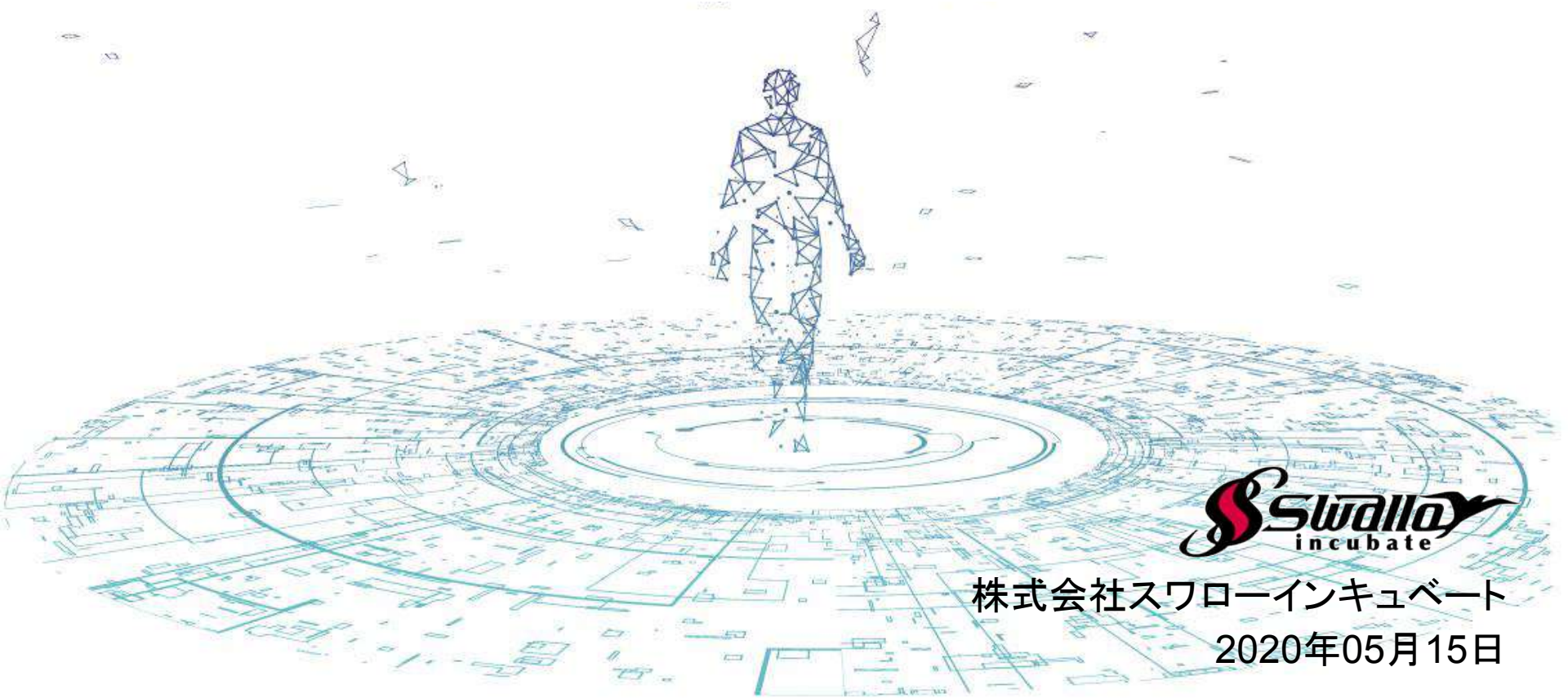
# 生体(ライブネス)判定技術
## SDK/インターフェース仕様書



株式会社スワローインキュベート

2020年05月15日

# ■はじめに

生体判定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。

・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**

**生体判定技術　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<br>iOS / Android OS | |
| CPUアーキテクチャ | x86系64bit、x86、Armv7系、Armv8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画像のみ** |
| 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 **横1280 x 縦960 px** / スマホ版 **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><br>**最低 width 300px 以上** |
| 撮影距離 | <カメラから顔までの距離><br>**～1m程度まで**<br>※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 | **顔検出　:2名以上**(2名以上の顔を検出した場合は以降の処理を行いません )<br>**目検出以降:1名のみ** |

※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)

