虹彩認証技術
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

2021年02月01日

■はじめに

虹彩認証技術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

虹彩認証技術　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	
CPUアーキテクチャ	x86系64bit、Armv7系、Armv8系、Arm64系	
推奨メモリ	1GB以上を推奨	
依存ライブラリ	OpenCV 4系 / OpenCV Contrib 4系（faceモジュール / dnnモジュール）	
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 10 以降	MicroSoft Visual C++コンパイラを推奨

※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。

■入力画像

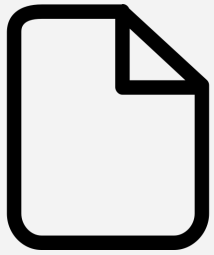
現在のバージョンでは、以下の入力画像を推奨しています。

項目	内容
画像カラー仕様	**グレースケール画像(8bit256階調)のみ**
センサ種別	**赤外線センサ画像**（可視光カットフィルタは任意）
入力画像解像度	<検出される虹彩直径ピクセル数> **虹彩直径ピクセル 最低 120px 以上 推奨**
入力画像タイプ	**顔を含む画像 / 顔画像 / 目画像　いずれも対応**
同時検出可能人数	**1名（両目 or 片目）**

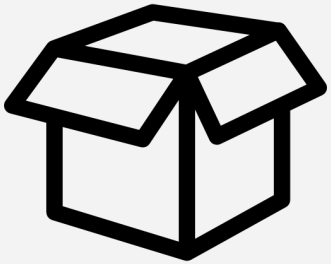
※その他の入力画像でのご利用を希望される場合は、お問い合わせください。

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)

GitHub

C++コード / ビルド手順

虹彩認証ライブラリ

虹彩認証サンプルアプリ

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。

OS	提供物
macOS	IrisProc.hpp（インターフェースファイル） libIrisProc.dylib サンプルアプリ（C++）
Windows OS	IrisProc.hpp IrisProc.dll IrisProc.lib サンプルアプリ（C++）
各種Linux OS **(Raspbian OS含む)**	IrisProc.hpp（インターフェースファイル） libIrisProc.so（ELF形式） サンプルアプリ（C++）

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200ms (500万画素)** （CPU: Core i7 / メモリ16GB のマシン検証時）

