まばたき検出技術
SDK/インターフェース仕様書(ver.1.0.1)

ディープラーニング対応版

株式会社スワローインキュベート

2023年09月15日

■はじめに

まばたき検出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

まばたき検出技術　テクニカルサポート窓口

MAIL: support@swallow-incubate.com

■更新履歴

バージョン	日時	変更内容サマリー
ver.1.0.0	2023年03月20日	初版
ver.1.0.1	2023年09月15日	OpenCVリンクの変更(4.5.5=>4.7.0)、その他微修正

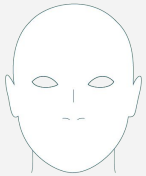
このSDKでできること

■このSDKでできること

まばたき検出技術SDKでは、ディープラーニングによる学習モデルを用いて、顔・目の位置を検出し、それらの情報を元に「まばたきの検出」を行うことが可能です。これらの検出・判定処理を、 WEBカメラなどでも可能なことが特徴となります。

DL = 深層学習ベース

特徴点検出



DL
顔位置検出

DL
目位置検出

DL
メガネ・サングラス判定

まばたき検出

DL
目の開閉検出

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)	
対応OS	Windows OS / Linux OS / macOS / Raspberry Pi OS iOS / Android OS	
CPUアーキテクチャ	64bit系（x86_64 / Arm64） ※一部32bit系にも対応します	
推奨メモリ	1GB以上を推奨	
依存ライブラリ	OpenCV 4.7.0 / OpenCV Contrib 4.7.0	
実行環境 / ビルド環境	Linux kernel 4.9以降	gccを推奨
	macOS 11以降 推奨	
	Raspberry Pi OS以降 推奨	
	Windows 10 以降 64bit系	MicroSoft Visual C++コンパイラを推奨
	iOS 15以降	最新のXcodeの利用を推奨
	Android OS 7.0(API 24)以降	最新のAndroid Studio / Gradle / NDK の利用を推奨

※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

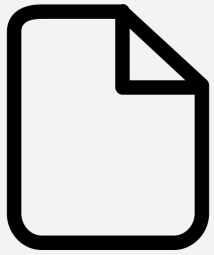
現在のバージョンでは、以下の入力画像を推奨しています。

項目	内容
画像カラー仕様	**RGB形式画像** (8bit3ch または 8bit4ch) ※RGB565は非対応です
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 100px 以上 推奨**
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。
検出可能人数	**1名を想定**

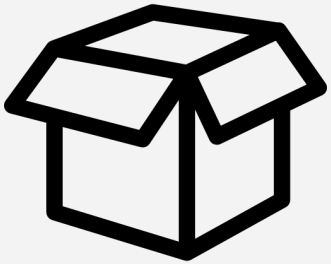
※その他の入力画像でのご利用を希望される場合は、お問い合わせください。

SDK構成

■SDK構成



interfaceファイル
(hpp)



動的リンクライブラリ
(dll/so/dylibなど)

C++コード + ビルド手順

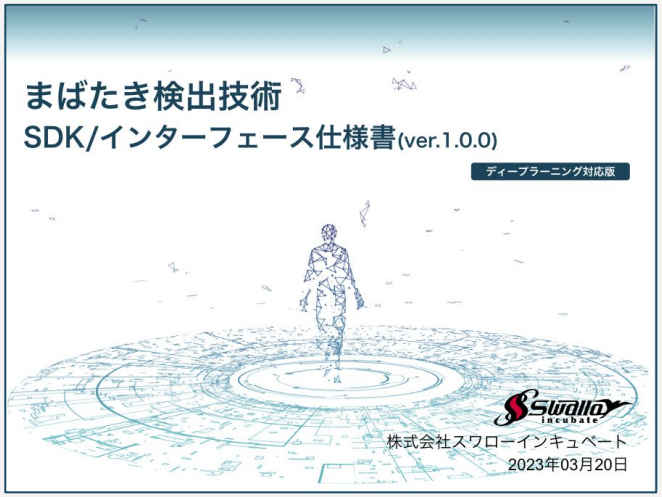
まばたき検出ライブラリ

まばたき検出サンプルアプリ

期限付きトークン

USBドングル

アクティベーションツール
(いずれか1つ)



