目検出技術(可視光版)
SDK/インターフェース仕様書

株式会社スワローインキュベート

2020年06月02日

■はじめに

目検出SDK(可視光版)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。

・本書に基づき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

目検出技術　テクニカルサポート窓口

TEL: 029-886-9912　MAIL: support@swallow-incubate.com

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 / Raspbian OS iOS / Android OS	
CPUアーキテクチャ	x86_64、x86、Armv7系、Arm64系	
推奨メモリ	2GB以上を推奨	
依存ライブラリ	OpenCV 4系 / OpenCV Contrib 4系	
実行環境 / ビルド環境	Linux kernel 4.9以降	gccを推奨
	macOS 10.11以降	
	Raspbian Buster 推奨	
	Windows 7 以降	MicroSoft Visual C++コンパイラを推奨
	iOS 10以降	最新のXcodeの利用を推奨
	Android OS 5.0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カラー画像 (8bit3ch または 8bit4ch)**
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	PC版 **Quad-VGA**（横1280 x 縦960 px） / **720p**（横1280 x 縦960 px） スマホ版 **1080p**（横1080 x 縦1920 px）
入力画像解像度	<検出される顔領域のピクセル数> **最低 width 300px 以上 推奨**
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。
同時検出可能人数	**1名**

※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)

目検出ライブラリ

GitHub

C++コード / ビルド手順

目検出サンプルアプリ

USBドングル
(PC版のみ)

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	EyeSensing.hpp （インターフェースファイル） libEyeSensing.dylib サンプルアプリ（C++）
Windows OS	EyeSensing.hpp EyeSensing.dll (実行時参照ライブラリ) EyeSensing.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)**	EyeSensing.hpp （インターフェースファイル） libEyeSensing.so サンプルアプリ（C++）
iOS	EyeSensing.Framework (EyeSensing.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ビルド方法マニュアル
Android OS	EyeSensing.hpp libEyeSensing.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 OS向けOpenCV + OpenCV Contribビルド方法マニュアル

ライブラリ仕様

■ライブラリ仕様

現在のバージョンでは、SDKの仕様は以下の通りとなります。

実行処理速度の目安

項目	内容
処理速度	**1フレーム 約150～200ms (1280 x 960 px)** （CPU: Core i7 / メモリ16GB のマシン検証時）

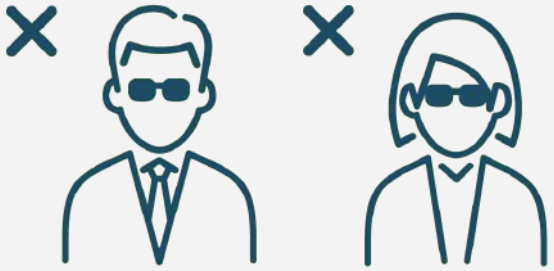
装着具の対応状況

装着具	裸眼	メガネ	コンタクト	カラコン	目の傷 病気	サングラス	眼帯
対応状況	◎	○	◎	○	△	×	△

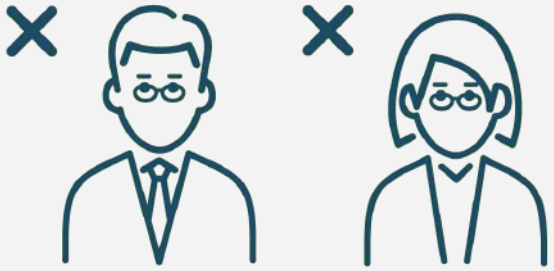
※虹彩境界（黒目）から強膜（白目部分）にかけての傷・病気がある場合は、目検出エラーになる可能性があります