## 生体判定ライブラリ

GitHub

C++コード / ビルド手順

## 生体判定サンプルアプリ


**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** | LivenessCheck.hpp（インターフェースファイル）<br>libLivenessCheck.dylib<br>サンプルアプリ（C++） |
| **Windows OS** | LivenessCheck.hpp<br>LivenessCheck.dll<br>LivenessCheck.lib<br>サンプルアプリ（C++） |
| **各種Linux OS**<br>**(Raspbian OS含む)** | LivenessCheck.hpp（インターフェースファイル）<br>libLivenessCheck.so<br>サンプルアプリ（C++） |
| **iOS** | LivenessCheck.FrameworkLivenessCheck.hpp含む）<br>サンプルアプリ（Objective-C ラッパーサンプルコード込み）<br>iOS向けOpenCV + OpenCV Contribビルド方法マニュアル |
| **Android OS** | LivenessCheck.hpp<br>libLivenessCheck.so（arm64-v8a / armeabi-v7a / x86 / x86_64）<br>サンプルアプリ（JNI ラッパーサンプルコード込み）<br>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150ms (1280 x 960 px)**<br>（CPU: Core i7 /  メモリ16GB のマシン検証時） |

※別途プロセス初期化の時間がかかります。

### 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<br>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

## 現在のバージョンでは、以下のケースではエラーになることが想定されます

| 顔検出エラー | |
|---|---|
| **原因** | **詳細** |
| **周囲が暗い・明るすぎる** | 周囲の照度が暗いまたは明るすぎることにより、顔検出に失敗しているケースが想定されます。<br>適度な顔領域の平均輝度値は、8bit256階調グレースケールで、80～180程度 |
| **顔領域が隠れている** | マスク、サングラスなどの着用により、顔検出に失敗しているケースが想定されます。 |
| **撮影距離と顔領域のサイズ** | 撮影距離が遠いほど、顔検出サイズが小さくなり、目検出や目検出のための十分なサイズが取れないケースが想定されます。顔検出サイズは、横幅300px以上を推奨しています。 |
| **顔が横向きになっている** | カメラを正面とした場合、左右約30度を超えて横を向いた場合に顔検出に失敗するケースが想定されます。 |

| 目検出エラー | |
|---|---|
| **原因** | **詳細** |
| **目の髪の毛がかかっている** | 目に髪の毛がかかっている場合に、目検出に失敗するケースが想定されます。 |
| **まばたきをしている** | まばたきや目を閉じているフレームでは、目検出に失敗するケースが想定されます。 |
| **細目** | 上まぶたと下まぶたとの距離が10pxを下回る場合は、目検出に失敗するケースが想定されます。 |
| **光の映り込み** | 黒目(虹彩)の領域やメガネに外光などが強く映り込んでいる場合、目検出に失敗するケースが想定されます。 |
| **大きめのほくろ** | 目の近辺に瞳(虹彩)サイズに近似したホクロなどがある場合は、瞳と誤検出してしまうケースが想定されます。 |

# ■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** | LivenessCheckオブジェクトの実行ごとにオフラインまたはオンラインによる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** | LivenessCheck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。 |
| | **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。<br>また検出した顔の数を取得することができます。 |
| | **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。<br>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。 |
| | **顔中心x座標検出機能** | 検出された顔領域画像から顔中心x座標を検出します。 |
| | **くちびる位置検出機能** | 検出された顔領域画像からくちびる位置を検出します。<br>取得できる位置は、くちびる左右端と中央上下端の点座標です。 |
| | **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

| 項目 | 機能項目 | 要件 |
| --- | --- | --- |
| **生体情報機能** | **顔の向き検出** | 各種センシング情報をもとに、顔の向きを検出します。 |
| | **目の動き検出** | 各種センシング情報をもとに、目の向きを検出します。 |
| | **目の開閉状態判定** | 各種センシング情報をもとに、目の開閉状態を判定します。 |
| | **くちびる開閉状態検出** | 各種センシング情報をもとに、口唇の開閉状態を検出します。 |
| | **メガネ装着判定** | 各種センシング情報をもとに、メガネ装着有無を検出します。 |

