FAQ

General questions

What skills do I need in order to use visage|SDK?

01 Jun 2020

Software development skills are required to make use of visage|SDK.

visage|SDK is a Software Development Kit - a set of documented software libraries that software developers can use to integrate Visage Technologies' face tracking, analysis and recognition algorithms into their applications.

See also:

- Can I install your software on my computer and just run it?
- Can you develop an application for me?
- Am I entitled to technical support?

Can I install your software on my computer and just run it?

02 Jun 2020

No. visage|SDK is not a finalized application, but a Software Development Kit that you use to develop /integrate your application. This is what your software development team can hopefully do, based on the documentation delivered as part of visage|SDK.

See also:

- What skills do I need in order to use visage|SDK?
- Can you develop an application for me?

Can you develop an application for me?

02 Jun 2020

Visage Technologies' software development team is available to develop custom-made applications to your requirements, using our face technology as the basis (we don't do general app development). To do that, we need specific and detailed requirements for the application, including:

- What should the application do?
- What sort of user interface should it have?
- On what kind of computer/device should it run (Windows, iOS, Android, or other)?

Some additional remarks on the way of working:

- Please note that these are just initial questions to get the discussion going, quite a bit more detail would probably be required, depending on the complexity of your requirements.
- Based on the requirements worked out to a suitable level, we could make a Project Proposal, including the time and cost, for building your application.
- Please note that working out your requirements and preparing the Project Proposal may be considerable work in itself so, depending on the complexity of your requirements, we may need to charge for that work as well.

Your contact person can provide further details.

See also:

- What skills do I need in order to use visage|SDK?
- Can I install your software on my computer and just run it?

Am I entitled to receive technical support?

30 May 2020

- Most of our licenses include the initial 5 hours of support. If you have purchased a license for visage|SDK, then you can use this support (delivered via email).
- For the majority of our clients, the initial support hours are more than sufficient, but it is also possible to order additional support. Your contact person can advise you on this.
- If you are evaluating visage|SDK and have technical issues, we will do our best, within reasonable limits, to support your evaluation.

See also:

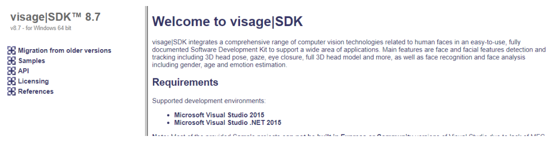
- How to request technical support?

How to request technical support?

30 May 2020

If you have a technical issue using visage|SDK, please email your Visage Technologies contact person the following information:

- error report, including the error messages you receive and anything else you think may help our team resolve the issue,
- the operating system you are using,
- the version of visage|SDK (you can find that information in the upper left corner of the offline documentation, as depicted in the image below. The offline documentation is found in the root-folder of visage|SDK packages as *Documentation.html*).


visage|SDK™ 8.7
v8.7 - for Windows 64 bit

- Migration from older versions
- Samples
- API
- Licensing
- References

Welcome to visage|SDK

visage|SDK integrates a comprehensive range of computer vision technologies related to human faces in an easy-to-use, fully documented Software Development Kit to support a wide area of applications. Main features are face and facial features detection and tracking including 3D head pose, gaze, eye closure, full 3D head model and more, as well as face recognition and face analysis including gender, age and emotion estimation.

Requirements

Supported development environments:

- **Microsoft Visual Studio 2015**
- **Microsoft Visual Studio .NET 2015**

See also:

- Am I entitled to technical support?

How do I migrate to a newer version of visage|SDK?

05 Jun 2020

Migration to a new version of visage|SDK is covered on the migration page. See Migration guide.

Typical steps when upgrading visage|SDK are:

1. Obtain new library files (usually in **visageSDK/lib**) (.dll, .so, .a, .lib) and overwrite the old ones in your application.
2. Obtain new header files (usually in **visageSDK/include**) and overwrite the old ones in your application.
3. Obtain new data files (usually in **visageSDK/Samples/data**) and overwrite the old ones in your application (models as well).
4. Read about the changes in parameters of the configuration files and apply them to your configuration. In case you use the default configuration, just overwrite it with the new one.

How does visage|SDK handle privacy? GDPR compliance?

30 Jun 2020

Regarding GDPR and privacy issues in general: visage|SDK runs entirely on the client device and it does not store or transmit any personal information (photos, names, or similar), or even any other information (other than Visage Technologies License Key File, for licensing purposes).

Languages, platforms, tools, etc.

Can I use visage|SDK with Unity?

21 Jul 2020

Integration with Unity 3D game engine is available with visage|SDK packages for Windows, iOS, Android, MAC OS X and HTML5. Each of these packages includes Unity sample projects to help you get started. Unity 3D integration is implemented via the VisageTrackerUnityPlugin plugin. This is a wrapper that exposes visage|SDK functionalities for use in C#.

You can either start your own project from scratch (Unity3D integration) or look for the Windows: VisageTrackerUnityDemo sample that comes with full source code so you can use it as know-how for your project.

Can I use visage|SDK with C#?

30 May 2020
For the C# API, please look in the API Reference. This is a managed C# wrapper that exposes all of visage|SDK functionalities – face tracking, analysis and recognition.

The C# API is implemented in VisageCSWrapper.dll library (libVisageVision64.dll and other dependencies are required for running).

Additionally, we provide VisageTrackerUnityPlugin, which is a C# wrapper made specifically for integration with Unity 3D engine. For more information, please take a look at Windows: Unity3D. The VisageTrackerUnityPlugin comes included with visage|SDK for Windows, MAC OS X, iOS and Android.

Can I use visage|SDK with Java?

30 May 2020

visage|SDK is implemented in C++. It provides C++ and C# APIs, but unfortunately currently does not include a Java API. visage|SDK for Android includes JNI-based Java wrappers as part of the sample projects provided in the package. You can use these wrappers as a starting point for your own projects in Java. In conclusion, it is certainly possible to use visage|SDK with Java, but it requires some additional effort for interfacing via JNA wrappers.

Can I use visage|SDK with VB.NET (Visual Basic)?

30 May 2020
visage|SDK does not currently provide a VB.NET API.

For users willing to experiment, there is an undocumented and currently unsupported workaround that should allow using it.

visage|SDK is implemented in C++. It provides C++ and C# APIs. On Windows, the C# API is implemented as a C++/CLI library (VisageCSWrapper.dll) which, in theory, should be usable in other NET languages, e.g. VB.NET. However, we have not tested this. The C# API documentation may be used as a basis for using VisageCSWrapper C++/CLI library in VB.NET.

Can I use visage|SDK with Python?

