector.cfg`(located in *Samples/data/bdstdata*) and set to value 1.

VNN algorithm will be used in `VisageFeaturesDetector` API by default.

use_vnn configuration parameter values changed and additional value added:

Parameter value	Effect
0	VNN algorithm is disabled, default tracking algorithm and detection model are used
1	VNN algorithm enabled in fast mode, at the cost of feature points precision
2	VNN algorithm enabled, improved precision, accuracy and robustness

For detailed description of these changes, consult VisageTracker Configuration Manual, paragraph *2.1.Configuration parameters*.

If you want to update your existing configuration files, it is recommended to copy the parameters values from *Facial Features Tracker - Ultra.cfg* configuration file supplied in this package.

Data files changes

As a consequence of improving algorithms, removing in-house developed runner, and improving VNN algorithm, there are certain changes in the data files.

API	Status	Location	Files/Folders
`VisageFaceRecognition`	REMOVED	*bdtsdata/NN/*	**fr.bin**
`VisageFaceAnalyser`	REMOVED	*bdtsdata/LBF/vfadata/ad/*	**ae.bin**
`VisageTracker` `VisageFeaturesDetector`	REMOVED	*bdtsdata/NN/*	**pr.bin**, **model.bin**
`VisageFaceRecognition`	ADDED	*bdtsdata/FF/vnn/*	**fr.vino.bin**, **fr.vino.xml**
`VisageTracker` `VisageFeaturesDetector`	ADDED	*bdtsdata/FF/vnn/*	**ff.vino.bin**, **ff.vino.xml**
`VisageTracker` `VisageFeaturesDetector`	MODIFIED	*bdtsdata/NN/vnn*	**bdtsdata/NN/vnn**

Projects using older versions of these files should be updated to contain the newest data files from the *bdtsdata* folder.

Sample changes

VisageTrackerUnityPlugin

Function `_grabFrame()` captures RGB image, instead of BGR image.

VisageTrackerUnityDemo

The previously used BGRATex texture shader for applying captured frame image data was replaced with RGBATex texture shader (*Tracker.cs* script).

---

visage|SDK 8.6.1

General

To improve the performance of our algorithms and to support a wider variety of neural network models, we are introducing a configurable framework for choosing between different neural network runners.

As a result, additional configuration file *NeuralNet.cfg* is now included in visage|SDK (located in *Samples/data*). This file allows the users to configure which runner will be used by visage|SDK. Users can choose between:

- Visage Technologies' runner developed in-house and
- OpenVINO™ toolkit.

For more information on the parameters of *NeuralNet.cfg* file see the API page.

OpenVINO is a trademark of Intel Corporation or its subsidiaries.

New experimental algorithm for face tracking and alignment introduced - *VNN algorithm*.

For the price of slightly reduced tracking speed/performance, it significantly improves tracking quality and robustness. `VisageTracker` and `VisageFeaturesDetector` can be configured to use VNN algorithm via configuration parameter - *use_vnn*.

For more information on *use_vnn* parameter, please consult VisageTracker Configuration Manual, paragraph *2.1.Configuration parameters* and `VisageTracker` class documentation.

It is recommended to use VNN algorithm with OpenVINO™ toolkit which significantly improves the speed of running neural networks, thus mitigating any performance reductions.

OpenVINO is a trademark of Intel Corporation or its subsidiaries.

API changes

Introducing new C++ and C# API for programmatically changing `VisageTracker` configuration parameters, including new additional classes and templates:

`VisageConfiguration`	Modify configuration parameters on the fly
`VsCfgArr`	Helper template structure for storing various `VisageConfiguration` array data types

The aforementioned classes are used in conjunction with new `VisageTracker` methods:

C++	`VisageConfiguration VisageTracker::getTrackerConfiguration()`
	`void VisageTracker::setTrackerConfiguration(VisageConfiguration &configuration)`
C#	`VisageConfiguration VisageCSWrapper.VisageTracker.GetTrackerConfiguration()`
	`void VisageCSWrapper.VisageTracker.SetTrackerConfiguration (VisageConfiguration configuration)`

There are slight differences in usage between C++ and C#. For example, C++ API uses helper structure `VsCfgArr` to return specific data types where C# uses native C# types.

`VisageConfiguration` and `VsCfgArr` class documentation contains more details and examples of usage in code.

FDP group 10 (ears) has been extended from 10 to 24 points (12 points per ear) as part of the ear tracking feature.

FDP files saved with visage|SDK 8.6 will **not** be backwards compatible with the previous versions due to the addition of new FDP points.

Using `FDP::readFromFile()` to load an 8.6 FDP file in an earlier version of the visage|SDK will lead to undefined behavior.

`VisageTracker` method stop has been deprecated from both APIs.