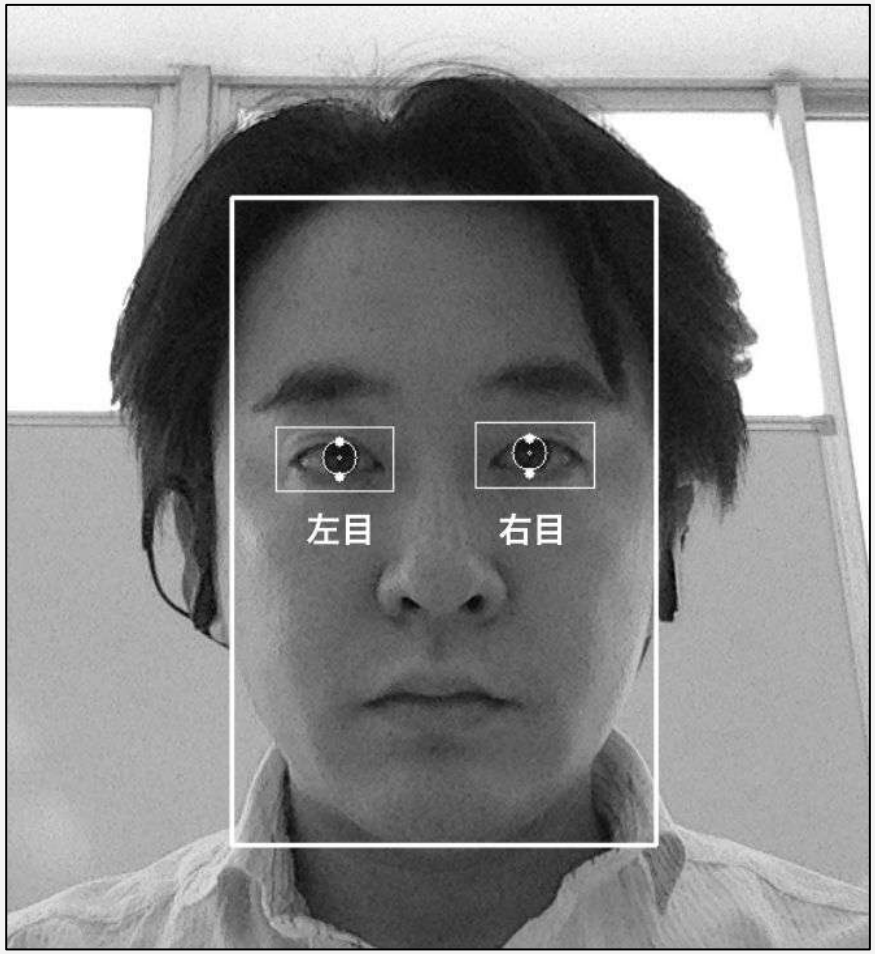
# ■ライブラリ仕様 - 生体判定機能例一覧

顔の向き検出、目の動き検出、目の開閉状態判定、くちびる開閉状態検出、メガネ装着判定を、自由に組み合わせてユーザーに指示することにより、生体かどうかの判定を行うことが可能です。

| 項目 | 生体判定手段 | 詳細 |
|---|---|---|
| **生体判定機能** | 首ふり動作検出 | ユーザーに顔の向きを左右に動かす指示を行うことで生体かどうかを判定させることが可能です。 |
| | アイコトバ口唇検出 | ユーザーに指定のアイコトバを発話させることで、くちびるの開閉状態検出を行い、生体かどうかを判定させることが可能です。 |
| | 左右の目の動き検出 | 目の動きを左右正面いずれかに指示させることで生体判定可能です。 |
| | まばたき検出 | 所定の回数まばたきを指示させることで生体判定可能です。 |
| | メガネ脱着検出 | メガネ装着ユーザーに対して、メガネの脱着を指示することで生体判定可能です。 |
| | ランダムアクション | ユーザーに、顔の動き、目の動き、まばたき、くちびる開閉などを認証のたびに、毎回ランダムな順番で行わせることで、ビデオなりすましから保護します。 |

# ■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 **BioData** | このクラスについて | | 生体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。 |
| | int | faceCenterX | 顔矩形領域において、顔中心位置となるx座標を取得可能です。 |
| | cv::Mat | faceRectSecontImg | 複数顔が検出された場合、2番目に大きなサイズの顔矩形領域画像を取得可能です。 |
| | cv::Rect | faceRectSecontArea | 複数顔が検出された場合、2番目に大きなサイズの顔矩形領域情報を取得可能です。(x座標,y座標,width,height) |
| | int | faceDetectNum | 検出された顔の数を取得可能です。2以上の場合は以降の検出処理を中止します。 |

# ■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 **BioData** | cv::Mat | 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::Mat | 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。 |
| | bool | eyeGlassStatus | メガネ装着の有無を判定可能です。 |
| | int | irisRadiusLeft | 検出された左目瞳(虹彩)半径情報を取得可能です。 |
| | int | 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。 |