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、
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	Blink.hpp （インターフェースファイル） libBlink.dylib サンプルアプリ（C++）
Windows OS	Blink.hpp Blink.dll (実行時参照ライブラリ) Blink.lib (ビルド時取込ライブラリ) サンプルアプリ（C++）
各種Linux OS **(Raspbian OS含む)**	Blink.hpp（インターフェースファイル） libBlink.so サンプルアプリ（C++）
iOS	Blink.Framework (Blink.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 + OpenCV Contribビルド済ライブラリ
Android OS	Blink.hpp libBlink.so（arm64-v8a / armeabi-v7a / x86 / x86_64） サンプルアプリ（JNI ラッパーサンプルコード込み） Android向けOpenCV + OpenCV Contribビルド済ライブラリ

カメラについて

■カメラについて

■必要となる撮影スペック

項目	要件
カメラ解像度	VGAサイズ(640 x 480)以上を推奨 (640 x 480 / 1280 × 720 / 1280 x 960 / 1920 x 1080 など)
撮影距離	30cm～1m程度まで

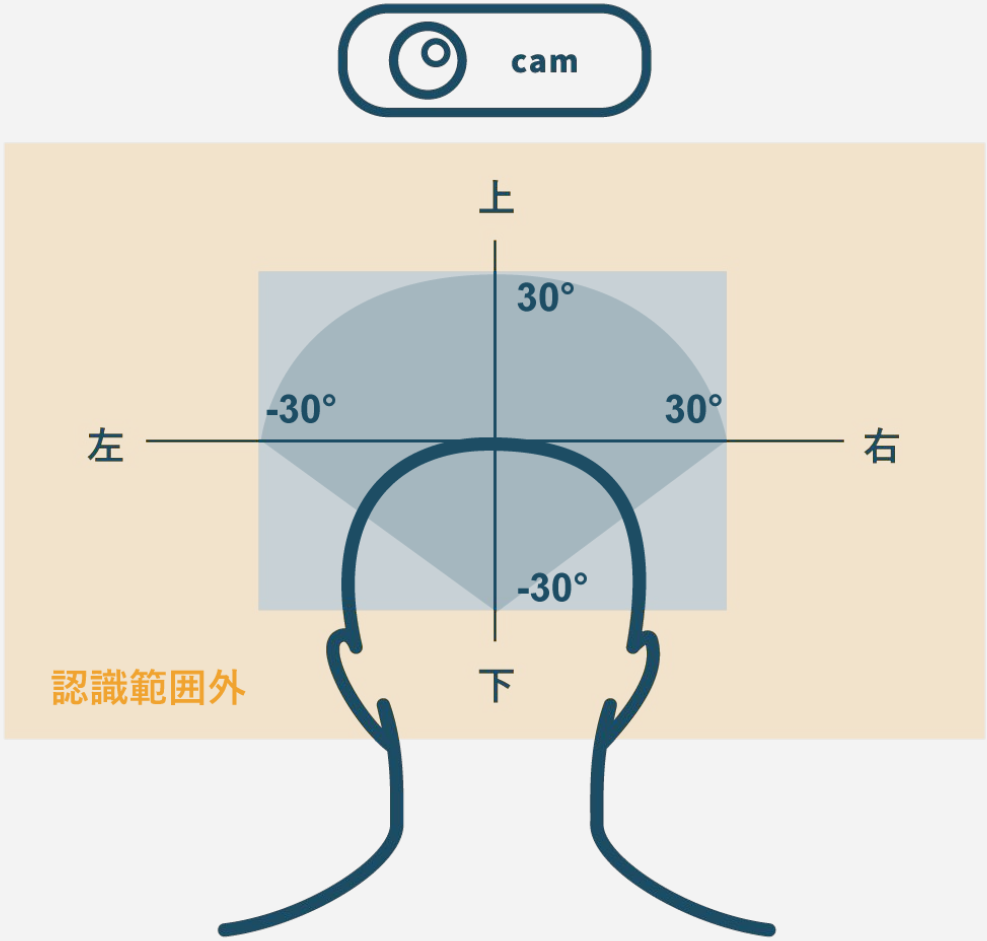
■必要となる撮影要件

項目	要件
顔横幅サイズ	最低100px以上
顔領域明るさ	平均輝度値 50.0以上（8bit256階調）

ライブラリ仕様

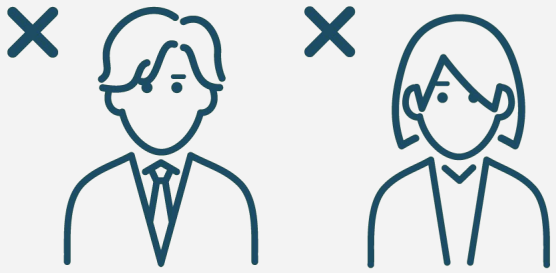
■ライブラリ仕様 - 検出可能画角

現在のバージョンでは、検出可能な画角は、上下左右ともに30°程度ですが
顔の形状などによるため、個人差があります。

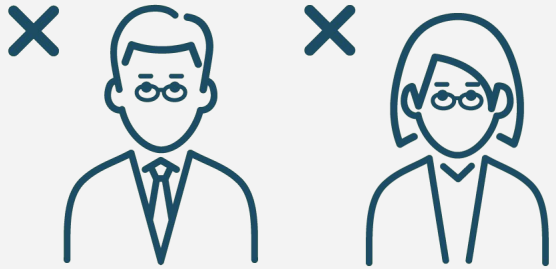