C++	`VisageTracker::stop()`
C#	`VisageCSWrapper.VisageTracker.Stop()`

Prototype of method

```
void initializeLicenseManager(const char *licenseKeyFileName)
```

changed to

```
int initializeLicenseManager(const char *licenseKeyFileName)
```

It is no longer necessary to declare the licensing function prototype explicitly within your code. Including any of the following headers will also include the licensing prototype:

- *VisageTracker.h*
- *VisageFeaturesDetector.h*
- *VisageFaceRecognition.h*
- *VisageFaceAnalyser.h*

`FeaturePoint` class and `FDP` class have additional property and functions, respectively to conform with the C++ API.

`FeaturePoint`	`FeaturePoint.detected`
`FDP`	`bool FDP::FPIsDetected(int group, int n)` `bool FDP::FPIsDetected(String name)`

1 indicates that the feature point is obtained from a 2D image using the tracking algorithm. 0 indicates that the feature point is estimated from fitting a 3D model onto the detected feature points of the face.

Changes in configuration file

Two additional configuration files have been added. One for the ear tracking feature and one for the novel tracking algorithm.

Configuration name	Parameter changed /added	Parameter value
Facial Features Tracker - High - With Ears.cfg	efine_ears *_fitting_model *_fitting_fdp	1 jk_300_wEars.wfm jk_300_wEars.fdp
Facial Features Tracker - Ultra.cfg	use_vnn	1

For detailed description of these changes consult *VisageTracker Configuration Manual*, paragraph *1.1.Standard configuration files*

If you want to update your existing configuration files, it is recommended to copy the parameters values from *Facial Features Tracker - Ultra.cfg* configuration file supplied in this package.

- **refine_ears** parameter added, off (0) by default. Toggles the tracking and refinement of ear points (group 10) for `VisageTracker` and `VisageFeaturesDetector`.

Used together with ears 3D model - jk_300_wEars.wfm

- **use_vnn** parameter added, off (0) by default. Toggles usage of the new experimental algorithm for face tracking and alignment - VNN algorithm

For detailed description of these changes, consult *VisageTracker Configuration Manual*, paragraph *2.1.Configuration parameters.*

If you want to update your existing configuration files, it is recommended to copy the parameters values from *Facial Features Tracker - High.cfg* configuration file supplied in this package.

3D Model changes

A new model file has been added for ear tracking functionality - *jk_300_wEars.wfm*. The model contains an additional 334 polygons and its vertices are mapped to 14 new FDP points in group 10 (10.11 - 10.24).

For detailed description of these changes, consult *VisageTracker Configuration Manual*, paragraph *2.3.The 3D models used in tracking*

Data files changes

New data files and model files for ear tracking added

Location	Files
Samples/data/bdtsdata/NN	efa.lbf efc.lbf
Samples/data/	jk_300_wEars.wfm jk_300_wEars.fdp

New data folder and data files added for VNN algorithm in *Samples/data/bdtsdata/NN/vnn*

Projects using older versions of these files should be updated to contain the newest data files from the *bdtsdata* folder.

Sample changes

FacialAnimationUnityDemo sample application has been removed and will no longer be distributed.

VisageTrackerUnityDemo sample application is distributed as a Unity project, not as a prebuilt application.

Build and run instructions are provided in the *VisageTrackerUnityDemo* sample documentation.

Instructions on visage|SDK integration with Unity can be found here.

---

visage|SDK 8.5

Changes in configuration file

- **smoothing_factors** parameter
  Due to the re-implementation of the smoothing algorithm in `VisageTracker`, default values and optimal ranges for this parameter have been changed in all configurations.

Please consult *VisageTracker Configuration Manual*, paragraph *2.1.Configuration parameters* for additional information.

If you want to update your existing configuration files, it is recommended to copy the parameters values from *Facial Features Tracker - High.cfg* configuration file supplied in this package.

Sample changes

VisageRendering.cs:

Methods `DisplayEmotion()` and `DisplayAgeGenderName()` have changed the prototypes from:

```
public void DisplayEmotion(FaceData trackingData, float[] emotions, int
width, int height,
                           bool face_recognition_mode=false)
public void DisplayAgeGenderName(FaceData trackingData, float age, int
gender, string name,
                                 int width, int height, int
drawingOptions,
                                 bool face_recognition_mode=false)
```

to:

```
public void DisplayEmotion(FaceData trackingData, float[] emotions, int
width, int height,
                           bool display_background = false)
public void DisplayAgeGenderName(FaceData trackingData, float age, int
gender, string name,
                                 int width, int height, bool
display_background = true)
```

Build tools changes

Libraries built with msvc100 are no longer supported within the package.

Data files changes

visage|SDK data files located in *Samples/data/bdtsdata* folder have been updated.

Copy all files located in the *Samples/data/bdtsdata* folder to appropriate folders in any existing application.