# ■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 **BioData** | 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::vector <cv::Point> | lipFeaturePosition | 検出されたくちびる位置情報を取得可能です。<br>vectorコンテナには、0=唇左端点、1=唇中央上端点、2=唇右端点、3=唇中央下端点の順に、それぞれの位置情報がcv::Point型で格納されます。 |
| | 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() | BioData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 **BioData** | double | faceDirection | 顔向き具合を表すパラメータです。正面を0とし、顔を傾け具合に応じて、<br>左に−15.0～0まで、右に0～15.0の値を返します。 |
| | int | eyeStatusLeft | 目の開閉状態ステータスです。<br>0は目検出エラー、1は閉目状態、2は開目状態を表します。 |
| | int | eyeStatusRight | |
| | double | eyeDirection | 目の左右向き具合を表すパラメータです。正面を0とし、目の左右における向き具合に応じて、左に−15.0～0まで、右に0～15.0の値を返します。 |
| | double | lipStatus | 口の開閉状態ステータスです。0はくちびる検出エラーです。<br>上限値を1.0とし、0.0～1.0の間で値を返します。<br>通常の閉口状態での値は、個人差はありますが、0.5未満におさまります。 |

| 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、ランドマーク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、LivenessCheck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 | 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。 |
| | std::string | landmarksConfigFileName | ランドマーク検出に用いるモデルファイル名を読み込む変数です。 |

# ■ライブラリ仕様 - クラス構成

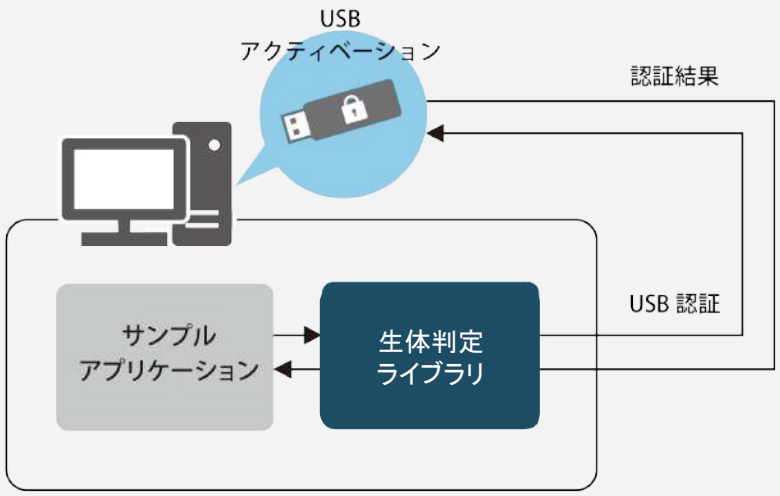
| クラス名 | データ型 | メンバ名 | 説明 |
|---|---|---|---|
| class **LivenessCheck** | このクラスについて | | 生体判定に必要な顔・目周辺情報を検出するための実行クラスとなります。 |
| | bool | init | 初期化を行うメンバ関数です。引数にExternalFilePath構造体をとります。 |
| | bool | registerBioData | メモリ確保したBioDataクラスを本クラスに登録するための関数です。 |
| | bool | process | cv::Matを引数とし、顔・目周辺情報検出を行うクラスです。<br>検出結果はすべてBioData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 | | LivenessCheck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、LivenessCheck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。<br>本関数からtrueが返ることにより、LivenessCheck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。 |

# ■ライブラリ仕様 - アクティベーション

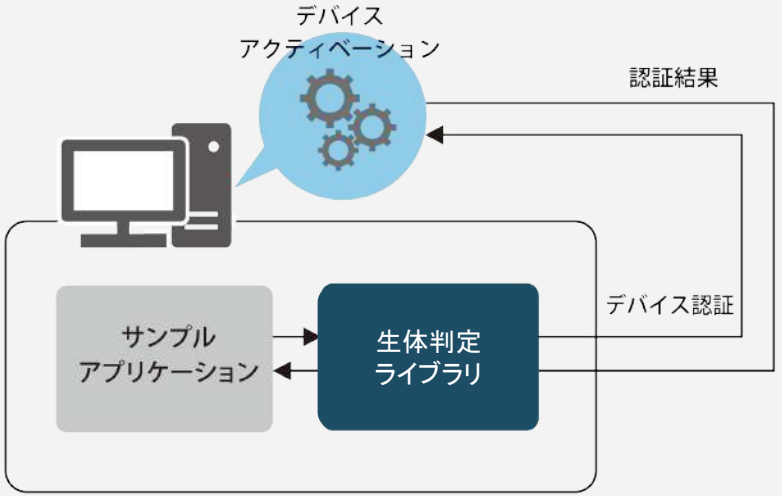
開発版ライセンスでは、本ライブラリをご利用いただくにあたって、アクティベーションが必要となります。アクティベーションには、USB方式と有効期限を利用したデバイス時刻取得方式の2タイプがあります。