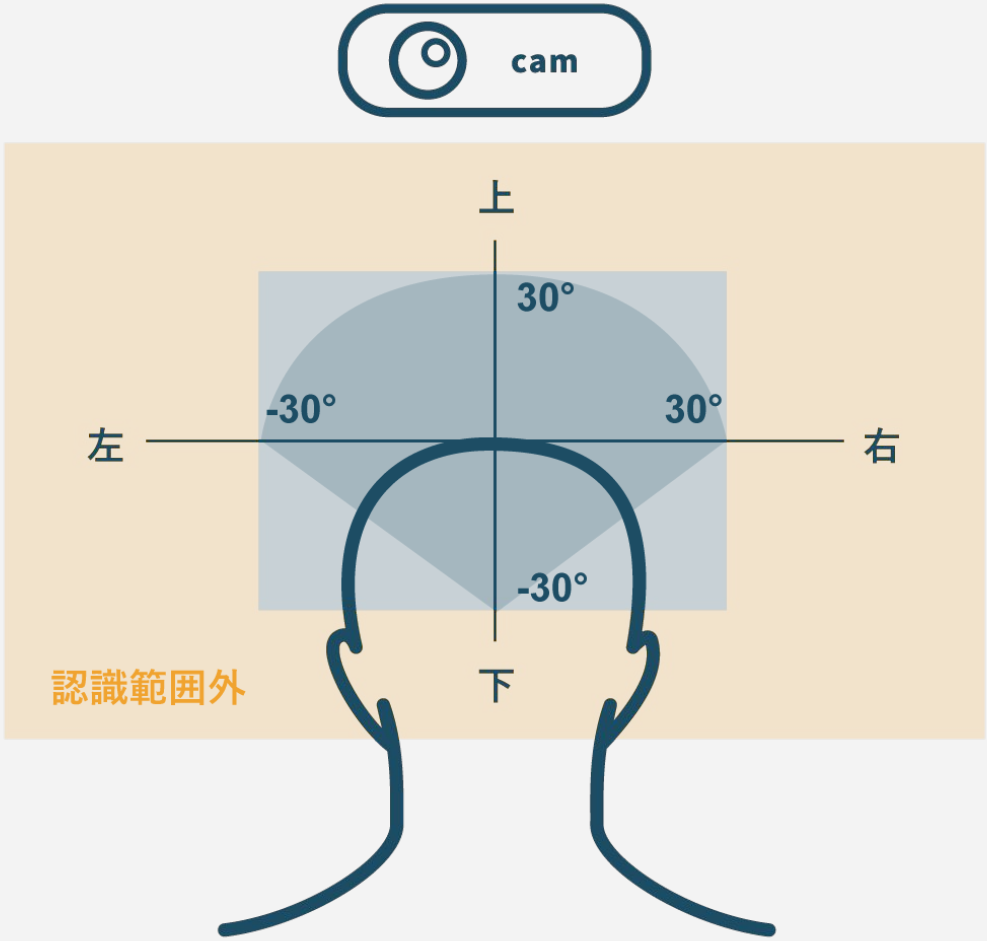
装着具の対応状況

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、虹彩認証エラーになる可能性があります

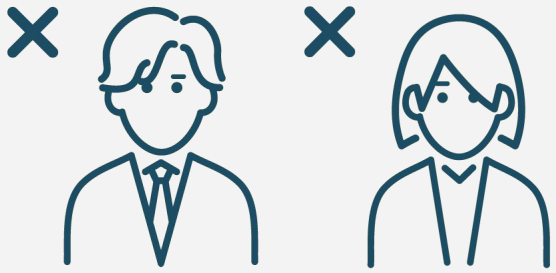
■ライブラリ仕様 - 検出可能画角

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。

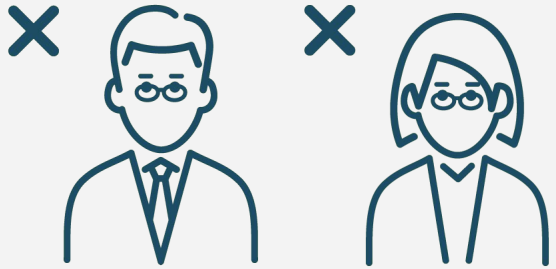


■ライブラリ仕様 - ✕検出不可となるケース

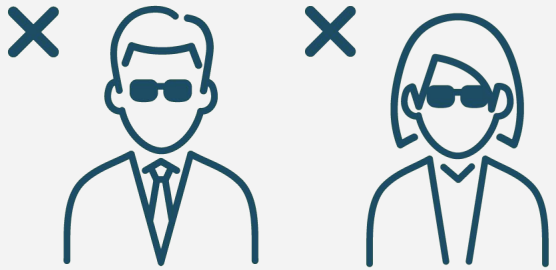
現在のバージョンでは、以下のケースで検出エラーになりやすくなります。



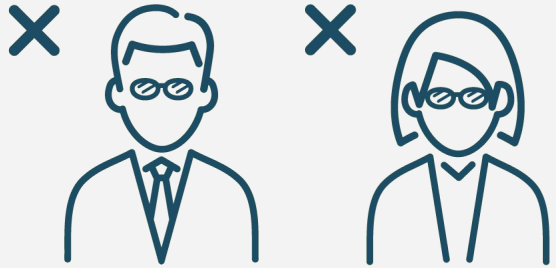
髪の毛が目にかかっている



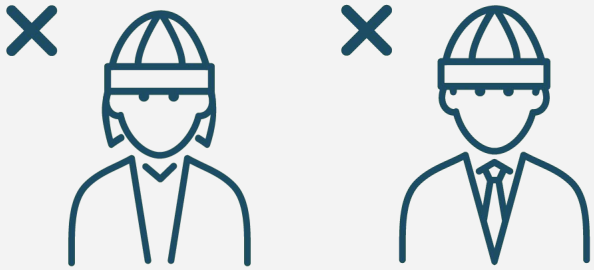
メガネフレームが目にかかっている



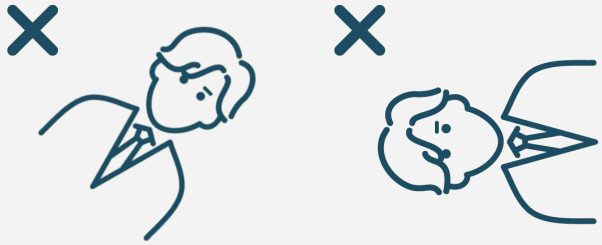
サングラス



赤外光の映り込みが激しい



帽子を深くかぶっている

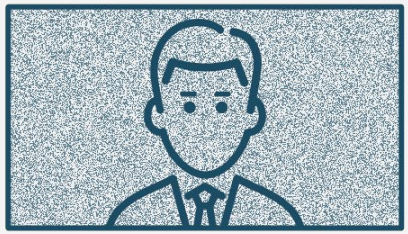


顔が傾いて撮影されている

■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

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。

△



×赤外光が不十分
(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

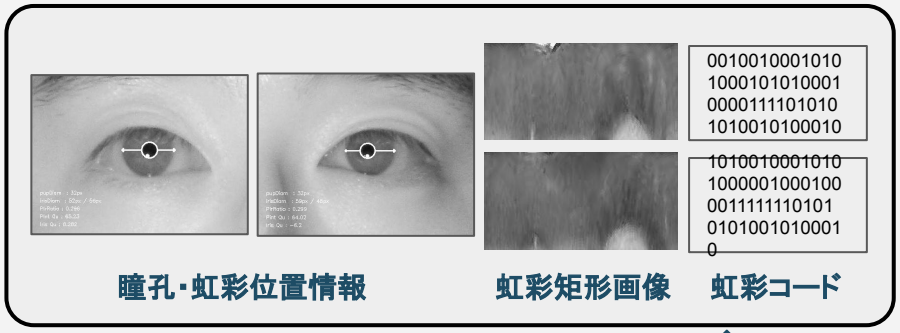
△

×画角から外れている
(顔検出を行う場合のみ)

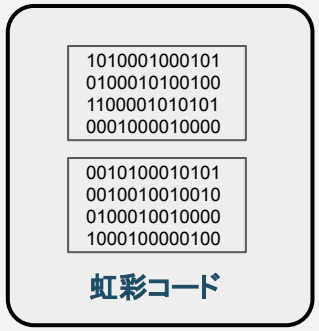
△

×画角から外れている
