視距離推定技術
SDK/インターフェース仕様書(ver.1.0.1)

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

視距離推定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

視距離推定技術　テクニカルサポート窓口

MAIL: support@swallow-incubate.com

■SDK更新履歴

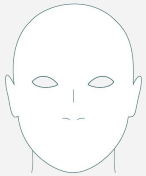
バージョン	日時	変更内容サマリー
ver.1.0.0	2023年03月20日	初版
ver.1.0.1	2023年09月15日	OpenCVリンクの変更(4.5.5=>4.7.0)、その他微修正

このSDKでできること

■このSDKでできること

視距離推定技術SDKでは、ディープラーニングや機械学習を用いて、顔・目の特徴点などを検出し、それらの情報を元に「カメラから目までの距離を推定する」ことが可能です。これら一連の検出処理を、WEBカメラなどの可視光RGBカメラを用いてできることが特徴となります。

特徴点検出



DL

顔位置検出

DL

メガネ・サングラス判定

DL

目位置検出

DL

虹彩位置検出

視距離推定

特許

視距離推定

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	2GB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/g++/clangを推奨
	macOS 10.11以降	
	Raspberry Pi OS 以降 推奨	
	Windows 10 以降	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

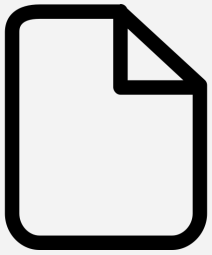
現在のバージョンでは、以下の入力画像を推奨しています。

項目	内容
画像カラー仕様	**RGBカラー画像** (8bit3ch または 8bit4ch) ※RGB565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 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。
撮影可能画角	原則、**正面視**を対象とします

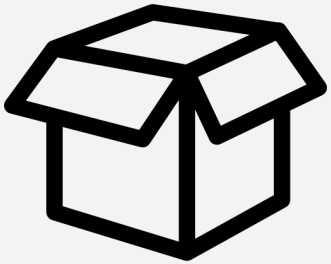
※その他の入力画像でのご利用を希望される場合は、お問い合わせください。

SDK構成

■SDK構成



interfaceファイル
(hpp)



動的リンクライブラリ
(dll/so/dylibなど)

視距離推定ライブラリ

C++コード / ビルド手順

視距離推定サンプルアプリ

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。

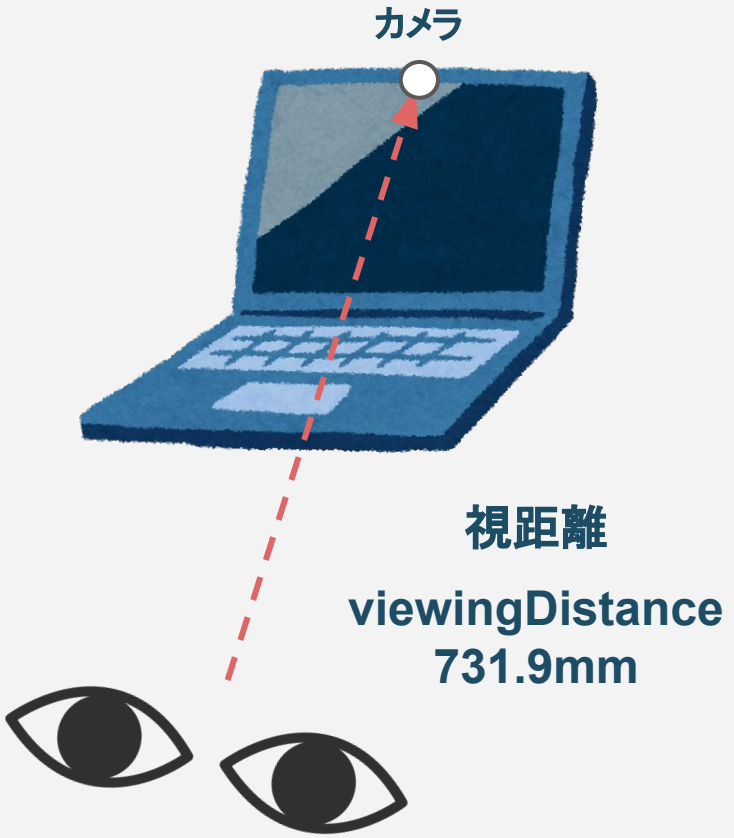
OS	提供物
macOS	ViewDist.hpp （インターフェースファイル） libViewDist.dylib サンプルアプリ（C++）
Windows OS	ViewDist.hpp ViewDist.dll (実行時参照ライブラリ) ViewDist.lib (ビルド時取込ライブラリ) サンプルアプリ（C++）
各種Linux OS **(Raspberry Pi OS含む)**	ViewDist.hpp（インターフェースファイル） libViewDist.so サンプルアプリ（C++）
iOS	ViewDist.Framework（ViewDist.hpp含む） サンプルアプリ（Objective-C 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	ViewDist.hpp libViewDist.so（arm64-v8a / armeabi-v7a / x86 / x86_64） Android向けOpenCV + OpenCV Contrib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）