■ライブラリ仕様 - X 検出不可となるケース

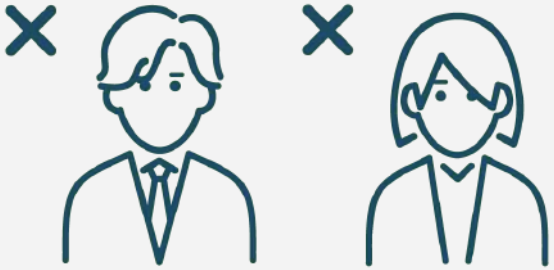
現在のバージョンでは、以下のケースで検出エラーになりやすくなります。



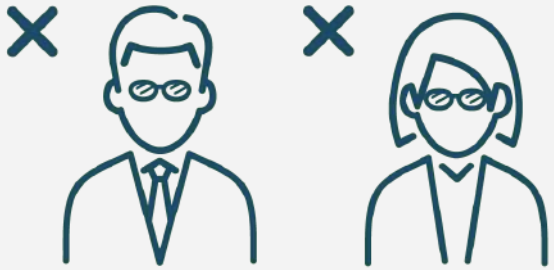
レンズが黒いサングラス



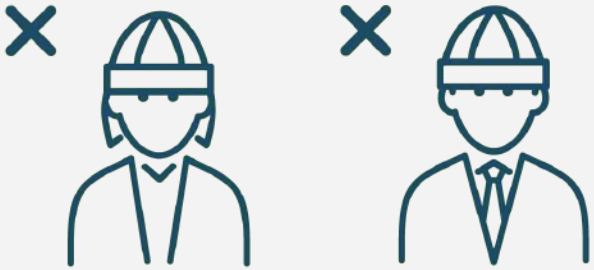
メガネフレームと目にかかっている



髪の毛が目にかかっている



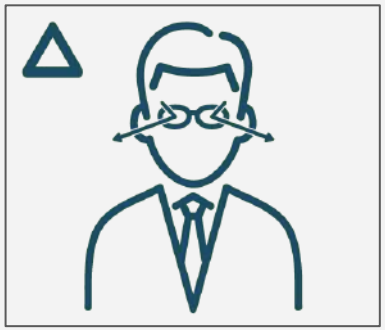
外光の映り込みが激しい

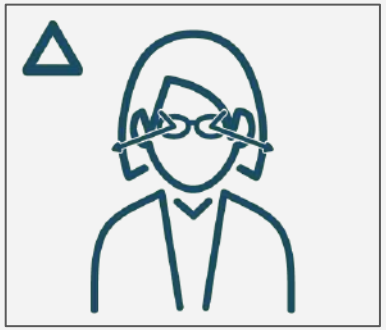


帽子を深くかぶっている

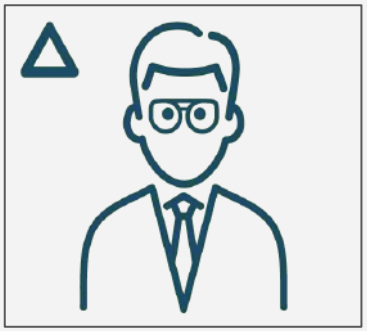
■ライブラリ仕様 - △ 検出に影響を与える場合があるケース

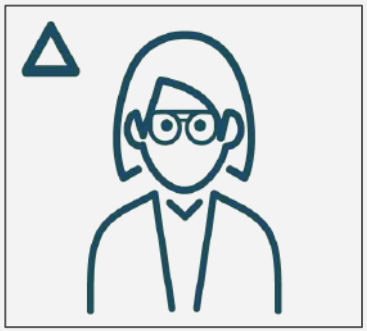
現在のバージョンでは、以下のケースで検出エラーとなる場合があります。



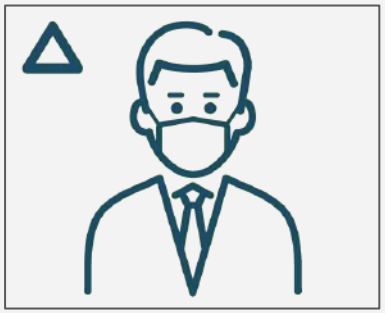


ブルーライトカットメガネ
(ブルーライト反射が強いとエラーになる場合あり





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



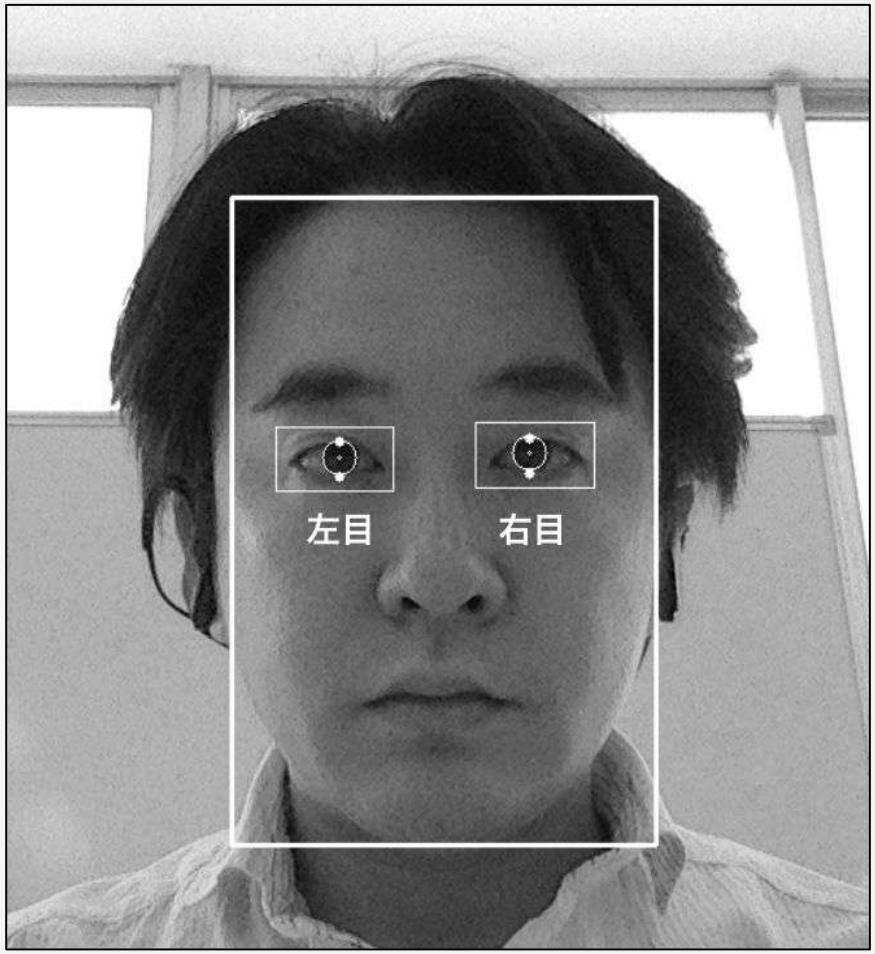
マスク
(顔検出エラーとなる場合あり

■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**	EyeSensing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。
	パラメータ調整	構造体をEyeSensingオブジェクトのinit()に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	EyeSensing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	**顔検出機能**	入力された画像から、顔領域候補を検出します。
	目検出機能	検出された顔領域画像の中から、目を検出します。 両目を検出できた場合は、左右判定を行います。
	虹彩(黒目)検出機能	検出された目画像から、虹彩(黒目)位置検出を行います。虹彩検出できた場合は、虹彩中心座標と虹彩左右半径を取得することができます。
	上下まぶた検出機能	検出された目画像から、虹彩中心座標と同じx座標上のまぶた位置のxy座標を上下それぞれ検出します。
	目尻目頭検出機能	検出された目画像から、目尻目頭の位置のxy座標の検出を行います。
	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。
	メガネ装着判定機能	検出された顔画像から、メガネの装着状態の有無を判定します。

■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。上下まぶた、目尻目頭についても以下同様です。



■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		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は取得可能です。
	cv::Mat	originalImg	入力した元画像(RGB)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(RGB)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(RGB)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。
	cv::Mat	eyeRectImgLeft	入力元画像から、検出された左目画像(RGB)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(RGB)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。
	int	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)
	int	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)
	bool	eyeGlassStatus	メガネ装着の有無を判定可能です。(true=メガネあり / false=メガネなし)