■ライブラリ仕様 - X 検出不可となるケース

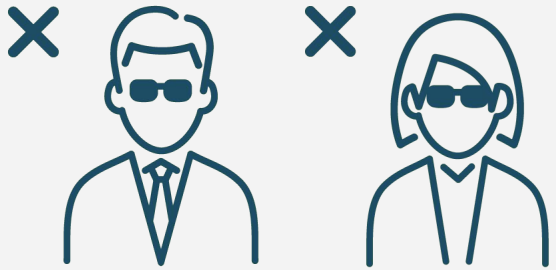
現在のバージョンでは、以下のケースで検出エラーになりやすくなります。



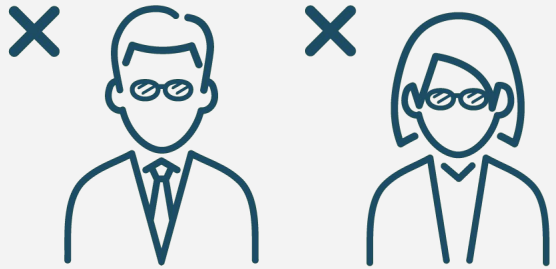
髪の毛が目にかかっている



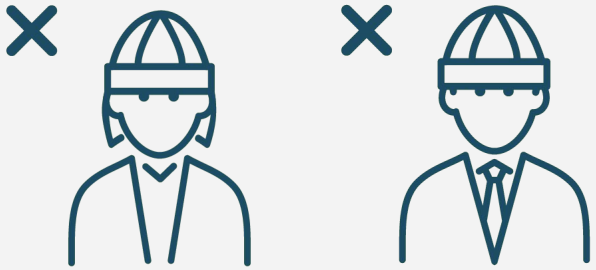
メガネフレームが目にかかっている



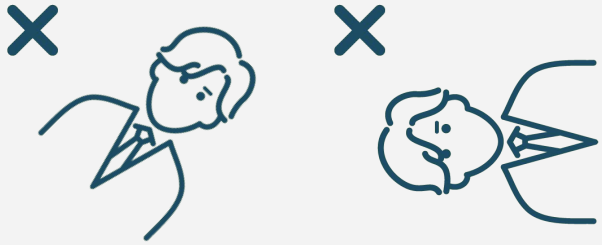
レンズが黒いサングラス



外光の映り込みが激しい



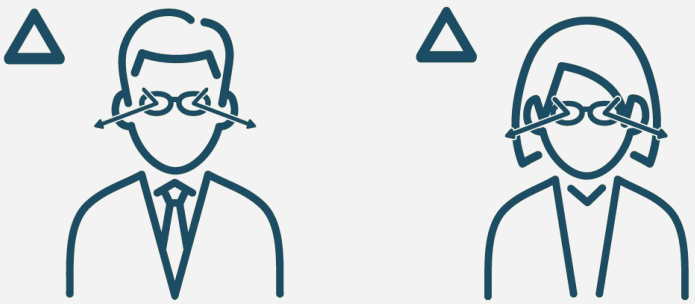
帽子を深くかぶっている



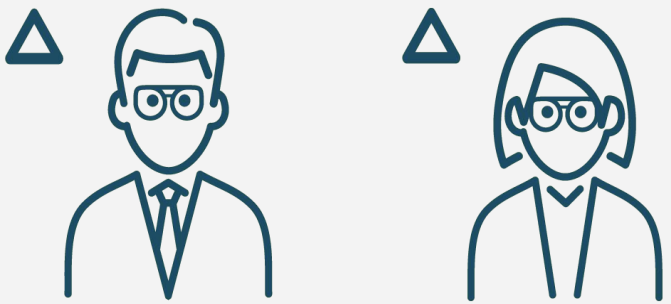
顔が傾いて撮影されている

■ライブラリ仕様 - △ 検出に影響を与える場合があるケース

現在のバージョンでは、以下のケースで検出エラーとなる場合があります。



ブルーライトカットメガネ
(ブルーライト反射が強いとエラーになる場合あり)

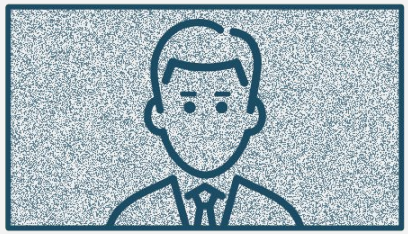


色付き透明サングラス
(色調変化によりエラーとなる場合あり)

■ライブラリ仕様 - △ 検出に影響を与える場合がある撮影状況

現在のバージョンでは、以下の撮影状況で検出エラーとなる場合があります。

△



×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**	FaceAnalyzer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	弊社で用意したテンプレートファイルを読み込みます。
	画像データの読み込み	FaceAnalyzer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、顔領域候補を検出します。
	メガネ装着判定機能	検出された顔画像から、メガネまたはサングラスの装着状態の有無を判定します。