(顔検出を行う場合のみ)

■ライブラリ仕様 - フロー概略図



Step.5
比較



Step.4
出力

Step.1
入力

IrisProc.hpp

class
EyeDetect

格納

class
EyeData

class
Activation
Challenge

class
IrisProc

格納

class
IrisCode

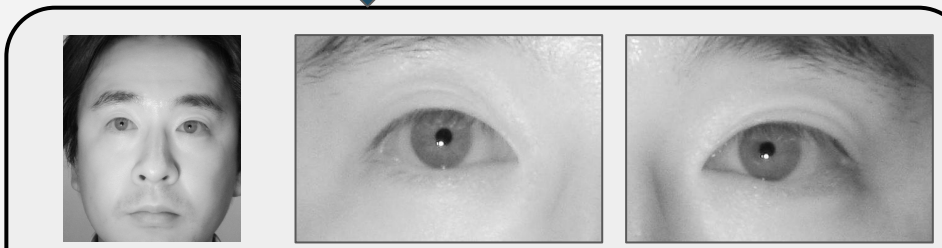
呼び出し

IrisProc.dll

ライブラリ

Step.2 出力

Step.3 入力



※短距離カメラでは、
Step.3から入力してください

■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**	IrisProcオブジェクトの実行ごとにアクティベーションを行います。
	設定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	EyeDetectクラスおよびIrisProcクラスにて、画像データを読み込ませることができます。
各種情報検出機能	**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。 目を検出できた場合は、左右判定を行います。
	瞳孔検出機能	検出された目画像から、瞳孔位置検出を行います。瞳孔検出できた場合は、瞳孔中心座標と瞳孔直径を取得することができます。
	虹彩外縁検出機能	検出された目画像から、虹彩外縁位置検出を行います。虹彩外縁検出ができた場合は、瞳孔中心から虹彩外縁までの左右それぞれの長さピクセルを取得することができます。
	虹彩模様極座標変換機能	瞳孔外縁位置から虹彩外縁位置との間に囲まれた虹彩領域を極座標変換処理により矩形領域として取得することができます。
	ピント評価機能	極座標変換を行った虹彩矩形領域を用いてピント評価を行います。
	虹彩クオリティ評価機能	極座標変換を行った虹彩矩形領域を用いて虹彩模様のクオリティ評価を行います。
	虹彩コード化機能	虹彩矩形領域を虹彩コードに変換します。