30 May 2020
visage|SDK does not currently provide a Python API.

For users willing to experiment, there is an undocumented workaround that allows users to use Python.

visage|SDK is implemented in C++ and provides C++ API which cannot easily be used directly in Python without a C functions wrapper. visage|SDK provides such a wrapper in the form of VisageTrackerUnityPlugin which was made specifically for integration with Unity 3D engine. However, it can also be used by other applications/languages that support importing C functions from a library. At its core, the VisageTrackerUnityPlugin is a high-level C functions API wrapper around C++ API. In the case of Python, ctypes library (foreign function library for Python) can be used to import and use C functions from VisageTrackerUnityPlugin. As the source code is provided, VisageTrackerUnityPlugin can also be used to implement a custom Python wrapper.
Even though it was tested, such usage of VisageTrackerUnityPlugin is not officially supported.

Can I use visage|SDK with UWP?

30 May 2020

Our API is a lower-level C++ API and we have no specific UWP integration features, but we see no reason why you would not be able to use it in a UWP app.

Can I use visage|SDK with React Native?

01 Jun 2020

visage|SDK does not currently provide direct support for React Native.

For users willing to experiment, there is an undocumented workaround that allows the use of visage|SDK in React Native.

visage|SDK is implemented in C++ and provides C++ API, therefore direct calls from React Native are not possible without a wrapper with C-interface functions. An example of such a wrapper is provided in visage|SDK in the form of VisageTrackerUnityPlugin which provides simpler, high-level API through a C-interface. It is intended for integration with Unity 3D (in C# with P/Invoke), but it can also be used by other applications/languages that support importing and calling C-functions from a native library, including React Native.

Can I use visage|SDK with Flutter?

09 Jul 2020

visage|SDK does not currently provide direct support for Flutter.

For users willing to experiment, there is an undocumented workaround that allows the use of visage|SDK in Flutter.
visage|SDK is implemented in C++ and provides C++ API. Therefore, direct calls from Flutter are not possible without a wrapper with C-interface functions. An example of such a wrapper is provided in visage|SDK in the form of VisageTrackerUnity Plugin which provides simpler, high-level API through a C-interface. It is intended for integration with Unity 3D (in C# with P/Invoke), but it can also be used by other applications/languages that support importing and calling C-functions from a native library, including Flutter.

See also

- C and C++ interop: https://flutter.dev/docs/development/platform-integration/c-interop

Can I use visage|SDK with Postman?

27 Nov 2020

visage|SDK does not currently provide direct support for Postman.

Postman is a scalable Web API testing tool that quickly integrates into CI/CD pipeline. As visage|SDK does not provide an out-of-the-box Web (REST/SOAP) API it is not possible to directly call to visage|SDK from Postman. To use Postman you would first need to implement a Web API on your own server using visage|SDK for Linux, Windows or HTML5 depending on your choice of server OS.

Can I use visage|SDK with Swift?

30 Mar 2022

visage|SDK provides an Objective-C wrapper for the native C++ API, which gives the possibility to invoke SDK functionality from Swift through the use of a bridging header.

Swift support is available only on the iOS and macOS platforms.

Can I use visage|SDK in WebView (in iOS and Android)?

30 May 2020

We believe that it should be possible to use visage|SDK in WebView, but we have not tried that nor have any clients currently who have done that so we cannot guarantee it. The performance will almost certainly be lower than with a native app.

Is there visage|SDK for Raspberry PI?

30 May 2020
visage|SDK 8.4 for rPI can be downloaded from the following link: https://www.visagetechnologies.com/downloads/visageSDK-rPI-linux_v8.4.tar.bz2
Once you unpack it, you will find the documentation in the root folder. It will guide you through the API and the available sample projects.

Important notes:

- Because of very low demand we currently provide visage|SDK for rPI on-demand only.
- The package you have received is visage|SDK 8.4. The latest release is visage|SDK 8.7 but that is not available for rPI yet. If your initial tests prove interesting, we will need to discuss the possibility to build the latest version on-demand for you. visage|SDK 8.7 provides better performance and accuracy, but the API is mostly unchanged so you can run relevant initial tests.
- visage|SDK 8.4 for rPI has been tested with rPI3b+; it should work with rPI4 but we have not tested that.

Will visage|SDK for HTML5 work in browsers on smartphones?

30 Sep 2020
The HTML5 demo page contains the list of supported browsers: https://visagetechnologies.com/demo/#supported-browsers

Please note that the current HTML5 demos have not been optimized for use in mobile browsers. Therefore, for the best results, it is recommended to use a desktop browser.

On iOS, the HTML5 demos work in Safari browser version 14 and higher. They do not work in Chrome and Firefox browsers due to limitations on camera access.
(https://stackoverflow.com/questions/59190472/webrtc-in-chrome-ios)

Which browsers does visage|SDK for HTML5 support?

30 May 2020
The HTML5 demo page contains the list of supported browsers: https://visagetechnologies.com/demo/#supported-browsers

Please note that the current HTML5 demos have not been optimized for use in mobile browsers. Therefore, for the best results, it is recommended to use a desktop browser.

Can visage|SDK libraries for HTML5 be used as JavaScript modules (in Node.js, ReactJS, Angular...)?

09 Mar 2021
From visage|SDK 8.7 Stable version it is no longer possible. However, the HTML5 libraries can be included in Node.js projects or any JavaScript framework using the script element (https://developer.mozilla.org/en-US/docs/Web/API/HTMLScriptElement#dynamically_importing_scripts), taking into account the order of loading scripts described on the API page.

Unity

How can I use VisageTrackerUnityDemo exported as a Unity Library in an Android Studio project?

The MainUnityActivity.java source file must be modified slightly for this to work:

- "*import com.visagetechnologies.androidcameraplugin.CameraActivity;*" line added to top of file
- "*public class MainUnityActivity extends UnityPlayerActivity*" must be changed to "*public class MainUnityActivity extends CameraActivity {*"
- "*protected void onCreate(Bundle savedInstanceState) {*" changed to "*public void onCreate(Bundle savedInstanceState) {*"

Why does my Unity application including visage|SDK crash on Unity versions 2019.2.0 and above?

There is a known bug involving the Intel® Open Image Denoise library included in Unity versions >2019.2.0 which causes a crash.

The bug was acknowledged and fixed in Unity 2021.2.0 and higher, so it is recommended to upgrade to this version.

Alternatively, if you do not have the option to upgrade; a workaround is to replace the libOpenImageDenoise.dylib used by Unity with at least version 1.2.3 or above which fixes this: https://github.com/OpenImageDenoise/oidn/releases/tag/v1.2.3

The affected library is located in (/Applications/Unity/Hub/Editor//Unity/Contents/Frameworks).

