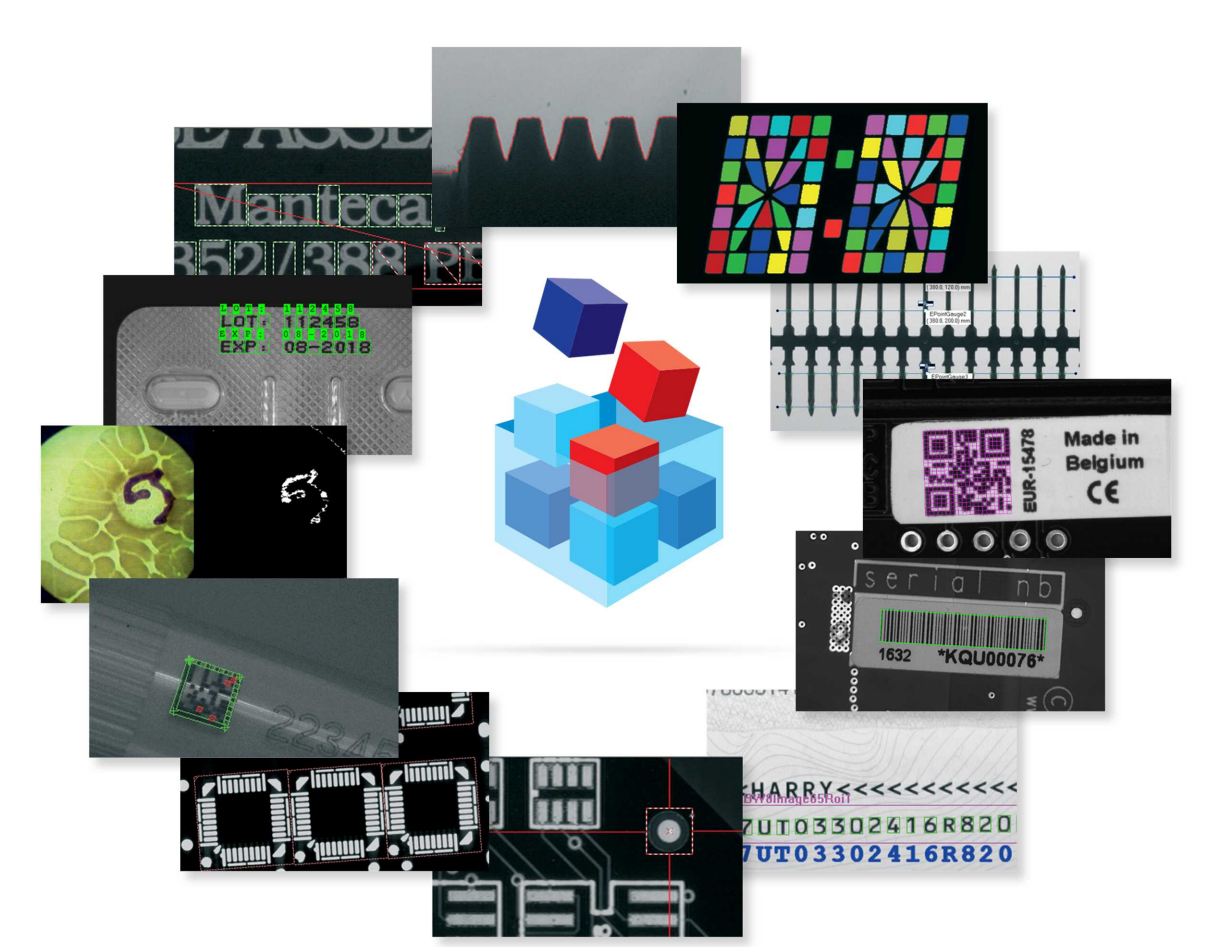
EURESYS™
Excellence in vision

ユーザーガイド

Open eVision

C++ 2.3



利用規約

本書はOpen eVision 2.3.0 (doc build 2018-01-19) とともに提供される付属ドキュメントです。

目 次

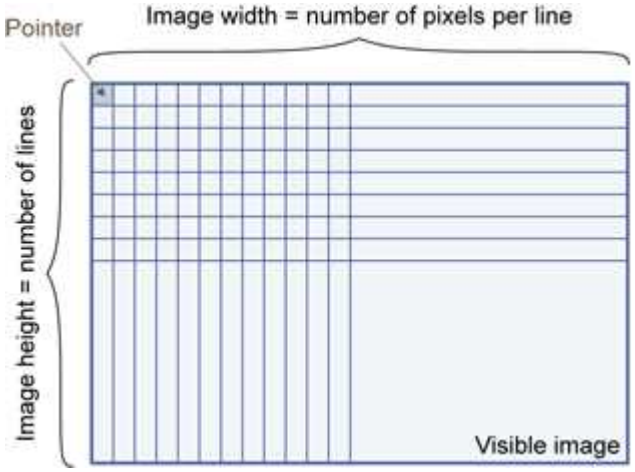
1. ピクセルコンテナおよびファイルの処 理

1.1. ピクセルコンテナの定 義

画 像

Open eVision画 像オブジェクトには、長 方 形 の画 像を表 す画 像データが含 まれます。

各 画 像オブジェクトにはポインタによってアクセスできるデータバッファがあり、ピクセル値 が行 ごとに連 続して保 存されます。



画 像の主 要 パラメータ

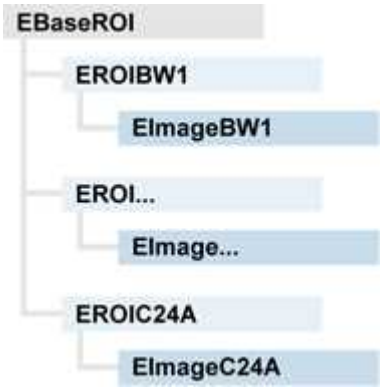
Open eVision 画 像オブジェクトには長 方 形 のピクセルアレイがあり、その特 性 はEBaseROIパラメータによって決 められます。

- Width(幅) は画 像1行 当 たりの列 (ピクセル) の数 です。
- Height(高 さ) は画 像 の行 の数 です。(最 大 幅 / 高 さは32ビット版 Open eVisionでは32,767 (2^{15}-1)、64ビット版 Open eVisionでは2,147,483,647 (2^{31}-1) です。)
- Size(サイズ) は幅 と高 さです。

Plane(平 面) パラメータには色 成 分 の数 が含 まれます。グレーレベル画 像 = 1。カラー画 像 = 3。

クラス

画 像およびROIクラスは抽 象 クラスEBaseROIから派 生 し、そのすべての属 性 を引 き継 ぎます。



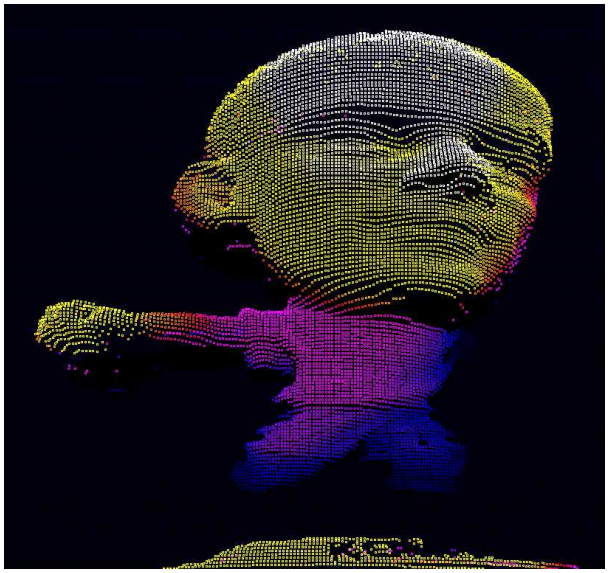
深 度 マップ

深 度 マップは、2Dグレースケール画 像を使 用して3Dオブジェクトを表 現する方 法であり、画 像 内の各ピクセルは3Dポイントを表します。

画 素の座 標は点のX座 標とY座 標を表し、画 素のグレースケール値は点のZ座 標を表します。

ポイント クラウド

ポイント クラウド (https://en.wikipedia.org/wiki/Point_cloud) は、オブジェクトの表 面 上の離 散 位 置を表す3D点の非 構 造 化 セットです。



3Dポイントクラウドは、レーザー三角測量、飛行時間または構造化照明などのさまざまな3Dスキャニング技術によって作成されます。

1.2. ピクセルコンテナの種類

画像

白黒、グレーレベル、カラーなど、ピクセル形式に応じてさまざまな画像形式がサポートされています。

Easy.GetBestMatchingImageTypeは、ディスクにある指定されたファイルに最も適合する画像形式を返します。

BW1	1ビットの白黒画像(8ピクセルが1バイトに保存される)	EImageBW1
BW8	8ビットのグレースケール画像(各ピクセルが1バイトに保存される)	EImageBW8
BW16	16ビットのグレースケール画像(各ピクセルが2バイトに保存される)	EImageBW16
BW32	32ビットのグレースケール画像(各ピクセルが4バイトに保存される)	EImageBW32
C15	15ビットのカラー画像(各ピクセルが2バイトに保存される) Microsoft® Windows RGB15カラー画像およびMultiCam RGB15形式と互換性あり。	EImageC15

C16	16ビットのカラー画像(各ピクセルが2バイトに保存される) Microsoft® Windows RGB16カラー画像およびMultiCam RGB16形式と互換性あり。	EImageC16
C24	C24画像は24ビットのカラー画像を保存(各ピクセルが3バイトに保存される) Microsoft® Windows RGB24カラー画像およびMultiCam RGB24形式と互換性あり。	EImageC24
C24A	C24A画像は32ビットのカラー画像を保存(各ピクセルが4バイトに保存される) Microsoft® Windows RGB32カラー画像およびMultiCam RGB32形式と互換性あり。	EImageC24A

深度マップ

8および16ビットの深度マップ値は、2D Open eVisionイメージと互換性のあるバッファに保存されます。

EDepth8	8ビット深度マップ(各ピクセルは整数として1バイトに保存される)	EDepthMap8
EDepth16	16ビット深度マップ(各ピクセルは整数として2バイトに保存される)	EDepthMap16
EDepth32f	32ビット深度マップ(各ピクセルは整数として4バイトに保存される)	EDepthMap32f

ポイントクラウド

ポイントクラウド	ポイント座標のセット(浮動小数点として保存)	E3DPointCloud

1.3. サポートされている画像ファイル形式

形式	説明
BMP	非圧縮画像データフォーマット(Windowsビットマップ形式)
JPEG	Joint Photographic Expert Groupが作成した非可逆データ圧縮規格で、ISO/IEC 10918-1として登録されている。圧縮により品質が低下し、完全に元のデータに戻ら

形式	説明
	ない。
JFIF	JPEG File Interchange Format
JPEG-2000	Joint Photographic Expert Groupが作成したデータ圧縮規格で、ISO/IEC 15444-1およびISO/IEC 15444-2として登録されている。Open eVisionは非可逆な圧縮形式、ファイルフォーマットおよびコードストリームのみに対応しています。 - **コードストリーム**には画像サンプルが記述されます。 - **ファイルフォーマット**には、画像解像度や色空間などのメタ情報が含まれます。
PNG	可逆のデータ圧縮方法 (Portable Network Graphics)。
Serialized	Open eVision画像オブジェクトの**シリアライゼーション**から得られるEuresys独自の画像ファイルフォーマット。
TIFF	Tag Image Fileフォーマットは現在Adobe Systemsによって管理されており、5.0または6.0 TIFF仕様向けに書き込まれた画像の処理にはLibTIFFサードパーティー製ライブラリが使用される。 ファイルの**保存**演算は可逆的で、1ビットのバイナリピクセルにはCCITT 1D圧縮、その他の種類のピクセルにはLZW圧縮が行われます。 ファイルの**読み込み**演算は、LibTIFF仕様書に列挙されている全TIFFバージョンに対応しています。

1.4. ピクセルとファイルタイプの互換性

画像変換の深度マップ

8ビットおよび16ビットの深度マップの場合、この`AsImage()`メソッドはOpen eVisionの2D処理機能で使用できる互換性のあるイメージオブジェクト(`EImageBW8`および`EImageBW16`それぞれ)を戻します。

ピクセルとファイルタイプの互換性

ピクセルのアクセス

ピクセルにアクセスするメソッドとして、`SetImagePtr`および`GetImagePtr`を使用して画像バッファのアクセスをコードに埋め込むことを推奨します。画像の構築とメモリの割り当ておよび「ピクセル値を取得する」を参照。

関数呼び出しのたびにオーバーヘッドが発生するため、以下のメソッドはあまり使用しないでください。

ダイレクトアクセス

`EROIBW8::GetPixel`および`SetPixel`メソッドは、与えられた座標のピクセル値を読み込む/書き込むためにすべての画像およびROIクラスに実装されています。画像の全ピクセルをスキャンするた

めにXおよびY座標でダブルループを実行してGetPixelまたはSetPixelをループごとに実行することも可能ですが、推奨されません。

パフォーマンス上の理由から、かなりの数のピクセルを処理する必要がある場合は、これらのアクセサを使用しないでください。そのような場合は、 GetBufferPtr()を使用する内部バッファポインタを取得してポインタを反復することをお勧めします。

BW8ピクセルへの C++ クイックアクセス

BW8画像では、EBW8PixelAccessor::GetPixelまたはSetPixelは、直接EROIBW8::GetPixelまたはSetPixelを呼び出すよりも早く呼び出せます。

サポートされている構造

- EBW1、EBW8、EBW32
- EC15 (*)、EC16 (*)、EC24 (*)
- EC24A
- EDepth8、EDepth16、EDepth32f、

(*) これらのフォーマットはRGB15(5-5-5 ビットパッキング) 、RGB16(5-6-5 ビットパッキング) およびRGB32(RGB + アルファチャネル) に対応していますが、処理を行う前にEasyImage::Convertを使ってEC24に / から変換する必要があります。

> **注記:** *eVision 6.5 以前のバージョンでの移行はスムーズです: 画像ピクセルの種類は整数タイプのtypedefを用いて定義されており、ピクセル値は符号のない数値として扱われていた上に、前のタイプへ / からの暗黙的な変換が含まれていました。*

読み込み / 保存演算中のピクセル / ファイル形式の互換性

形式	BMP	JPEG	JPEG2000	PNG	TIFF	Serialized
BW1	OK	N/A	N/A	OK	OK	OK
BW8	OK	OK	OK	OK	OK	OK
BW16	N/A	N/A	OK	OK	OK (***)	OK
BW32	N/A	N/A	N/A	N/A	OK (***)	OK
C15	OK	OK (**)	OK (**)	OK (**)	OK (**)	OK
C16	OK	OK (**)	OK (**)	OK (**)	OK (**)	OK
C24	OK	OK	OK	OK	OK (**)	OK
C24A	OK	N/A	N/A	OK	N/A	OK
Depth8	OK	OK	OK	OK	OK	OK
Depth16	N/A	N/A	OK	OK	OK (***)	OK
Depth32f	N/A	N/A	N/A	N/A	N/A	OK

N/A: 対応していません。この組み合わせを用いると例外が発生します。

OK: データ損失がなく、画像の完全性が保持されます(JPEGおよびJPEG2000、非可逆圧縮を除く) 。

(**) C15およびC16フォーマットは保 存 演 算 中 に自 動 的 にC24に変 換 されます。

(***) BW16およびBW32はベースラインTIFFリーダーに対 応 していません。

1.5. 色 の形 式

EISH：輝 度 、彩 度 、色 相 のカラーシステム

ELAB：CIE 明 度 、a*、b* のカラーシステム

ELCH：明 度 、クロマ、色 相 のカラーシステム

ELSH：明 度 、彩 度 、色 相 のカラーシステム

ELUV：CIE 明 度 、u*、v* のカラーシステム

ERGB：NTSC/PAL/SMPTE 赤 緑 青 のカラーシステム

EVSH：値 、彩 度 、色 相 のカラーシステム

EXYZ：CIE XYZ カラーシステム

EYIQ：CCIRルマ、同 位 相 、直 角 位 相 のカラーシステム

EYSH：CCIRルマ、彩 度 、色 相 のカラーシステム

EYUV：CCIRルマ、Uクロマ、Vクロマのカラーシステム

2. ピクセルコンテナおよびファイルの操 作

2.1. ピクセルコンテナファイルの保 存

イメージおよび深 度 マップ

Saveメソッドは、次 の2つの引 数を使 用して、イメージまたは深 度 マップオブジェクトのデータをファイルに保 存します。

- パス: パス、ファイル名 、ファイル名 の拡 張 子
- 画 像ファイル形 式 。ない場 合 、ファイル名 の拡 張 子 が使 用されます。

画 像 が65,536(幅 または高 さ) より大 きい場 合 は、Open eVision独 自 の形 式 で保 存してください。

Saveは以 下 の場 合 に例 外 を投 げます:

- 指 定された画 像ファイル形 式 が画 像ピクセルの種 類 と互 換 性 がない場 合
- 自 動ファイル形 式 選 択メソッドとファイル名 の拡 張 子 がサポートされていない場 合

16ビット深 度 マップを保 存 すると、固 定 小 数 点 精 度 が失 われ、ピクセルは16ビット整 数 とみなされます。

画 像ファイル形 式 の引 数

引 数	画 像ファイル形 式
EImageFileType_Auto(*)	ファイル名 の拡 張 子 から自 動 的 に指 定されます。下 記を参 照。
EImageFileType_Euresys	Open eVision Serialization
EImageFileType_Bmp	Windows ビットマップ - BMP
EImageFileType_Jpeg	JPEG File Interchange Format - JFIF
EImageFileType_Jpeg2000	JPEG 2000 ファイルフォーマット / コードストリーム JPEG2000
EImageFileType_Png	Portable Network Graphics - PNG
EImageFileType_Tiff	Tagged Image File Format - TIFF

(*) デフォルト値

引数がImageFileType_Autoの場合 / ない場合に指定される画像ファイル形式

ファイル名拡張子 (*)	自動的に指定される画像ファイル形式
BMP	Windowsビットマップフォーマット
JPEG、JPG	JPEG File Interchange Format - JFIF
JP2	JPEG 2000ファイルフォーマット
J2K、J2C	JPEG 2000コードストリーム
PNG	Portable Network Graphics
TIFF、TIF	Tagged Image File Format

(*) 大文字と小文字の区別なし

JPEGおよびJPEG2000非可逆圧縮で保存する

SaveJpegおよびSaveJpeg2Kは、圧縮された画像を保存するときの圧縮品質を指定します。これには2つの引数があります:

- パス: パス、ファイル名、ファイル名の拡張子を含む文字列
- 画像ファイルの圧縮品質、[0: 100] 範囲内の整数の値。
  SaveJpegはJPEG File Interchange Format – JFIF形式で画像データを保存します。
  SaveJpeg2KはJPEG 2000ファイル形式で画像データを保存します。

JPEG圧縮値

JPEG 圧縮	説明
JPEG_DEFAULT_QUALITY (-1)	デフォルトの品質 (*)
100	優れた画像品質、最も低い圧縮率
75	良好な画像品質 (*)
50	標準的な画像品質
25	平均的な画像品質
10	低い画像品質

(*) デフォルトの品質は良好な画像品質 (75) に相当します。

JPEG 2000圧縮品質値の目安

JPEG 2000圧縮	説明
-1	デフォルトの品質 (*)
1	最高の画像品質、最も低い圧縮率

JPEG 2000圧縮	説明
16	良好な画像品質 (*) (比率 16:1)
512	最低の画像品質、最も高い圧縮率

(*) デフォルトの品質は良好な画像品質 (比率 16:1) に相当します。

ポイントクラウド

- `Save`方法を使用して、ポイントクラウドをOpen eVision独自のファイル形式で保存します。
- `SavePCD`方法を使用して、ポイントクラウドをPCL(ポイントクラウドライブラリ) などの他のソフトウェアと互換性のあるファイルに保存します。

2.2. ピクセルコンテナファイルの読み込み

イメージおよび深度マップ

- `Load`メソッドを使用して画像データを画像オブジェクトに読み込みます:
  - このメソッドには引数が1つだけあります: **パス:** パス、ファイル名、ファイル名の拡張子。
  - ファイルの種類はファイルフォーマットによって決まります。
  - 目的画像はディスク上の画像のサイズに応じて自動的にサイズ調整されます。
- `Load`メソッドは以下の場合に例外を投げます:
  - ファイル形式の識別に失敗したとき
  - ファイル形式と画像オブジェクトのピクセル形式に互換性がない場合

Open eVision 1.1以降のシリアライズされた画像ファイルは、Open eVision前バージョンのシリアライズされた画像ファイルと互換性がありません。

深度マップにBW16画像(整数値付き) を読み込むと、深度マップ(デフォルトでは0) に設定された固定小数点精度は変更されずにそのまま使用されます。

ポイントクラウド

- `Load`方法を使用して、ポイントクラウドをOpen eVision独自のファイル形式で保存します。
- `LoadPCD`方法を使用して、ポイントクラウドをPCL(ポイントクラウドライブラリ) などの他のソフトウェアと互換性のあるファイルに保存します。

2.3. メモリ割り当て

画像を構築する際、内部または外部メモリの割り当てが行われます。

内部メモリ割り当て

画像オブジェクトにより、バッファが動的に割り当てられたり、割り当てが解除されます。メモリ管理は透過的に行われます。
画像サイズが変更されると、再割り当てが行われます。
画像オブジェクトが破棄されると、バッファの割り当てが解除されます。
内部メモリ割り当ての画像を宣言するには:

1. 画像オブジェクト(例えば`EImageBW8`)を構築し、その際、幅および高さの引数を使用するか、あるいは`SetSize`関数を使用します。
2. 任意のピクセルにアクセスします。そのために複数の関数があります。`GetImagePtr`は、特定の座標でのピクセルの最初のバイトへのポインタを返します。

外部メモリ割り当て

ユーザーはバッファの割り当てを制御するか、メモリバッファにある第三者製の画像をOpen eVision画像にリンクします。
画像サイズおよびバッファアドレスを指定する必要があります。
画像オブジェクトが破棄されるとき、バッファは影響を受けません。
外部メモリ割り当ての画像を宣言するには:

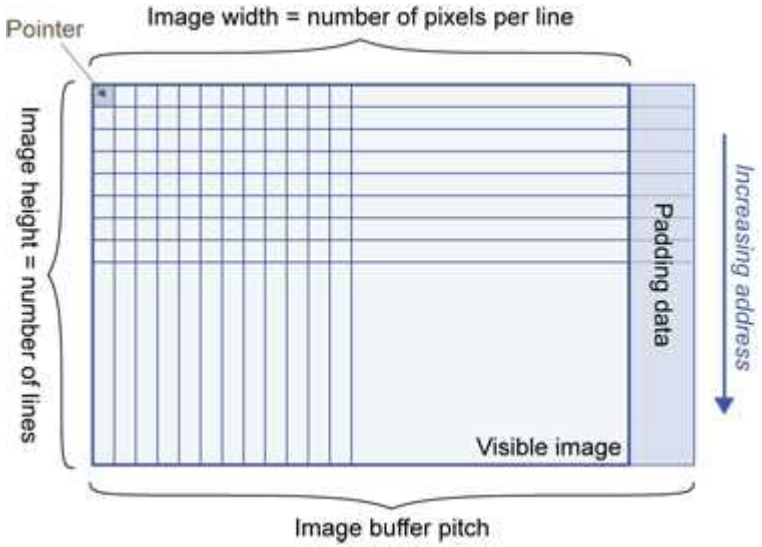
1. 画像オブジェクト(例えば`EImageBW8`)を宣言します。
2. 適切なサイズおよび配列のバッファを作成します(画像バッファを参照)。
3. `SetSize`関数を使って画像サイズを設定します。
4. `GetImagePtr`でバッファにアクセスします。「ピクセル値を取得する」を参照。

2.4. 画像および深度マップバッファ

画像と深度マップのピクセルは、最上行から最下行へ、左から右へ、関連付けられたバッファにWindowsビットマップ形式(トップダウン DIB[1])で継続的に保存されます。

バッファアドレスはバッファの開始アドレスのポインタで、画像左上の角のピクセルに当たります。

[1]デバイス独立ビットマップ



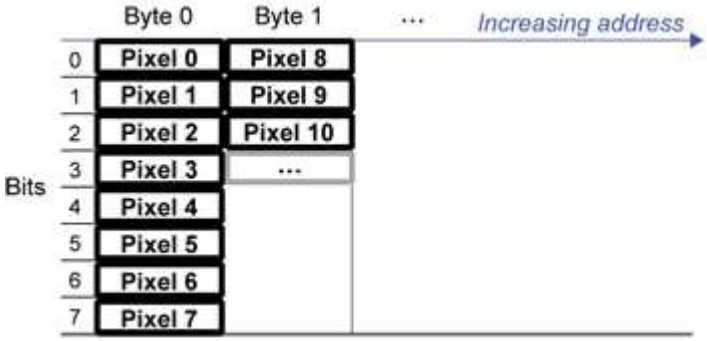
画像バッファのピッチ

- 配列は4バイトの倍数になります。
- Open eVision 1.2のデフォルトの前方ピッチは、性能上の理由から32バイトとなっています(Open eVision 1.1.5 では 8 バイトでした) 。

メモリレイアウト

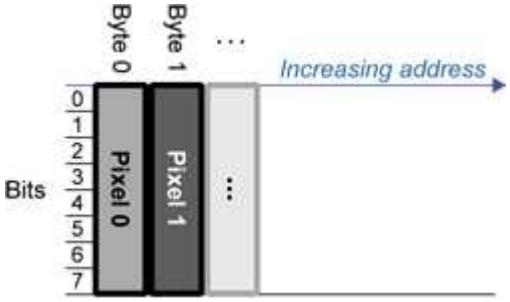
- `EImageBW1`では1バイトに8ピクセルが保存されます。

  BW1画像バッファの最初の2ピクセルのメモリレイアウト例：



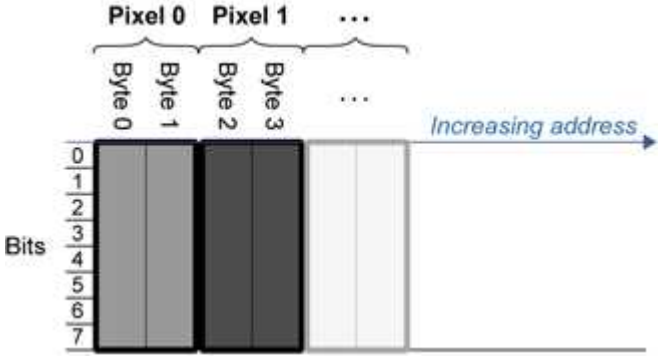
- `EImageBW8`と`EDepthMap8`では各ピクセルが1バイトで保存されます。