■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**	int	irisRadiusLeft	検出された左目瞳(虹彩)半径サイズ(px)を取得可能です。
	int	irisRadiusRight	検出された右目瞳(虹彩)半径サイズ(px)を取得可能です。
	cv::Point	irisCenterLeft	検出された左目瞳(虹彩)中心位置情報を取得可能です。
	cv::Point	irisCenterRight	検出された右目瞳(虹彩)中心位置情報を取得可能です。
	cv::Point	eyeLeftUpperEyelid	検出された左目上まぶた位置情報を取得可能です。
	cv::Point	eyeLeftLowerEyelid	検出された左目下まぶた位置情報を取得可能です。
	cv::Point	eyeRIghtUpperEyelid	検出された右目上まぶた位置情報を取得可能です。
	cv::Point	eyeRightLowerEyelid	検出された右目下まぶた位置情報を取得可能です。
	cv::Point	eyeLeftCornerL	検出された左目目尻位置情報を取得可能です。
	cv::Point	eyeLeftCornerR	検出された左目目頭位置情報を取得可能です。
	cv::Point	eyeRightCornerL	検出された右目目頭位置情報を取得可能です。
	cv::Point	eyeRightCornerR	検出された右目目尻位置情報を取得可能です。
	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 **EyeSensing**	このクラスについて		顔・目周辺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ExternalFilePath構造体をとります。 第二引数にParams構造体をとりますが、省略可能です。
	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	detectRGB	cv::Matを引数とし、顔・目周辺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。

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		EyeSensing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。
	bool	checkUSB()	**[USB版のみ]** 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版のみ]** USBドングルとライブラリに埋め込まれたIDとが正しいもの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、EyeSensing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。
	bool	checkToken	**[デバイス版のみ]** ライブラリに埋め込まれた有効期限と、端末時刻を読み取り、有効期限内かどうかをチェックするための関数です。 本関数からtrueが返ることにより、EyeSensing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。
	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。

■ライブラリ仕様 - クラス構成

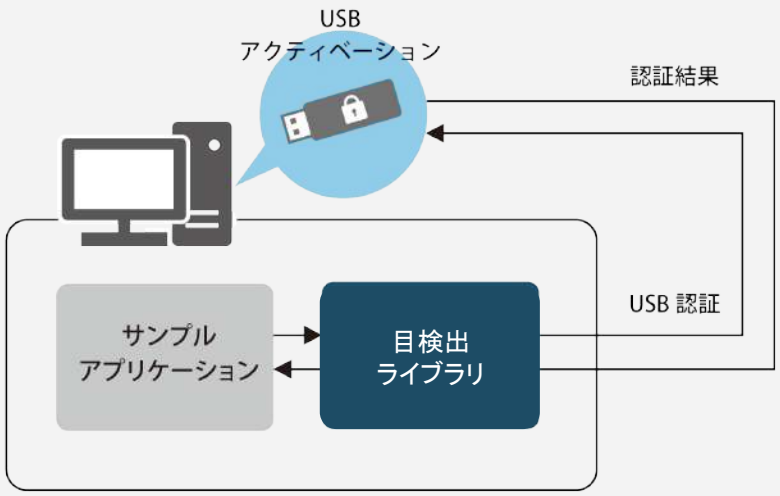
クラス名	データ型	メンバ名	説明
struct **ExternalFilePath**	この構造体について		顔検出、目検出を行うためのモデルファイルを読み込ませるために、EyeSensingクラスのオブジェクト化直後に、本構造体をinit()関数の第一引数として与えます。
	std::string	faceDetectConfigFileName	顔検出に用いるモデルファイル名を読み込む変数です。
	std::string	faceDetectWeightFileName	顔検出に用いるモデルファイル名を読み込む変数です。
	std::string	eyeDetectConfigFileName	目検出に用いるモデルファイル名を読み込む変数です。