Internet connection and privacy issues

Does my device/app need internet connection?

09 Jul 2020

Applications developed using visage|SDK do not need an internet connection for operation.

Internet connection is needed only for license registration, which happens only once when the application is used for the first time.

For hardware devices, this can typically be done at installation time – you would connect the device to the network, run the application once, the license registration would happen automatically, and after that, no network connection is needed.

How do you handle the privacy of user data? Does visage|SDK store or send it to a cloud server?

09 Jul 2020

visage|SDK never transfers any user information to any server.

visage|SDK never automatically stores any user information locally.

In summary, you as the application developer have full control and responsibility for any storage or transfer of user data that you may implement in your application.

Cameras, hardware

Can I use visage|SDK with an IP camera?

30 May 2020

Yes, visage|SDK can be used with any IP camera. However, note that visage|SDK actually does not include camera support as part of the API; the API simply takes a bitmap image. Image grabbing from a camera is implemented at the application level.

Our sample projects show how to access local cameras and we currently have no ready-to-go code for accessing IP cameras. There are various ways to grab images from IP cameras, for example by using libVLC.
Usually, IP cameras should provide a URL from which you can access the raw camera feed. If you can open the URL and see the feed in VideoLAN VLC Media Player, the same URL can be used in libVLC to access the feed programmatically.

See also:

- libVLC (C++): https://wiki.videolan.org/LibVLC_Tutorial/
- libVLC (C#): https://github.com/videolan/libvlcsharp
- VideoLAN VLC Media Player: https://www.videolan.org/vlc/index.html

Are these camera specs OK for visage|SDK: 1920 x 1080, 30 fps, mono?

12 Jun 2020

The mentioned camera parameters (1920 x 1080, 30 fps, mono) should be appropriate for our software and most of the use cases. With these camera parameters visage|SDK can provide sustainable tracking up to 5 meters.

For further details on inputs and other features specifically for face tracking, please see https://visagetechnologies.com/facetrack-features.

What should be the camera Field of View (FoV)?

12 Jun 2020

Regarding the FoV, the selection should primarily depend on the use case (planned position of the camera and the captured scene). Our algorithm should be robust enough to handle some optical distortions that may be a consequence of lenses with large FoV. However, extreme distortions (e.g. fish-eye lens) will negatively affect the algorithm's performance.

Face Tracking

Can I process a video and save tracking data/information to a file?

07 Jul 2020

There is no ready-made application to do this, but it can be achieved by modifying the existing sample projects. Such modification should be simple to do for any software developer, based on the documentation delivered as part of visage|SDK. We provide some instructions here.

To get you started, each platform-specific visage|SDK package contains sample projects with full source code that can be modified for that purpose. See the Samples page to find the documentation of the specific sample projects.

On Windows, there are Visual Studio 2015 samples: *Windows: ShowcaseDemo* which is written in C# and *Windows: FaceTracker2* which is written in C++.

On macOS, there is an Xcode sample: *VisageTrackerDemo* which is written in Objective-C++.

Processing and saving information to a file can be implemented in parts of the sample projects where tracking from a video file is performed:

- In *ShowcaseDemo* project on Windows, the appropriate function is `worker_DoWorkVideo(` .
- In *FaceTracker2* sample project on Windows, the appropriate function is `CVisionExampleDoc::trackFromVideo()`.
- In *VisageTrackerDemo* sample project on macOS, the appropriate function is `trackingVideoThread()`.

Please search the source code of each sample for the exact location.

Sample projects have “video_file_sync“ (or similarly named functionality) enabled, which skips video frames if tracking is slower than real-time. This functionality should be disabled for full video processing i.e processing of all video frames.

See also:

- What skills do I need in order to use visage|SDK?
- Can you develop an application for me?

How many faces can be tracked simultaneously?

30 May 2020

The maximum number of faces that can be tracked simultaneously is internally limited to 20 for performance reasons.

Using API Reference VisageFeaturesDetector, up to 512 faces can be detected in an image.

How far from the camera can the face be tracked/detected?

30 May 2020

The face detection distance is limited by the size of the facial bounding box relative to the long side of the input image (i.e. width or height). For a face to be detected, it's bounding box size should amount to at least 5% of the long side. The exact distance in meters depends on the camera parameters such as focal length and aspect ratio. For example, for a Logitech C920 webcam set to use 1920×1080 resolution, the minimal facial bounding box size is 96×96 pixels (1920 x 0,05) which corresponds to faces up to ~5 meters from the camera.

See also:

- How far from the camera can the face be recognized?

Face tracking does not work as I expected

30 May 2020

Our face tracking algorithm is among the best ones available, but like all Computer Vision algorithms, it has its limits related to image quality, light conditions, occlusions, or specific reasons such as head pose.

If you notice specific issues or have special requirements, you may send us your test video footage and any specific requests, and we will process it and send you back the tracking results. This can allow us to fine-tune the tracker configuration to your specific requirements and send you the best possible results.

How can I test and use ear tracking?

19 Jun 2020

The fastest way to test ear tracking is in the online Showcase Demo (https://visagetechnologies.com/demo/). Simply enable Ears in the Draw Options menu.

The online demo is based on visage|SDK for HTML5 and has some limitations due to the HTML5 implementation. For even better performance, you may want to download other visage|SDK for Windows, Android or iOS - each of these packages contains a native *ShowcaseDemo* in which ear tracking can be tried.

If you are already developing using visage|SDK and want to enable ear tracking, you can use the ready-made configuration file “*Facial Features Tracker - High - With Ears.cfg*“. Ear tracking is enabled using the *refine_ears* configuration parameter.

See also:

- VisageTracker Configuration Manual.pdf for more information on tracker configuration and *refine_ears* configuration parameter
- API Reference VisageConfiguration for more information on modifying tracker configuration parameters during runtime
- API Reference FaceData structure for more information on accessing the tracking results

Face Analysis

Can I process a video and save analysis data to a file?

28 Dec 2020

There is no ready-made application to do this, but it can be achieved by modifying the ShowcaseDemo sample project in visage|SDK for Windows. Such modification should be simple to do for any software developer, based on the documentation delivered as part of visage|SDK and the instructions below.

Saving analysis data to a file can be achieved by inserting an export function to the *VisageSDK::VisageFaceAnalyser* functions. In *visageSDK\Samples\OpenGL\build\msvc140\ShowcaseDemo\ShowcaseDemo.xaml.cs* you will find *gVisageFaceAnalyser::estimateEmotion(), ::EstimateAge()* and *::EstimateGender()* in the *AnalyzeFace()* function. The *gEmotionFilterList[index]*, *gAgeArray[index]* and/or *gGenderArray[index]* are the raw results that can be saved to a file.