## ◆ 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

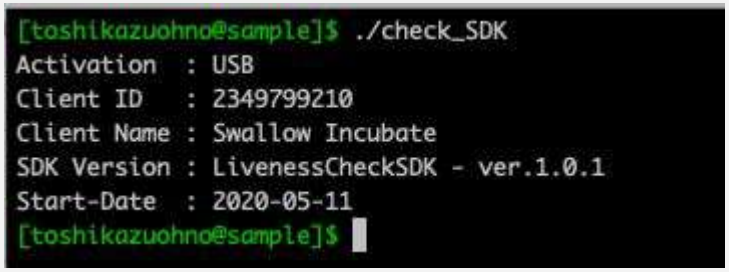



**無償**
※主にスマホ/タブレット端末で利用企業様向けとなります

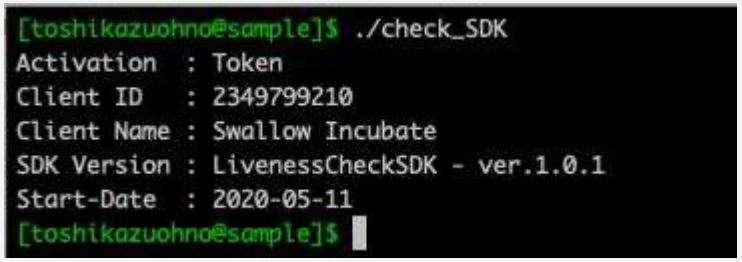
# ■ライブラリ仕様 - アクティベーション情報

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



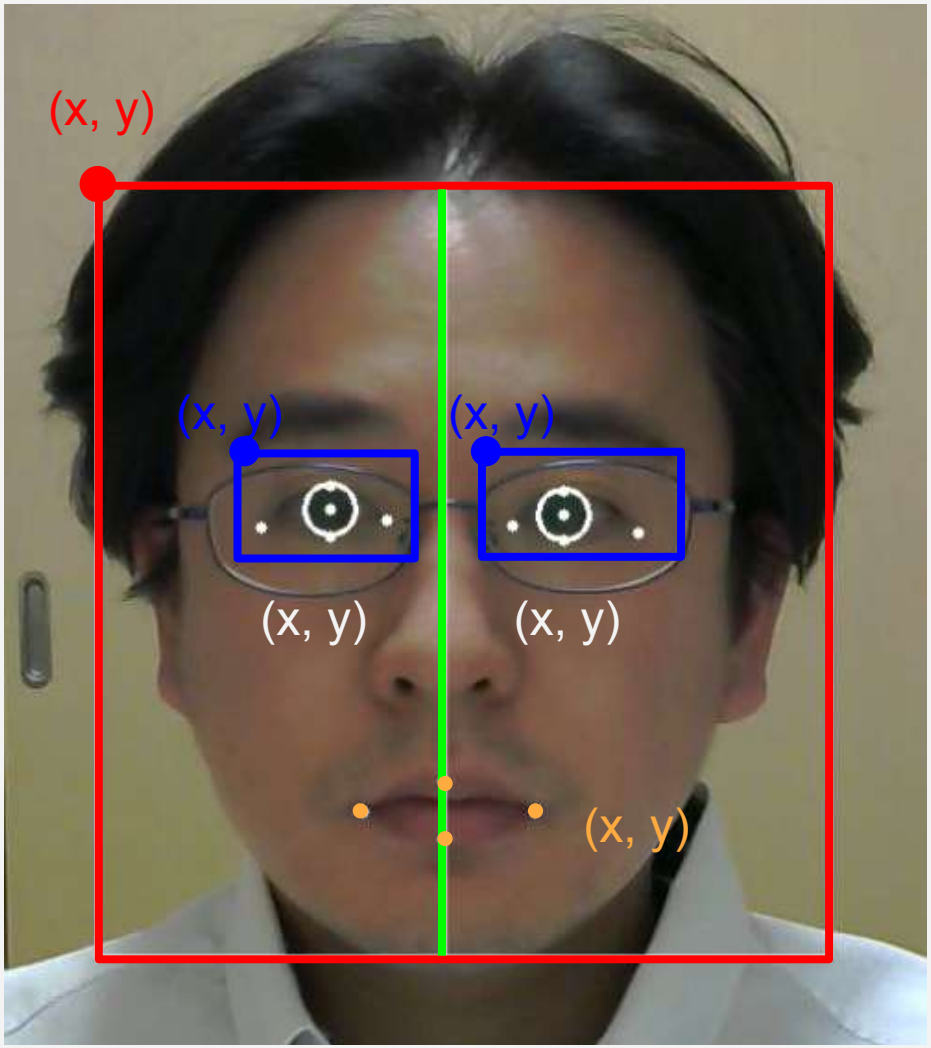

### ◆ デバイスアクティベーション版 config() 実行結果例



# 出力データ詳細

# ■出力データ図解

BioData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, width, height)

= faceCenterX

= lipFeaturePosition