	目位置検出機能	検出された顔領域画像の中から、目の位置を検出します。 両目を検出できた場合は、左右判定を行います。
	目の開閉状態判定機能	検出された目画像から、目の開閉状態を判定します。

■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おおよその処理時間は以下の通りとなります。
ライブラリはCPU動作での参考値となります。

機能	アルゴリズム	出力	処理時間※
顔検出	CNN系物体検出	cv::Rect型（x, y, width, height）	約15ms程度
メガネ判定	CNN系多値分類	0 = なし / 1 = クリアレンズ / 2 = サングラス	約5ms程度
目検出	CNN系物体検出	cv::Rect型（x, y, width, height）	約5ms程度
目の開閉判定	CNN系多値分類	0 = エラー / 1 = 閉じ目 / 2 = 開き目	約5ms程度

※処理時間は、macBookPro(Intel Core i7 第8世代 CPU / 16GB)をもとに算出しています

■ライブラリ仕様 - 機能一覧

入力画像に対する左右目の判定は、
画像向かって左側の目を左目、画像向かって右側の目を右目として出力します。

■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	
struct **ModelFilePath**	この構造体について		顔/目の位置検出などを行うためのモデルファイルを読み込ませるために、FaceAnalyzerクラスのオブジェクト化直後に、本構造体をinit()関数の引数として与えます。	
	顔検出用	std::string	faceDetectModel	顔の位置検出に用いるモデルファイルを読み込む変数です。
		std::string	faceDetectParam	
	メガネ判定用	std::string	glassesModel	メガネ判定に用いるモデルファイルを読み込む変数です。
	目検出用	std::string	eyeDetectModel	目の位置検出に用いるモデルファイルを読み込む変数です。
	目の開閉判定用	std::string	eyeStatusModel	目の開閉判定に用いるモデルファイルを読み込む変数です。