In the sample, the *Filter()* function normalizes the outputs. You can also save these normalized results: *gEmotionsSmoothed[index]* , *gAgeSmoothed[index]* and *gGenderSmoothed[index]*.

The *index* in all these values is the face-index. If multiple faces appear in a video, the analysis of their faces will be stored in a separate index.

The ShowcaseDemo analyzes every tracked frame by default. If so desired, you could implement a counter in the *runFaceAnalysis()* function to perform the analysis (and added save-to-file) less frequently. The demo has *video_file_sync* enabled, which skips video frames if tracking is slower than real-time. This functionality should be disabled for full video processing, i.e processing of all video frames.

Frame grabbing and the processing of video files is implemented in *worker_DoWorkVideo()*. Here you can deal with the creation of a file for the analysis results and what to do once a video ends. Automatically running the next video in the same directory can also be implemented here.

See also:

- API Reference VisageFaceAnalyser
- What skills do I need in order to use visage|SDK?
- Can you develop an application for me?

Face Recognition

How far from a camera can a face be recognized?

24 Jun 2020

This depends on camera resolution. Face Recognition works best when the size of the face in the image is at least 100 pixels.

See also:

- How far from the camera can the face be tracked?

How do I perform liveness detection with visage|SDK?

30 May 2020

visage|SDK includes active liveness detection. The user is required to perform a simple facial gesture (smile, blink, or raise eyebrows). Face tracking is then used to verify that the gesture is actually performed. You can configure which gesture(s) you want to include. As the app developer, you also need to take care of displaying appropriate messages to the user.

visage|SDK includes the API for liveness detection. However, only the visage|SDK for Windows and visage|SDK for Android contains a ready-to-run demo of Liveness Detection. So, for a quick test of the liveness detection function, it would probably be the easiest to download visage|SDK for Windows, run “DEMO_FaceTracker2.exe” and select “Perform Liveness” from the Liveness menu.

The technical demos in Android and Windows packages of visage|SDK include the source code intended to help you integrate liveness detection into your own application.

See also:

- Windows: FaceTracker2
- API Reference VisageLivenessBlink
- API Reference VisageLivenessBrowRaise
- API Reference VisageLivenesSmile
- API Reference VisageLivenessAction

How do I perform the identification of a person from a database?

29 Jun 2020

This article outlines the implementation of using face recognition for identifying a person from a database of known people. It may be applied to cases such as whitelists for access control or attendance management, blacklists for alerts, and similar. The main processes involved in implementing this scenario are registration and matching, as follows.

Registration

Assuming that you have an image and an ID (name, number or similar) for each person, you register each person by storing their *face descriptor* into a *gallery* (database). For each person, the process is as follows:

- ***Locate the face in the image***:
    - To locate the face, you can use detection (for a single image) or tracking (for a series of images from a camera feed).
        - See function *VisageTracker::track()* or *VisageFeaturesDetector::detectFacialFeatures()*.
    - Each of these functions returns the number of faces in the image - if there is not exactly one face, you may report an error or take other actions.
    - Furthermore, these functions return the *FaceData* structure for each detected face, containing the face location.
- Use *VisageFaceRecognition::addDescriptor()* to ***get the face descriptor and add it to the gallery*** of known faces together with the name or ID of the person.
    - The descriptor is an array of short integers that describes the face - similar faces will have similar descriptors.
    - The gallery is simply a database of face descriptors, each with an attached ID.
        - Note that you could store the descriptors in your own database, without using the provided gallery implementation.
- ***Save the gallery*** using *VisageFaceRecognition::saveGallery()*.

Matching

At this stage, you match a new facial image (for example, a person arriving at a gate, reception, control point or similar) against the previously stored gallery, and obtain IDs of one or more most similar persons registered in the gallery.

- First, ***locate the face(s)*** in the new image.
    - The steps are the same as explained above in the Registration part. You obtain a *Face Data* structure for each located face.
- Pass the *FaceData* to *VisageFaceRecognition::extractDescriptor()* to get the face descriptor of the person.
- Pass this descriptor to *VisageFaceRecognition::recognize()*, which will match it to all the descriptors you have previously stored in the gallery and return the name/ID of the most similar person (or the desired number of most similar persons);
    - the *Recognize()* function also returns a similarity value, which you may use to cut off the false positives.

See also:

- API Reference FaceData
- API Reference VisageFaceRecognition
- API Reference VisageTracker
- API Reference VisageFeaturesDetector

How do I perform verification of a live face vs. an ID photo?

25 Jun 2020

The scenario is the verification of a live face image against the image of a face from an ID. This is done in four main steps:

1. Locate the face in each of the two images;
2. Extract the face descriptor from each of the two faces;
3. Compare the two descriptors to obtain the similarity value;
4. Compare the similarity value to a chosen threshold, resulting in a match or non-match.

These steps are described here with further detail:

1. In each of the two images (live face and ID image), the ***face first needs to be located***:
    - To locate the face, you can use detection (for a single image) or tracking (for a series of images from a camera feed).
        - See function *VisageSDK::VisageTracker::track()* or *VisageSDK::VisageFeaturesDetector::detectFacialFeatures().*
    - Each of these functions returns the number of faces in the image - if there is not exactly one face, you may report an error or take other actions.

    - Furthermore, these functions return the *FaceData* structure for each detected face, containing the face location.
    - Note: the ID image should be cropped so that the ID is occupying most of the image (if the face on the ID is too small relative to the whole image it might not be detected).
2. The next step is to ***extract a face descriptor*** from each image. The descriptor is an array of short integers that describes the face. Similar faces will have similar descriptors.
    - From the previous step, you have one *FaceData* structure for the ID image and one *FaceData* structure for the live image.
    - Pass each image with its corresponding FaceData to the function *VisageFaceRecognition::extractDescriptor()*.
3. Pass the two descriptors to the function *VisageFaceRecognition::descriptorsSimilarity()* to ***compare the two descriptors*** to each other and obtain the measure of their similarity. This is a float value between 0 (no similarity) and 1 (perfect similarity).
4. If the similarity is ***greater than the chosen threshold***, consider that the live face ***matches*** the ID face.
    - By choosing the threshold, you control the trade-off between False Positives and False Negatives:
        - If the threshold is very high, there will be virtually no False Positives, i.e. the system will never declare a correct match when, in reality, the live person is not the person in the ID.
        - However, with a very high threshold, a False Negative may happen more often - not matching a person who really is the same as in the ID, resulting in an alert that will need to be handled in an appropriate way (probably requiring human intervention).
        - Conversely, with a very low threshold, such “false alert” will virtually never be raised, but the system may then fail to detect True Negatives - the cases when the live person really does not match the ID.
        - There is no “correct” threshold because it depends on the priority of a specific application. If the priority is to avoid false alerts, the threshold may be lower; if the priority is to avoid undetected non-matches, then the threshold should be higher.