クラス名	データ型	メンバ名	説明
struct **Params**	この構造体について		顔検出、目検出を行うためのパラメータ値を読み込ませるために、EyeSensingクラスのオブジェクト化直後に、本構造体をinit()関数の第二引数として与えます。本引数は省略可能で、省略した場合は、内部であらかじめ定められた初期設定値で処理が行われます。
	int	minFaceWidth	顔検出を行う最小顔横幅サイズ(px)です。
	int	maxFaceWidth	顔検出を行う最大顔横幅サイズ(px)です。
	int	minEyeAreaBrightness	目検出以後の処理を行うために必要な目領域平均輝度値の最低値です。

■ライブラリ仕様 - アクティベーション

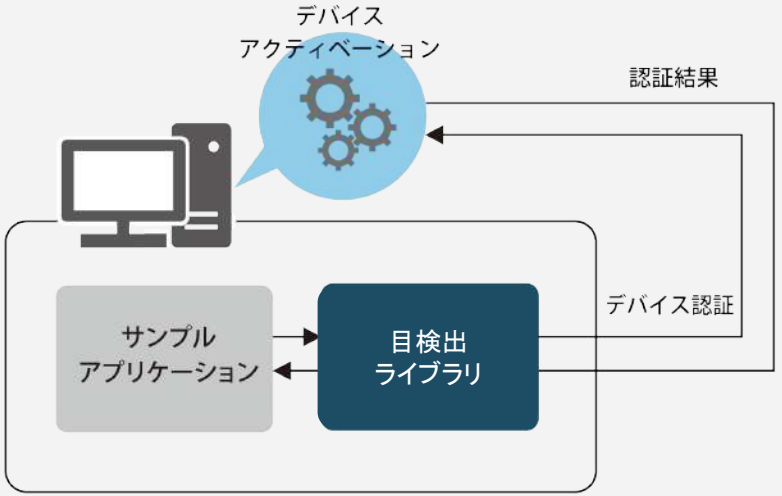
開発版ライセンスでは、本ライブラリをご利用いただくにあたって、アクティベーションが必要となります。アクティベーションには、USB方式と有効期限を利用したデバイス時刻取得方式の2タイプがあります。

◆ USBアクティベーション



有償（USBドングル1本1万円）
※PC版は原則こちらでお願いしております

◆ デバイスアクティベーション



無償
※主にスマホ/タブレット端末で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。

◆ USBアクティベーション版 config() 実行結果例

```
[toshikazuohno@sample]$ ./get_config
Activation   : USB
Client ID    : 2349799210
Client Name : Swallow Incubate
SDK Version : EyeSensingSDK - ver.2.0.0
Start-Date   : 2020-04-14
[toshikazuohno@sample]$
```