■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 **EyeData**	このクラスについて		顔・目情報センシング結果格納クラスです。 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の値は取得可能です。
	cv::Mat	originalImg	入力した元画像を取得可能です。
	cv::Mat	checkImg	入力元画像に、検出結果を描画プロットした画像です。
	cv::Mat	faceRectImg	入力元画像から、検出された顔画像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明るさ・サイズが適正かどうかの判定結果です。
	std::vector <cv::Rect>	vFaceRect	複数の顔が検出された場合の、それぞれの顔矩形領域の情報を取得可能です。 (x座標,y座標,width,height)
	short	eyeGlassStatus	メガネ装着の有無を判定可能です。 (0 = メガネなし / 1 = クリアレンズメガネ / 2 = サングラス)

■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 **EyeData**	cv::Mat	eyeRectImgLeft	入力元画像から、検出された左目画像を切り出した画像を取得可能です。
	cv::Mat	eyeRectImgRight	入力元画像から、検出された右目画像を切り出した画像を取得可能です。
	cv::Rect	eyeRectAreaLeft	検出された左目矩形領域情報を取得可能です。(x座標,y座標,width,height)
	cv::Rect	eyeRectAreaRight	検出された右目矩形領域情報を取得可能です。(x座標,y座標,width,height)
	float	eyeRectAvgBrightLeft	検出された左目矩形領域の平均輝度値を8bit256階調で取得可能です。
	float	eyeRectAvgBrightRight	検出された右目矩形領域の平均輝度値を8bit256階調で取得可能です。
	short	eyeStatusLeft	左目の開閉状態を判定可能です。(2 = 開き目 / 1 = 閉じ目 / 0 = 検出エラー)
	short	eyeStatusRight	右目の開閉状態を判定可能です。(2 = 開き目 / 1 = 閉じ目 / 0 = 検出エラー)
	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()	EyeDataクラスの全メンバ変数を初期化します。

■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 **FaceAnalyzer**	このクラスについて		顔・目情報を検出する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。引数にModelFilePath構造体をとります。
	bool	registerEyeData	メモリ確保したEyeDataクラス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・目情報検出を行うクラスです。 検出結果はすべてEyeData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		FaceAnalyzer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ドングル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ドングル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、FaceAnalyzer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。 **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。**
	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、FaceAnalyzer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「20231231」であれば、2023年12月31日まで有効なトークンです。

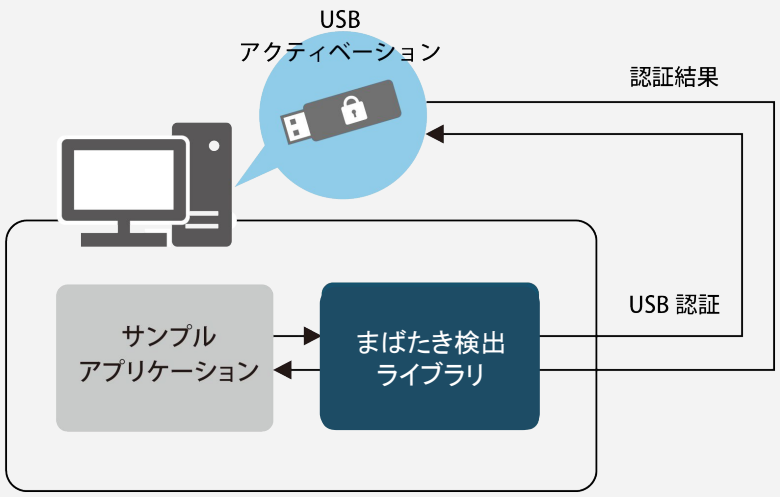
■ライブラリ仕様 - ファクトリ関数

返り値	関数名	説明
FaceAnalyzer*	newFaceAnalyzer()	FaceAnalyzer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FaceAnalyzer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	releaseFaceAnalyzer()	FaceAnalyzerオブジェクトをメモリ開放させるための関数です。 FaceAnalyzer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*	newActivationChallenge()	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。
void	releaseActivationChallenge()	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