See also:

- API Reference FaceData
- API Reference VisageFaceRecognition
- API Reference VisageTracker
- API Reference VisageFeaturesDetector

How-to technical articles

How do I determine if a person is looking at the screen?

17 Jul 2020

visage|SDK does not have an out-of-the-box option to determine if the person is looking at the screen. However, it should not be too difficult to implement that. What visage|SDK does provide is:

- The 3D position of the head with respect to the camera;
- The 3D gaze direction vector.

Now, if you also know the size of the screen and the position of the camera with respect to the screen, you can:

- Calculate the 3D position of the head with respect to your screen;
- Cast a ray (line) from the 3D head position in the direction of the gaze;
- Verify if this line intersects the screen.

Please also note that the estimated gaze direction may be a bit unstable (the gaze vectors appearing “shaky”) due to the difficulty of accurately locating the pupils. At the same time, the 3D head pose (head direction) is much more stable. Because people usually turn their heads in the direction in which they are looking, it may also be interesting to use the head pose as the approximation of the direction of gaze.

Here is a code snippet used to roughly calculate screen space gaze by combining data from visage|SDK and external information about screen (in meters) and camera relation which can be used to determine if the user is looking at the screen or not (if calculated coordinates are outside the screen range):

```
   // formula to get screen space gaze

    x = faceData->faceTranslation[2] * tan(faceData->faceRotation[1] +
faceData->gazeDirection[0] + rOffsetX) / screenWidth; // rOffsetX angle of
camera in relation to screen, ideally 0

    y = faceData->faceTranslation[2] * tan(faceData->faceRotation[0] +
faceData->gazeDirection[1] + rOffsetY) / screenHeight; // rOffsetY angle
of camera in relation to screen, ideally 0

    // apply head and camera offset

    x += -(faceData->faceTranslation[0] + camOffsetX); // camOffsetX in
meters from left edge of the screen

    y += -(faceData->faceTranslation[1] + camOffsetY); // camOffsetY in
meters from top edge of the screen
```

See also:

- API Reference FaceData
- API Reference FaceData featurePoints3D
- API Reference FaceData faceRotation

Can images of crowds be processed?

24 Jul 2020

visage|SDK can be used to locate and track faces in group/crowd images and also to perform face recognition (identity) and face analysis (age, gender, emotion estimation). Such use requires particular care related to performance issues since there may be many faces to process. Some initial guidelines:

- Face tracking is limited to 20 faces (for performance reasons). To locate more faces in the image, use face detection (class `FacialFeaturesDetector`).
- visage|SDK is capable of detecting/tracking faces whose size in the image is at least 5% of the larger resolution i.e. image width in case of landscape images and image height in case of portrait images.
    - The default setting for the `VisageFeaturesDetector` is to detect faces larger than 10% of the image size and 15% in case of the `VisageTracker` . The default parameter for minimal face scale needs to be modified to process smaller faces.
    - If you are using high-resolution images with many faces, so that each face is smaller than 5% of image width, a custom version of visage|SDK may be discussed.
- For optimal performance of algorithms for face recognition and analysis (age, gender, emotion), faces should be at least 100 pixels wide.

See also:

- API Reference VisageTracker
- API Reference VisageFeaturesDetector
- VisageTracker Configuration Manual.pdf for more information on tracker configuration and *min_face_scale* and *max_face_scale* parameter

If tracking is temporarily broken, will the tracker upon recovered tracking know it is the same person?

20 Nov 2020

The tracker of visage|SDK has a configuration parameter called recovery_timeout. You can adjust this value to your liking, and the tracker will assume (without Face Recognition it cannot know) that any face that reappears in frame within that recovery timeout is the same person it was tracking before.

Adjusting this value can help accommodate temporary disruptions in tracking. However, there is the risk that if a person leaves and the time window is long enough, a new person can walk into the frame and be tracked under the assumption of being the same person.

In case a lot of people freely come and go, using Face Recognition would be a must to avoid the situation where people quickly swap in and out of frame and the exposure time of a new person is falsely added to someone who left the frame a few seconds ago.

See also:

- VisageTracker Configuration Manual.pdf for more information on tracker configuration and the *recovery_timeout* parameter

How do I animate a face using visage|SDK?

02 Feb 2021

visage|SDK provides accurate real-time face tracking that can be used to drive facial animation. It is important to note that visage|SDK provides only face tracking and does ***not*** include animation tools. The user of visage|SDK, i.e. you, needs to implement or use a facial animation system and map the tracking results from visage|SDK to drive animation - this article provides guidelines for doing this. The provided guidelines are general and the actual implementation will greatly depend on the type of facial rig you use for your animated character, e.g. blendshapes, bones or other types.

While tracking a face in video, visage|SDK provides a large set of face tracking data for each video frame. The data that is most relevant for driving facial animation is:

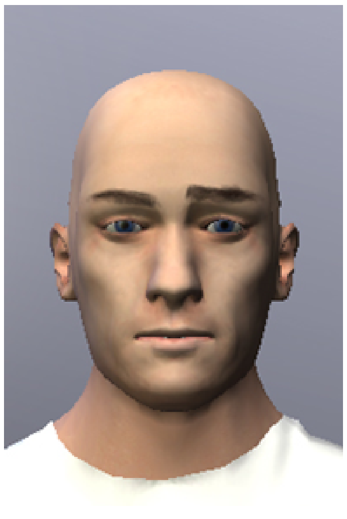
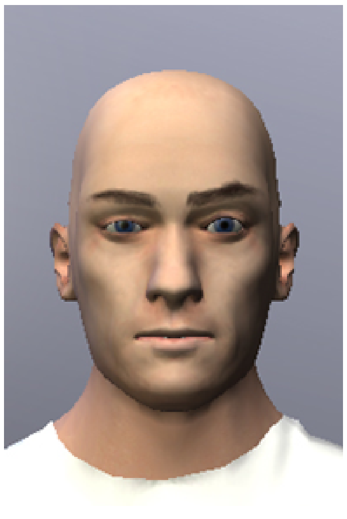
- 3D head pose (translation, rotation) can be mapped to the head pose of your animated character;
- Gaze direction can be mapped to the gaze (eyes) direction of your character;
- 3D feature points may be mapped onto your character’s animation rig;
- Action Units may be mapped onto your character’s animation rig.

