OpenCV プロバイダ

デンソー ロボット視覚

(DENSO Robot Imaging Library)

Version 1.5.5

ユーザーズ ガイド

December 24, 2015

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2007-01-30	初版
1.0.1.0	2007-04-06	OpenCV 関数の追加. OcvTester のバグ修正
1.1.0.0	2007-08-06	パターンマッチング機能の強化, イメージ ID の開始を 0 から 1 へ変更; 各種コマンド追加
1.2.0.0	2007-11-21	三角測量機能の拡張, 各種コマンド追加
1.2.1.0	2007-12-05	カメラ設定機能の拡張, ブロブ機能追加
1.2.2.0	2008-02-06	ビデオ制御モードの取得・設定コマンド追加, 'Database' オプションの追加, エラー一覧追加
1.3.0.0	2008-02-29	内積・外積コマンド追加
1.3.1.0	2008-05-07	各種コマンド追加, MatchTemplate コマンド仕様変更
1.3.2.0	2008-09-17	各種コマンド追加, MatchTemplate2, MatchShapes2, Undistort2 コマンド仕様変更
1.3.3.0	2009-01-20	各種コマンド追加, OcvTester にキャリブレーションウィザードを追加, メッセージ転送機能追加, デフォルトカメラ機能追加
1.3.4.0	2009-06-03	各種コマンド追加
1.3.5.0	2010-02-12	各種コマンド追加, 各コマンドの回転方向を統一, CamShift 廃止, エラーコード追加, サンプルの追加
1.4.0.0	2010-7-01	CaoController::get_VariableNames 実装
1.4.1.0	2010-11-02	カメラキャリブレーションのコマンドにカメラ ID パラメータを追加
1.4.1.1	2011-06-13	キャプチャウィンドウ追加
1.4.2.0	2012-01-11	各種コマンド追加, SetRobCalDat 誤植修正
1.4.2	2012-07-17	ドキュメントのバージョンルールを変更
1.4.3	2012-09-06	データベースファイル（mdb）のリカバリ機能追加
	2012-09-20	SetCameraCtrl の説明追記
1.4.4	2012-10-23	GetCameraFormatList, GetCameraFormat, SetCameraFormat コマンド追加
	2012-12-05	[バグ修正] メッセージ転送機能のメモリリーク
1.4.5	2013-03-13	各種コマンド追加 SetCameraFrameRate, GetCameraFrameRate, IsUpdated, ClearUpdated
	2013-04-01	コマンド独自エラー追加 CaoFile::get_Attribute 実装
1.5.0	2013-07-22	FrameRate オプション追加 拡張カメラ機能追加

		名称変更 ORiN Vision ⇒ DENSO Robot Imaging Library 誤植修正:BlobMatchTemplate
	2013-09-24	CARD コマンド追加
1.5.1	2013-11-18	拡張カメラコマンド追加
1.5.2	2014-05-27	CARD コマンド修正 拡張カメラコマンド追加 Canon WebView Livescope カメラ対応
1.5.3	2014-10-01	CARD コマンド追加
1.5.4	2015-02-25	ADO のバグ修正 一部エラーコードの見直し
	2015-03-24	CARD コマンドの中間画像の説明追加
	2015-11-04	Cross の説明の誤植修正
1.5.5	2015-12-24	カメラキャリブレーション ID の数をロボット, カメラそれぞれ 100 に拡張 変数追加:@EXT_CAM_COUNTS, @CAM_CAL_MAX, @ROB_CAL_MAX
	2019-05-15	QRDecode コマンドの読み取り失敗時のエラーコードを追記

【対応機器】

機種	バージョン	注意事項

【ご注意】

本プロバイダを使用する場合は “DENSO Robot Imaging Library”ライセンスが必要です.

目次

1. はじめに

DENSO Robot Imaging Library (以下 RIL) とはロボット・ビジョン・システムのプラットフォームとして ORiN (Open Robot interface for the Network)と OpenCV (Open Source Computer Vision Library)1を融合したツールセットです. このプラットフォームは単に 2 つの技術を活用したというだけではなく, ORiN の一プロバイダとして OpenCV を融合しています. その為, ORiN の一貫したプログラミングモデルに従って画像処理することができアプリケーションプログラムの記述性・可読性が向上させることができます. また, モデル管理などの共通処理をプロバイダ内で実装することで高速処理と使い易さを両立させています.

RIL の目的は難易度が中程度の視覚アプリケーションを簡単に安価に作れる視覚機能を提供することです. その為に, ソフトウェアだけの画像処理を実現し, カメラも市販されているあらゆるカメラを使えるようにしました. ORiN Vision のアーキテクチャを下記に示します.

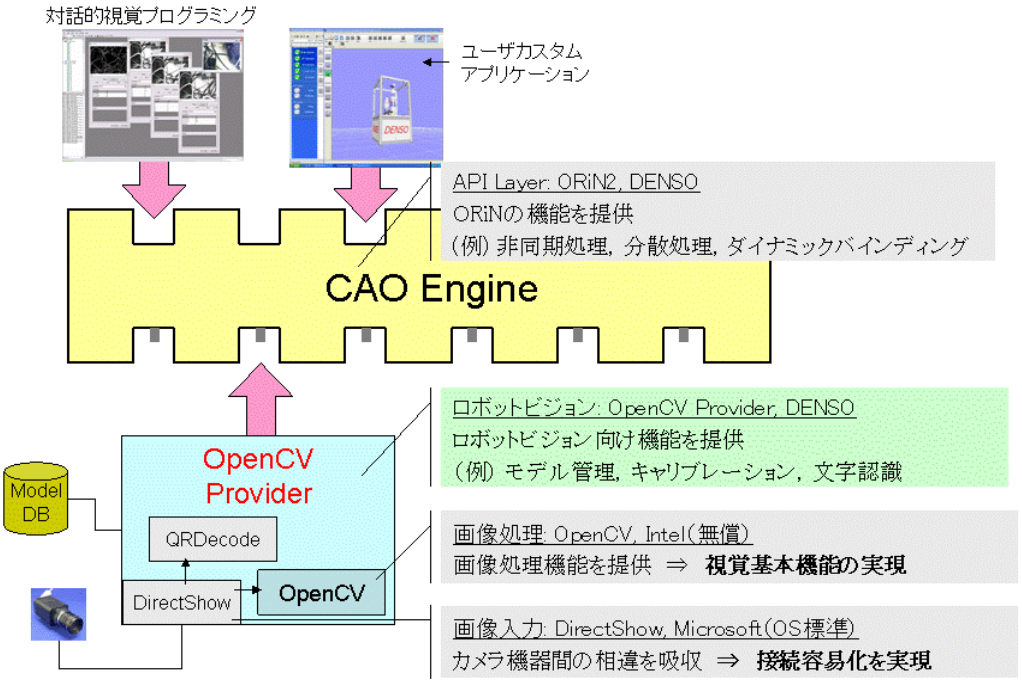


図 1-1 RIL のアーキテクチャ

1.1. ライセンスの追加

本プロバイダを使用可能にするには ORiN2 SDK をインストール後, 別途「DENSO Robot Imaging Library」ライセンスを入力する必要があります. 評価用にインストールする場合は下記のライセンスキーをご使用ください.

CVG3-MZPB-7W2G-L43Q (評価用 3 ヶ月)

下記に「DENSO Robot Imaging Library」ライセンスの追加手順を示します.

[1] OpenCV は Intel 社が開発したオープンソースの画像処理ライブラリで, 5.10.付録 C の License Agreement の下で公開されています.

1. CaoConfig を起動し，[Cao Provider]タブを選択する
2. Provider List から[OpenCV CAO Provider]項目を選択する
3. License 項目の[…]ボタンをクリックする
4. ORiN2 License Manager で[Add]ボタンをクリックする
5. 入手したライセンスキーを入力後，[OK]ボタンをクリックする
6. [Close] ボタンをクリックし，CaoConfig を終了する

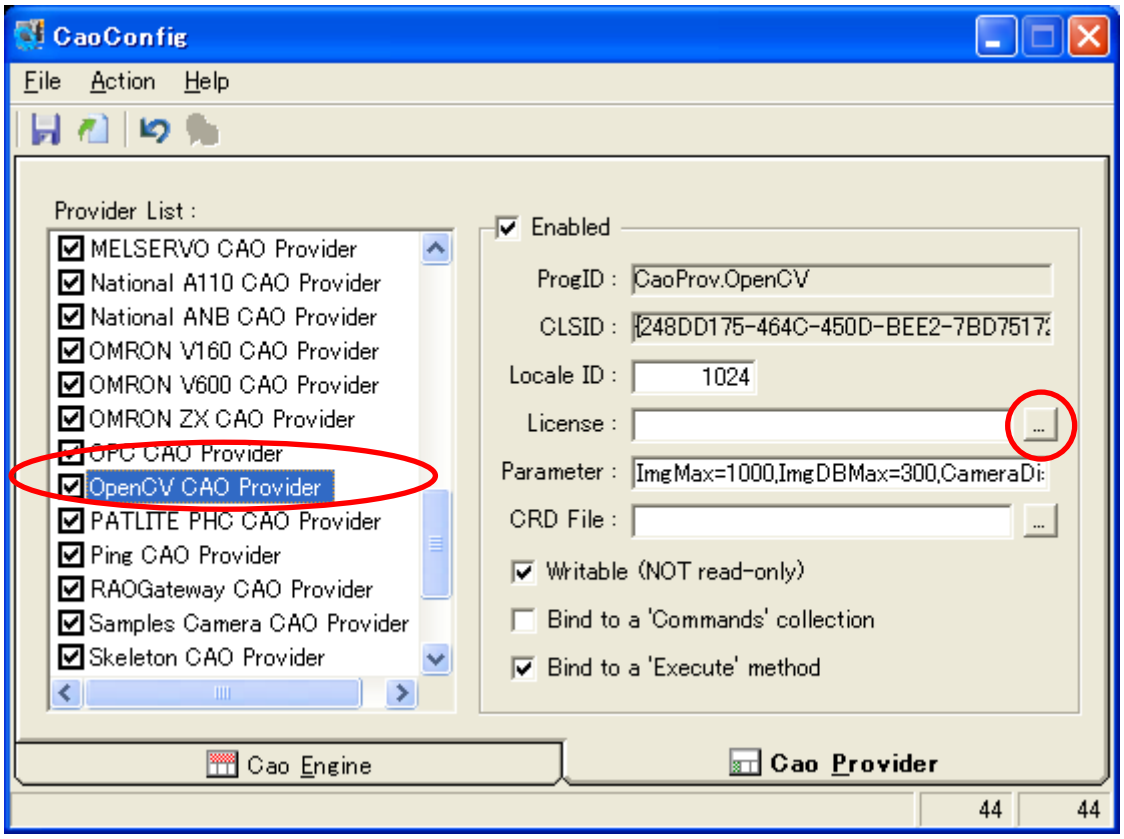


図 1-2「DENSO Robot Imaging Library」ライセンス追加

2. プロバイダの概要

2.1. 概要

OpenCV プロバイダは，DirectShow を使用してカメラデバイスからの画像を取得し，OpenCV を使用して画像処理を行います．したがって，DirectShow に対応している市販の様々なカメラを使うことができます．処理結果の画像は，イメージメモリ領域やデータベース領域に格納されます．これらの領域に関しての詳細については後述します．

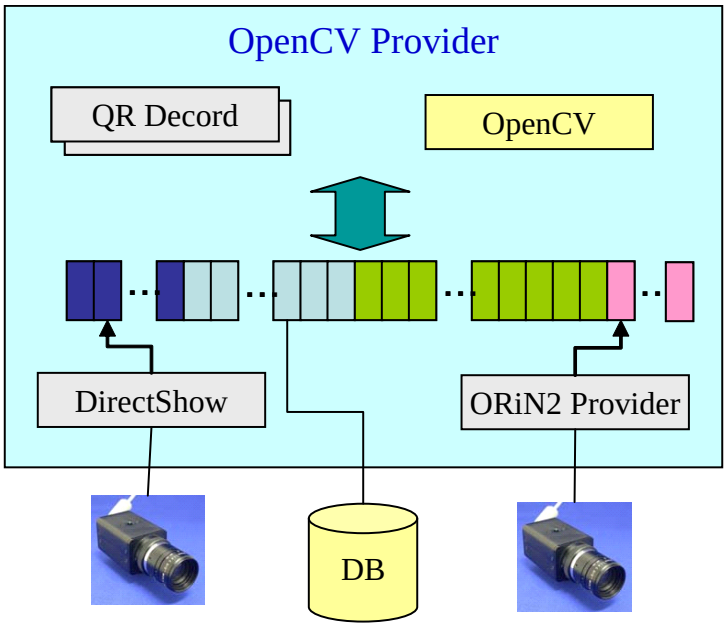


図 2-1 システムコンフィギュレーション

特定の機種のカメラは拡張カメラとして使用することができます．拡張カメラでは，各カメラ用の ORiN2 プロバイダを使用してカメラとの通信を行うことにより，詳細な設定を行うことが可能となります．

拡張カメラで使用する ORiN2 プロバイダでは，各社のドライバが必要な場合があります．詳細は各プロバイダのマニュアルを参照してください．また，ドライバのインストール時に DirectShow のドライバはインストールしないでください．[2]

拡張カメラとして識別できるカメラは以下の 2 種類になっています．

表 2-1 拡張カメラ対応機種

拡張カメラ1	Basler GigE カメラ ORiN2¥CAO¥ProviderLib¥Basler¥Pylon¥GigE
拡張カメラ2	IDS uEye カメラ

[2] DirectShow ドライバをインストールした場合，拡張カメラとして識別されません．

	ORiN2¥CAO¥ProviderLib¥IDS¥uEye
拡張カメラ3	Canon ネットワークカメラ WebView Livescope ORiN2¥CAO¥ProviderLib¥Canon¥WebView

OpenCV プロバイダのファイル形式は DLL(Dynamic Link Library)であり，CAO エンジンから使用時に動的にロードされます．OpenCV プロバイダを使用するにあたっては ORiN2SDK をインストールするか，下表を参照して手作業でレジストリ登録を行う必要があります．

表 2-2 OpenCV プロバイダ

ファイル名	CaoProvOpenCV.dll
ProgID	CaoProv.OpenCV
レジストリ登録[3]	regsvr32 CaoProvOpenCV.dll
レジストリ登録の抹消	regsvr32 /u CaoProvOpenCV.dll

OpenCV プロバイダでは，イメージメモリ及びカメラの設定をレジストリに登録します．登録内容の変更は CaoConfig を使用して行います．以下に設定内容を示します．

表 2-3 CaoConfig の Parameter 文字列

パラメータ	意味
DataBase=<データベースファイルのパス>	データベースファイルの絶対パスを指定します． (デフォルト:デフォルトデータベース) データベースファイルは，以下のファイルをコピーして使用してください． <OpenCV>¥Bin¥opencv_master_en.mdb
DefaultCamera=<既定カメラ ID>	既定のカメラ ID を指定します． (デフォルト:1) AddFile で ID オプションを省略したときは，既定カメラ ID でファイルオブジェクトを作成します．
ImgMax=<イメージメモリのサイズ>	イメージメモリ全体のサイズを指定します． (デフォルト:200)
ImgDBMax=<データベース領域のサイズ>	イメージメモリのデータベース領域のサイズを指定します．(デフォルト:100)

[3] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません．

	指定した値がイメージメモリのサイズよりも大きいときは，イメージメモリ内のすべてをデータベース領域とします．
CameraDisable [=<未使用カメラ指定>]	使用しないカメラをビット指定します．ビットは 0 ビットが ID=0 に対応して 9 ビット目まで指定します． （デフォルト：0x00） ビット ON：使用しない． ビット OFF：使用する．
FormatType [=<カメラ 0 設定>: [<カメラ 1 設定>: ・・・ [<カメラ 9 設定]]]]]]]	各カメラの画面設定を指定します． （デフォルト：0:0:0:0:0:0:0:0:0:0） -1：起動時の設定 指定可能な値の範囲はカメラによって異なります．（0～） 不正な値を指定した場合，デフォルトの設定でカメラを起動します．
FrameRate [=<カメラ 0 設定>: [<カメラ 1 設定>: ・・・ [<カメラ 9 設定]]]]]]]	各カメラのフレームレートを指定します． （デフォルト：0:0:0:0:0:0:0:0:0:0） -1：起動時の設定 指定可能な値の範囲はカメラによって異なります． 不正な値を指定した場合，デフォルトの設定でカメラを起動します．
ExtCamera [=<Basler GigE カメラ領域>: [<iDS uEye カメラ領域>]]	追加する拡張カメラの領域サイズを指定します．（デフォルト：0:0）

2.1.1. イメージメモリ

OpenCV プロバイダでは，画像を 2 つのイメージメモリ（データベース領域とメモリ領域）に格納することができます．データベース領域は，OpenCV プロバイダが終了してもデータの内容を保存されますが，アクセス速度が低速ですので，保存する必要がないときにはメモリ領域を使ってください．メモリ領域は，アクセス速度が高速ですが OpenCV プロバイダ終了時にデータが破棄されます．

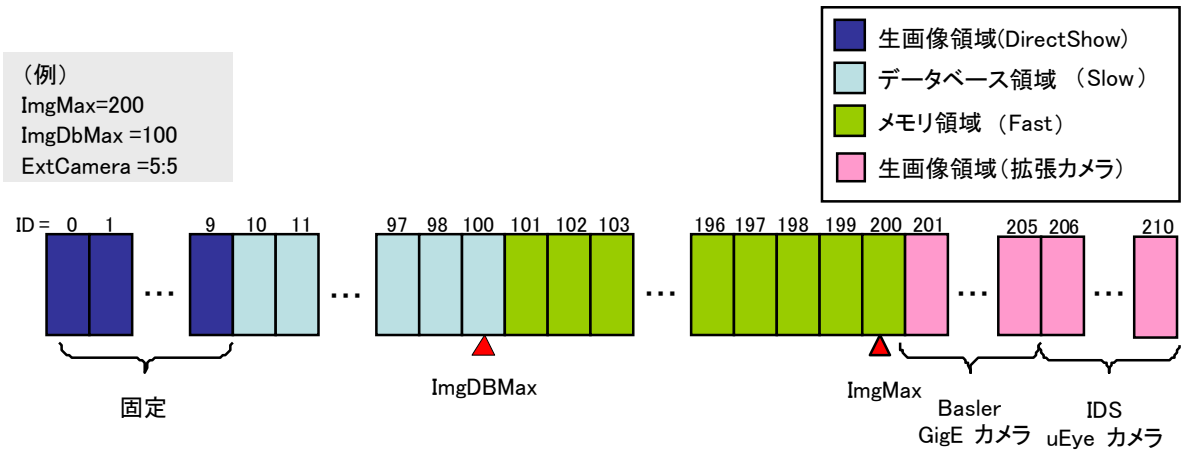


図 2-2 イメージ格納領域

2.1.2. キャリブレーション

OpenCV プロバイダのキャリブレーションコマンドは，カメラ座標-ワールド座標間をキャリブレーションをするカメラキャリブレーションとワールド座標-ロボット座標間をキャリブレーションするロボットキャリブレーションの 2 種類に大別されます．以下にそれぞれに分類されるコマンドを示します．

・ カメラキャリブレーション用コマンド：

CalibrateCamera, GetCamCalDat, SetCamCalDat, GetCamCalExtDat, SetCamCalExtDat, GetPosFromCam, GetCamPos, Undistort2

・ ロボットキャリブレーション用コマンド：

CalibrateRobot, GetRobCalDat, SetRobCalDat, GetPosFromRob, GetRobPos

ワールド座標，カメラ座標，ロボット座標の関係を示した図を以下に示します．このときワールド座標系は“Origin”を原点とする座標系， ロボット座標系は“Base”を原点とする座標系，カメラ座標系はカメラが撮影した画面上の座標となります．

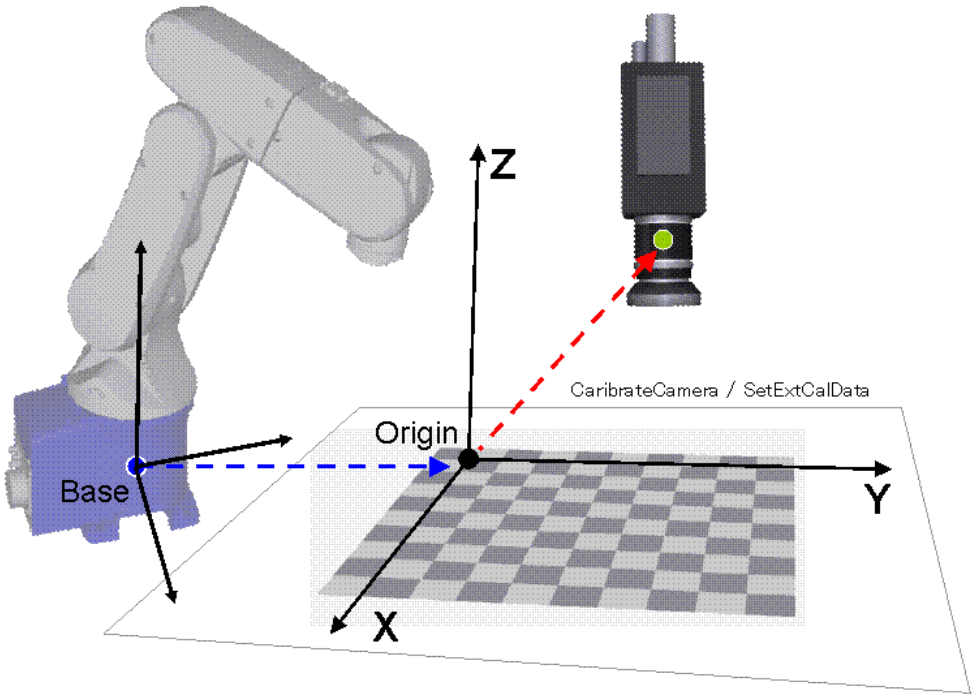


図 2-3 カメラ-ロボットキャリブレーション

CaoCommand クラスの三角測量コマンドを使う場合は，カメラキャリブレーションを実施する必要があります．下記にキャリブレーションの手順を紹介します．

2.1.2.1. カメラキャリブレーション

カメラキャリブレーションに必要となるカメラの内部パラメータや外部パラメータ，およびレンズの歪を補正するためのパラメータを算出するために，OpenCV プロバイダではチェスボードを使った手順を提供しています．下記にその手順を示します．

[Step 1] チェスボード画像の準備

1 つのカメラに対して少なくとも 5 つのチェスボード画像が必要です．それらの画像を適当な画像領域(例：101-105)に保存します．その中の 1 枚目の画像が外部パラメータを算出するために用いられる基準画像になります．基準画像の ID は'CalibrateCamera'コマンドの'Input ID'に設定します．基準画像を撮像したときのチェスボート面を基準面と呼びます（図 2-4）．このチェスボードのファイルは ORiN2 SDK の中の下記のフォルダに格納されていますので活用ください．

<ORiN2>/CAO/ProviderLib/OpenCV/Doc/Chessboard.pdf

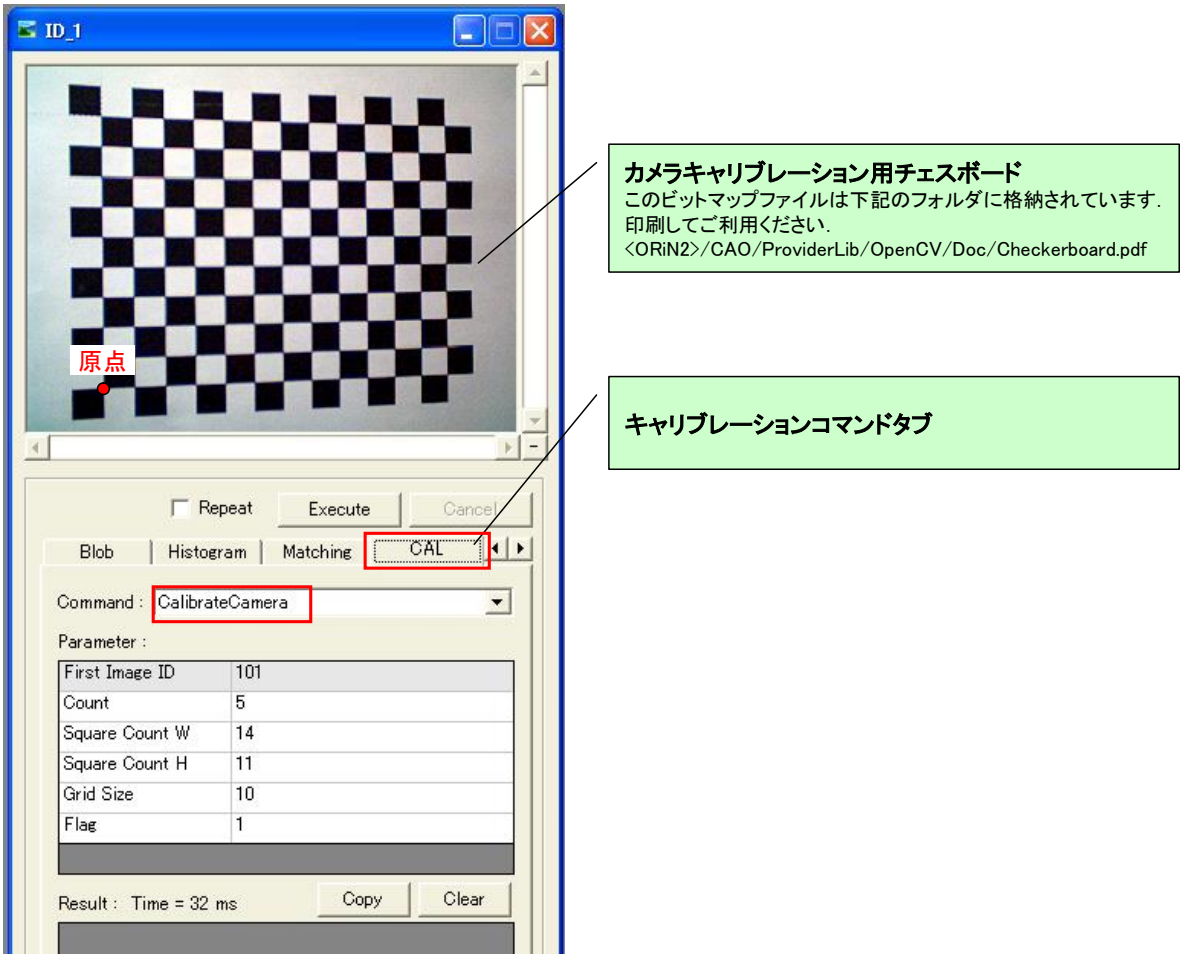


図 2-4 キャリブレーションダイアログ

5 つの画像の撮影方法には，カメラを移動する場合とチェスボードを移動する場合の 2 つが考えられます(図 2-5). どちらの場合でも基準面の左下の 1 つ目の"**黒ブロック**"の右上の座標がワールド座標の原点になります. [4]

[4] 左下に黒ブロックがないチェスボード画像の場合は，原点位置が右上になることがあります. CalibrateCamera コマンドを実行後に，GetCamPos コマンドで原点の画面上での位置を確認してください.

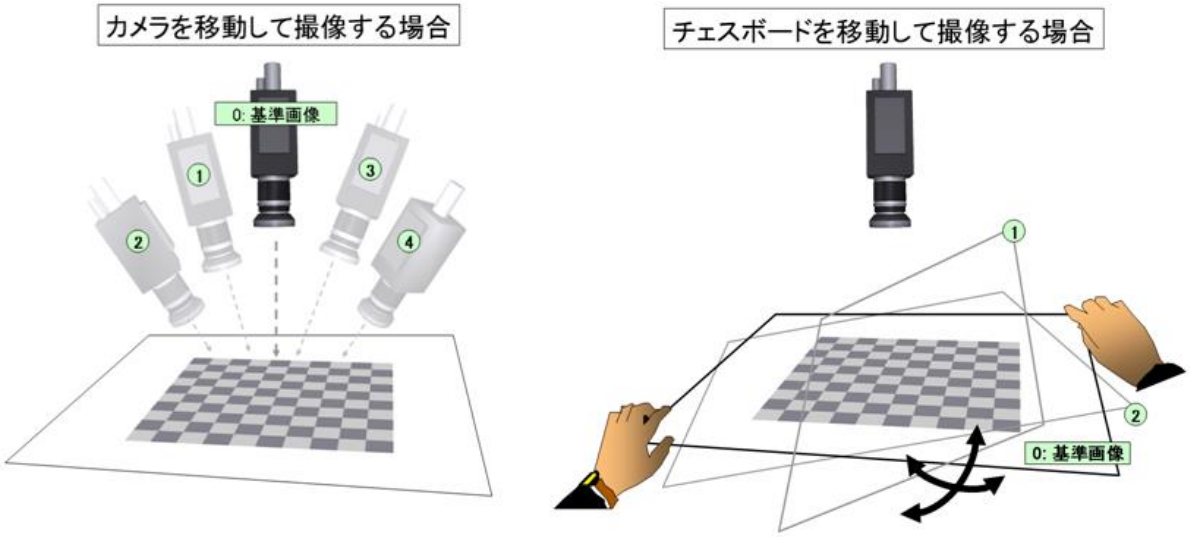


図 2-5 チェスボード画像の撮り方

[Step 2] カメラパラメータの算出

'CalibrateCamera'コマンドを用いてカメラパラメータを算出します. SDK に格納されているチェスボードファイルを使って撮像した場合, 'Square count W', 'Square count H'及び 'Grid size (mm)'の値は各画像に記述されている値を使用してください. 独自にチェスボートを作成した場合や縮小・拡大印刷した場合などはそれに合わせてください. それらの値を設定後, [Execute]ボタンで実行するとカメラパラメータが算出されます. 算出された値はデータベースに自動的に保存されます. 保存されている値を取得するには, 'GetCamCalDat'コマンドを使います.

[Step 3] 算出結果の検証

カメラのキャリブレーションが完了した後であればワールド座標の X-Y-Z の値を正確に取得することができます. 'GetPosFromCam'コマンドで適当なカメラ座標(Xc,Yc)を指定して実行してください. それに対応するワールド座標の(Xw,Yw,Zw)が取得できます. これで基準画上の 2 点であれば正確に距離の算出も可能になります.

また, Undistort2 コマンドを使うとレンズの歪を補正した画像を出力することができます. 周辺の画像の直線性が改善されていることを確かめてください.

[注意]

GetPosFromCam 変換後の点はワールド座標の Z=0 の面上の点になります. したがって, カメラに写っている点が上記の面上にない場合は, 誤差が発生します(例えば, 図 2-6 において X1obj-X1cal が誤差). 対象物が立体である場合は, キャリブレーション時の基準面をオブジェクトの高さに合わせる必要があります. 正確な三次元座標を測定する場合には, 立体視(三角測量)のコマンドを使用してください.

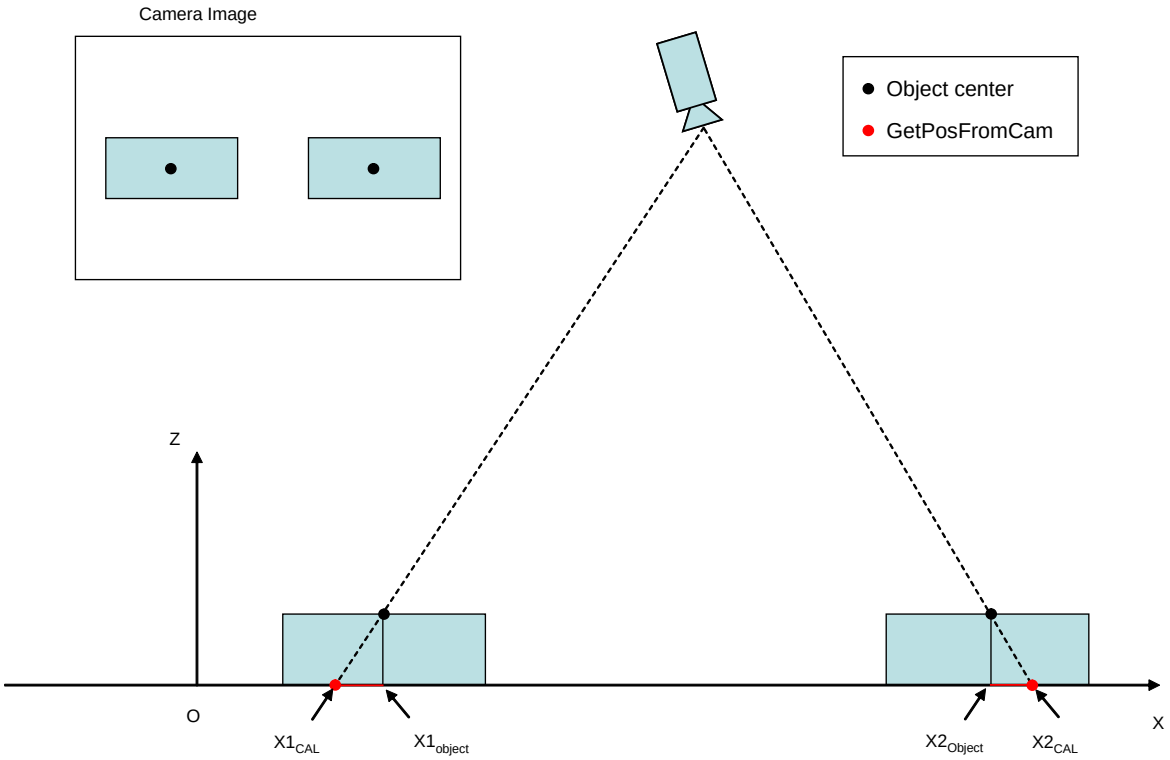


図 2-6 カメラキャリブレーションによる写像面との誤差

2.1.2.1.1. 歪み補正

カメラレンズには歪みがあるため，撮影した画像は歪んだ状態で表示されます．この歪みを補正するためのパラメータはカメラキャリブレーションを行うことで計算されます．

‘Undistort2’コマンドは，歪みがある画像から歪みを補正した画像への変換を行います．

‘GetPosFromCam’コマンドは，カメラ座標をワールド座標に変換を行います．このコマンドを使用する際に引数として歪み補正フラグを TRUE にすることにより，歪みがある画像から直接ワールド座標に変換することができます．

ここで，歪みのある画像から，歪みのない画像への変換を行うことは可能ですが，逆方向の変換をすることはできません．

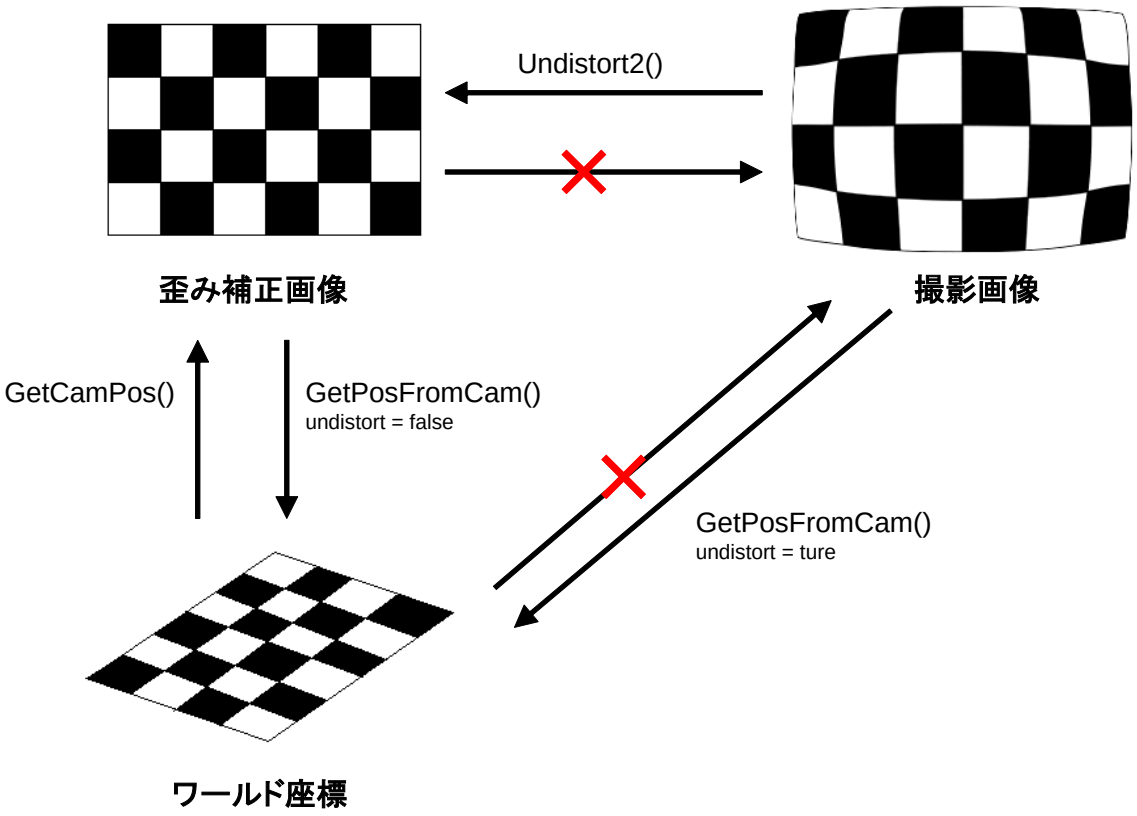


図 7 コマンドによる歪み補正

2.1.2.2. ロボットキャリブレーション

ワールド座標系とロボットの座標系を関連付ける方法として以下の 2 つの方法があります.

