

顔認証技術

SDK/インターフェース仕様書 (ver.1.0.0)



株式会社スワローインキュベート

2026年10月02日

■はじめに

顔認証技術 SDKは、株式会社スワローインキュベートが提供しています。

本書に基づき、当 SDKをご利用いただく前に、以下のご注意事項を十分に読んだ上で、ご利用いただきますようお願いいたします。

■ご注意事項

- ・本書は、予告なしに変更されることがあります。
- ・本書を無断で、複製、転用、公衆送信、貸与等を行わないようお願いします。
- ・SDKをご利用いただくには、あらかじめ当社利用規約に同意いただく必要があります。
詳しくは営業担当までお問い合わせください。

お問い合わせ

株式会社スワローインキュベート

顔認証技術 テクニカルサポート窓口

MAIL: support@swallow-incubate.com

■SDK更新履歴

バージョン	日時	変更内容サマリー
ver.1.0.0	2026年10月02日	初版

このSDKでできること

■このSDKでできること

顔認証技術SDKでは、OpenCVに含まれる顔検出モデル「YuNet」および顔認識モデル「SFace」というオープンソース技術を用いて、入力された画像から顔を検出して特徴量（128次元）を抽出し、本人かどうかを判定する「1:1認証」と、入力された顔が登録済みの顔のうち誰にあたるかを判定する「1:N認証」を行うことができます。

1:1認証 verify()

この人は本人か？



登録済みの1人と照合し、類似度がしきい値以上なら本人と判定します。

1:N認証 identify()

この人は誰か？



登録ユーザ全員と照合し、最も似ている人を特定します（該当なしは-1）。

📄 特徴量は512バイト固定。
DBやファイルにそのまま保存

🔒 外部通信なし・CPUのみで動作

🔒 なりすまし判定SDKと併用可能

ご利用にあたって

■ご利用環境

現在のバージョンでは、以下のご利用環境に対応しています。

項目	内容	
インターフェース	C++言語 (C++11以降) / C#言語 (.NET 8 推奨)	
対応OS	Windows OS / Linux OS / macOS iOS / Android OS	
CPUアーキテクチャ	64bit系 (x86_64 / Arm64) ※Android は32bit系 (armeabi-v7a / x86) にも対応	
推奨メモリ	2GB以上を推奨	
依存ライブラリ	OpenCV 4.10 系 (SDKに同梱) ※Windowsは Microsoft Visual C++ 2015-2022 再頒布可能パッケージ (x64) が必要です	
実行環境 / ビルド環境	Windows	Windows 10 / 11 (64-bit)・Windows Server 2019 / 2022
	Linux OS	Ubuntu 22.04以降 (x86_64 / aarch64)
		Amazon Linux 2023 (x86_64 / aarch64)
	iOS	iOS 15以降
	Android	API 28以上 (Android 9.0)
	macOS	macOS 13 (Ventura) 以降 (Apple Silicon)

※その他の環境でのご利用を希望される場合は、お問い合わせください。

■推奨入力画像

現在のバージョンでは、以下の入力画像を推奨しています。

項目	内容
画像カラー仕様	BGRカラー画像 (8bit3ch: CV_8UC3 または 8bit4ch: CV_8UC4) ※RGB565・グレースケールは非対応です
センサ種別	可視光センサ画像のみ (赤外線センサ画像には対応していません)
推奨フレームサイズ	VGA (640 x 480) サイズ以上
入力画像解像度	<検出される顔領域のピクセル数> 最低 width 80px 以上 推奨 (上限は既定値 1000px)
顔領域の明るさ	顔領域 平均輝度値 50.0 以上 (8bit256階調)
撮影距離	<カメラから顔までの距離> ～1m程度まで ※入力画像解像度を上げることで距離を伸ばすことも可能です。顔領域最低サイズを参考にしてください。

※1フレームにつき1人の顔が写っていることを前提とします

(複数の顔を検出した場合はエラーとなります)。

※その他の入力画像でのご利用を希望される場合は、お問い合わせください。

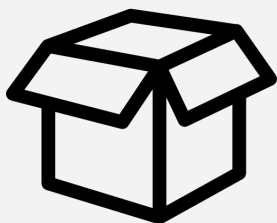
SDK構成

■SDK構成



interfaceファイル
(hpp)

※C#版の場合はラッパーファイルが含まれます。



動的リンクライブラリ
(dll/so/dylibなど)