  BW8画像バッファの最初のピクセルのメモリレイアウト例：



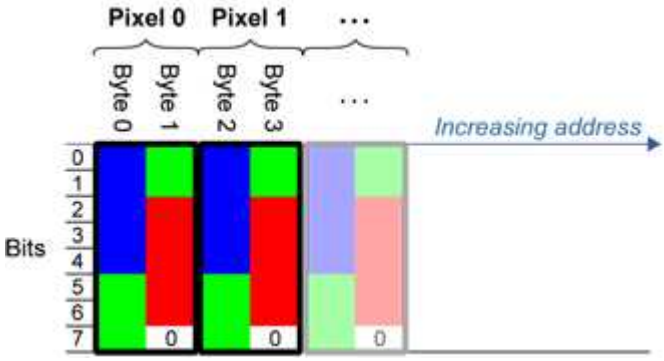
- `EImageBW16` では各ピクセルが16ビットワード(2バイト) に保存されます。

BW16 画像バッファの最初のピクセルのメモリレイアウト例：



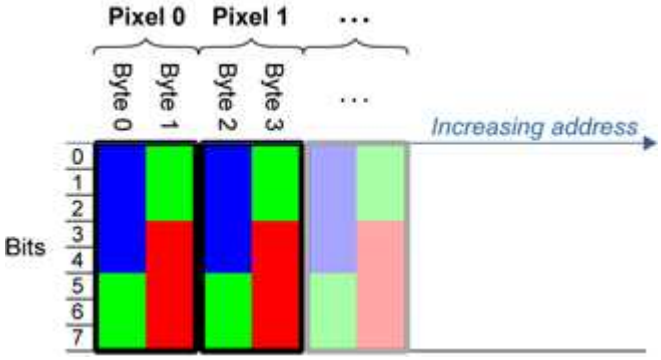
- EImageC15では各ピクセルが2バイトに保存されます。各色成分は 5 ビットにコード化されます。16番目のビットは使用されません。

C15 画像バッファの最初のピクセルのメモリレイアウト例：



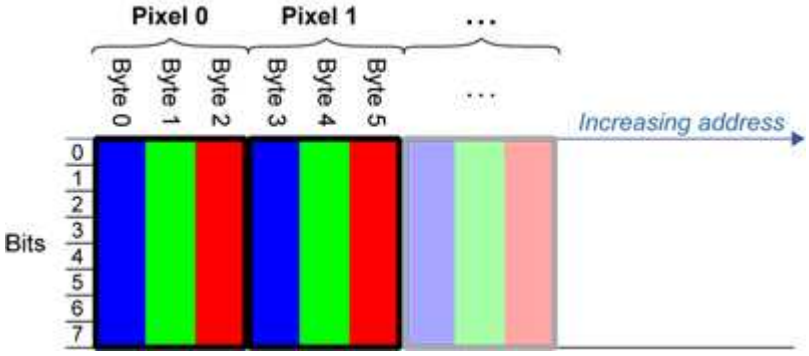
- EImageC16では各ピクセルが2バイトに保存されます。1番目と3番目の色成分が5ビットにコード化されます。
  2番目の色成分は6ビットにコード化されます。

C16 画像バッファの最初のピクセルのメモリレイアウト例：



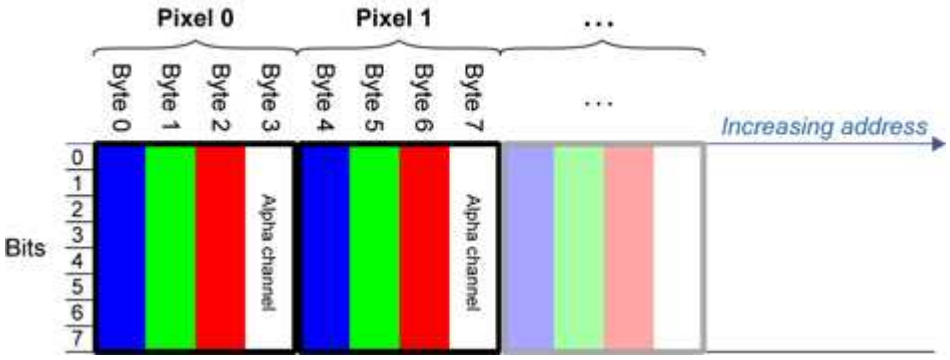
- EDepthMap16は固定小数点形式を使用して各ピクセルを2バイトで保存します。
- EImageC24では各ピクセルが3バイトに保存されます。各色成分は 8 ビットにコード化されます。

C24 画像バッファの最初のピクセルのメモリレイアウト例：



- `EImageC24A`では各ピクセルが4バイトに保存されます。各色成分は 8 ビットにコード化されます。
  アルファチャネルも 8 ビットにコード化されます。

  C24A 画像バッファの最初のピクセルのメモリレイアウト例：



- `EDepthMap32f`は浮動小数点形式を使用して各ピクセルを4バイトで保存します。

2.5. 画像の描画とオーバーレイ

- 描画ではWindows GDI[1]システムコールが使用されます。
  MFC[2]アプリケーションは、通常、描画のために`OnDraw`イベントハンドラを使用します。このとき、デバイスコンテキストのポインタを使用できます。
  Borland/CodeGearのOWLまたはVCLでは、**Paint**イベントハンドラが使用されます。
- 256色表示モードのカラーパレットにより最適なレンダリングが行われます。グレーレベル画像は LUT[3] を用いて改善できます(ヒストグラムの引き伸ばしまたは疑似カラー表示)。
- 画像は水平方向および垂直方向に個別にズームできます。
- `DrawFrameWithCurrentPen`メソッドによりフレームが描画されます。

---

[1]Graphics Device Interface

[2]Microsoft Foundation Class

[3]ルックアップテーブル

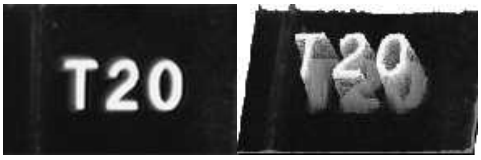
- **非破壊オーバーレイ**の描画演算では、画像の内容が変わることはありません(`MoveTo/LineTo` など) 。
- **破壊オーバーレイ**の描画演算では、画像内部に描画が行われるため画像の内容が変わります(`Easy::OpenImageGraphicContext` など) 。 グレーレベル [カラー] 画像はグレーレベル [カラー] オーバーレイにのみ対応しています。

2.6. 2D画像の3Dレンダリング

これらの画像はX軸で回転した後にY軸で回転したものです。

グレー3Dレンダリング

`Easy::Render3D`は3次元レンダリングの準備をします。このとき、グレーレベル値が高さになります。
3方向(X = 幅、Y = 高さ、Z = 深さ) の倍率を指定できます。
レンダリングされた画像は独立したドットとして表示され、そのサイズを調整することで表面の透明性を調整できます。

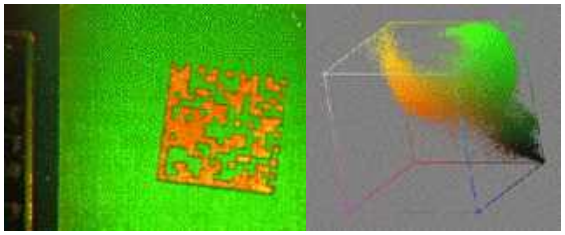


3Dレンダリング

カラーヒストグラムの3Dレンダリング

`Easy::RenderColorHistogram`はカラー画像ヒストグラムの3次元レンダリングを準備します。
RGB値のクラスタと分散を表すため、ピクセルはRGB空間(XY平面ではない) で描画されます。
この関数で他のカラーシステムのピクセルを処理することもできますが(EasyColorにより変換) 、通常の色でピクセルを表示するにはRGB原画像が必要になります。

3方向すべて(X = 幅、Y = 高さ、Z = 深さ) の倍率を指定できます。

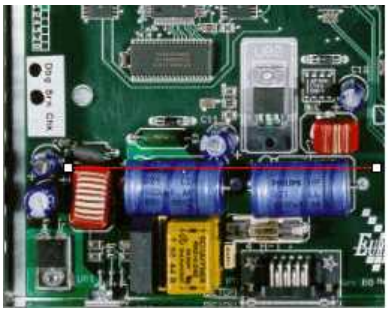


カラーヒストグラムのレンダリング

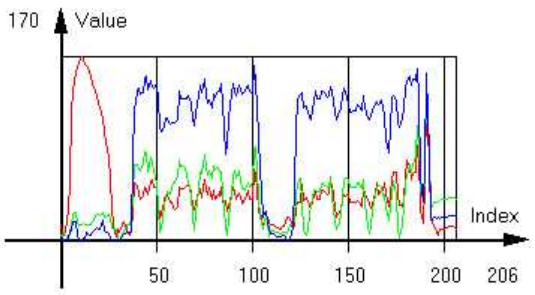
2.7. ベクトルの種類と主な特性

ベクトルとは1次元のピクセル配列です(画像のプロファイルまたは輪郭から取得)。

EVectorは全ベクトルの基本クラスです。これには、主に要素カウントおよびシリアライゼーションのための、型が特定されていないすべてのメソッドが含まれています。



C24画像のプロファイル



プロファイルに沿ったRGB値のプロット

Index	Red	Green	Blue
0	15	5	3
1	7	4	0
2	5	8	0
3	9	5	0
4	29	1	0
5	55	6	9
6	120	15	9
7	139	24	17
8	157	26	18
9	161	17	6
10	165	13	0
11	170	14	1

RGB値の配列 (EC24Vector)

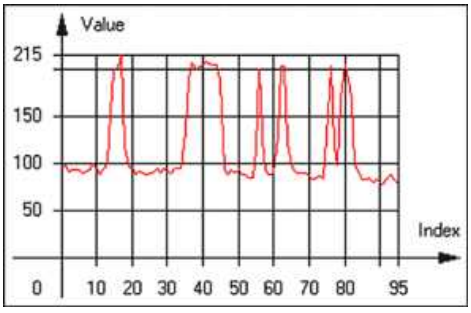
ベクトルによって各要素の配列が管理されます。メモリの割り当ては透過的に行われますので、ベクトルのサイズは動的に変更されます。関数がベクトルを使用するときは、関数の必要性に合わせてベクトルの種類、サイズ、構造が自動的に調整されます。

ベクトルは極めて簡単に使用できます:

1. 必要に応じてコンストラクタおよび事前割り当て要素を使って、**適切な種類のベクトルを作成します**。

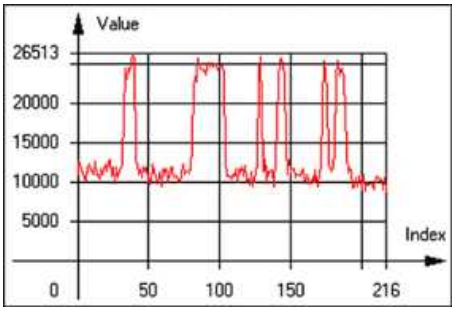
ベクトルの種類

- EBW8Vector: 多くの場合、画像プロファイルから抽出される、グレーレベル値のシーケンス(EasyImage::Lut、EasyImage::SetupEqualize、EasyImage::ImageToLineSegment、EasyImage::LineSegmentToImage、EasyImage::ProfileDerivativeなどで使用される)。



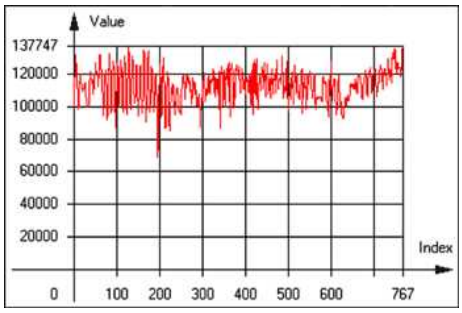
EBW8Vectorのグラフ表示(Drawメソッドを参照)

- EBW16Vector: 拡張された範囲(16ビット)を用いたグレーレベル値のシーケンス。主に中間計算に使用されます。



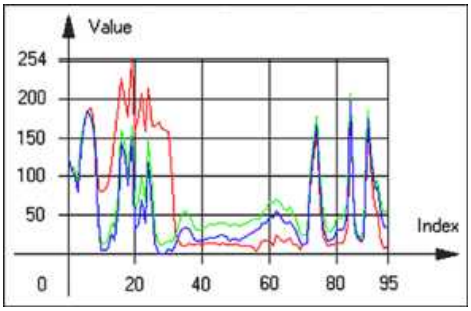
EBW16Vectorのグラフ表示

- EBW32Vector: 拡張された範囲(32ビット)を用いたグレーレベル値のシーケンス。主に中間計算に使用されます。
  (EasyImage::ProjectOnARow、EasyImage::ProjectOnAColumnなどで使用される)



EBW32Vectorのグラフ表示

- EC24Vector: 多くの場合、画像プロファイルから抽出される、カラーピクセル値のシーケンス
  (EasyImage::ImageToLineSegment、EasyImage::LineSegmentToImage、EasyImage::ProfileDerivativeなどで使用される)。

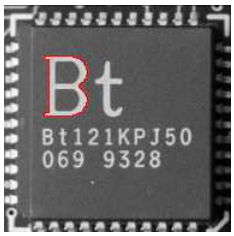


EC24Vectorのグラフ表示

- EBW8PathVector: 画像プロファイルまたは輪郭からピクセル座標とともに抽出されたグレーレベルピクセル値のシーケンス
  (EasyImage::ImageToPath、EasyImage::PathToImageなどで使用される)。

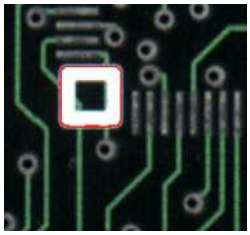
EBW8PathVectorのグラフ表示(Drawメソッドを参照)

- EBW16PathVector: 画像プロファイルまたは輪郭からピクセル座標とともに抽出されたグレーレベルピクセル値のシーケンス
  (EasyImage::ImageToPath、EasyImage::PathToImageなどで使用される)。



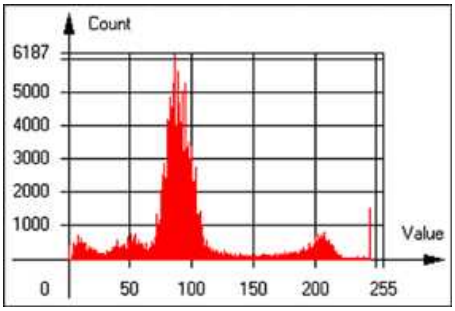
EBW16PathVectorのグラフ表示(Drawメソッドを参照)

- EC24PathVector: 画像プロファイルまたは輪郭からピクセル座標とともに抽出されたカラー値のシーケンス
  (EasyImage::ImageToPath、EasyImage::PathToImageなどで使用される)。



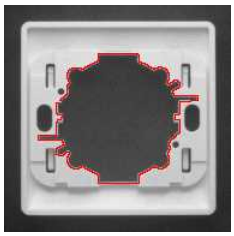
EC24PathVectorのグラフ表示(Drawメソッドを参照)

- EBWHistogramVector: BW8またはBW16画像に含まれる各ピクセルの頻度のシーケンス
  (EasyImage::IsodataThreshold、EasyImage::Histogram、
  EasyImage::AnalyseHistogram、EasyImage::SetupEqualizeなどで使用される)。



EBWHistogramVectorのグラフ表示(Drawメソッドを参照)

- EPathVector:ピクセル座標のシーケンス。ピクセルは連続していなければなりません(EasyImage::PathToImageおよびEasyImage::Contourで使用される)。



EPathVectorのグラフ表示(Drawメソッドを参照)

- EPeakVector:画像プロファイルで検出されたピーク(EasyImage::GetProfilePeaksで使用される)。
- EColorVector:色の記述(EasyColor::ClassAveragesおよびEasyColor::ClassVariancesで使用される)。

2. **ベクトルに値を設定します**。まず、EVector::Emptyメンバを使って空にして、その後でEC24Vector::AddElementメンバを呼び出して要素を同時に追加します。インデックスを使用してどの要素にもアクセスできます。
3. 読み取りまたは書き込みのために**ベクトル要素にアクセスします**。例えばEC24Vector::operator[]などの括弧演算子を使用します。
4. **現在の要素の数を確認します**。EVector::NumElementsメンバを使用します。
5. **ベクトルを描画します**。
ピクセルベクトルは要素インデックスの関数として要素値をプロットしたものであるため、グラフ表示はベクトルの種類に応じて異なります。ベクトルはウィンドウ内に描画できます。可読性を確保するため、描画は無彩色の背景に行ってください。
描画は、希望のウィンドウに関連付けられたデバイスコンテキスト内で行われます。デフォルトでは曲線は青色、注釈は黒で描画されます。パラメータgraphicContext、width、height、origin、origin、color0、color1、color2を定義できます。
EC24Vectorでは1本ではなく3本の曲線が描画され、それぞれが各色成分に相当します。デフォルトでは、赤、青、緑のペンが使用されます。

2.8. ROIの主な特性

ROIは、幅、高さ、**原点のx座標とy座標によって定義されます**。
原点は親画像またはROIの左上の角を基準にして指定されます。
ROI全体がその親画像内に含まれていなければなりません。
`OrgX`および`Width`が8の倍数である場合、BW1 ROIの処理 / 解析時間が短縮されます。

保存と読み込み

ROIは独立した画像として保存したり読み込むことができ、1枚の完全な画像であるかのように使用できます。ROIには**メモリの割り当てが一切行われず**、親画像の一部が複製されることはありません。ROIからは親画像の画像データにアクセスされます。

新しいファイルの画像サイズは、それに読み込まれるROIのサイズと一致しなければなりません。ROI周囲の画像は変化しません。

ROIクラス

Open eVision ROIは抽象クラス`EBaseROI`のパラメータを継承します。

ピクセル形式に応じてさまざまな種類のROIがあります。そして、それぞれ対応する画像形式と同じ特性を持っています。

- `EROIBW1`
- `EROIBW8`
- `EROIBW16`
- `EROIBW32`
- `EROIC15`
- `EROIC16`
- `EROIC24`
- `EROIC24A`

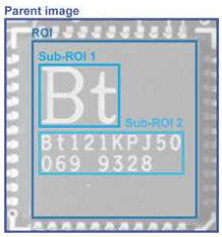
アタッチメント

ROIは、親に設定されているパラメータとともに親(画像 / ROI)にアタッチされます。これらのリンクはダングリングポインタを避けて透過的に更新されます。
通常の画像を別の画像やROIにアタッチすることはできません。

ネスティング

SetおよびGet関数は、ROIの幅、高さおよび原点の位置を、その近接または一番上の親画像を基準にして変更または照会します。

1枚の画像が任意の数のROIに対応しており、ROIは階層状にネストされます。ROIを移動すると、それに合わせて埋め込まれているROIも移動します。画像 / ROIクラスでは、画像に関連付けられているROIの階層を横断するためのメソッドがいくつかあります。



ネストされた ROI: 2つのサブROIがROIにアタッチされており、そのROI自体も親画像にアタッチされている

クロッピング

`CropToImage`は一部が画像の外側にあるROIをクロップします。サイズを変更したROIが拡大されることはありません。
何らかの関数が親の外側を超える境界を持つROIを使用しようとすると例外が投げられます。

> **注記:** *(Open eVision 1.0.1以前では、ROIが画像の外側に位置付けられるとROIの大きさが少し調整されたり位置が変更されたり、時折拡大されたりしていました。ROIの境界が親の外側になると、ROIは親の境界内に収まるように大きさが少し調整されていました。)*

サイズ変更と移動

- ROIは2つの関数とハンドルのドラッグによって簡単にサイズ変更したり位置決めできます。
  - `EBaseROI::Drag`はカーソルの移動に伴ってROI座標を調整します。
  - `EBaseROI::HitTest`はハンドルのドラッグによってカーソルが配置されているかどうかを知らせます。ハンドルが分かると、`OnSetCursor` MFCイベントハンドラによってカーソルの形状が変更することがあります。ドラッグ中に呼び出された場合、HitTestは予測不能です。
    HitTestはOnSetCursor MFCイベントハンドラでカーソルの形状を変えるために使用することもできますし、ドラッグ操作の前であれば、OnLButtonDown
    (またはBorland/CodeGearのOWLのEvSetCursorおよびEvLButtonDown)
    (またはBorland/CodeGearのVCLのFormMouseMoveおよびFormMouseDown) のように使用できます。
    VB6では、MouseDown、MouseMove、MouseUpイベントは現在のカーソル位置をピクセルではなく、twipで戻すため、変換は必須です。

2.9. フレキシブルマスク

ROI対フレキシブルマスク

ROIとマスクは処理の対象を画像の一部に制限します:

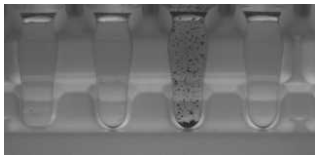
- 「ROIの主な特性」前ページのはOpen eVisionの全機能に適用されます。関心領域を使用すると、ピクセル数が減少するため処理速度が高まります。Open eVisionは階層状にネストされた長方形ROIをサポートしています。

- 繋がっていないROIや長方形でない形状を処理する場合は、フレキシブルマスクが推奨されます。これはいくつかのEasyObjectおよびEasyImageライブラリ関数でもサポートされています。

フレキシブルマスク

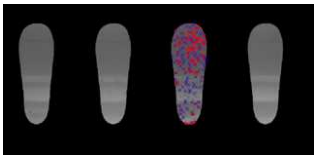
フレキシブルマスクは、ソース画像と同じ高さおよび幅を持つBW8画像です。処理が行われる領域と無視する領域(処理の際に考慮されない領域)の形状が含まれています:

- フレキシブルマスクの中で0の値を持つ全ピクセルが無視される領域として定義されます。
- フレキシブルマスクの中で0以外の値を持つ全ピクセルが処理の対象となる領域として定義されます。



ソース画像

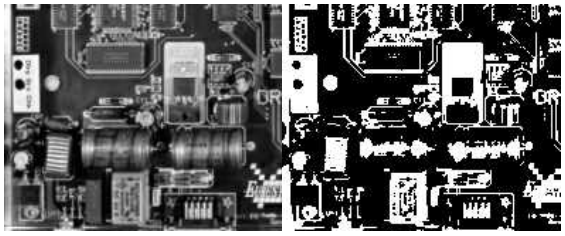
関連付けられたマスク



処理後のマスクされた画像

フレキシブルマスクは、BW8画像を出力する任意のアプリケーションによって、またはEasyObjectおよびEasyImageのいくつかの関数を使用して生成できます。

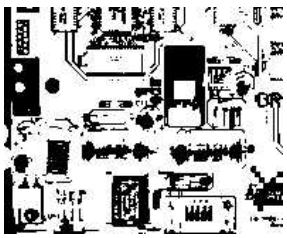
EasyImageのフレキシブルマスク



ソース画像(左)とマスク変数(右)

Easyimageでフレキシブルマスクを使用する簡単な手順

1. **入力マスクを引数としてとる関数をEasyImageから呼び出します**。例えば、白のレイヤのピクセルの平均値を計算した後で黒のレイヤの平均値を評価します。
2. **結果を表示します。**



結果画像

フレキシブルマスクをサポートしているEasyImageの関数

- EImageEncoder::Encodeにはフレキシブルマスクの引数があり、BW1、BW8、BW16およびC24ソース画像に使用できます。
- AutoThreshold
- Histogram(HistogramThreshold関数にはマスク引数のオーバーロードなし)
- RmsNoise、SignalNoiseRatio
- オーバーレイ(BW8ソース画像にはマスク引数のオーバーロードなし)
- ProjectOnAColumn、ProjectOnARow(ベクトル投影)
- ImageToLineSegment、ImageToPath(ベクトルプロファイル)

EasyObjectのフレキシブルマスク

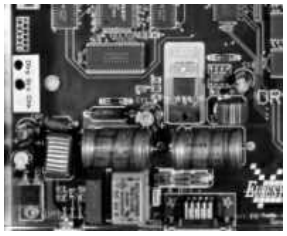
フレキシブルマスクは、BW8画像を出力する任意のアプリケーションによって、またはOpen eVision画像処理関数を使用して生成できます。
EasyObjectでは、ブロブ解析を複雑な / つながっていない形状の画像領域に制限するためにフレキシブルマスクを使用できます。

関心オブジェクトが画像の他の領域と同じグレーレベルである場合、フレキシブルマスクとEncode関数を用いて「保持」と「無視」を定義できます。

フレキシブルマスクは、ソース画像と同じ高さおよび幅を持つBW8画像です。

- フレキシブルマスクのピクセル値0のピクセルはソース画像のピクセルをマスクし、エンコードされた画像には現れません。
- フレキシブルマスクのそれ以外の値を持つピクセルはエンコードされます。

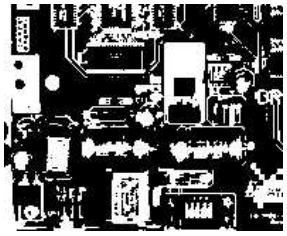
フレキシブルマスクを作成するEasyObject関数



ソース画像

1) ECodedImage2::RenderMask: エンコードされた画像のレイヤから

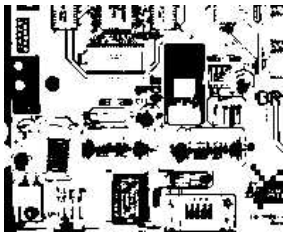
1. フレキシブルマスクをエンコードおよび抽出するには、まず、ソース画像からコード化された画像を構築します。
2. セグメンテーションメソッドを選択します(上の画像にはデフォルトのメソッドGrayscaleSingleThresholdが適しています) 。
3. エンコードするコード化された画像のレイヤを選択します(最小剰余閾値を用いて白と黒のレイヤ) 。
4. mask.SetSize(sourceImage.GetWidth(), sourceImage.GetHeight())を用いてマスクサイズを希望する大きさに調整します。
5. フレキシブルマスクを引数としてECodedImage2::RenderMaskに使用します。



フレキシブルマスクとして使用可能なBW8結果画像

2) ECodedElement::RenderMask: ブロブやホールから

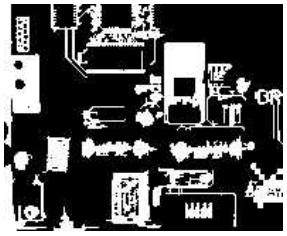
1. 関心があるコード化された要素を選択します。
2. `ECodedElement::RenderMask`を用いて、コード化された画像で選択したコード化された要素から、マスクを抽出するループを作成します。
3. オプションとして、これらの選択されたそれぞれのコード化要素から特徴値を計算します。



フレキシブルマスクとして使用可能なBW8結果画像

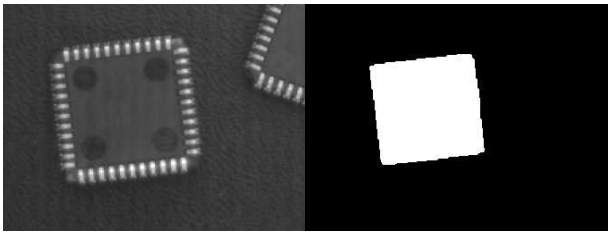
3) EObjectSelection::RenderMask: 選択したブロブから