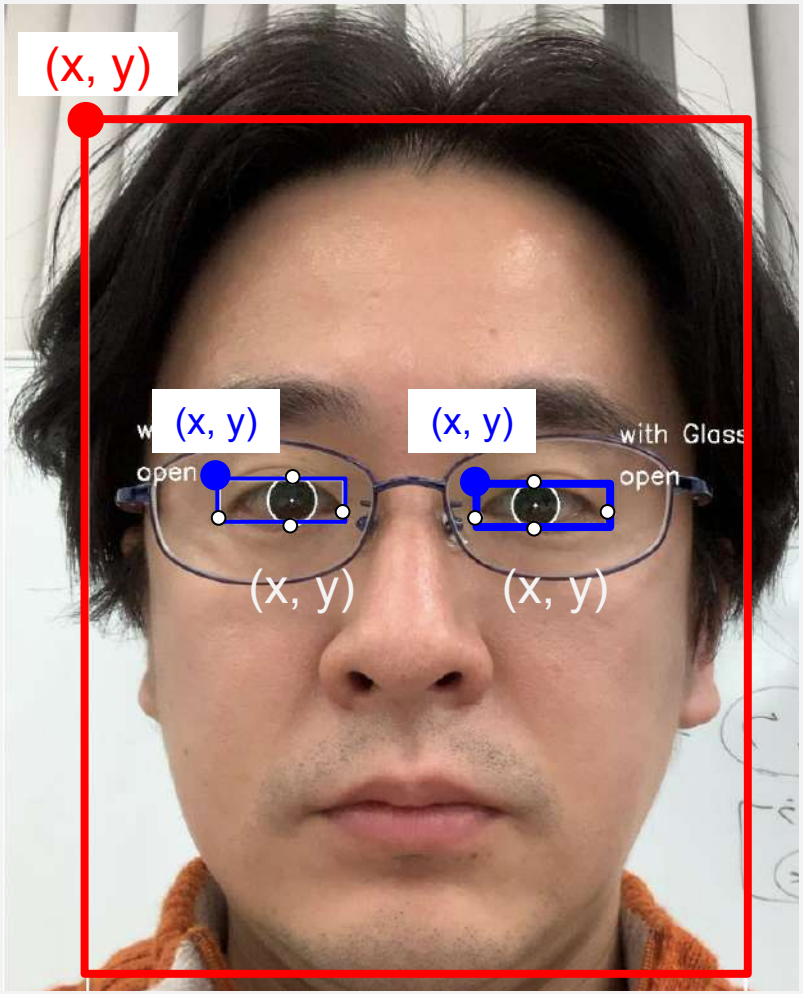
◆ デバイスアクティベーション版 config() 実行結果例

```
[toshikazuohno@sample]$ ./get_config
Activation   : Token
Client ID    : 2349799210
Client Name : Swallow Incubate
SDK Version : EyeSensingSDK - ver.1.0.1
Start-Date   : 2020-05-11
[toshikazuohno@sample]$
```

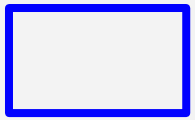
出力データ詳細

■出力データ図解

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。

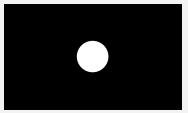


= faceRectArea
(x, y, width, height)



= eyeRectArea
(x, y, width, height)

= irisDiameter



= upperEyelid / lowerEyelid
eyeCornerL / eyeCornerR
(x, y)

■出力データ - 目の開閉状態判定

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

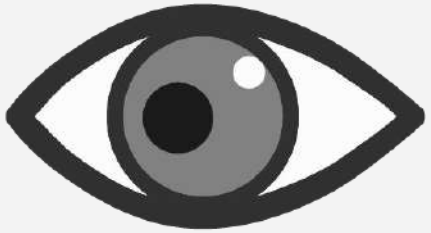
Error

目検出エラー

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: init() first	detectRGB()実行前に、init()を実行してください
Error: registerEyeData() first	detectRGB()実行前に、registerEyeData()を実行してください
Error: Empty input image	入力画像が存在しません
Error: Input RGB image	RGBカラー画像を入力してください
Error: Failed to detect face	顔検出に失敗しました (顔がない or 横幅サイズ不足 or 超過)
Error: Lack of face area brightness	顔領域の明るさ不足です

■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。

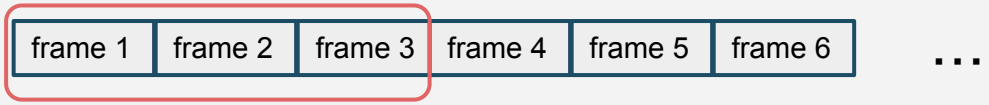
メッセージ内容	内容
Error: Unable to load haarcascade file	カスケード検出器を読み込めませんでした
Error: Failed to detect eyes	目検出に失敗しました（両目位置推定エラー）
Error: Failed to detect eye	目検出に失敗しました（片目位置推定エラー）
Error: Failed to detect open eye	目検出に失敗しました（開き目検出エラー）
Error: Failed to detect iris（3001）	瞳(虹彩)位置検出に失敗しました（目領域の明るさ不足）
Error: Failed to detect iris（3002）	瞳(虹彩)位置検出に失敗しました（メガネへの外光強い映り込み）
Error: Failed to detect iris（3003）	瞳(虹彩)位置情報に失敗しました（その他）
Error: Failed to detect eyelid	両目ともまぶた情報の検出に失敗しました
Error: Failed to detect eyeCorner	両目とも目尻目頭検出に失敗しました
Error: Iris position invalid	視距離推定を行うための虹彩位置が適切ではありません（端目）