1. OpenCV プロバイダのワールド座標系-ロボット座標系間のロボットパラメータを算出する.
2. ロボットでワールド座標系をロボットの作業座標系(ワーク座標系)に設定する.

ここでは, 前者の OpenCV プロバイダによるワールド座標系-ロボット座標系間のキャリブレーション手順を示します. 後者の方法については各ロボットのマニュアルを参照してください. ワールド座標系とロボット座標系を合わせる必要がない場合はこれらの設定をする必要はありません.

[Step 1] ロボットパラメータの算出

'CalibrateRobot'コマンドを用いてロボットパラメータを算出します. このコマンド実行時には以下の 3 点の座標が必要になります.

- ワールド座標の原点
- ワールド座標の X 軸上の任意の点
- ワールド座標の XY 平面上の任意の点

これらの点はロボット基準座標系で見た座標で入力する必要があります.

それらの値を設定後, [Execute]ボタンで実行するとロボットパラメータが算出されます. 算出された値

はデータベースに自動的に保存されます. 保存されている値を取得するには, 'GetRobCalDat'コマンドを使います.

[Step 2] 算出結果の検証

ロボットののキャリブレーションが完了した後であればワールド座標系の X-Y-Z の値を正確に取得することができます. 'GetPosFromRob'コマンドで適当なロボット座標(Xr,Yr,Zr)を指定して実行してください. それに対応するワールド座標の(Xw,Yw,Zw)が取得できます.

2.1.3. 三角測量機能

OpenCV プロバイダは, 2 台のカメラを使用して三角測量を行う機能を提供しています. この機能を使うためには, 前出のカメラキャリブレーションとカメラ-ロボットキャリブレーションを完了している必要があります. 三角測量関連のコマンドには下記の 4 つがあります.

Triangulation, TriHaarDetect, TriMatchShapes, TriMatchTemplate

'Triangulation'コマンドは 2 台のカメラで対応する 2 点の座標(X1c,Y1c),(X2c,Y2c)を指定して三次元座標を算出します(図 2-8). その他のコマンドはその対応点をHaarDetect, MatchShapes2, MactchTemplate2を使って自動検出後, 'Triangulation'コマンドと同じ処理をします. この 3 つのコマンドの戻り値(X,Y)は意味が異なりますので注意してください. HaarDetect の場合のみ下記のように補正します.

- HaarDetect : 1 つ目の結果の(X,Y)にその W/H の 1/2 を足した値, (X + W/2, Y + H/2).
- MatchShapes2 : 結果をそのまま使用.
- MactchTemplate2 : 結果をそのまま使用.

同様に, 対応点を検出するユーザ独自の処理と'Triangulation'コマンドを組み合わせれば独自の三角測量コマンドを作ることもできます. 実際のアプリケーションの場合, 対応点の検出はそのアプリケーションに合わせて最適化すれば高速な演算が可能になりますので, 思ったように精度がでない場合や, さらに高速化したい場合は独自に実装することをお奨めします.

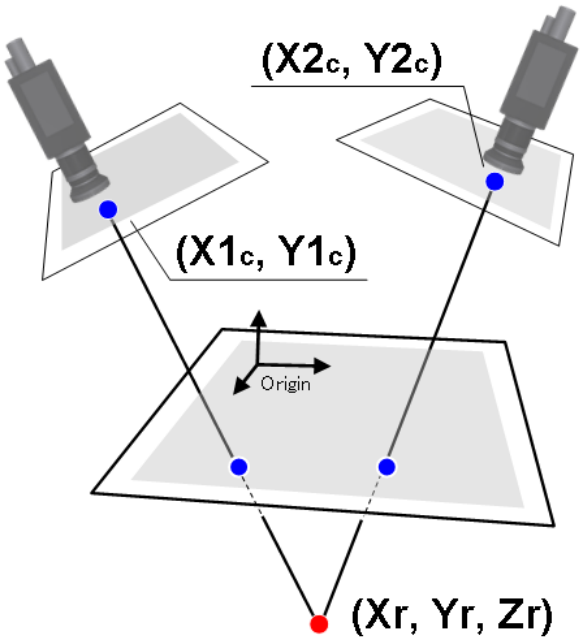


図 2-8 角測量

三角測量のコマンドで得られた座標は原点を基準にした座標です．カメラ自身の位置も同じ原点基準であるため Distance コマンドを使えば簡単にカメラと対象物の距離を算出することができます（図 2-9）．

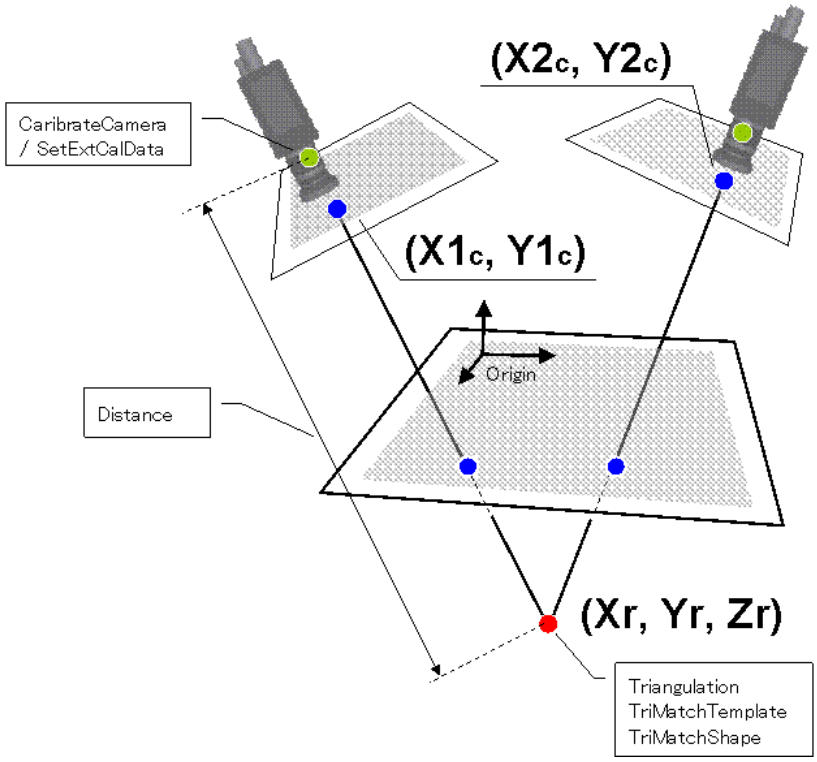


図 2-9 距離測定

2.1.4. メッセージ転送機能

CAO エンジンのメッセージ転送機能により，受信したメッセージ内に格納されている画像データを指定したイメージ ID に格納することができます．

格納先の指定は，AddController()の MsgDestID オプションで行います．複数の送信元からの画像をそれぞれ別のイメージ ID に格納する場合には，CaoController オブジェクトを複数作成し，それぞれの AddController() で格納先のイメージ ID を指定する必要があります．

転送メッセージのデータがビットマップファイルでないときは，データの格納は行いません．

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

OpenCV プロバイダでは AddController 時に，カメラの検索を行い，接続処理を行います．
接続オプションには表 2-3 のパラメータも指定できます．

書式 AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPcName:BSTR > [,<bstrOption:BSTR>])

bstrCtrlName : [in] コントローラ名
bstrProvName : [in] プロバイダ名．固定値 =" CaoProv.OpenCV".
bstrPcName : [in] プロバイダの実行マシン名
bstrOption : [in] オプション文字列

表 2-4 CaoWorkspace::AddController のオプション文字列

オプション	意味
QREnabled=True/False	"QRDecode"コマンドを有効にします．デフォルト=False
OCREnabled =True/False	"OCRead"コマンドを有効にします．デフォルト=False
MsgDestID=<イメージ ID>	メッセージ転送時の格納先イメージ ID を指定します．
FormatType=t1:t2:t3:t4:t5:t6:t7:t8:t9:t10	起動時のカメラのフォーマットを指定します． 指定しない場合又は設定できない値の時は，デフォルトのカメラ設定で起動します． 例:2 個目のカメラを 2 番のカメラ設定で起動する FormatType=0:2:0:0:0:0:0:0:0:0
FrameRate=f1:f2:f3:f4:f5:f6:f7:f8:f9:f10	起動時のカメラのフレームレートを指定します． 設定しない場合又は設定できない値の時は，デフォルトのフレームレートで起動します． 例:2 個目のカメラのフレームレートを 20 で起動する FrameRate=0:20:0:0:0:0:0:0:0:0

AddController が失敗する場合，原因として以下の 2 つが考えられます．

・ カメラのデバイス異常
カメラが正しく動作していない可能性があります．DirectX 付属のサンプル"amcap.exe"でカメラが正しく動作することを確認して下さい．

・ データベースファイル異常
データベースファイルが破損している可能性があります．"CAO/ProviderLib/OpenCV/Bin"ディレクトリにあるデータベースファイル"opencv.mdb"を削除してください．次の起動時に新規にデータベ

ースファイルが自動作成されます. ただし, データベース領域に保存してあったデータは破棄されます.

2.2.2. CaoController::AddCommand メソッド

三角測量を行う CaoCommand を生成します.

書式 AddCommand(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] コマンド名

bstrOption : [in] オプション文字列(未使用)

使用することができるコマンド名は 4.3.1 を参照してください.

2.2.3. CaoController::AddFile メソッド

カメラデバイス及びイメージメモリにアクセスするファイルオブジェクトを作成します.

書式 AddFile(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] 任意の名前

bstrOption : [in] オプション文字列

表 2-5 CaoWorkspace::AddFile のオプション文字列

オプション	意味
ID[=<イメージ ID>]	初期に接続されるイメージメモリの番号 (デフォルト:既定カメラ ID) このオプションを省略したときは, 表 2-3 の”DefaultCamera“で指定した既定カメラ ID のイメージに接続します.

2.2.4. CaoController::AddVariable メソッド

カメラデバイス及びイメージメモリ情報の変数オブジェクトを作成します. 変数名には, 2.3.1 の変数のみ使用することができます.

書式 AddVariable(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] 任意の名前

bstrOption : [in] オプション文字列

2.2.5. CaoController::Execute メソッド

コントローラクラスのコマンドを実行します.

各コマンドの詳細は 4.1 を参照してください.

2.2.6. CaoController::get_VariableNames プロパティ

変数リストを取得します. 取得する変数については, 2.3.1 を参照してください.

2.2.7. CaoCommand::Execute メソッド

コマンドクラスのコマンドを実行します.

各コマンドの詳細は 4.3 を参照してください.

2.2.8. CaoCommand::put_Parameter プロパティ

コマンドのパラメータを設定します.

各コマンドのパラメータは 4.3 を参照して下さい. このプロパティでは, パラメータが不正な場合のチェックは行いません.

2.2.9. CaoCommand::get_Parameter プロパティ

2.2.8 で設定したパラメータを取得します. パラメータが設定されていないときは VT_EMPTY を返します.

2.2.10. CaoCommand::get_Result プロパティ

最後に実行した 2.2.7 の実行結果を取得します. 各コマンドの結果は 4.3.1 を参照して下さい.

2.2.11. CaoFile::Execute メソッド

コマンド名で指定した画像処理又は, 演算処理を行います.

Execute メソッドの引数は, コマンドを BSTR, パラメータを VARIANT 配列で指定します.

書式 [<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

bstrCmd : [in] コマンド

vntParam : [in] パラメータ

vntRet : [out] 返り値

コマンド仕様の詳細に関しては, 4.2 を参照して下さい.

2.2.12. CaoFile::get_Attribute プロパティ

参照しているイメージメモリの種別を取得します.

0x0002	DirectShow カメラ
0x0003	データベース領域
0x0004	メモリ領域
0x0100	Basler GigE カメラ領域
0x0101	IDS uEye カメラ領域
0x0102	Canon WebView カメラ領域

2.2.13. CaoFile::put_ID プロパティ

参照するイメージメモリを切り替えます.

2.2.14. CaoFile::get_ID プロパティ

現在参照しているイメージメモリの ID を返します.

2.2.15. CaoFile::get_DateLastModified プロパティ

現在参照しているイメージメモリの更新時間を取得します.

イメージメモリに画像がない場合は VT_EMPTY を取得します.

2.2.16. CaoFile::Get_Size プロパティ

現在参照しているイメージメモリのファイルサイズを取得します.

2.2.17. CaoFile::put_Value プロパティ

現在参照しているイメージメモリに BMP 形式の画像を上書きします.

設定した画像は, カラー画像として書き込みます.

2.2.18. CaoFile::get_Value プロパティ

現在参照しているイメージメモリの画像を BMP 形式で取得します.

カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで取得します.

2.2.19. CaoFile::get_Help プロパティ

PutHelp コマンドで設定された文字列を取得します.

カメラ領域の場合は, カメラ名を取得します.

2.2.20. CaoController::OnMessage イベント

画像データが更新された時, CaoController クラスの OnMessage イベントが発生します. このとき Message::Number プロパティは1, Message::Value プロパティにはイメージ ID が格納されます.

2.3. 変数一覧

2.3.1. コントローラクラス

表 2-6 コントローラクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@IMG_MAX	VT_I4	イメージメモリ全体のサイズ	○	-
@IMGDB_MAX	VT_I4	イメージメモリのデータベース領域のサイズ	○	-
@CAM_COUNT	VT_I4	接続されているカメラの数	○	-
@VERSION [V1.3.5 以降]	VT_BSTR	プロバイダのバージョン	○	-
@EVENT_ENABLED [V1.3.5 以降]	VT_BOOL	CAO メッセージイベントの発生を切り替えます.	○	○
@EXT_CAM_COUNT [V1.4.6 以降]	VT_I4 | VT_ARRAY	各拡張カメラ領域のサイズ 以下の順番で配列に格納されています. <Basler GigE カメラ領域> <IDS uEye カメラ領域> <Canon uEye カメラ領域>	○	-
@EXT_CAM_COUNTS [V1.5.5 以降]	VT_I4	拡張カメラの領域のサイズの合計	○	-
@CAM_CAL_MAX [V1.5.5 以降]	VT_I4	カメラキャリブレーションの領域サイズ	○	-
@ROB_CAL_MAX [V1.5.5 以降]	VT_I4	ロボットキャリブレーションの領域サイズ	○	-

2.3.2. ファイルクラス

表 2-7 ファイルクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@VALUE [V1.3.5 以降]	VT_UI1 | VT_ARRAY	イメージメモリの画像 CaoFile::get_Value(),CaoFile::put_Value() と同じ動作をします.	○	○

2.4. エラーコード

OpenCV プロバイダでは，以下の固有エラーコードが定義されています．ORiN2 共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください．

表 2-8 独自エラーコード一覧

エラー名	エラー番号	説明
E_CAOP_NO_LICENSE	0x80100000	ライセンスがありません． 追加ライセンスを購入してください．
E_CAOP_DB_RESTORE	0x80100001	データベースファイルが破損していました． 前回起動時のバックアップからリカバリしたので，プログラムを再起動してください．
E_CAOP_INITTERM	0x80100002	別プロセスが初期化または終了処理を実行しています． しばらく待ってから実行してください．
E_CAOP_NOIMAGE	0x80100003	画像がありません．
E_CAOP_LOCK_IMAGE	0x80100004	画像が別のプログラムによってロックされています．
コマンド独自エラー	0x801010xx	OpenCV プロバイダの Execute()の各コマンドの独自エラーを返します． エラーコードの内容については，4 を参照してください．
OpenCV API エラー	0x8011xxxx	OpenCV API 実行時にエラーが発生した場合は，OpenCV のエラーコードを xxxx の箇所に入れて返します． エラーコードの内容については OpenCV のリファレンスを参照してください．

3. サンプルプログラム

RIL のプログラムは ORiN と同様に様々な市販のプログラミング言語(C/C++, VB 等)を使って開発することができます. 最も簡単な方法は ORiN2 SDK に付属の CaoScript という VBScript 言語ベースのスクリプト言語を使うことです. 3.1 で CaoScript 言語によるサンプルを紹介します. その他のサンプルは 3.2 節を参照してください.

3.1. CaoScript サンプル

このサンプルは, 対象物(ID=101 に予め登録されているもの)の位置を検出し, デンソーロボットをその位置へ移動させるプログラムです.

```
' CAO オブジェクトの作成
Set rc = Cao.AddController("rc", "CaoProv.DENSO.NetwoRC", "", "Conn=eth:192.168.0.1")
Set robo = rc.AddRobot("vp")
Set vis = Cao.AddController("cv", "CaoProv.OpenCV", "", "")
Set rawImg = vis.AddFile("cam1", "ID=1")
Set tmpImg = vis.AddFile("mem1", "ID=101")
' パターンマッチングでターゲットをサーチ＆追跡
OldX = -1: OldY = -1
Do
   ' 判別分析法による閾値の算出
   iT = rawImg.AutoThreshDiscrim(rawImg.CalcHistEx(255))
   ' 2 値化 & 白黒反転(1)
   rawImg.ThresholdEx 101, iT, 255, 1
   ' 形状マッチング
   res = tmpImg.MatchShapes2(11, 2, 0.2)
   ' 前回の位置との差分だけロボットを並進移動
   If (OldX <> -1) Then
      v = "V(" & (OldX - res(0)) & "," & (OldY- res(1)) & ",0)"
      robo.Draw 1, v, "next"
   End If
   OldX = res(0): OldY = res(1)
Loop
```

3.2. その他のサンプル

その他の RIL 用のサンプルプログラムが以下の場所にありますので参考にしてください.

<ORiN2>¥CAO¥ProviderLib¥OpenCV¥Samples

以下にサンプルの一覧を示します.

表 3-1 サンプルプログラム一覧

サンプルフォルダ	内容	言語
3DTracking	パターンマッチングによるロボット動作プログラム.	Visual Basic 6
Benchmark	基本的なコマンドのベンチマークプログラム. CPU やカメラの性能を確認するのに便利です.	Excel VBA
CutImage	カメラ 0 の画像を座標(0,0)から幅 100, 高さ 100 で切り取った画像を表示します. 切り取った画像はメ	Visual Basic 2005

	モリ 11 にも保存します.	
DENSO NetwoRC	指定したターゲット画像をカメラ画像内から探索し, その結果の座標を ID アドレス“10.6.235.60”にあるロボットコントローラの変数に格納します.	Visual Basic 6
Filter	カメラからの画像を以下の 4 種類で表示します. ・ 生画像 ・ グレースケール画像 ・ 2 値化 ・ Canny フィルタ	Visual Basic 2005
FindContoursEx	FindContoursEx を実行するサンプルです.	C 言語
Histogram	カメラ1の画像のヒストグラムを生成します.	Visual Basic 2005
Others	マッチングによるロボット動作等. CaoScript による最も簡単なサンプルです.	CaoScript
SaveImage	カメラ 1 の画像をイメージ ID=11 のメモリに保存します.	Visual Basic 2005

4. コマンドリファレンス

本章では各コマンドについて解説します. OpenCV ライブラリに依存するコマンドの詳細な動作は下記の OpenCV のマニュアル等を参考にしてください. また, そのようなコマンドが内部的にどの OpenCV 関数を使っているかは 5.10.付録 A を参照してください.

[OpenCV 日本語マニュアル] http://opencv.jp/opencv-1.0.0/document/

[OpenCV 英語マニュアル] http://opencv.jp/opencv-1.0.0_org/docs/index.htm

表 4-1 CaoController::Execute コマンド一覧

カテゴリ	コマンド	機能	
ビデオ設定			
	~~SetFormat~~[5]	~~ビデオフォーマットの設定~~	
	~~GetFormatList~~[5]	~~ビデオフォーマットリストの取得~~	
	OpenFileterProperty	フィルタのプロパティウィンドウを表示	P. 36
	OpenPinProperty	出力ピンのプロパティウィンドウを表示	P. 36
	SetCtrlMode	ビデオコントロールモードを設定します.	P. 36
	GetCtrlMode	ビデオコントロールモードを取得します.	P. 37
	GetRangeCameraCtrl	カメラコントロールのパラメータ範囲取得	P. 37
	GetCameraCtrl	カメラコントロールのパラメータ取得	P. 38
	SetCameraCtrl	カメラコントロールのパラメータ設定	P. 39
	GetRangeVideoProcAmp	ビデオコントロールのパラメータ範囲取得	P. 39
	GetVideoProcAmp	ビデオコントロールのパラメータ取得	P. 40
	SetVideoProcAmp	ビデオコントロールのパラメータ設定	P. 41
	GetCameraFormatList	カメラフォーマットリストの取得	P. 42
	GetCameraFormat	カメラフォーマットの ID 取得	P. 42
	SetCameraFormat	カメラフォーマットの ID 設定	P. 43
	GetCameraFrameRate	カメラのフレームレート取得	P. 43
	SetCameraFrameRate	カメラのフレームレート設定	P. 43
	ExtExecSoftTrigger	ソフトウェアトリガの実行	P. 44
	ExtRefreshImage	拡張カメラの画像更新	P. 44
	ExtInvoke	拡張カメラのコマンド実行	P. 44
	ExtConnect	拡張カメラの接続	P. 45
	ExtDisconnect	拡張カメラの切断	P. 45
	ExtIsConnected	拡張カメラの接続確認	P. 46
	ExtGetConnectOption	拡張カメラの接続パラメータ取得	P. 46
	ExtSetConnectOption	拡張カメラの接続パラメータ設定	P. 46

[5] OpenPinProperty に統合されました.

表 4-2 CaoFile::Execute コマンド一覧

カテゴリ	コマンド	機能	
一般			
	SetROI	範囲指定値の設定	P. 48
	GetROI	範囲指定値の取得	P. 48
	ResetROI	範囲指定のリセット	P. 49
	PutColor	色設定	P. 49
	GetColor	色取得	P. 50
	SearchPoint	ポイント検索	P. 50
	Trim	トリミング	P. 51
	ImageSize	画像サイズの取得	P. 52
	IsColor	画像のカラー判別	P. 52
	IsEmpty	画像の存在確認	P. 53
	IsUpdated	画像の更新確認	P. 53
	ClearUpdated	画像の更新フラグリセット	P. 53
	Distance	距離計測	P. 54
	InnerProduct	内積	P. 54
	OuterProduct	外積	P. 55
	PutHelp	文字列の設定	P. 55
編集			
	Copy	コピー	P. 55
	Cut	切り取り	P. 56
	Paste	貼り付け	P. 56
	Rotate	回転	P. 57
	Flip	反転	P. 58
	Resize	リサイズ	P. 58
	Split	色空間の分割	P. 59
	Merge	色空間の結合	P. 60
フィルタ			
	ConvertGray	グレースケール変換	P. 60
	ThresholdEx	閾値処理	P. 61
	Threshold2	閾値処理(2 値化)	P. 62
	AdaptiveThresholdEx	適応的閾値処理	P. 62
	Smooth	平滑化	P. 63
	Sobel	Sobel フィルタ	P. 65
	Laplace	Laplace フィルタ	P. 66
	CannyEx	Canny フィルタ	P. 66
	WarpAffine	Affine 変換	P. 67
	WarpPerspective	透視変換	P. 68
	PreCornerDetectEx	コーナー検出	P. 69
	CornerHarrisEx	ハリスエッジ検出	P. 70

[6] v.1.3.5 から廃止されました.

表 4-3 CaoCommand::Execute コマンド一覧

4.1. コントローラクラス
4.1.1. ビデオ設定

OpenFilterProperty

構文 *object*.OpenFilterProperty <Camera ID>, <Window Handle>

引数 <Camera ID> = VT_I4: カメラ番号[7]
<Window Handle> = VT_I4: 親またはオーナーウィンドウのハンドル

戻り値 なし

説明 カメラのフィルタプロパティウィンドウを開きます.

OpenPinProperty

構文 *object*.OpenPinProperty(<Camera ID>, <Window Handle>)

引数 <Camera ID> = VT_I4: カメラ番号[8]
<Window Handle> = VT_I4: 親またはオーナーウィンドウのハンドル

戻り値 <Format ID> = VT_I4: Camera format ID

説明 出力ピンプロパティウィンドウを開きます.

SetCtrlMode

構文 *object*.SetCtrlMode(<Camera ID>, <Mode>)

引数 <Camera ID> = VT_I4:カメラ番号
<Mode> = VT_I4:モード

1	VideoControlFlag_FlipHorizontal	水平フリップ
2	VideoControlFlag_FlipVertical	垂直フリップ
4	VideoControlFlag_ExternalTriggerEnable	外部トリガ
8	VideoControlFlag_Trigger	外部トリガシミュレート

戻り値 なし

[7] The number 1 to 10 are called 'Camera ID' for convenience.
[8] The number 1 to 10 are called 'Camera ID' for convenience.

説明 指定カメラのビデオコントロールモードを設定します.
詳細については MSDN の IAMVideoControl::SetMode()を参照してください.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 GetCtrlMode

GetCtrlMode

構文 *object*.GetCtrlMode(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 <Mode> = VT_I4:モード

1	VideoControlFlag_FlipHorizontal	水平フリップ
2	VideoControlFlag_FlipVertical	垂直フリップ
4	VideoControlFlag_ExternalTriggerEnable	外部トリガ
8	VideoControlFlag_Trigger	外部トリガシミュレート

説明 指定カメラのビデオコントロールモードを取得します.
詳細については MSDN の IAMVideoControl::GetMode()を参照してください.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 SetCtrlMode

GetRangeCameraCtrl

構文 *object*.GetRangeCameraCtrl(<CameraID>, <Property>)

引数 <CameraID> = VT_I4:カメラ番号
<Property> = VT_I4:プロパティ

0	CameraControl_Pan	パン(度単位)
1	CameraControl_Tilt	傾き(度単位)
2	CameraControl_Roll	ロール(度単位)
3	CameraControl_Zoom	ズーム(ミリ単位)
4	CameraControl_Exposure	露出(2n 秒)
5	CameraControl_Iris	絞り(fstop * 10 単位)
6	CameraControl_Focus	焦点距離(ミリ単位)

戻り値

<Min> = VT_I4:最小値

<Max> = VT_I4:最大値

<Step> = VT_I4:刻み

<Default> = VT_I4:デフォルト値

<Flag> = VT_I4:フラグ

0x0001	CameraControl_Flags_Auto	自動制御
0x0002	CameraControl_Flags_Manual	手動制御

説明

指定したカメラコントロールのパラメータ範囲を取得します.

詳細については MSDN の IAMCameraControl::GetRange()を参照してください.

[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目

GetCameraCtrl, SetCameraCtrl

GetCameraCtrl

構文

object.GetCameraCtrl(<CameraID>, <Property>)

引数

<CameraID> = VT_I4:カメラ番号

<Property> = VT_I4:プロパティ

0	CameraControl_Pan	パン(度単位)
1	CameraControl_Tilt	傾き(度単位)
2	CameraControl_Roll	ロール(度単位)
3	CameraControl_Zoom	ズーム(ミリ単位)
4	CameraControl_Exposure	露出(2n 秒)
5	CameraControl_Iris	絞り(fstop * 10 単位)
6	CameraControl_Focus	焦点距離(ミリ単位)

戻り値

<Value> = VT_I4:設定値

<Flag> = VT_I4:フラグ

0x0001	CameraControl_Flags_Auto	自動制御
0x0002	CameraControl_Flags_Manual	手動制御
0x0000	CameraControl_Flags_Absolute	絶対制御
0x0010	CameraControl_Flags_Relative	相対制御

説明

指定したカメラコントロールのパラメータを取得します.

詳細については MSDN の IAMCameraControl::Get()を参照してください.

[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 GetRangeCameraCtrl, SetCameraCtrl

SetCameraCtrl

構文 *object*.SetCameraCtrl(<CameraID>, <Property>, <Value>, <Flag>)

引数 <CameraID> = VT_I4:カメラ番号

<Property> = VT_I4:プロパティ

0	CameraControl_Pan	パン(度単位)
1	CameraControl_Tilt	傾き(度単位)
2	CameraControl_Roll	ロール(度単位)
3	CameraControl_Zoom	ズーム(ミリ単位)
4	CameraControl_Exposure	露出(2n 秒)
5	CameraControl_Iris	絞り(fstop * 10 単位)
6	CameraControl_Focus	焦点距離(ミリ単位)

<Value> = VT_I4:設定値

<Flag> = VT_I4:フラグ

1	CameraControl_Flags_Auto	自動制御
2	CameraControl_Flags_Manual	手動制御

戻り値 なし

説明 指定したカメラコントロールのパラメータ範囲を設定します.

詳細については MSDN の IAMCameraControl::Set()を参照してください.

[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 GetRangeCameraCtrl, GetCameraCtrl

GetRangeVideoProcAmp

構文 *object*.GetRangeVideoProcAmp(<CameraID>, <Property>)

引数 <CameraID> = VT_I4:カメラ番号

<Property> = VT_I4:プロパティ

0	VideoProcAmp_Brightness	輝度レベル
1	VideoProcAmp_Contrast	コントラスト(ゲイン係数×100)

2	VideoProcAmp_Hue	色相(度×100)
3	VideoProcAmp_Saturation	彩度
4	VideoProcAmp_Sharpness	鮮明度
5	VideoProcAmp_Gamma	ガンマ(ガンマ×100)
6	VideoProcAmp_ColorEnable	色の有効化設定 (0:OFF, 1:ON)
7	VideoProcAmp_WhiteBalance	ホワイトバランス
8	VideoProcAmp_BacklightCompensation	バックライト補正設定 (0:OFF, 1:ON)
9	VideoProcAmp_Gain	ゲイン調整

戻り値

<Min> = VT_I4:最小値

<Max> = VT_I4:最大値

<Step> = VT_I4:刻み

<Default> = VT_I4:デフォルト値

<Flag> = VT_I4:フラグ

1	CameraControl_Flags_Auto	自動制御
2	CameraControl_Flags_Manual	手動制御

説明

指定したカメラのビデオコントロールのパラメータ範囲を取得します.

詳細については MSDN の IAMVideoProcAmp::GetRange()を参照してください.

[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目

GetVideoProcAmp, SetVideoProcAmp

GetVideoProcAmp

構文

object.GetVideoProcAmp(<CameraID>, <Property>)

引数

<CameraID> = VT_I4:カメラ番号

<Property> = VT_I4:プロパティ

0	VideoProcAmp_Brightness	輝度レベル
1	VideoProcAmp_Contrast	コントラスト(ゲイン係数×100)
2	VideoProcAmp_Hue	色相(度×100)
3	VideoProcAmp_Saturation	彩度
4	VideoProcAmp_Sharpness	鮮明度
5	VideoProcAmp_Gamma	ガンマ(ガンマ×100)

6	VideoProcAmp_ColorEnable	色の有効化設定 (0:OFF, 1:ON)
7	VideoProcAmp_WhiteBalance	ホワイトバランス
8	VideoProcAmp_BacklightCompensation	バックライト補正設定 (0:OFF, 1:ON)
9	VideoProcAmp_Gain	ゲイン調整

戻り値

<Value> = VT_I4:設定値

<Flag> = VT_I4:フラグ

1	CameraControl_Flags_Auto	自動制御
2	CameraControl_Flags_Manual	手動制御

説明

指定したカメラのビデオコントロールのパラメータを取得します.

詳細については MSDN の IAMVideoProcAmp::Set()を参照してください.