■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**	このクラスについて		顔・目位置検出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::Mat1b	originalImg	入力した元画像を取得可能です。
	cv::Mat1b	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::Mat1b	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)
	cv::Mat1b	eyeRectImgLeft	入力した元画像から、検出された左目画像のみを切り出した画像を取得可能です。
	cv::Mat1b	eyeRectImgRight	入力した元画像から、検出された右目画像のみ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	検出された左目の平均輝度値(256階調)を取得可能です。
	float	eyeRectAvgBrightRight	検出された右目の平均輝度値(256階調)を取得可能です。
	std::string	msg	検出結果メッセージを取得できます。
	返り値	関数名	説明
	void	clear()	EyeDataオブジェクト内の全ての変数の中身をリセットします。

■ライブラリ仕様 - クラス構成

構造体名	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、EyeDetect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。**
	std::string	modelFile1	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	modelFile2	
	std::string	modelFile3	目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	modelFile4	

構造体名	データ型	メンバ名	説明
struct **EyeParams**	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設定用構造体です。**本構造体をinit()関数の引数として与えます。**
	const bool	faceDetect	顔検出を行うかどうかを真偽値で設定します。
	const int	minFaceWidth	顔検出における顔の最小横幅サイズ(px)です。
	const int	maxFaceWidth	顔検出における顔の最大横幅サイズ(px)です。
	const int	minEyeWidth	目検出における目の最小横幅サイズ(px)です。
	const int	maxEyeWidth	目検出における目の最大横幅サイズ(px)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 **EyeDetect**	このクラスについて		顔・目の位置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、EyeParams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	顔が含まれた画像を引数とし、顔・目位置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

クラス名	データ型	メンバ名	説明
class **IrisCode**	このクラスについて		虹彩情報検出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::Mat1b	inputOrgImg	入力元となる画像を取得可能です。
	cv::Mat1b	checkOrgImg	顔が含まれた元画像に瞳孔・虹彩検出情報を描画した画像を取得可能です。
	cv::Mat1b	inputEyeImg	入力元となる目画像を取得可能です。
	cv::Mat1b	checkEyeImg	入力元となる目画像に、瞳孔・虹彩検出位置情報を描画した画像を取得可能です。
	double	eyeRectAvgBright	入力された目画像の平均輝度値(256階調)を取得可能です。
	cv::Point	pupilCenter	検出された瞳孔中心位置座標(x,y)を取得可能です。
	int	pupilDiameter	検出された瞳孔直径サイズ(px)を取得可能です。
	int	irisSizeLeft	瞳孔中心から検出された虹彩外縁左端までのサイズ(px)を取得可能です。
	int	irisSizeRight	瞳孔中心から検出された虹彩外縁右端までのサイズ(px)を取得可能です。
	double	pupilIrisRatio	瞳孔径虹彩径比を取得可能です。算出式は瞳孔直径(px)÷虹彩直径(px)です。
	cv::Mat1b	polarImg	虹彩領域を極座標変換して得られた矩形画像を取得可能です。
	double	focusQuality	虹彩矩形領域のピント評価値を取得可能です。
	cv::Mat1b	polarImgLocal	虹彩矩形領域からクオリティを判定するための局所領域画像を取得可能です。

■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 **IrisCode**	このクラスについて		虹彩情報検出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::Mat1b	irisCodeImg	虹彩コード化元画像を取得可能です。
	double	irisQuality	虹彩クオリティ値を取得可能です。
	const unsigned char*	payload	虹彩コード本体を取得可能です。
	const unsigned int	payloadLength	虹彩コード本体の長さを取得可能です。
	std::string	msg	検出結果メッセージを取得可能です。
	double	compare()	IrisCodeオブジェクトのpayload同士比較し、ハミング距離を求めます。
	void	clear()	IrisCodeオブジェクト内の全ての変数の中身をリセットします。

■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 **IrisParams**	この構造体について		虹彩情報を取得するための処理におけるパラメータを読み込ませるために、IrisProc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。
	const int	minEyeBrightness	処理を行うに当たっての目領域の最低平均輝度値（明るさ）を設定します。
	const int	minIrisDiameter	検出する虹彩の最小半径サイズ(px)を設定します。
	const int	maxIrisDiameter	検出する虹彩の最大半径サイズ(px)を設定します。
	const double	minFocusQuality	虹彩コード化を行うために必要な最小ピント評価値を設定します。 マイナスの値をとることも可能です。
	const double	minIrisQuality	虹彩コード化を行うために必要な最小虹彩クオリティ値を設定します。 マイナスの値をとることも可能です。