例えば、`EObjectSelection::RenderMask`はノイズによって生じた小さいオブジェクトを破棄できます。



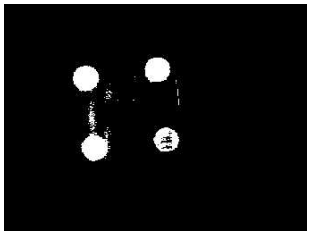
フレキシブルマスクとして使用可能なBW8結果画像

例：EasyObjectによってエンコードされた領域を制限する



4つの円を検出（左）フレキシブルマスクによって中央のチップを分離（右）

1. 新しいECodedImage2オブジェクトを宣言します。
2. セットアップ変数：最初にソース画像とフレキシブルマスクを宣言し、その後でそれらを読み込みます。
3. EImageEncoderオブジェクトを宣言し、場合に応じて適切なセグメンタを選択します。セグメンタをセットアップし、エンコードする適切なレイヤを選択します。
4. ソース画像をエンコードします。フレキシブルマスクの領域のみを持つレイヤのエンコードはスムーズに行われます。
   グレースケールの単閾値セグメンタによって黒のレイヤで適切に円がセグメント化されたことが分かります：



5. コード化された画像のすべてのオブジェクトを選択します。
6. 大きさが小さすぎるオブジェクトをフィルタ処理で除外し、関心オブジェクトを選択します。
7. 選択された特徴を表示するため、選択されたオブジェクトで反復してブロブの特徴を表示します。

2.10. プロファイル

プロファイルのサンプリング

プロファイルとは一連のピクセル値で、画像の行 / パス / 輪郭に沿ってサンプリングされます。

- EasyImage::ImageToLineSegmentは、与えられたラインセグメント（任意の方向で全体が画像内に含まれる）に沿ってピクセル値をベクトルにコピーします。ベクトルの長さは自動的に調整されます。この関数はフレキシブルマスクに対応しています。

- パスとは、ベクトル内に保存されている一連のピクセル座標です。
  EasyImage::ImageToPathは対応するピクセル値をベクトルにコピーします。この関数はフレキシブルマスクに対応しています。
- **輪郭**とは、オブジェクトの境界を形成する、閉じた / 閉じて(つながって)いないパスのことです。
  EasyImage::Contourはオブジェクトの輪郭をたどり、その構成ピクセルをプロファイルベクトル内に保存します。

プロファイルの解析

プロファイルの処理はピークや遷移を検出するために行われます:

- 遷移とはオブジェクトのエッジに相当します(黒から白、または白から黒)。遷移を検出するには、信号の一次**導関数**をとり(遷移(エッジ)をピークに変換)、その中でピークを探します。
  EasyImage::ProfileDerivative はグレーレベル画像から抽出された一次導関数を計算します。
  EBW8データタイプは符号なしの値のみをハンドルするため、導関数は128を境に切り替わります。128未満 [以上] の値は負 [正] の導関数(下り [上り] 勾配)に相当します。
- ピークとは、与えられた閾値の上 [または下] にある信号の一部分で、信号の最大値または最小値です。これは黒 / 白のライン、または細い形状の交差位置であることがあります。これは次によって定義されます:
  - 振幅:閾値と最大 [または最小] 信号値の差。
  - 面積:与えられた閾値における水平線と信号曲線にはさまれた範囲。

  EasyImage::GetProfilePeaksはグレーレベルのプロファイルで最大ピークと最小ピークを検出します。ノイズによる偽のピークを排除するため、2つの選択基準が用いられます。結果はピークのベクトルに保存されます。

プロファイルを画像に挿入する

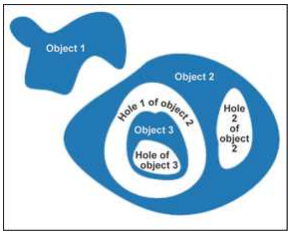
EasyImage::LineSegmentToImageは、ベクトルまたは定数から、与えられたラインセグメント(任意の方向で全体が画像内に含まれる)のピクセルにピクセル値をコピーします。

EasyImage::PathToImageは、ベクトルまたは定数から、与えられたパスのピクセルにピクセル値をコピーします。

3. 検査ツール

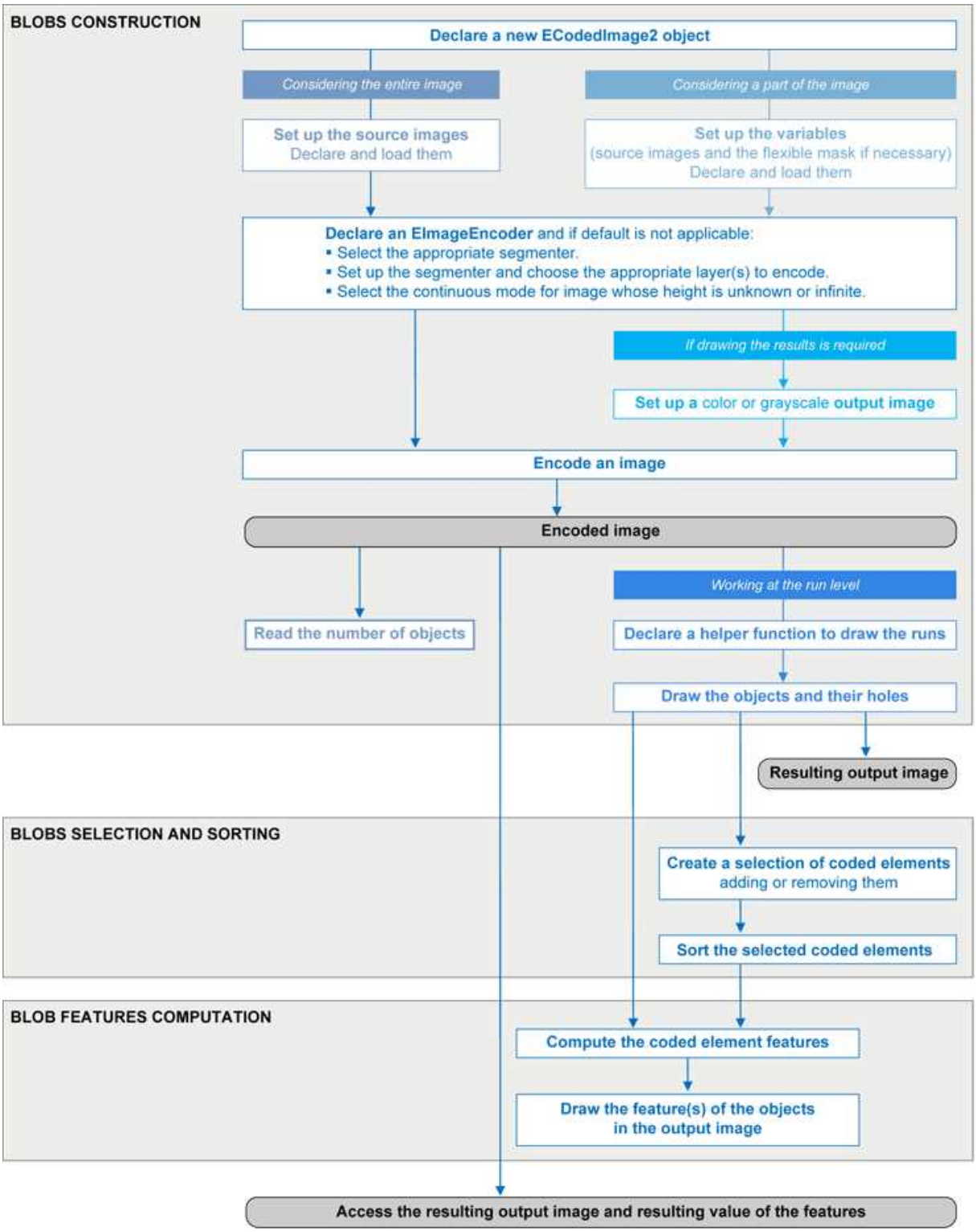
3.1. EasyObject - ブロブの解析

EasyObjectライブラリはブロブ(同じグレーレベル範囲を持つオブジェクトまたはホール) を作成して処理することで、画像にある特徴を抽出します。



このライブラリはBW1、BW8、BW16およびC24ソース画像に使用することができます。そして、多くのオブジェクトを含む大画像のために実行時間が改善された `ECodedImage2`クラスからアクセス可能です。

ワークフロー


BLOBS CONSTRUCTION
Declare a new ECodedImage2 object
Considering the entire image
Considering a part of the image
Set up the source images
Declare and load them
Set up the variables
(source images and the flexible mask if necessary)
Declare and load them
Declare an EImageEncoder and if default is not applicable:
▪ Select the appropriate segmenter.
▪ Set up the segmenter and choose the appropriate layer(s) to encode.
▪ Select the continuous mode for image whose height is unknown or infinite.
If drawing the results is required
Set up a color or grayscale output image
Encode an image
Encoded image
Working at the run level
Read the number of objects
Declare a helper function to draw the runs
Draw the objects and their holes
Resulting output image
BLOBS SELECTION AND SORTING
Create a selection of coded elements
adding or removing them
Sort the selected coded elements
BLOB FEATURES COMPUTATION
Compute the coded element features
Draw the feature(s) of the objects
in the output image
Access the resulting output image and resulting value of the features

ブロブの定義

ブロブとは、同じグレーレベル範囲の近傍ピクセルの集まりです。
オブジェクトまたはオブジェクト内のホールがブロブとなります。EasyObjectの機能はオブジェクトとホールの両方を解析します。
ブロブが構築されると、ホールとオブジェクト間の包含関係が計算されます。

ホールが関心オブジェクトである場合でも、まず関心オブジェクトを検出してからそのホールを検出し(EasyObjectを使用)、その特性を測定する(EasyGaugeまたはEasyObjectを使用)方が簡単です。

ブロブは独立したエンティティとして取り扱われます:

- ブロブが属するレイヤ、位置、長方形ROIまたは計算された特徴によってブロブを選択できます。選択基準を組み合わせることができます(小さいオブジェクトを選択、その中から右側のエッジに近いものを選択など)。
- ブロブをリスト表示して幾何学的な特徴(面積、幅、慣性楕円など)によって並び替えることができます。

ブロブ解析の対象は、長方形 / ネストされたROI、あるいはフレキシブルマスクを用いて複雑な / つながっていない形状の領域に制限できます。

ブロブの構築

EasyObject では次の2つのステップによって関心オブジェクトを選択してブロブ / ホールを構築します:

1. セグメント:ソース画像のピクセルを分類してレイヤを作成し、ラン(run)を構築します(ランとは、同じ属性を共有する、連続する隣接ピクセルのシーケンスです)。
2. エンコード:ランを組み立てて各レイヤのブロブを構築します。
   保持するオブジェクトまたはホールを選択します。
   `EImageEncoder::Encode` はブロブを解析してその結果をコード化された画像に保存します。
   この画像には重ね合わされた相互に排他的な画像レイヤのセットが含まれ、各レイヤのピクセルは同じ属性を持っています(例えば閾値の上であるなど)。
   フレキシブルマスクによってエンコードの対象を任意の形状の領域に制限することもできます。

ホールを構築する必要はありません。ホールは必要となったときにオンザフライ方式で構築されます。

機能

- セグメンテーション GetSegmentationMethod および SetSegmentationMethod
- グレースケール単閾値 EGrayscaleSingleThresholdSegmenter
- グレースケール2重閾値 EGrayscaleDoubleThresholdSegmenter
- カラー単閾値 EColorSingleThresholdSegmenter
- カラー範囲閾値 EColorRangeThresholdSegmenter
- 参照画像 EReferenceImageSegmenter
- 画像範囲 EImageRangeSegmenter
- ラベル付けされた画像 ELabeledImageSegmenter
- 2値画像 EBinaryImageSegmenter

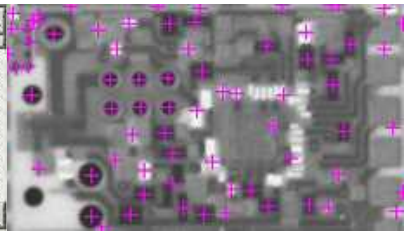
ピクセル合算(エンコーダ)

- レイヤの選択
- オブジェクトの作成:ランを合算してオブジェクトを構築
- ホール構築:ランをホールへ合算

オブジェクトの抽出(幾何学パラメータを使用)

画像がエンコードされると、抽象クラスECodedElementからコード化された要素(オブジェクトまたはホール)にアクセスできます。このクラスには、コード化された特定の要素に適用するための多数のメソッドがあります:

Num	Area	Gravity Center X	Gravity Center Y
59	2221	17.67	95.55
37	387	161.69	53.46
47	344	166.43	90.24
32	327	226.23	72.07
40	251	111.45	62.04
50	239	120.41	91.51
4	144	220.98	2.44
68	142	72.05	123.10



特徴の計算と表示

オブジェクトおよびホール、そしてその特徴に効率良く任意にアクセスできます(インデックス方式)。

画像セグメンタ

ピクセルをセグメント化する方法はいくつかあります:メソッドはGetSegmentationMethodおよびSetSegmentationMethodで選択します。

1. グレースケール単閾値(デフォルト)

EGrayscaleSingleThresholdSegmenterはBW8およびBW16グレースケール画像に適用することができ、次の2つのレイヤを持つコード化された画像が作成されます:

- **黒のレイヤ**(通常はレイヤ 0) には閾値より小さいグレー値を持つ、マスクされていないピクセルが含まれます。
- **白のレイヤ**(通常はレイヤ1) には残りのマスクされていないピクセルが含まれます。すなわち、閾値と同じかそれより大きいグレー値を持つ、マスクされていないピクセルが含まれます。

EasyObjectには5つの閾値化メソッドがあります:

- **Absolute(絶対値)** (整数値) : 白のレイヤの最初のグレー値を示します。SetAbsoluteThresholdメソッドで設定してGetAbsoluteThresholdメソッドで取得します。
- **Relative(相対値)** (%): 黒のレイヤに属する画像ピクセルの割合を表す、0～1までのユーザー定義のfloat値。SetRelativeThresholdメソッドで設定してGetRelativeThresholdメソッドで取得します。
- **Minimum Residue(最小剰余)** (デフォルト) : ソース画像と閾値化された画像の2次差分が最小になるように自動的に計算された値が閾値となります。
- **Maximum Entropy(最大エントロピー)** : 閾値化された結果画像のエントロピー(情報量) が最大になるように自動的に計算された値。
- **IsoData**: 暗いグレー値 (閾値より低いグレーレベル) の平均と明るいグレー値 (閾値より大きいグレーレベル) の平均の中間にあたる値が自動的に計算されます。

最小剰余閾値化メソッドのグレースケール単閾値がデフォルトです。閾値より大きい値を持つピクセルのオブジェクトのみがエンコードされます。

2. グレースケール2重閾値

EGrayscaleDoubleThresholdSegmenterはBW8およびBW16グレースケール画像に適用することができ、次の3つのレイヤを持つコード化された画像が作成されます:

- **黒のレイヤ**(通常はレイヤ 0) には低い閾値より小さいグレー値を持つ、マスクされていないピクセルが含まれます。
- **白のレイヤ**(通常はレイヤ2) には高い閾値と同じかそれより大きいグレー値を持つ、マスクされていないピクセルが含まれます。
- **中間レイヤ**(通常はレイヤ1) には残りのマスクされていないピクセルが含まれます。

Low Threshold(低い閾値) と**High Threshold(高い閾値)** はユーザー定義の整数値で、SetLowThresholdおよびSetHighThresholdメソッドで設定し、GetLowThresholdおよびGetHighThresholdメソッドで取得します。

3. カラー単閾値

EColorSingleThresholdSegmenterはC24カラー画像に適用することができ、次の2つのレイヤを持つコード化された画像が作成されます:

- **白のレイヤ**(通常はレイヤ1) には、閾値の点と白の点 (255、255、255) によって定義される色空間のキューブに属する、マスクされていないピクセルが含まれます。
- **黒のレイヤ**(通常はレイヤ 0) には残りのマスクされていないピクセルが含まれます。

Color Threshold(カラー閾値) は色空間の中で特定の色を指定するための3つの**ユーザー定義**の整数値で、SetThresholdメソッドで設定し、GetThresholdメソッドで取得します。

4. カラー範囲閾値

EColorRangeThresholdSegmenterはC24カラー画像に適用することができ、次の2つのレイヤを持

つコード化された画像が作成されます：

- **白のレイヤ**(通常はレイヤ1) には、低い閾値の点と高い閾値の点によって定義される色空間のキューブに属する、マスクされていないピクセルが含まれます。
- **黒のレイヤ**(通常はレイヤ 0) には残りのマスクされていないピクセルが含まれます。

低い閾値と高い閾値はそれぞれ、色空間内で特定の色を指定するためのユーザー定義の3つの整数値で、`SetLowThreshold`および`SetHighThreshold`メソッドで設定し、`GetLowThreshold`および`GetHighThreshold`メソッドで取得します。

5. 画像範囲

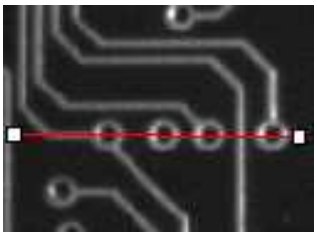
次のケースでは、**ピクセル単位の閾値化**によってセグメント化することで、各ピクセルの値の許容範囲を設定する必要があります：

- 背景が十分に均一でない場合
- 画像全体で照明が均一でない場合
- 画像と参照画像(理想) の相違のみを強調する場合

各ピクセルの許容範囲は2枚の画像を使って指定します：低い参照画像で各ピクセルの最小許容値を、高い参照画像で最大値を指定します。つまり、参照画像はソース画像全体から固定値を差し引いた(足した) ものとなります(ノイズ分布が均一で加法性である場合) 。
そのため、適切な高い / 低い参照画像を準備することが重要になります。

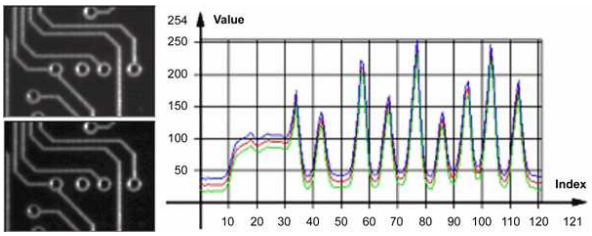
高い / 低い参照画像の準備

最初の画像として欠陥のないものを選び、比較前に安全マージンを追加します。



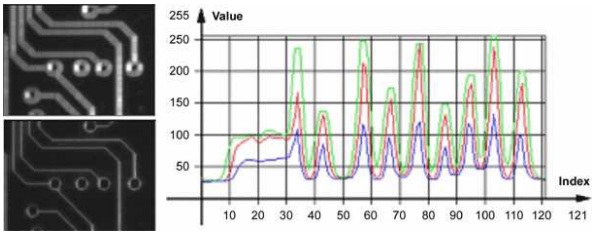
ソース画像

ノイズや照明の変化に対応するためにグレーレベルの許容値が必要です。



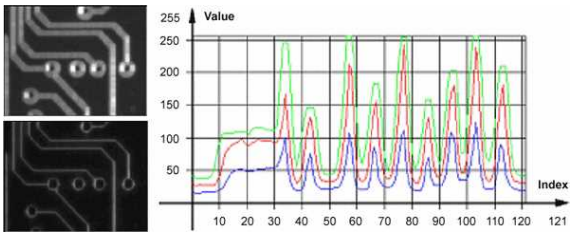
グレーレベルの許容マージン

画像にはいずれかの方向にわずかにずれることがありますが、膨張および侵食のモルフォロジー演算によって明るい / 暗い領域を大きくすることで補正できます。幾何学的な許容マージンはだいたいモルフォロジーフィルタサイズと同じ大きさです。



幾何学的な許容マージン

2つの許容マージンを組み合わせることで最良の結果が得られます。



組み合わせたマージン

画像セグメンタ

EImageRangeSegmenterおよびEReferenceImageSegmenterはBW8、BW16、C24画像に適用することができ、2つのレイヤを持つコード化された画像が作成されます。

低い閾値および高い閾値は、ソース画像と同じ画像形式の2枚の参照画像(低い画像および高い画像)を用いて各ピクセルに対して個別に定義されます。参照画像は各ピクセルの参照ピクセルを個別に定義します。

- **グレースケール**画像の場合、**白のレイヤ**(通常はレイヤ1)には、低い/高い/参照画像内のマスクされていないピクセルのグレー値によって定義される範囲内のグレー値を持つ、マスクされていないピクセルが含まれます。
- **カラー**画像の場合、**白のレイヤ**(通常はレイヤ1)には、低い/高い/参照画像内のマスクされていないピクセルの色によって定義される色空間のキューブ内部の色を持つ、マスクされていないピクセルが含まれます。
- **黒のレイヤ**(通常はレイヤ0)には残りのマスクされていないピクセルが含まれます。

低い画像のポインタは、ソース画像の画像形式に関連付けられている関数を使って設定または取得します:

- BW8: SetLowImageBW8 GetLowImageBW8
- BW16: SetLowImageBW16GetLowImageBW16
- C24: SetLowImageC24GetLowImageC24

高い画像のポインタは、ソース画像の画像形式に関連付けられている関数を使って設定または取得します:

- BW8: SetHighImageBW8GetHighImageBW8
- BW16 SetHighImageBW16GetHighImageBW16
- C24 SetHighImageC24GetHighImageC24

参照画像のポインタは、ソース画像の画像形式に関連付けられている関数を使って設定または取得します：

- BW8: SetReferenceImageBW8、GetReferenceImageBW8
- BW16: SetReferenceImageBW16、GetReferenceImageBW16
- C24: SetReferenceImageC24 、GetReferenceImageC24

6. ラベル付けされた画像

ELabeledImageSegmenter はBW8およびBW16グレースケール画像に適用することができ、グレー値の最大数(BW8画像の場合は256、BW16画像の場合は 65536) に等しいレイヤ数を持つ、コード化された画像を作成します。レイヤnには、nに等しいグレー値を持つ、マスクされていない全ピクセルが含まれます。

デフォルトではすべてのレイヤがエンコードされます。あるいは、インデックス範囲の最小値および最大値を返すSetMinLayerおよびSetMaxLayer関数を使って、1つの範囲のレイヤにエンコードを制限することもできます。

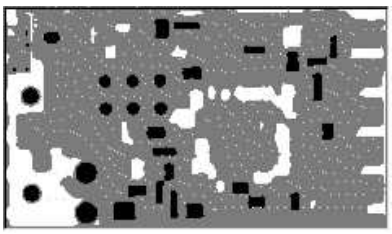
7. 2値画像

EBinaryImageSegmenterは2値画像BW1に適用することができ、次の2つのレイヤを持つコード化された画像が作成されます：

- 黒のレイヤ(通常はインデックス 0) には、0に等しい2進値を持つ、マスクされていないピクセルが含まれます。
- 白のレイヤ(通常はインデックス1) には残りのマスクされていないピクセル、すなわち、 1に等しい2進値を持つ、マスクされていないピクセルが含まれます。

画像エンコーダ

オブジェクトを表すクラス (EObject) は抽象クラスECodedElementから派生したものです。



オブジェクトの構築

エンコードするレイヤを選択する

セグメンテーションメソッド (画像セグメンタ参照) はデフォルトでエンコードするレイヤを設定するもので、その他のレイヤのピクセルはエンコードされなくなります。

GetMaxLayerIndex関数は最大のレイヤインデックス値を返します。すべてのセグメントに使用できます。

各レイヤに個別にエンコードを有効化 / 無効化する

下の表は、レイヤごとの Set/Get関数とデフォルトの有効 / 無効の値をまとめたものです。

2レイヤ式セグメンタ

レイヤ	Set LayerEncoded関数	Get LayerEncoded関数	デフォルト値
黒のレイヤ	SetBlackLayerEncoded	IsBlackLayerEncoded	FALSE
白のレイヤ	SetWhiteLayerEncoded	IsWhiteLayerEncoded	TRUE

3レイヤ式セグメンタ

レイヤ	Set LayerEncoded関数名	Get LayerEncoded関数名	デフォルト値
黒のレイヤ	SetBlackLayerEncoded	IsBlackLayerEncoded	FALSE
白のレイヤ	SetWhiteLayerEncoded	IsWhiteLayerEncoded	FALSE
中間レイヤ	SetNeutralLayerEncoded	IsNeutralLayerEncoded	TRUE

各レイヤにレイヤインデックスを手動で割り当てる

下の表は、レイヤごとの Set/Get 関数とデフォルト値をまとめたものです。

2レイヤ式セグメンタ

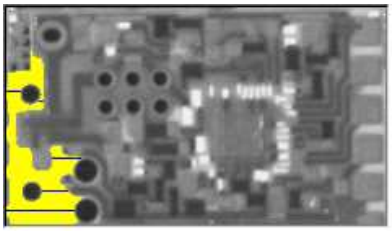
レイヤ	Set LayerEncoded関数名	Get LayerEncoded関数名	デフォルト値
黒のレイヤ	SetBlackLayerIndex	IsBlackLayerIndex	0
白のレイヤ	SetWhiteLayerIndex	IsWhiteLayerIndex	1

3レイヤ式セグメンタ

レイヤ	Set LayerEncoded関数名	Get LayerEncoded関数名	デフォルト値
黒のレイヤ	SetBlackLayerIndex	IsBlackLayerIndex	0
中間レイヤ	SetNeutralLayerIndex	IsNeutralLayerIndex	1
白のレイヤ	SetWhiteLayerIndex	IsWhiteLayerIndex	2

ラン (run)

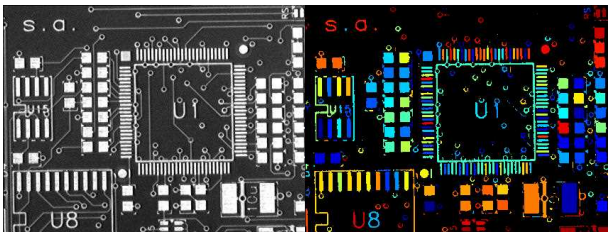
計算の効率を上げるため、オブジェクトは列挙されたランとして記述されます。ランとは、同じ属性 (例えば、与えられた閾値より大きいなど) を共有する隣接ピクセルのシーケンスです。これらのランは画像エンコーダによって結合されてオブジェクトになります。



強調された5つのランの単一オブジェクト

EasyObject はオブジェクトレベルとランレベルで機能しますが、ランレベルではクリティカルなケースでも処理速度が速くなります。この方法は、オブジェクトのカスタム特徴を計算した後で所定のオブジェクトに属するすべてのランを列挙する際に便利です。このようなランレベル処理の例は、ランがカラー表示された出力画像を参照してください。

1. 新しい`ECodedImage2`オブジェクトを**宣言**します。
2. `EImageEncoder`を**宣言**し、場合に応じて適切なセグメンタを選択します。セグメンタをセットアップし、エンコードする適切なレイヤを選択します。
3. **出力画像をセットアップします**。
4. **画像をエンコードします**。
5. **出力画像でランをカラー表示します**。ループ、そして`RunsIterator`オブジェクトを構築して、特定のレイヤのオブジェクトに反復します。このイテレータにより、考慮されるオブジェクトのランを閲覧することができます。考慮されるオブジェクトのランがイテレータによって終了すると、出力画像でそのランの範囲に当たるピクセルが内部ループにより処理されます。
6. **特定のレイヤを選択します**。



ソース画像(左)とレンダリングされた白のレイヤ(右)