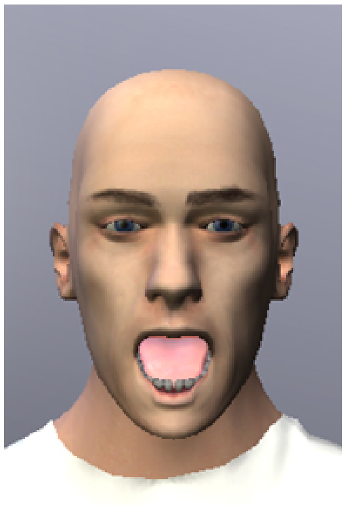
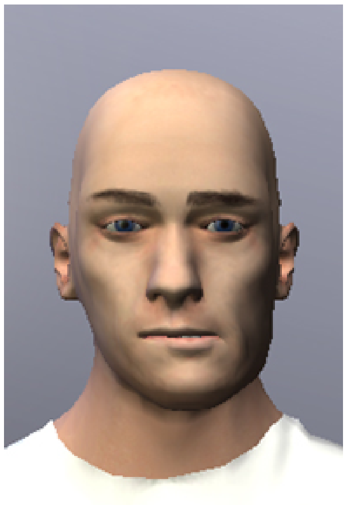
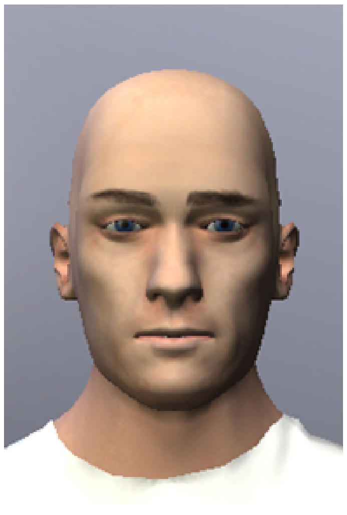
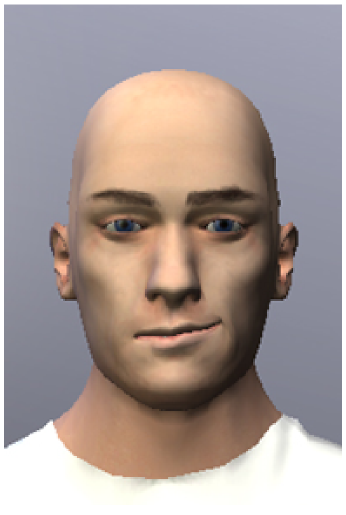
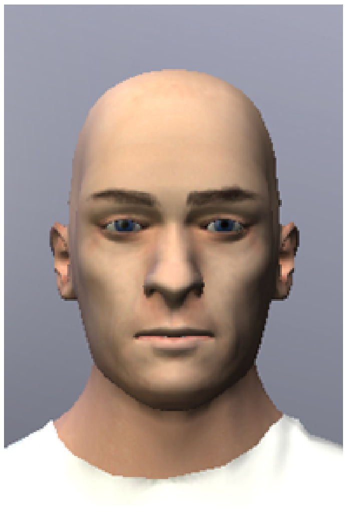
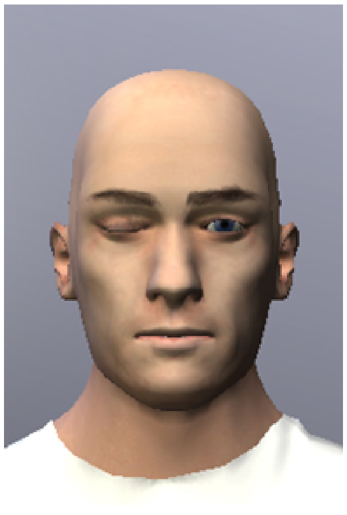
Animation using Feature Points

The feature points (specifically, featurePoints3DRelative) are key points on the lips, eyes, eyebrows, chin etc. They are given in a local 3D coordinate system of the face which makes them suitable to map onto the 3D animation rig of your character. For example, they could be mapped onto the bones of a bones-based animation rig.

Animation using Action Units

Action Units represent high-level facial actions such as mouth opening or eyebrow raising. The main Action Units are visualized in the following figures (symmetrical ones shown only for the left side).

neutral expression	 AU11: Left inner brows raiser	 AU10: Left outer brow raiser

 AU4: Jaw drop	 AU3: Jaw x-push	 AU2: Jaw z-push
 AU7: Lip stretcher left	 AU1: Nose wrinkler	 AU13: Left eye closed

Note that Action Units are not expressed in units such as meters - these images correspond to the Action Unit values of 1. So, for example, the value of 1 for *AU4: Jaw drop* signifies a fully open mouth; a value of 0.5 would mean a half-open mouth.

Action Units may be naturally mapped onto a blendshape-based animation system; to do that, it is necessary to create blendshapes corresponding to these Action Units.

Note that Action Units are actually defined as blendshapes of an internal 3D face model in visage|SDK; this internal face model is used during tracking to estimate the Action Unit values. It is possible to configure or even fully replace the internal 3D face model if that is desired, however this is a fairly advanced topic. Details are available in the Visage Tracker Configuration Manual.

See also:

- API Reference FaceData
- API Reference FaceData featurePoints3DRelative
- API Reference FaceData ActionUnits
- VisageTracker Configuration Manual.pdf to configure the internal 3D face model (advanced topic)

How do I implement AR applications such as virtual glasses?

02 Feb 2021

visage|SDK contains several sample AR projects with full source code and models, that can be used as a starting point for your own projects. An overview of these sample projects is given in the table below.

Sample project	Available in	AR effect demonstrated	Implementation
VisageTrackerUnityDemo	visage|SDK for Windows visage|SDK for iOS visage|SDK for Android visage|SDK for Mac OS X visage|SDK for HTML5	virtual glasses face mask (tiger)	Unity 3D
Virtual Eyewear Try-on	visage|SDK for HTML5	virtual glasses	HTML5/three.js
Windows ShowcaseDemo	visage|SDK for Windows	face mask (tiger)	C#/OpenGL

This article provides general considerations about 3D object positioning and occlusion handling for AR, based on the example of 3D glasses. These considerations are applied in the above-mentioned sample projects, and can also be used in your own projects.