# ■message一覧

BioData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: registerBioData() first | process()実行前に、registerBioData()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画像を入力してください |
| Error: Failed to detect face(1001) | 顔検出に失敗しました（横幅サイズ不足：初期値300px） |
| Error: Failed to detect face(1002) | 顔検出に失敗しました（横向き） |
| Error: Failed to detect face(1003) | 顔検出に失敗しました（顔が存在しない） |
| Warning: Multiple face detected | 【警告】複数の顔が検出されました |
| 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80以上 － 8bit256階調） |

# ■message一覧

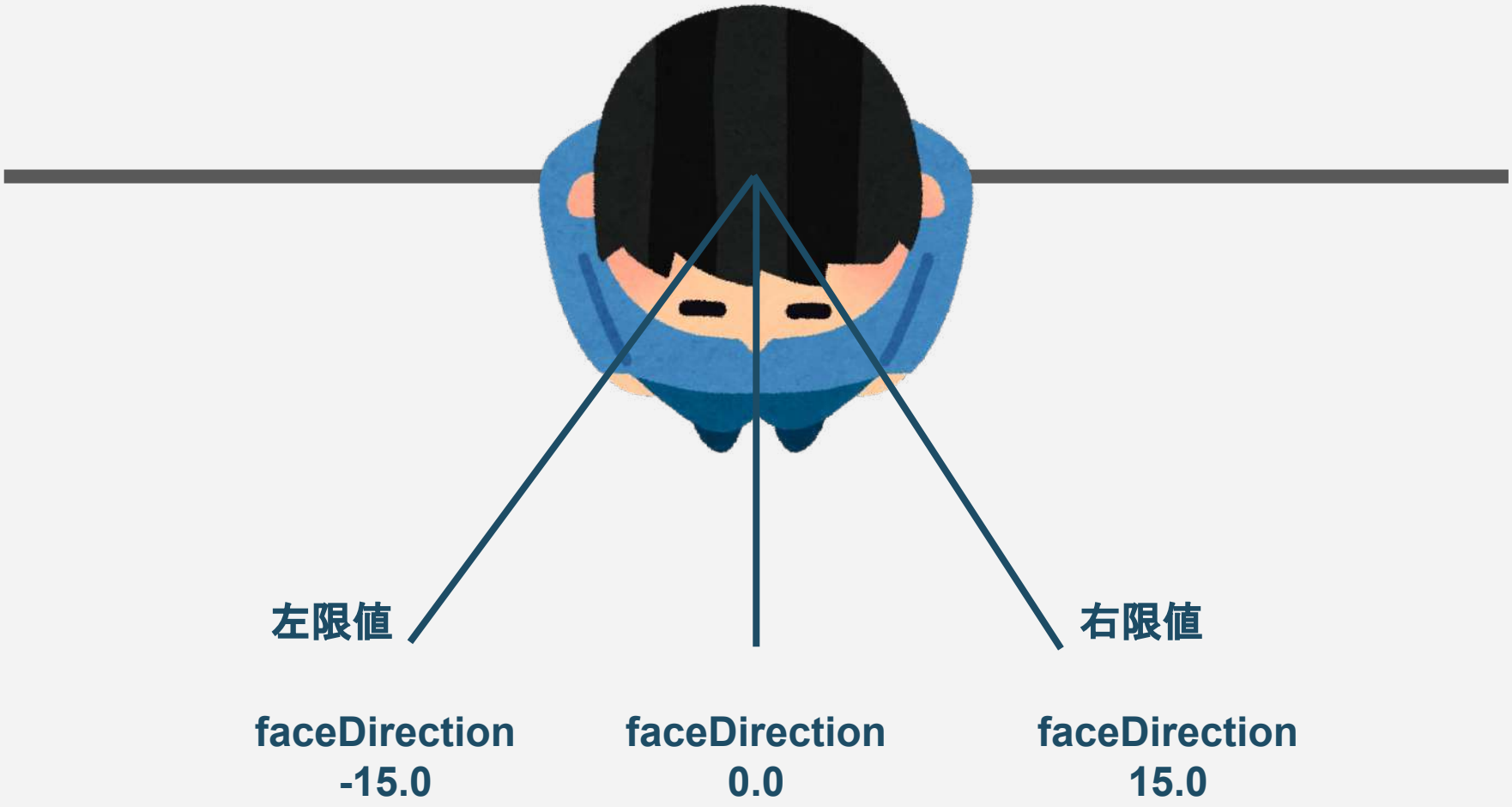
BioData型のメンバ変数であるmsgにて取得できる主なメッセージは以下のいずれかとなります。その他のエラーが発生した場合はお問合せください。