[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目

GetRangeVideoProcAmp, SetVideoProcAmp

SetVideoProcAmp

構文

object.SetVideoProcAmp(<CameraID>, <Property>, <Value>, <Flag>)

引数

<CameraID> = VT_I4:カメラ番号

<Property> = VT_I4:プロパティ

0	VideoProcAmp_Brightness	輝度レベル
1	VideoProcAmp_Contrast	コントラスト(ゲイン係数×100)
2	VideoProcAmp_Hue	色相(度×100)
3	VideoProcAmp_Saturation	彩度
4	VideoProcAmp_Sharpness	鮮明度
5	VideoProcAmp_Gamma	ガンマ(ガンマ×100)
6	VideoProcAmp_ColorEnable	色の有効化設定 (0:OFF, 1:ON)
7	VideoProcAmp_WhiteBalance	ホワイトバランス
8	VideoProcAmp_BacklightCompensation	バックライト補正設定 (0:OFF, 1:ON)
9	VideoProcAmp_Gain	ゲイン調整

<Value> = VT_I4:設定値

<Flag> = VT_I4:フラグ

1	CameraControl_Flags_Auto	自動制御
2	CameraControl_Flags_Manual	手動制御

戻り値 なし

説明 指定したカメラのビデオコントロールのパラメータを設定します.
詳細については MSDN の IAMVideoProcAmp::Get()を参照してください.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 GetRangeVideoProcAmp, GetVideoProcAmp

GetCameraFormatList

構文 *object*.GetCameraFormatList(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 <Lists> = VT_VARIANT|VT_ARRAY:フォーマットリスト (<List1>, <List2>, …)
<Listn> = VT_I4|VT_ARRAY:フォーマット (<Format ID>,<Width>,<Height>)
<Format ID> = VT_I4:Camera format ID (0～)
<Width> = VT_I4:X 方向のカメラ解像度
<Hegiht> = VT_I4:Y 方向のカメラ解像度

説明 指定したカメラで設定できるフォーマットの一覧を取得します.
指定できないカメラフォーマットのリストは, すべて-1 の値が返ります
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 OpenPinProperty, SetCameraFormat

GetCameraFormat

構文 *object*.GetCameraFormat(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 <Format ID> = VT_I4:Camera format ID (0～)

説明 指定したカメラに設定されているフォーマットの ID を取得します.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 GetCameraFormatList

SetCameraFormat

構文 *object*.SetCameraFormat(<CameraID>, <Format ID>)

引数 <CameraID> = VT_I4:カメラ番号
<Format ID> = VT_I4:Camera format ID (0～)

戻り値 なし

説明 指定したカメラのフォーマットを設定します.
GetCameraFormatList で取得した際に「-1」のパラメータが返ってきた Format ID は使用しないでください.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 OpenPinProperty, GetCameraFormatList, GetCameraFormat

GetCameraFrameRate

構文 *object*.GetCameraFrameRate(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 <FrameRate> = VT_I4:カメラのフレームレート(FPS)

説明 指定したカメラのフレームレートを取得します.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 SetCameraFrameRate

SetCameraFrameRate

構文 *object*.SetCameraFrameRate(<CameraID>, <FrameRate>)

引数 <CameraID> = VT_I4:カメラ番号
<FrameRate> = VT_I4:カメラのフレームレート(FPS)

戻り値 なし

説明 指定したカメラのフレームレートを設定します.
設定できない値を設定した場合の挙動は, カメラドライバに依存します.
[注意] カメラのドライバによっては, 正常に動作しない可能性があります.

関連項目 GetCameraFrameRate

ExtExecSoftTrigger

[V1.4.6 以降]

構文 *object*.ExtExecSoftTrigger(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 なし

説明 カメラのソフトウェアトリガを実行します.
このコマンドは, 拡張カメラのみ使用することができます.
このコマンドは拡張カメラに対応した ORiN2 プロバイダの CaoController::Execute()の“OCV_ExecSoftTrigger”コマンドを実行します.

ExtRefreshImage

[V1.4.6 以降]

構文 *object*.ExtRefreshImage(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 なし

説明 拡張カメラの画像を更新します.
このコマンドは, 拡張カメラのみ使用することができます.
このコマンドは拡張カメラに対応した ORiN2 プロバイダの CaoController::Execute()の“OCV_GetImage”コマンドを実行し, 取得画像で内部バッファを更新します.

ExtInvoke

[V1.4.6 以降]

構文 *object*.ExtInvoke(<CameraID>, <Command>, <Parameter>)

引数 <CameraID> = VT_I4:カメラ番号

<Command> = VT_BSTR:コマンド名
<Parameter> = VT_VARIANT:パラメータ

戻り値 <Result> = VT_VARIANT:結果

説明 拡張カメラのコマンドを実行します.
このコマンドは, 拡張カメラのみ使用することができます.
このコマンドは拡張カメラに対応した ORiN2 プロバイダの CaoController::Execute()を実行します.
<Command>で指定可能なコマンド名及び<Parameter>及び<Result>の内容については拡張カメラに対応した ORiN2 プロバイダのユーザーズガイドを参照してください.

ExtConnect

[V1.5.1 以降]

構文 *object*.ExtConnect(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 <Connected> = VT_VARIANT:結果

TRUE	接続済み
FALSE	新規接続

説明 <CameraID>で指定した拡張カメラとの接続を行います.
拡張カメラが未接続の場合, このコマンドは拡張カメラの接続処理を行います. 成功すると戻り値に FALSE を返します.
指定した拡張カメラが接続済みの場合, このコマンドは処理を行わず, 必ず成功します. このとき戻り値に TRUE を返します.

関連項目 ExtDisconnect, ExtIsConnected

ExtDisconnect

[V1.5.1 以降]

構文 *object*.ExtDisconnect(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 なし

説明 <CameraID>で指定した拡張カメラとの切断を行います.
このコマンドは拡張カメラとの接続状態にかかわらず, 必ず成功します.

関連項目 ExtConnect, ExtIsConnected

ExtIsConnected

[V1.5.1 以降]

構文 *object*.ExtIsConnected(<CameraID>)

引数 <CameraID> = VT_I4:カメラ番号

戻り値 <Result> = VT_VARIANT:結果

TRUE	通信可能
FALSE	通信不可

説明 <CameraID>で指定した拡張カメラとの通信状態を確認します.
このコマンドは拡張カメラとの接続状態にかかわらず, 必ず成功します.
拡張カメラが接続状態かつ通信可能な場合, 戻り値として TRUE を返します. それ以外の場合は FALSE を返します.

関連項目 ExtConnect, ExtDisconnect

ExtGetConnectOption

[V1.5.2 以降]

構文 *object*. ExtGetConnectOption(<CameraID>)

引数 <CameraID> = VT_BSTR:カメラ番号

戻り値 <Parameter> = VT_BSTR:パラメータ

説明 <CameraID>で指定した拡張カメラの接続パラメータを取得します.

関連項目 ExtConnect, ExtDisconnect

ExtSetConnectOption

[V1.5.2 以降]

構文 *object*. ExtSetConnectOption(<CameraID>, <Parameter>)

引数 <CameraID> = VT_BSTR:カメラ番号
<Parameter> = VT_BSTR:パラメータ

戻り値 なし

説明 <CameraID>で指定した拡張カメラの接続パラメータを設定します.

関連項目 ExtConnect, ExtDisconnect

4.2. ファイルクラス

4.2.1. 一般

SetROI

構文 *object*.SetROI <ROI>

引数 <ROI>= VT_I4|VT_ARRAY:ROI 情報(<X>, <Y>, <W>, <H>)
<X> = VT_I4:始点の X 座標
<Y> = VT_I4:始点の Y 座標
<W> = VT_I4:範囲指定の幅
<H> = VT_I4:範囲指定の高さ

戻り値 なし

説明 画像処理範囲を設定します.
範囲指定設定後は, 座標に関する入力及び結果は<X>,<Y>で指定した始点を基準とした値になります.

関連項目 GetROI, ResetROI

用例 [VB6]

```
vntParam = Array(0, 0, 200, 100)
caoFile.Execute "SetROI", vntParam  ' (0,0) - (200,100) の範囲を基準とする
```

GetROI

構文 *object*.GetROI()

引数 なし

戻り値 <ROI>= VT_I4|VT_ARRAY:ROI 情報(<X>, <Y>, <W>, <H>)
<X> = VT_I4:始点の X 座標
<Y> = VT_I4:始点の Y 座標
<W> = VT_I4:範囲指定の幅
<H> = VT_I4:範囲指定の高さ

説明 SetROI コマンドで設定された値を取得します. ROI が設定されていないときは VT_EMPTY が返ります.

関連項目 SetROI, ResetROI

用例 [VB6]

```
' 現在の設定範囲を取得する
vntRet = caoFile.Execute( "GetROI" )
x = vntRet(0) ' <X>
y = vntRet(1) ' <Y>
w = vntRet(2) ' <W>
h = vntRet(3) ' <H>
```

ResetROI

構文 *object*.ResetROI()

引数 なし

戻り値 なし

説明 SetROI で指定した範囲をリセットします.

関連項目 SetROI, GetROI

用例 [VB6]

```
caoFile.Execute "ResetROI"
```

PutColor

構文 *object*.PutColor <Output ID>, <X>, <Y>, <R>, <G>, <B>

引数 <Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<R> = VT_I4:赤濃度
<G> = VT_I4:緑濃度
<B> = VT_I4:青濃度

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 指定した座標点の色を設定します.
グレースケール画像で実行するときは<B>の値を設定します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号のイメージメモリに変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー

画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 GetColor, SearchPoint

用例

[VB6]
```
'XY 位置(100,200)に赤で点描画し, 101 番のイメージに出力します.
vntParam = Array(101, 100, 200, 255, 0, 0)
caoFile.Execute "PutColor", vntParam
```

GetColor

構文 *object*.GetColor(<X>, <Y>)

引数

<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

戻り値

<Value>= VT_I4:輝度値 または VT_I4|VT_ARRAY:色濃度(<R>, <G>, <B>)

<R> = VT_I4:赤濃度

<G> = VT_I4:緑濃度

<B> = VT_I4:青濃度

説明

指定した座標点の色を取得します.

カラー画像の場合は, 色濃度(VT_I4|VT_ARRAY)を取得します.

グレースケール画像の場合は輝度値(VT_I4)を取得します.

関連項目 PutColor, SearchPoint

用例

[VB6]
```
'XY 位置(100,200)の色を取得します.
vntParam = Array(100, 200)
vntRet = caoFile.Execute( "GetColor", vntParam )
r = vntRet(0) ' <R>
g = vntRet(1) ' <G>
b = vntRet(2) ' <B>
```

SearchPoint

構文 *object*.SearchPoint(<Start X>, <Start Y>, <Direction>, <Search value>, <Condition>)

引数

<Start X> = VT_I4:始点 X 座標

<Start Y> = VT_I4:始点 Y 座標

<Direction> = VT_I4:探索方向

0	左
1	右
2	上
3	下

<Search value> = VT_I4:探索値

<Condition> = VT_I4:探索条件

0	一致	[ポイントデータ] = <Search value>
1	より大きい	[ポイントデータ] > <Search value>
2	未満	[ポイントデータ] < <Search value>

戻り値

<SerchPoint> = VT_I4|VT_ARRAY:探知座標

<X> = VT_I4:探知 X 座標

<Y> = VT_I4:探知 Y 座標

説明

座標検索を行います.

カラー画像の場合は, グレースケールに変換してから, 探索を行います.

<Condition>で指定した条件を最初に満たした座標点を取得します. 条件を満たす点を検出できないときは(-1, -1)を返します.

関連項目

PutColor, GetColor

用例

[VB6]

```
'XY 位置(10,20)の右方向へ 255 の値を持つポイントを検索します.
vntParam = Array(10, 20, 1, 255, 0)
vntRet = caoFile.Execute( "SearchPoint", vntParam )
x = vntRet(0) ' <X>
y = vntRet(1) ' <Y>
```

Trim

構文

object.Trim(<Threshold>, <Condition>)

引数

<Threshold> = VT_I4:閾値

<Condition> = VT_I4:条件

0	より大きい	[ポイントデータ] > <Threshold>
1	未満	[ポイントデータ] < <Threshold>

戻り値

<Area> =VT_I4|VT_ARRAY:トリミング範囲

<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

<W> = VT_I4:幅

<H> = VT_I4:高さ

説明 引数の条件を満たす範囲を囲むようにトリミングを行います.
カラー画像の場合はグレースケール画像に変換します.
条件を満たす範囲が見つからない時は, (-1,-1,-1,-1)を返します.

関連項目 SearchPoint, SetROI

用例 [VB6]

```
'閾値 128 以上の範囲を囲むようにトリミングします.
vntParam = Array(128, 0)
vntRet = caoFile.Execute( "Trim", vntParam )
x = vntRet(0) ' <X>
y = vntRet(1) ' <Y>
w = vntRet(2) ' <W>
h = vntRet(3) ' <H>
```

ImageSize

構文 *object*.ImageSize()

引数 なし

戻り値 <Size> = VT_I4|VT_ARRAY:画像サイズ

<W> = VT_I4:画像の幅

<H> = VT_I4:画像の高さ

説明 画像サイズを取得します.

用例 [VB6]

```
vntParam = Array(100, 200)
vntRet = caoFile.Execute( "ImageSize" )
w = vntRet(0) ' <W>
h = vntRet(1) ' <H>
```

IsColor

[V1.3.5 以降]

構文 *object*.IsColor()

引数 なし

戻り値 <IsColor> = VT_BOOL:画像の色

TRUE	カラー画像
FALSE	グレースケール画像

説明 画像がカラーであるかを判別します.

IsEmpty

[V1.4.0 以降]

構文 *object*.IsEmpty()

引数 なし

戻り値 <IsEmpty> = VT_BOOL:画像の存在

TRUE	画像なし
FALSE	画像あり

説明 画像が存在するかを判別します.

IsUpdated

[V1.4.5 以降]

構文 *object*.IsUpdated()

引数 なし

戻り値 <IsUpdated> = VT_BOOL:画像の更新状態

TRUE	更新あり(初期値)
FALSE	更新なし

説明 画像が更新されているかを判別します.
画像更新後に状態を更新なしの状態に戻す場合は“ClearUpdated”コマンドを実行してください. 初期値は更新ありです. “ClearUpdated”と組み合わせて使用して下さい.

ClearUpdated

[V1.4.5 以降]

構文 *object*.ClearUpdated()

引数 なし

戻り値 なし

説明 画像の更新フラグをクリアします.

Distance

構文 *object*.Distance<Point1>, <Point2>

引数 <Point1> = VT_I4|VT_ARRAY:座標点1 (<X>,<Y>,<Z>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Z> = VT_I4:Z 座標
<Point2> = VT_I4|VT_ARRAY:座標点 2 (<X>, <Y>, <Z>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Z> = VT_I4:Z 座標

戻り値 <Distance> = VT_R8:2 点間の距離

説明 <Point1>と<Point2>の距離を計算します.

InnerProduct

構文 *object*.InnerProduct <Vector1>, <Vector2>

引数 <Vector1> = VT_I4|VT_ARRAY:ベクトル1 (<X>,<Y>,<Z>)
<X> = VT_I4:X 要素
<Y> = VT_I4:Y 要素
<Z> = VT_I4:Z 要素
<Vector2> = VT_I4|VT_ARRAY:ベクトル 2 (<X>,<Y>,<Z>)
<X> = VT_I4:X 要素
<Y> = VT_I4:Y 要素
<Z> = VT_I4:Z 要素

戻り値 <Inner Product> = VT_R8:内積

説明 <Vector1>と<Vector2>の内積を計算します.

OuterProduct

構文 *object*.OuterProduct <Vector1>, <Vector2>

引数 <Vector1> = VT_I4|VT_ARRAY:ベクトル1 (<X>,<Y>,<Z>)
<X> = VT_I4:X 要素
<Y> = VT_I4:Y 要素
<Z> = VT_I4:Z 要素
<Vector2> = VT_I4|VT_ARRAY:ベクトル 2 (<X>,<Y>,<Z>)
<X> = VT_I4:X 要素
<Y> = VT_I4:Y 要素
<Z> = VT_I4:Z 要素

戻り値 <Outer product> = VT_R8|VT_ARRAY:外積 (<X>,<Y>,<Z>)
<X> = VT_I4:X 要素
<Y> = VT_I4:Y 要素
<Z> = VT_I4:Z 要素

説明 <Vector1>と<Vector2>の外積を計算します.

PutHelp

構文 *object*.PutHelp <Help>

引数 <Help> = VT_BSTR:文字列

戻り値 なし

説明 CaoFile::get_Help で取得できる文字列を設定します.
カメラ領域には使用できません.

4.2.2. 編集

Copy

構文 *object*.Copy <Output ID>

引数 <Output ID> = VT_I4:出力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像のコピーを行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号のイメージメモリに変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 Cut, Paste

Cut

構文 *object*.Cut <Output ID>, <X>, <Y>, <W>, <H>

引数 <Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<W> = VT_I4:幅
<H> = VT_I4:高さ

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像の切り取りを行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 Copy, Paste

Paste

構文 *object*.Paste <Output ID>, <Input ID>, <X>, <Y>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:貼り付けイメージ ID
<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <Input ID>で指定した画像を<X>,<Y>で指定した座標を始点にして貼り付けます.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 Copy, Cut

Rotate

構文 *object*.Rotate <Output ID>, <X>, <Y>, <Angle>, <Flag>

引数 <Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Angle> = VT_I4:回転角度(degree)
<Flag> = VT_I4:フラグ (<Warp> | <Interpolation>)

<Interpolation> =

0	CV_INTER_NN	最近隣接補間
1	CV_INTER_LINEAR	バイリニア補間
2	CV_INTER_AREA	ピクセル領域の関係を用いてリサンプリングする. 画像縮小の際は, モアレの無い処理結果を得ることができる手法である. 拡大の際は, CV_INTER_NN と同様
3	CV_INTER_CUBIC	バイキュービック補間

<Warp> =

8	CV_WARP_FILL_OUTLIERS	出力画像の全ピクセルの値を埋める. 対応ピクセルが入力画像外であるようなピクセルである場合は, 0 がセットされる.
16	CV_WARP_INVERSE_MAP	このフラグは map_matrix が出力画像から入力画像への逆変換のための行列であることを意味するので, 直接ピクセル補間に用いることができる. これがセットさ

		れていない場合，この関数は map_matrix を使って逆変換を計算する.

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像の回転を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します.
変換画像データは，Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します.
[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

関連項目 Resize, Flip

Flip

構文 *object*.Flip <Output ID>, <Mode>

引数 <Output ID> = VT_I4:出力イメージ ID
<Mode> = VT_I4:反転モード

0	Y 軸反転
1	X 軸反転
2	両軸反転

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像の反転を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します.
変換画像データは，Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します.

関連項目 Resize, Rotate

Resize

構文 *object*.Resize <Output ID>, <W>, <H>, <Interpolation>

引数 <Output ID> = VT_I4:出力イメージ ID
<W> = VT_I4:幅
<H> = VT_I4:高さ
<Interpolation> = VT_I4:補間方法

0	CV_INTER_NN	最近隣接補間
1	CV_INTER_LINEAR	バイリニア補間
2	CV_INTER_AREA	ピクセル領域の関係を用いてリサンプリングする. 画像縮小の際は, モアレの無い処理結果を得ることができる手法である. 拡大の際は, CV_INTER_NN と同様
3	CV_INTER_CUBIC	バイキュービック補間

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像のリサイズを行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 Rotate, Flip

Split

構文 *object*.Split <Output ID(R)>, <Output ID(G)>, <Output ID(B)>

引数 <Output ID(R)> = VT_I4:R 成分出力イメージ ID
<Output ID(G)> = VT_I4:G 成分出力イメージ ID
<Output ID(B)> = VT_I4:B 成分出力イメージ ID

戻り値 <Images> = VT_VARIANT|VT_ARRAY:分離画像
(<Image(R)>,<Image(G)>,<Image(B)>)
<Image(R)> = VT_UI1|VT_ARRAY:R 成分画像
<Image(G)> = VT_UI1|VT_ARRAY:G 成分画像
<Image(B)> = VT_UI1|VT_ARRAY:B 成分画像

説明 カラー画像を RGB の三成分に分離し, それぞれ<Output ID(R)>, <Output ID(G)>, <Output ID(B)>にグレイスケール画像として出力します.
グレースケール画像の場合はエラーを返します.

出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

関連項目 Merge

Merge

構文 *object*.Merge <Output ID>, <InputID(R)>, <InputID(G)>, <InputID(B)>

引数
<Output ID> = VT_I4:出力イメージ ID
<InputID(R)> = VT_I4:R 成分入力イメージ ID
<InputID(G)> = VT_I4:G 成分入力イメージ ID
<InputID(B)> = VT_I4:B 成分入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 グレースケール画像の RGB 三成分をそれぞれ<InputID(R)>, <InputID(G)>, <InputID(B)>とし, 合成画像を<Output ID>に出力します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 24bit ビットマップファイル形式で出力されます.

関連項目 Split

4.2.3. フィルター

ConvertGray

構文 *object*.ConvertGray <Output ID>

引数 <Output ID> = VT_I4:出力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 グレースケール変換を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

用例 [VB6]

```
caoFile.Execute “ConvertGray”, 101          ‘101 番のイメージに出力します.
```

ThresholdEx

構文　*object*.ThresholdEx <Output ID>, <Threshold>, <Max>, <Mode>

引数

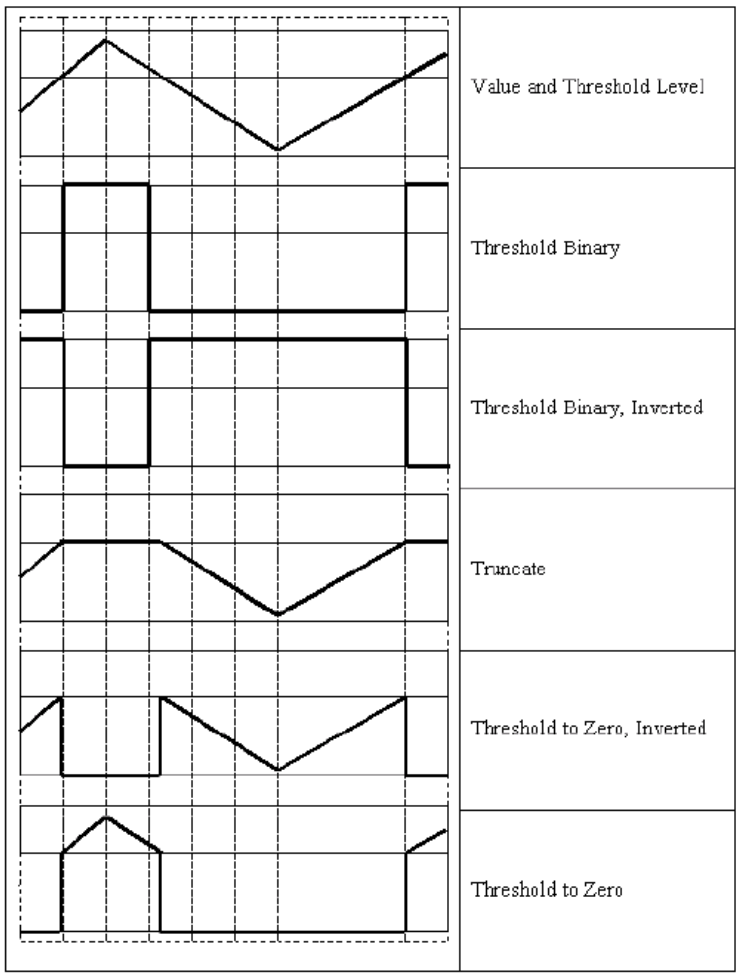
<Output ID> = VT_I4:出力イメージ ID

<Threshold> = VT_I4:閾値

<Max> = VT_I4:最大値

<Mode> = VT_I4:閾値処理タイプ

0	CV_THRESH_BINARY
1	CV_THRESH_BINARY_INV
2	CV_THRESH_TRUNC
3	CV_THRESH_TOZERO
4	CV_THRESH_TOZERO_INV



戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

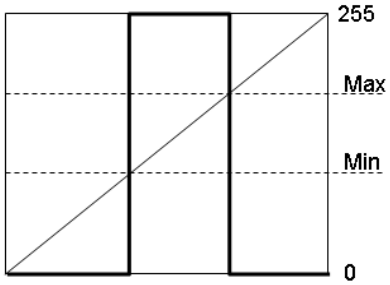
説明 閾値処理を行います.
カラー画像の場合は, グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

関連項目 Threshold2, AdaptiveThresholdEx

Threshold2

構文 *object*.Threshold2 <Output ID>, <Min>, <Max>

引数 <Output ID> = VT_I4:出力イメージ ID
<Min> = VT_I4:下限閾値
<Max> = VT_I4:上限閾値



戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 閾値処理(2 値化)を行います.
カラー画像の場合は, グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

関連項目 ThresholdEx, AdaptiveThresholdEx

AdaptiveThresholdEx

構文 *object*.AdaptiveThresholdEx <Output ID>, <Max value>, <Method>, <Type>, <Block size>, <Parameter>

引数

<Output ID> = VT_I4:出力イメージ ID

<Max value> = VT_I4:最大値

<Method> = VT_I4:適応的閾値処理アルゴリズム

0	CV_ADAPTIVE_THRESH_MEAN_C
1	CV_ADAPTIVE_THRESH_GAUSSIAN_C

<Type> = VT_I4:閾値処理タイプ

0	CV_THRESH_BINARY
1	CV_THRESH_BINARY_INV

<Block size> = VT_I4:隣接領域サイズ(3,5,7,・・・)

<Parameter> = VT_R8:パラメータ

戻り値

<Image> = VT_UI1|VT_ARRAY:変換画像

説明

適応的閾値処理を行います.

カラー画像の場合は, グレースケールに自動的に変換します.

出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.

変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

閾値は Method により求め方が異なり, それぞれ以下のようにして求めます.

CV_ADAPTIVE_THRESH_MEAN_C	注目ピクセルの block_size × block_size 隣接領域の平均から, param1 を引いた値.
CV_ADAPTIVE_THRESH_GAUSSIAN_C	注目ピクセルの block_size × block_size 隣接領域の重み付き総和(ガウシアン)から param1 を引いた値.

関連項目

Threshold2, ThresholdEx

Smooth

構文

object.Smooth <Output ID>, <Type>, <Parameter1>, <Parameter2>, <Parameter3>, <Parameter4>

引数

<Output ID> = VT_I4:出力イメージ ID

<Type> = VT_I4:平滑化タイプ

0	CV_BLUR_NO_SCALE (スケーリング無しの単純平滑化)	ピクセルの<Parameter1>×<Parameter2> 隣接の総和.
1	CV_BLUR (単純平滑化)	ピクセルの <Parameter1>×<Parameter2> 隣接の総和を計算した後, 1/(<Parameter1>·<Pparameter2>)によってスケーリングする.
2	CV_GAUSSIAN (ガウシアン平滑化)	画像とサイズ <Parameter1>×<Parameter2> のガウシアンカーネルの畳み込み.
3	CV_MEDIAN (中央値平滑化)	<Parameter1>×<Parameter1> 隣接(隣接形状が正方形)の中央値の取得.
4	CV_BILATERAL (エッジ保持平滑化フィルタ)	色(輝度値)について sigma=<Parameter1> と領域(距離)について sigma=<Parameter2> を持つ 3×3 のバイラテラルフィルタを適用する.

<Parameter1> = VT_I4:パラメータ 1

<Parameter2> = VT_I4:パラメータ 2

スケーリング有り/無しの単純平滑またはガウシアン平滑化の場合, <Parameter2> が 0 のときは <Parameter1> にセットされる.

<Parameter3> = VT_I4:パラメータ 3

ガウシアン平滑化 の場合, このパラメータがガウシアン σ (標準偏差)を示す. 0 の場合, 以下のようにカーネルサイズから計算する.

$$\sigma = \left(\frac{n}{2} - 1\right) \times 0.3 + 0.8$$

ここで 水平カーネルの場合は, n=<Parameter1>,

垂直カーネルの場合は, n=<Parameter2>

小さいサイズのカーネル (3×3 から 7×7 まで) の場合は標準偏差を用いる方がパフォーマンスが良い. <Parameter1> と <Parameter2>が 0 で, <Parameter3> が非0の場合, カーネルサイズは σ より計算される.

<Parameter4> = VT_I4:パラメータ 4

非正方形のガウシアンカーネルを使用する場合, 垂直方向に異なる σ 値(<Parameter3>と違う値)指定するために用いられる.

戻り値

<Image> = VT_UI1|VT_ARRAY:変換画像

説明

平滑化処理を行います.

出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.

変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー

画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．
スケーリング無しの平滑化は，グレースケール画像を対象としています．
それ以外の平滑化タイプはクレースケール画像，カラー画像の両方をサポートしています．

Sobel

構文

object.Sobel <Output ID>, <X order>, <Y order>, <Aperture>

引数

<Output ID> = VT_I4：出力イメージ ID
<X order> = VT_I4：x-導関数の次数
<Y order> = VT_I4：y-導関数の次数
<Aperture> = VT_I4：拡張 Sobel カーネルのサイズ

1,3,5,7 のいずれかである必要がある．aperture_size=1 の場合を除いて，<Aperture>×<Aperture>の，（二つのベクトルの積に）分離可能なカーネルが導関数の計算に用いられる． aperture_size=1 の場合は，3x1 あるいは 1x3 のカーネルが用いられる（ガウシアン（Gaussian）による平滑化は行わない). 特別な値である aperture_size=CV_SCHARR(=-1) もあり，3x3 Sobel よりも精度の良い結果が得られる 3x3 の Sharr のフィルタである．ここで Scharr のアパーチャは以下の通り．

$$\begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix}$$

これは x-導関数のためのカーネルであり，転置することによって y-導関数のためのカーネルとなる．

戻り値

<Image> = VT_UI1|VT_ARRAY：変換画像

説明

拡張 Sobel 演算子を用いて 1 次，2 次，3 次または混合次数の微分画像を計算します．
出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．
Sobel コマンドは以下のように，画像と適切なカーネルの畳み込みによって，微分画像を計算します．

$$dst(x,y) = \frac{d^{xorder+yorder} src}{dx^{xorder} \cdot dy^{yorder}}\Big|_{(x,y)}$$

Laplace

構文 *object*.Laplace <Output ID>, <Aperture>

引数 <Output ID> = VT_I4:出力イメージ ID
<Aperture> = VT_I4:アパーチャサイズ
1,3,5,7 のいずれかである必要がある. Sobel コマンドと同じ

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像のラプラシアン(Laplacian)を計算します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.
以下のように Sobel 演算子を用いて計算された x と y の 2 次微分を加算することで, 入力画像のラプラシアン(Laplacian)を計算する.

$$dst(x,y)=\frac{d^2src}{dx^2}+\frac{d^2src}{dy^2}$$

<aperture>=1 を指定した場合は, 以下のカーネルを用いた入力画像との畳み込みと同じ処理を, 高速に行う.

$$\begin{pmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{pmatrix}$$

CannyEx

構文 *object*.CannyEx <Output ID>, <Threshold1>, <Threshold2>, <Aperture>

引数 <Output ID> = VT_I4:出力イメージ ID
<Threshold1> = VT_I4:閾値 1
<Threshold2> = VT_I4:閾値 2
<Aperture> = VT_I4:アパーチャサイズ
3,5,7 のいずれかである必要がある. Sobel コマンドと同じ

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明

Canny フィルタ処理を行います.
カラー画像の場合は, グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.
<Threshold1>と<threshold2>のうち小さいほうがエッジ同士を接続するために用いられ, 大きいほうが強いエッジの初期検出に用いられます.

WarpAffine

構文

object.WarpAffine <Output ID>, <Ax>, <Bx>, <Dx>, <Ay>, <By>, <Dy>, <Flag>

引数

<Output ID> = VT_I4:出力イメージ ID
<Ax> = VT_I4:Affine 変換行列
<Bx>
<Dx>
<Ay>
<By>
<Dy>

$$\begin{pmatrix} Ax & Bx & Dx \\ Ay & By & Dy \end{pmatrix}$$

<Flag> = VT_I4:フラグ (<Warp> | <Interpolation>)

<Interpolation> =

0	CV_INTER_NN	最近隣接補間
1	CV_INTER_LINEAR	バイリニア補間
2	CV_INTER_AREA	ピクセル領域の関係を用いてリサンプリングする. 画像縮小の際は, モアレの無い処理結果を得ることができる手法である. 拡大の際は, CV_INTER_NN と同様
3	CV_INTER_CUBIC	バイキュービック補間

<Warp> =

8	CV_WARP_FILL_OUTLIERS	出力画像の全ピクセルの値を埋める. 対応ピクセルが入力画像外であるようなピクセルである場合は, 0 がセットされる.
16	CV_WARP_INVERSE_MAP	このフラグは map_matrix が出力画像から入力画像への逆変換のための行列で

		あることを意味するので，直接ピクセル補間に用いることができる．これがセットされていない場合，この関数は map_matrix を使って逆変換を計算する．

戻り値

<Image> = VT_UI1|VT_ARRAY：変換画像

説明

Affine 変換を行います．
出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

WarpPerspective

構文

object.WarpPerspective<Output ID>, <Extrinsic matrix>, <Flag>

引数

<Output ID> = VT_I4：出力イメージ ID
<Extrinsic matrix> = VT_R8|VT_ARRAY：変換行列
(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>)
<r11> = VT_R8：
<r21> = VT_R8：
<r31> = VT_R8：
<r12> = VT_R8：
<r22> = VT_R8：
<r32> = VT_R8：
<r13> = VT_R8：
<r23> = VT_R8：
<r33> = VT_R8：

$$\begin{pmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{pmatrix}$$

<Flag> = VT_I4：フラグ (<Warp>|<Interpolation>)
<Interpolation> =

0	CV_INTER_NN	最近隣接補間

1	CV_INTER_LINEAR	バイリニア補間
2	CV_INTER_AREA	ピクセル領域の関係を用いてリサンプリングする. 画像縮小の際は, モアレの無い処理結果を得ることができる手法である. 拡大の際は, CV_INTER_NN と同様
3	CV_INTER_CUBIC	バイキュービック補間

<Warp> =

8	CV_WARP_FILL_OUTLIERS	出力画像の全ピクセルの値を埋める. 対応ピクセルが入力画像外であるようなピクセルである場合は, 0 がセットされる.
16	CV_WARP_INVERSE_MAP	このフラグは map_matrix が出力画像から入力画像への逆変換のための行列であることを意味するので, 直接ピクセル補間に用いることができる. これがセットされていない場合, この関数は map_matrix を使って逆変換を計算する.

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 透視変換を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

PreCornerDetectEx

構文 *object*.PreCornerDetectEx <Output ID>, <Aperture>

引数 <Output ID> = VT_I4:出力イメージ ID
<Aperture> = VT_I4:アパーチャサイズ
3,5,7 のいずれかである必要がある. Sobel コマンドと同じ

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 コーナー検出を行います.
カラー画像の場合は, グレースケールに自動的に変換します.

出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.
PreCornerDetectEx コマンドは,

$$D_x^2 D_{yy} + D_y^2 D_{xx} - 2D_x D_y D_{xy}$$

を計算する. ここで D? は画像の 1 次微分を D??は画像の 2 次微分を表す. コーナーは, この関数の極大値を求めることで検出される.

CornerHarrisEx

構文 *object*.CornerHarrisEx <Output ID>, <Block size>, <Aperture>, <K>

引数
<Output ID> = VT_I4:出力イメージ ID
<Block size> = VT_I4:隣接ブロックサイズ
<Aperture> = VT_I4:アパーチャサイズ
3,5,7 のいずれかである必要がある. Sobel コマンドと同じ
<K> = VT_R8:Harris 検出器のパラメータ

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明
ハリスエッジ検出を行います.
カラー画像の場合は, グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

CalcBackProjectEx

構文 *object*.CalcBackProjectEx <Output ID>, <Input ID>

引数
<Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明
バックプロジェクションの算出を行います.
入力イメージからヒストグラムを作成し, バックプロジェクションの計算を行います.