ライブラリ仕様

■視距離推定について

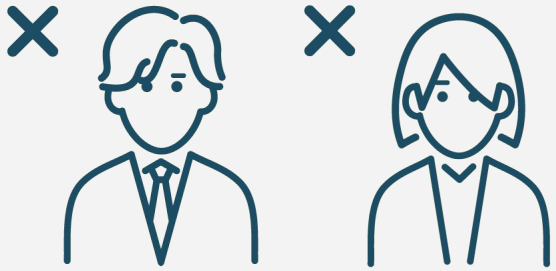
特許取得済

目の特徴点検出結果をもとに、カメラから目までの距離を取得できます。
視距離の値は、EyeDataクラスのメンバ変数であるviewingDistanceより取得することができ、単位mm（ミリメートル）で取得できます。
単位cm（センチメートル）にする場合は、この値を10で割ってください。

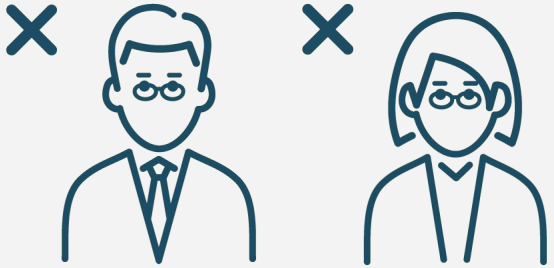


■ライブラリ仕様 - ✕検出不可となるケース

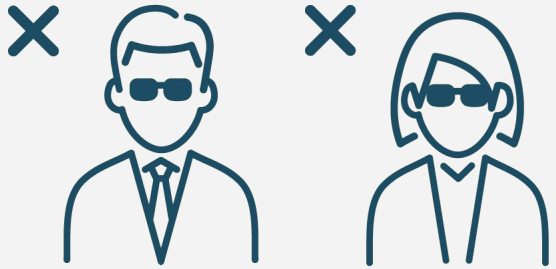
目の位置検出においては、以下のケースで検出エラーになりやすくなりますのでご注意ください。