顔認証ライブラリ



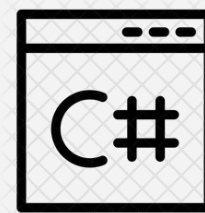
interfaceファイル
(hpp)

※C#版の場合はラッパーファイルが含まれます。



動的リンクライブラリ
(dll/so/dylibなど)

OpenCVライブラリ



C++コード / ビルド手順
C#コード / ビルド手順

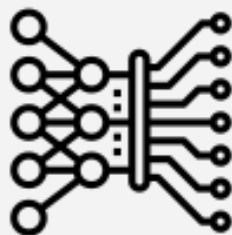


顔認証 サンプルアプリ



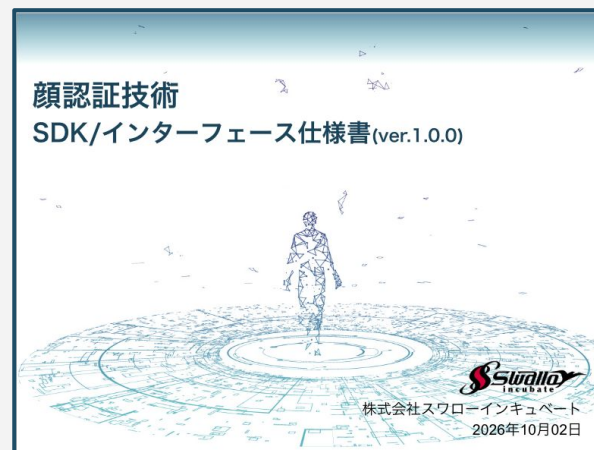
期限付きトークン

アクティベーション トークン



ONNXファイル(2点)

深層学習 モデル



ご利用マニュアル (本書)

■SDK構成

本SDKは動的リンクライブラリとそのインターフェースであるヘッダーファイルで構成されています。C++インターフェースとなっていますが、Windows C#版には、C API経由でC#から呼び出すラッパー(P/Invoke)とサンプルを用意しています。

スマホOSには、C++言語向けのラッパーサンプルコードを用意しています。

OS	提供物
macOS	FaceAuth.hpp (インターフェースファイル) libFaceAuth.dylib (Apple Silicon) サンプルアプリ (C++)
Windows OS	FaceAuth.hpp FaceAuth.dll (実行時参照ライブラリ) FaceAuth.lib (ビルド時取込ライブラリ) サンプルアプリ (C++ / C#)・C#ラッパー (FaceAuth.cs)
各種Linux OS (Ubuntu / Amazon Linux)	FaceAuth.hpp (インターフェースファイル) libFaceAuth.so サンプルアプリ (C++)
iOS	FaceAuth.Framework (FaceAuth.hpp含む) サンプルアプリ (Objective-C ラッパーサンプルコード込み) iOS向けOpenCVビルド済ライブラリ
Android OS	FaceAuth.hpp libFaceAuth.so (arm64-v8a / armeabi-v7a / x86 / x86_64) Android向けOpenCVビルド済ライブラリ サンプルアプリ (JNI ラッパーサンプルコード込み)

顔認証 - 1:1 認証モード

■顔認証 - 1:1認証 (verify)

登録時に抽出した顔の特徴量 (128次元・512バイト)と、照合時に入力された画像から抽出した特徴量とコサイン類似度を算出し、しきい値 (既定値 0.363) 以上であれば本人と判定します。