出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

Inpaint

構文 *object*.Inpaint <Output ID>, <Mask ID>, <Range>, <Flag>

引数
<Output ID> = VT_I4:出力イメージ ID
<Mask ID> = VT_I4:修復マスク
グレースケール画像. 非 0 のピクセルが, 修復の必要がある領域であることを示す.
<Range> = VT_I4:隣接範囲
<Flag> = VT_I4:修復方法

0	CV_INPAINT_NS	ナビエ・ストークス ベースの手法
1	CV_INPAINT_TELEA	Alexandru Telea による手法

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明
画像修復処理を行います.
マスクデータは<MaskID>画像の輝度が 1 以上の値から作成されます.
カラー画像の場合は, グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

Erode

構文 *object*.Erode <Output ID>, <Interations>, <Cols>, <Rows>, <AnchorX>, <AnchorY>, <Shape>

引数
<Output ID> = VT_I4:出力イメージ ID
<Iterations> = VT_I4:収縮回数
<Cols> = VT_I4:構造要素の列数
<Rows> = VT_I4:構造要素の行数
<Anchor X> = VT_I4:アンカーポイントの水平方向相対オフセット値

<Anchor Y> = VT_I4:アンカーポイントの垂直方向相対オフセット値

<Shape> = VT_I4:構造要素の形状

0	CV_SHAPE_RECT	矩形の構造要素
1	CV_SHAPE_CROSS	十字の構造要素
2	CV_SHAPE_ELLIPSE	楕円の構造要素

戻り値

<Image> = VT_UI1|VT_ARRAY:変換画像

説明

隣接ピクセルの形状を決定する指定された構造要素を用いて，入力画像を収縮します．出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目

Dilate

Dilate

構文

object.Dilate <Output ID>, <Interations>, <Cols>, <Rows>, <AnchorX>, <AnchorY>, <Shape>

引数

<Output ID> = VT_I4:出力イメージ ID

<Iterations> = VT_I4:膨張回数

<Cols> = VT_I4:構造要素の列数

<Rows> = VT_I4:構造要素の行数

<Anchor X> = VT_I4:アンカーポイントの水平方向相対オフセット値

<Anchor Y> = VT_I4:アンカーポイントの垂直方向相対オフセット値

<Shape> = VT_I4:構造要素の形状

0	CV_SHAPE_RECT	矩形の構造要素
1	CV_SHAPE_CROSS	十字の構造要素
2	CV_SHAPE_ELLIPSE	楕円の構造要素

戻り値

<Image> = VT_UI1|VT_ARRAY:変換画像

説明

隣接ピクセルの形状を決定する指定された構造要素を用いて，入力画像を膨張します．出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目 Erode

PyrDown

構文 *object*.PyrDown <Output ID>

引数 <Output ID> = VT_I4:出力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 ガウシアンピラミッド分解の1ステップであるダウンサンプリングを行います． 最初に入力画像と指定されたフィルタの畳み込みを行い，偶数行と偶数列を間引くことでダウンサンプリングを行います．
出力画像の幅および高さは，入力画像の半分のサイズになります．
出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目 PyrUp

PyrUp

構文 *object*.PyrUp <Output ID>

引数 <Output ID> = VT_I4:出力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 ガウシアンピラミッド分解の1ステップであるアップサンプリングを行います． 最初に入力画像に 0 の行と列を挿入することでアップサンプリングを行った後，補間のために 4 倍した指定フィルタとの畳み込みを行う．
出力画像の幅および高さは，入力画像の2倍のサイズになります．
出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目 PyrDown

4.2.4. マスク

NOT

構文 *object*.NOT <Output ID>

引数 <Output ID> = VT_I4:出力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 画像のビット反転を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 AND, OR, XOR, ADD, SUB, MAXEx, MINEx, ABS

AND

構文 *object*.AND <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との論理積演算を行い, 変換画像を<Output ID>あるいは戻り値に出力します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 NOT, OR, XOR, ADD, SUB, MAXEx, MINEx, ABS

OR

構文 *object*.OR <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との論理和演算を行い，変換画像を<Output ID>あるいは戻り値に出力します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します.
変換画像データは，Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します.

関連項目 NOT, AND, XOR, ADD, SUB, MAXEx, MINEx, ABS

XOR

構文 *object*.XOR <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との排他的論理和演算を行い，変換画像を<Output ID>あるいは戻り値に出力します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します.
変換画像データは，Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します.

関連項目 NOT, AND, OR, ADD, SUB, MAXEx, MINEx, ABS

ADD

構文 *object*.ADD <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との加算を行い，変換画像を<Output ID>あるいは戻り値に出力します．
出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目 NOT, AND, OR, XOR, SUB, MAXEx, MINEx, ABS

SUB

構文 *object*.SUB <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との減算を行い，変換画像を<Output ID>あるいは戻り値に出力します．
出力イメージ ID が 0 のときは返り値に変換画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します．
変換画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目 NOT, AND, OR, XOR, ADD, MAXEx, MINEx, ABS

MAXEx

構文 *object*.MAXEx <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との比較を行い，より大きい値で構成される変換画像を<Output ID>あるいは戻り値に出力します.
カラー画像の場合は，グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します.
変換画像データは，Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します.

関連項目 NOT, AND, OR, XOR, ADD, SUB, MINEx, ABS

MINEx

構文 *object*.MINEx <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との比較を行い，より小さい値で構成される変換画像を<Output ID>あるいは戻り値に出力します.
カラー画像の場合は，グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは，指定した番号に変換画像を出力し，返り値に Empty を返します.
変換画像データは，Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します.

関連項目 NOT, AND, OR, XOR, ADD, SUB, MAXEx, ABS

ABS

構文 *object*.ABS <Output ID>, <Input ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像との減算の絶対値で構成される変換画像を<Output ID>あるいは戻り値

に出力します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.

変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 NOT, AND, OR, XOR, ADD, SUB, MAXEx, MINEx

LUT

[V1.3.5 以降]

構文 *object*.LUT <Output ID>, <LUT ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<LUT ID> = VT_I4:ルックアップテーブル番号

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 <InputID>画像に対しルックアップテーブルによる変換を行い, 変換画像を<Output ID>あるいは戻り値に出力します.
<InputID>画像がカラーの場合は, 各色相をルックアップテーブルの対応するテーブルで変換します.
<InputID>画像がグレースケールの場合は, ルックアップテーブルの青色相テーブルを使用して変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号のイメージメモリに変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 SetLUT, GetLUT

SetLUT

[V1.3.5 以降]

構文 *object*.SetLUT <LUT ID>, <Table R>, <Table G>, <Table B>

引数 <LUT ID> = VT_I4:ルックアップテーブル番号
<Table R> = VT_UI1|VT_ARRAY:赤色相のルックアップテーブル
<Table G> = VT_UI1|VT_ARRAY:緑色相のルックアップテーブル

<Table B> = VT_UI1|VT_ARRAY:青色相のルックアップテーブル

戻り値 なし

説明 指定したルックアップテーブルへ設定を行います.
各色相テーブルは, 256 要素必要になります.
色相テーブルを指定しないで VT_EMPTY を設定した場合は, そのテーブルの内容は変更されません.

関連項目 LUT, GetLUT

GetLUT

[V1.3.5 以降]

構文 *object*.GetLUT <LUT ID>

引数 <LUT ID> = VT_I4:ルックアップテーブル番号

戻り値 <LUT> = VT_VARIANT|VT_ARRRAY:ルックアップテーブル
(<Table R>, <Table G>, <Table B>)
<Table R> = VT_UI1|VT_ARRAY:赤色相のルックアップテーブル
<Table G> = VT_UI1|VT_ARRAY:緑色相のルックアップテーブル
<Table B> = VT_UI1|VT_ARRAY:青色相のルックアップテーブル

説明 指定したルックアップテーブルを取得します.

関連項目 LUT, SetLUT

4.2.5. 描画

Line

構文 *object*.Line <Output ID>, <Start X>, <Start Y>, <End X>, <End Y>, <R>, <G>, <B>, <Thick>, <Type>

引数 <Output ID> = VT_I4:出力イメージ ID
<Start X> = VT_I4:始点 X 座標
<Start Y> = VT_I4:始点 Y 座標
<End X> = VT_I4:終点 X 座標
<End Y> = VT_I4:終点 Y 座標
<R> = VT_I4:色の R 成分

<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分
16	アンチエイリアスされた線分

戻り値

<Image> = VT_UI1|VT_ARRAY:描画画像

説明

現在の画像に対して線描画を行い，描画画像を<Output ID>あるいは戻り値に出力します．
グレースケール画像の場合，<B>の値で描画します．
出力イメージ ID が 0 のときは返り値に描画画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号のイメージメモリに描画画像を出力し，返り値に Empty を返します．
描画画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

関連項目

Line2

Line2

構文

object.Line2 <Output ID>, <Start X>, <Start Y>, <Length>, < Angle >, <R>, <G>, <B>, <Thick>, <Type>

引数

<Output ID> = VT_I4:出力イメージ ID
<Start X> = VT_I4:始点 X 座標
<Start Y> = VT_I4:始点 Y 座標
<Length> = VT_I4:長さ
<Angle> = VT_I4:回転角度(degree)
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分

16	アンチエイリアスされた線分

戻り値 <Image> = VT_UI1|VT_ARRAY:描画画像

説明 現在の画像に対して線描画を行い，描画画像を<Output ID>あるいは戻り値に出力します．
グレースケール画像の場合，<B>の値で描画します．
出力イメージ ID が 0 のときは返り値に描画画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に描画画像を出力し，返り値に Empty を返します．
描画画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．
[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています．

関連項目 Line

Rectangle

構文 *object*.Rectangle <Output ID>, <Start X>, <Start Y>, <End X>, <End Y>, <R>, <G>, <B>, <Thick>, <Type>

引数
<Output ID> = VT_I4:出力イメージ ID
<Start X> = VT_I4:始点 X 座標
<Start Y> = VT_I4:始点 Y 座標
<End X> = VT_I4:終点 X 座標
<End Y> = VT_I4:終点 Y 座標
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分
16	アンチエイリアスされた線分

戻り値 <Image> = VT_UI1|VT_ARRAY:描画画像

説明 現在の画像に対して四角の描画を行い，描画画像を<Output ID>あるいは戻り値に出力します．

グレースケール画像の場合, <B>の値で描画します.
出力イメージ ID が 0 のときは返り値に描画画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に描画画像を出力し, 返り値に Empty を返します.
描画画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

Circle

構文 *object*.Circle <Output ID>, <X>, <Y>, <Radius>, <R>, <G>, <B>, <Thick>, <Type>

引数
<Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標
<Radius> = VT_I4:半径
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分
16	アンチエイリアスされた線分

戻り値 <Image> = VT_UI1|VT_ARRAY:描画画像

説明 現在の画像に対して円の描画を行い, 描画画像を<Output ID>あるいは戻り値に出力します.
グレースケール画像の場合, <B>の値で描画します.
出力イメージ ID が 0 のときは返り値に描画画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に描画画像を出力し, 返り値に Empty を返します.
描画画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

Ellipse

構文

object.Ellipse <Output ID>, <X>, <Y>, <X radius>, <Y radius>, <Angle>, <Start angle>, <EndAngle>, <R>, <G>, <B>, <Thick>, <Type>

引数

<Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標
<X radius> = VT_I4:X 軸半径
<Y radius> = VT_I4:Y 軸半径
<Angle> = VT_I4:回転角度(degree)
<Start angle> = VT_I4:開始角度(degree)
<End angle> = VT_I4:終了角度(degree)
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分
16	アンチエイリアスされた線分

戻り値

<Image> = VT_UI1|VT_ARRAY:描画画像

説明

現在の画像に対して楕円の描画を行い，描画画像を<Output ID>あるいは戻り値に出力します．
グレースケール画像の場合，<B>の値で描画します．
出力イメージ ID が 0 のときは返り値に描画画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に描画画像を出力し，返り値に Empty を返します．
描画画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．
詳細については，OpenCV リファレンスの Ellipse の項目を参照してください．
[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています．

Sector

構文

object.Sector <Output ID>, <X>, <Y>, <X radius>, <Y radius>, <Angle>, <Start

angle>, <End angle>, <R>, <G>, <B>, <Thick>, <Type>

引数

<Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標
<X radius> = VT_I4:X 軸半径
<Y radius> = VT_I4:Y 軸半径
<Angle> = VT_I4:回転角度(degree)
<Start angle> = VT_I4:開始角度(degree)
<End angle> = VT_I4:終了角度(degree)
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分
16	アンチエイリアスされた線分

戻り値

<Image> = VT_UI1|VT_ARRAY:描画画像

説明

現在の画像に対して扇形の描画を行い, 描画画像を<Output ID>あるいは戻り値に出力します.
グレースケール画像の場合, <B>の値で描画します.
出力イメージ ID が 0 のときは返り値に描画画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に描画画像を出力し, 返り値に Empty を返します.
描画画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.
[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

Cross

構文

object.Cross <Output ID>, <X>, <Y>, <XRadius>, <YRadius>, <Angle>, <R>, <G>, <B>, <Thick>, <Type>

引数

<Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標

<XRadius> = VT_I4:X 軸半径
<YRadius> = VT_I4:Y 軸半径
<Angle> = VT_I4:回転角度(degree)
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Thick> = VT_I4:太さ
<Type> = VT_I4:線種

0,8	8 連結による線分
4	4 連結による線分
16	アンチエイリアスされた線分

戻り値

<Image> = VT_UI1|VT_ARRAY:描画画像

説明

現在の画像に対して十字形の描画を行い, 描画画像を<Output ID>あるいは戻り値に出力します.
グレースケール画像の場合, <B>の値で描画します.
出力イメージ ID が 0 のときは返り値に描画画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に描画画像を出力し, 返り値に Empty を返します.
描画画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.
[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

Text

構文

object.Text <Output ID>, <X>, <Y>, <Text>, <R>, <G>, <B>, <Font>, <H scale>, <V scale>, <Shear>, <Thick>

引数

<Output ID> = VT_I4:出力イメージ ID
<X> = VT_I4:始点 X 座標
<Y> = VT_I4:始点 Y 座標
<Text> = VT_BSTR:表示テキスト
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分
<Font> = VT_I4:フォント種別

0	CV_FONT_HERSHEY_	普通サイズの sans-serif フォント

	SIMPLEX	
1	CV_FONT_HERSHEY_PLAIN	小さいサイズの sans-serif フォント
2	CV_FONT_HERSHEY_DUPLEX	普通サイズの sans-serif フォント（CV_FONT_HERSHEY_SIMPLEX よりも複雑）
3	CV_FONT_HERSHEY_COMPLEX	普通サイズの serif フォント
4	CV_FONT_HERSHEY_TRIPLEX	普通サイズの serif フォント（CV_FONT_HERSHEY_COMPLEX よりも複雑）
5	CV_FONT_HERSHEY_COMPLEX_SMALL	CV_FONT_HERSHEY_COMPLEX の小さいバージョン
6	CV_FONT_HERSHEY_SCRIPT_SIMPLEX	手書きスタイルのフォント
7	CV_FONT_HERSHEY_SCRIPT_COMPLEX	CV_FONT_HERSHEY_SCRIPT_SIMPLEX の複雑なバージョン

<H scale> = VT_R8:幅比率

1.0f にした場合，文字はそれぞれのフォントに依存する元々の幅で表示される．

0.5f にした場合，文字は元々の半分の幅で表示される．

<V scale> = VT_R8:高さ比率

1.0f にした場合，文字はそれぞれのフォントに依存する元々の高さで表示される．

0.5f にした場合，文字は元々の半分の高さで表示される．

<Shear> = VT_R8:垂直線からの相対的角度

ゼロの場合は非イタリックフォントで，例えば，1.0f は≈45° を意味する．

<Thick> = VT_I4:太さ

戻り値

<Image> = VT_UI1|VT_ARRAY:描画画像

説明

現在の画像に対してテキスト文字列の描画を行い，描画画像を<Output ID>あるいは戻り値に出力します．

グレースケール画像の場合，<B>の値で描画します．

<Shear>は斜体の度合いを意味し，0.0 で 0 度，1.0 で 45 度です．

出力イメージ ID が 0 のときは返り値に描画画像を出力します．出力イメージ ID が 0 以外のときは，指定した番号に描画画像を出力し，返り値に Empty を返します．

描画画像データは，Windows 標準のビットマップファイル形式で出力されます．カラー画像は 24bit ビットマップ，グレースケール画像は 8bit ビットマップで出力します．

4.2.6. 輪郭

FindContoursEx

構文

object.FindContoursEx(<Mode>, <Method>)

引数

<Mode> = VT_I4:抽出モード

0	CV_RETR_EXTERNAL	最も外側の輪郭のみ抽出
1	CV_RETR_LIST	全ての輪郭を抽出し，リストに追加
2	CV_RETR_CCOMP	全ての輪郭を抽出し，二つのレベルを持つ階層構造を構成する．1 番目のレベルは連結成分の外側の境界線，2 番目のレベルは穴（連結成分の内側に存在する）の境界線．
3	CV_RETR_TREE	全ての輪郭を抽出し，枝分かれした輪郭を完全に表現する階層構造を構成する．

<Method> = VT_I4:近似手法

0	CV_CHAIN_CODE	出力はフリーマンチェーンコード（Freeman chain code）で表現される．他の手法ではポリゴン（頂点のシーケンス）．
1	CV_CHAIN_APPROX_NONE	全ての点をチェーンコードから点へ変換する．
2	CV_CHAIN_APPROX_SIMPLE	水平・垂直・斜めの線分を圧縮する．すなわち，この関数はそれぞれの端点のみを残す．
3	CV_CHAIN_APPROX_TC89_L1	Teh-Chin チェーンの近似アルゴリズム中の一つを適用する．
4	CV_CHAIN_APPROX_TC89_KCOS	Teh-Chin チェーンの近似アルゴリズム中の一つを適用する．
5	CV_LINK_RUNS	値 1 の水平セグメントの接続に基づく，異なる輪郭を抽出する全く異なるアルゴリズムを適用する．抽出モードが CV_RETR_LIST の場合のみ指定可能．

戻り値

<Count> = VT_I4:検知輪郭数

説明 輪郭検知を行います.
カラー画像の場合は, グレースケールに自動的に変換します.
検出した輪郭には, 0 から順番に輪郭番号が割り当てられます.

CopyContours

構文 *object*.CopyContours <Output ID>, <Contour ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Contour ID> = VT_I4:輪郭番号

戻り値 <Image> = VT_UI1|VT_ARRAY:輪郭抽出画像

説明 輪郭画像のコピーを行います.
このコマンドは, 指定した輪郭にフィットする矩形サイズで画像を範囲指定保存します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します.
出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
描画画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.
FindContoursEx コマンドを実行するまでは, このコマンドはエラーになります.

エラー 0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

ContoursNumber

構文 *object*.ContoursNumber(<X>, <Y>)

引数 <X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標

戻り値 <Contour ID> = VT_I4:輪郭番号

説明 輪郭番号の検索を行います.
指定座標が輪郭番号に対応していない場合は, 0 を返します.
FindContoursEx コマンドを実行するまでは, このコマンドはエラーになります.

エラー　0x80101001 : 輪郭検出がされていません．“FindContoursEx”を実行してください．

上記以外のエラーについては，2.4 を参照してください．

PointPolygonTest

構文　*object*.PointPolygonTest(<Contour ID>, <X>, <Y>, <Measure distance>)

引数　<Contour ID> = VT_I4:輪郭番号

<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

<Measure distance> = VT_I4:距離計測フラグ

0	距離計測なし
0 以外	距離計測あり

戻り値　<Distance> = VT_R8:計測距離

説明　点と輪郭の関係を調べます．

このコマンドの戻り値の距離により，負の値なら内側，正の値なら外側，0 であるなら輪郭上にあることが求まります．

距離計測を行う場合，計測距離は最近傍輪郭線までの距離になります．

FindContoursEx コマンドを実行するまでは，このコマンドはエラーになります．

エラー　0x80101001 : 輪郭検出がされていません．“FindContoursEx”を実行してください．

上記以外のエラーについては，2.4 を参照してください．

BoundingRect

構文　*object*.BoundingRect(<Contour ID>)

引数　<Contour ID> = VT_I4:輪郭番号

戻り値　<Rectangle> = VT_I4|VT_ARRAY: 輪郭を包含する矩形(<X>, <Y>, <W>, <H>)

<X> = VT_I4:矩形の左上 X 座標

<Y> = VT_I4:矩形の左上 Y 座標

<W> = VT_I4:幅

<H> = VT_I4:高さ

説明 輪郭を包含するまっすぐな矩形を求めます.
FindContoursEx コマンドを実行するまでは, このコマンドはエラーになります.

エラー 0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

FitEllipse

構文 *object*.FitEllipse(<Contour ID>)

引数 <Contour ID> = VT_I4:輪郭番号

戻り値 <Ellipse> = VT_VARIANT|VT_ARRAY:輪郭にフィットする最良楕円
(<X>,<Y>,<W>,<H>,<Agnle>)

<X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標
<W> = VT_I4:幅
<H> = VT_I4:高さ
<Angle> = VT_R4:回転角度(degree)

説明 指定輪郭にフィットする最良の楕円を取得します.
FindContoursEx コマンドを実行するまでは, このコマンドはエラーになります.
[備考] 戻り値の内容がEllipse の引数と異なります.

エラー 0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

ArcLength

構文 *object*.ArcLength(<Contour ID>)

引数 <Contour ID> = VT_I4:輪郭番号

戻り値 <Length> = VT_R8:輪郭の周囲長

説明 輪郭の周囲の長さを取得します.
FindContoursEx コマンドを実行するまでは, このコマンドはエラーになります.

エラー 0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

CheckContourConvexity

構文 *object*.CheckContourConvexity(<Contour ID>)

引数 <Contour ID> = VT_I4:輪郭番号

戻り値 <Convexity> = VT_I4:凸型判定

説明 輪郭が凸型であるかを判定します.
このコマンドの戻り値が 0 であるとき凹型, 1 のときは凸型となります.
FindContoursEx コマンドを実行するまでは, このコマンドはエラーになります.

エラー 0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

DrawContours

V1.3.5 以降

構文 *object*.DrawContours <Output ID>, <InputID>, <Contour ID>, <External R>, <External G>, <External B>, <Hole R>, <Hole G>, <Hole B>, <Max level>, <Thick>, <Type>, <Offset X>, <Offset Y>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID> = VT_I4:入力イメージ ID
<Contour ID> = VT_I4:輪郭番号
<External R> = VT_I4:外側輪郭線の R 成分
<External G> = VT_I4:外側輪郭線の G 成分
<External B> = VT_I4:外側輪郭線の B 成分
<Hole R> = VT_I4:内側輪郭線の R 成分
<Hole G> = VT_I4:内側輪郭線の G 成分
<Hole B> = VT_I4:内側輪郭線の B 成分

<Max level> = VT_I4:描画される輪郭の最大レベル.

0	contour のみが描画される
0 <	contour の同レベルの輪郭と contour の子の輪郭を abs(max_level)-1 のレベルまで描画する.
0 >	contour の後に続く同レベルの輪郭を描画しないが, contour の子の輪郭を abs(max_level)-1 のレベルまで描画する.

<Thick> = VT_I4:太さ
<Type> = VT_I4:線種
<Offset X> = VT_I4:X 方向オフセット
<Offset Y> = VT_I4:Y 方向オフセット

戻り値

<Image> = VT_UI1|VT_ARRAY:描画画像

説明

現在の画像に対して外側輪郭線, または内側輪郭線を描画の描画を行い, 描画画像を<Output ID>あるいは戻り値に出力します.
グレースケール画像の場合, <B>の値で描画します.
出力イメージ ID が 0 のときは返り値に描画画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に描画画像を出力し, 返り値に Empty を返します.
描画画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

エラー

0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.

上記以外のエラーについては, 2.4 を参照してください.

4.2.7. ブロブ

FindBlobs

構文

object.FindBlobs(<Mask ID>, <Threshold>, <Moments>, [<IBlobCmt>])

引数

<Mask ID> = VT_I4:マスクイメージ ID
<Threshold> = VT_I4:閾値
<Moments> = VT_BOOL:モーメント算出実行フラグ
<IBlobCmt> = 検出ブロブ数の制限値 (デフォルト値:100)

戻り値

<Count> = VT_I4:検知ブロブ数

説明 ブロブ検知を行います.
カラー画像の場合は, グレースケールに自動的に変換します.
マスクイメージ ID が 0 のときは, マスク処理は無効になります.
モーメント算出が True のときは検出したブロブ毎にモーメントを算出します.
検出したブロブには, 0 から順番にブロブ番号が割り当てられます.
検出したブロブ数が制限値(100)を超えた場合, エラーを返します.

エラー 0x80101002 : 検出したブロブ数が制限値を超えました.
上記以外のエラーについては, 2.4 を参照してください.

関連項目 BlobsFilter, BlobResult, BlobResults, BlobEllipse, BlobMatchTemplate, BlobMatchShapes

BlobsFilter

構文 *object*.BlobsFilter <Action>, <Evaluador>, <Condition>, <Low limit>, <High limit>

引数 <Action> = VT_I4:フィルターアクション

0	Include
1	Exclude

<Evaluador> = VT_I4:評価項目

0	Area	1	AreaElipseRatio
2	AxisRatio	3	Breadth
4	Compactness	5	DiffX
6	DiffY	7	DistanceFromPoint
8	Elongation	9	Exterior
10	ExternHullPerimeterRatio	11	ExternPerimeter
12	ExternPerimeterRatio	13	HullArea
14	HullPerimeter	15	Length
16	MajorAxisLength	17	MaxX
18	MaxXatMaxY	19	MaxY
20	MaxYatMinX	21	Mean
22	MinorAxisLength	23	MinX
24	MinXatMinY	25	MinY
26	MinYatMaxX	27	Moment
28	Orientation	29	OrientationCos

30	Perimeter	31	Roughness
32	StdDev	33	XCenter
34	XYInside	35	Ycenter

<Condition> = VT_I4:フィルター条件

0	Equal	1	Not Equal
2	Greater	3	Less
4	Greater or equal	5	Less or equal
6	Inside	7	Outside

<Low limit> = VT_I4:下限値
<High limit> = VT_I4:上限値

戻り値 <Count> = VT_I4:フィルター後のブロブ数

説明 FindBlobs で検出したブロブリストにフィルターをかけます.
上限値を使用しないときは 0 を指定してください.

エラー 0x80101001 : ブロブ検出がされていません. “FindBlobs”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

関連項目 FindBlobs

BlobResult

構文 *object*.BlobResult(<Blob ID>, <Evaluador>, <Parameter1>, <Parameter2>)

引数 <Blob ID> = VT_I4:ブロブ番号
<Evaluador> = VT_I4:取得項目

0	Area	1	AreaElipseRatio
2	AxisRatio	3	Breadth
4	Compactness	5	DiffX
6	DiffY	7	DistanceFromPoint
8	Elongation	9	Exterior
10	ExternHullPerimeterRatio	11	ExternPerimeter
12	ExternPerimeterRatio	13	HullArea
14	HullPerimeter	15	Length
16	MajorAxisLength	17	MaxX
18	MaxXatMaxY	19	MaxY

20	MaxYatMinX	21	Mean
22	MinorAxisLength	23	MinX
24	MinXatMinY	25	MinY
26	MinYatMaxX	27	Moment
28	Orientation	29	OrientationCos
30	Perimeter	31	Roughness
32	StdDev	33	XCenter
34	XYInside	35	Ycenter

<Parameter1> = VT_I4:パラメータ 1

<Parameter2> = VT_I4:パラメータ 2

戻り値

<Value> = VT_I4:指定した項目の値

説明

ブロブ検出結果から指定した項目を取得します.

FindBlobs コマンドを実行するまでは, このコマンドはエラーになります.

<Parameter1>及び<Parameter2>の内容は<Evaluador>の値によって異なります. 以下に設定内容の一覧を示します.

表 4-4 BlobResult の引数指定

Evluador	Parameter1	Parameter2
7:DistanceFromPoint	指定点の X 座標	指定点の Y 座標
27:Moment	X 微分回数	Y 微分回数
34:XYInside	指定点の X 座標	指定点の Y 座標
その他	なし	なし

エラー

0x80101001 : ブロブ検出がされていません. “FindBlobs”を実行してください.

上記以外のエラーについては, 2.4 を参照してください.

関連項目

FindBlobs

BlobResults

構文

object.BlobResults(<Blob ID>)

引数

<Blob ID> = VT_I4:ブロブ番号

戻り値

<Result> = VT_VARIANT|VT_ARRAY:ブロブ検出結果

(**<Label>**,**<Exterior>**,**<Perimeter>**,**<External perimeter>**,**<Parent>**,

<M00>,<M10>,<M01>,<M20>,<M11>,<M02>,<Min X>,<Max X>,<Min Y>,
<MaxY>,<Mean >,<StdDev>)

<Label> = VT_I4:ブロブのラベル番号.

<Exterior> = VT_I4:外側フラグ

<Perimeter> = VT_R8:外周

<External perimeter> = VT_R8:外部の周辺

<Parent> = VT_I4:親ブロブのラベル番号

<M00> = VT_R8:モーメント

<M10>

<M01>

<M20>

<M11>

<M02>

<Min X> = VT_R8:外接矩形

<Max X>

<Min Y>

<Max Y>

<Mean> = VT_R8:グレースケール時の平均画素値

<StdDev> = VT_R8:グレースケール時の画素値の標準偏差

説明 指定したブロブ検出結果を取得します.
FindBlobs コマンドを実行するまでは, このコマンドはエラーになります.

エラー 0x80101001 : ブロブ検出がされていません. “FindBlobs”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

関連項目 FindBlobs

BlobEllipse

構文 *object*.BlobEllipse(<Blob ID>)

引数 <Blob ID> = VT_I4:ブロブ番号

戻り値 <X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標
<W> = VT_I4:幅
<H> = VT_I4:高さ

<Angle> = VT_I4:回転角度(degree)

説明 指定ブロブの最小楕円を取得します.
FindBlobs コマンドを実行するまでは, このコマンドはエラーになります.

エラー 0x80101001 : ブロブ検出がされていません. “FindBlobs”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

関連項目 FindBlobs

BlobMatchTemplate

構文 *object*.BlobMatchTemplate(<Input ID>, <Method>, <Threshold>, <Start angle>, <End angle>, <Step angle>, <Down sizing>, <Max counts>, <Min distance>)

引数 <Input ID> = VT_I4:テンプレート画像のイメージ ID
<Method> = VT_I4:マッチング方法
(I は画像を, T はテンプレートを, R は結果をそれぞれ表す. 総和計算は, 以下のようにテンプレートと(または)画像領域に対して行われる. x'=0..w-1, y'=0..h-1)

0	CV_TM_SQDIFF	$R(x,y)=\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2$
1	CV_TM_SQDIFF_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
2	CV_TM_CCORR	$R(x,y)=\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')]$
3	CV_TM_CCORR_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')]}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
4	CV_TM_CCOEFF	$R(x,y)=\sum_{x',y'}[T'(x',y')\cdot I'(x+x',y+y')]$ このとき

		$T'(x',y') = T(x',y') - \frac{\sum_{x'',y''} T(x'',y'')}{(w \cdot h)}$ $I'(x+x',y+y') = I(x+x',y+y') - \frac{\sum_{x'',y''} I(x+x'',y+y'')}{(w \cdot h)}$
5	CV_TM_CCOEFF_NORMED	$R(x,y) = \frac{\sum_{x',y'} [T'(x',y') \cdot I'(x+x',y+y')]}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$

<Threshold> = VT_R8:閾値
<Start angle> = VT_I4:検索開始角度(degree)
<End angle> = VT_I4:検索終了角度(degree)
<Step angle> = VT_I4:角度刻み(degree)
<Down sizing> = VT_I4:画面縮小回数
<Max counts> = VT_I4:検出数
<Min distance> = VT_I4:近傍判定距離(0:テンプレート画像サイズ)

戻り値

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>,<Angle>,<Value>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Angle> = VT_R8:回転角(degree)
<Value> = VT_R8:相関値

説明

検出した各ブロブと<Input ID>画像の拡張テンプレートマッチング処理を行います.
拡張テンプレートマッチング処理は, MatchTemplate2 コマンドと同じ処理を行います.
FindBlobs コマンドを実行するまでは, このコマンドはエラーになります.
[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

エラー

0x80101001 : ブロブ検出がされていません. “FindBlobs”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

関連項目

FindBlobs, MatchTemplate2

高速化 Tips

・ BlobsFilter を使用して, 検索対象となる検出 Blob 数を減らす.

BlobMatchShapes

構文

object.BlobMatchShapes(<Input ID>, <Method>, <Min scale>, <Similarity>, <Max counts>)

引数

<Input ID> = VT_I4:テンプレート画像のイメージ ID

<Method> = VT_I4:マッチング方法

ここで A は元画像, B はテンプレート画像を示す

0	CV_CONTOUR_MATCH_I1	$I_1(A,B)=\sum_{i=1}^{7}\left\lvert\frac{1}{m^A{}_i}-\frac{1}{m^B{}_i}\right\rvert$
1	CV_CONTOUR_MATCH_I2	$I_2(A,B)=\sum_{i=1}^{7}\left\lvert m^A{}_i-m^B{}_i\right\rvert$
2	CV_CONTOUR_MATCH_I3	$I_3(A,B)=\sum_{i=1}^{7}\frac{\left\lvert m^A{}_i-m^B{}_i\right\rvert}{\left\lvert m^A{}_i\right\rvert}$

ここで

$$m^A{}_i=\sin\left(h^A{}_i\right)\cdot\log\left(h^A{}_i\right)$$
$$m^B{}_i=\sin\left(h^B{}_i\right)\cdot\log\left(h^B{}_i\right)$$

$h^A{}_i$, $h^B{}_i$ は, A と B それぞれの Hu モーメント.

<Min scale> = VT_R8:最小倍率

<Similarity> = VT_R8:輪郭の一致度

<Max counts> = VT_I4:検出数

戻り値

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)

<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>,<Angle>,<Value>)

<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

<Angle> = VT_I4:回転角(degree)

<Value> = VT_R8:輪郭の一致度

説明

検出した各ブロブと<Input ID>画像の拡張形状比較処理を行います.

拡張テンプレートマッチング処理は, MatchShapes2 コマンドと同じ処理を行います.

FindBlobs コマンドを実行するまでは, このコマンドはエラーになります.

エラー

0x80101001 : ブロブ検出がされていません. “FindBlobs”を実行してください.

上記以外のエラーについては, 2.4 を参照してください.

関連項目 FindBlobs, MatchShapes2

4.2.8. ヒストグラム

CalcHistEx

構文 *object*.CalcHistEx(<Size>)

引数 <Size> = VT_I2:ヒストグラムの要素数

戻り値 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

説明 ヒストグラムの計算を行います.
カラー画像の場合は, グレースケールに自動的に変換します.