| 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: Failed to detect lip | くちびる位置検出に失敗しました |

# ライブネス判定データ

# ■ライブネス判定データ - 顔の向き検出

顔の向きは、BioDataクラスのメンバ変数であるfaceDirectionより取得できます。
faceDirectionの値は、カメラに対して正面を0として、顔向きの変化に応じて左側へ 0 ～ -15.0、右側へ0 ～ 15.0の間で定量化されます。



# ■ライブネス判定データ - 目の動き検出

目の向きは、BioDataクラスのメンバ変数であるeyeDirectionより取得できます。
eyeDirectionの値は、カメラに対して正面を0として、目の左右への向き具体の変化に応じて左側へ -15.0～0、右側へ0 ～ 15.0の間で定量化されます。

**左限値**

**eyeDirection**
**-15.0**

**中央値**

**eyeDirection**
**0.0**

**右限値**

**eyeDirection**
**15.0**

# ■ライブネス判定データ - 目の開閉状態判定

目の開閉パラメータは、BioData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**

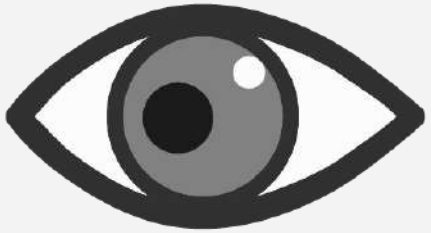