特徴量の抽出は、所定サイズ (初期値80px)以上の顔が検出できた場合のみ行われ、所定サイズ未満の場合は抽出されません。

抽出した特徴量はバイト列 (payload)としてファイルやデータベース等に保存し、setPayload()で復元して照合に用いることができます。



コサイン類似度 (-1.0~1.0) と判定結果



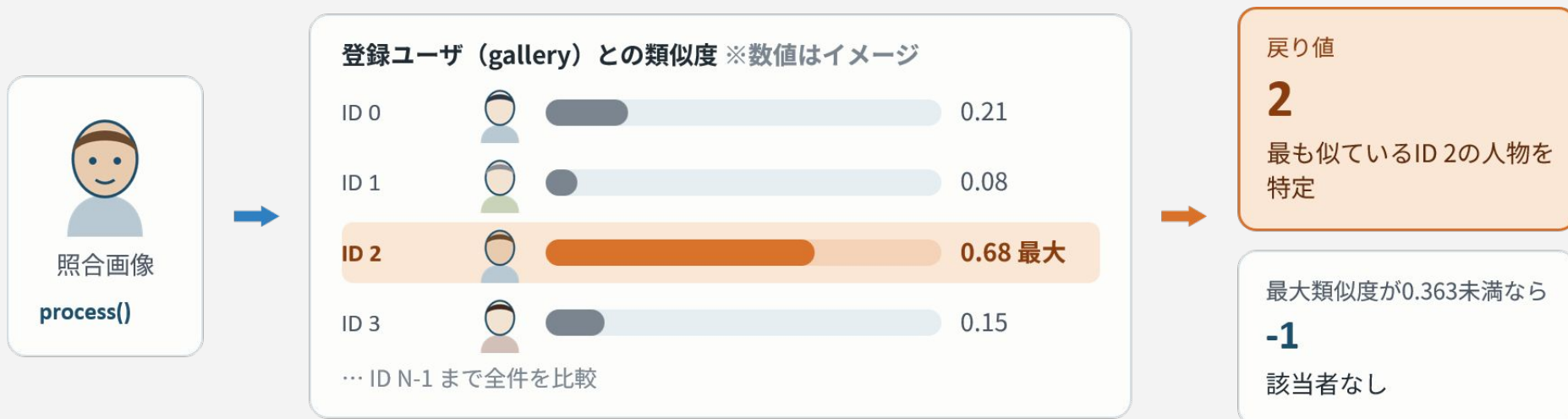
顔認証 – 1:N認証モード

■顔認証 - 1:N認証 (identify)

照合時に入力された画像から抽出した特徴量を、登録済みの複数ユーザの特徴量(ギャラリー)と全件比較し、コサイン類似度が最も高いユーザのインデックスを返します。

最も高い類似度がしきい値(既定値 0.363)未満の場合は、該当者なしとして -1 を返します。

参考値として、登録ユーザ 10,000 件との照合は約5ms(macOS / Apple Silicon)で完了します。

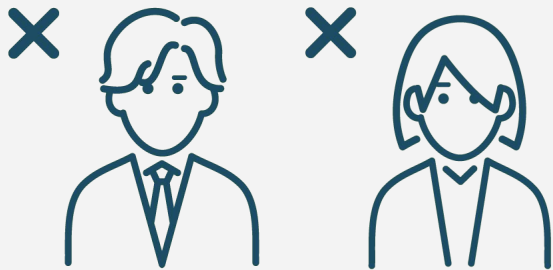


⚡ 登録ユーザ **10,000件** との照合が **約5ms** (macOS / Apple Silicon での実測値)

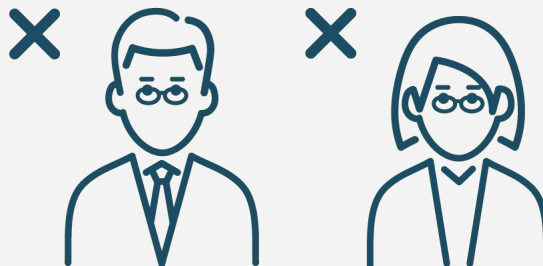
ライブラリ仕様

■ライブ러리仕様 - ✕ 検出不可となるケース

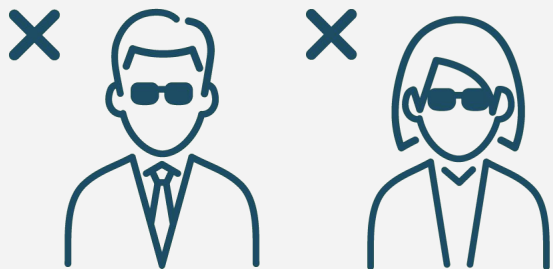
顔の検出・特徴量の抽出においては、以下のケースで検出エラーや照合精度の低下が起きやすくなりますのでご注意ください。