関連項目 NormalizeHistEx, ThreshHistEx, HistAve, AutoThreshPTile, AutoThreshMode, AutoThreshDiscrim

NormalizeHistEx

構文 *object*.NormalizeHistEx(<Histogram>, <Factor>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム
<Factor> = VT_R8:正規化係数

戻り値 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

説明 ヒストグラムの正規化を行います.

関連項目 CalcHistEx

ThreshHistEx

構文 *object*.ThreshHistEx(<Histogram>, <Threshold>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム
<Threshold> = VT_R8:閾値レベル

戻り値 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

説明 ヒストグラムの指定した閾化以下のヒストグラムのビンをクリアします.

関連項目 CalcHistEx

EqualizeHistEx

構文 *object*.EqualizeHistEx <Output ID>

引数 <Output ID> = VT_I4:出力イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 ヒストグラムの均一化を行うことで画像のコントラストを上げます.
カラー画像の場合は, グレースケールに自動的に変換します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

GetMinMaxHistValue

構文 *object*.GetMinMaxHistValue(<Histogram>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

戻り値 <Min Value> = VT_R4:最小値
<Max Value> = VT_R4:最大値
<Min Index> = VT_I4:最小値の輝度値
<Max Index> = VT_I4:最大値の輝度値

説明 ヒストグラムの最大値, 最小値とそれらの輝度値を取得します.

HistAve

構文 *object*.HistAve(<Histogram>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

戻り値 <Average> = VT_R4:平均値

説明 <Histogram>の平均輝度値を計算します.

関連項目 CalcHistEx

AutoThreshPTile

構文 *object*.AutoThreshPTile(<Histogram>, <Rate>, <Forward>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム
<Rate> = VT_R8:面積の割合
<Forward> = VT_BOOL:検索方向

TRUE	前方検索
FALSE	後方検索

戻り値 <Threshold> = VT_I4:閾値

説明 P-タイル法による 2 値化の閾値の計算を行います.

関連項目 CalcHistEx, AutoThreshMode, AutoThreshDiscrim

AutoThreshMode

構文 *object*.AutoThreshMode(<Histogram>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

戻り値 <Threshold> = VT_I4:閾値

説明 モード法による 2 値化の閾値の計算を行います.

関連項目 CalcHistEx, AutoThreshPTile, AutoThreshDiscrim

AutoThreshDiscrim

構文 *object*.AutoThreshDiscrim(<Histogram>)

引数 <Histogram> = VT_R8|VT_ARRAY:ヒストグラム

戻り値 <Threshold> = VT_I4:閾値

説明 判別分析法による 2 値化の閾値の計算を行います.

関連項目 CalcHistEx, AutoThreshPTile, AutoThreshMode

4.2.9. マッチング

MatchTemplate

構文 *object*.MatchTemplate(<Input ID>, <Method>, <Retruen points>)

引数 <Input ID> = VT_I4:テンプレート画像のイメージ ID

<Method> = VT_I4:マッチング方法

(I は画像を, T はテンプレートを, R は結果をそれぞれ表す. 総和計算は, 以下のようにテンプレートと(または)画像領域に対して行われる. x'=0..w-1, y'=0..h-1)

0	CV_TM_SQDIFF	$R(x,y)=\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2$
1	CV_TM_SQDIFF_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
2	CV_TM_CCORR	$R(x,y)=\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')]$
3	CV_TM_CCORR_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')]}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
4	CV_TM_CCOEFF	$R(x,y)=\sum_{x',y'}[T'(x',y')\cdot I'(x+x',y+y')]$ このとき $T'(x',y')=T(x',y')-\frac{\sum_{x'',y''}T(x'',y'')}{(w\cdot h)}$ $I'(x+x',y+y')=I(x+x',y+y')-\frac{\sum_{x'',y''}I(x+x'',y+y'')}{(w\cdot h)}$

5	CV_TM_CCOEFF_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T'(x',y')\cdot I'(x+x',y+y')]}{\sqrt{\sum_{x',y'}T'(x',y')^2\cdot\sum_{x',y'}I'(x+x',y+y')^2}}$

<Max count> = VT_I4:検出数

戻り値

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>,<Value>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Value> = VT_R8:相関値

説明

現在の画像と<Input ID>画像のテンプレートマッチング処理を行います.
戻り値は, 処理結果の中から相関値の高いものを<RetCnt>で指定した数だけ返します. このとき, 検出画像の中心座標を返します.
[注意] バージョン 1.3.1 から引数及び戻り値の仕様が変更されています.

関連項目

MatchTemplate2, MatchShapesEx, MatchShapes2

高速化 Tips

・ 検索対象が特定の範囲内に収まる場合は, ROI を設定して検索範囲を限定する.

MatchShapesEx

構文

object.MatchShapesEx(<Input ID>, <Method>)

引数

<Input ID> = VT_I4:テンプレート画像のイメージ ID
<Method> = VT_I4:マッチング方法
ここで A は元画像, B はテンプレート画像を示す

0	CV_CONTOUR_MATCH_I1	$I_1(A,B)=\sum_{i=1}^{7}\left\lvert\frac{1}{m^A{}_i}-\frac{1}{m^B{}_i}\right\rvert$
1	CV_CONTOUR_MATCH_I2	$I_2(A,B)=\sum_{i=1}^{7}\left\lvert m^A{}_i-m^B{}_i\right\rvert$
2	CV_CONTOUR_MATCH_I3	$I_3(A,B)=\sum_{i=1}^{7}\frac{\left\lvert m^A{}_i-m^B{}_i\right\rvert}{\left\lvert m^A{}_i\right\rvert}$

ここで

$$m^{A}{}_{i} = \sin\left(h^{A}{}_{i}\right)\cdot\log\left(h^{A}{}_{i}\right)$$
$$m^{B}{}_{i} = \sin\left(h^{B}{}_{i}\right)\cdot\log\left(h^{B}{}_{i}\right)$$

$h^{A}{}_{i}$, $h^{B}{}_{i}$ は, A と B それぞれの Hu モーメント.

戻り値 <Similarity> = VT_R8:輪郭の一致度

説明 現在の画像と<Input ID>画像の形状比較処理を行います.

関連項目 MatchTemplate, MatchTemplate2, MatchShapes2, CalcBackProjectEx

MatchTemplate2

構文 *object*.MatchTemplate2(<Input ID>, <Method>, <Threshold>, <Start angle>, <End angle>, <Step angle>, <Down sizing>, <Max counts>, <Min distance>)

引数 <Input ID> = VT_I4:テンプレート画像のイメージ ID

<Method> = VT_I4:マッチング方法

(I は画像を, T はテンプレートを, R は結果をそれぞれ表す. 総和計算は, 以下のようにテンプレートと(または)画像領域に対して行われる. x'=0..w-1, y'=0..h-1)

0	CV_TM_SQDIFF	$R(x,y)=\sum_{x',y'}\left[T(x',y')-I(x+x',y+y')\right]^2$
1	CV_TM_SQDIFF_NORMED	$R(x,y)=\frac{\sum_{x',y'}\left[T(x',y')-I(x+x',y+y')\right]^2}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
2	CV_TM_CCORR	$R(x,y)=\sum_{x',y'}\left[T(x',y')\cdot I(x+x',y+y')\right]$
3	CV_TM_CCORR_NORMED	$R(x,y)=\frac{\sum_{x',y'}\left[T(x',y')\cdot I(x+x',y+y')\right]}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
4	CV_TM_CCOEFF	$R(x,y)=\sum_{x',y'}\left[T'(x',y')\cdot I'(x+x',y+y')\right]$ このとき

		$T'(x',y') = T(x',y') - \frac{\sum_{x'',y''} T(x'',y'')}{(w \cdot h)}$ $I'(x+x',y+y') = I(x+x',y+y') - \frac{\sum_{x'',y''} I(x+x'',y+y'')}{(w \cdot h)}$
5	CV_TM_CCOEFF_NORMED	$R(x,y) = \frac{\sum_{x',y'} [T'(x',y') \cdot I'(x+x',y+y')]}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$

<Threshold> = VT_R8:閾値
<Start angle> = VT_I4:検索開始角度(degree)
<End angle> = VT_I4:検索終了角度(degree)
<Step angle> = VT_I4:角度刻み(degree)
<Down sizing> = VT_I4:画面縮小回数
<Max counts> = VT_I4:検出数
<Min distance> = VT_I4:近傍判定距離(0:テンプレート画像サイズ)

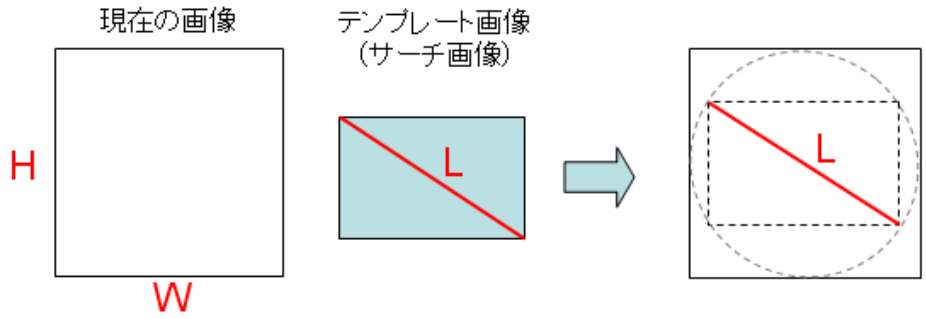
戻り値

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (**<Point1>**, **<Point2>**, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (**<X>**,**<Y>**,**<Angle>**,**<Value>**)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Angle> = VT_I4:回転角(degree)
<Value> = VT_R8:相関値

説明

現在の画像と<Input ID>画像の拡張テンプレートマッチング処理を行います.
指定した角度範囲でテンプレートマッチング処理を行い, 一致度の高い点を<RetCnt>で指定した数だけ返します. このとき, 一致度が<Threshold>以下であった点または一致度の高い点の近傍は検出しません. 各点の結果には, 点の座標, 回転角及び相関値を返します. このとき, 検出画像の中心座標を返します.
現在の画像またはテンプレート画像のどちらかがグレースケール画像の場合, もう一方の画像をグレースケールに変換した後テンプレートマッチング処理を実行します.
現在の画像の幅 W,高さ H,<Input ID>画像の対角線の長さ L とした場合は, 次式の条件を満たす必要があります.

in(W,H) >= L



[注意] バージョン 1.3.2 から引数及び戻り値の仕様が変更されています.
バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

関連項目 MatchTemplate, MatchShapesEx, MatchShapes2, CalcBackProjectEx

高速化 Tips
- 検索対象が特定の範囲内に収まる場合は, ROI を設定して検索範囲を限定する.
- 回転角度が一定の範囲に収まるときは, 開始角度と終了角度の範囲を適切に設定する.
- 円形のような回転した場合でも形状が変化しない場合は, 開始角度と終了角度を0に設定する.

MatchShapes2

構文 *object*.MatchShapes2(<Input ID>, <Method>, <Min scale>, <Similarity> , <Max counts>)

引数 <Input ID> = VT_I4:テンプレート画像のイメージ ID
<Method> = VT_I4:マッチング方法
ここで A は元画像, B はテンプレート画像を示す

0	CV_CONTOUR_MATCH_I1	$I_1(A,B)=\sum_{i=1}^{7}\left\lvert\frac{1}{m^A{}_i}-\frac{1}{m^B{}_i}\right\rvert$
1	CV_CONTOUR_MATCH_I2	$I_2(A,B)=\sum_{i=1}^{7}\left\lvert m^A{}_i-m^B{}_i\right\rvert$
2	CV_CONTOUR_MATCH_I3	$I_3(A,B)=\sum_{i=1}^{7}\frac{\left\lvert m^A{}_i-m^B{}_i\right\rvert}{\left\lvert m^A{}_i\right\rvert}$

ここで

$$m^A{}_i=\sin\left(h^A{}_i\right)\cdot\log\left(h^A{}_i\right)$$
$$m^B{}_i=\sin\left(h^B{}_i\right)\cdot\log\left(h^B{}_i\right)$$

$h^{A}{}_{i}$, $h^{B}{}_{i}$ は, A と B それぞれの Hu モーメント.

<Min scale> = VT_R8:最小倍率
<Similarity> = VT_R8:輪郭の一致度
<Max counts> = VT_I4:検出数

戻り値

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>,<Angle>,<Similarity>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Angle> = VT_I4:回転角(degree)
<Similarity> = VT_R8:輪郭の一致度

説明

現在の画像と<Input ID>画像の拡張形状比較処理を行います.
現在の画像から抽出した輪郭画像と<Input ID>画像の形状比較処理を行い, 最も一致度の高い輪郭に外接する最小楕円の中心座標, 回転角及び輪郭の一致度を返します.
検出した輪郭の一致度が<Similarity>以下のものを探索します.
現在の画像, <Input ID>画像ともに 2 値化画像でないと, 輪郭が正しく抽出することができません. また, 輪郭の抽出は 2 値化画像の白色の箇所に対して輪郭抽出を行います. 必ず検索対象が白抜きになっている 2 値化画像を使用してください.
<Input ID>画像から輪郭が複数抽出される場合, 正しく検出できない可能性があります. <Input ID>画像には, FindContours コマンドの結果が 1 になる画像を使用してください.
現在の画像から検出された輪郭画像の内, <Input ID>画像のサイズ×<Min scale>より小さいサイズは, 比較対象外とします.
[注意] 引数及び戻り値の仕様がバージョン 1.3.2 から変更されています.

関連項目

MatchTemplate, MatchTemplate2, MatchShapesEx, CalcBackProjectEx

HaarDetect

構文

object.HaarDetect(<Path>, <Scale>, <MinNeighbors>)

引数

<Path> = VT_BSTR:Haar ファイルのパス
<Scale> = VT_R8:スケール
<Min neighbors> = VT_I4:最小隣接数

戻り値 <Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>,<W>,<H>)
<X> = VT_I4:検出矩形の左上の X 座標
<Y> = VT_I4:検出矩形の左上の Y 座標
<W> = VT_I4:幅
<H> = VT_I4:高さ

説明 現在の画像に対して Haar マッチング処理を行い, 検出したオブジェクトのリストを返します.
検出されなかった場合は, Empty を返します. このとき, 関数の結果は S_FALSE になります.

4.2.10. CARD

CARDInit2

[V1.5.0 以降]

構文 *object*.CARDInit2(<Input ID>, <X>, <Y>)

引数 <Input ID> = VT_I4:テンプレート画像のイメージ ID
<X> = VT_I4:テンプレート画像内の検出点の X 座標
<Y> = VT_I4:テンプレート画像内の検出点の Y 座標

戻り値 <Count> = VT_I4:テンプレート画像の特徴点の数

説明 CARD の初期化としてテンプレート画像の登録を行います.
<X>, <Y>は CARDRun2 の実行結果として検出する点を指定します. -1 を指定した場合は, テンプレート画像の中央を自動的に設定します.
<Count>は, CARDRun2 で使用するテンプレート画像の特徴点の数を返します.
テンプレート画像を変更する場合は, 再度コマンドを実行してください.

CARDRun2

[V1.5.0 以降]

構文 *object*.CARDRun2(<Threshold>, <Count>)

引数 <Threshold> = VT_R8:閾値

戻り値 <Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>,<Angle>,<Scale>,<Value>)
<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標
<Angle> = VT_I4:回転角(degree)
<Scale> = VT_I4:スケール
<Value> = VT_R8:相関値

説明

CARD による画像探索を行います.
このコマンドを実行する前に CARDInit2 を実行する必要があります.
CARD による探索結果のうち, 一致度が<Threshold>で指定した値以上の点を検出点として返します.
検出点が見つからなかった場合, 戻り値は Empty になります.

CARDInitMulti

[V1.5.3 以降]

構文

object.CARDInitMulti(<Input ID>, <X>, <Y>, <Intermediate ID>)

引数

<Input ID> = VT_I4:テンプレート画像のイメージ ID
<X> = VT_R8:テンプレート画像内の検出点の X 座標
<Y> = VT_R8:テンプレート画像内の検出点の Y 座標
<Intermediate ID> = VT_R8:CARD 初期化の中間画像の出力先 ID(デフォルト:0)

戻り値

<Count> = VT_I4:テンプレート画像の特徴点の数

説明

CARD の初期化としてテンプレート画像の登録を行います.
テンプレート画像は画素数が 2000 ピクセル以上, 300 万ピクセル以下のものを指定してください.
<X>, <Y>は CARDRunMulti の実行結果として検出する点を指定します.
<Count>は, CARDRunMulti で使用するテンプレート画像の特徴点の数を返します.
テンプレート画像を変更する場合は, 再度コマンドを実行してください.
<Intermediate ID>に 0 を指定した場合は中間画像の生成を行いません. 高速で実行したい場合は中間画像の生成を行わないでください.
中間画像とは, テンプレート画像上に以下の 2 種類の点を描画した画像になります.

- 検出に利用される CARD 特徴点(青)
- 検出結果の補正に利用される輪郭点(赤)

図 4-1 CARDInitMulti コマンドの中間画像例

エラー

0x80100005 : テンプレート画像の画素数が大きすぎます. 画素数を 300 万ピクセル以下にしてください.

0x80101102 : テンプレート画像の画素数が小さすぎます. 画素数を 2000 ピクセル以上にしてください.

上記以外のエラーについては, 2.4 を参照してください.

CARDRunMulti

[V1.5.3 以降]

構文

object.CARDRunMulti(<Threshold>, <Count>, <Min distance>, <Intermediate ID>)

引数

<Threshold> = VT_R8:閾値

<Count> = VT_R8:検出数

<Min distance> = VT_I4:近傍判定距離(デフォルト:-1)

<Intermediate ID> = VT_R8:CARD 探索の中間画像の出力先 ID(デフォルト:0)

戻り値

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)

<Point1> = VT_R8|VT_ARRAY:検出位置 (<X>,<Y>,<Angle>,<Scale>,<Value>)

<X> = VT_R8:X 座標

<Y> = VT_R8:Y 座標

<Angle> = VT_R8:回転角(degree)

<Scale> = VT_R8:スケール

<Value> = VT_R8:相関値

説明

CARD による画像探索を行います.

このコマンドを実行する前に CARDInitMulti を実行する必要があります.

探索画像は 300 万ピクセル以下の画素数にしてください.

相関値が<Threshold>で指定した値より小さい場合, 探索結果から除外されます.

探索結果として検出された物体の中心点間距離が<Min distance>より小さい場合は, 相

関値の小さい方の検出点が探索結果から除外されます.
<Min distance>に-1 を指定した場合, テンプレート画像の幅, 高さのうち, 小さい方の大きさの半分が採用されます.（例:640×480 → <Min distance> = 240）
<Min distance>に 0 を指定した場合, 距離による除外処理は行いません.
結果は相関値で降順に格納されます.
検出点が見つからなかった場合, 戻り値は Empty になります.
<Intermediate ID>に 0 を指定した場合は中間画像の生成を行いません. 高速で実行したい場合は中間画像の生成を行わないでください.
中間画像とは, CARDInitMulti コマンドで指定したテンプレート画像－探索画像間における CARD 特徴点の対応を描画します.
同一検出物に対応する特徴点は同色の線で描画されます.
この中間画像で検出された結果は, CARDRunMulti コマンドの結果とは異なります. これは, 中間画像の結果を<Threshold>や<Min Distance>で除外することがあるためです.

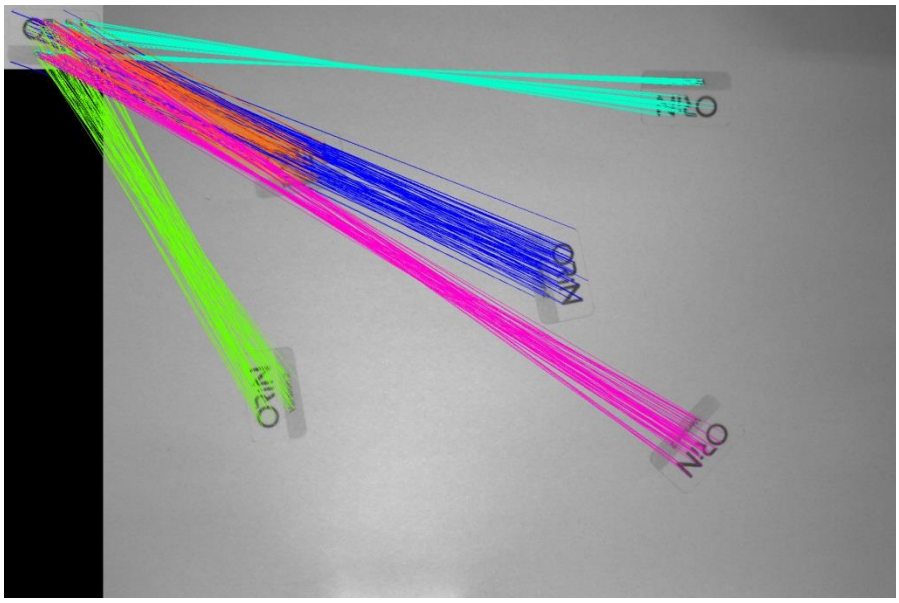


図 4-2 CARDRunMulti コマンドの中間画像例

エラー

0x80101001 : CARD の初期化がされていません.
CARDInitMulti コマンドを実行してください.
0x80100005 : 画像サイズが大きすぎます. 画面サイズを 300 万ピクセル以下にしてください.

上記以外のエラーについては, 2.4 を参照してください.

4.2.11. CAL

CalibrateCamera

構文 *object*.CalibrateCamera <Input ID>, <Count>, <Square count W>, <Square count H>, <Grid Size>, <Flag>, <Camera CAL ID>

引数

<Input ID> = VT_I4:チェスボード画像の先頭(基準画像)

<Count> = VT_I4:画像枚数

<Square count W> = VT_I4:横方向の升目数

<Square count H> = VT_I4:縦方向の升目数

<Grid Size> = VT_R8:グリッドサイズ

<Flag> = VT_I4:フラグ

1	CV_CALIB_CB_ADAPTIVE_THRESH	画像を 2 値化する際に, 固定の閾値を使うのではなく, (画像の平均輝度値から計算される)適応的な閾値を用いる.
2	CV_CALIB_CB_NORMALIZE_IMAGE	固定閾値処理または適応的閾値処理を行う前に, cvNormalizeHistを用いて画像を正規化する.
4	CV_CALIB_CB_FILTER_QUADS	輪郭の探索 段階で抽出される間違った四角形を無視するために, 追加基準(輪郭面積, 周囲長, 形は正方形など)を使用する.

<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

戻り値 なし

説明

カメラキャリブレーションを行います.

複数枚のチェスボード画像から内部パラメータを算出し, <Input ID>の画像から外部パラメータを計算します.

チェスボード画像の枚数は 5 枚以上を指定する必要があります. コーナー数とは升目の数です.

計算結果はデータベースに保存されます.

関連項目 FindChessBoardCorners, SetCamCalDat, GetCamCalDat, SetCamCalExtDat,

GetCamCalExtDat, ModifyCamCalExtDat, GetPosFromCam, GetCamPos, GetRobPosFromCam, GetCamPosFromRob, Undistort2

CalibrateRobot

構文 *object*.CalibrateRobot <Robot CAL ID>, <Points>

引数

<Robot CAL ID> = VT_I4:ロボットキャリブレーション ID

<Points> = VT_VARIANT|VT_ARRAY:ワールド-ロボット対応点リスト

(<Point1>, <Point2>, …)

<Pointn> = VT_VARIANT |VT_ARRAY:ワールド-ロボット対応点

(<World Point>,<Robot Point>)

<World Point> = VT_R8 |VT_ARRAY:ワールド座標 (<X>,<Y>,<Z>)

<X> = VT_R8:X 座標

<Y> = VT_R8:Y 座標

<Z> = VT_R8:Z 座標

<Robot Point> = VT_R8 |VT_ARRAY:ロボット座標 (<X>,<Y>,<Z>)

<X> = VT_R8:X 座標

<Y> = VT_R8:Y 座標

<Z> = VT_R8:Z 座標

戻り値 なし

説明

ロボットキャリブレーションを行います.

ワールド-ロボット座標の対応点を任意の個数だけ指定してキャリブレーションデータを算出します.

関連項目 SetRobCalDat, GetRobCalDat, GetPosFromRob, GetRobPos, GetRobPosFromCam, GetCamPosFromRob

FindChessBoardCorners

構文 *object*.FindChessBoardCorners <Square Count W>, <Square Count H>, <Flag>

引数

<Square Count W> = VT_I4:横方向の升目数

<Square Count H> = VT_I4:縦方向の升目数

<Flag> = VT_I4:フラグ

1	CV_CALIB_CB_ADAPTIVE_THRESH	画像を 2 値化する際に, 固定の

		閾値を使うのではなく,(画像の平均輝度値から計算される)適応的な閾値を用いる.
2	CV_CALIB_CB_NORMALIZE_IMAGE	固定閾値処理または適応的閾値処理を行う前に,cvNormalizeHistを用いて画像を正規化する.
4	CV_CALIB_CB_FILTER_QUADS	輪郭の探索 段階で抽出される間違った四角形を無視するために,追加基準(輪郭面積,周囲長,形は正方形など)を使用する.

戻り値

<Pattern was found> = VT_BOOL:検出結果 (0:失敗, 0 以外:成功)
<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標

説明

画面からチェスボードのコーナーを検出します.

関連項目

CalibrateCamera, DrawChessBoardCorners

DrawChessBoardCorners

構文

object.DrawChessBoardCorners <Output ID>, <Square Count W>, <Square Count H>, < Pattern was found > ,<Points>

引数

<Output ID> = VT_I4:出力イメージ ID
<Square Count W> = VT_I4:横方向の升目数
<Square Count H> = VT_I4:縦方向の升目数
<Pattern was found> = VT_BOOL:FindChessBoardCorners の検出結果

0	失敗
0 以外	成功

<Points> = VT_VARIANT|VT_ARRAY:検出位置リスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:検出位置 (<X>,<Y>)
<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

戻り値 <Image> = VT_UI1|VT_ARRAY:描画画像

説明 チェスボードのコーナーの検出結果を表示します.
コーナーを完全に検出した場合は, 色付けされた各コーナーを線分で接続して表示します. 完全な検出ができなかった場合は, 検出したコーナーを赤色の円で表示します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 FindChessBoardCorners

DrawXYAxes

構文 *object*.DrawChessBoardCorners <Output ID>, <Camera CAL ID>, <R>, <G>, <B>

引数 <Output ID> = VT_I4:出力イメージ ID
<Camera CAL ID> = VT_I4:カメラキャリブレーション番号
<R> = VT_I4:色の R 成分
<G> = VT_I4:色の G 成分
<B> = VT_I4:色の B 成分

戻り値 <Image> = VT_UI1|VT_ARRAY:描画画像

説明 <Camera CAL ID>で指定したカメラのキャリブレーションデータの X 軸及び Y 軸を<Output ID>で指定したイメージメモリに描画します.
グレースケール画像の場合, <B>の値で描画します.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.

関連項目 CalibrateCamera

SetCamCalDat

構文 *object*.SetCamCalDat <Intrinsic matrix>, <Distortion coeffs>, <Extrinsic

matrix>, <Camera CAL ID>

引数

<Intrinsic matrix> = VT_R8|VT_ARRAY:内部パラメータ(<fx>,<fy>,<dx>,<cy>)
<fx> = VT_R8:焦点距離 X
<fy> = VT_R8:焦点距離 Y
<cx> = VT_R8:中心座標 X
<cy> = VT_R8:中心座標 Y
<Distortion coeffs> = VT_R8|VT_ARRAY:歪み係数(<k1>,<k2>,<p1>,<p2>)
<k1> = VT_R8:半径方向の歪み係数
<k2> = VT_R8:半径方向の歪み係数
<p1> = VT_R8:円周方向の歪み係数
<p2> = VT_R8:円周方向の歪み係数
<Extrinsic matrix> = VT_R8|VT_ARRAY:外部パラメータ
(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>,<dx>,<dy>,<dz>)
<r11> = VT_R8:回転ベクトル
<r21> = VT_R8:
<r31> = VT_R8:
<r12> = VT_R8:
<r22> = VT_R8:
<r32> = VT_R8:
<r13> = VT_R8:
<r23> = VT_R8:
<r33> = VT_R8:
<dx> = VT_R8:並進移動ベクトル
<dy> = VT_R8:
<dz> = VT_R8:
<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} r11 & r12 & r13 & dx \\ r21 & r22 & r23 & dy \\ r31 & r32 & r33 & dz \end{pmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

戻り値

なし

説明

カメラの内部, 外部パラメータ及び歪み係数をデータベースに設定します.

関連項目

CalibrateCamera, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, ModifyCamCalExtDat, GetPosFromCam, GetCamPos, GetRobPosFromCam,

GetCamPosFromRob, Undistort2

GetCamCalDat

構文 *object*.GetCamCalDat(<Camera CAL ID>)

引数 <Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

戻り値 VT_VARIANT|VT_ARRAY:(<Intrinsic matrix>, <Distortion coeffs>, <Extrinsic matrix>)

<Intrinsic matrix> = VT_R8|VT_ARRAY:内部パラメータ(<fx>,<fy>,<cx>,<cy>)

<fx> = VT_R8:焦点距離 X

<fy> = VT_R8:焦点距離 Y

<cx> = VT_R8:中心座標 X

<cy> = VT_R8:中心座標 Y

<Distortion coeffs> = VT_R8|VT_ARRAY:歪み係数(<k1>,<k2>,<p1>,<p2>)

<k1> = VT_R8:半径方向の歪み係数

<k2> = VT_R8:半径方向の歪み係数

<p1> = VT_R8:円周方向の歪み係数

<p2> = VT_R8:円周方向の歪み係数

<Extrinsic matrix> = VT_R8|VT_ARRAY:外部パラメータ

(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>,<dx>,<dy>,<dz>)

<r11> = VT_R8:回転ベクトル

<r21> = VT_R8:

<r31> = VT_R8:

<r12> = VT_R8:

<r22> = VT_R8:

<r32> = VT_R8:

<r13> = VT_R8:

<r23> = VT_R8:

<r33> = VT_R8:

<dx> = VT_R8:並進移動ベクトル

<dy> = VT_R8:

<dz> = VT_R8:

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} r11 & r21 & r13 & dx \\ r21 & r22 & r23 & dy \\ r31 & r32 & r33 & dz \end{pmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

説明 カメラの内部, 外部パラメータ及び歪み係数をデータベースから取得します.

関連項目 CalibrateCamera, SetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, ModifyCamCalExtDat, GetPosFromCam, GetCamPos, GetRobPosFromCam, GetCamPosFromRob, Undistort2

SetCamCalExtDat

構文 *object*.SetCamCalExtData <Extrinsic matrix>, <Camera CAL ID>

引数 <Extrinsic matrix> = VT_R8|VT_ARRAY:外部パラメータ
(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>,<dx>,<dy>,<dz>)
<r11> = VT_R8:回転ベクトル
<r21> = VT_R8:
<r31> = VT_R8:
<r12> = VT_R8:
<r22> = VT_R8:
<r32> = VT_R8:
<r13> = VT_R8:
<r23> = VT_R8:
<r33> = VT_R8:
<dx> = VT_R8:並進移動ベクトル
<dy> = VT_R8:
<dz> = VT_R8:
<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} r11 & r12 & r13 & dx \\ r21 & r22 & r23 & dy \\ r31 & r32 & r33 & dz \end{pmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

戻り値 なし

説明 カメラキャリブレーションの外部パラメータを<Camera CAL ID>で指定したデータベースに

設定します.

関連項目 CalibrateCamera, SetCamCalDat, GetCamCalDat, GetCamCalExtDat, ModifyCamCalExtDat, GetPosFromCam, GetCamPos, GetRobPosFromCam, GetCamPosFromRob, Undistort2

GetCamCalExtDat

構文 *object*.GetCamCalExtDat(<Inverse>, <Camera CAL ID>)

引数 <Inverse> = VT_BOOL :逆行列フラグ
<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

戻り値 <Extrinsic matrix> = VT_R8|VT_ARRAY:外部パラメータ
(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>,<dx>,<dy>,<dz>)
<r11> = VT_R8:回転ベクトル
<r21> = VT_R8:
<r31> = VT_R8:
<r12> = VT_R8:
<r22> = VT_R8:
<r32> = VT_R8:
<r13> = VT_R8:
<r23> = VT_R8:
<r33> = VT_R8:
<dx> = VT_R8:並進移動ベクトル
<dy> = VT_R8:
<dz> = VT_R8:

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} r11 & r12 & r13 & dx \\ r21 & r22 & r23 & dy \\ r31 & r32 & r33 & dz \end{pmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

説明 <Camera CAL ID>で指定したカメラキャリブレーションの外部パラメータをデータベースから取得します.
<Inverse>が TRUE のとき, 外部パラメータの逆行列を取得します.

関連項目 CalibrateCamera, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, ModifyCamCalExtDat, GetPosFromCam, GetCamPos, GetRobPosFromCam,

GetCamPosFromRob, Undistort2

ModifyCamCalExtDat

構文 *object*.ModifyCamCalExtDat <Input ID>, <Square count W>, <Square count H>, <Grid Size>, <Flag>, <Camera CAL ID>

引数

<Input ID> = VT_I4:チェスボード画像
<Square count W> = VT_I4:横方向の升目数
<Square count H> = VT_I4:縦方向の升目数
<Grid Size> = VT_R8:グリッドサイズ
<Flag> = VT_I4:フラグ

1	CV_CALIB_CB_ADAPTIVE_THRESH	画像を 2 値化する際に，固定の閾値を使うのではなく，(画像の平均輝度値から計算される)適応的な閾値を用いる.
2	CV_CALIB_CB_NORMALIZE_IMAGE	固定閾値処理または適応的閾値処理を行う前に，cvNormalizeHistを用いて画像を正規化する.
4	CV_CALIB_CB_FILTER_QUADS	輪郭の探索 段階で抽出される間違った四角形を無視するために，追加基準(輪郭面積，周囲長，形は正方形など)を使用する.

<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

戻り値 なし

説明 指定した画像から外部パラメータを更新します.

関連項目 CalibrateCamera, FindChessBoardCorners, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, GetPosFromCam, GetCamPos, GetRobPosFromCam, GetCamPosFromRob, Undistort2

SetRobCalDat

構文 *object*.SetRobCalDat <Robot CAL ID>, <r11>, <r21>, <r31>, <r12>, <r22>, <r32>, <r13>, <r23>, <r33>, <dx>, <dy>, <dz>

引数 <Robot CAL ID> = VT_I4:ロボットキャリブレーションデータの格納先 ID
(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>,<dx>,<dy>,<dz>) : 同次変換行列
<r11> = VT_R8:回転ベクトル
<r21> = VT_R8:
<r31> = VT_R8:
<r12> = VT_R8:
<r22> = VT_R8:
<r32> = VT_R8:
<r13> = VT_R8:
<r23> = VT_R8:
<r33> = VT_R8:
<dx> = VT_R8:並進移動ベクトル
<dy> = VT_R8:
<dz> = VT_R8:

$$\begin{pmatrix} Xr \\ Yr \\ Zr \\ 1 \end{pmatrix} = \begin{pmatrix} r11 & r12 & r13 & dx \\ r21 & r22 & r23 & dy \\ r31 & r32 & r33 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} Xw \\ Yw \\ Zw \\ 1 \end{pmatrix}$$

戻り値 なし

説明 ロボットのキャリブレーションデータをデータベースに設定します.