■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 **IrisProc**	このクラスについて		瞳孔・虹彩位置情報及び虹彩コード生成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。引数にParams構造体をとります。
	bool	registerIrisCode	メモリ確保したIrisCodeクラスを本クラスに登録するための関数です。
	bool	process	目画像を第1引数とし、虹彩情報・虹彩コード生成を行うクラスです。 **第2引数以降は、目検出前の元画像に本クラスでの検出結果を描画したい場合に利用しますが省略可能です。** 検出結果はすべてIrisCode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		EyeDetect / IrisProc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が返ることにより該当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

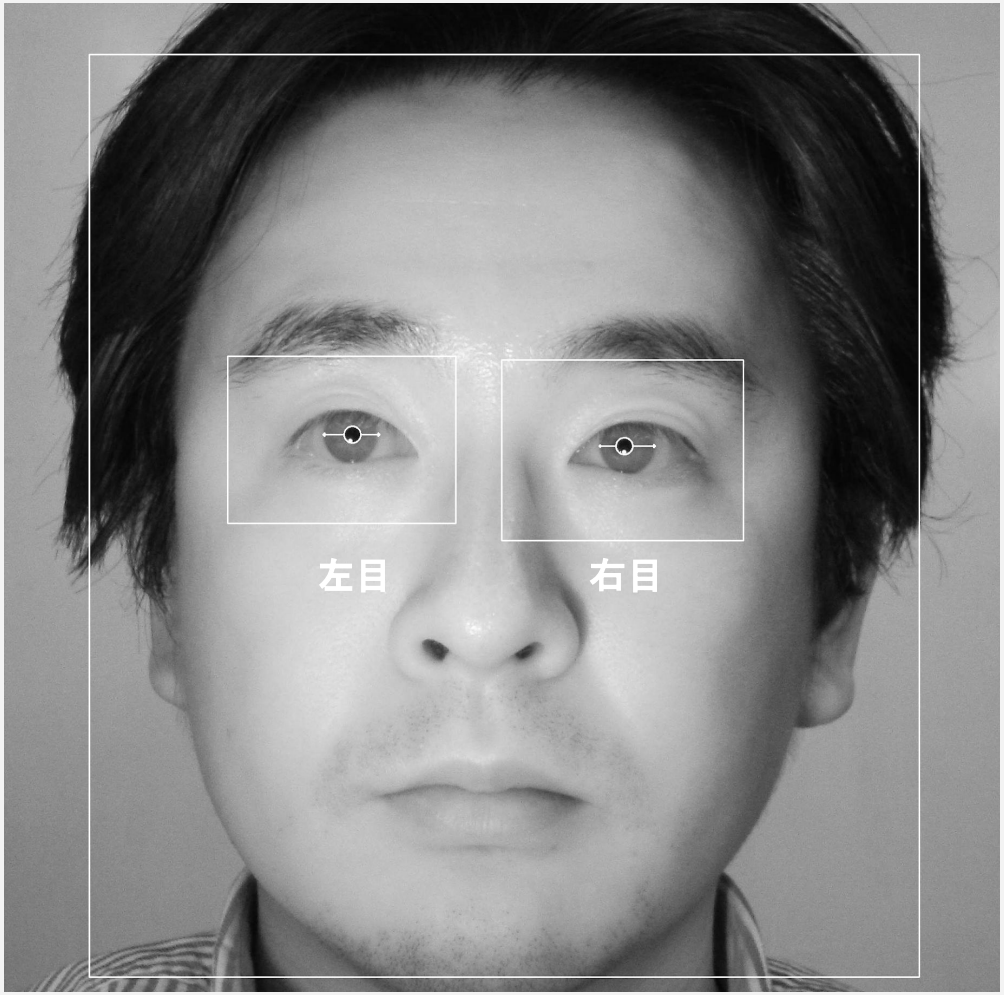
クラス名	データ型	メンバ名	説明
class **ActivationChallenge**	std::string	config()	アクティベーション情報および本ライブラリの情報を取得可能です。
	bool	checkToken()	本関数の引数にとるトークン文字列が、ライブラリに埋め込まれた対応トークンとマッチするかどうかのみを判定します。 **本関数からtrueが返るだけではアクティベーションは完了しません。**
	bool	validToken()	本関数の引数にとるトークン文字列が、ライブラリに埋め込まれた対応するトークン情報を読み取り、有効期限内かどうかをチェックするための関数です。有効期限内であれば本関数からtrueが返り、該当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「20201231」であれば、2020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
EyeDetect*	newEyeDetect()	EyeDetect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、EyeDetect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	releaseEyeDetect()	EyeDetectオブジェクトをメモリ開放させるための関数です。 EyeDetect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。
IrisProc*	newIrisProc()	IrisProc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、IrisProc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	releaseIrisProc()	IrisProcオブジェクトをメモリ開放させるための関数です。 IrisProc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 - 機能一覧