連結

隣り合うピクセルは辺または角が接触します。4連結の場合、辺が接触しているピクセルのみが近傍と見なされます。8連結の場合(デフォルト)、角が接触しているピクセルも近傍と見なされます。

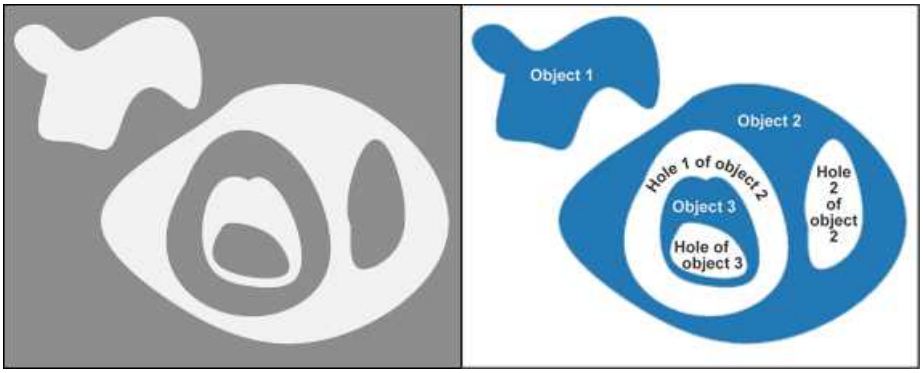
複数の画像は連続モードでエンコードできます。

ホール構築

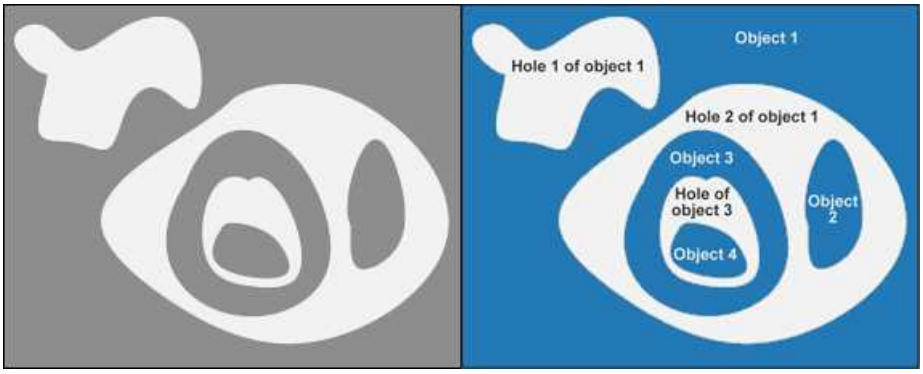
ホールとは、親オブジェクトに全体が囲まれた連結するピクセルの集まりです(連結モードに応じて4ピクセルまたは8ピクセル)。

ホールには子はありません。ホールの中にあるオブジェクトは個別のオブジェクトと見なされます。

EObjectおよびEHoleクラスは両方ともECodedElementから派生しているため、オブジェクトとホールは同じやり方で管理され、同じ関数を共有します。



白のレイヤのエンコード(オブジェクト3個とホール3個)

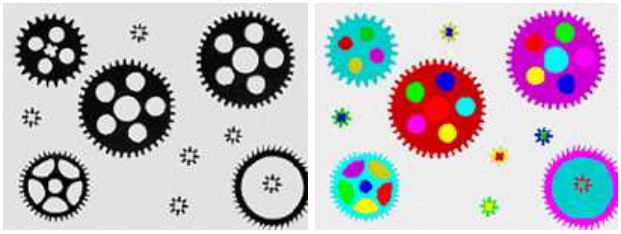


黒のレイヤのエンコード(オブジェクト4個とホール3個)

ホールをカラー表示する手順

1. 新しいECodedImage2オブジェクトを**宣言**します。
2. EImageEncoderを**宣言**し、場合に応じて適切なセグメンタを選択およびセットアップして、エンコードする適切なレイヤを選択します。
3. **出力画像をセットアップします**。
4. **画像をエンコードします**。
5. **ランを描画するためにヘルパー関数を宣言します。**ヘルパー関数(「オブジェクト作成」/「ランレベルの処理」を参照)は、例えば指定された色で出力画像のランを描画します。この関数はオブジェクトとホールで共有できます。

6. **オブジェクトとそのホールを出力画像に描画します。** 選択したレイヤのオブジェクトに反復する必要があります。
   a. ヘルパー関数によって各オブジェクトのランが特定の色で描画されます (`DrawRuns`)。
   b. 現在のオブジェクトでホールが反復処理され、そのランが描画されます。
   c. オブジェクトのホールは、グローバル関数 (`GetFadedColor`) で計算された異なる色でそれぞれ描画されます。この関数はホールのインデックス(例えば赤から緑への勾配) に応じて色を返します。



原画像(左) とオブジェクトとホールの構築(右)

標準モードと連続モード

標準モード(デフォルト)

標準モードでは、画像エンコーダは連続する複数の画像にまたがるブロブを追跡しません。EasyObjectは画像1枚に機能します。その際ブロブがメモリに保持されることはありません。全ブロブがオブジェクトとして返されます。

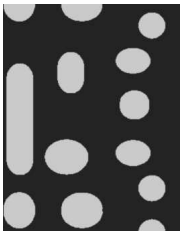
連続モード

連続モードでは、EasyObjectは高さが未知または無限の画像(例えばラインスキャンカメラからの画像) を処理できます。画像は複数のチャンクに分割されて画像エンコードに供給されます。この場合、連続する複数の画像チャンクにまたがるオブジェクトも検出できます。

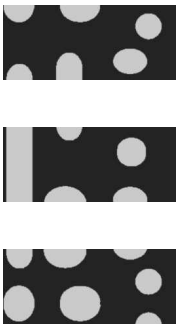
画像エンコーダは、ソース画像の最後の行に接触しているランを含まないオブジェクトのみをエンコードします。画像の下の境界に接触するオブジェクトは、後続の画像チャンクに持続することが見込まれるため、コード化された画像には書き込まれません。しかし、メモリに保持され、後続の画像が解析されるときに処理されます。

大画像は複数のチャンクに分割されることになっており、アレイ `EImageBW8`チャンク[x] に保存されます。

この例では、連続チャンクにエンコードされたオブジェクトを含む一続きのカラー画像を生成する

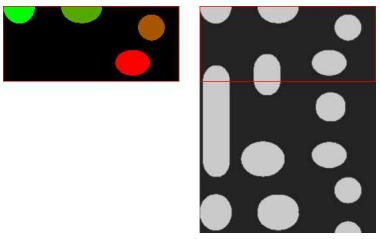


原画像

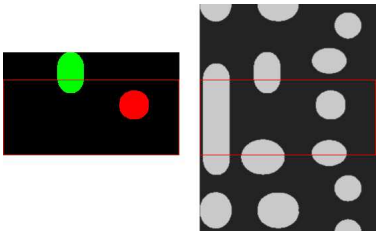


画像の3つのチャンク

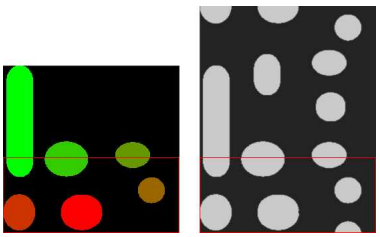
1. **コード化された画像のレイヤにエンコードされたオブジェクトを描画します。** このコードは基本的に「ランの閲覧」コードスニペットと同じです。1つだけ異なる点は、Y軸に沿ってオフセットを適用できる点です。
2. **レイヤのオブジェクトを描画するための関数を定義します。** コード化された画像に前の画像で開始したオブジェクトが含まれている場合：前の画像のこのオブジェクトのランに負のY座標が割り当てられます。
   直近でエンコードされた画像の最初の行がY座標ゼロになります。規則としては、エンコードされたオブジェクト内の最も古いランに最も小さいY座標を割り当てることになっています。
   `EImageEncoder::GetStartY`メソッドは、最も古いランのY座標を取得します。コード化された画像のレイヤの内容を表示する関数を定義する必要があります。
   各オブジェクトは異なる色で表示できます(`GetFadedColor`によって計算されます)。この関数は`DrawRuns`関数のすぐ後に続きますが、連続モードに合わせるために**GetStartY**を考慮します。
3. `EImageEncoder::SetContinuousModeEnabled`属性で**連続モードを有効化**します。例えば、エンコードされた連続画像を保存するため、あるいは出力画像を保持するために、その他の変数を宣言することもできます。
4. **連続するチャンクを解析します。** 連続するチャンクをエンコードするには、`Encode(chunk [count]、codedImage)`、それから`DrawLayer`を使用します。**注記：** 変数カウントは、0、1および2の整数になります。チャンクのオブジェクトが完全でない場合、画像エンコーダの内部メモリに保持されます。



カウントが0に等しい場合、DrawLayer適用後のlayerImageの内容。
考慮される大画像のチャンク。
画像チャンクの左下の2つのオブジェクトはエンコードされません。
これらはチャンクの境界に接触しています。



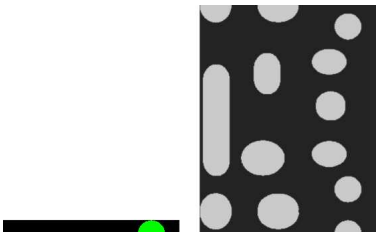
カウントが1に達すると、2つのうち1つのオブジェクトが完全になります。
それにより、次の画像がエンコードされます。
この時点では他の2つのオブジェクトはまだエンコードされません。
これが最後のチャンクのエンコードの結果です(カウント =2)。



前のチャンクから続く3つのオブジェクトが閉じられたため、エンコードされました。

連続モードのフラッシュ

3つの画像チャンクをエンコードした後、完全でないオブジェクトが1つだけ残っています(大画像の右下のコーナー)。しかしこれ以外のチャンクはないため、このオブジェクトを明確に閉じ、画像エンコーダのフラッシュによって残ったオブジェクトをエンコードする必要があります。それによって画像エンコーダの内部メモリは空になります。



フラッシュの結果

ブロブを選択する / 並び替える

オブジェクトセグメンテーションのプロセスでは、小さいオブジェクトとして現れるノイズピクセルを含め、あらゆるブロブがオブジェクトと見なされます。その中でどのブロブを保持するのか、`EObjectSelection`クラスを使って選択することができます。

選択範囲を作成する / 変更する

EObjectSelection `Add`および`Remove`メソッドを次の目的で使用できます:

- 単一オブジェクト、ホールまたはレイヤ全体を選択範囲に追加または削除するため
- 指定されたいくつかの**特徴**に基づいてオブジェクトまたはホールを追加または削除するため(コード化された要素の特徴を計算するの特徴リストを参照) 。
- 特定の**位置**、あるいは指定された長方形ROI内にあるかどうかに基づいてオブジェクトまたはホールを追加または削除するため

これらの操作は1つの選択範囲で好きなようにカスケードしたり組み合わせることができます。

選択範囲をクリアする

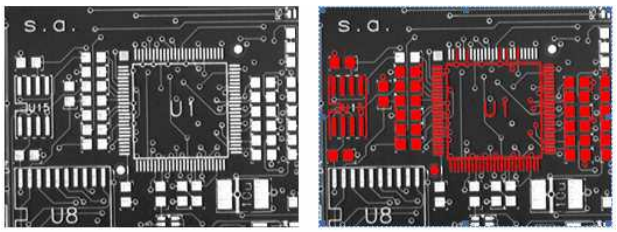
前の選択範囲はEObjectSelection::Clearを用いてクリアすることができます。

選択要素を並び替える

選択要素は、それらの任意の特徴によって並び替えできます。

例

この例では、画像の中央バンドにあるオブジェクトから、表面が100ピクセル未満のものを選択します。



ソース画像と選択されたオブジェクト

1. 新しいECodedImage2オブジェクトを**宣言**します。
2. EImageEncoderオブジェクトを**宣言**し、場合に応じて適切なセグメンタを選択およびセットアップして、エンコードする適切なレイヤを選択します。
3. **ソース画像をエンコードします**。
4. **オブジェクトの選択範囲を作成します。** EObjectSelectionクラスのインスタンスを作成し、この選択範囲にオブジェクトを追加します。例えば、EObjectSelection::AddObjectsを使用します。
5. **いずれかの特徴の値に基づいてオブジェクトを一度に削除します。** EObjectSelection::Removeメソッドのいずれかを呼び出すことで、選択範囲に含まれるオブジェクトの選択を解除できます。
6. EObjectSelection::RemoveUsingFloatFeatureを用いて、**位置に基づいてオブジェクトを削除**します。詳細については、「ランレベルの処理」を参照してください。
7. EObjectSelection::Sort を用いて**選択されたオブジェクトを並び替え**ます。
8. **選択されたオブジェクトにアクセスします。**

高度な機能

計算可能な特徴

Getというプレフィックスが付いているメソッドは遅延評価を示します。メソッドを初めて呼び出したときに結果が計算され、キャッシュされます。

Computeというプレフィックスが付いているメソッドは、呼び出されるたびに再評価され、結果がキャッシュされることはない関数を示します。

位置

境界 (上、下、左、右)	ECodedElement::GetTopLimit ECodedElement::GetBottomLimit ECodedElement::GetLeftLimit ECodedElement::GetRightLimit
重心 (X および Y)	ECodedElement::GetGravityCenter ECodedElement::GetGravityCenterX ECodedElement::GetGravityCenterY
重み付き重心 (X および Y)	ECodedElement::ComputeWeightedGravityCenter

重心と重み付き重心

重心は、コード化された要素の重心の横座標を返します。

重み付き重心は、コード化された要素に対する、与えられた画像の重心を計算します。

範囲

面積(ピクセル数)	ECodedElement::Area
フェレットボックス (中心XおよびY、 22、45、68度の異なる 方向角度の高さと幅)	ECodedElement::ComputeFeretBox ECodedElement::GetFeretBox22Box ECodedElement::GetFeretBox22Center ECodedElement::GetFeretBox22CenterX

	ECodedElement::GetFeretBox22CenterY ECodedElement::GetFeretBox22Height ECodedElement::GetFeretBox22Width ECodedElement::GetFeretBox45Box ECodedElement::GetFeretBox45Center ECodedElement::GetFeretBox45CenterX ECodedElement::GetFeretBox45CenterY ECodedElement::GetFeretBox45Height ECodedElement::GetFeretBox45Width ECodedElement::GetFeretBox68Box ECodedElement::GetFeretBox68Center ECodedElement::GetFeretBox68CenterX ECodedElement::GetFeretBox68CenterY ECodedElement::GetFeretBox68Height ECodedElement::GetFeretBox68Width
バウンディングボックス (中心XおよびY、高さ、幅)	ECodedElement::GetBoundingBox ECodedElement::GetBoundingBoxCenter ECodedElement::GetBoundingBoxCenterX ECodedElement::GetBoundingBoxCenterY ECodedElement::GetBoundingBoxHeight ECodedElement::GetBoundingBoxWidth
最小囲み長方形 (角度、中心XおよびY、 高さ、幅)	ECodedElement::MinimumEnclosingRectangle ECodedElement::MinimumEnclosingRectangleAngle ECodedElement::MinimumEnclosingRectangleCenter ECodedElement::MinimumEnclosingRectangleCenterX ECodedElement::MinimumEnclosingRectangleCenterY ECodedElement::MinimumEnclosingRectangleHeight ECodedElement::MinimumEnclosingRectangleWidth

フェレットボックス

フェレットボックスとは、指定した角度に回転された最小表面の長方形で、オブジェクトの全ピクセルが含まれます。

- **バウンディングボックス**は0°のフェレットボックスです。
- **最小囲み長方形**は、あらゆる角度の中で表面が最も小さくなるフェレットボックスです。
- **フェレットボックス長方形の幅**は、X軸との角度が最も小さい長方形の辺の長さです。必ずしも最も短い辺ではありません。
- フェレットボックス長方形の**高さ**は、長方形のその他の辺の長さです。

その他

オブジェクト輪郭の始点 (X および Y)	ECodedElement::GetContour ECodedElement::GetContourX ECodedElement::GetContourY
最大ラン	ECodedElement::GetLargestRun
ランのカウント	ECodedElement::GetRunCount
オブジェクト数 (インデックス)	ECodedElement::GetLayerIndex ECodedElement::GetElementIndex
ピクセルグレーレベル値 (平均、偏差、分散)	ECodedElement::ComputePixelGrayAverage ECodedElement::ComputePixelGrayDeviation ECodedElement::ComputePixelGrayVariance
ピクセルグレーレベル値 (最小および最大)	ECodedElement::ComputePixelMax ECodedElement::ComputePixelMin

慣性楕円

慣性楕円の離心率	ECodedElement::Eccentricity
モーメント	ECodedElement::GetCentralMoment ECodedElement::GetMoment ECodedElement::GetNormalizedCentralMoment

楕円 (角度、高さ、幅)	ECodedElement::GetEllipseAngle ECodedElement::GetEllipseHeight ECodedElement::GetEllipseWidth
2次幾何モーメント (シグマ: X、XX、XY、Y、YY)	ECodedElement::GetSigmaX ECodedElement::GetSigmaXX ECodedElement::GetSigmaXY ECodedElement::GetSigmaY ECodedElement::GetSigmaYY

注記: *オブジェクト周辺長は、輪郭メソッドでオブジェクトの輪郭を追跡してピクセルをカウントすることで間接的に測定できます。*

標準の幾何学的特徴から別の特徴を派生させることができます。例えば、オブジェクトの伸長は、楕円の長い軸と短い軸の比率、または最大幅に対する最大高さの比率として計算されます。オブジェクトの真円度は、周囲長の二乗をオブジェクト面積掛けるπ掛ける4で割った割合として定義されています。

注記: ***注記:*** *計算式(N = 面積) :*

$$\sigma_x = I_x = \frac{1}{N}\sum(y_i - \bar{y})^2$$

$$\sigma_y = I_y = \frac{1}{N}\sum(x_i - \bar{x})^2$$

$$\sigma_{xx} = I_a = \frac{I_x + I_y}{2} + \sqrt{\left(\frac{I_x + I_y}{2}\right)^2 + I_x I_y + I_{xy}^2}$$

$$\sigma_{xy} = I_{xy} = \frac{1}{N}\sum(x_i - \bar{x})(y_i - \bar{y})$$

$$\sigma_{yy} = I_b = \frac{I_x + I_y}{2} - \sqrt{\left(\frac{I_x + I_y}{2}\right)^2 + I_x I_y + I_{xy}^2}$$

$$\text{WIDTH} = 4\sqrt{I_a}$$

$$\text{HEIGHT} = 4\sqrt{I_b}$$

$$\text{ANGLE} = \operatorname{arccotan}\left(\frac{I_x - I_a}{I_{xy}}\right)$$

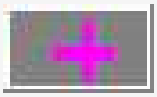
凸包

形状の凸包はオブジェクトを完全に囲む最小面積の凸状多角形です。凸包はオブジェクトの占有面積を特徴づけたり、凹面を観測するために用いることができます。凸包の頂点の数が変数である場合、EPathVectorコンテナに保存されます。

対応する関数はECodedElement::ComputeConvexHullです。

グラフィック表現

オブジェクトは`ECodedImage2::Draw`を使ってソース画像に描画できます。以下の特徴にも`ECodedImage2::DrawFeature`によって描画されるグラフィックがあります。

オブジェクト	グラフィック
バウンディングボックス	
凸包	
楕円	
フェレットボックス	
角度が22°のフェレットボックス	
角度が45°のフェレットボックス	
角度が68°のフェレットボックス	
重心	
最小囲み長方形	
重み付き重心	

座標系と規則

座標系

EasyObjectでは、原則として画像の左上のピクセルの左上の角を原点とするピクセル座標系が使用されます。その結果、ピクセル中心の座標の少数部は「5」となります。この規則はサブピクセル座標の表現に適しています。

角度

数学上の規則に従って、角度は逆向きにカウントされます。そのため、正の角度ではX軸はY軸へ移動します。

特徴を評価する

特徴ごとに1つの属性があるため、**enum**から特徴にアクセスする必要はありません。

コード化された要素を描画する

画像がエンコードされると、抽象クラスECodedElementおよび多数のメソッドセットを通して、コード化された要素(オブジェクトまたはホール)にアクセスできます:

コード化された要素を描画するには

1. 新しいECodedImage2オブジェクトを**宣言**します。
2. EImageEncoderオブジェクトを**宣言**し、場合に応じて適切なセグメンタを選択およびセットアップして、エンコードする適切なレイヤを選択します。
3. **出力画像を作成します**:結果の特徴の描画をカラー表示する場合は、(グレースケール)ソース画像を1ピクセルごとに(カラー)出力画像にコピーします。
4. **ソース画像をエンコードします**。
5. **各オブジェクトの特徴を1つのレイヤに描画します**。
6. 結果を読み取ります。結果は切り捨てられることもあります。特徴をマークするために特別な描画が作成されることがあります(例えば重心のターゲットの描画)。

フレキシブルマスクをレンダリングするには、ECodedElement::RenderMaskを使用します。

オブジェクトおよびホール、そしてその特徴に効率良く任意にアクセスできます(インデックス方式)。

EasyObjectのフレキシブルマスク

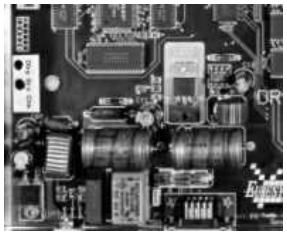
フレキシブルマスクは、BW8画像を出力する任意のアプリケーションによって、またはOpen eVision画像処理関数を使用して生成できます。
EasyObjectでは、ブロブ解析を複雑な / つながっていない形状の画像領域に制限するためにフレキシブルマスクを使用できます。

関心オブジェクトが画像の他の領域と同じグレーレベルである場合、フレキシブルマスクとEncode関数を用いて「保持」と「無視」を定義できます。

フレキシブルマスクは、ソース画像と同じ高さおよび幅を持つBW8画像です。

- フレキシブルマスクのピクセル値0のピクセルはソース画像のピクセルをマスクし、エンコードされた画像には現れません。
- フレキシブルマスクのそれ以外の値を持つピクセルはエンコードされます。

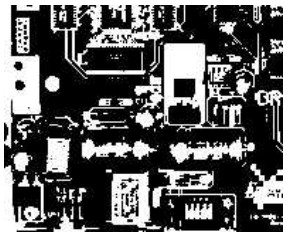
フレキシブルマスクを作成するEasyObject関数



ソース画像

1) ECodedImage2::RenderMask: エンコードされた画像のレイヤから

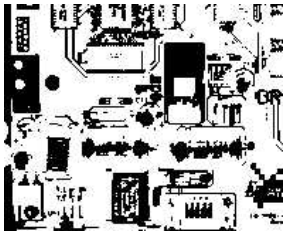
1. フレキシブルマスクをエンコードおよび抽出するには、まず、ソース画像からコード化された画像を構築します。
2. セグメンテーションメソッドを選択します(上の画像にはデフォルトのメソッド `GrayscaleSingleThreshold`が適しています) 。
3. エンコードするコード化された画像のレイヤを選択します(最小剰余閾値を用いて白と黒のレイヤ) 。
4. `mask.SetSize(sourceImage.GetWidth(), sourceImage.GetHeight())`を用いてマスクサイズを希望する大きさに調整します。
5. フレキシブルマスクを引数として`ECodedImage2::RenderMask`に使用します。



フレキシブルマスクとして使用可能なBW8結果画像

2) ECodedElement::RenderMask: ブロブやホールから

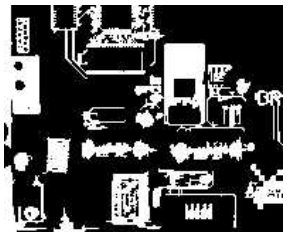
1. 関心があるコード化された要素を選択します。
2. `ECodedElement::RenderMask`を用いて、コード化された画像で選択したコード化された要素から、マスクを抽出するループを作成します。
3. オプションとして、これらの選択されたそれぞれのコード化要素から特徴値を計算します。



フレキシブルマスクとして使用可能なBW8結果画像

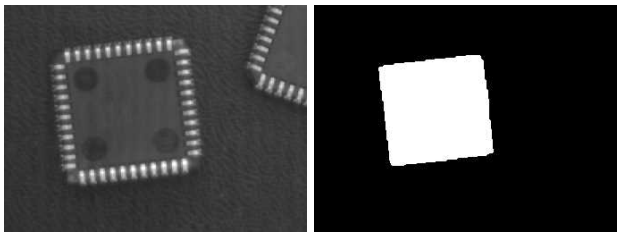
3) EObjectSelection::RenderMask：選択したブロブから

例えば、EObjectSelection::RenderMaskはノイズによって生じた小さいオブジェクトを破棄できます。



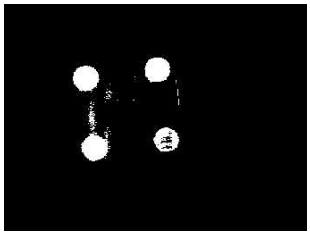
フレキシブルマスクとして使用可能なBW8結果画像

例：EasyObjectによってエンコードされた領域を制限する



4つの円を検出（左）フレキシブルマスクによって中央のチップを分離（右）

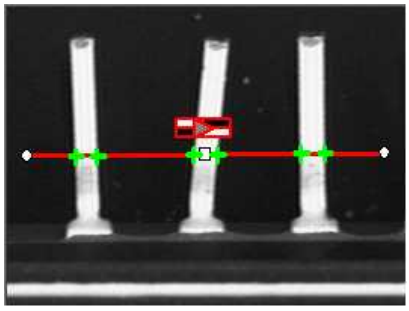
1. 新しいECodedImage2オブジェクトを宣言します。
2. セットアップ変数：最初にソース画像とフレキシブルマスクを宣言し、その後でそれらを読み込みます。
3. EImageEncoderオブジェクトを宣言し、場合に応じて適切なセグメンタを選択します。セグメンタをセットアップし、エンコードする適切なレイヤを選択します。
4. ソース画像をエンコードします。フレキシブルマスクの領域のみを持つレイヤのエンコードはスムーズに行われます。
   グレースケールの単閾値セグメンタによって黒のレイヤで適切に円がセグメント化されたことが分かります：



5. コード化された画像のすべてのオブジェクトを選択します。
6. 大きさが小さすぎるオブジェクトをフィルタ処理で除外し、関心オブジェクトを選択します。
7. 選択された特徴を表示するため、選択されたオブジェクトで反復してブロブの特徴を表示します。

3.2. EasyGauge - サブピクセルまで測定

EasyGaugeライブラリは寸法を制御します。各部品の位置、方向、曲率およびサイズを高精度で測定します。グラフィックで位置ゲージとサイズゲージを連携させ、グループ化した階層でそれらを組み合わせて、すべてのパラメータとともに保存および取得することができます。



ワークフロー

ゲージモデルはプログラミングにより、あるいはグラフィックエディタで構築して、最終アプリケーションで「再生」します。
対象となるモデルの複雑さと必要とされる精度に合ったワークフロー(キャリブレーションなし、キャリブレーションあり、グループ化) を選択してください。

キャリブレーションなしのゲージング: シンプルなモデル向け

EasyGaugeの基本的な使用方法は簡単です。

1. 必要な測定に合わせてゲージオブジェクトを作成します。
2. デフォルト値が適切でないパラメータを変更します。
3. 希望する測定関数を呼び出します。
4. 結果の位置パラメータを読み取ります。