まばたき判定の例

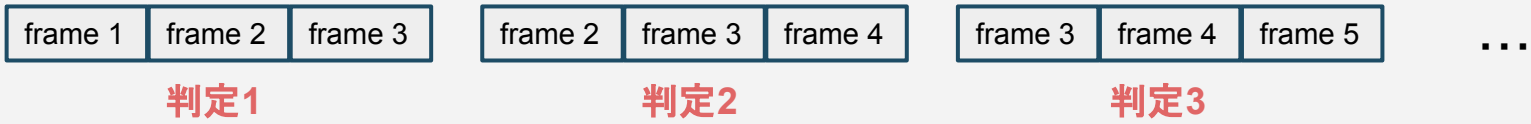
■まばたき判定 - サンプル

目検出ライブラリは、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。

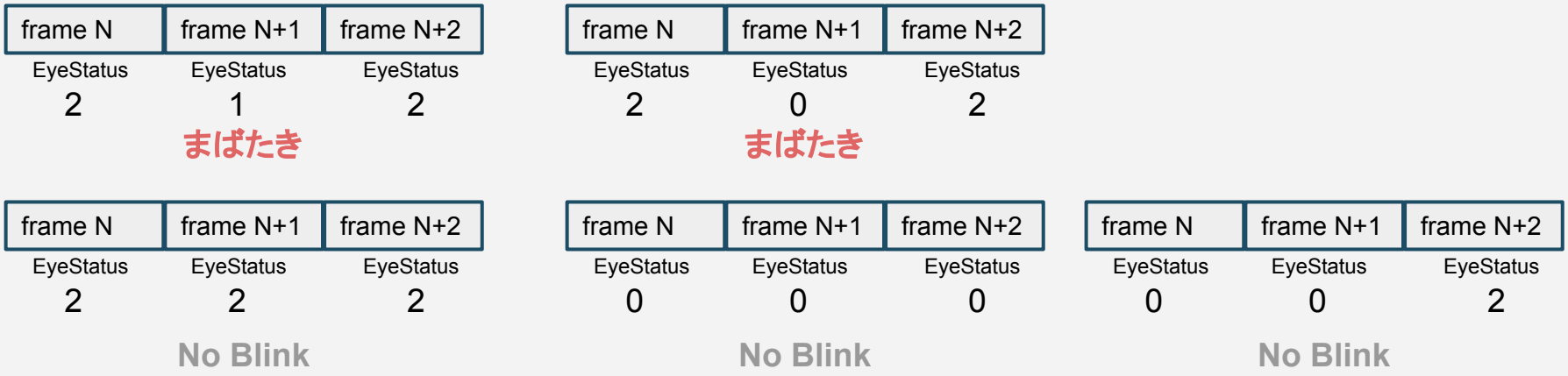
<まばたき判定に用いるフレーム数 > - 3つずつ



<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 >

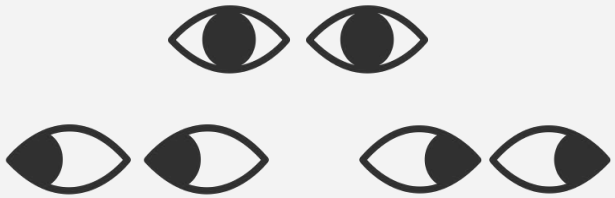


目検出の応用

■生体判定SDK

顔の動き、目の動き、まばたき、くちびるの動きを組み合わせることで、生体かどうかのなりすましチェックを行うことが可能です。
詳しくは、生体判定SDKのWEBサイトをご確認ください。