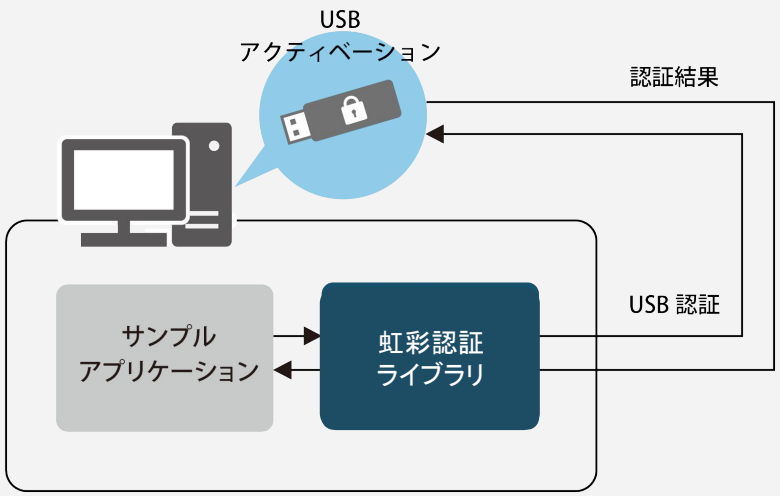
入力画像に対する左右目の判定は、画像向かって左側の目を左目、
画像向かって右側の目を右目として出力します。



■ライブラリ仕様 - アクティベーション

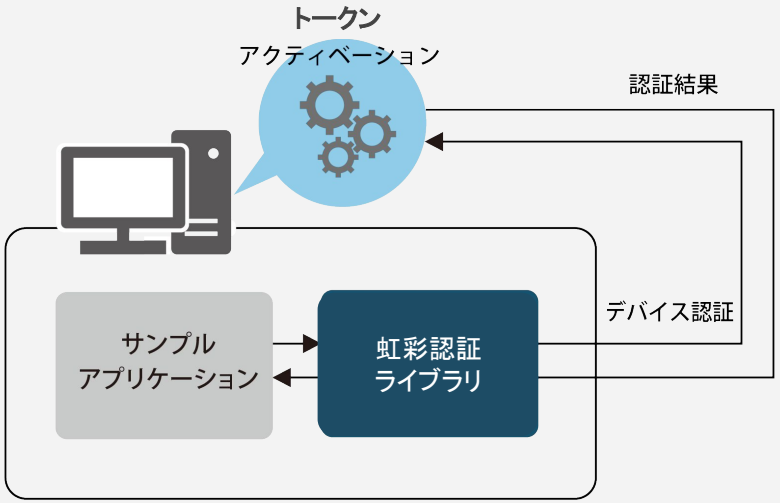
開発版ライセンスでは、本ライブラリをご利用いただくにあたって、アクティベーションが必要となります。アクティベーションには、USB方式と有効期限を利用したトークン方式の2タイプがあります。

◆ USBアクティベーション



有償（USBドングル5本まで対応可能）
※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

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 IR Image(channels()==1)	モノラルチャンネル以外の画像が入力されました
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）
Error: Failed to detect eyes	目検出に失敗しました