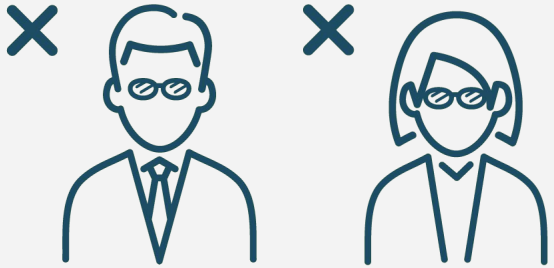
髪の毛が目にかかっている



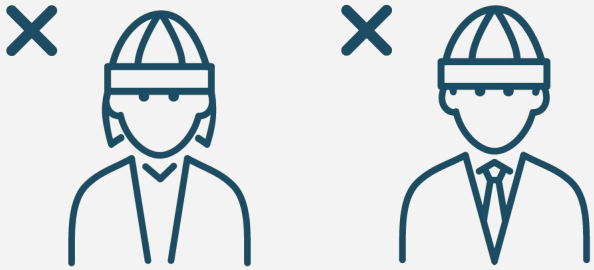
メガネフレームが目にかかっている



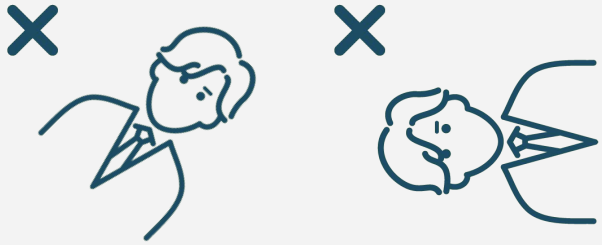
レンズが黒いサングラス



外光の映り込みが激しい



帽子を深くかぶっている

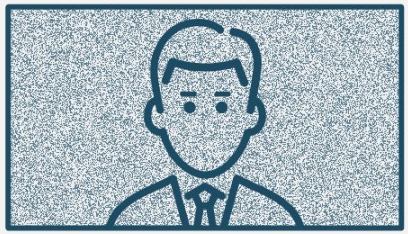


顔が傾いて撮影されている

■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

各種検出においては、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**	CalcViewDist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を読み込みます。
	パラメータ調整	CalcViewDist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	CalcViewDist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、顔領域候補位置を検出します。
	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以降の処理を中止します。
	顔向き判定機能	検出された顔領域画像から、深層学習モデルを用いた推論を行い、水平方向・垂直方向それぞれの顔向きを判定します。
	メガネ装着判定機能	検出された顔領域画像から、深層学習モデルを用いた推論を行い、メガネの装着の有無及びメガネありの場合は、クリアレンズメガネかサングラスかの区別も可能です。
	目の位置検出機能	検出された顔領域画像から、深層学習モデルを用いた推論を行い、両目の検出を行います。
	目の開閉判定機能	検出された目画像から、深層学習モデルを用いた推論を行い、目の開閉状態を判定します。
	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。虹彩検出に成功した場合は、虹彩中心座標と虹彩半径を取得することができます。
	視距離推定機能	各種センシング情報をもとに、カメラから目までの距離を推定します。

■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			視距離推定を行うためのモデルファイルを読み込ませるために、CalcViewDist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。
	顔位置検出用	std::string	faceDetectModel	顔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	
	顔向き判定用	std::string	faceDirHModel	顔向き判定に用いるモデルファイルを読み込む変数です。
		std::string	faceDirVModel	
	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。
	目位置検出用	std::string	eyeDetectModel	目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。
	目向き判定用	std::string	eyeDirModel	目の向き判定に用いるモデルファイルを読み込む変数です。
	目の特徴点検出用	std::string	eyeLandmarkModel	目の特徴点検出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
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、CalcViewDist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。
	float	calibParam	カメラのレンズ、画角に応じて調整する視距離推定パラメータです。
	float	vdParam	視距離推定結果にmm単位で加減算する視距離推定パラメータです。

■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		視距離推定結果格納クラスです。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。
	cv::Mat	originalImg	入力した元画像を取得可能です。
	cv::Mat	checkImg	入力した元画像に、検出結果を矩形や丸点などで描画プロットした画像です。
	cv::Mat	faceRectImg	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、検出位置、顔の明るさ、顔のサイズ、フレーム面積と顔面積の割合が適正かどうかを判定します。
	std::vector<cv::Rect>	vFaceRect	検出したすべての複数の顔領域情報を取得可能です。(x座標,y座標,width,height)
	short	eyeGlassStatus	検出された顔をもとにした、深層学習モデルによるメガネ装着の有無の推論結果です。初期値は-1となっており、判定が行われると出力は0～2の整数値となります。 0は裸眼、1はクリアレンズメガネ、2はサングラスであると推定されます。
	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 **EyeData**	cv::Mat	eyeRectImgLeft	入力した元画像から、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	入力した元画像から、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	検出された顔をもとにした、深層学習モデルによる目の開閉状態の推論結果です。 初期値は-1となっており、判定が行われると出力は0～2の整数値となります。 0は判定エラー、1は閉じ目、2は開き目であると推定されます。
	short	eyeStatusRight	
	short	irisRadiusLeft	検出された左目瞳(虹彩)半径情報を取得可能です。
	short	irisRadiusRight	検出された右目瞳(虹彩)半径情報を取得可能です。
	cv::Point	irisCenterLeft	検出された左目瞳(虹彩)中心位置座標を取得可能です。
	cv::Point	irisCenterRight	検出された右目瞳(虹彩)中心位置座標を取得可能です。
	float	viewingDistance	カメラから目までの距離(単位:mm)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
class **CalcViewDist**	このクラスについて		視距離推定に必要な顔および目の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。第一引数にModelFilePath構造体、第二引数にParams構造体をとります。
	bool	registerEyeData	メモリ確保済みの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、顔位置検出、目位置検出、視距離推定処理を行うクラスです。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		CalcViewDist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CalcViewDist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、CalcViewDist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「20221231」であれば、2022年12月31日まで有効なトークンです。