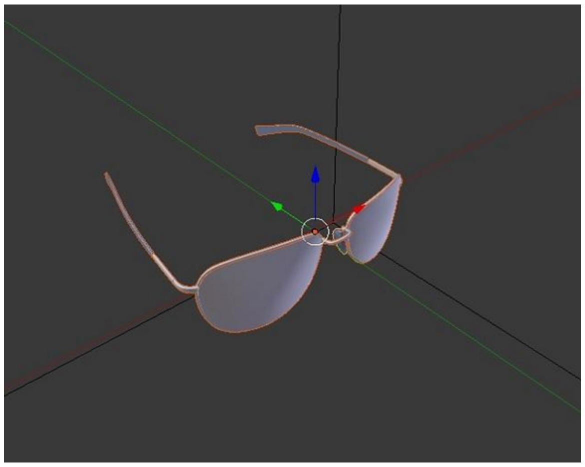
Positioning of AR objects on the face

The 3D objects need to be placed into the local 3D coordinate system of the face, as defined in visage|SDK: the origin (0,0,0) is placed at the center between the eyes; the x-axis points laterally towards the side of the face, y-axis points up and z-axis points into the face - see illustration here. The units are expressed in meters.

Therefore the glasses should be modeled in the following scale and position:

- Real-life size, expressed in meters.
- X positioning: center of glasses at X=0
- Y positioning: visual axes of the glasses at Y = 0
- Z positioning: eyeball front edge at Z = 0 (therefore lenses at approx. Z = -0.02 depending on glasses model)

Note that the coordinate system can be different in your modeling tool.



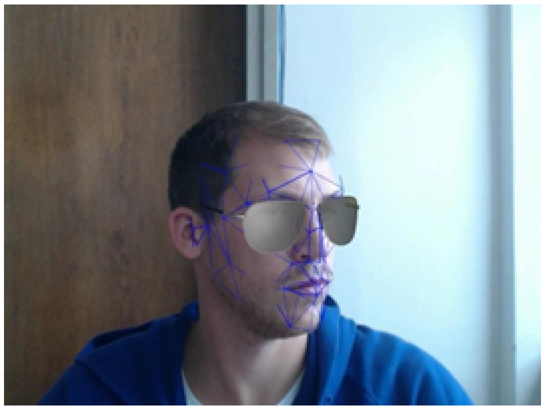
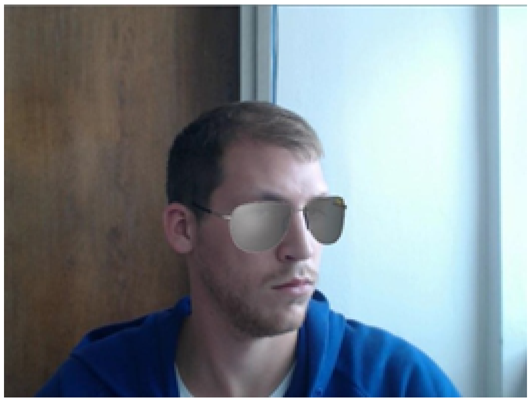
Occlusion

If occlusions are not handled, glasses would always appear on top of the face like this:

In reality, parts of the glasses are occluded by the head, nose or ears. To achieve the same effect in augmented reality, an occlusion mask is used. This is an object shaped roughly like a human head and placed inside the glasses. At runtime, the occlusion mask covers the object behind it but is not itself visible, achieving the desired effect. This is achieved using material settings for the occluder object.

The occluder object can be loaded separately or saved together with the 3D glasses model in a single 3D model file. The model *glasses8.fbx* in some of *VisageTrackerUnityDemo* sample projects contains both glasses and the occluder object in the same file; the material is found in *occluder_mat.mtl*. In the HTML5 *Virtual Eyewear Try-on* sample project, the occluder is found in *occlusion_mask.obj*. and the material in *o cclusion_mask.mtl*.

The next two images show the occlusion mesh highlighted as wireframe (left), and the final effect (right).

	
Occlusion final (wireframe)	Occlusion final

See also:

- API Reference FaceData featurePoints3DRelative
- Samples

I've seen the tiger mask in your demo - how I can build my own masks?

24 Jul 2020

Our sample projects give you a good starting point for implementing your own masks and other effects using powerful mainstream graphics applications (such as Blender, Photoshop, Unity 3D, and others). Specifically:

Windows: ShowcaseDemo, available as a sample project with full source code, includes a basic face mask (the tiger mask) effect. It is implemented by creating a face mesh during run-time, based on data provided by *VisageTracker* through *FaceData::faceModel** class members, applying a tiger texture and rendering it with OpenGL.

The mesh uses static texture coordinates so it is fairly simple to replace the texture image and use other themes instead of the tiger mask. We provide the texture image in a form that makes it fairly easy to create other textures in Photoshop and use these textures as a face mask. This is the ***template texture file (jk_300_textureTemplate.png)*** found in *visageSDK\Samples\data\* directory. You can simply create a texture image with facial features (mouth, nose, etc.) placed according to the template image, and use this texture instead of the tiger. You can modify texture in the ShowcaseDemo sample by changing the texture file which is set in line 331 of ShowcaseDemo.xaml.cs source file:

```
331: gVisageRendering.SetTextureImage(LoadImage(@"..
\Samples\OpenGL\data\ShowcaseDemo\tiger_texture.png"));
```

For a sample project based on Unity 3D, see Windows: VisageTrackerUnityDemo page. It includes the tiger mask effect and a 3D model (glasses) superimposed on the face. Unity 3D is an extremely powerful game/3D engine that gives you much more choices and freedom in implementing your effects, while starting from the basic ones provided in our project.

In VisageTrackerUnityDemo, the tiger face mask effect is achieved using the same principles as in ShowcaseDemo. Details of implementation can be found in Tracker.cs file (located in *visageSDK\Samples\Unity\VisageTrackerUnityDemo\Assets\Scripts\* directory) by searching for keyword "tiger".

See also:

- API Reference FaceData faceModelTextureCoords
- API Reference FaceData faceModelTextureCoordsStatic
- API Reference FaceData faceModelTriangleCount
- API Reference FaceData faceModelTriangles
- API Reference FaceData faceModelVertexCount
- API Reference FaceData faceModelVertices
- API Reference FaceData faceModelVerticesProjected

Troubleshooting

I am using visage|SDK FaceRecognition gallery with a large number of descriptors (100.000+) and it takes 2 minutes to load the gallery. What can I do about it?

05 Jun 2020

visage|SDK FaceRecognition simple gallery implementation was not designed for a use case with a large number of descriptors.

Use API functions that provide raw descriptor output (the function *VisageFaceRecognition::extractDescriptor()*) and descriptor similarity comparison function (*the function VisageFaceRecognition::descriptorsSimilarity()*) to implement your own gallery solution in the technology of your choice that is appropriate for your use case.

See also:

- API Reference VisageFaceRecognition ExtractDescriptor
- API Reference VisageFaceRecognition DescriptorsSimilarity

I want to change the camera resolution in Unity application VisageTrackerDemo. Is this supported and how can I do this?