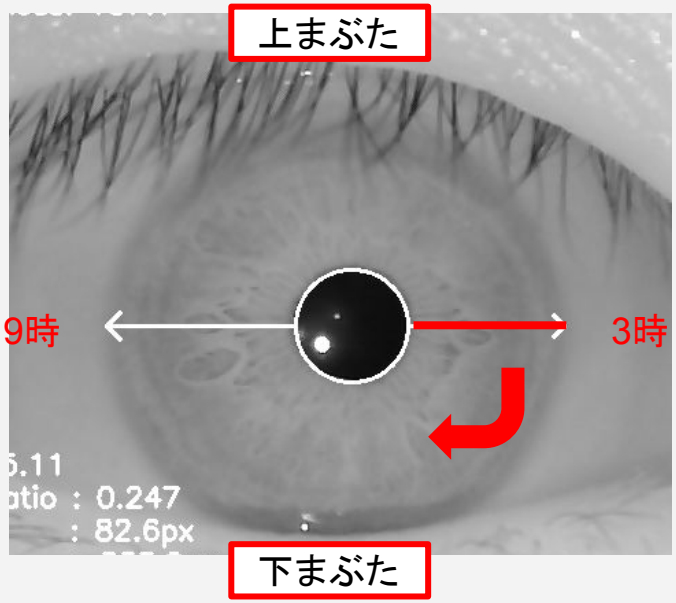
■message一覧

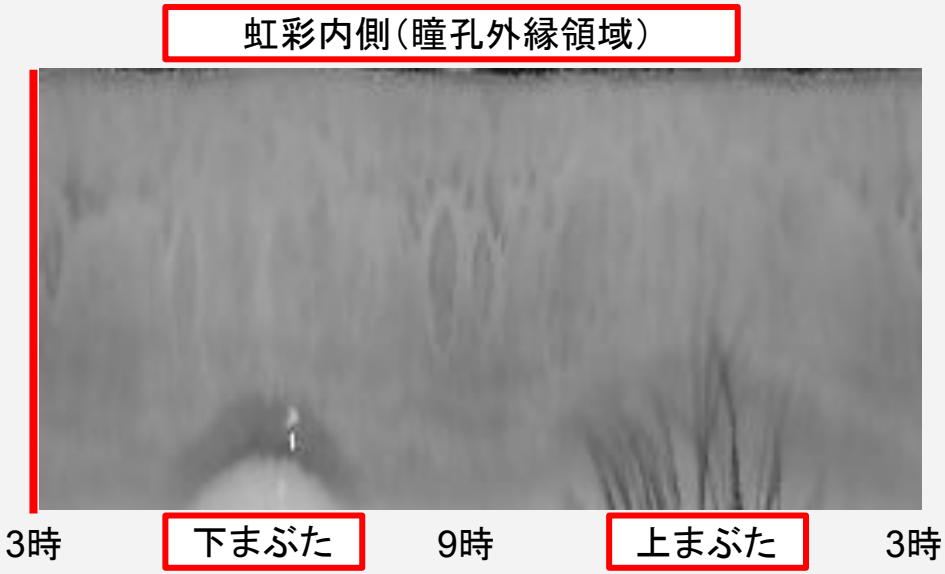
IrisProc型のメンバ変数であるmsgにて取得できる主なメッセージは以下のいずれかとなります。その他のエラーが発生した場合はお問合せください。

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: registerIrisCode() first	process()実行前に、registerIrisCode()を実行してください
Error: Empty input eye image	入力画像が存在しません
Error: Input only CV_8UC1(8bit GrayScale) image	8bitグレースケール画像以外の画像形式が読み込まれました
Error: Lack of eye area brightness	目領域の明るさ不足です
Error: Failed to detect pupil	瞳孔検出ができませんでした
Error: Failed to detect iris	虹彩位置情報の検出ができませんでした
Error: Focus quality is low	ピント評価値が閾値を下回りました
Error: Iris quality is low	虹彩クオリティ値が閾値を下回りました

虹彩品質評価の流れ

■虹彩画像の品質評価の流れ





①瞳孔外縁・虹彩外縁を検出

当社独自のアルゴリズムで、瞳孔の外縁と虹彩の外縁を検出します。

②極座標変換で長方形に整形

検出した虹彩を図1の赤線部である3時の位置から時計回りに走査し、極座標変換により虹彩領域のみをくり抜いた長方形画像を取得します。

■虹彩画像の品質評価の流れ

切り出し

③虹彩局所領域を切り出し

極座標変換の素画像では、アジア人には細目、長まつげが多いため、虹彩模様のみで占められる局所領域を指定して、切り出します。瞳孔領域の影響もあるため上端も除外します。