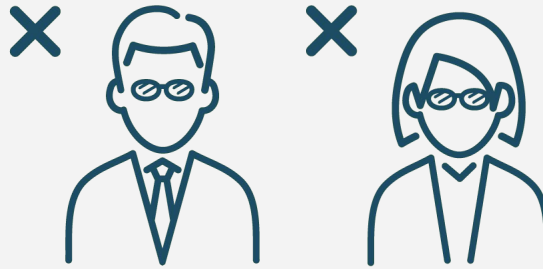
髪の毛が目にかかっている



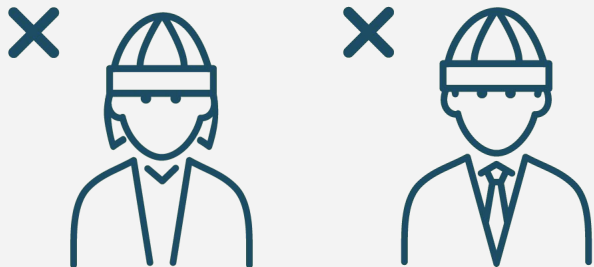
メガネフレームが目にかかっている



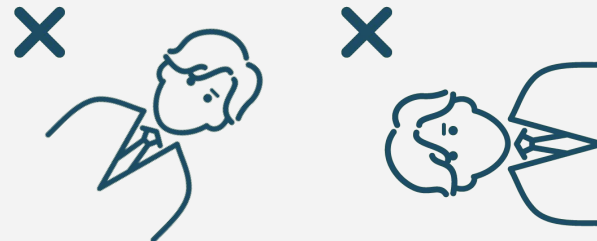
レンズが黒いサングラス



外光の映り込みが激しい



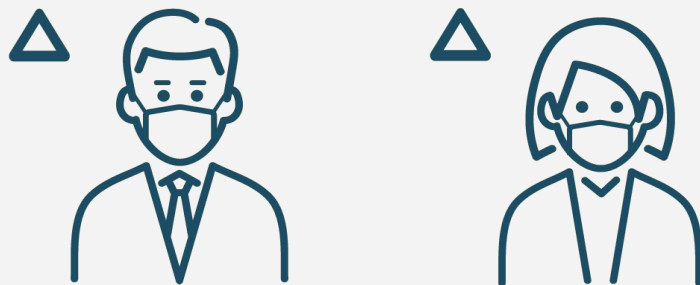
帽子を深くかぶっている



顔が傾いて撮影されている

■ 라이브러리仕様 - △ 検出に影響を与える場合があるケース

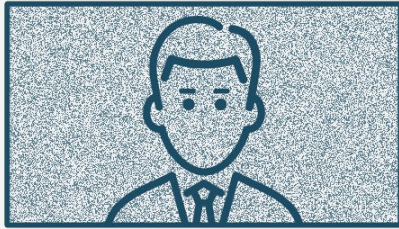
顔の位置検出においては、以下のケースで検出に影響を与えやすくなりますのでご注意ください。



マスク
(照合精度が低下する場合あり)

■ライブ러리仕様 - △ 検出に影響を与える場合がある撮影状況

各種検出・判定においては、以下の撮影の状況では検出に影響を与えやすくなりますのでご注意ください。



×露光が不十分
(環境光の差に影響を受けます)



×オートフォーカス制御
(フォーカシング中検出エラーになりやすい)



×カメラの設置が
まっすぐではない



×画角から外れている
(鼻より上しか写っていない)



×画角から外れている
(片目しか写っていない)

■ライブラリ仕様 - 機能一覧

項目	機能項目	要件
基本機能	アクティベーション機能	FaceAuthオブジェクトのプロセス立ち上げごとに、USBまたはトークンによるアクティベーションを行います。
	モデルファイルの読み込み	弊社で用意した学習モデルファイル(顔検出:YuNet / 顔特徴量:SFace の2ファイル)を読み込みます。
	パラメータ調整	FaceAuthオブジェクトのinit()に構造体を読み込ませることで所定のパラメータ調整を行うことができます。
	画像データの読み込み	FaceAuthクラスにて、画像データを読み込ませることができます。
各種情報 センシング機能	顔位置検出機能	入力された画像から、顔領域および顔の特徴点(両目・鼻先・両口角の5点)を検出します。外部で検出した顔矩形を指定することも可能です。
	複数顔検出機能	入力された画像に複数の顔が検出された場合はエラーを返し、特徴量の抽出を行いません。
	顔特徴量抽出機能	検出された顔を整列(アライメント)したうえで解析し、128次元の特徴量(512バイト)を抽出します。
	1:1認証機能	抽出した特徴量と、指定したユーザの特徴量とのコサイン類似度を算出し、しきい値以上であれば本人と判定します。
	1:N認証機能	抽出した特徴量と、登録ユーザ全員の特徴量を比較し、類似度が最も高いユーザを特定します(しきい値未満の場合は該当者なし)。