キャリブレーションなしのゲージングは簡単に実装できますが、デメリットもあります:

- 測定はミリメートル単位ではなくピクセル単位で実行されます。
- 測定モデルは移植性がありません: 観察条件が変わったらゲージの位置およびサイズを修正する必要があります。
- 光学ディストーションやパースペクティブにより不正確な測定が行われます。

キャリブレーションありのゲージング: 1つまたは2つのシンプルな測定位置向け

キャリブレーションありのゲージングは精度が高く、観察条件に関係なく検査対象の部品を測定できます。
すべての測定はキャリブレーション単位で行われ、あらゆるディストーションが暗黙のうちに補正されます。 視野キャリブレーションの詳細を知るには、キャリブレーションを参照してください。

1. キャリブレータオブジェクトを作成します。
2. 検査対象の画面上にそれを配置します。
3. キャリブレーションパラメータを調整します。
4. ゲージをアタッチします。

複雑なゲージング

ゲージをグループ化して(ゲージ操作プロセスを参照) 、別のアイテムにアタッチすることができます:

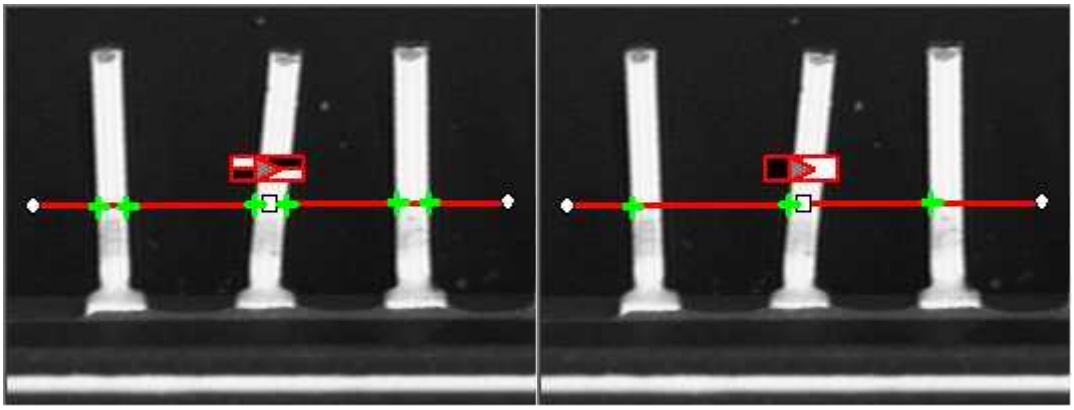
- EFrameShapeオブジェクトに**ゲージをアタッチする**と、ゲージがフレームと一緒に移動します(平行移動および / または回転) 。アプリケーションプログラムは検査対象の部品を追跡するためにフレーム位置を変更しなければなりません。
- **別のゲージにゲージをアタッチする**と、支持ゲージの測定位置に応じてそれらが移動します。 例えば、ある部品の輪郭の検出に使用している長方形ゲージに複数のゲージがアタッチされている場合、長方形の輪郭が適合すると全ゲージが自動的に部品を追跡します。

複数の測定位置がある場合、完全なモデルをキャリブレーションモード、係数、アタッチされているゲージと一緒に1つのファイルに保存できます。

ゲージの定義

ポイントゲージ

1つまたは複数のオブジェクトエッジを交差するラインセグメント測定に沿って、最も重要な遷移点を選択できます。 ノイズを除去するために横方向および縦方向のフィルタリングを実行できます。



点の位置検出。 コントラストベースの選択

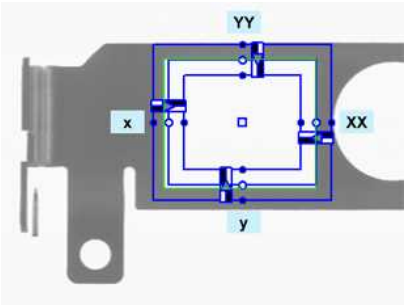
長方形ゲージ

長方形ゲージの位置は、そのノミナル位置(中心の座標によって決まる)、ノミナルサイズおよび回転角度によって定義されます。

長方形の各辺には独自の遷移検出パラメータを設定することができ、ActiveEdges属性によって有効または無効に設定できます。辺が有効である場合:

- パラメータの値を設定すると、現在有効な辺のみに適用されます。
- この属性の値がすべての有効な辺で同じ場合にのみ、パラメータの値を取得したときに結果が得られます。
- 有効な辺のみが測定およびモデルフィッティングに使用されます。

このような規則により各辺に異なるパラメータを設定することができるため、長方形全体の代わりに平行な辺または角の点のみを測定できます。4つの辺はそれぞれ「x」、「y」、「XX」および「YY」の文字で表します。



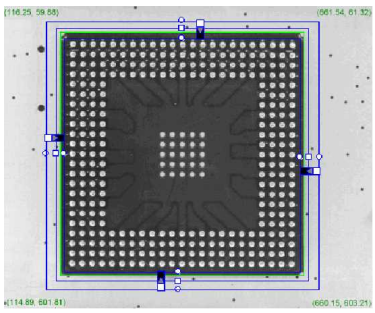
長方形ゲージの各辺のネーミング規則

使用方法

ゲージを定義して位置を合わせ、Measureを使用してラインを合わせます。
長方形のプロパティを得るには、ActualShapeを**TRUE**に設定して、フィッティングされたライン(**TRUE**値)が返されるようにします(デフォルトは**FALSE**)。

あるいは、MeasuredRectangleによりERectangleオブジェクトとして結果を出力します。

例えば長方形フィッティングゲージを使用して、長方形の4つの角(ランドマーク)の位置を正確に特定できます。



長方形の角の位置を特定する

ウェッジ型ゲージ

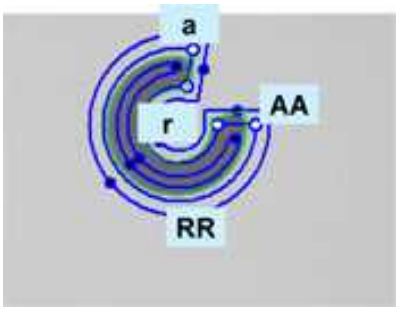
ウェッジ型ゲージの位置は、ノミナル位置(中心の座標によって決まる)、ノミナル内半径および外半径(内径および外径)、幅(半径の差)、角度の開始位置、角度の大きさによって定義されます。

Setメンバは、完全な円、扇形、ディスクを区別します。

ウェッジの各辺には独自の遷移検出パラメータを設定することができ、ActiveEdges属性によって有効または無効に設定できます。辺が有効であるとき:

- パラメータの値を設定すると、現在有効な辺のみに適用されます。
- このパラメータの値がすべての有効な辺で同じ場合にのみ、パラメータの値を取得したときに結果が得られます。
- 有効な辺のみが測定およびモデルフィッティングに使用されます。

このように各辺に異なるパラメータを設定できるため、ウェッジ全体の代わりに平行な弧または傾斜した辺またはコーナーの点のみを測定できます。4つの辺はそれぞれ「a」、「r」、「AA」および「RR」の文字で表します。



ウェッジ型ゲージの各辺のネーミング規則

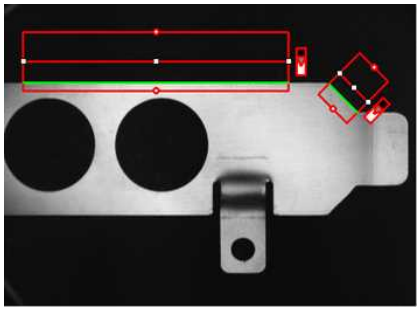
使用方法

ゲージを定義して位置を合わせ、Measureを使用してラインを合わせます。
ウェッジの属性を得るには、ActualShape**属性をTRUE**に設定して、フィッティングされたラインが(デフォルトのノミナルライン位置FALSEの代わりに)返されるようにします。

あるいは、MeasuredWedgeによりEWedgeオブジェクトとして結果を出力します。

ラインゲージ

ラインゲージの場所は、その中心の座標、長さ、X軸を基準とした角度によって定義されます。ラインの傾斜値を指定するには、AngleおよびKnownAngleを設定します。



ラインフィッティング

使用方法

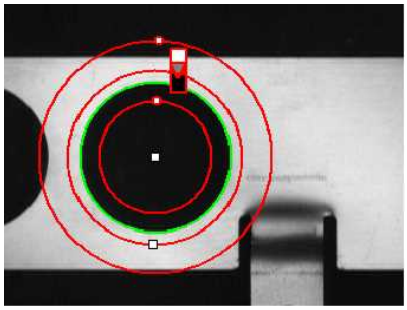
ゲージを定義して位置を合わせ、Measureを使用してラインを合わせます。ラインの属性を得るには、ActualShape**属性をTRUE**に設定して、フィッティングされたライン(**TRUE**値) が(デフォルトのノミナルライン位置FALSE値の代わりに) 返されるようにします。

あるいは、MeasuredLineによりELineオブジェクトとして結果を出力します。

円形ゲージ

円形ゲージの位置は、ノミナル位置(中心の座標によって決まる) 、ノミナル直径(または半径) 、角度の開始位置、角度の大きさによって定義されます。

Setメンバは円と弧を区別できます(弧の大きさを指定する必要があります) 。



円形フィッティング

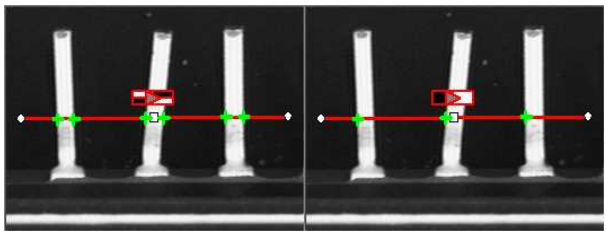
使用方法

ゲージが定義されて位置決めされたら、Measureを使用して円形フィッティングの演算を開始します。測定結果を得るには、ActualShapeモードを**TRUE**に設定します。ActualShapeモードにより、照会時にフィッティングされた円(**TRUE** 値) または円のノミナル位置(**FALSE**値、デフォルト) が返されるのかが決まります。そして、要求された情報が円のプロパティを用いて取得されます。

あるいは、MeasuredCircleによりECircleオブジェクトとして結果を出力します。

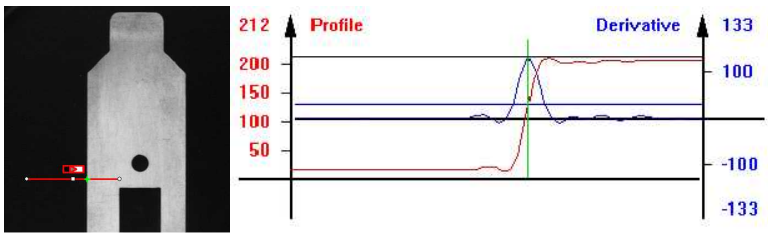
ピーク解析により遷移点を検出する

ラインセグメント測定に沿って1つまたは複数のオブジェクトエッジを交差するすべての遷移点の位置を検出した後で、その中で最も重要な箇所を選択できます。ノイズを除去するために横方向および縦方向のフィルタリングを実行できます。



点の位置検出。コントラストベースの選択

点の位置検出の原理



点の位置検出の原理(左)、S字曲線とその導関数(右)

画像から抽出した線形のプロファイルでは、暗い色から明るい色(またはその逆)への遷移がエッジとして表されます。ピクセル値をゲージに沿ってプロットすると、このような遷移はS字状の曲線となります。この曲線の一次導関数は遷移点付近のピークとなって表れます。コントラストがはっきりしているほど遷移が鮮鋭になり、ピークが大きくなります。

EasyGaugeはプロファイル(赤色の曲線)に沿ってピクセル値を抽出し、ピーク解析によって遷移の位置を検出します。ピーク領域[1]にある全ピクセル値が遷移の位置検出を計算するために使用されます。

- 遷移が少なくとも2ピクセル幅のほぼ均一な領域に囲まれている場合にのみ、サブピクセル精度が可能になります。
- BWB[2] 遷移のプロファイルは上り曲線となり、ピークは正の値をとります。それ以外の場合は、曲線は下りでピークは負になります。

[1]導関数曲線とユーザー定義の水平閾値レベルの間の領域

[2]Black / White / Black(黒 / 白 / 黒)

- BWBまたはWBWの遷移は一次導関数ではなく、EPointGauge(またはサンプルパス)に沿ったグレーレベルプロファイルでのピーク解析に基づいて検出されるため、通常はデフォルトの閾値(20)を用いてピークを検出することはできません。

EPointGaugeには点測定の全パラメータが含まれ、コントラストがはっきり分かれているエッジはデフォルト値で検出されます。

EPointGaugeのパラメータ

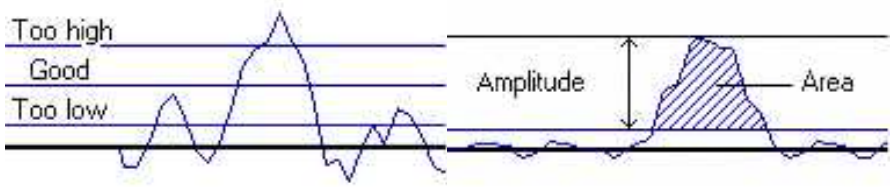
Center:点のノミナル位置(通常は測定の前と後で異なる)。
Tolerance:許容値とゲージの方向。
TransitionType、TransitionChoice、TransitionIndex:ピーク選択の方法。
Threshold:雑音排除性。
MinAmplitude、MinArea:ピーク強度。
Thickness、Smoothing:ローカルフィルタ幅。
RectangularSamplingAreaはサンプリング領域(デフォルトは長方形)に横断フィルタリングモードを設定します。
Measure:オブジェクトを測定します。
- 単一遷移モードでは、適切な点が検出されるとValidがTrueを返します。測定結果を得るには、ActualShapeをTrueに設定することで、Centerにより検出点が返されるようにします(デフォルト値のFalseの場合は点のノミナル位置が返されます)。
- 複数遷移モードでは、NumMeasuredPointsは検出された点の数を返し、GetMeasuredPointは検出された点の情報を含むEPointオブジェクトを返します。0～GetNumMeasuredPoints-1までの整数インデックスを渡す必要があります。
GetMeasuredPeak:ピークのAreaとAmplitudeを含むEPeakを返し、測定セグメントに沿って座標を制限します(Start、LengthおよびCenterの値)。

ピークを選択してエッジ精度を高める

閾値レベルが重要になります:

- 高すぎると重要なピークを見落としてしまうおそれがあり、優れた精度を得るにはピクセル値が不十分であることがあります。
- 低すぎるとノイズがピークと見なされるおそれがあります。

このジレンマを解消するため、EasyGaugeではピークを選択する際に低コントラストまたは偽のエッジを拒否することができるようになっています:遷移の強度はピークの*振幅と面積*によって測定されます。エッジ測定ごとにピークの振幅と面積が求められます。最小振幅または最小面積より値が小さくなると、その位置ではピークが無視されて点が考慮されなくなります。

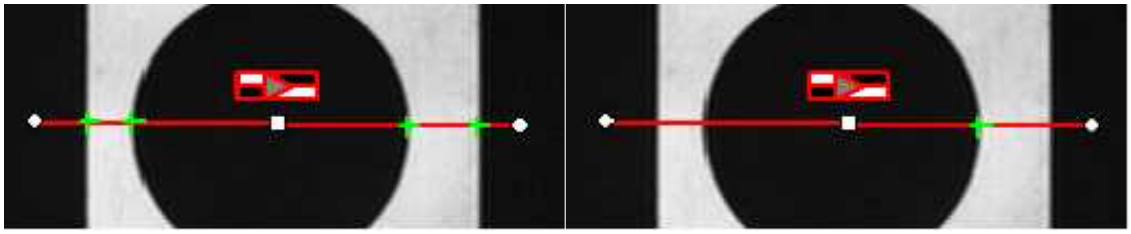


閾値レベルの選択(左)とピークの振幅 / 面積(右)

複数遷移と単一遷移

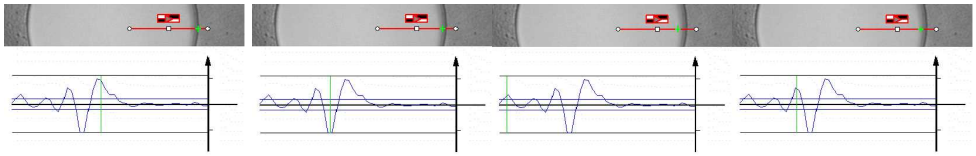
*複数遷移モード*では、複数のエッジ点を一度に測定して、後からすべての結果を取得することがで

きます。



複数遷移(左)と単一遷移(右)

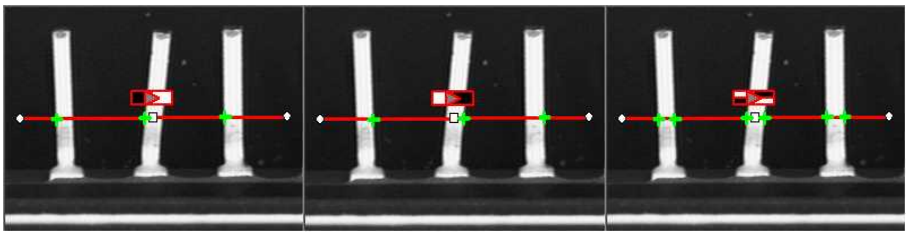
最も重要な遷移を1つだけ選択するには、最高ピーク、最大面積のピーク、ゲージ中心に最も近いピーク、ゲージの片方の端からN番目のピークの4つの基準を使用できます。



最良の面積(1番目の画像)、最良の振幅(2番目の画像)、最も近い(3番目の画像)、端から3番目の画像(4番目の画像)

正または負のピークの選択

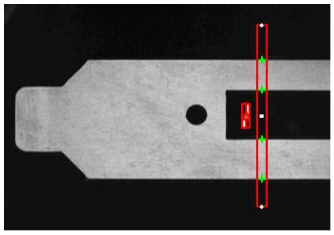
遷移の極性を選択することでピークを絞り込むこともできます:白から黒または黒から白(すなわち正または負のピーク)または無関係。



黒から白、白から黒の極性、無関係

プレフィルタリング

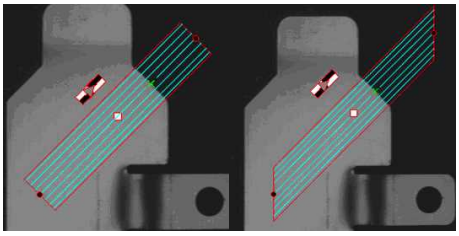
画像の局所的なプレフィルタリングにより、ノイズを除去できます。
横手方向(縦方向)のフィルタリングによって、画像をサンプリングする際に複数の平行線が平均化されます。
結果のプロファイル曲線に長手方向(横方向)の均一フィルタリングを適用することもできます。



フィルタリングのための厚めのポイントゲージ

横手方向のフィルタリング

横手方向のフィルタリングでは、平行四辺形または長方形(デフォルト)内に平行なラインセグメントが配置されます。この動作は切り替え可能です。
角度が0°または90°に近い場合、あるいは厚さが5未満の場合は、平行四辺形モードの方が長方形よりも速く処理されます。厚さ = 1の場合、2つのモードの間に違いはありません。
厚さは平行線の数を決定します。
サンプリング領域はすべての平行線分を含む最小領域です。



長方形のサンプリング領域(左)と平行四辺形のサンプリング領域(右)

点の測定位置

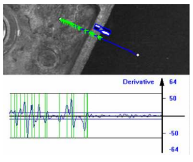
ポイントゲージの予想される**ノミナル**位置は、その**中心**、X軸を基準とした方向**角度**、長さ**許容値**によって指定され、それによって点の位置が変化することがあります。

その結果は、検出された点の座標(**現在**位置)と遷移の強度(振幅および面積)です。
低い値は、信頼性または精度が低い測定による弱いエッジを意味します。

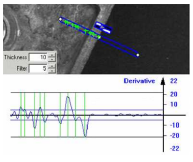
不鮮明なエッジ用に点測定パラメータを調整する

EasyGaugeのデフォルトのパラメータおよび動作モードは鮮鋭なエッジに適しています。しかし、より複雑なケースではパラメータの調整が必要になる場合もあります。

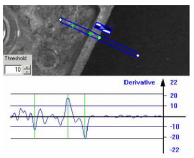
1. ゲージ点の位置と許容値を設定します。
中心位置と方向はサンプル画像または考慮する座標に基づいて簡単に決まります。許容値はエッジ位置の変動に応じて設定します。許容値を大きくするとエッジに当たる可能性は高くなりますが、それが偽のエッジであったり、関係のない形状であったりする場合もあります。



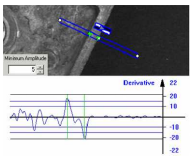
2. ノイズ除去が必要かどうかを決めます。ゲージを希望する位置に配置し、プロファイル曲線とその導関数を観察します(プロットされた曲線を見ながらフィルタリングパラメータを変化させます)。曲線の規則性からグレーレベル値の分布の様子が分かります。



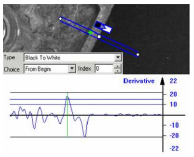
これらの係数が設定されると、グレーレベルのプロファイルはそれ以上変化しません。



3. 閾値を設定します。その際、ピーク領域のピクセルが十分に含まれるように低めに設定します(サブピクセル精度を高めるため)。ただし、周辺ノイズより低くならないようにしてください。



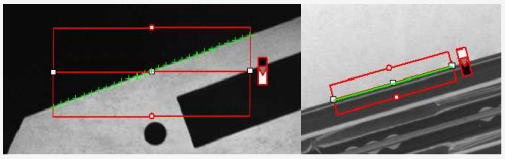
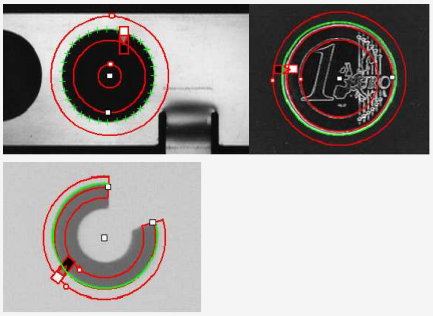
4. 弱い / 偽のエッジを除去します。その際、ピーク振幅および面積のリストを参考にします。これらの値を正しい / 無関係のピークに沿ってプロットすることで、ピーク拒否の適切なリミットを見つけるのに役立ちます。

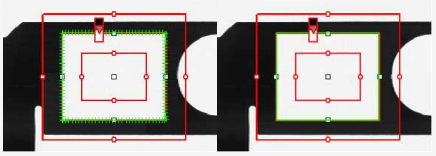
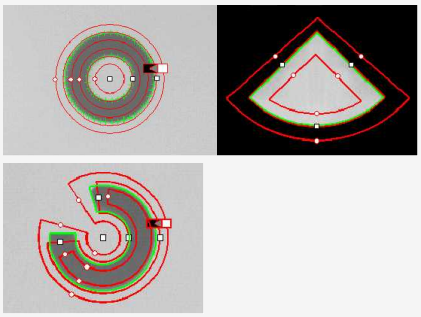


5. すべての遷移点または最も重要な遷移点のみが必要であるのかを選択します。すべての点が必要な場合、1つずつ順に照会できます。それ以外の場合は、強度、順序または遷移の極性(黒から白またはその逆)に基づいて点を選択するのか、選択方法を選ぶ必要があります。

幾何学モデルを使って形状を見つける

ELineGauge、ECircleGauge、ERectangleGaugeまたはEWedgeGaugeといったあらかじめ定義されている幾何学モデルをオブジェクトのエッジにフィッティングすることができます。ターゲットのエッジを定義し、等間隔の点測定ゲージでエッジに沿って点をサンプリングします。その際、最小二乗のモデルフィッティングを適用できます:

ライン: 直線エッジの位置および方向を測定します。	
円: 円または弧の位置および曲率を測定します。	

長方形：長方形の位置、方向、大きさを測定します。	
ウェッジ：円 / 扇形 / 曲線長方形の位置、方向、大きさを測定します。	

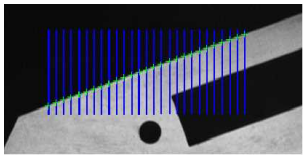
すべてのタイプのゲージに共通の機能があります：

ポイントサンプリング

ポイントゲージはエッジに沿って配置され、等間隔の点において測定が行われます。各点は長方形およびウェッジゲージの辺ごとに調整できます。すべての点測定パラメータおよび動作モードに対応しています。

`SamplingStep`ではモデルに沿って点の位置検出ゲージの間隔を設定します。

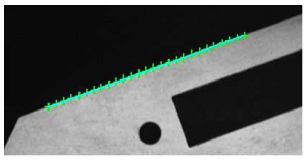
`NumSamples`は、モデルのフィッティング演算中にサンプリングされた点の数を返します。



パスのサンプリングとサンプリングされた点

モデルフィッティング

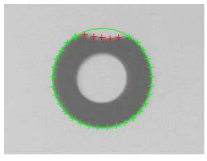
エラーを最小限に抑えてエッジパラメータを最適に推計できるようにモデルが調整されます。長方形とウェッジには平行性と同心性について制約があります。この図はサンプリングされた点とフィッティングされたラインを示します。



外れ値の排除

モデルフィッティングの後、いくつかの点がフィッティングされたモデルからかなり離れた場所にあり、それによって位置特定精度が損なわれることがあります。EasyGaugeでは`FilteringThreshold`属性を使ってこのような点を外れ値としてタグ付けし、無視することができます。

外れ値を除外するプロセスは、`NumFilteringPasses`を使って数回繰り返すことができます。
モデルフィッティング演算後に残る有効なサンプリング点の数は`NumValidSamples`で保持されます。
これらの点からフィッティングされたモデルまでの平均距離は`AverageDistance`によって返されます。



ゲージ操作：Draw(描画)、Drag(ドラッグ)、Plot(プロット)、Group(グループ化)

EasyGaugeではグラフィック上でゲージと連携することができます。それにより、ゲージの位置決め / サイズ変更、グループ化されたアイテムの階層化、モデルファイルから動作パラメータの保存 / 取得が可能になります。

Draw(描画)

Drawはゲージのグラフィック表示を行います。描画は、希望のウィンドウに関連付けられたデバイスコンテキスト内で現在のペンを使って行われます。演算に応じてハンドルが表示されます。

Drag(ドラッグ)

対話形式で画像上にゲージをドラッグすることができます。さまざまなドラッグハンドルを使用できます。

- HitTestによって、マウスカーソルがいつハンドルに合わせられたかを判断します。そしてカーソルが合わせられると、カーソルの形状が変化してドラッグ可能な状態であることを知らせます。
- Dragはハンドルを移動し、それに合わせてゲージも移動します。

Plot(プロット)

EasyGaugeではサンプリングされたパスおよび / またはその導関数に沿ってグレーレベル値をプロットすることができます。これはパラメータの調整に便利です。
点測定ゲージはMeasureを呼び出した後でプロットできます。
モデルフィッティングゲージは、**0**から**GetNumSamples-1**(を含む)の間のインデックス引数でMeasureSampleを呼び出した後でプロットできます。
対応するサンプリングパスを表示するには、**EDrawingMode_SampledPath**モードでDrawメソッドを使用してください。