関連項目 CalibrateRobot, GetRobCalDat, GetPosFromRob, GetRobPos, GetRobPosFromCam, GetCamPosFromRob

GetRobCalDat

構文 *object*.GetRobCalDat(<Robot CAL ID>, <Inverse>)

引数 <Robot CAL ID> = VT_I4:ロボットキャリブレーション ID
<Inverse> = VT_BOOL :逆行列フラグ

戻り値 <Matrix> = VT_R8|VT_ARRAY:同次変換行列
(<r11>,<r21>,<r31>,<r12>,<r22>,<r32>,<r13>,<r23>,<r33>,<dx>,<dy>,<dz>)
<r11> = VT_R8:回転ベクトル
<r21> = VT_R8:
<r31> = VT_R8:
<r12> = VT_R8:
<r22> = VT_R8:
<r32> = VT_R8:
<r13> = VT_R8:
<r23> = VT_R8:
<r33> = VT_R8:
<dx> = VT_R8:並進移動ベクトル
<dy> = VT_R8:
<dz> = VT_R8:

$$\begin{pmatrix} Xr \\ Yr \\ Zr \\ 1 \end{pmatrix} = \begin{pmatrix} r11 & r12 & r13 & dx \\ r21 & r22 & r23 & dy \\ r31 & r32 & r33 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} Xw \\ Yw \\ Zw \\ 1 \end{pmatrix}$$

説明 ロボットのキャリブレーションデータをデータベースから取得します.
<Inverse>が TRUE のとき, 外部パラメータの逆行列を取得します.

関連項目 CalibrateRobot, SetRobCalDat, GetPosFromRob, GetRobPos, GetRobPosFromCam, GetCamPosFromRob

SetCamDescription

構文 *object*.SetCamDescription <Camera CAL ID>, <Description>

引数 <Camera CAL ID> = VT_I4:カメラキャリブレーション ID
<Description> = VT_BSTR :説明内容

戻り値 なし

説明 カメラのキャリブレーションの説明をデータベースに設定します.

関連項目 CalibrateCamera, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, GetCamDescription, GetPosFromCam, GetCamPos, GetRobPosFromCam, GetCamPosFromRob, Undistort2

GetCamDescription

構文 *object*.GetCamDescription(<Camera CAL ID>)

引数 <Camera CAL ID> = VT_I4:カメラキャリブレーション ID

戻り値 <Description> = VT_BSTR :説明内容

説明 カメラのキャリブレーションの説明をデータベースから取得します.

関連項目 CalibrateCamera, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, SetCamDescription, GetPosFromCam, GetCamPos, GetRobPosFromCam, GetCamPosFromRob, Undistort2

SetRobDescription

構文 *object*.SetRobDescription <Robot CAL ID>, <Description>

引数 <Robot CAL ID> = VT_I4:ロボットキャリブレーション ID
<Description> = VT_BSTR :説明内容

戻り値 なし

説明 ロボットのキャリブレーションの説明をデータベースに設定します.

関連項目 CalibrateRobot, SetRobCalDat, GetRobCalDat, GetRobDescription, GetPosFromRob, GetRobPos, GetRobPosFromCam, GetCamPosFromRob

GetRobDescription

構文 *object*.GetRobDescription(<Robot CAL ID>)

引数 <Robot CAL ID> = VT_I4:ロボットキャリブレーション ID

戻り値 <Description> = VT_BSTR :説明内容

説明 ロボットのキャリブレーションの説明をデータベースから取得します.

関連項目 CalibrateRobot, SetRobCalDat, GetRobCalDat, SetRobDescription, GetPosFromRob, GetRobPos, GetRobPosFromCam, GetCamPosFromRob

GetPosFromCam

構文 *object*.GetPosFromCam(<Xc>, <Yc> [, <ZOffset>, <Camera CAL ID>, <Undistort>])

引数
<Xc> = VT_I4:カメラ座標の X
<Yc> = VT_I4:カメラ座標の Y
<ZOffset> =VT_I4:ワールド座標の Z(デフォルト:0)
<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)
<Undistort> = VT_BOOL:歪み補正(デフォルト:False)

戻り値
<Xw> = VT_R8:ワールド座標の X
<Yw> = VT_R8:ワールド座標の Y
<Zw> = VT_R8:ワールド座標の Z (引数の ZOffset の値)

説明
カメラ座標をワールド座標 Z=<ZOffset>の面の座標に変換します.
変換には<Camera CAL ID>で指定したカメラのキャリブレーションデータを使用します.
<Camera CAL ID>に 0 を指定したときは, 以下の値を使用します.

イメージ ID	使用されるカメラ ID
カメラ(1～10)	イメージ ID
カメラ以外(10～)	1

<Undistort>が TRUE のときは, 歪みのある画像からワールド座標への変換を行います.
FALSE のときは, 歪み補正済みの画像からワールド座標への変換を行います.

関連項目 CalibrateCamera, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, ModifyCamCalExtDat, GetCamPos, GetRobPosFromCam, Undistort2

GetCamPos

構文 *object*.GetCamPos(<Xw>, <Yw>, <Zw>[, <Camera CAL ID>])

引数
<Xw> = VT_R8:ワールド座標の X
<Yw> = VT_R8:ワールド座標の Y
<Zw> = VT_R8:ワールド座標の Z

<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)

戻り値

<Xc> = VT_I4:カメラ座標の X
<Yc> = VT_I4:カメラ座標の Y

説明

ワールド座標をカメラ座標に変換します.
変換には<Camera CAL ID>で指定したカメラのキャリブレーションデータを使用します.
カメラ ID に 0 を指定したときは, 以下の値を使用します.

イメージ ID	使用されるカメラ ID
カメラ(1～10)	イメージ ID
カメラ以外(10～)	1

関連項目

CalibrateCamera, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, ModifyCamCalExtDat, GetPosFromCam, GetCamPosFromRob, Undistort2

GetPosFromRob

構文

object.GetPosFromRob(<Robot CAL ID>, <Xr>, <Yr>, <Zr>)

引数

<Robot CAL ID> = VT_I4:ロボットキャリブレーション ID
<Xr> = VT_R8:ロボット座標の X
<Yr> = VT_R8:ロボット座標の Y
<Zr> = VT_R8:ロボット座標の Z

戻り値

<Xw> = VT_R8:ワールド座標の X
<Yw> = VT_R8:ワールド座標の Y
<Zw> = VT_R8:ワールド座標の Z

説明

ロボット座標をワールド座標に変換します.
変換には<Robot CAL ID>で指定したキャリブレーションデータを使用します.

関連項目

CalibrateRobot, GetRobPos, GetRobCalDat, SetRobCalDat, GetCamPosFromRob

GetRobPos

構文

object.GetRobPos(<Robot CAL ID>, <Xw>, <Yw>, <Zw>)

引数

<Robot CAL ID> = VT_I4:ロボットキャリブレーション ID
<Xw> = VT_R8:ワールド座標の X

<Yw> = VT_R8:ワールド座標の Y
<Zw> = VT_R8:ワールド座標の Z

戻り値

<Xr> = VT_R8:ロボット座標の X
<Yr> = VT_R8:ロボット座標の Y
<Zr> = VT_R8:ロボット座標の Z

説明

ワールド座標系の座標をロボット座標に変換します.
変換には<Robot CAL ID>で指定したキャリブレーションデータを使用します.

関連項目

CalibrateRobot, GetPosFromRob, GetRobCalDat, SetRobCalDat, GetRobPosFromCam

GetRobPosFromCam

構文

object.GetRobPosFromCam(<Xc>, <Yc> [, <ZOffset>, <Camera CAL ID>, <Robot CAL ID>, <Undistort>])

引数

<Xc> = VT_R8:カメラ座標の X
<Yc> = VT_R8:カメラ座標の Y
<ZOffset> =VT_I4:ワールド座標の Z(デフォルト:0)
<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)
<Robot CAL ID> = VT_I4:ロボットキャリブレーション ID(デフォルト:1)
<Undistort> = VT_BOOL:歪み補正(デフォルト:False)

戻り値

<Xw> = VT_R8:ロボット座標の X
<Yw> = VT_R8:ロボット座標の Y
<Zw> = VT_R8:ロボット座標の Z

説明

カメラ座標をロボット座標に変換します.
このコマンドは, 以下の順番に座標系を変換していきます.
　カメラ座標→ワールド座標→ロボット座標
カメラ座標からワールド座標に変換するときは Z=<ZOffset>の面の座標に変換します.
変換には<CameraID>, <RobotID>で指定したキャリブレーションデータを使用します.
カメラ ID に 0 を指定したときは, 以下の値を使用します.

イメージ ID	使用されるカメラ ID
カメラ(1～10)	イメージ ID
カメラ以外(10～)	1

<Undistort>が TRUE のときは, 歪みのある画像からワールド座標への変換を行います.

FALSE のときは，歪み補正済みの画像からワールド座標への変換を行います．

関連項目 CalibrateCamera, CalibrateRobot, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, ModifyCamCalExtDat, SetRobCalDat, GetRobCalDat, GetPosFromCam, GetCamPos, GetPosFromRob, GetRobPos, GetCamPosFromRob, Undistort2

GetCamPosFromRob

構文 *object*.GetCamPosFromRob(<Xw>, <Yw>, <Zw>[, <Camera CAL ID>, <Robot CAL ID>])

引数
<Xw> = VT_R8:ロボット座標の X
<Yw> = VT_R8:ロボット座標の Y
<Zw> = VT_R8:ロボット座標の Z
<Camera CAL ID> = VT_I4:カメラキャリブレーション ID(デフォルト:0)
<Robot CAL ID> = VT_I4:ロボットキャリブレーション ID(デフォルト:1)

戻り値
<Xc> = VT_R8:カメラ座標の X
<Yc> = VT_R8:カメラ座標の Y

説明
ロボット座標をカメラ座標に変換します．
このコマンドは，以下の順番に座標系を変換していきます．
ロボット座標→ワールド座標→カメラ座標
変換には<Camera CAL ID>，<Robot CAL ID>で指定したキャリブレーションデータを使用します．
カメラ ID に 0 を指定したときは，以下の値を使用します．

イメージ ID	使用されるカメラ ID
カメラ(1～10)	イメージ ID
カメラ以外(10～)	1

関連項目 CalibrateCamera, CalibrateRobot, SetCamCalDat, GetCamCalDat, SetCamCalExtDat, GetCamCalExtDat, ModifyCamCalExtDat, SetRobCalDat, GetRobCalDat, GetPosFromCam, GetCamPos, GetPosFromRob, GetRobPos, GetRobPosFromCam, Undistort2

Undistort2

構文 *object*.Undistort2 <Output ID>, <Camera CAL ID>

引数 <Output ID> = VT_I4:出力イメージ ID
<Camera CAL ID> = VT_I4:カメラ ID(デフォルト:0)

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 歪み補正を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
歪み補正には, 指定したカメラ ID のパラメータを使用します.
カメラ ID に 0 を指定したときは, 以下の値を使用します.

イメージ ID	使用されるカメラ ID
カメラ(1～10)	イメージ ID
カメラ以外(10～)	1

変換画像データは, Windows 標準のビットマップファイル形式で出力されます. カラー画像は 24bit ビットマップ, グレースケール画像は 8bit ビットマップで出力します.
詳細については, OpenCV リファレンスの Undistort2 の項目を参照してください.
[注意] 引数及び戻り値の仕様がバージョン 1.3.2 から変更されています.

関連項目 CalibrateCamera, SetCamCalDat, GetCamCalDat, GetPosFromCam, GetRobPosFromCam

4.2.12. 拡張カメラ

ExtExecSoftTrigger [V1.5.1 以降]

構文 *object*.ExtExecSoftTrigger

引数 なし

戻り値 なし

説明 カメラのソフトウェアトリガを実行します.
このコマンドは, 拡張カメラのみ使用することができます.
このコマンドは拡張カメラに対応した ORiN2 プロバイダの CaoController::Execute()の“OCV_ExecSoftTrigger”コマンドを実行します.

ExtRefreshImage

[V1.5.1 以降]

構文 *object*.ExtRefreshImage

引数 なし

戻り値 なし

説明 拡張カメラの画像を更新します.
このコマンドは, 拡張カメラのみ使用することができます.
このコマンドは拡張カメラに対応した ORiN2 プロバイダの CaoController::Execute()の“OCV_GetImage”コマンドを実行し, 取得画像で内部バッファを更新します.

ExtInvoke

[V1.5.1 以降]

構文 *object*.ExtInvoke(<Command>, <Parameter>)

引数 <Command> = VT_BSTR:コマンド名
<Parameter> = VT_VARIANT:パラメータ

戻り値 <Result> = VT_VARIANT:結果

説明 拡張カメラのコマンドを実行します.
このコマンドは, 拡張カメラのみ使用することができます.
このコマンドは拡張カメラに対応した ORiN2 プロバイダの CaoController::Execute()を実行します.
<Command>で指定可能なコマンド名及び<Parameter>及び<Result>の内容については拡張カメラに対応した ORiN2 プロバイダのユーザーズガイドを参照してください.

4.2.13. その他

GoodFeaturesToTrackEx

構文 *object*.GoodFeaturesToTrackEx(<Max count>, <Quality>, <Distance>, <Block size>)

引数 <Max count> = VT_I4:最大検知数
<Quality> = VT_R8:クオリティー

<Distance> = VT_I4:最小距離
<Block size> = VT_I4:平均化ブロックのサイズ

戻り値

<Points> = VT_VARIANT|VT_ARRAY:角検知ポイントリスト
(<Point1>,<Point2>,…)
<Pointn> = VT_I4|VT_ARRAY:座標配列 (<X>,<Y>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標

説明

画像中から大きな固有値を持つコーナーを検出します.
カラー画像の場合は, グレースケールに自動的に変換します.

関連項目

FindCornerSubPixEx

FindCornerSubPixEx

構文

object.FindCornerSubPixEx(<Points>, <Win X>, <Win Y>, <Zero X>, <Zero Y>, <Term type>, <Max iteration>, <Epsilon>)

引数

<Points> = VT_VARIANT|VT_ARRAY:角検知ポイントリスト
(<Point1>,<Point2>,…)
<Pointn> = VT_I4|VT_ARRAY:座標配列 (<X>,<Y>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<Win X> = VT_I4:検索領域の X 方向サイズ
<Win Y> = VT_I4:検索領域の Y 方向サイズ
<Zero X> = VT_I4:除外領域の X 方向サイズ
<Zero Y> = VT_I4:除外領域の Y 方向サイズ
<Term type> = VT_I4:反復終了条件タイプ

1	反復使用
2	達成精度使用

<Max iteration> = VT_I4:最大反復回数
<Epsion> = VT_R8:達成精度

戻り値

<Points> = VT_VARIANT|VT_ARRAY:角検知ポイントリスト
(<Point1>,<Point2>,…)
<Pointn> = VT_I4|VT_ARRAY:座標配列 (<X>,<Y>)
<X> = VT_I4:X 座標

<Y> = VT_I4:Y 座標

説明 角検知結果で得られた角位置を高精度化します.
検索領域<Win X>, <Win Y>と除外領域<Zero X>, <Zero Y>のサイズは検索領域の半分のサイズを指定しなければなりません.

関連項目 GoodFeaturesToTrackEx

MomentsEx

構文 *object*.MomentsEx(<Contour ID>)

引数 <Contour ID> = VT_I4:輪郭番号

-1	画面全体
-1 以外	指定した輪郭番号

戻り値 <Moments> = VT_VARIANT|VT_ARRAY:モーメント
(<SpatialMoments>,<CentralMoments>,<inv_sqrt_m00>)
<Spatial moments> = VT_R8|VT_ARRAY:空間モーメント
<M00> = VT_R8:
<M10> = VT_R8:
<M01> = VT_R8:
<M20> = VT_R8:
<M11> = VT_R8:
<M02> = VT_R8:
<M30> = VT_R8:
<M21> = VT_R8:
<M12> = VT_R8:
<M03> = VT_R8:
<Central moments> = VT_R8|VT_ARRAY:中心モーメント
<M20> = VT_R8:
<M11> = VT_R8:
<M02> = VT_R8:
<M30> = VT_R8:
<M21> = VT_R8:
<M12> = VT_R8:
<M03> = VT_R8:
<inv_sqrt_m00> = VT_R8:1/sqrt(M00)

説明 3次までの空間モーメントあるいは中心モーメントを計算を行います.
<Contour ID>に-1 以外を指定するときは, 事前に FindContoursEx コマンドを実行する必要があります.
カラー画像の場合は, グレースケールに自動的に変換します.

エラー 0x80101001 : 輪郭検出がされていません. “FindContoursEx”を実行してください.
上記以外のエラーについては, 2.4 を参照してください.

MeasureInfo

構文 *object*.MeasureInfo(<Moments>)

引数 <Moments> = VT_R4|VT_ARRAY:モーメント
<M00> = VT_R8:
<M10> = VT_R8:
<M01> = VT_R8:
<M20> = VT_R8:
<M11> = VT_R8:
<M02> = VT_I4:

戻り値 <Area> = VT_R8:面積
<Center of gravity X> = VT_R8:重心の X 座標
<Center of gravity Y> = VT_R8:重心の Y 座標
<Principal axis angle> = VT_R8:主軸角

説明 面積, 重量, 主軸角の計算を行います.
<Moments>には, 上記の書式以外にも MomentsEx の結果を入れることも可能です.

HoughLines

構文 *object*.HoughLines(<Method>, <Rho>, <Theta>, <Threshold>, <Parameter1>, <Parameter2>)

引数 <Method> = VT_I4:Hough 変換の種類

0	CV_HOUGH_STANDARD	標準的ハフ変換. 全ての線分は 2 つの浮動小数点値 (ρ , θ)で表される. ここ

		で ρ は点(0,0) から線分までの距離，θ は x 軸と線分の法線が成す角度.
1	CV_HOUGH_PROBABILISTIC	確率的ハフ変換（画像に長い線が少ない場合に有効）. 全ての線を返すのではなく，線分を返す. 全ての線分は始点と終点で表される.
2	CV_HOUGH_MULTI_SCALE	マルチスケール型の古典的ハフ変換. 線は CV_HOUGH_STANDARD と同様の方法でエンコードされる.

<Rho> = VT_R8: ρ
<Theta> = VT_R8: θ
<Threshold> = VT_I4:閾値
<Parameter1> = VT_R8:パラメータ 1
<Parameter2> = VT_R8:パラメータ 2

戻り値

<Lines> = VT_VARIANT|VT_ARRAY:直線検知結果リスト (<Line1>, <Line2>, …)
<Linen> = VT_I4|ARRAY :直線座標配列
(<Start X>,<Start Y>,<End X>,<End Y>)
<Start X> = VT_I4:開始 X 座標
<Start Y> = VT_I4:開始 Y 座標
<End X> = VT_I4:終了 X 座標
<End Y> = VT_I4:終了 Y 座標

説明

Hough 変換を用いて 2 値画像から直線を検出します.

関連項目

HoughCircles

HoughCircles

構文

object.HoughCircles(<dp>, <Min distance>, <Canny threshold>, <Center threshold>, <Min radius>, <Max radius>)

引数

<dp> = VT_R8:計算時の解像度
<Min distance> = VT_R8:中心座標間の最小間隔
<Canny threshold> = VT_R8:Canny で使用する高い方の閾値
<Center threshold> = VT_R8:中心検出計算時の閾値
<Min radius> = VT_I4:最小半径

<Max radius> = VT_I4:最大半径

戻り値 <Circles> = VT_VARIANT|VT_ARRAY:円検知結果リスト
(<Circle1>, <Circle2>, …)
<Circlen> = VT_R4|VT_ARRAY:円 (<X>,<Y>,<Radius>)
<X> = VT_R4:中心 X 座標
<Y> = VT_R4:中心 Y 座標
<Radius> = VT_R4:半径

説明 Hough 変換を用いて円を検出する

関連項目 HoughLines

DFTEx

構文 *object*.DFTEx <Output ID>, <Output ID(R)>, <Output ID(I)>

引数 <Output ID> = VT_I4:出力イメージ ID
<Output ID(R)> = VT_I4:出力実数部イメージ ID
<Output ID(I)> = VT_I4:出力虚数部イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 離散フーリエ変換を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
<Output ID(R)>と<Output ID(I)>は, データベース領域を使用できません. 一時変数領域を使用してください.
カラー画像の場合は, グレースケールに自動的に変換します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

関連項目 IDFT

IDFT

構文 *object*.IDFT <Output ID>, <Input ID(R)>, <Input ID(I)>

引数 <Output ID> = VT_I4:出力イメージ ID
<Input ID(R)> = VT_I4:入力実数部イメージ ID

<Input ID(I)> = VT_I4:入力虚数部イメージ ID

戻り値 <Image> = VT_UI1|VT_ARRAY:変換画像

説明 離散フーリエ逆変換を行います.
出力イメージ ID が 0 のときは返り値に変換画像を出力します. 出力イメージ ID が 0 以外のときは, 指定した番号に変換画像を出力し, 返り値に Empty を返します.
変換画像データは, Windows 標準の 8bit ビットマップファイル形式で出力されます.

関連項目 DFTEx

OpticalFlowEx

構文 *object*.OpticalFlowEx(<Input ID>, <X size>, <Y size>)

引数 <Input ID> = VT_I4:比較画像のイメージ ID
<X size> = VT_I4:X 軸計測単位
<Y size> = VT_I4:Y 軸計測単位

戻り値 <Points> = VT_VARIANT|VT_ARRAY:ポイントリスト (<Point1>, <Point2>, …)
<Pointn> = VT_I4|VT_ARRAY:位置と変化量 (<X>,<Y>,<dX>,<dY>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標
<dX> = VT_I4:X 方向変化量
<dY> = VT_I4:Y 方向変化量

説明 現在の画像と<Input ID>画像のオプティカルフロー処理を行い, ポイントリストを返します.
詳細については, OpenCV リファレンスの CalcOpticalFlowLK の項目を参照してください.

OpticalFlowPyrEx

構文 *object*.OpticalFlowPyrEx(<Input ID>, <Points>, <Win X>, <Win Y>, <Level>)

引数 <Input ID> = VT_I4:比較画像のイメージ ID
<Points> = VT_VARIANT|VT_ARRAY:探索点リスト
(<Point1>, <Point2>, …)
<World Point> = VT_R8 |VT_ARRAY:探索点座標 (<X>,<Y>)

<X> = VT_R8:X 座標
<Y> = VT_R8:Y 座標
<Win X> = VT_I4:探索ウィンドウサイズ(X 方向)
<Win Y> = VT_I4:探索ウィンドウサイズ(Y 方向)
<Level> = VT_I4:ピラミッドレベルの最大値

0	ピラミッドは用いられない(シングルレベル). レベル 2 となる.
1	ピラミッドレベルを2に設定します.
2	指定したレベルをピラミッドレベルの最大値とします.

戻り値

<Points> = VT_VARIANT|VT_ARRAY:ポイントリスト (<Point1>, <Point2>, …)
<Pointn> = VT_R4|VT_ARRAY:移動後の座標 (<X>,<Y>)
<X> = VT_R4:X 座標
<Y> = VT_R4:Y 座標

説明

Lucas-Kanade オプティカルフローの, ピラミッドを用いて疎な特徴に対応した反復バージョンです.
<Input ID>画像上の特徴点の座標から, 現在の画像における特徴点の座標を計算します. 座標はサブピクセル精度で検出します.
詳細については, OpenCV リファレンスの cvCalcOpticalFlowPyrLK の項目を参照してください.

BoxPoints

構文

object.BoxPoints(<X>, <Y>, <W>, <H>, <Angle>)

引数

<X> = VT_I4:中心 X 座標
<Y> = VT_I4:中心 Y 座標
<W> = VT_I4:幅
<H> = VT_I4:高さ
<Angle> = VT_I4:回転角度

戻り値

<Points> = VT_VARIANT|VT_ARRAY:四隅座標リスト
(<Point1>,<Point2>,<Point3>,<Point4>)
<Pointn> = VT_I4|VT_ARRAY:位置 (<X>,<Y>)
<X> = VT_I4:X 座標
<Y> = VT_I4:Y 座標

説明

指定矩形の四隅の座標を算出します.

[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

FindHomography

構文 *object*.FindHomography(<Point1>, <Point2>, <Point3>)

引数 <Points> = VT_VARIANT|VT_ARRAY:射影変換対応点リスト
(<Point1>, <Point2>, …)
<Pointn> = VT_VARIANT |VT_ARRAY:射影変換前後対応点
(<Before Point>,<After Point>)
<Before Point> = VT_R8 |VT_ARRAY:変換前座標 (<X>,<Y>)
<X> = VT_R8:X 座標
<Y> = VT_R8:Y 座標
<After Point> = VT_R8 |VT_ARRAY:変換後座標 (<X>,<Y>)
<X> = VT_R8:X 座標
<Y> = VT_R8:Y 座標

戻り値 <Matrix> = VT_R8|VT_ARRAY:射影行列
(<r11>,<r12>,<r13>,<r21>,<r22>,<r23>,<r31>,<r32>,<r33>)
<r11> = VT_R8:
<r12> = VT_R8:
<r13> = VT_R8:
<r21> = VT_R8:
<r22> = VT_R8:
<r23> = VT_R8:
<r31> = VT_R8:
<r32> = VT_R8:
<r33> = VT_R8:

$$\begin{pmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{pmatrix}$$

説明 射影変換行列を算出します.
射影変換の対応点を任意の個数だけ指定して射影変換行列を算出します.

関連項目 WarpPerspective

QRDecode

構文 *object*.QRDecode(<Code>)

引数 <Code> = コード種別

0	CODE_QR	QR コード Model 1 又は Model 2 をデコード
1	CODE_MICROQR	MicroQR をデコード
2	CODE_DATAMATRIX	DataMatrix をデコード
3	CODE_PDF417	PDF417 をデコード
4	CODE_BARCODE	バーコード (UPC/EAN , CODE39 , CODABAR(NW-7) , Interleaved 2 of 5(ITF), CODE128, EAN-128, RSS)をデコード
5	CODE_MICROPDF	MicroPDF417 をデコード
6	CODE_COMPOSITE	EAN.UCC Composite をデコード

戻り値 VT_VARIANT|VT_ARRAY:(<Data>, <Decode info>)

<Data> = VT_BSTR:デコードデータ

<Decode info> = VT_VARIANT|VT_ARRAY:デコード結果情報

説明 QR コード等のデコードを実行し, 読み取りデータを返します.
<Code>及び結果の<Decode info>の詳細については, QRdecoder のマニュアルを参照してください.

エラー

0x80101001 : QR デコーダの初期化がされていません. AddController 時に“QREnabled=True ”を指定してください.

0x80004005 QR コードの読み取りに失敗しました.

上記以外のエラーについては, 2.4 を参照してください.

OCRead

構文 *object*.OCRead()

引数 なし

戻り値 <Data> = VT_BSTR:識別文字列

説明 画像から文字を識別し読み込みます. ただし, 識別可能な文字は, 英数字に限定され

ています.
認識率を向上させるため, 事前に 2 値化処理をしてください.

エラー 0x80101001 : OCR の初期化がされていません. AddController 時に“OCREnabled=True ”を指定してください.
上記以外のエラーについては, 2.4 を参照してください.

4.3. コマンドクラス
4.3.1. 三角測量

Triangulation

構文 *object*.Triangulation(<Camera1 CAL ID>, <Camera2 CAL ID>, <Camera3 CAL ID>, <Xc1>, <Yc1>, <Xc2>, <Yc2>, <Xc3>, <Yc3>, <Tru-Method>)

引数
<Camera1 CAL ID> = VT_I4:カメラ 1 のキャリブレーション ID
<Camera2 CAL ID> = VT_I4:カメラ 2 のキャリブレーション ID
<Camera3 CAL ID> = VT_I4:カメラ 3 のキャリブレーション ID (0:無効)
<Xc1> = VT_R8:カメラ 1 の X 座標
<Yc1> = VT_R8:カメラ 1 の Y 座標
<Xc2> = VT_R8:カメラ 2 の X 座標
<Yc2> = VT_R8:カメラ 2 の Y 座標
<Xc3> = VT_R8:カメラ 3 の X 座標
<Yc3> = VT_R8:カメラ 3 の Y 座標
<Tru-Method> = VT_I4:三角測量の計算方法

0	Liner	特異値解析
1	Midpoint	中点解析

戻り値
<X> = VT_R8:X 座標
<Y> = VT_R8:Y 座標
<Z> = VT_R8:Z 座標

説明
2 台又は 3 台のカメラを用いて三角測量を行います.
事前にそれぞれのカメラのキャリブレーション及びカメラ設置位置を設定する必要があります.
<Camera3 ID>を 0 に設定したとき, カメラ 3 は無効になります. この場合は残りの 2 台のカメラによる三角測量を行います.

関連項目 CalibrateCamer, SetCamCalDat, GetPosFromCam

TriMatchTemplate

構文 *object*.TriMatchTemplate(<Camera1 ID>, <Camera2 ID>, <Camera3 ID>, <Input ID>, <Method>, <Threshold>, <Start angle>, <End angle>, <Step angle>,<Down sizing>, <Undistorted>, <Tru-Method>)

引数 <Camera1 ID> = VT_I4:カメラ 1 のイメージ ID

<Camera2 ID> = VT_I4:カメラ 2 のイメージ ID

<Camera3 ID> = VT_I4:カメラ 3 のイメージ ID(0:無効)

<Input ID> = VT_I4:テンプレート画像のイメージ ID

<Method> = VT_I4:マッチング方法

(I は画像を, T はテンプレートを, R は結果をそれぞれ表す. 総和計算は, 以下のようにテンプレートと(または)画像領域に対して行われる. x'=0..w-1, y'=0..h-1)

0	CV_TM_SQDIFF	$R(x,y)=\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2$
1	CV_TM_SQDIFF_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T(x',y')-I(x+x',y+y')]^2}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
2	CV_TM_CCORR	$R(x,y)=\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')]$
3	CV_TM_CCORR_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T(x',y')\cdot I(x+x',y+y')]}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$
4	CV_TM_CCOEFF	$R(x,y)=\sum_{x',y'}[T'(x',y')\cdot I'(x+x',y+y')]$ このとき $T'(x',y')=T(x',y')-\frac{\sum_{x'',y''}T(x'',y'')}{(w\cdot h)}$ $I'(x+x',y+y')=I(x+x',y+y')-\frac{\sum_{x'',y''}I(x+x'',y+y'')}{(w\cdot h)}$

5	CV_TM_CCOEFF_NORMED	$R(x,y)=\frac{\sum_{x',y'}[T'(x',y')\cdot I'(x+x',y+y')]}{\sqrt{\sum_{x',y'}T'(x',y')^2\cdot\sum_{x',y'}I'(x+x',y+y')^2}}$

<Threshold> = VT_R8:閾値

<Start angle> = VT_I4:検索開始角度(degree)

<End angle> = VT_I4:検索終了角度(degree)

<Step angle> = VT_I4:角度刻み(degree)

<Down sizing> = VT_I4:画面縮小回数

<Undistorted> = VT_BOOL:歪み補正フラグ

TRUE	歪み補正あり
FALSE	歪み補正なし

<Tru-Method> = VT_I4:三角測量の計算方法

0	Liner	特異値解析
1	Midpoint	中点解析

戻り値

<X> = VT_R8:X 座標

<Y> = VT_R8:Y 座標

<Z> = VT_R8:Z 座標

説明

2台又は3台のカメラのそれぞれでテンプレートマッチングを行い, 検出した座標を用いて三角測量を行います.

事前にそれぞれのカメラのキャリブレーション及びカメラ設置位置を設定する必要があります.

<Camera3 ID>を0に設定したとき, カメラ3は無効になります. この場合は残りの2台のカメラによる三角測量を行います.

[注意] バージョン 1.3.5 から回転角度の方向が時計回りに変更されています.

関連項目

CalibrateCamer, SetCamCalDat, GetPosFromCam, MatchTemplate2

TriMatchShapes

構文

object.TriMatchShapes(**<Camera1 ID>**, **<Camera2 ID>**, **<Camera3 ID>**, **<Input ID>**, **<Threshold>**, **<Type>**, **<Method>**, **<Min scale>**, **<Similarity>**, **<Undistorted>**, **<Tru-Method>**)

引数

<Camera1 ID> = VT_I4:カメラ 1 のイメージ ID

<Camera2 ID> = VT_I4:カメラ 2 のイメージ ID

<Camera3 ID> = VT_I4:カメラ 3 のイメージ ID(0:無効)

<Input ID> = VT_I4:テンプレート画像のイメージ ID

<Threshold> = VT_I4:閾値

<Type> = VT_I4:閾値処理タイプ

0	CV_THRESH_BINARY
1	CV_THRESH_BINARY_INV

<Method> = VT_I4:マッチング方法

ここで A は元画像, B はテンプレート画像を示す

0	CV_CONTOUR_MATCH_I1	$I_1(A,B)=\sum_{i=1}^{7}\left\lvert\frac{1}{m^A{}_i}-\frac{1}{m^B{}_i}\right\rvert$
1	CV_CONTOUR_MATCH_I2	$I_2(A,B)=\sum_{i=1}^{7}\left\lvert m^A{}_i-m^B{}_i\right\rvert$
2	CV_CONTOUR_MATCH_I3	$I_3(A,B)=\sum_{i=1}^{7}\frac{\left\lvert m^A{}_i-m^B{}_i\right\rvert}{\left\lvert m^A{}_i\right\rvert}$

ここで

$$m^A{}_i=\sin\left(h^A{}_i\right)\cdot\log\left(h^A{}_i\right)$$
$$m^B{}_i=\sin\left(h^B{}_i\right)\cdot\log\left(h^B{}_i\right)$$

$h^A{}_i$, $h^B{}_i$ は, A と B それぞれの Hu モーメント.