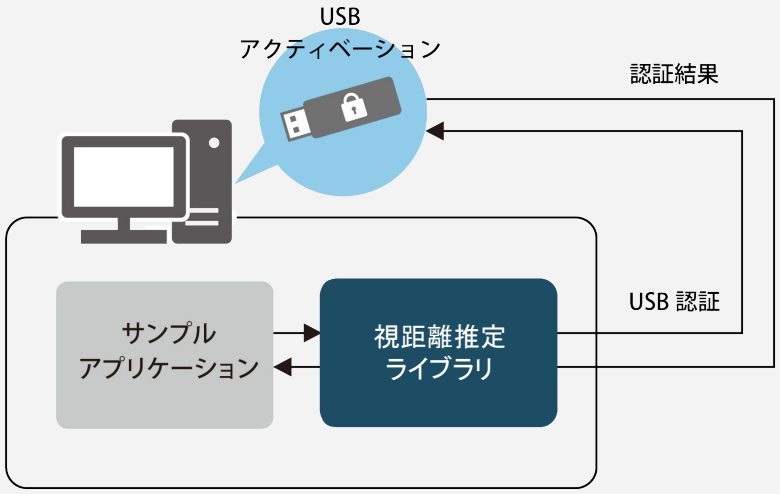
■ライブラリ仕様 - ファクトリ関数

返り値	関数名	説明
CalcViewDist*	newCalcViewDist()	CalcViewDist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、CalcViewDist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	releaseCalcViewDist()	CalcViewDistオブジェクトをメモリ開放させるための関数です。 CalcViewDist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

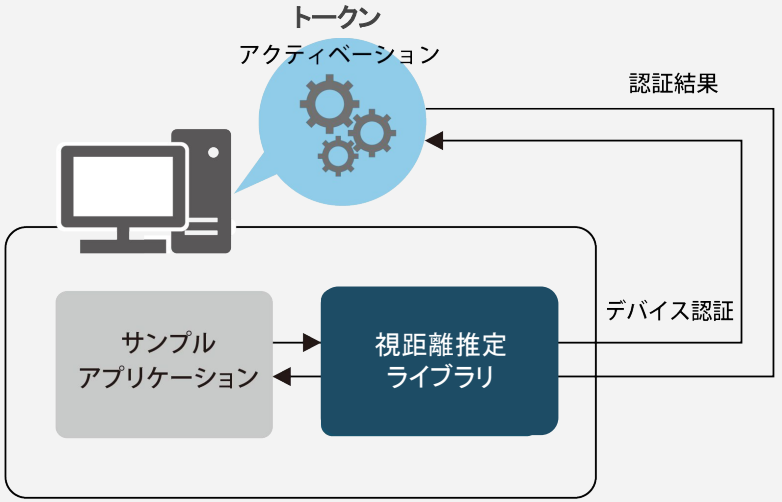
開発版ライセンスでは、本ライブラリをご利用いただくにあたって、アクティベーションが必要となります。アクティベーションには、USB方式と有効期限を利用したトークン方式の2タイプがあります。

◆ USBアクティベーション



有償（USBドングル5本まで無料）
※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。

◆ config() 実行結果例

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

出力データ詳細

■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	process()実行前に、init()を実行してください
Error: registerEyeData() first	process()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(24bit)	RGB画像(24bit)を入力してください
Error: Failed to detect face	顔位置検出に失敗しました（顔が存在しない / サイズ不足）
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以上 – 8bit256階調）
Error: Failed to detect eyes(wearing sunglasses)	目の位置検出に失敗しました（サングラス装着のため）
Error: Failed to detect eyes	目の位置検出に失敗しました
Error: Failed to detect open eye	閉じ目のみを検出しました（開き目なし）
Error: Face direction is Lower	顔向きが下向きのため目向き判定・目の開閉判定ができません
Error: Failed to detect eye landmark points	目の特徴点検出に失敗しました