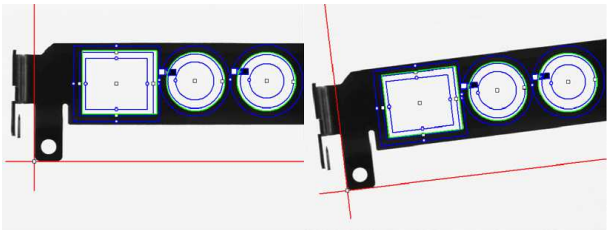
Group(グループ化)

測定前に検査対象アイテム / サンプルの動きを追うために、測定ゲージをグループ化して移動(平行移動および回転)可能な専用ツールを形成することができます(相対位置は固定されたままです)。

Attachによりゲージが母ゲージまたはEFrameShapeオブジェクトに関連付けられます。

NumDaughters、GetDaughterまたはMotherにより、アタッチされた娘または母に関する情報が取得されます。

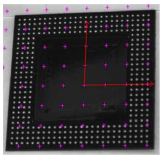
Detach、DetachDaughtersはゲージまたは娘ゲージを母ゲージから切り離します。



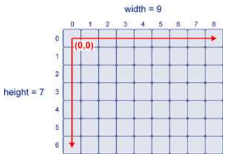
キャリブレーションと変換

視野キャリブレーション

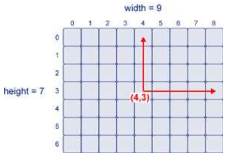
キャリブレーションにより、リアルワールドの点座標と画像ピクセル間の関係が定義されます。シンプルなキャリブレーションモデルは計算時間が短縮されます。また、再現性のある部品位置は特定が容易になります。



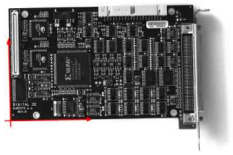
未処理センサー座標系は左上の角から始まり、右方向および下方向へ広がります。
横座標の範囲は**0**～**幅-1**で、縦座標の範囲は**0**～**高さ-1**です。このとき、整数の座標値はピクセルの中心に相当します。



中心配置センサー座標系は中心(未処理座標系では[幅-1]/2、[高さ-1]/2) から始まり、右方向および上方向に広がります。



リアルワールドの3D座標は、参照平面に関連付けられた2D参照フレームで定義されます。通常、軸の原点および方向は、検査対象部品の主要な特徴に基づいて調整されます。



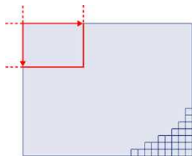
ワールドからセンサーへの変換前に

ワールド座標からセンサー座標への変換を行う前に、ディストーションの要因を取り除いてください:

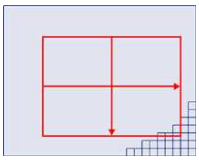
- 非スクエアピクセルを避けるため、掃引周波数またはスキャン速度を調整します。
- パースペクティブ効果を最小限に抑えるために光学アライメントを調整します。視野とセンサー面を平行にしてください。
- 光学ディストーションを最小限に抑えるため、長い焦点距離と高品質のレンズを使用してください。
- レンズ倍率、観察距離および焦点に応じて適切なスケールファクターを使用してください。
- 確実な固定、部品の動き / 取り込みトリガーの同期により、スキューおよび平行移動が最小限に抑えられます。

ワールドからセンサーへの変換の効果

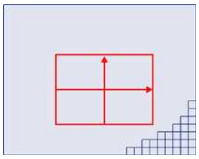
- **キャリブレーションなし**。ワールド座標とセンサー座標が一致します。



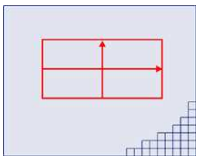
- **平行移動キャリブレーション**: 座標の原点を移動できます。ワールド座標がピクセル単位に相当します。



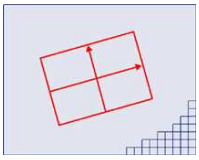
- **等方性スケーリング**(スクエアピクセル) : スケールファクターによりピクセル値が物理的な測定値に変換されます。



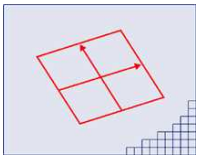
- **異方性スケーリング**(非スクエアピクセル) : [-4/3、-3/4](または[3/4、4/3]) 範囲内のピクセルアスペクト比 (X /Y) を持つ2つのスケールファクターを使用します。ピクセルは常に正方形として表示されるため、引き伸ばされた画像になります。



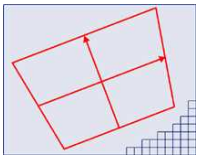
- **スケーリングとスキュー**(スクエアピクセル) : 平行移動、回転、スケーリングによって、回転した検査対象部品に合わせてリアルワールドの軸が調整されます。



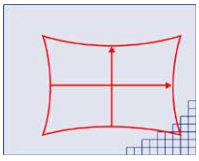
- **スケーリングとスキュー**(非スクエアピクセル) : ディストーションが見られます。これはカメラのスキャン速度とピクセル間隔が合致していない場合に生じます。



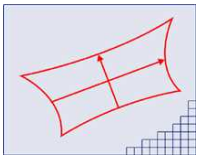
- **パースペクティブディストーション**の結果、オブジェクトはさらに離れ、より小さく見えます。ラインは直線のままですが、角度は保持されません。



- **光学ディストーション**により、長方形がクッションまたは樽のような形状になります。



- **複数ディストーションの組み合わせ**の結果、複雑で非線形なリアルワールドからセンサー空間への変換になります。



EWorldShapeを用いたキャリブレーション

光学セットアップが変更された場合、EWorldShapeオブジェクトにより(カメラ位置とレンズ倍率が固定された特定の画像条件のもとで) 視野全体をキャリブレーションできます。
EWorldShapeは適切なキャリブレーション係数を計算し、それに関連付けられた測定ゲージを変換します。
それにより、ワールドからセンサーへの変換パラメータの設定、一方の座標系から別の座標系への変換の実行、未知のキャリブレーションパラメータの決定、後の再利用を見据えた変換パラメータの保存を行えます。

キャリブレーションの後、次のような目的のためにEWorldShapeはSensorToWorldおよびWorldToSensorによって任意の点の座標変換を実行できます:

- 非スクエアピクセルと回転座標軸を測定するため
- パフォーマンスを低下することなくパースペクティブおよび光学ディストーションを補正するため

キャリブレーション係数を取得するにはいくつかの方法があります：

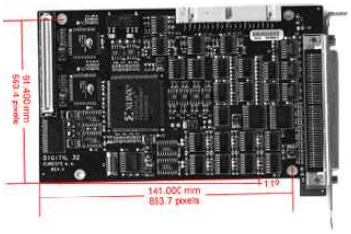
推定する(ディストーション補正が不要で精度要件が低い場合に可能)

キャリブレーション係数を推定するには、視野の境界を決めて画像解像度を視野サイズで割るか、または以下の手順に従います：

1. 検査対象部品またはキャリブレーションターゲット(長方形など)の画像を取り込みます。
2. 画像上でコーナーなどの特徴点の位置を(目で)確認し、その座標をピクセル単位で決めます。ここでは `(i,j)` とします。
3. ユークリッド距離式を使ってキャリブレーション係数を導きます：$C = \frac{\sqrt{(i_1 - i_0)^2 + (j_1 - j_0)^2}}{D}$ このとき、`C`は単位当たりのピクセル数で表されたキャリブレーション係数で、`D`は対応する点間の各単位でのワールド距離です。
4. 非スクエアピクセルの場合は、水平点と垂直点の組にこの演算を繰り返します。

スキュー角を推定するには、ワールド座標系のX軸上にある2点に次の式を適用します：

$$\theta = \arctan \frac{j_1 - j_0}{i_1 - i_0}$$



スケールファクターとスキュー角を推定する

キャリブレーション係数を使用できる場合、`SetSensor`を使って調整してキャリブレーションモードを設定するか、あるいは`SetSensorSize`、`SetFieldSize`、`SetResolution`、`SetCenter`、`SetAngle`を使って個別に設定します。

参照点(ランドマーク)のセットをキャリブレーション関数に渡す

少なくとも4つのランドマークを特定し、センサー座標系(画像処理により)およびワールド座標系(現在の測定)での座標を取得します。ランドマークの数が増えるとキャリブレーション精度が高まります。

結果ピクセルのアスペクト比(X解像度 / Y解像度)は、[-4/3、-3/4](または[3/4、4/3])の範囲内でなければなりません。

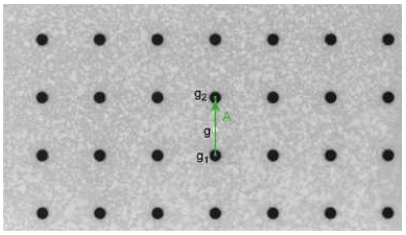
キャリブレーションターゲットを分析する

キャリブレーションターゲットを自動的に分析し、適切なランドマークのセットを取得することができます。希望するランドマークの点座標を抽出する適切な手順を実行できるのであれば、この方法で簡単に自動キャリブレーションを行えます。

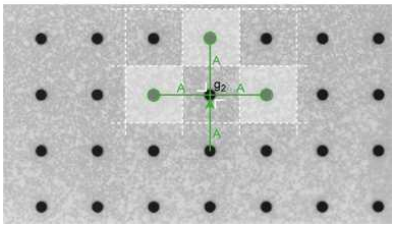
Open eVisionでは、(任意の形状の)対称的なドットの長方形グリッドを持つ、その他にオブジェクトがない特別なターゲットを使用します。

ドットグリッドベースのキャリブレーションの例

1. キャリブレーションターゲットが視野全体を覆うようにキャリブレーションターゲットの画像を取り込みます(またはドットのみが含まれるROIに表示画像を制限します)。
2. EasyObjectの場合と同様にして、ブロブ解析を実行してドットの中心座標を抽出します。
3. 検出されたすべての点をAddPointに渡します(センサー座標のみ)。
4. RebuildGridを呼び出し、各ドットのワールド座標を計算するための対話式アルゴリズムを用いて視野のキャリブレーション用のグリッドを再構築します。
   a. グリッド点の重心 (g) に最も近いグリッド点 (g_1およびg_2) が選択され、長さAの最初の参照セグメントが形成されます。



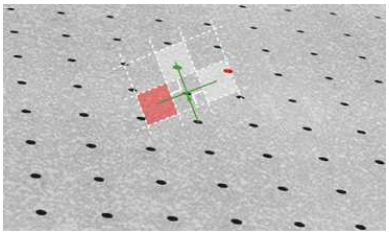
   b. 参照セグメントの先端 (g_2) から垂直方向に3つの許容範囲が決められます(図の白の正方形)。これらの許容範囲の中心は、(g_2) から距離A(参照セグメントの長さ) だけ離れた場所に位置決めされます。それぞれは正方形で、辺の長さはAです。
   3つの許容範囲内で近傍点1点が探索されます。
   各許容範囲に近傍点が1点含まれる場合、グリッドは正しくキャリブレーションされています。



   c. 3つの垂直のセグメントは、次回の対話式サーチの際に参照用に使用されます。アルゴリズムが2番目のステップに戻ります。
5. `Calibrate`を呼び出してランドマークにアプローチします。

グリッドに過度のディストーションがある場合、期待したようには再構築が行われないことがあります。次のエラーが発生することがあります:

1. 許容範囲に近傍点が含まれていない(図の赤色の正方形)。
2. 許容範囲に複数の近傍点が含まれている。
3. 許容範囲内の点が適切でない。例えば、点が斜めに繋がっている(図の赤色の点)。



高度な機能

視野キャリブレーションのモデルは次のパラメータを使って調整できます：

センサーの幅と高さ

sensor width(センサーの幅) とsensor height(センサーの高さ) で論理的な画像サイズをピクセル単位(常に整数) で設定します。

視野の幅と高さ

field-of-view (f-o-v) width(視野の幅) とheight(高さ) で実際の画像サイズを長さの単位で設定します。すなわち、長方形の大きさがワールド空間での画像エッジに相当します。これらの値は次の計算式によってピクセル解像度に関連付けられます：

$$\text{視野幅} = \text{ピクセル幅} * \text{センサー幅}$$
$$\text{視野高さ} = \text{ピクセル高さ} * \text{センサー高さ}$$

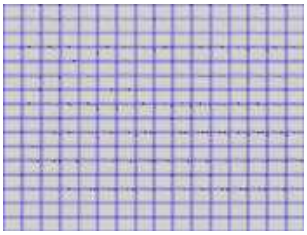
または

$$\text{センサー幅} = \text{視野幅} * \text{水平解像度}$$
$$\text{センサー高さ} = \text{視野高さ} * \text{垂直解像度}$$

ピクセルは正方形であるという前提で、デフォルトではピクセル高さは指定しません(ピクセル幅 = ピクセル高さ) 。

スケール

比率



異方性のアスペクト比

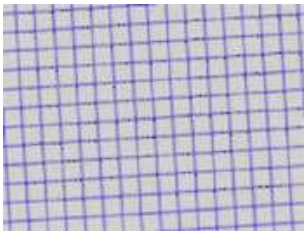
中心の横座標と縦座標

center abscissa(中心の横座標) (x) とordinate(縦座標) (y) は、画像の原点を示します(ワールド

座標 (0,0)) 。デフォルトは画像の中心です。

スキュー角

skew angle(スキュー角) は、リアルワールドの参照フレーム(X軸) と画像エッジ(水平) によって形成される角度です。デフォルトはスキューなしです。

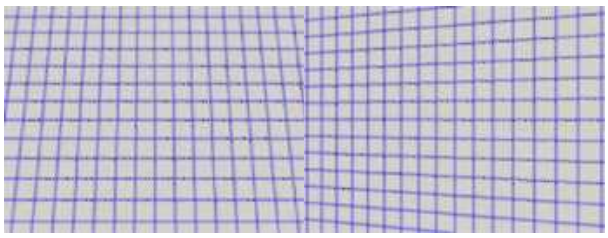


スキュー角

> **注記：** *ピクセルが正方形でない場合は、`EWorldShape`オブジェクトによってワールド空間とセンサー空間間の変換を行えます。*

X/Yの傾斜角度

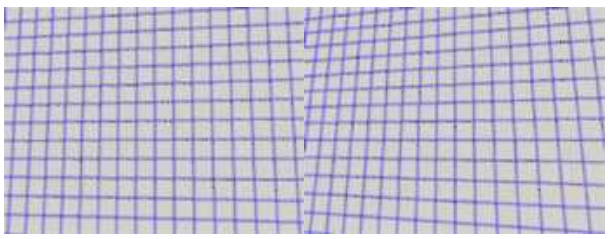
XとYの傾斜角度は、観察面の方向を表します。この角度は、Z軸が光学軸に平行になるようにX/Y軸まわりに回転させた角度に相当します。



傾斜X / 傾斜Yの角度

パースペクティブの強度

perspective strength(パースペクティブ強度) ではパースペクティブ効果の相対的な程度を設定します。焦点距離が短いほど、この値は大きくなります。

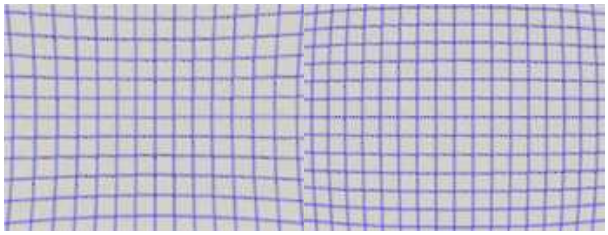


弱い / 強いパースペクティブ

ディストーションの強度

Distortion strengthおよび`GetDistortionStrength2`により、画像の角のラジアルディストーションの

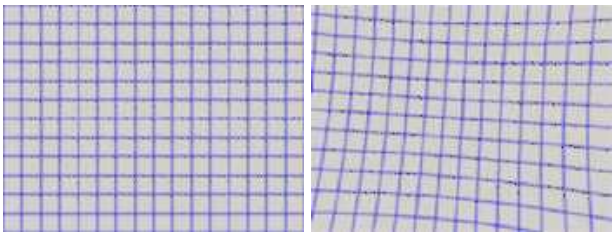
相対的な程度が設定されます。これは、ディストーションがある場合とない場合の画像の対角線の長さの比率です。



正 / 負のディストーション

オプションの組み合わせとして表されるキャリブレーションモードはCalibrationModesからアクセスできます。

キャリブレーション係数の効果



キャリブレーション係数なし / 全係数の組み合わせ

画像を逆ワープする

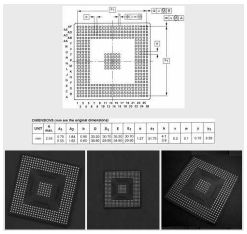
EWorldShapeオブジェクトは視野キャリブレーションのコンテキストを管理します。このオブジェクトはワールド座標(物理的な単位)とセンサー座標(ピクセル)の関係を示し、画像形成プロセス固有のディストーションを明らかにします。

画像キャリブレーションは定量的測定アプリケーションの重要なプロセスです。それにより、検査対象のアイテム上で、画像上の点の位置(ピクセルインデックス)とその点のリアルワールドでの実際の位置との関係を構築します。

キャリブレーションを設定するには、キャリブレーションモデルのキャリブレーションパラメータを明確に指定するか、あるいは分かっている点(ランドマーク)のセットまたはキャリブレーションターゲットを使用します。

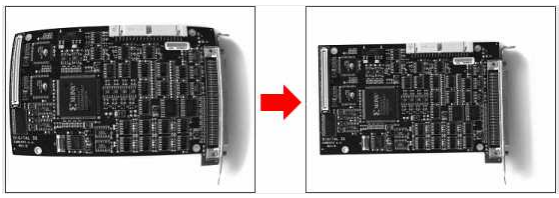
キャリブレーションの目的は2とおりあります:

- 検査対象アイテムのあらゆる絶対測定について一度だけ設定することで、どの観察条件(視野での部品の場所、レンズ倍率、センサー解像度など)でも対応できるようにするため



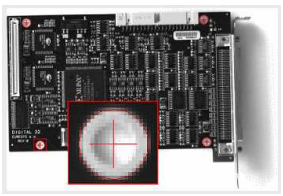
単一モデルと複数の観察条件

- イメージングプロセスに関係するいくつかのディストーション(パースペクティブ効果、光学収差など) を補正するため



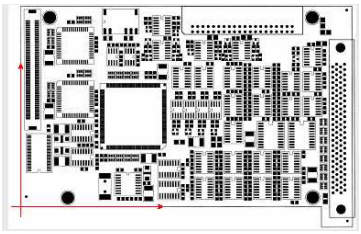
画像ディストーションの除去

画像のピクセルインデックスは通常は整数ですが、サブピクセルメソッドを使用する際には小数の値となることもあります。これは多くの場合、画像を処理したり既知の特徴点の位置を特定する際に取得されます。このような値はセンサー座標と呼ばれます。



センサー空間内の特徴点

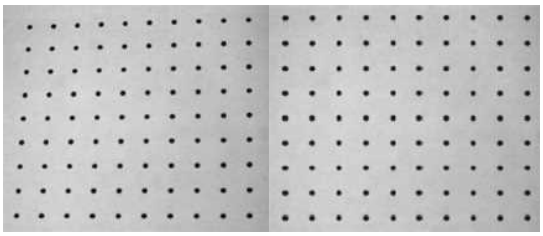
検査対象アイテム上の点の位置を示すワールド座標は、適切な長さの測定単位で表されます。
ワールド座標は実際の寸法で、多くの場合、設計図や機械的な測定値から集められます。
これには定義するための参照フレームが必要です。



ワールド空間の参照フレーム

逆ワープ

画像の逆ワープにはUnwarp、SetupUnwarpおよびUnwarpAfterSetupを使用します。
逆ワープの前にルックアップテーブルを使用すると処理時間が短縮されます。

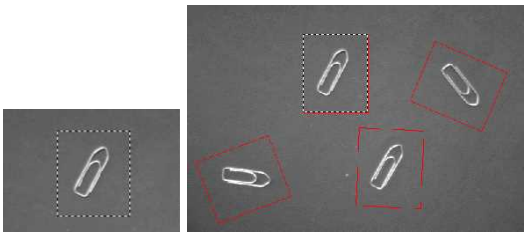


ディストーション画像と逆ワープした画像

3.3. EasyMatch - 領域パターンの一致

EasyMatch はパターンを学習し、完全な一致を見つけます：

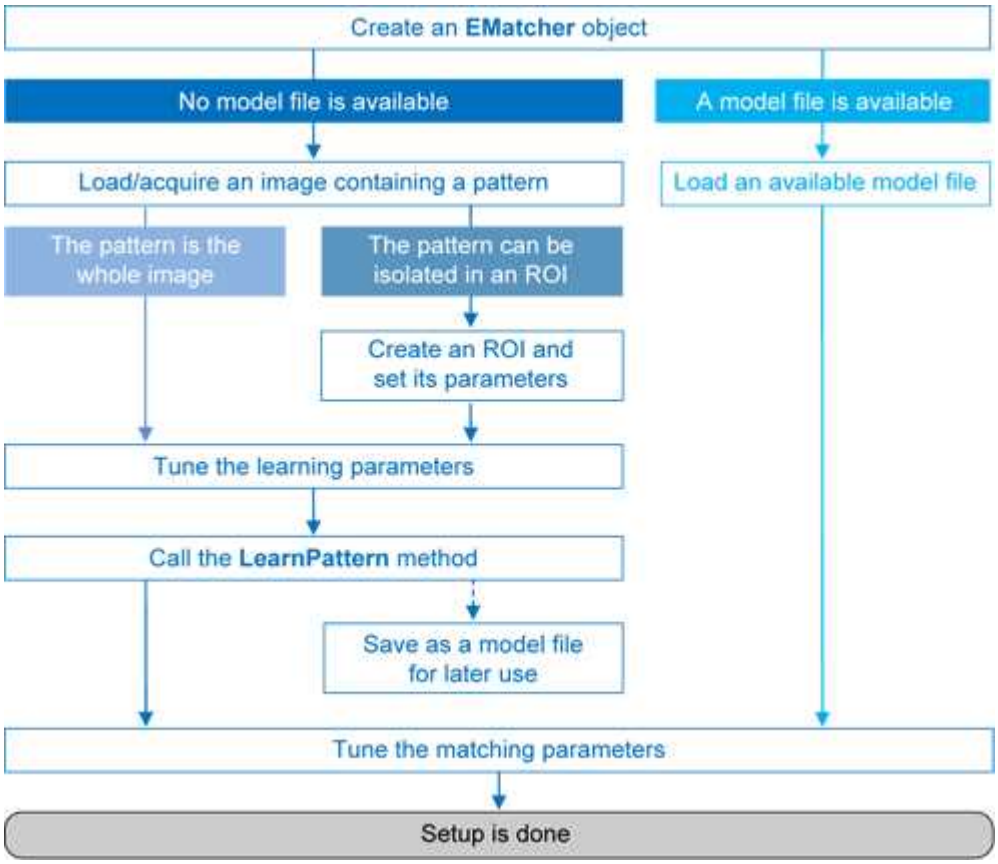
1. 対象となるオブジェクトを含むROIを定義して、パターンを学習します。
   このROIは、オブジェクトを含む複数の画像から繰り返し学習することで作成されます。
2. パターンが確実に見つかるようにパラメータが調整されます。
3. これで画像から1つまたは複数のパターンの出現を探すことができます。平行移動、回転またはスケーリングされているパターンも探索されます。



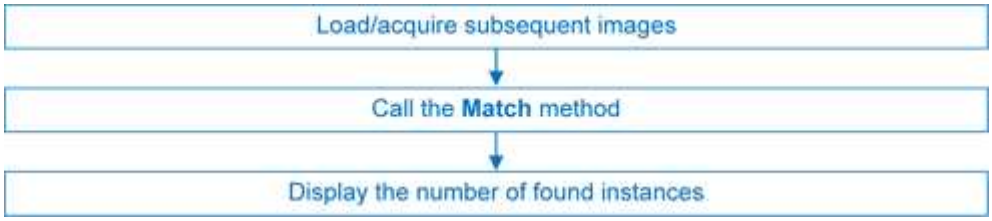
パターンの学習とマッチング

ワークフロー

学習のワークフロー



マッチングのワークフロー



学習プロセス

探索対象のパターン / ROIを含む画像を選択して、LearnPatternを呼び出します。

結果のパターンは後で使用するために保存することができます。このプロセスを繰り返して複数のパターンを検出、保存できます。

最良のパターンの特性

- **再現性がある:** パターンが平行移動、回転またはスケーリングされているかをあらかじめ知っておく必要があります。
- **位置検出の対象となるオブジェクトが表されていること。**

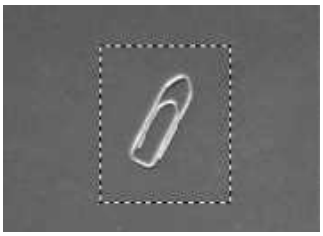
  条件として:

  - 照明条件に関係なく外観が同じであること
  - 対象となる部品の固定された位置にあること
  - 固定されており、形状が変化しないこと
- **縮小または拡大スケーリングでもコントラストが良好であること。** 遠くから見た画像でも、縮小された画像でも、コントラストがはっきりと分かる状態でなければなりません。
- **測定の自由度に影響されないこと。** 例えば、水平方向の白黒の縞模様のパターンでは、水平方向への平行移動を検出できません。また、はめ歯歯車の場合は大きい回転を測定できません。
- **背景に他の要素が写っていないこと。** ROI内のパターン付近のオブジェクトが変化した場合、「無視」ピクセルまたはマスクを使ってこの領域を無効化する必要があります。
- 前景と背景の輝度が見えるように**オブジェクト周辺にコントラストがはっきりしたマージンがあること**。

パラメータのカスタマイズ

処理時間を短縮するためにパラメータを調整することができますが、選択された領域全体でマッチングが行われるため、EasyFindよりも時間がかかります。

- `DontCareThreshold`: 無視領域が必要な場合、そのピクセルの値は`DontCareThreshold`未満でなければなりません。
  背景全体を無視してもいい場合、`DontCareThreshold`を正しい閾値に調整するだけで行えます。
  あるいは無視領域が閾値パターン画像と関連していない場合は、`DontCareThreshold`を1に設定し、無視領域に属する全ピクセルを黒 (値0) に設定します。
- `MinReducedArea`: 許容できるタイムパフォーマンスを得るため、EasyMatch はパターンをサブサンプリングします。このパラメータは、対象となるパターン画像の最小ピクセル数を規定します。この値が小さいとマッチングプロセスが速くなりますが、信頼性の低い結果が出力されます。多くの場合、デフォルト値 (64) が妥当な妥協点です。
- `FilteringMode`: 画像に鮮鋭なグレーレベルの遷移がある場合、通常の均一カーネルではなくローパスカーネルを選択することを推奨します。



パターンの学習

マッチングプロセス

新しい画像ごとに1または複数のパターンの出現を探します。平行移動、回転またはスケーリングされているパターンにも1回の関数呼び出しで対応します:

- Match: ターゲット画像 / ROI を引数として受け取り、希望するパターンの出現を特定します。

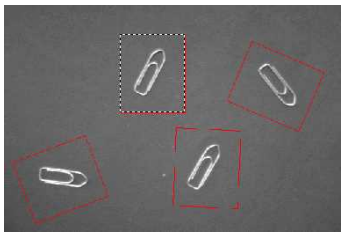
次のパラメータを設定できます:

- 回転範囲: MinAngle、MaxAngle
- スケーリング範囲: MinScale、MaxScale
- 異方性スケーリング範囲: MinScaleX、MaxScaleX、MinScaleY、MaxScaleY

以下の関数はマッチングの結果を返します:

- NumPositionsは良好な一致の数を返します。良好な一致とは、あらかじめ指定された値(MinScore閾値)より高いスコアが出たときとして定義されます。
- GetPositionは、N番目の良好な一致の座標を返します。位置はスコアの降順に並べられます。

同じ画像で複数のパターンを一致させる場合は、パターンごとにEMatcherオブジェクトを作成します。



パターンマッチング

高度な機能

このプロセスを高速化する最良の方法は、回転とスケーリングを最小限にして、探索対象の出現数を制限することです。

- 学習時間:
  - サーチ回数を最適化する: すべての位置で探索すると時間がかかり過ぎるため、さまざまなスケールのサーチシーケンスを実行できます(リダクション)。最も粗いリダクションの場合は、迅速に大まかな計算が行われます。位置特定精度を改善するためその後のリダクションは近い近傍で行われ、探索対象となる位置の数が大幅に減少します。位置精度は2^Kです。Kはリダクション数になります。

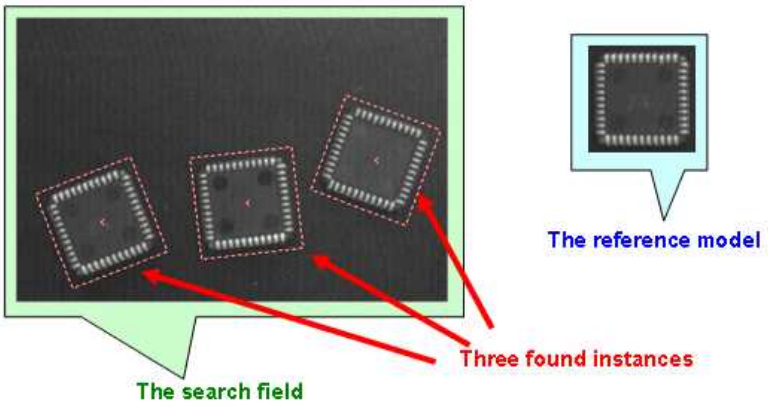
- MinReducedArea。大まかな位置特定のためにどれぐらい小さいパターンを使用できるかを示します。

- マッチング時間：
  - 相関モード（パターンと画像を比較する方法）：CorrelationMode。**標準、オフセット正規化、ゲイン正規化、完全正規化を選択できます：**相関は連続する色調値で計算されます。正規化により比較前にパターンのコントラストや輝度を自動的に調整することで、変化する照明条件に適切に対処します。
  - コントラストモード（コントラストの反転に対処する方法）：ContrastMode。照明の効果によってオブジェクトが反転したコントラストで表されることがあります。このような場合、反転したインスタンスを保持するかどうか、正の出現のみまたは負の出現のみ、あるいはその両方を一致するかを選択できます。
  - 最大位置（予想される一致の数）：MaxPositions、MaxInitialPositions。MaxInitialPositionsパラメータを使用すると、粗い段階で必要とされるより多くのインスタンスを考慮するようにEasyMatchに強いこともできます（この数は最終段階でMaxPositionsに達するように徐々に減少します）。
  - 最少スコア（これを下回る一致は偽と見なされて破棄されます）：MinScore、InitialMinScore。
  - サブピクセル精度：Interpolate。パターン計測の精度性が調整可能です（精度を落とすと処理スピードは速くなります）。デフォルトでは、各自由度の位置パラメータがピクセル単位の精度で計算されます。低い精度は強化できます。10分の1ピクセル精度まで可能です。
  - リダクションステップ数：FinalReduction。大まかな位置特定で十分である場合に、マッチングの速度を高めることができます。範囲は [0～NumReductions-1] です。
  - 非スクエアピクセル：GetPixelDimensions、SetPixelDimensions。画像が非スクエアピクセルで取り込まれた場合、回転したオブジェクトは斜めになったように見えます。この効果はピクセル縦横比を考慮に入れて補正できます。
  - DontCareThreshold値の下の「無視」ピクセル（相関スコアで無視されます）。パターンが長方形ROIに内接していない場合、閾値レベルより下のピクセル値を設定することによってROIを部分的に無視できます。テンプレートの部分がサンプルごとに変わっても、同じ特徴を使用できます。

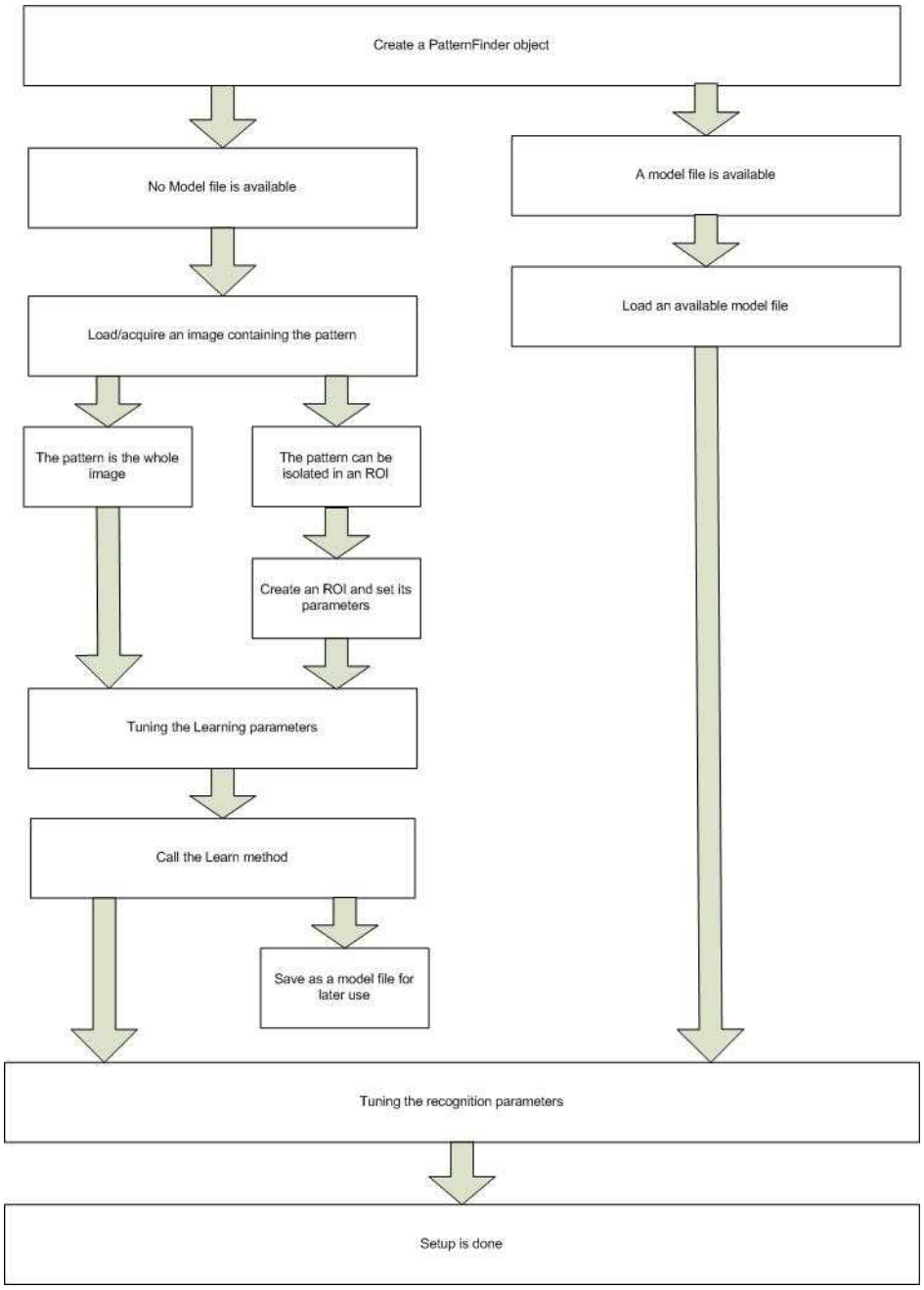
3.4. EasyFind - 幾何学パターンの一致

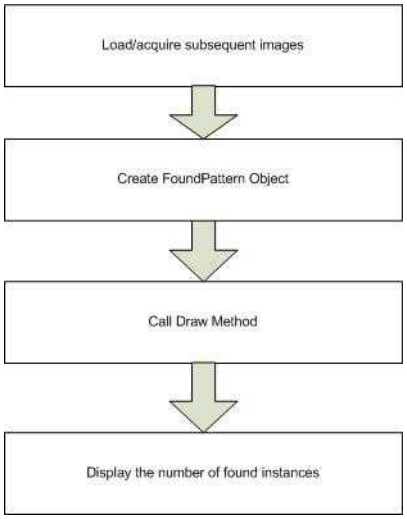
ワークフロー

EasyFindはパターンから参照モデルを学習します。学習されたモデルは、他の画像で類似するパターンを探してこのインスタンスについての情報を取得するために使用されます。
処理は迅速かつロバストで、さらにノイズ、不鮮明性、オクルージョン、欠落部分、照明の変化などの影響を受けにくいという特徴があります。


The reference model
Three found instances
The search field

ワークフロー

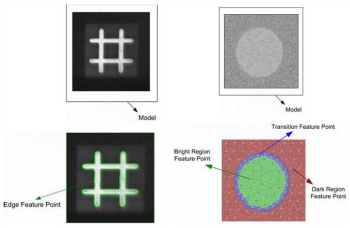

Create a PatternFinder object
No Model file is available
A model file is available
Load/acquire an image containing the pattern
Load an available model file
The pattern is the whole image
The pattern can be isolated in an ROI
Create an ROI and set its parameters
Tuning the Learning parameters
Call the Learn method
Save as a model file for later use
Tuning the recognition parameters
Setup is done



特徴点の定義

特徴点とは、座標 (X、Y) とタイプ(エッジ、遷移または領域) の組み合わせです。

EasyFindでは、サーチ領域内でインスタンスを見つけるために特徴点が使用されます。

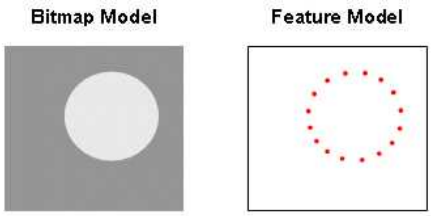


- **エッジの特徴点**: 2つの領域間でのグレーレベルの急激な変化は、サーチ領域内のその位置にエッジがあることを示します。
- **遷移の特徴点**: 2つの領域間でのグレーレベルの滑らかな変化は、その近傍の遷移領域を示します(上の例では青色の部分のドットに当たります) 。近傍のサイズは変更できます。
- **領域の特徴点**: だいたい均一なグレーレベルの2つの領域を識別します:
  - 暗い領域(上の例では赤色の部分のドット群に当たります) 。
  - 明るい領域(緑色の部分のドット群に当たります) 。

学習プロセス

EasyFindはさまざまなタイプのパターンに対応しています(均一なエッジ、細い構造、コントラスト領域) 。
EasyFindは学習プロセスの際、パターンのビットマップ表示から抽出された全特徴点のセットである特徴モデルを自動的に計算します。

この特徴モデルがあれば、EasyFindでサーチ関数を開始できます。その一方で、最適なモデルをユーザーが作成することもできます。



最適なモデルは、探索対象のパターンのタイプに応じて異なります。

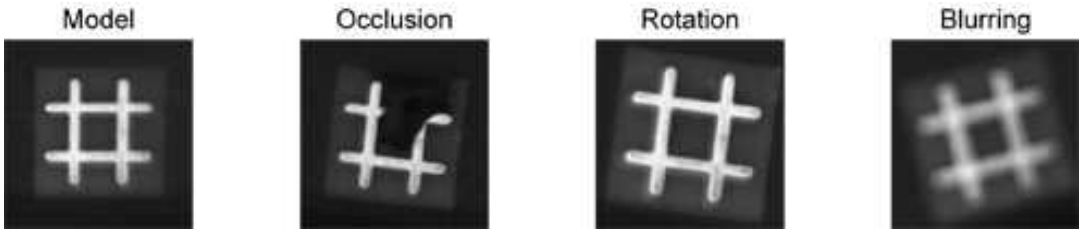
一貫したエッジ

モデルは、鋭いエッジとは対照的でなければなりません。また、モデルがサーチ領域の残りの部分と大きく異なることが必要です。スケーリングまたは回転にも対応し、不鮮明性、ノイズ、オクルージョン、照明の変動に対し非常にロバストです(ポイントバイポイントスコアにより、探索段階でロバスト性が高まり計算時間が短縮されます)。

適したモデル: サーチ領域全体でインスタンスごとにほぼ同じ場所にある均一なエッジ、鮮鋭なコントラスト遷移、適切に定義されたエッジに囲まれた領域を持つモデル。

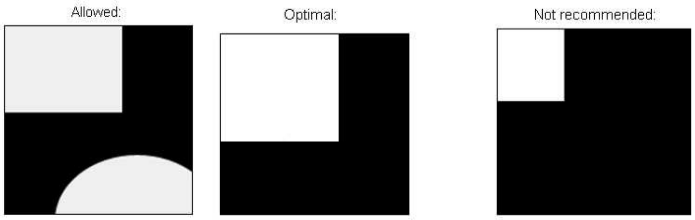
と同様の点を選択します `EPatternFinder::ContrastMode` プロパティ:

- PointByPointNormal: ポイントが同じコントラストの極性を共有する場合
- PointbyPointInverse: ポイントが反対のコントラストの極性を示す場合
- PointByPointAny: それぞれのコントラストの極性にかかわらず



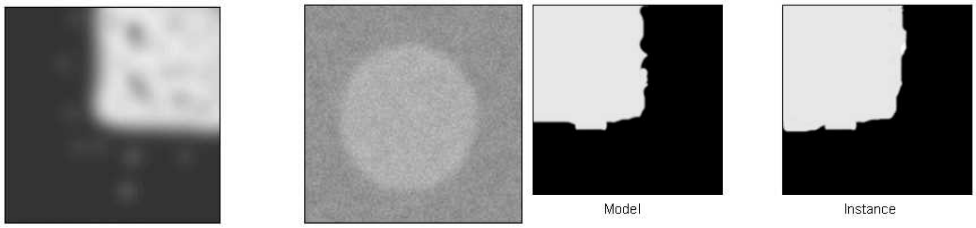
コントラスト領域(領域の特徴点と遷移の特徴点によって定義される)

理想では、モデルには明るい領域が50%、暗い領域が50%含まれていることが望ましいとされています。複数の暗い / 明るい領域を定義することもできます。



スケーリングまたは回転には対応していません。不鮮明性、ノイズ、照明の変動に対してロバストです(オクルージョンは不可)。

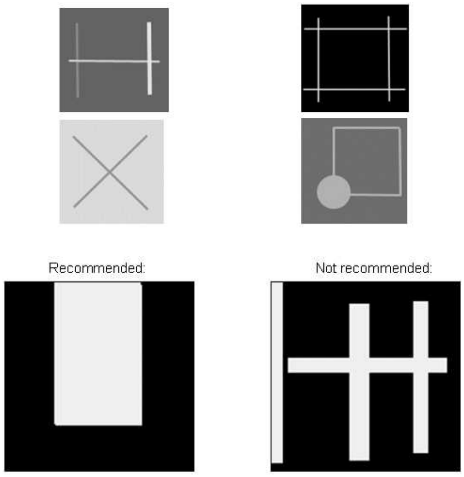
適したモデル: 均一な遷移またはエッジ、あるいは遷移またはエッジによって囲まれた複数の領域を持つモデル。



細い構造(エッジの特徴点によって定義される)

スケーリングまたは回転に対応し、不鮮明性、ノイズ、オクルージョン、照明の変動に対してロバストです。細い要素および領域間のエッジが均一で、細い要素のコントラストがすべて同じでなければなりません。

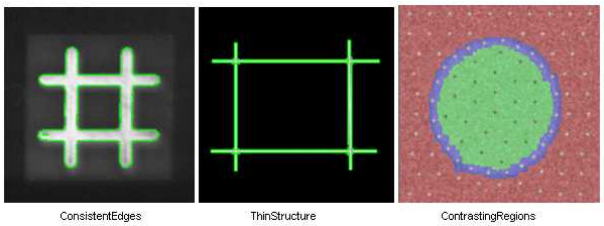
適したモデル: 細い要素を含むモデル。



学習したモデルが適切であるかチェックする

EasyFindでは、`PatternFinder`オブジェクトの`DrawModel`メソッドを用いて、抽出された特徴点をモデルに描画できます。

下の例では、エッジの特徴点が均一なエッジと細い構造には緑色の点、コントラスト領域には十字で示されています。

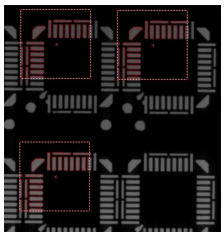


サーチプロセス

サーチプロセスは、いくつかのパラメータを設定してコンフィギュレーションファイルに保存することで最適化することができます。このファイルを読み込んで実行してみて、EasyFindで何が見つかったのか、報告された情報で確認してください。

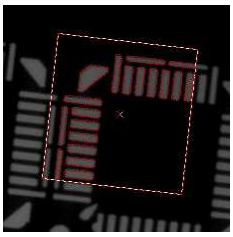
予想されるインスタンスの最大数

EasyFindが返すインスタンスの最大数を設定します。この例では最大数は3です。



角度とスケール(細い構造および均一なエッジのパターンタイプ)

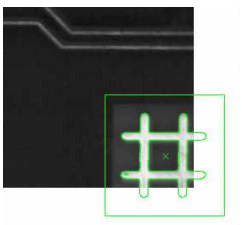
範囲にはバイアスと許容値があります。例えば、角度バイアスが20°、角度許容値が5°の場合、EasyFindは学習したモデル (20° ± 5°) に対して15°～25°の角度のインスタンスを返します。



高度な機能

部分的なパターンを探索する

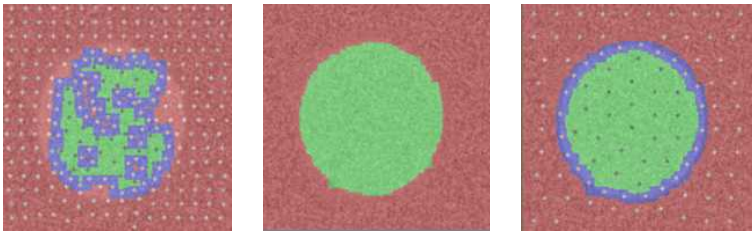
EasyFindでは、サーチ領域の範囲が > 0ピクセルに設定されていれば、部分的にサーチ領域外にある細い構造や均一なエッジのインスタンスの位置を特定できます。



パラメータを調整する

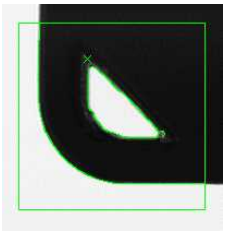
全モデルで次のパラメータを調整できます：

- モデルをプレビュー表示するためのモデル描画を用いてライトバランスを調整してモデルをパターンの対象部分に適合させ、モデルをもう一度学習します。



悪い抽出物(左) - 光のバランスを調整(中央) - 良い抽出物(右)

- グレーレベル閾値を設定して(ライトバランスが上書きされます)、モデルをもう一度学習します。
- コーナーやホールなど、モデルの特別な場所にピボットを移動します。ピボットとは、EasyFindがインスタンスを見つけたときに返す位置です(デフォルトは中心)。

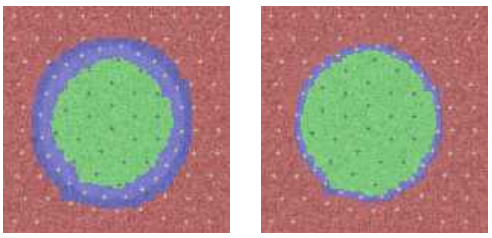


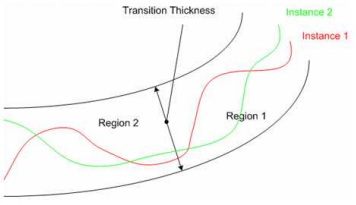
▶ 細い構造には次のパラメータを調整できます：

- 自動(細い要素とその近傍領域のコントラストが同じ)
- 細い要素が近傍より暗い
- 細い要素が近傍より明るい

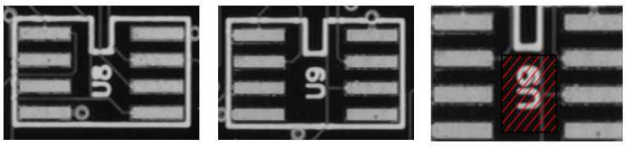
▶ コントラスト領域には次のパラメータを調整できます：

- 遷移厚さ: 遷移の特徴点は、遷移が生じている場所を示す*遷移帯*(下の例の青色の領域)内にあります。遷移厚さパラメータは、さまざまなインスタンスの境界がすべて遷移帯に入るように調整する必要があるため、インスタントの最大の変動と同じように設定することを推奨します。下の2つの例は、厚さと推奨される厚さを示しています：





- **無視される領域**：ゼロの値は無視される領域を示します。255の値は考慮される領域を表します。例えば、モデル中心のテキストがインスタンスとは異なる場合、モデルのこの部分から特徴点を抽出しないように指定できます。



3.5. ゴールデンテンプレート検証 (EChecker)

EChecker には、学習と検査の2つの処理ステップが必要です。この2つの演算は全く異なるもので、個別のアプリケーションに分けてプログラミングできます。

学習では一連の参照画像の前処理を行います。各ピクセルの許容範囲を計算し、EasyObjectで使用できるように2枚の閾値画像に保存します。ゴールデンテンプレートが適切であるかテストして、必要に応じて学習プロセスパラメータを調整します。学習は1回行えば繰り返し行う必要はありません。学習結果は後から使用できるようにゴールデンテンプレートにアーカイブされます。

検査には、画像処理、位置合わせおよび正規化、許容範囲外のピクセルの検出、品質チェック、必要であれば検査パラメータの調整が含まれます。

以下では、学習および検査ステップで使用されるAPI関数について紹介します。

学習

参照画像が前処理されてピクセルの許容範囲が計算され、EasyObject(レガシー) で使用できるように2枚の閾値画像に保存されます。学習結果は後から使用できるようにモデルファイルにアーカイブされます。

参照画像がディスクに保存済みであるのか、取り込まれた画像をオンザフライ方式で処理するのかによって、2とおりの演算モードがあります。

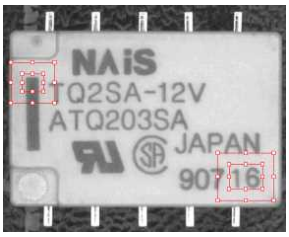
いずれの演算もオンライン処理中は使用できません。

モデルを定義するには、次の手順で演算を実行します：

1. 最初の参照画像で1つまたは2つのROIを配置し、ロケーションパターンを定義します(基準マークまたはランドマーク) 。
   移動の自由度を定義するためにこれらのROIは他の2つのROIに囲まれており、パターンマッチング用のサーチエリアとして使用されます。ROIはハンドルをドラッグすることで対話形式で位置決めできます。
   EChecker ではドラッグ操作が制御されます。
   - Draw: ROIおよびハンドルをグラフィックとしてレンダリングします。
   - HitTest: いずれかのハンドル上にあるカーソルを検出します。
   - Drag: ハンドルを移動します。

これらの関数は3つすべてのROI(パターン、サーチ、検査対象)に適用されます。その際、ユーザーは次のことにすぐに気づきます:

- パターンROIをドラッグすると、許容値が一定になるように、対応するサーチROIが自動的に調整されます。
- サーチROIをドラッグすると、パターンに対して対称的になるようにそのサイズが調整されます。
- サーチエリアを変更すると、検査対象のROIが画像内で最も大きい使用可能な大きさに設定されます。

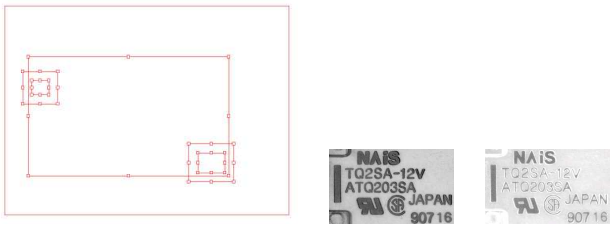


パターンROIとサーチROI

- 学習する画像がディスク上に保存されている場合、ファイル名のリストを登録し、後で単一コマンド内で学習を実行する際に使用できます(*バッチ学習*、*オンザフライ学習*と対照的)。
- 検査される領域を区切るために別のROIが定義されます。この領域には固定された部分(基準マークとともに移動)のピクセルのみが含まれるようにし、背景が含まれないようにします。

検査対象ROI

- 全画像が統計の学習を使用して処理および平均化されます。視野内での検査対象部分のずれに対処するために位置合わせが、全体的な照明の変化に対処するためにグレーレベル正規化が行われます。
- 理想では少なくとも16枚の画像を学習パスで使用し、低い / 高い閾値画像を作成します。ユーザーはグローバルな許容値パラメータを使って許容範囲を強化または緩和できます。`Learn`(ELearningMode_Ready)を呼び出します。属性`RelativeTolerance`によって許容範囲が調整されます。
- モデルをファイルに保存する際には、さまざまなROIの位置、基準マークのパターン画像、グレーレベル正規化パラメータ、閾値画像などの関連情報もまとめて保存できます。



EChecker**モデルのコンポーネント**

参照画像の信頼性チェック

アライメントROIが設定されたら、その基準マークのロケーションの信頼性をチェックする必要があります。最良の方法は、学習画像を読み込んでそれを表示させ、`Register`メンバ関数を使ってその中でパターンの位置を確認するやり方です。それで位置確認が行えない場合は、それぞれの問題に応じたさまざまな補正方法があります：

- パターンの選択が不適切な場合：より安定した他のROIを定義してください。
- サーチエリアが狭すぎるため、パターンが途中で切れている場合：サーチエリアを拡大してください。
- 代表的な画像として適切でない(欠陥がある)場合：学習セットから削除します。

> **注記：** *メンバ`Register`を最初に呼び出す際、EasyMatchコンテキストのアライメントの`Learn`メンバが呼び出され、パターンで学習が実行されます。メンバ`Learn`(ELearningMode_Reset)が呼び出されない限り、これらのパターンは後続のアライメント演算に使用されます。最初に使用される画像は、アライメントパターンの定義とコントラスト測定の両方の基準として使用されます。この画像は母画像と呼ばれます。*

学習画像がディスクに保存されている場合：ファイルが正常に読み込まれて参照画像と見なされるたびに、`EChecker`オブジェクトによってリストにファイルのパス名が追加されます。その後、すべてのファイルを単一コマンド内で処理できます。

さらなる学習

ROIの位置決めとパターン学習が実行された後(Register 演算)、2つのパスがまだ必要です:

- 部品画像の中心傾向を表す、理想的なノイズフリーの画像を計算するには**「平均」パス**が必要です。
  各画像で位置調整および正規化を行います(Register)。演算が正常に行われたら(適切なパターン位置)、Learn(ELearningMode_Average)を呼び出してすぐに処理を行うか(オンザフライ学習)、または
  AddPathNameで遅延処理(バッチ学習)を行います。
- **「偏差」パス**は、平均画像を基準とした変動を測定します。
  各画像で位置調整および正規化を行います(Register)。演算が完了したら、
  Learn(ELearningMode_RmsDeviation)(大きな偏差を強調)または
  Learn(ELearningMode_AbsDeviation)を呼び出して、すぐに処理を行います(よりロバストで、たいていの状況で推奨)。

原則として、この2つのパス中に表示される画像は同じでなければなりません。
画像をアーカイブしたくない場合は、2つの画像セットを使用できます(オンザフライ学習)。この2枚の画像が同じサイズである必要はありません。

学習セットサイズとして少なくとも16枚の画像を推奨します。

あるいは、BatchLearnを呼び出して、ファイルリストに追加されている全画像に2つのパスを実行することもできます。

閾値化の調整

Learn(ELearningMode_Ready)。属性RelativeToleranceを変更して許容範囲を調整できます。

検査

検査手順は簡単です。サンプル画像がモデルファイルに合わせて位置合わせされ、グレーレベルが正規化されます。

> **注記:** *検査対象ROIは母画像上になければなりません。パターンとサーチエリアと同じように対話形式で位置決めできます。このROIも他のROIと同時に設定できます。探索許容値を変更すると、検査対象ROIが使用可能な最も大きい範囲にリセットされます。*
> *EChecker*によって検査時にグローバルな*RelativeTolerance*属性が調整されることがあり、そのときは*EasyObject(レガシー)*で使用された低い/高い閾値画像が変更されます。
> *学習後に変更できるのは*EChecker*パラメータのみです。*

検査する画像...