■ライブラリ仕様 - 機能別処理速度目安

現在のバージョンでは、ライブラリ各機能ごとの処理時間は以下の通りとなります。ライブラリは GPUを使用せず、CPUのみで動作します。process()に画像を読み込ませると、顔位置検出から特徴量抽出までを順に実行します。照合(1:1 / 1:N)は、抽出した特徴量に対して別途関数を呼び出して行います。また、顔位置検出においては、本ライブラリ以外のツール(なりすまし判定 SDK等)で検出した結果を指定して処理を進めることも可能です。

機能	On/Off	アルゴリズム	出力	処理時間※
顔位置検出	必須 (位置指定可)	CNN系物体検出 (YuNet)	cv::Rect型(x, y, width, height) + 特徴点 5点	約16ms程度
顔特徴量抽出	必須	CNN系特徴量抽出 (SFace)	128次元特徴量(512バイト)	
1:1認証	任意 (関数呼出)	コサイン類似度	類似度 float(-1.0~1.0) 判定結果 bool	1ms程度
1:N認証	任意 (関数呼出)	コサイン類似度(全 件比較)	インデックス int(該当なしは -1)	約5ms程度 (N = 10,000)

※処理時間は、macOS(Apple Silicon)・入力解像度 1280×960 での実測値をもとに算出しています

■ライブラリ仕様 - クラス構成

クラス名	データ型		メンバ名	説明
struct ExternalFilePath	この構造体について			顔認証を行うためのモデルファイルを読み込ませるために、FaceAuthクラスのオブジェクト化直後に、本構造体をinit()関数の第一引数として与えます。
	顔位置検出用	std::string	faceDetectModel	顔位置検出に用いるモデルファイル (faceDetect.onnx: YuNet) のパスを指定する変数です。
	顔特徴量抽出用	std::string	faceFeatureModel	顔特徴量の抽出に用いるモデルファイル (faceFeature.onnx: SFace) のパスを指定する変数です。

■ライブラリ仕様 - クラス構成

クラス名	機能	データ型	メンバ名	説明
struct Params	この構造体について			各検出パラメータの値を読み込ませるために、FaceAuthクラスのオブジェクト化後、本構造体をinit()関数の第二引数として与えます。省略した場合、および各メンバに0を指定した場合は既定値が適用されます。
	顔位置検出用	int	minFaceWidth	顔位置検出を行う最小顔横幅サイズ(px)を指定します。既定値は80pxで、80px未満は指定できません。
		int	maxFaceWidth	顔位置検出を行う最大顔横幅サイズ(px)を指定します。既定値は1000pxです。
		float	faceAreaMinRatio	顔の接近度を判定するため、「顔位置検出サイズ面積・入力画像サイズ面積」により算出される顔サイズの許容する下限・上限割合を指定します0.05～1.00の値の範囲で指定します。既定値は下限0.1・上限0.65です。
		float	faceAreaMaxRatio	
		bool	edgePosErrMode	検出された顔位置が上下左右端(画像端から3%以内)に位置する場合に、エラーとして以降の処理を停止するかどうかを真偽値で指定します。既定値はfalseです。
	顔位置検出用	float	faceDetectScoreTh	顔位置検出の信頼度(スコア)のしきい値です。0.0～1.0の値の範囲で指定します。既定値は0.6です。

■ライブラリ仕様 - クラス構成

●FaceFeatureクラス - 顔検出関連メンバ

クラス名	データ型	メンバ名	説明
class FaceFeature	このクラスについて		顔検出および特徴量の抽出結果を格納するクラスです。処理途中での検出エラー時でも、検出できた部分までのメンバは取得可能です。 std::make_shared<FaceFeature>()で生成し、registerFaceFeature()でFaceAuthクラスに登録します。
	cv::Mat	originallmg	入力した元画像(3chに正規化したコピー)を取得可能です。
	cv::Mat	checklmg	入力した元画像に、検出結果(顔矩形・特徴点・処理結果)を描画プロットした画像です。
	cv::Mat	faceRectlmg	入力した元画像から、検出された顔画像のみを切り出した画像を取得可能です。
	cv::Rect	faceRectArea	検出された顔矩形領域情報を取得可能です。(x座標,y座標,width,height)
	float	faceBrightness	検出された顔矩形領域の平均輝度値を8bit256階調で取得可能です。
	bool	isValidFace	検出された顔において、顔のサイズ、フレーム面積と顔面積の割合、検出位置、顔の明るさが適正かどうかを判定します。
	std::vector<cv::Rect>	vFaceRect	検出したすべての顔領域情報を取得可能です(先頭が採用された顔)。(x座標,y座標,width,height)
	std::string	msg	処理結果メッセージを取得することができます。
	void	clear()	FaceFeatureクラスの全メンバ変数の値を初期化します(特徴量もゼロクリアされます)。