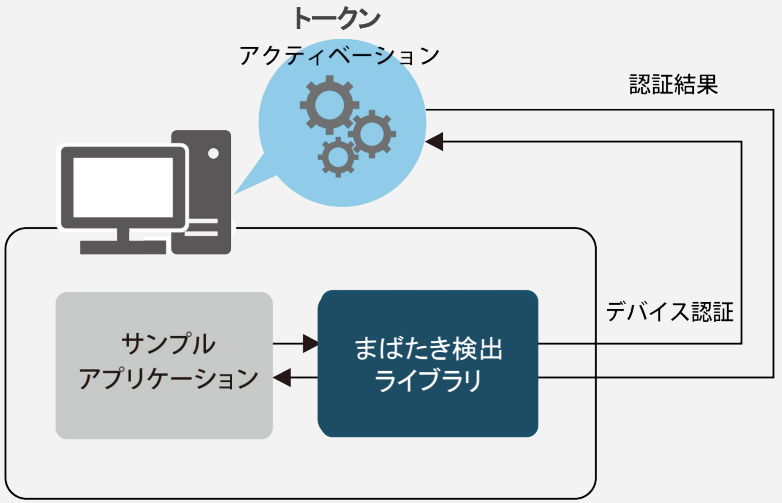
開発版ライセンスでは、本ライブラリをご利用いただくにあたって、アクティベーションが必要となります。アクティベーションには、USB方式と有効期限を利用したトークン方式の2タイプがあります。

◆ USBアクティベーション



有償（USBドングル5本まで込み）
※PC版は原則こちらでお願いしております

◆ トークンアクティベーション



無償
※主にスマホ/タブレット端末での利用企業様向けとなります

■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。

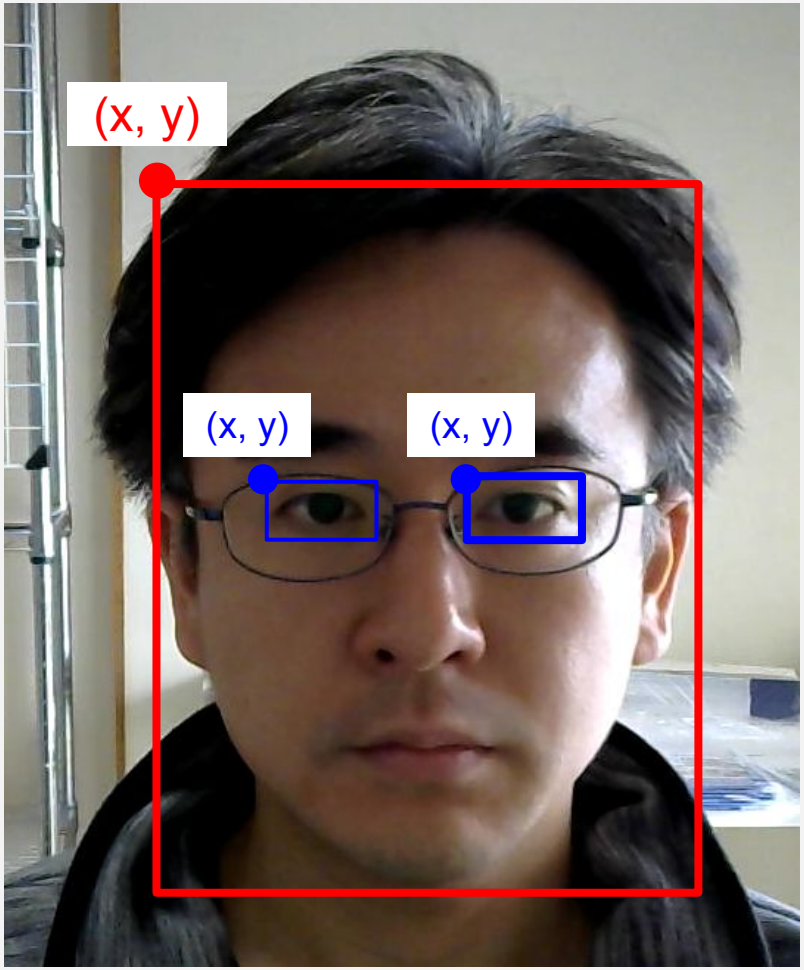
◆ トークンアクティベーション版
config() 実行結果例

```
[toshikazuohno@sample]$ ./get_config
Activation   : Token
Client ID    : 2349799210
Client Name  : Swallow Incubate
SDK Version  : BlinkSDK - ver.1.0.0
Build Date   : 2023-03-20
[toshikazuohno@sample]$
```