<Min scale> = VT_R8:最小倍率

<Similarity> = VT_R8:輪郭の一致度

<Undistorted> = VT_BOOL:歪み補正フラグ

TRUE	歪み補正あり
FALSE	歪み補正なし

<Tru-Method> = VT_I4:三角測量の計算方法

0	Liner	特異値解析
1	Midpoint	中点解析

戻り値

<X> = VT_R8:X 座標

<Y> = VT_R8:Y 座標

<Z> = VT_R8:Z 座標

説明

2 台又は 3 台のカメラのそれぞれで形状比較を行い, 検出した座標を用いて三角測量を

行います.
事前にそれぞれのカメラのキャリブレーション及びカメラ設置位置を設定する必要があります.
カメラの画像は, Canny フィルタで 2 値化画像に変換します. このため, <Input ID>画像は, Canny フィルタで変換した画像を指定してください.
<Input ID>画像から輪郭が複数抽出される場合, 正しく検出できない可能性があります. <Input ID>画像には, FindContours コマンドの結果が 1 になる画像を使用してください.
<Camera3 ID>を 0 に設定したとき, カメラ 3 は無効になります. この場合は残りの 2 台のカメラによる三角測量を行います.

関連項目 CalibrateCamer, SetCamCalDat, GetPosFromCam, MatchShapes2

TriHaarDetect

構文 *object*.TriHaarDetect(<Camera1 ID>, <Camera2 ID>, <Camera3 ID>, <Path>, <Scale>, <Min Neighbors>, <Undistorted>, <Tru-Method>)

引数

<Camera1 ID> = VT_I4:カメラ 1 のイメージ ID
<Camera2 ID> = VT_I4:カメラ 2 のイメージ ID
<Camera3 ID> = VT_I4:カメラ 3 のイメージ ID(0:無効)
<Path> = VT_BSTR:Haar ファイルのパス
<Scale> = VT_R8:スケール
<MinNeighbors> = VT_I4:最小隣接数
<Undistorted> = VT_BOOL:歪み補正フラグ

TRUE	歪み補正あり
FALSE	歪み補正なし

<Tru-Method> = VT_I4:三角測量の計算方法

0	Liner	特異値解析
1	Midpoint	中点解析

戻り値

<X> = VT_R8:X 座標
<Y> = VT_R8:Y 座標
<Z> = VT_R8:Z 座標

説明 2 台又は 3 台のカメラのそれぞれで Haar マッチングを行い, 検出した座標を用いて三角測量を行います.

事前にそれぞれのカメラのキャリブレーション及びカメラ設置位置を設定する必要があります.

<Camera3 ID>を 0 に設定したとき, カメラ 3 は無効になります. この場合は残りの 2 台のカメラによる三角測量を行います.

関連項目 CalibrateCamer, SetCamCalDat, GetPosFromCam, HaarDetect

5. OcvTester

5.1. 概要

OcvTester は OpenCV プロバイダを使用したアプリケーションで，画像処理のプロセスを対話的に進めるためのツールです．

OcvTester では，画像の処理を段階的に進めてそれぞれを別のウィンドウで表示することできます．また，実行した操作をスクリプト言語（CaoScript）として変換出力することもできます．

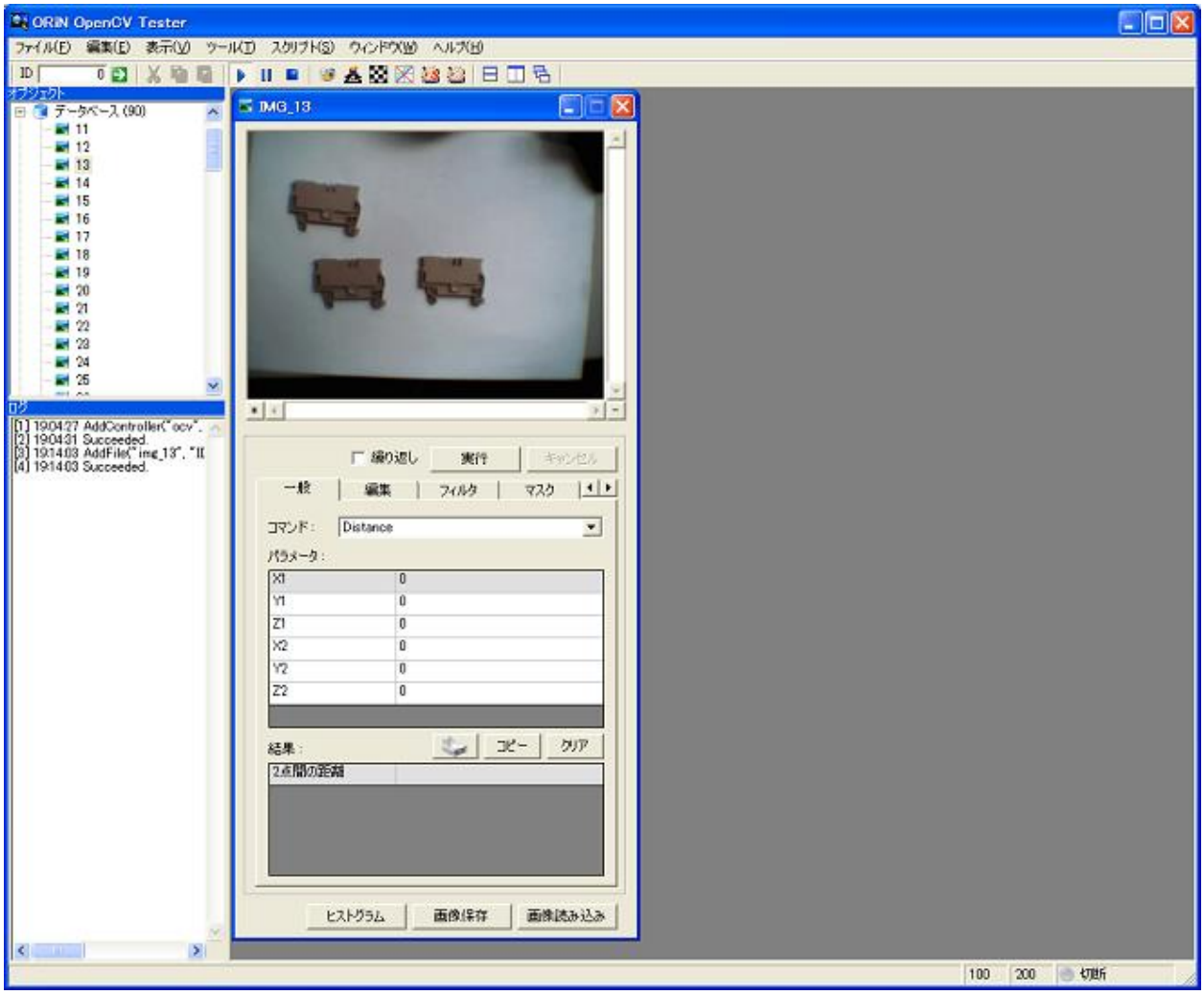


図 5-1 OcvTester 画面

メイン画面左上のツリービューをダブルクリックすることによって，選択されたノードに対応したイメージウィンドウを画面右側の領域に表示します．イメージウィンドウではコマンドの実行を行い．コマンドの実行ログはメイン画面左下のログウィンドウに表示されます．

5.2. メイン画面

メイン画面では，OpenCV プロバイダの設定，各ウィンドウの管理，ファイルの入出力等を行います．

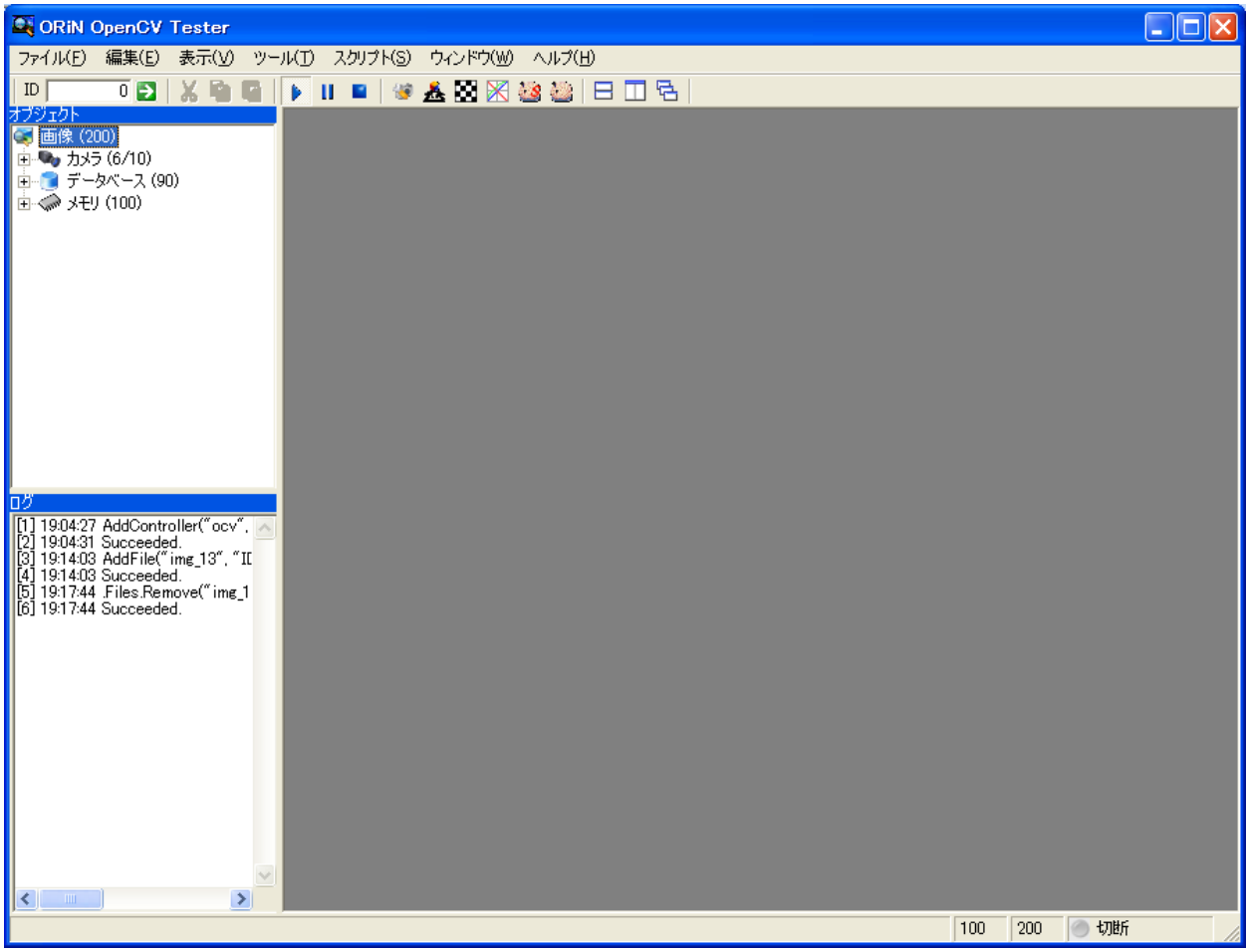


図 5-2　OcvTester 画面

5.2.1. オブジェクトウィンドウ

オブジェクトウィンドウでは，カメラ及びメモリ領域のイメージを管理しています．

表示したいオブジェクトをダブルクリックすることで，イメージウィンドウを表示します．

5.2.2. ログウィンドウ

コマンド実行時のログを出力します．

5.2.3. メニュー

5.2.3.1. ファイルメニュー

ファイルの保存を行います．

[ログのファイル出力] – Export Log

ログをファイルに出力します.

[スクリプトのファイル出力] – Export Script

記録されている, コマンドの実行内容を CaoScript ファイルとして出力します.

[接続] – Connect

デンソーロボットへの接続を行います. 接続時にデンソーロボット接続ウィンドウを表示します. 詳細は 5.4 を参照してください.

[切断] – Disconnect

デンソーロボットとの通信を切断します.

[終了] – Exit

OcvTester を終了します.

5.2.3.2. 編集メニュー

ログ出力やスクリプト記録などの編集処理を設定します.

[ログの出力] – Output Log

コマンドの実行ログをのログウィンドウに出力します.

[ログのクリア] – Clear Log

ログウィンドウの内容をクリアします.

[コピー] – Copy

テキストをクリップボードにコピーします.

[切り取り] – Cut

テキストを切り取ってクリップボードにコピーします.

[貼り付け] – Paste

クリップボードの内容をテキストに貼り付けします.

5.2.3.3. 表示メニュー

画面表示の設定を行います.

[オブジェクトウィンドウの切り替え] – Object Window

オブジェクトウィンドウの表示・非表示を切り替えます.

[ログトウィンドウの切り替え] – Log Window

ログウィンドウの表示・非表示を切り替えます.

[CAO スクリプトビューアの表示] – Script Viewer

CAO スクリプトビューアを表示します.

[CAO イベントビューアの表示] – Event Viewer

CAO イベントビューアを表示します.

[メインスクリーンの表示] – Main Screen

メインスクリーンを表示します.

5.2.3.4. ツールメニュー

ツールを表示します.

[カメラ設定] – Camera Setting

カメラ設定ツールを表示します. 詳細は 5.5 を参照してください.

[三角測量] – Triangulation

三角測量ウィンドウを表示します. 詳細は 5.6 を参照してください.

[キャリブレーション] – Calibration

キャリブレーションウィザードを表示します. 詳細は 5.7 を参照してください.

[ルックアップテーブルエディタ] – Look-up table editor

ルックアップテーブルエディタを表示します. 詳細は 5.8 を参照してください.

[イメージサンプリング] – Image Samples

イメージサンプリング作成ツールを表示します. 詳細は 5.9 を参照してください.

[Haar トレーニング] – Haar Traning

Haar トレーニングツールを表示します. 詳細は 5.10 を参照してください.

5.2.3.5. スクリプトメニュー

CAO スクリプトの自動生成機能に関する設定を行います.

[コマンド記録の開始] – Start Recording

コマンド実行内容の記録を開始します.

[コマンド記録の一時停止] – Pause Recording

コマンド実行内容の記録を一時停止します. 次にコマンド実行記録を開始したときは, これまでの記録に追加します.

[コマンド記録の停止] – Stop Recording

コマンド実行内容の記録を停止します. 次にコマンド実行記録を開始したときは, これまでの記録が破棄します.

[CAO スクリプトのコピー] – Copy Script

現在記録されている CAO スクリプトをクリップボードにコピーします.

[スクリプトのクリア] – ClearScript

現在記録されている CAO スクリプトをクリアします.

[CAO スクリプトマネージャ] – Script Manager

CAO スクリプトマネージャを起動します.

5.2.3.6. ウィンドウメニュー

ウィンドウの表示設定をします.

[垂直表示] – Horizontal

子ウィンドウを垂直方向に整列します.

[水平表示] – Vertical

子ウィンドウを水平方向に整列します.

[カスケード表示] – Cascade

子ウィンドウをカスケード状に整列します.

[最小化表示の整理] – Arrange Icon

最小化しているウィンドウを整列します.

[全て閉じる] – Close All

全ての子ウィンドウを閉じます.

5.2.3.7. ヘルプメニュー

ヘルプ情報を表示します.

[バージョン] – Version

バージョン情報を表示します.

5.3. イメージウィンドウ

イメージウィンドウは, 画像の表示と画像処理コマンドの実行を行います.

コマンドの実行は以下の手順で行います.

(1) タブからコマンドの分類を選択します.

(2) Command コンボボックスからコマンドを選択します.

(3) Parameter で実行コマンドのパラメータを設定します.

(4) Execute ボタンをクリックして実行します.

(5) Result に取得した値を表示します.(コマンドによっては値を取得しないものもあります.)実行結果は, メイン画面左下のログウィンドウに表示されます.

図 5-3 OcvTester 画面

[画像モニタ]

イメージウィンドウに設定されている画像を表示します.

左下の[*]ボタンが ON のとき，メインスクリーンへの出力を行います．

右上の[=]ボタンで，キャプチャスクリーンを表示します．この機能はメモリ領域の画像でのみ使用することができます．

右下の[+]または[-]ボタンで画像モニタの以下の表示・非表示を切り替えます．

また，一部のコマンドでは，画像上をドラッグすることでパラメータを設定できます．以下に設定できるコマンドの一覧を示します．

表 5-1 範囲指定可能なコマンド一覧

形状	コマンド	操作	
矩形	SetROI Cut Rectangle CamShift	移動	5 画素 : [↑] [↓][←][→] 1 画素 : [Ctrl] + [↑] [↓][←][→]
		リサイズ	5 画素 : [Shift] + [↑] [↓][←][→] 1 画素 : [Ctrl] + [Shift] + [↑] [↓][←][→]
		自動フィッティング	[Shift]+[Enter]
線	Line	移動	5 画素 : [↑] [↓][←][→] 1 画素 : [Ctrl] + [↑] [↓][←][→]
		リサイズ	5 画素 : [Shift] + [↑] [↓][←][→] 1 画素 : [Ctrl] + [Shift] + [↑] [↓][←][→]
点	PutColor GetColor SearchPoint Paste Rotate Line2 Circle Ellipse Sector Cross Text ContoursNumber PointPolygonTest	移動	5 画素 : [↑] [↓][←][→] 1 画素 : [Ctrl] + [↑] [↓][←][→]

	GetPosFromCam BoxPoints	

[リピート] – Repeat

コマンド実行時に同じ処理を繰り返します．繰り返し実行を止める時は，Repeat のチェックを外すか，Cancel ボタンをクリックしてください．実行中はログを大量に出力します．

[実行] – Execute

コマンドを実行します．Command コンボで選択したコマンドを実行します．実行時のパラメータは，Parameter プロパティボックスで設定された内容を実行します．実行結果は Result プロパティボックスに表示します．実行結果はログウィンドウに出力されます．

Repeat がチェックされているときは，Cancel ボタンがクリックされるまで繰り返し実行します．

[キャンセル] – Cancel

繰り返し実行中のコマンドを停止します．

[コマンド種別]

このタグでコマンドの種別を選択します．

[コマンド] – Command

実行コマンドを選択します．

[パラメータ] – Parameter

コマンド実行時のパラメータを設定します．

[結果] – Result

コマンドの実行結果として取得したデータを表示します．

実行結果の内容は，Copy ボタンをクリックすることでクリップボードに保存することができます．

Clear ボタンをクリックすることで，現在表示されている実行結果を消去します．

データ転送ボタンをクリックすることで，デンソーロボットに結果を転送します．

[ヒストグラム] – Histgram

画像のヒストグラムを表示します．ヒストグラムの内容は，“CalcHistEx”コマンド実行時に更新します．

[画像の保存] – Save Image

画像モニタに表示されている画像をビットマップとして保存します．

[画像の読み込み] – Load Image

ビットマップを読み込み，画像モニタに表示します．このボタンは，カメラ領域のイメージウィンドウでは使用できません．

5.4. デンソーロボット接続ウィンドウ

デンソーロボットへの接続設定を行います．

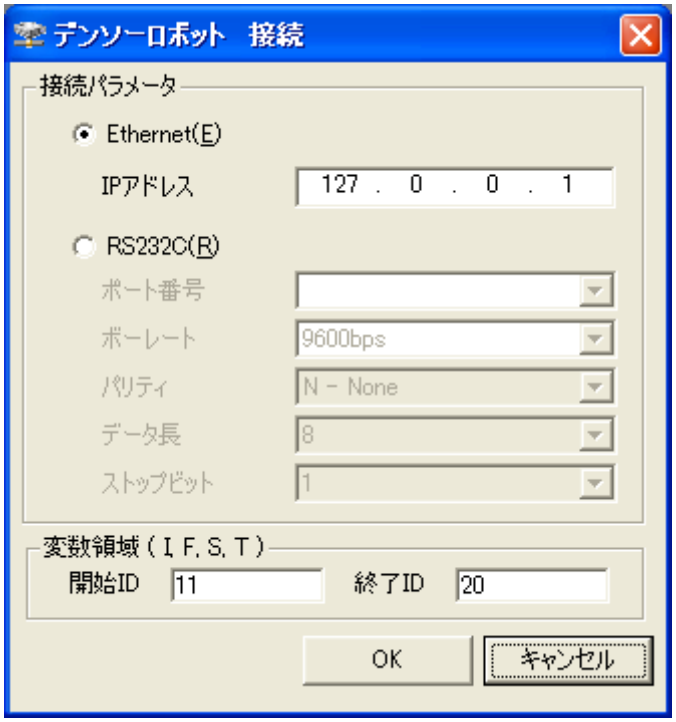


図　5-4　デンソーロボット接続ウィンドウ

[イーサネット接続] – Ethernet

ロボットコントローラにイーサネットで接続します.

[IP アドレス] – IP Address

ロボットコントローラの IP アドレスを指定します.

このパラメータはイーサネット接続を指定したときのみ設定することができます.

[RS232C 接続] – RS232C

ロボットコントローラに RS232C で接続します.

[ポート番号] – Port No.

ロボットコントローラと通信を行う COM ポート番号を指定します.

このパラメータは RS232C 接続を指定したときのみ設定することができます.

[ボーレート] – Baudrate

ロボットコントローラと通信を行う COM ポートの通信速度を指定します.

このパラメータは RS232C 接続を指定したときのみ設定することができます.

[パリティ] – Parity

ロボットコントローラと通信を行う COM ポートのパリティを指定します.

この項目は常にパリティなしに設定されており, 変更することはできません.

[データ長] – Data Length

ロボットコントローラと通信を行う COM ポートのデータ長を設定します.

この項目は常に8ビットに設定されており, 変更することはできません.

[ストップビット] – Stop Bit

ロボットコントローラと通信を行う COM ポートの通信速度を設定します.

この項目は常に 1 ビットに設定されており, 変更することはできません.

[変数領域] – Variable Range

デンソーロボット結果出力の実行時に使用する変数の範囲を設定します.

5.5. カメラ設定ウィンドウ

カメラの設定及び, イメージメモリサイズの設定を行います.

設定内容を反映するためには, OcvTester を再起動する必要があります.

設定内容はレジストリに保存されます. (2.1 章参照)

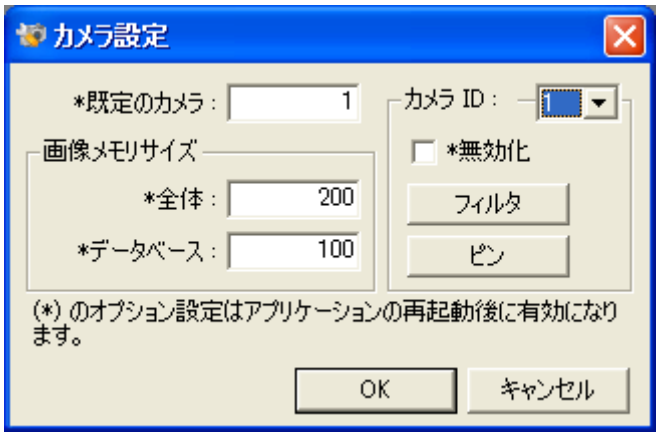


図 5-5 カメラ設定ウィンドウ

[カメラ ID] – Camera ID

設定を行うカメラ ID を指定します.

[カメラ使用許可] – Disable

カメラの使用許可設定を行います. この項目にチェックが入っているときは, このカメラの使用を禁止します.

[フィルタ] – Filter

カメラフィルタのプロパティウィンドウを表示します.

[ピン] – Pin

カメラの出力ピンのプロパティウィンドウを表示します.

[イメージメモリサイズ設定] – Image Max

イメージメモリの最大インデックス番号を指定します.

[データベース領域サイズ設定] – DB Max

イメージメモリの最大インデックス番号を指定します.

5.6. 三角測量ウィンドウ

三角測量で座標を算出します.

この機能を使用するためにはキャリブレーション済みのカメラが2～3台必要になります.

図 5-6 三角測量ウィンドウ

[コマンド] – Command

実行コマンドを選択します.

[パラメータ] – Parameter

コマンド実行時のパラメータを設定します.

[リピート] – Repeat

コマンド実行時に同じ処理を繰り返します. 繰り返し実行を止める時は, Repeat のチェックを外すか, Cancel ボタンをクリックしてください. 実行中はログを大量に出力します.

[実行] – Execute

コマンドを実行します. Command コンボで選択したコマンドを実行します. 実行時のパラメータは, Parameter プロパティボックスで設定された内容を実行します. 実行結果は Result プロパティボックスに表示します. 実行結果はログウィンドウに出力されます.

Repeat がチェックされているときは, Cancel ボタンがクリックされるまで繰り返し実行します.

[キャンセル] – Cancel

繰り返し実行中のコマンドを停止します.

[結果] – Result

コマンドの実行結果として取得したデータを表示します.

5.7. キャリブレーションウィザード

5.7.1. 概要

カメラキャリブレーション及びロボットキャリブレーションを行うウィザードを表示します.

キャリブレーションウィザードは以下の4つのキャリブレーション対象を指定することができます.

[カメラ] – Camera

カメラキャリブレーションを実行します.

実行するコマンドは, CalibreateCamera を実行します.

[ロボット] – Robot

ロボットキャリブレーションを実行します.

実行するコマンドは, CalibreateRobot を実行します.

[カメラ＋ロボット] – Camera attached on a cell & Robot

定点カメラのカメラキャリブレーションとロボットキャリブレーションを実行します.

実行するコマンドは, CalibrateCamera と CalibreateRobot を実行します.

また, キャリブレーションウィザードは6つのステップでキャリブレーションを実行します.

[Step 0] キャリブレーション対象の指定

[Step 1] カメラキャリブレーションのパラメータ設定

[Step 2] チェスボード画像の取込み

[Step 3] ワールド-ロボット座標間のマッピング

[Step 4] 完了画面

各ステップの詳細については後述します.

実行されるステップは選択したキャリブレーション対象によって異なります. 以下にキャリブレーション対象ごとの実行ステップの一覧を示します.

表 5-2 キャリブレーション対象ごとの実行ステップ

キャリブレーション対象	Step 0	Step 1	Step 2	Step 3	Step 4
カメラ	○	○	○	→	○
ロボット	○	→	→	○	○
カメラ＋ロボット	○	○	○	○	○

5.7.2. Step 0： キャリブレーション対象の指定

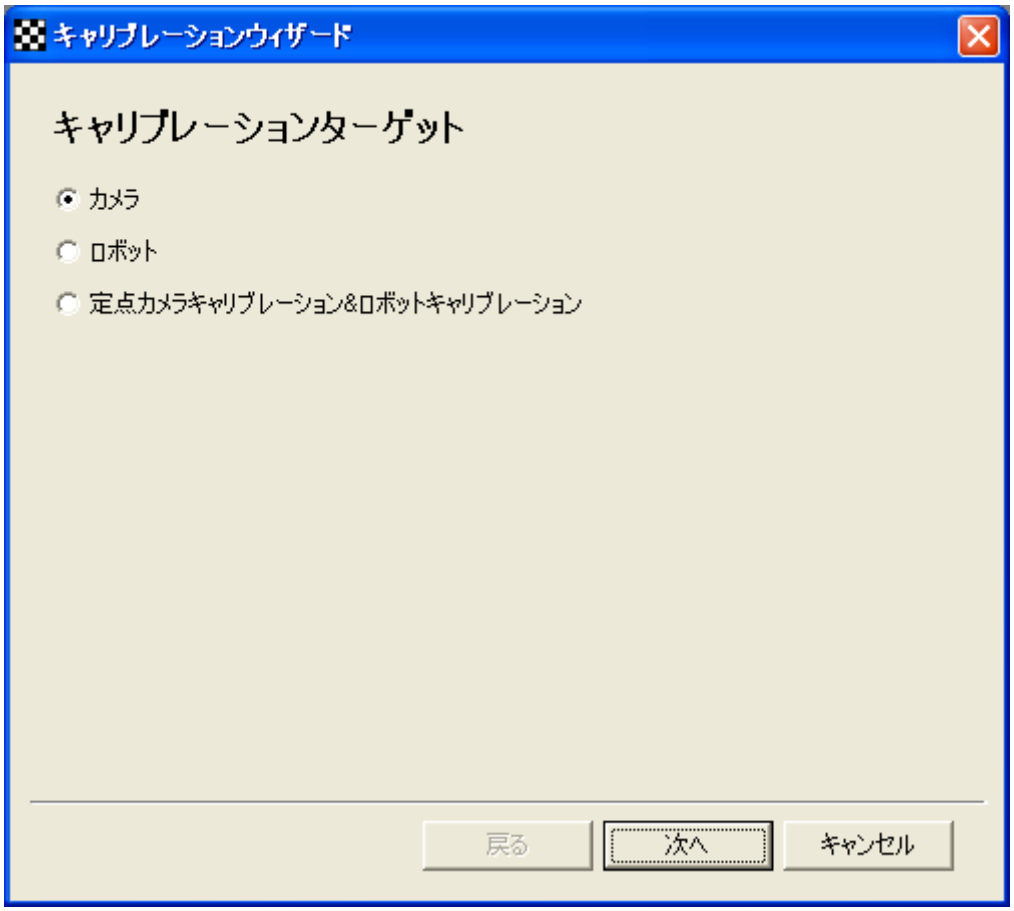


図 5-7 Step 0： キャリブレーションターゲットの指定

この画面では，実行するキャリブレーションターゲットを指定します．各ターゲットの詳細については5.7.1を参照してください．

5.7.3. Step 1： カメラキャリブレーションのパラメータ設定

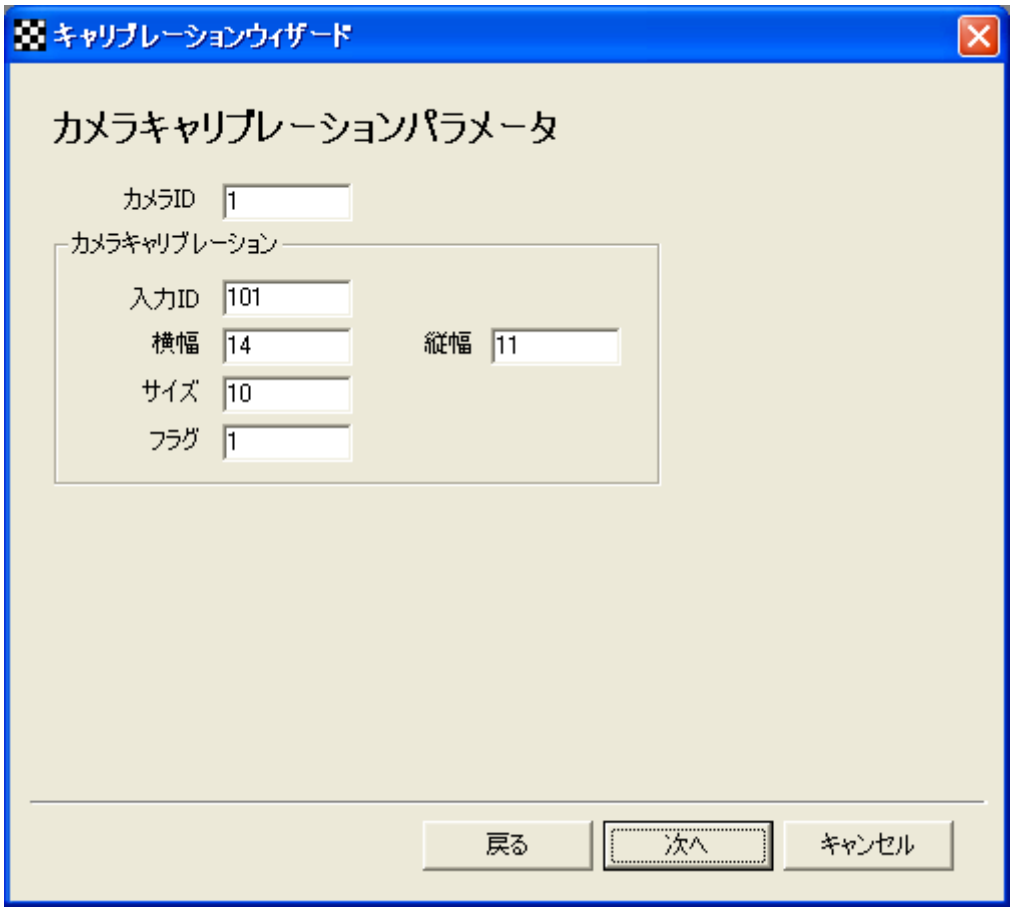


図 5-8 Step 1 : カメラキャリブレーションのパラメータ設定

[カメラ ID] – Camera ID

キャリブレーションを行うカメラの ID を指定します.

[入力画像の格納先 ID] – Input ID

カメラキャリブレーション時のチェスボード画像格納先の ID を指定します. 指定した ID を先頭として画像を格納していきます.

[チェスボードの幅] – Width

画像読取りを行うチェスボードの横方向の升目数を指定します.

[チェスボードの高さ] – Height

画像読取りを行うチェスボードの縦方向の升目数を指定します.

[チェスボードの升目の大きさ] – Size

画像読取りを行うチェスボードの升目の大きさ.

[カメラキャリブレーションのフラグ] – Flag

カメラキャリブレーションのフラグを指定します.

5.7.4. Step 2： チェスボード画像の取込み

図 5-9 Step 3： チェスボード画像の取込み

[現在のカメラ画像] – Current Image

指定した ImageID の画像を表示します.

[イメージ ID] – Image ID

画像取り込みを行うイメージ ID を指定します.

[イメージリスト] – Image List

取り込んだ画像の一覧を表示します. リストをクリックすると選択した画像をサムネイルに表示します.

[基準画像] – BaseImage

カメラキャリブレーション時に基準画像として使用する取り込み画像の番号を指定します.

[画像追加] – Add

現在の画像をリストに取り込みます. このときチェスボードのチェックに成功したものをイメージリストに追加します. チェスボードのチェック結果はサムネイルに表示されます.

[画像削除] – Delete

現在の選択されている画像をイメージリストから削除します.