■ 라이브러리仕様 - クラス構成

● FaceFeatureクラス - 顔検出関連メンバ

クラス名	データ型	メンバ名	説明
class FaceFeature	float	faceDetectScore	顔位置検出の信頼度(スコア)です。0.0～1.0の値で出力されます。
	std::vector <cv::Point2f>	faceLandmarks	検出された顔の特徴点5点([0]右目・[1]左目・[2]鼻先・[3]右口角・[4]左口角／被写体基準・原画像座標)です。検出成功時のみ5要素が格納されます。
	cv::Mat	alignedFaceImg	特徴量の抽出に用いた、整列(アライメント)済みの顔画像(112×112・BGR)です。特徴量の抽出成功時のみ格納されます。
	const unsigned char*	payload	抽出された顔の特徴量です(128次元 × float32 リトルエンディアン=512バイト固定)。データベース等にそのまま保存できます。hasFeatureがfalseの場合は全て0となります。
	const unsigned int	payloadLength	payloadのバイト長です。常に512となります。
	bool	hasFeature	特徴量を保持しているかどうかの真偽値です。process()の成功、またはsetPayload()の成功でtrueとなります。
	bool	setPayload()	保存済みの特徴量(512バイト)を復元するための関数です。長さ不一致・非有限値を含む・零ベクトルの場合はfalseを返します。
	float	compare()	他のFaceFeatureとのコサイン類似度(-1.0～1.0)を返します。値が高いほど同一人物である可能性が高くなります。どちらかに特徴量がない場合は例外を送出します。
	bool	verify()	1:1認証を行う関数です。コサイン類似度がしきい値(既定値 0.363)以上であればtrueを返します。引数で類似度を受け取ることもできます。
	int	identify()	1:N認証を行う関数です。登録ユーザの特徴量リスト(gallery)の中で類似度が最も高いユーザのインデックスを返します。最大類似度がしきい値(既定値 0.363)未満の場合は-1を返します。

■ライブラリ仕様 - クラス構成

クラス名	データ型	メンバ名	説明
class FaceAuth	このクラスについて		顔の検出および特徴量の抽出を行うための実行クラスとなります。
	bool	init	初期化を行うメンバ関数です。第一引数にExternalFilePath構造体、第二引数にParams構造体(省略可)をとります。アクティベーション前に実行すると例外を送出します。
	bool	registerFaceFeature	メモリ確保したFaceFeatureクラス(std::shared_ptr)を本クラスに登録するための関数です。
	bool	process	cv::Matを引数とし、顔位置検出・特徴量抽出を行う関数です。第二引数に顔矩形(cv::Rect)を指定すると、外部で検出した顔領域をもとに処理を行います。検出結果はすべてFaceFeatureクラスに格納されます。

■ライブラリ仕様 - クラス構成

◆ USBアクティベーション方式

クラス名	データ型	メンバ名	説明
class ActivationChallenge	このクラスについて		FaceAuthクラスを実行する前にアクティベーションを行うためのクラスとなります。
	std::string	config()	アクティベーション情報および本ライブラリの情報を取得可能です。
	bool	checkUSB()	USB dongleがマシンに接続されているかをチェックするための関数です。
	bool	validUSB()	USB dongleとライブラリに埋め込まれたIDとが一致するかをチェックするための関数です。本関数からtrueが返ることによりFaceAuthクラスの初期化処理であるinit()関数が成功するようになります。