Error

**目検出エラー**

**eyeStatus = 1**

**目が閉じている**

**eyeStatus = 2**

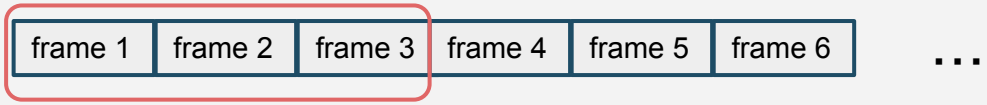


**目が開いている**

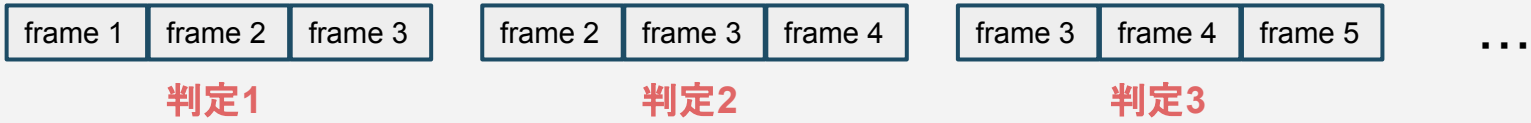
# ■ライブネス判定データ - まばたき判定例

生体判定ライブラリは、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。

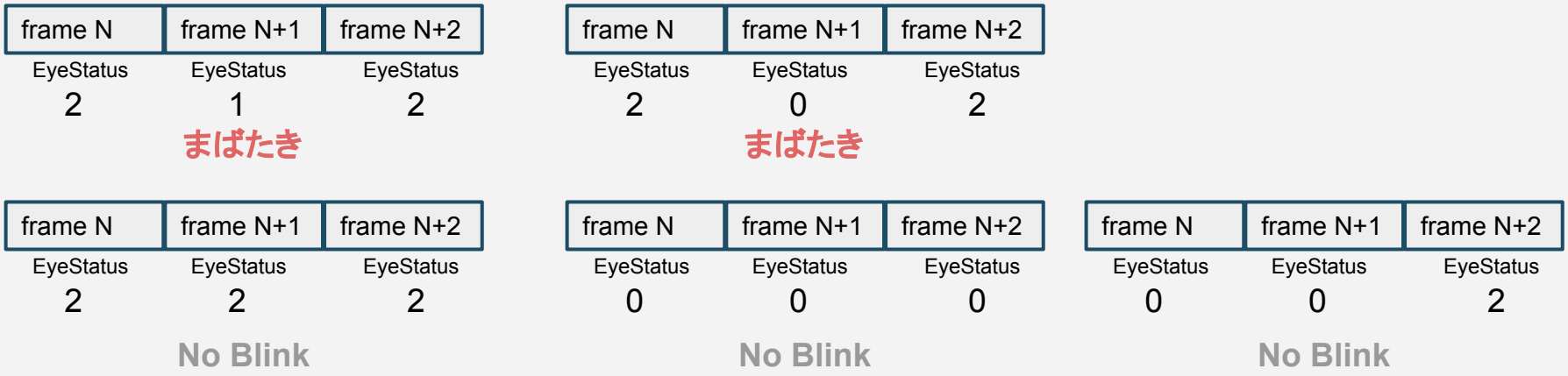
**<まばたき判定に用いるフレーム数 >** - 例）3つずつ




**<まばたき判定に使う時間フレーム >** 例）連続する3フレーム

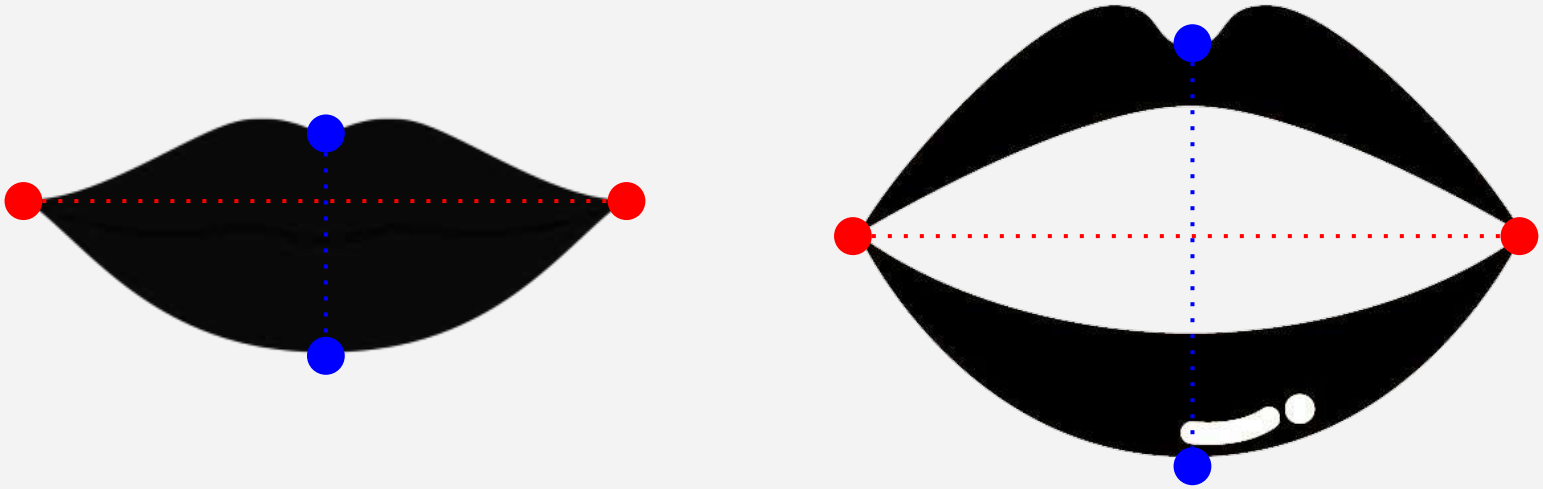



**<まばたき判定アルゴリズム >（例）**



# ■ライブネス判定データ - くちびる開閉状態検出

くちびる開閉状態パラメータは、BioDataクラスのメンバ変数であるlipStatusより取得できます。lipStatusの値は、くちびるの左右端距離とくちびるの中央上下端距離との比率により求められます。下限値は個人差がありますが、上限値は1.0としています。くちびる位置が正常に検出できなかった場合は、0を返します。



**lipStatus = 中央上下端距離 ÷ 左右端距離**