5.7.5. Step 3： ワールド-ロボット座標間のマッピング

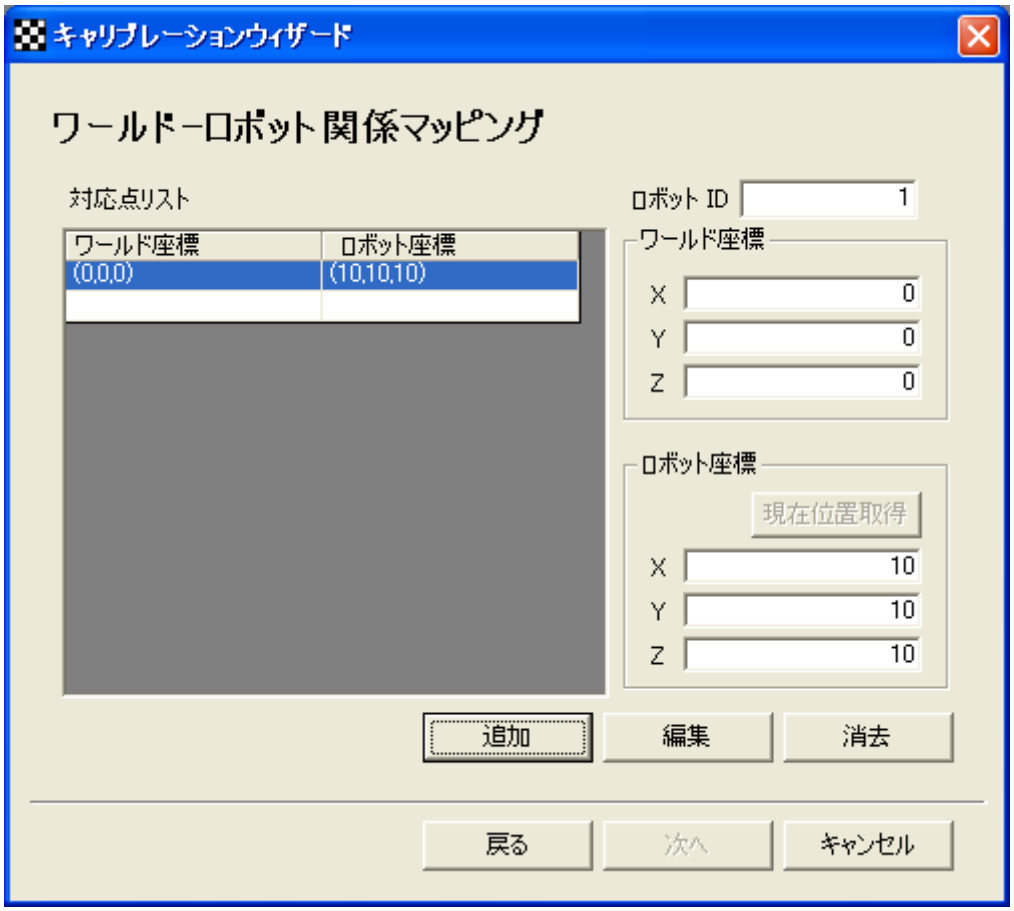


図 5-10 Step 4： ワールド-ロボット座標間のマッピング

[マッピング一覧] – Points

ワールド-ロボット座標間のマッピング一覧を表示します.

選択した対応点リストは, ワールド座標及びロボット座標の入力項目に表示します.

[ワールド座標] – World Point

ワールド座標の 1 点を指定します.【参考】

[ロボット座標] – Robot Point

ロボット座標の 1 点を指定します.【参考】

[現在座標取得] – GetCurrentPos

ロボットコントローラから現在の座標を取得して, ロボット座標の各項目を設定します.

[マッピングリスト追加] – Add

入力されているワールド座標とロボット座標の対応点としてリストに追加します.

[マッピングリスト編集] – Edit

入力されているワールド座標とロボット座標の対応点で選択されているリスト項目を変更します.

[マッピングリスト削除] – Delete

現在選択しているマッピング情報をリストから削除します.

【参考】　カメラ＋ロボットのキャリブレーションの場合

カメラキャリブレーションで指定したベース画像（チェスボード）の左下がワールド座標系原点（0, 0, 0）になっているので，カメラ＋ロボットのキャリブレーションの場合は，そのチェスボードを使ってそのままワールド座標系とロボット座標系を指定するのが簡単です．具体的にはワールド座標はそのチェスボードの任意の点(X, Y, 0)を指定します．次にその点にロボットの先端を移動して[現在位置取得]をクリックします．この作業を 3 点以上で行います．ロボットの先端にはティーチング用のポインタ等を設置しておくと便利です．

5.7.6. Step 4： 完了画面

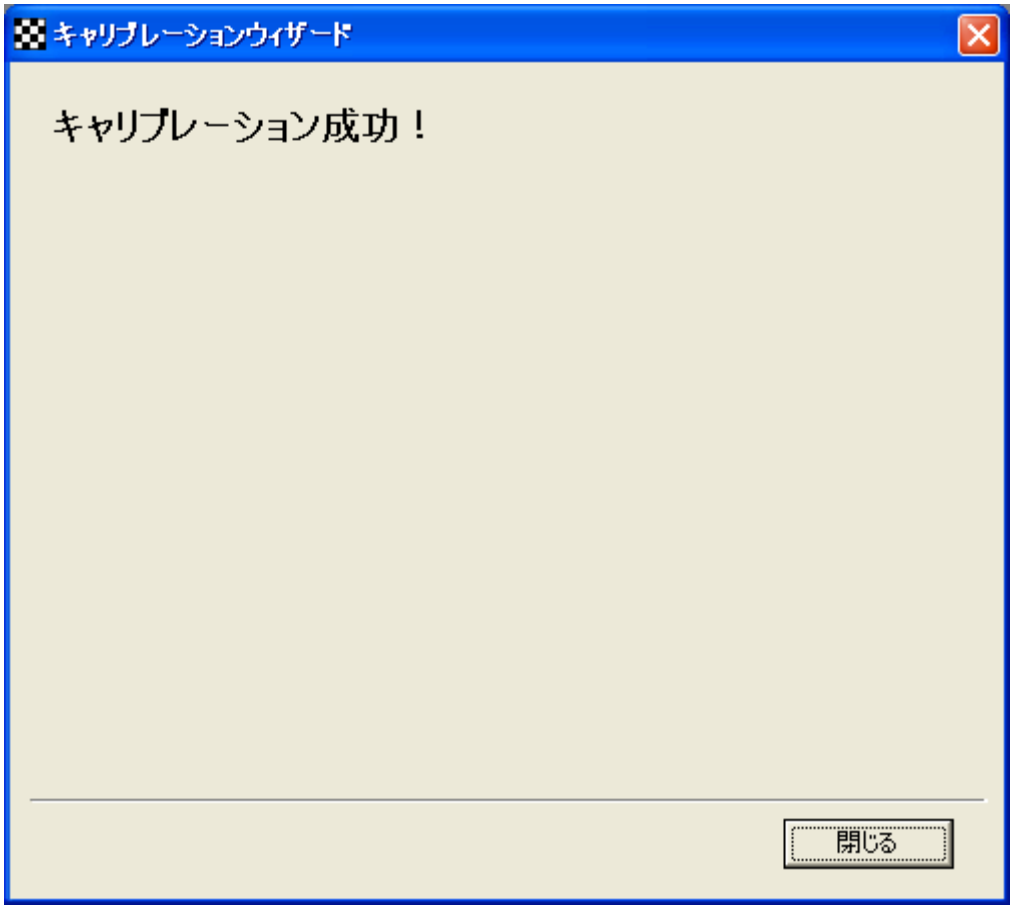


図　5-11 Step 5： 完了画面

キャリブレーションが完了したときは，この画面が表示されます．Close ボタンをクリックしてキャリブレーションウィザードを終了します．

5.8. ルックアップテーブルエディタ

ルックアップテーブルの編集を行います．

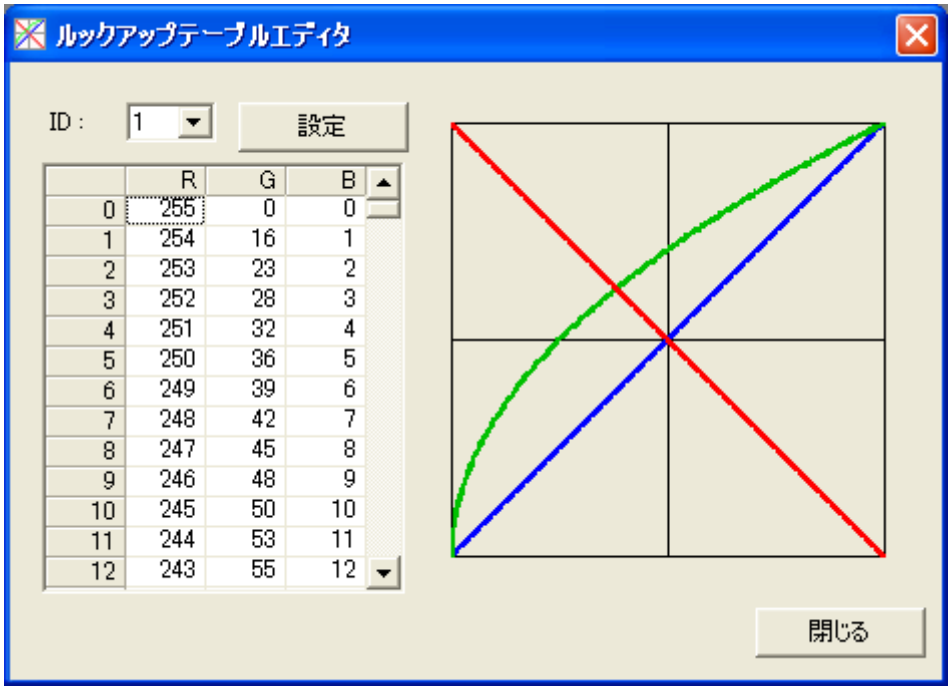


図 5-12　ルックアップテーブルエディタ

[テーブル ID] – ID

編集を行うルックアップテーブル ID を指定します.

ルックアップテーブルは 10 個用意しています.

[ルックアップテーブル] – Look-up table

現在のルックアップテーブルの表示・編集を行います.

R・G・B は色空間を示します.

[テーブルグラフ] – Graph

現在のルックアップテーブルの各色空間のデータをグラフで表示します.

[設定] – Setting

編集したルックアップテーブルを設定します.

[閉じる] – Close

ルックアップテーブルエディタを終了します.

5.9. イメージサンプリングウィンドウ

OpenCV に付属する CreateSamples.exe を用いて Haar トレーニングで使用するポジティブイメージを作成します.

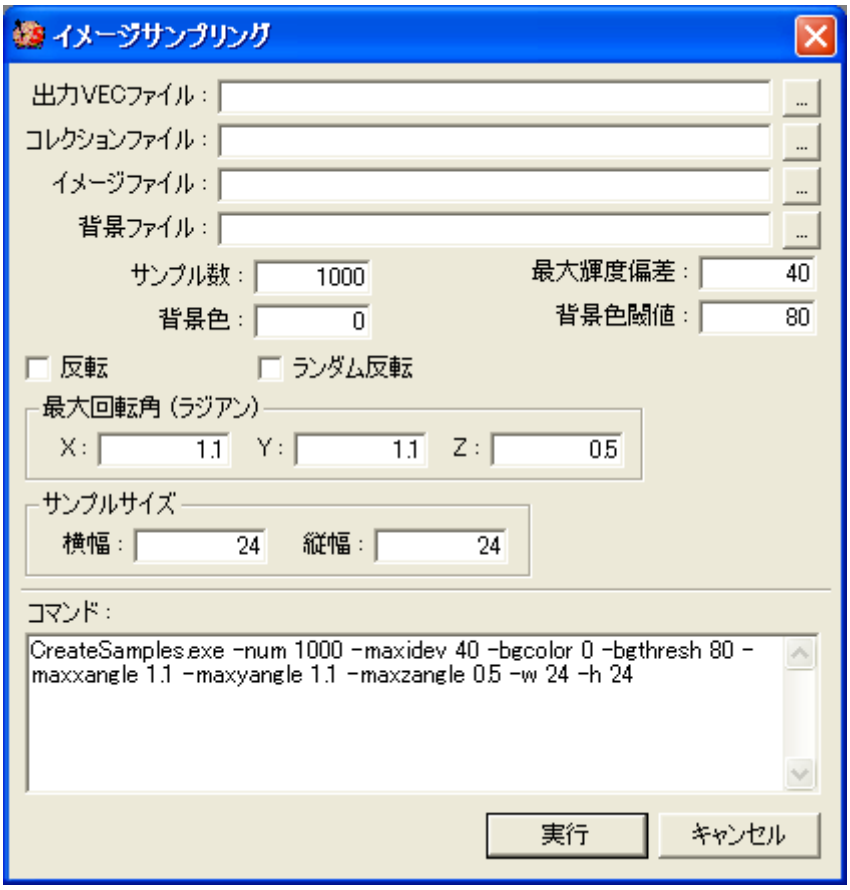


図 5-13 イメージサンプリングウィンドウ

[出力 VEC ファイル] – Output vec file

生成したポジティブイメージファイルの保存先を指定します.

[コレクションファイル] – Collection file

コレクションファイルを指定します. コレクションファイルには, 検出イメージが含まれる画像ファイルのリストが記述されています. 詳しい書式については, CreateSamples.exe のドキュメントを参照してください.

[イメージファイル] – Collection file

検出イメージの画像ファイルを指定します.

[背景ファイル] – Background file

背景画像のコレクションファイルを指定します.

[サンプル数] – Number of samples

出力するポジティブイメージのサンプル数を指定します.

[最大輝度偏差] – Max intensith deviation

指定された前面画像の最大輝度偏差を指定します.

[背景色] – Background color

指定された背景色を透明にします.

[背景色閾値] – Background color threshold

“Background color” ± “background color threshold”の範囲を透明にします.

[反転] – Inverse

色を反転します.

[ランダム反転] – Random inverse

色をランダムに反転します.

[最大回転角] – Max angle

検出画像の最大回転角を指定します.

[サンプルサイズ] – Sample size

作成するポジティブイメージのサイズを指定します.

[コマンド] – Command

実行する CreateSamples.exe コマンドラインで表現します.

[実行] – Run

Command で記述されたコマンドを実行します.

[キャンセル] –Cancel

コマンドを中止し, ウィンドウを閉じます.

5.10. Haar トレーニングウィンドウ

OpenCV に付属する HaarTraining.exe を用いて“HaarDetect”コマンドで使用する XML ファイルを生成します.

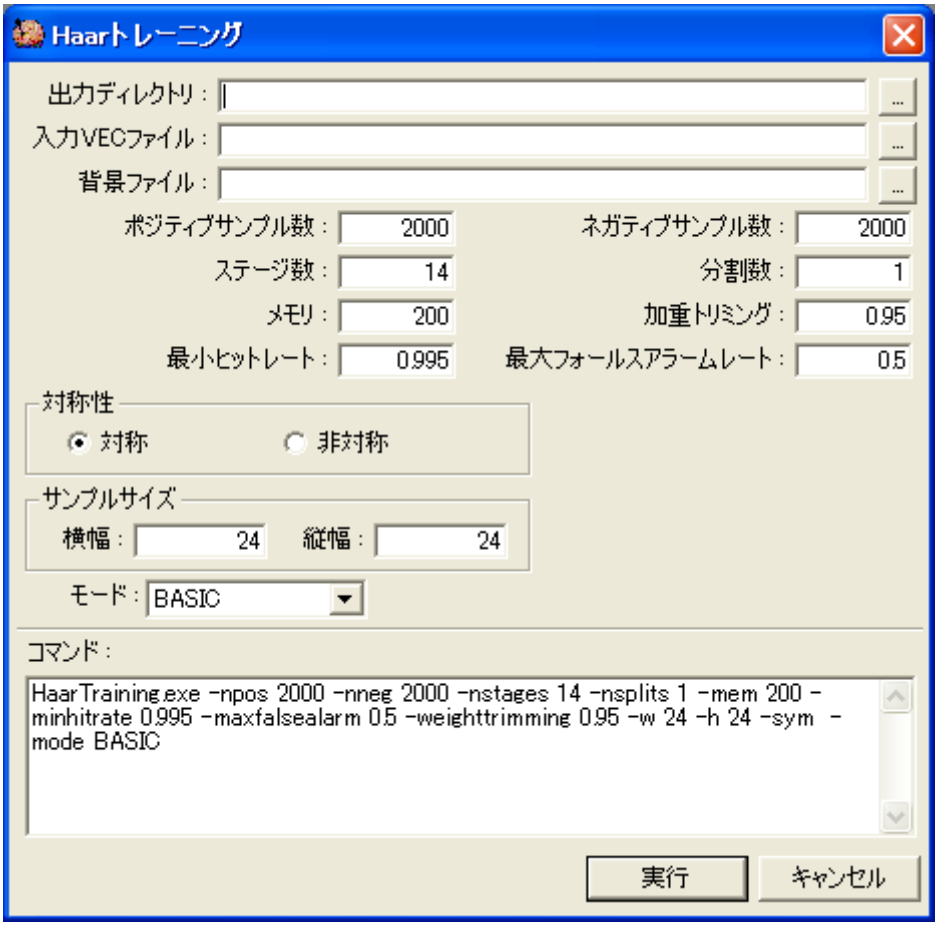


図 5-14　イメージサンプリングウィンドウ

[出力ディレクトリ] – Output directory

Haar トレーニングの結果の出力先を指定します.

[入力 VEC ファイル] – Vec file

ポジティブイメージファイルを指定します.

[背景ファイル] – Background file

背景ファイルを指定します.

[ポジティブサンプル数] – Number of positive samples

各分類器のステージトレーニングで使用するポジティブサンプル数を指定します.

[ネガティブサンプル数] – Number of negative samples

各分類器のステージトレーニングで使用するネガティブサンプル数を指定します.

[ステージ数] – Number of stage

トレーニングを行うステージの数を指定します.

[分割数] – Number of split

分類器のステージで使用される弱分類器を決定します. もし1ならば, 単純な切株分類器を使用します.

2以上ならば“Number of split”内部(分岐)ノードを持つ CART 分類器が用いられます.

[メモリ] – Memory in MB

呼び計算に使用するメモリ量を指定します.

[加重トリミング] – Weight trimming

加重トリミングをどれくらい使用するのかを指定します.

[最小ヒットレート] – Min hit rate

各ステージの分類器に必要なヒットレートの最小量を指定します.

[最大フォールスアラームレート] – Max false alarm rate

各ステージの分類器に対して要求するフォールスアラームレートの最大数を指定します.

[対称性] – Symmetry

トレーニングにおいて対象物が垂直軸に対して対象があるかどうかを指定します.

[サンプルサイズ] – Sample size

ポジティブイメージのサイズを指定します.

[モード] – Mode

訓練で使用する Haar 特徴集合のタイプを選択します.

[コマンド] – Command

実行する HaarTraining.exe コマンドラインで表現します.

[実行] – Run

Command で記述されたコマンドを実行します.

[キャンセル] – Cancel

コマンドを中止し，ウィンドウを閉じます.

付録A. OpenCV メソッド実装一覧

表 A-1 OpenCV メソッド実装一覧

CaoFile::Execute コマンド名	OpenCV メソッド
SetROI	cvRect cvSetImageROI
GetROI	cvGetImageROI
ResetROI	cvResetROI
PutColor	cvSet2D
GetColor	cvGet2D
ImageSize	-
Trim	cvCreateImage cvGetSize cvCvtColor cvCopy cvReleaseImage cvGet2D
SearchPoint	cvCreateImage cvGetSize cvCvtColor cvCopy cvReleaseImage cvGet2D
Distance	-
InnerProduct	-
OuterProduct	-
Copy	cvCreateImage cvGetSize cvCopy cvReleaseImage
Cut	cvReleaseImage cvRect cvSetImageROI cvCreateImage

	cvGetSize cvCopy
Paste	cvCreateImage cvGetSize cvCvtColor cvRect cvSetImageROI cvCopy cvConvert cvReleaseImage
Rotate	cvCreateImage cvGetSize cvCreateMat cv2DRotationMatrix cvWarpAffine cvReleaseImage cvReleaseMat
Flip	cvCreateImage cvGetSize cvFlip
Resize	cvCreateImage cvSize cvResize
Split	cvCreateImage cvGetSize cvSplit cvReleaseImage
Marge	cvCreateImage cvGetSize cvMerge
ConvertGray	cvCreateImage cvGetSize cvCvtColor cvCopy cvReleaseImage
ThresholdEx	cvCreateImage

	cvGetSize cvCvtColor cvCopy cvThreshold cvReleaseImage
Threshold2	cvCreateImage cvGetSize cvCvtColor cvCopy cvThreshold cvReleaseImage
AdaptiveThresholdEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvAdaptiveThreshold cvReleaseImage
Smooth	cvCreateImage cvGetSize cvSmooth cvReleaseImage
Sobel	cvCreateImage cvGetSize cvCvtColor cvCopy cvSobel cvConvert cvReleaseImage
Laplace	cvCreateImage cvGetSize cvCvtColor cvCopy cvLaplace cvConvert cvReleaseImage
CannyEx	cvCreateImage

	cvGetSize cvCvtColor cvCopy cvCanny cvReleaseImage
WarpAffine	cvCreateImage cvGetSize cvCreateMat cvSetReal2D cvWarpAffine cvReleaseImage cvReleaseMat
WarpPerspective	cvCreateImage cvCreateMat cvSetReal2D cvWarpPerspective cvReleaseMat
PreCornerDetectEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvConvert cvPreCornerDetect cvConvert cvReleaseImage
CornerHarrisEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvConvert cvCornerHarris cvReleaseImage
CalcBackProjectEx	cvCreateImage cvGetSize cvCvtColor cvCopy

	cvCreateHist cvCalcHist cvCalcBackProject
Inpaint	cvCreateImage cvGetSize cvInpaint
Erode	cvCreateStructuringElementEx cvCreateImage cvErode cvReleaseStructuringElement
Dilate	cvCreateStructuringElementEx cvCreateImage cvDilate cvReleaseStructuringElement
PyrDown	cvGetSize cvCreateImage cvPyrDown
PyrUp	cvGetSize cvCreateImage cvPyrUp
NOT	cvCreateImage cvGetSize cvNot
AND	cvCreateImage cvGetSize cvAnd cvReleaseImage
OR	cvCreateImage cvGetSize cvOr cvReleaseImage
XOR	cvCreateImage cvGetSize cvXor cvReleaseImage
ADD	cvCreateImage

	cvGetSize cvAdd cvReleaseImage
SUB	cvCreateImage cvGetSize cvSub cvReleaseImage
MAXEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvMax cvReleaseImage
MINEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvMin cvReleaseImage
ABS	cvCreateImage cvGetSize cvAbsDiff cvReleaseImage
Line	cvCreateImage cvGetSize cvCopy cvLine cvPoint cvReleaseImage
Line2	cvCreateImage cvGetSize cvCopy cvLine cvPoint cvReleaseImage
Rectangle	cvCreateImage

	cvGetSize cvCopy cvRectangle cvPoint cvReleaseImage
Circle	cvCreateImage cvCopy cvCircle cvPoint cvReleaseImage
Ellipse	cvCreateImage cvGetSize cvCopy cvEllipse cvPoint cvSize cvReleaseImage
Sector	cvCreateImage cvGetSize cvCopy cvLine cvEllipse cvPoint cvSize cvReleaseImage
Cross	cvCreateImage cvGetSize cvCopy cvLine cvPoint cvReleaseImage
Text	cvCreateImage cvGetSize cvFlip cvInitFont cvPutText

	cvPoint cvReleaseImage
FindContoursEx	cvCreateImage cvGetSize cvClearMemStorage cvFindContours cvReleaseImage
CopyContours	cvBoundingRect cvReleaseImage cvRect cvSetImageROI cvCreateImage cvGetSize cvCopy
ContoursNumber	cvPointPolygonTest
PointPolygonTest	cvPointPolygonTest
BoundingRect	cvBoundingRect
FitEllipse	cvFitEllipse2
ArcLength	cvArcLenth
CheckContourConvexity	cvCheckContourConvexity
FindBlobs	-
BlobsFilter	-
BlobResult	-
BlobResults	-
BlobEllipse	-
BlobMatchTemplate	cvCloneImage cvCreateImage cvCopy cvGetSize cvResize cvSet cvSetImageROI cvResetImageROI cvCreateMemStorage cvCreateSeq cvPoint2D32f

	cvSize2D32f cvCreateMat cv2DRotationMatrix cvWarpAffine cvBoxPoints cvMatchTemplate cvMinMaxLoc cvSeqPush cvSeqSort cvGetSeqElem cvClearSeq cvReleaseImage cvReleaseMat cvReleaseMemStorage
BlobMatchShapes	cvCloneImage cvCreateMemStorage cvCreateImage cvCvtColor cvCopy cvClone cvFindContours cvBoundingRect cvSetImageROI cvClearMemStorage cvFindContours cvMatchShapes cvFitEllipse2 cvReleaseImage cvReleaseMemStorage
CalcHistEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvCreateHist cvCalcHist cvQueryHistValue_1D

	cvReleaseHist cvReleaseImage
NormalizeHistEx	cvCreateHist cvSetReal1D cvNormalizeHist cvQueryHistValue_1D cvReleaseHist
ThreshHistEx	cvCreateHist cvSetReal1D cvThreshHist cvQueryHistValue_1D cvReleaseHist
EqualizeHistEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvEqualizeHist
GetMinMaxHistValue	cvCreateHist cvSetReal1D cvGetMinMaxHistValue cvReleaseHist
HistAve	-
AutoThresPTile	-
AutoThreshMode	-
AutoThreshDiscrim	-
MatchTemplate	cvCreateImage cvSize cvMatchTemplate cvMinMaxLoc cvReleaseImage
MatchShapesEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvMatchShapes cvReleaseImage

CamShift	cvRect cvTermCriteria cvCamShift
MatchTemplate2	cvCreateImage cvCopy cvGetSize cvResize cvSet cvSetImageROI cvResetImageROI cvCreateMemStorage cvCreateSeq cvPoint2D32f cvSize2D32f cvCreateMat cv2DRotationMatrix cvWarpAffine cvBoxPoints cvMatchTemplate cvMinMaxLoc cvSeqPush cvSeqSort cvGetSeqElem cvClearSeq cvReleaseImage cvReleaseMat cvReleaseMemStorage
MatchShapes2	cvCreateMemStorage cvCreateImage cvCvtColor cvCopy cvClone cvFindContours cvBoundingRect cvSetImageROI cvClearMemStorage

	cvFindContours cvMatchShapes cvFitEllipse2 cvReleaseImage cvReleaseMemStorage
HaarDetect	cvLoad cvCreateMemStorage cvHaarDetectObjects cvGetSeqElem cvReleaseMemStorage
CalibrateCamare	cvCreateMat cvCalibrateCamera2 cvRodrigues2 cvMat cvMatMul cvInvert cvSetReal2D cvGetReal2D cvSet2D cvReleaseMat cvCreateImage cvFindChessboardCorners cvCvtColor cvFindCornerSubPix cvReleaseImage
CalibrateRobot	cvCreateMat cvmSet cvSolve cvReleaseMat
FindChessBoardCorners	cvFindChessboardCorners
DrawChessBoardCorners	cvCreateImage cvCopy cvDrawChessboardCorners
DrawXYAxes	cvCreateImage cvGetSize cvCopy

	cvLine cvPoint cvReleaseImage cvFlip cvInitFont cvPutText
SetCamCalDat	-
GetCamCalDat	-
SetCamCalExtDat	-
GetCamCalExtDat	-
SetRobCalDat	-
GetRobCalDat	-
GetPosFromCam	cvCreateMat cvmSet cvmGet cvMatMul cvReleaseMat
GetCamPos	cvCreateMat cvmSet cvmGet cvMatMul cvReleaseMat
GetPosFromRob	cvInitMatHeader cvCreateMat cvmSet cvmGet cvMatMul cvReleaseMat
GetRobPos	cvInitMatHeader cvCreateMat cvmSet cvMatMul cvmGet cvReleaseMat
Undistort2	cvInitMatHeader cvCreateImage

	cvUndistort2
GoodFeatureToTrackEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvGoodFeaturesToTrack cvReleaseImage
FindCornerSubPixEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvFindCornerSubPix cvSize cvTermCriteria cvReleaseImage
MomentsEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvMoments cvReleaseImage
MeasureInfo	-
HoughLine	cvCreateMemStorage cvHoughLines2 cvGetSeqElem cvReleaseImage cvReleaseMemStorage
HoughCircles	cvCreateMemStorage cvHoughCircles cvGetSeqElem cvReleaseImage cvReleaseMemStorage
DFTEx	cvCreateImage cvGetSize cvCvtColor cvCopy

	cvConvart cvZero cvMerge cvDFT cvPow cvAdd cvReleaseImage
IDFT	cvCreateImage cvMerge cvDFT cvSplit cvConvert cvReleaseImage
OpticalFlowEx	cvCreateImage cvGetSize cvCvtColor cvCopy cvCalcOpticalFlowLK cvReleaseImage
BoxPoints	cvBoxPoints
FindHomography	cvCreateMat cvmSet cvFindHomography cvmGet cvReleaseMat
QRDecode	-
OCRead	-

付録B. μVision 対応表

表 B-1 μVision21 関数実現方法一覧

Vision 関数名	機能	OpenCVプロバイダ実現方法
CAMIN	カメラからの映像を画像メモリ(処理画面)に格納します.	File::Execute の Copy コマンド
CAMMODE	カメラ映像を格納する際の機能を設定します.	未実装 (カメラでセッティングすること)
CAMLEVEL	カメラ映像の入力レベルを設定します.	未実装 (カメラ側でセッティングすること)
VISCAMOUT	カメラからの映像をモニタに表示します.	File::get_Value(ID=0～9)
VISPLNOUT	格納メモリの画像をモニタに表示します.	File::get_Value(ID<10)
VISOVERLAY	描画画面の情報をモニタに表示します.	未実装 (画像表示方法はクライアントに依存)
VISDEFTABLE	カメラ映像の取り込み, 画像出力の際のルックアップテーブルデータを設定します.	File::Execute の LUT コマンド
VISREFTABLE	ルックアップテーブルのデータを参照します.	File::Execute の GetLUT コマンド
WINDMAKE	画像処理する範囲を指定します.	一部実装(矩形のみ対応, 回転はなし) File::Execute の SetROI コマンド
WINDCLR	設定したウィンドウ情報を消去します.	一部実装(矩形のみ対応) File::Execute の ResetROI コマンド
WINDCOPY	ウィンドウのデータをコピーします.	File::Execute の GetROI コマンドで情報を取得し, SetROI コマンドで設定します.
WINDREF	ウィンドウの情報を得ます.	File::Execute の GetROI コマンド
WINDDISP	設定したウィンドウを描画します.	File::Execute の SetROI コマンドが実行されている状態で Copy コマンドを実行します.
VISSCREEN	描画する画面を指定します.	File オブジェクトの複数生成が可能なため, 出力画像はクライアントで任意に設定することが可能
VISBRIGHT	描画する際の輝度値を指定します.	File::Execute の描画系コマンドで個別

		に指定可能
VISCLS	モードで指定した画面を指定輝度で塗りつぶし(クリア)します.	File::put_Value に EMPTY を設定することで画像をクリアします. File::Execute の Rectangle コマンドで画面全体を指定し, 線の太さに-1 を指定することで画面の塗りつぶしを行います.
VISPUTP	画面に点を描画します.	File::Execute の PutColor コマンド
VISLINE	画面に直線を描画します.	File::Execute の Line2 コマンド
VISPTP	画面に 2 点間を結ぶ直線を描画します.	File::Execute の Line コマンド
VISRECT	画面に矩形を描画します.	File::Execute の Rectangle コマンド
VISCIRCLE	画面に円を描画します.	File::Execute の Circle コマンド
VISELLIPSE	画面に楕円を描画します.	File::Execute の Ellipse コマンド
VISSECT	画面に扇を描画します.	File::Execute の Sector コマンド
VISCROSS	画面に十字マークを描画します.	File::Execute の Cross コマンド
VISLOC	文字の表示位置を指定します.	File::Execute の Text コマンド
VISDEFCHAR	文字のサイズと表示方法を指定します.	File::Execute の Text コマンド
VISPRINT	画面に文字・数字を表示します.	File::Execute の Text コマンド
VISWORKPLN	処理対象の格納メモリ(処理画面)を指定します.	File::ID
VISGETP	格納メモリ(処理画面)から指定座標の輝度を取得します.	File::Execute の GetColor コマンド
VISHIST	画面のヒストグラム(輝度分布)を得ます.	File::Execute の CalcHistEx コマンド
VISREFHIST	ヒストグラム結果を読み出します.	File::Execute の CalcHistEx コマンド
VISLEVEL	ヒストグラム結果に基づき 2 値化レベルを求めます.	File::Execute の以下のコマンド AutoThreshPTile AutoThreshMode AutoThreshDiscrim
VISBINA	画面を 2 値化処理します.	File::Execute の以下のコマンド ThresholdEx Threshold2
VISBINAR	画面を 2 値化表示します.	File::Execute の以下のコマンド ThresholdEx Threshold2
VISFILTER	画像にフィルタ処理を行ないます.	File::Execute の Smooth コマンド

VISMASK	画像間演算をします.	File::Execute の Mask 系コマンド（AND, OR 等）
VISCOPY	画面をコピーします.	File::put_Value = File::get_Value
VISMEASURE	ウィンドウ内の特徴（面積, 重心, 主軸角）を計測します.	File::Execute の以下のコマンドの組み合わせ Moments, MesureInfo, ImageSize
VISPROJ	ウィンドウ内の投影データを計測します.	未実装
VISEDGE	ウィンドウ内のエッジを計測します.	未実装
VISREADQR	コードを読み取ります.	File::Execute の QRDecode コマンド
BLOB	ラベリングを実行します.	File::Execute の FindContoursEx 又は FindBlobs
BLOBMEASURE	対象ラベル番号の特徴計測を行ないます.	File::Execute の以下のコマンドの組み合わせ FindContoursEx , CopyContours , Moments , MesureInfo , ImageSizeFitEllipse , ArcLength , CheckContourCnvexity , BlobResult , BlobResults
BLOBLABEL	指定座標のラベル番号を取得します.	File::Execute の以下のコマンド ContoursNumber PointPolygonTest
BLOBCOPY	対象ラベル番号をコピーします.	File::Execute の CopyContours コマンド
SHDEFMODEL	サーチモデルの登録をします.	Put_Value
SHREFMODEL	登録モデルデータを参照します.	File::Execute の以下のコマンドの組み合わせ MomentsEx, MesureInfo, ImageSize
SHCOPYMODEL	登録モデルをコピーします.	File::Execute の Copy コマンド
SHCLRMODEL	登録モデルを消去します.	File::put_Value に EMPTY を設定することで画像をクリアします. File::Execute の Rectangle コマンドで画面全体を指定し, 線の太さに-1 を指定することで画面の塗りつぶしを行います.
SHDISPMODEL	登録モデルを画面に表示します.	Get_Value
SHMODEL	モデルをサーチします.	File::Execute の以下のコマンド MatchTemplate

		MatchTemplate2 HaarDetect MatchEx
SHDEFCORNER	コーナーサーチの条件を設定します.	File::Execute の HoughLine コマンド
SHCORNER	コーナーをサーチします.	File::Execute の HoughLine コマンド
SHDEFCIRCLE	円サーチの条件を設定します.	File::Execute の HoughCircles コマンド
SHCIRCLE	円をサーチします.	File::Execute の HoughCircles コマンド
VISGETNUM	画像処理結果を格納メモリから取得します.	Execute の返り値 Get_Value(画像の場合)
VISGETSTR	コード認識結果を取得します.	File::Execute の返り値(QRDecode コマンド)
VISPOSX	画像処理結果(X 座標)を格納メモリから取得します.	File::Execute の返り値
VISPOSY	画像処理結果(Y 座標)を格納メモリから取得します.	File::Execute の返り値
VISSTATUS	各命令の処理結果をモニタします.	File::Execute の返り値
VISREFCAL	CAL(視覚－ロボット座標変換)データを取得します.	File::Execute の CAL 系コマンド

付録C. Intel License Agreement For Open Source Computer Vision Library

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistribution's of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistribution's in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of Intel Corporation may not be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall Intel or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

All information provided related to future Intel products and plans is preliminary and subject to change at any time, without notice.