05 Jun 2020

Depending on the platform, it’s already possible, out of the box.

On Windows and Android, the camera resolution can be changed via Tracker object properties **defaultCameraWidth** and **defaultCameraHeight** within the Unity Editor. When the default value -1 is used, the resolution is set to 800 x 600 in the native VisageTrackerUnityPlugin.

On iOS it's not possible to change the resolution out of the box from the demo application. The camera resolution is hard-coded to 480 x 600 within the native VisageTrackerUnityPlugin.

VisageTrackerUnityPlugin is provided with full source code within the package distribution.

I am getting EntryPointNotFoundException when attempting to run a Unity application in the Editor. Why does my Unity application does not work?

05 Jun 2020

Make sure that you had followed all the steps from the documentation Building and running Unity application.

Verify that the build target and visage|SDK platform match. For example, running *visage|SDK for Windows* inside the Unity Editor and for a Standalone application will work since both are run on the same platform. Attempting to run *visage|SDK for iOS* inside the Unity Editor on a MacOS will output an error because iOS architecture does not match MacOS architecture.

Errors concerning 'NSString' or other Foundation classes encountered in client project which includes iOS sample files (VisageRendering.cpp, etc.)?

02 Oct 2020

It is necessary to make sure that *VisageRendering.cpp* is compiled as an Objective-C++ source, and not as a C++ source by changing 'Type' property of the file on the right-hand property side in the Xcode editor. This applies to any source which includes/imports (directly or indirectly) any Apple classes.

Facial features data is not available when using Face Recognition

03 Jan 2022

This issue is related to licensing and relates to the license key file used. If the license is only for Face Recognition, and not for Face Tracking, then most of the face tracking outputs are not available.

Please note that the face detection and tracking algorithms are still fully functional because they are needed as the basis for Face Recognition - it is only the *access* to the face tracking outputs that is blocked if when Face Tracking is not licensed. More specifically, most of the facial features data in the FaceData structure is set to blank values if FaceTracking is not licensed.

Some of the basic information in FaceData structure is still available, even without the Face Track license:

- FaceData::trackingQuality
- FaceData::trackingQualityBdts
- FaceData::cameraFocus
- FaceData::frameRate
- FaceData::timeStamp
- FaceData::featurePoints2D (only feature point "12.1" is defined)
- FaceData::faceScale
- FaceData::faceRotationApparent

This is aimed at enabling the operations that may be required while performing face recognition, e.g. monitoring the tracking quality or displaying tracked/detected faces in a visual user interface.

Specifically, to visualize a bounding box around the face(s), it is recommended to use the feature point 12.1 from FaceData::featurePoints2D as the face center, and FaceData::faceScale for the size of the bounding box.

macOS: No valid license found, despite license being in correct location and initialization called

14 Apr 2021

If you are starting your own application from scratch and encountering the above problem, please note that by default, the working directory will be the root folder of the application, and not the resources folder where the license file is copied to. Hence the call fails to find the license key.

The solution is to change the working directory before calling license initialization. When using Swift, it can be done like this:

```
let fileManager = FileManager()
fileManager.changeCurrentDirectoryPath(Bundle.main.resourcePath!)
```

And for an Objective-C application:

```
NSString* resourcePath = [[NSBundle mainBundle] resourcePath];
NSFileManager *filemgr = [NSFileManager defaultManager];
[filemgr changeCurrentDirectoryPath:resourcePath];
```

Licensing Errors

17 Jul 2020

The following articles describe the various error codes produced by the Visage Technologies licensing system, and what may be done to remedy each error. They are valid for visage|SDK version 8.6b1 and higher.

Error code 0x00000001 (VS_ERROR_INVALID_LICENSE)

The error you received indicates that the license is not valid. The error occurs when the BundleID is not the same as the one in your application, or when the license that is registered to one product is used with another product. Please check if you’re using the correct license.

Error code 0x00000002 (VS_ERROR_EXPIRED_LICENSE)

The error you received indicates that the license has expired. Please consider renewing your license. Your Visage Technologies contact person may advise you on the options.

Error code 0x00000004 (VS_ERROR_EARLIER_VERSION_LICENSE)

The error you received indicates that the license version you are using is out of date, so you need to update to the newest available version. Please consider upgrading your license. Your Visage Technologies contact person may advise you on the options.

Error code 0x00000008 (VS_ERROR_MISSING_KEYFILE)

The error you received indicates that the application cannot locate the license key file. Please verify that the license key file is present in the folder used as a path to initialize the license manager. For more details, please follow the instructions from Documentation -> Licensing.

Error code 0x00000010 (VS_ERROR_NO_LICENSE)

The error you received indicates that there is currently no license available. Please follow the instructions from Documentation -> Licensing on how to correctly use the license key in your application.

Error code 0x00000020 (VS_ERROR_CORRUPT_VERSION_STRING)

The error you received indicates that the license key file was modified. Please try using a clean copy of the license key file that was sent to you, and see if the error still occurs.

Error code 0x00000040 (VS_ERROR_LICENSE_VALIDATION_FAILURE)

The error you received indicates that the server has rejected the license. Please get in touch with your Visage Technologies contact person, who will investigate the matter further and get back to you.

Error code 0x00000080 (VS_ERROR_NC_CONNECTION_FAILED)

The error you received indicates that there was a failure in connection to our licensing server for registering your license. Please make sure that the device on which you are using the software is connected to the internet.

Error code 0x00000100 (VS_ERROR_TEMPERED_KEYFILE)

The error you received indicates that the license key file was modified. Please try using a clean copy of the license key file that was sent to you, and see if the error still occurs.

Error code 0x00000200 (VS_ERROR_TEMPERED_KEYSTRING)

The error you received indicates that the license key file was modified. Please try using a clean copy of the license key file that was sent to you, and see if the error still occurs.

Error code 0x00000400 (VS_ERROR_TEMPERED_DATE)

The error you received indicates that the date on the computer was modified. Please make sure that the date on the computer is correct.

Error code 0x00000800 (VS_ERROR_UNREADABLE_KEYSTRING)

The error you received indicates that the license key file was modified. Please try using a clean copy of the license key file that was sent to you, and see if the error still occurs.

Error code 0x00001000 (VS_ERROR_INVALID_OS)

The error you received indicates that your license key does not support the platform (operating system) on which you are trying to use it.

Error code 0x00002000 (VS_ERROR_INVALID_URL)

The error you received indicates that the license was issued for a different domain (URL) than the one on which it is used. Please confirm that the domain name for which you have licensed visage|SDK matches the domain (URL) on which you are using it.