顔の動き判定



目の動き判定

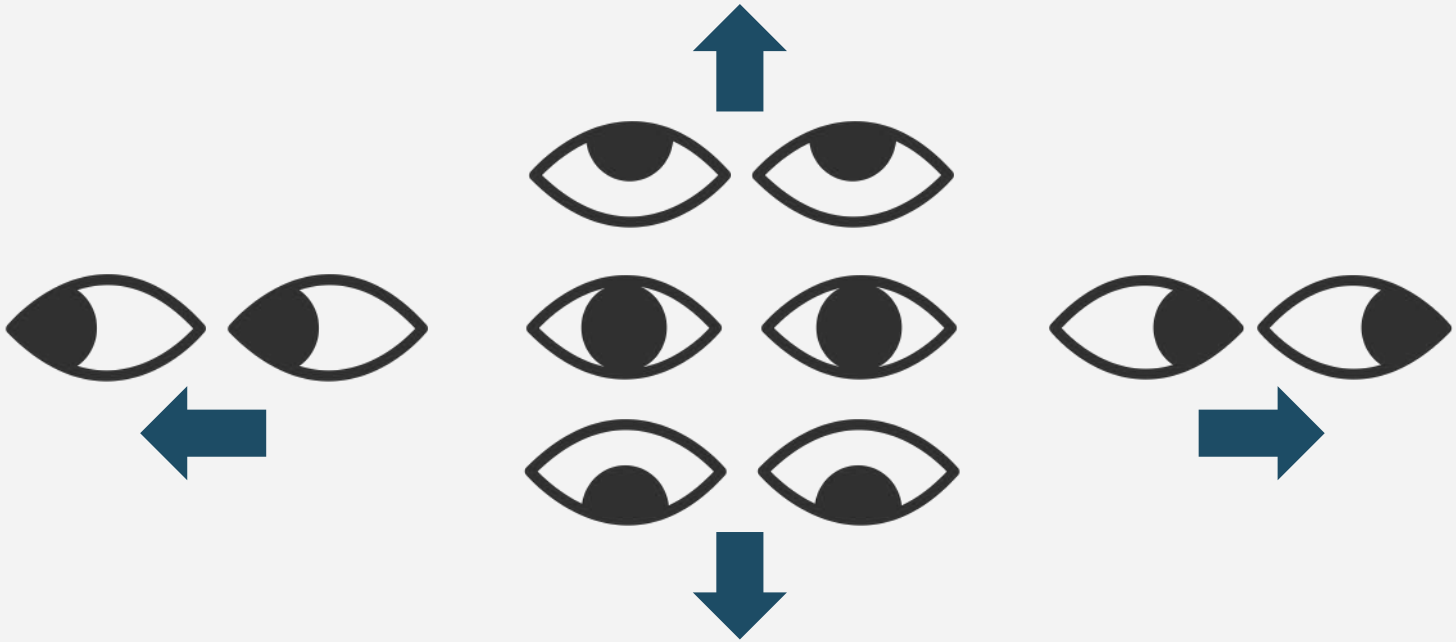
くちびるの動き判定

まばたき判定

https://bio-check.pas-ta.io/

■視線検出SDK

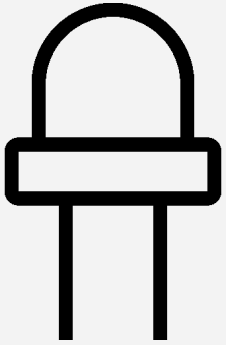
顔の向きや目の動きの変化量を計算して、視線推定を行うことが可能です。
詳しくは、視線検出SDKのWEBサイトをご確認ください。



https://eyetrack.pas-ta.io/

■目検出SDK(赤外光版)

目検出SDKには、本可視光版以外にも、赤外光版がございます。
瞳孔位置検出、瞳孔径の測定には、赤外光版をご利用いただく必要があります。
赤外光版では、可視光版で検出できる情報と同様の情報が取得可能ですが
赤外線センサと、赤外光照射装置が別途必要となります。



赤外LED

瞳孔検出

赤外線カメラ

https://pupil.pas-ta.io/