◆ トークンアクティベーション方式

クラス名	データ型	メンバ名	説明
class ActivationChallenge	std::string	config()	アクティベーション情報および本ライブラリの情報を取得可能です。
	bool	checkToken()	本関数の引数にとるトークン文字列が、ライブラリに埋め込まれた対応トークンとマッチするかどうかのみを判定します。 <u>本関数からtrueが返るだけではアクティベーションは完了しません。</u>
	bool	validToken()	本関数の引数にとるトークン文字列が、ライブラリに埋め込まれた対応するトークン情報を読み取り、有効期限内かどうかをチェックするための関数です。有効期限内であれば本関数からtrueが返り、FaceAuthクラスの初期化処理であるinit()関数が成功するようになります。
	long	getExpDate()	本関数の引数にとるトークン文字列が、ライブラリに埋め込まれた対応するトークン情報を読み取り、有効期限をlong型の数値で返します。例えば返り値が「20271231」であれば、2027年12月31日まで有効なトークンです。

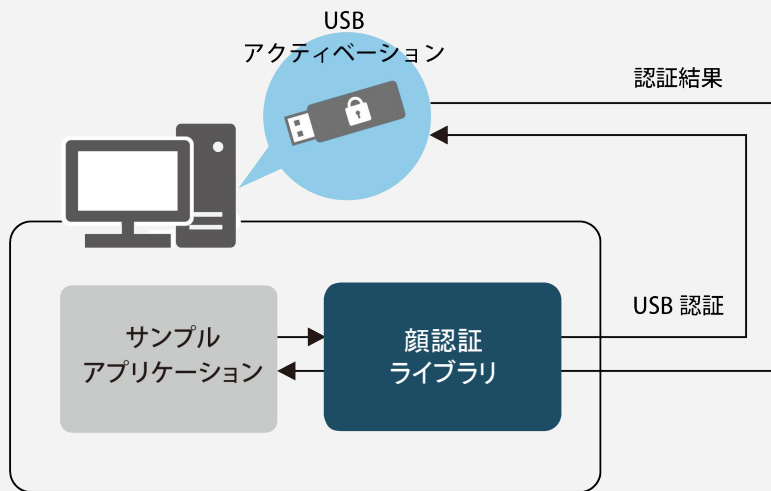
■ライブラリ仕様 - ファクトリ関数

返り値	関数名	説明
FaceAuth*	newFaceAuth()	FaceAuthクラスをオブジェクト化させるための関数です。 内部で派生クラスの生成などが行われるため、FaceAuthクラスのオブジェクト化は、必ずこの関数を用いて行ってください。
void	releaseFaceAuth()	FaceAuthオブジェクトをメモリ開放させるための関数です。 FaceAuthオブジェクトが不使用になった場合は、必ずこの関数を実行してください。
ActivationChallenge*	newFaceAuthActivationChallenge()	ActivationChallengeクラスをオブジェクト化させるための関数です。 内部で派生クラスの生成などが行われるため、ActivationChallengeクラスのオブジェクト化は、必ずこの関数を用いて行ってください。 ※なりすまし判定SDKとの併用時に関数名が衝突しないよう、「FaceAuth」を含む名称としています。
void	releaseFaceAuthActivationChallenge()	ActivationChallengeオブジェクトをメモリ開放させるための関数です。 ActivationChallengeオブジェクトが不使用になった場合は、必ずこの関数を実行してください。

■ライブラリ仕様 - アクティベーション

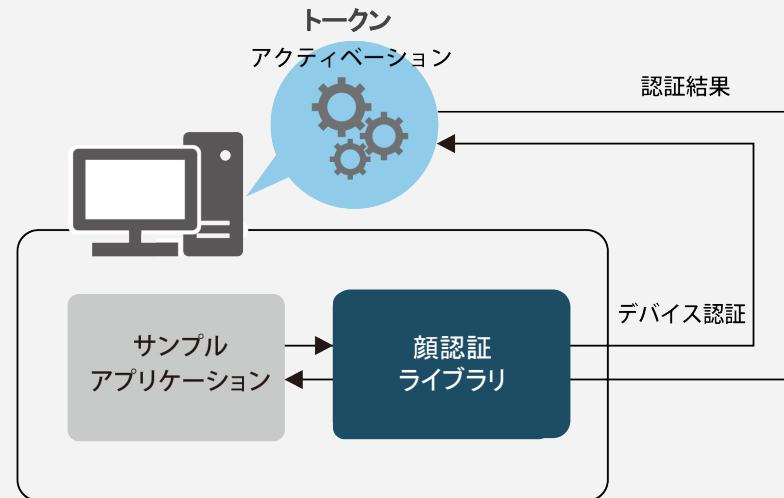
各種ライセンスでは、本ライブラリをご利用いただくにあたって、アクティベーションが必要となります。アクティベーションには、USB方式と有効期限を利用したトークン方式の2タイプがあります。