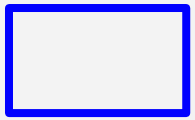
出力データ詳細

■出力データ - 顔･目特徴点検出

EyeData型より取得できる検出位置データの値は、以下の通りとなります。
cv::Rect型のxy座標は、矩形領域の左上頂点座標となります。



= faceRectArea
(x, y, width, height)



= eyeRectArea
(x, y, width, height)

■出力データ - メガネ装着状態判定

ディープラーニング

単一フレームの画像から、顔が検出された場合、メガネ装着の有無をディープラーニングモデルを用いて判定します。処理結果が格納されるEyeDataクラスの、eyeGlassStatusメンバより0～2のいずれかの値が出力されることによりメガネ装着判定を行うことが可能です。

裸眼

eyeGlassStatus = 0

クリアレンズ
メガネ

eyeGlassStatus = 1

サングラス

eyeGlassStatus = 2

■出力データ - 目の開閉状態判定

ディープラーニング

目の開閉パラメータは、EyeDataクラスのメンバ変数であるeyeStatus(Left/Right)より取得できます。値はint型の0～2のいずれかが返ります。値の詳細は以下の通りとなります。

eyeStatus = 0

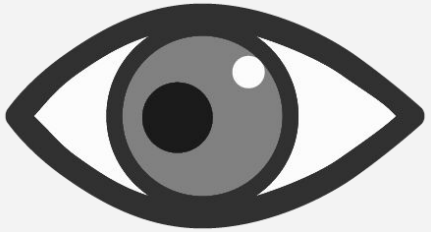
Error

判定エラー

eyeStatus = 1

目が閉じている

eyeStatus = 2



目が開いている

■message一覧

EyeData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。

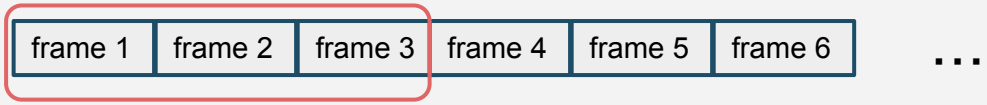
メッセージ内容	内容
Process Successful	全ての処理が完了しました
Error: Failed to detect face	顔検出に失敗しました
Error: Lack of face area brightness	顔領域の明るさ不足です（顔領域平均輝度値 50未満 - 8bit256階調）
Error: Failed to detect eyes	目検出に失敗しました
Error: Failed to detect eyes(wearing sunglasses)	目検出に失敗しました（サングラス着用のため）

まばたき判定の方法例

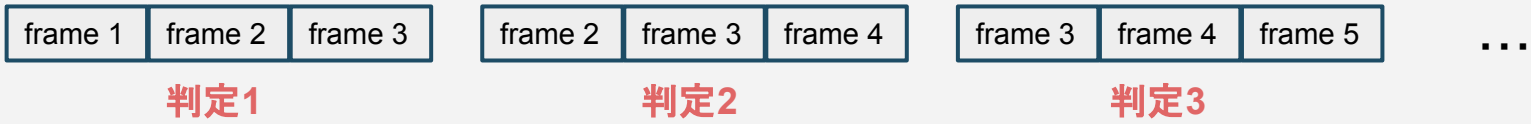
■まばたき判定 - サンプル

まばたき検出ライブラリは、1フレームの検出結果を返すのみとなりますので、まばたき判定は、検出結果の時間変化をもとに独自で判定していただく必要があります。以下にまばたき判定の例を掲載いたしますが、独自に判定アルゴリズムを策定いただくことも可能です。詳しくはサンプルアプリを参照してください。

<まばたき判定に用いるフレーム数 > - 3つずつ



<まばたき判定に使う時間フレーム > 連続する3フレーム



<まばたき判定アルゴリズム >