④解析対象虹彩領域の決定

切り出した虹彩のみを解析対象虹彩領域と決定し、クオリティ判定は本領域に限って評価・判定処理を行います。

図4

空間周波数フィルタ

- **高周波成分**
- **コントラストを評価**

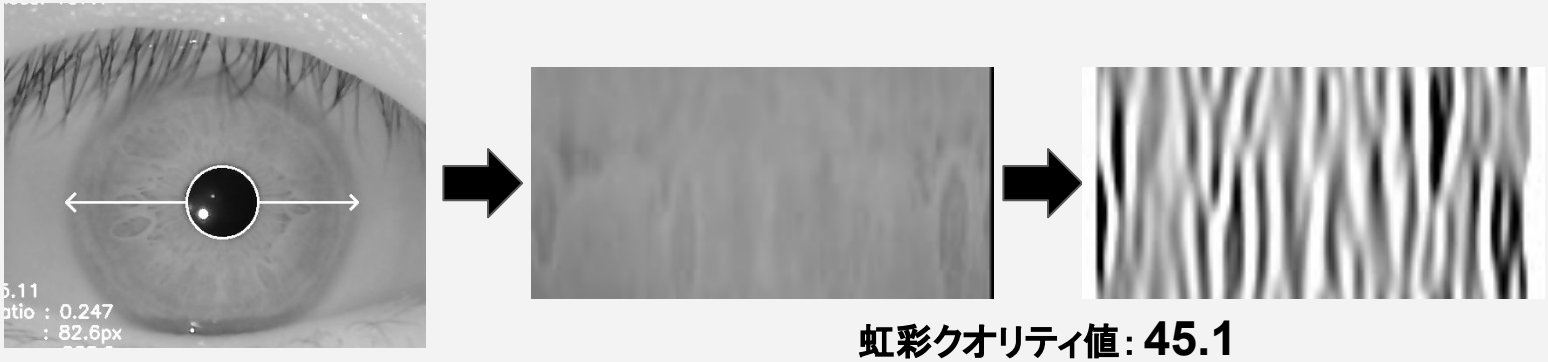
⑤空間周波数フィルタ

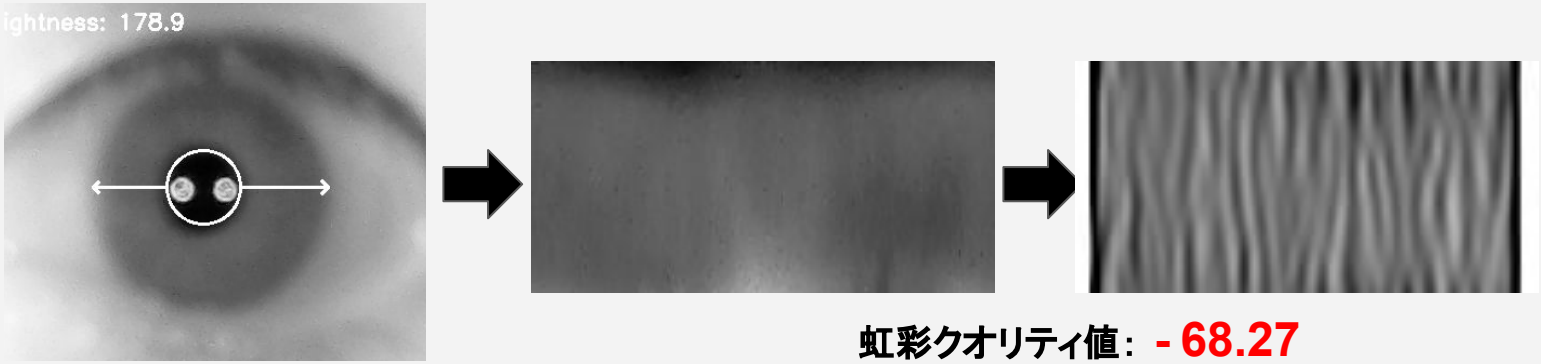
虹彩認証向け空間周波数フィルタをかけ、高周波成分を検出し、際立たせます。
（図4を得ます。）

⑥ フィルタ画像評価

8bitグレースケール256階調のフィルタ画像において、高周波成分及びコントラストを元に品質を評価します。

■虹彩画像の品質評価の流れ





最低基準値と乖離値の考え方

虹彩認証の他人弁別性を向上するためには、鮮明な撮影虹彩画像から、虹彩コードを精緻に取得し、
他人と重複しない本来の虹彩模様を、デジタルデータとして取得することが重要です。

合焦度が高く、投影赤外波長や照度が整った環境では、虹彩模様の窪みがはっきりと撮影されます。
窪みがはっきり取れれば、エッジが数多く検出できます。
虹彩模様の撮影の鮮明さは、精緻な虹彩コード取得において重要性が高くなります。

また、赤外光の照度不足の影響や、合焦度が低い場合は、フィルタ特性上、暗いフィルタ画像になるため、
良質な虹彩画像であるためには、十分なコントラストが必要にあります。