◆ USBアクティベーション



有償

◆ トークンアクティベーション



無償

■ライブラリ仕様 - アクティベーション情報

ActivationChallengeオブジェクトのconfig()を使用することで、返り値に、アクティベーション情報やライブラリ基本情報を取得することができます。弊社にお問い合わせいただく場合に取得をお願いする場合があります。

◆トークンアクティベーション版 config() 実行結果例

```
[toshikazuohno@sample]$ ./get_config
Activation   : Token
Client ID    : 9589959782
Client Name  : Swallow Incubate Co., Ltd.
SDK Version  : FaceAuthSDK - ver.1.0.0
Build Date   : 2026-09-01
[toshikazuohno@sample]$
```

出力データ詳細

■message一覧①

FaceFeature型のメンバ変数であるmsgにて取得できる主なメッセージは以下のいずれかとなります。その他のエラーが発生した場合はお問合せください。

メッセージ内容	内容
Process Successful	全ての処理が完了しました(特徴量の抽出に成功)
Error: Input RGB image(24bit)	BGR画像(8bit 3ch)またはBGRA画像(8bit 4ch)を入力してください
Error: Failed to detect face(1001)	顔位置検出に失敗しました (顔が存在しない)
Error: Failed to detect face(1002) - width: ...px / minFaceWidth: ...px	顔位置検出に失敗しました (横幅サイズ不足)
Error: Failed to detect face(1003) - width: ...px / maxFaceWidth: ...px	顔位置検出に失敗しました (横幅サイズ超過)
Error: Multiple faces detected(1101)	【警告】複数の顔が検出されました
Error: Failed to detect face landmarks in faceRect(1102)	指定した顔矩形内で顔の特徴点を検出できませんでした
Error: Failed to align face(1201)	顔の整列(アライメント)に失敗しました

■message一覧②

FaceFeature型のメンバ変数であるmsgにて取得できる主なメッセージは以下のいずれかとなります。その他のエラーが発生した場合はお問合せください。

メッセージ内容	内容
Error: Face position error(1004) - Too far	顔がカメラから設定値より離れすぎています(接近度判定エラー: faceAreaMinRatio 未満)
Error: Face position error(1005) - Too close	顔がカメラに設定値より接近しすぎています(接近度判定エラー: faceAreaMaxRatio 超過)
Error: Face position error(1006) - Too far	顔がカメラから規定値より離れすぎています(縦長画像: 顔面積比 0.12 未満)
Error: Face position error(1007) - Too close	顔がカメラに規定値より接近しすぎています(縦長画像: 顔面積比 0.50 超過)
Error: Face position error(1008) - Too far	顔がカメラから規定値より離れすぎています(横長画像: 顔面積比 0.05 未満)
Error: Face position error(1011) - move to the center	顔が上下左右端位置にあります(左端)
Error: Face position error(1012) - move to the center	顔が上下左右端位置にあります(上端)
Error: Face position error(1013) - move to the center	顔が上下左右端位置にあります(右端)
Error: Face position error(1014) - move to the center	顔が上下左右端位置にあります(下端)
Error: Lack of face area brightness(1021)	顔領域の明るさ不足です(顔領域平均輝度値 50以上 - 8bit256階調)

■message一覧③ / 例外一覧

msgにて取得できるメッセージに加え、引数の誤りや呼び出し順の誤りなどは例外 (const char*)として送出されます。その他のエラーが発生した場合はお問合せください。

メッセージ内容	内容
Error: Failed to extract face feature(1202)	特徴量の抽出に失敗しました
Activation error(should be activated firstly)	【例外】init()実行前に、アクティベーションを行ってください
Error: init() first	【例外】process()実行前に、init()を実行してください
Error: registerFaceFeature() first	【例外】process()実行前に、registerFaceFeature()を実行してください
Error: Empty input image	【例外】入力画像が存在しません
Error: compare() on empty feature(1301)	【例外】特徴量を持たない FaceFeature で照合を行おうとしました
Error: identify() gallery contains empty feature(1302)	【例外】照合対象リストに特徴量を持たない要素が含まれています