... 高い/低い閾値画像...

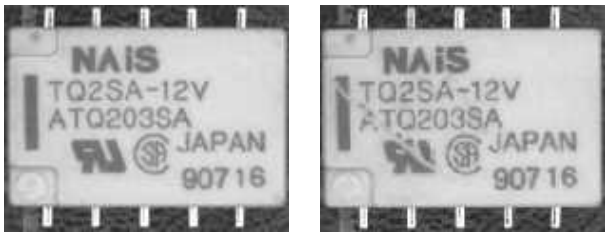
... 検出されたブロブ

結果画像はブロブ解析のためにECodedImageオブジェクトに渡されます。このオブジェクトは近傍ピクセルをグループ化してブロブを形成し、小さいブロブ(通常はノイズ)を破棄して、幾何学的な特徴(ロケーション、サイズ、方向など)を測定します。EasyObjectを参照。

それからこの画像は低い/高い参照画像と比較されます(サンプルとテンプレートがピクセル単位で比較され、許容範囲に入らないピクセルが検出されます)。欠陥ピクセルは標準のEasyObject機能によって対処されます。

ECheckerを用いて比較する

EChecker はテンプレートとサンプル間の目に見える相違点を検出し、重大な相違点が強調された欠陥マップで欠陥を報告します。その後、EasyObjectのブロブ解析ツールによって欠陥の位置を特定し、その大きさ、方向、明るさなどをもとに分類します。このような処理が最適であるのは、検査対象となるアイテムが固定されており(形状が変化しない)、照明が均等で、画像の視覚的な再現性が確保されている場合で、このようなときに点と点の比較が意味をなします。

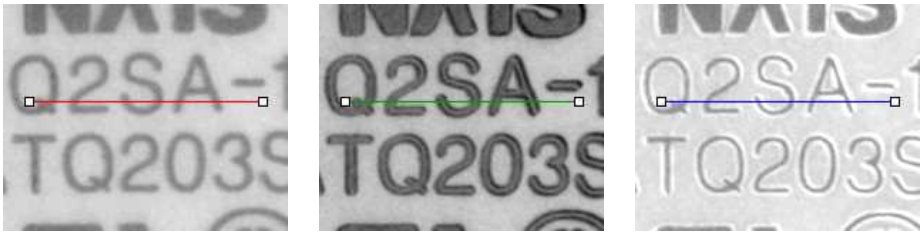


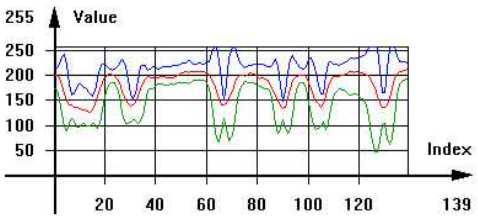
参照画像と欠陥のあるサンプル画像

統計の学習

複数の参照画像を取り込むことで、標準的なグレーレベルの変化や許容範囲の評価が最適化されます:

- 同じ部分の連続画像で何も変化がない場合(静的テスト)、ノイズ分布に相当するグレーレベル分布が得られます。
- 欠陥のない異なる部分の連続画像では、部分による変化が現れます。





許容されるグレーレベル範囲

画像比較

複数の画像を比較する最も有効な方法は、EChecker を使って画像集合を比較し、各ピクセルに許容範囲を設定することです。

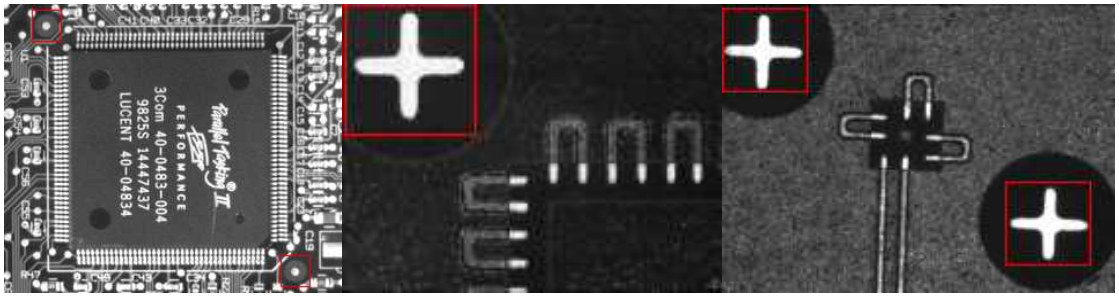
あるいは、2枚の画像を減算して(算術 / 論理演算を用いて) 絶対値を取得し、その後でその差を閾値化して異なるゼロ以外のピクセルを強調します。

EChecker の機能 :

アライメント

パターンマッチングでは、検査対象の部分の平行移動、回転および / またはスケーリングといった位置変化が測定されます(EasyMatch 章の説明を参照) 。
パターンが検出されると、検査対象の部分が参照画像と同じ位置になるように画像が位置決めされます。
1つのマッチングパターンで平行移動を容易に測定できます。
2つのマッチングパターンを使うと回転の測定精度が高まります。

基準マークがある場合は、正確で再現性のあるロケーションを得るためにこれをランドマークとして使用できます。
EChecker は1つまたは2つのマッチングコンテキストを内部でホストします。

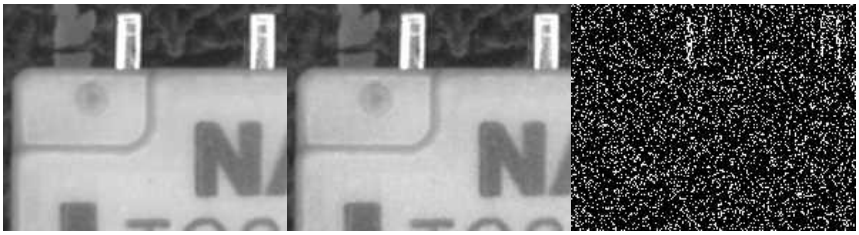


基準マークを用いた位置合わせ

このアライメント手法には3つの短所があります：

1) 避けられない変化

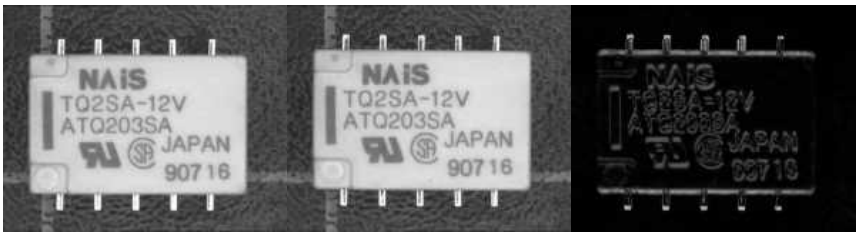
ノイズによって同じ画像に色や形状のわずかな変化が生じるため、許容値を設ける必要があります。



ノイズがある2枚の画像の厳格な同等性の比較
結果画像の黒のピクセルは同等でないピクセル。

2) 検査対象部品の位置にあまり再現性がない

わずかなずれでも比較されたピクセルどうしが一致しなくなり、その結果は特にエッジ周辺で顕著になります。



位置がずれている画像の比較

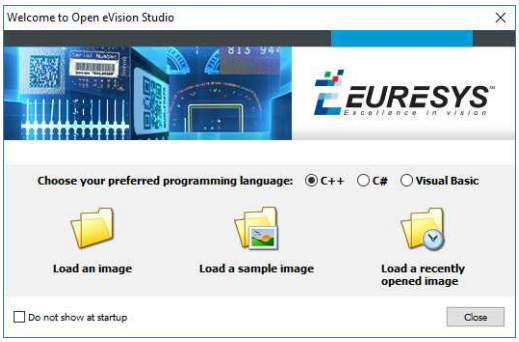
3) 照明を分離できない

異なる照明条件の画像の比較

4. Open eVision Studio 使用方法

4.1. プログラミング言語の選択

初めてOpen eVision Studioを起動すると、次の開始画面が表示されます。



1. プログラミング言語を選択します。

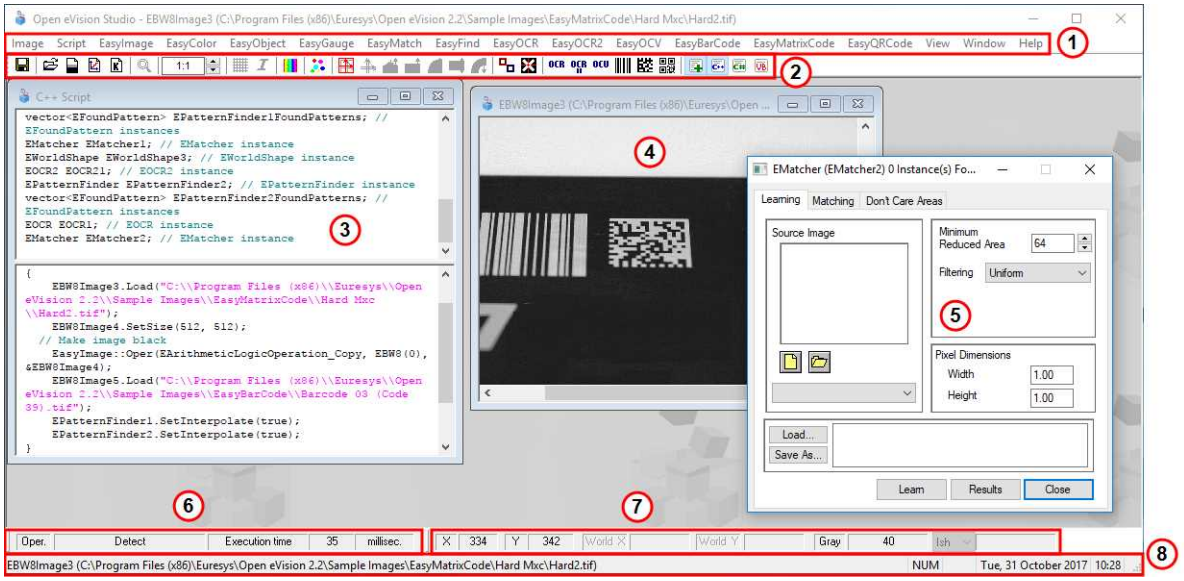
選択内容が保存され、次回Open eVision Studioの起動時にプログラミング言語が自動的に選択されます。

> **注記：** *プログラミング言語を変更すると、スクリプトウィンドウにあるスクリプトは自動的に削除され、ウィンドウの内容はリセットされます。*

2. **ロード**ボタンの1つをクリックして、後で処理するために1つまたは複数のイメージをロードします。
3. 次回Open eVision Studioの起動時にこのような開始画面を非表示にするには、**起動時に表示しない**チェックボックスをオンにします。

*このようなウェルカム画面にいつでもアクセスしてこの設定を変更するには、**ヘルプ**> **ウェルカム画面**に移動します。*

4.2. インターフェイスのナビゲート



Open eVision Studioグラフィカルユーザーインターフェイス(GUI) は、以下のように構成されています。

1. *メインメニューバー*では、すべてのライブラリの機能とツールにアクセスできます。

Open eVision Studioはライセンスを必要とせず、すべてのライブラリをテストすることができます。もちろん、自分のアプリケーション内のOpen eVision Studioからコードをコピーしても、必要なライセンスがない場合は、実行時に「ライセンス不足」エラーが表示されます。

2. *メインツールバー*は、イメージ、図形、ゲージ、バーコード、マトリックスコードなどの主要なOpen eVisionオブジェクトにすばやくアクセスできます...

3. *スクリプトウィンドウ*には、Open eVision Studioで実行したアクションに対応する、選択したプログラミング言語のコードが表示されます。このコードは、いつでも自分のアプリケーションに保存またはコピーできます。

4. *イメージウィンドウ*には、ライブラリとツールを使用して処理できる開いたイメージが表示されます。

5. *ツールウィンドウ*では、使用可能なすべてのツールを簡単に設定できます。対応する設定は自動的にスクリプトウィンドウに追加され、簡単に再利用できます。

ほとんどのツールウィンドウは浮動しており、簡単にOpen eVision Studioメインウィンドウの外側に移動して画面サイズをより有効に活用できます。

6. *実行タイムバー*には、選択した機能の実行に要した正確な時間(ミリ秒単位またはマイクロ秒単位) が表示されます。この正確な測定は、アプリケーションのパフォーマンスの評価に役立ちます。

7. *カラーツールバー*は、画像上のカーソルのX座標およびY座標、および対応するピクセル値などの現在の情報を表示します。

8. *ステータスバー*には、アクティブなイメージファイルのパスなど、アプリケーションに関する一般的な情報が表示されます。

4.3. 画像上のツールの実行

ステップ1：ツールの選択

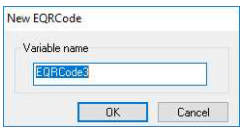
通常、Open eVision Studioを使用する際の最初のステップは、画像に使用するライブラリとツールを選択することです。

そのためには：

1. メインメニューバーで、使用するライブラリをクリックします。
2. 使用するツールをクリックします。

*すべてのライブラリ(EasyImage、EasyColor、EasyGaugeを除く) は、**New Xxx Tool**という名前のツールを1つだけ公開します。これらのライブラリの中には、追加機能を公開しているものもあります。*

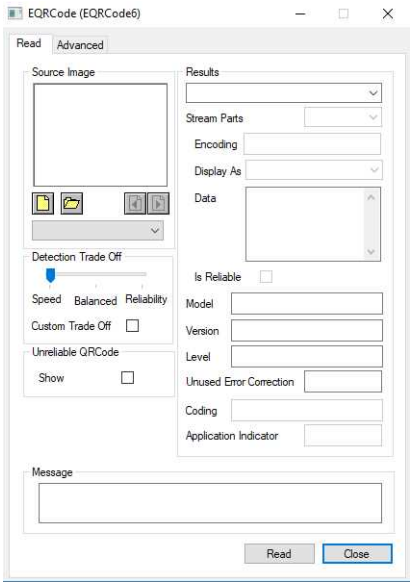
3. ダイアログボックスで、自動的に作成され、処理結果を含む変数の**変数名**を入力します。



EasyQRCodeの変数作成ダイアログボックスの例

4. **OK**をクリックします。

選択したツールダイアログボックスが開きます。



EasyQRCodeの変数作成ダイアログボックスの例

次のステップは「ステップ2：画像を開く」見開きページのです。

ステップ2: 画像を開く

ライブラリとツールを選択したら、このツールを適用するのに画像を開く必要があります。

選択したツールダイアログボックスの**ソース画像**エリアで次の操作を行います。

1. 次の画像を開きます。
   - **画像を開く**ボタンをクリックし、コンピュータ上のイメージを1つまたは複数(SHIFTとCTRLを使用して)選択します。
   - またはドロップダウンリストですでに開いている画像(またはROIがある場合はそのいずれか)を選択します。

   注記: *適切なファイル形式(JPG、PNG、TIFFまたはBMP)の画像のみを選択でき、ライブラリに応じて8ビットまたは24ビットの画像を選択できます。*

2. 複数の画像を選択した場合は、**前を読み込む**または**次を読み込む**ボタンで画像を有効にします。

ツールは読み込まれた画像に自動的に適用され、この段階でツールのデフォルト設定に基づいて結果が表示されます。

次のステップは「ステップ3: ROIの管理」下です。

ステップ3: ROIの管理

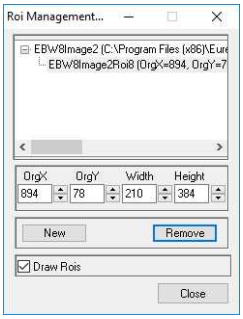
ほとんどの場合、1つまたは複数の明確に定義された画像の矩形部分、またはROI(関心領域)だけでなく、読み込むオブジェクトまたは処理しない画像全体の処理時間を短縮、またはシングルアウトします。

Open eVisionでは、ROIは画像に添付され、親画像が利用可能である限り存在します。

ROIの作成

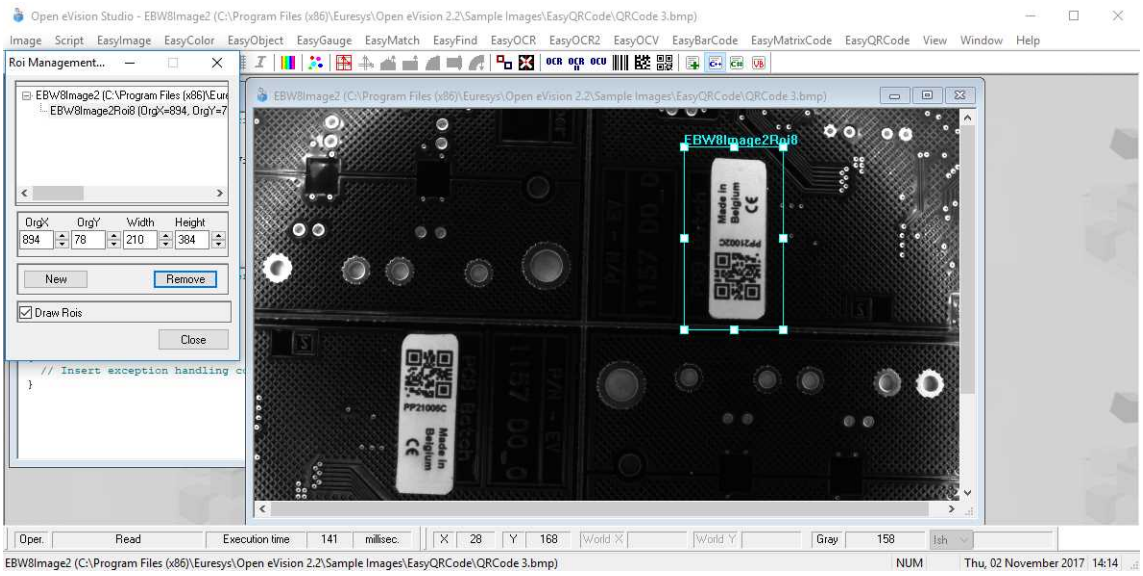
1. 次の画像を開きます。
   - 画像がすでに開いている場合は、対応するイメージウィンドウをアクティブにします。
   - 画像がまだ開いていない場合は、メインメニューの**画像**>**開く**...をクリックして開きます。
2. ROIを作成するには、メインメニュー**画像**> **ROI管理**...を選択します。

ROI管理ウィンドウが以下のように表示されます。



3. ツリー内の画像を選択します。
4. **新規**ボタンをクリックします。
5. ダイアログボックスで、新しいROIの**変数名**を入力します。

ROIは以下に示すように、画像上のカラー矩形で表されます。



6. ROIコーナーとサイドハンドルをドラッグして、必要な位置に移動します。
7. **閉じる**ボタンをクリックして、**ROI管理**ウィンドウを閉じます。

次のステップは「ステップ4: ツールの構成」見開きページのです。

ROIの管理

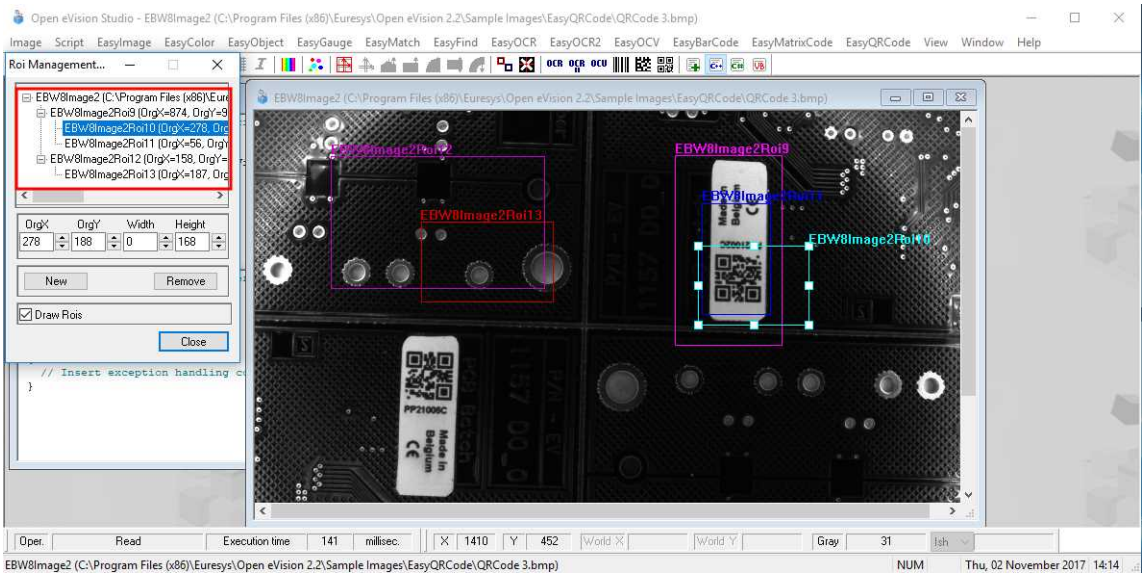
ROIを追加、変更、削除することができます。

画像には複数のROIがあります。各ROIは、画像に直接(その位置が画像に対して相対的であることを意味する)、または別のROI(その位置がこの親ROIに相対的であることを意味する)に付加することができます。

1. ROIを管理するには、メインメニューで**画像** > **ROI管理**...を選択します。

ROI管理ウィンドウには、以下のようなROI関係ツリーが表示されます。

Draw Roisボックスをチェックすると、すべてのROIが画像に異なる色で表示されます。



2. ROI関係ツリーでROIを選択します。
3. ROIコーナーとサイドハンドルをドラッグして、選択したROIの位置とサイズ(およびROIがあればそれに接続されているすべてのROIの位置) を変更します。
4. **新規**ボタンをクリックすると、選択したROIに関連付けられた新しいROIが追加されます。

ROIを画像に直接添付するには、ROI関係ツリーの上部にある画像を選択します。

5. **削除**ボタンをクリックすると、選択したROI(およびROIがあればそれに添付されているすべてのROI) が削除されます。
6. **閉じる**ボタンをクリックして、**ROI管理**ウィンドウを閉じます。

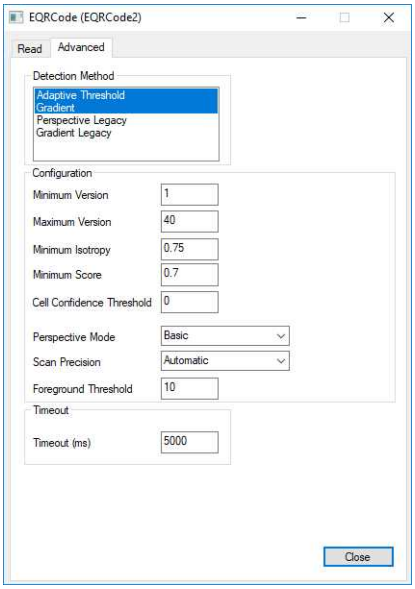
ステップ4: ツールの構成

作成したROIを含むイメージが準備できたら、ツールを設定する必要があります。

ツールウィンドウ:

1. さまざまなタブを開きます。

新しいツールを作成すると、すべてのパラメータがデフォルト値で設定されます。



EasyQRCodeツールのパラメータタブの例

2. 各タブで、必要に応じてパラメータの値を設定します。

パラメータ、機能、デフォルト値の詳細については、「機能ガイド」および「リファレンスマニュアル」を参照してください。

学習やゲージの使用などの特定のアクションについては、「機能ガイド」を参照してください。

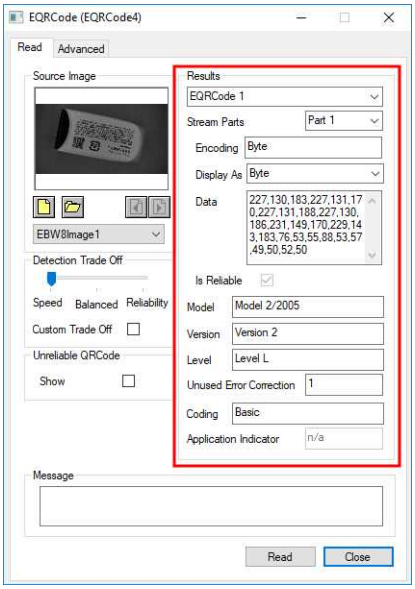
3. ツールを実行し、次のステップ「手順5: ツールの実行と実行時間の確認」下で説明されているように結果を分析します。

手順5: ツールの実行と実行時間の確認

ツールパラメータが設定されたら、ツールを実行し、必要に応じてコンピュータの実行時間を確認します。

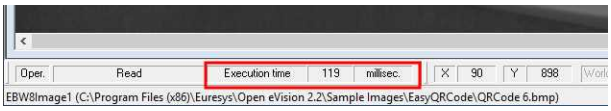
ツールウィンドウ:

1. 選択したイメージでツールを実行するには、**読み取り**、**検出**、**結果**または**実行**ボタン(ライブラリ関数に応じて) をクリックします。
2. 下図のように、画像上か結果フィールドまたはエリア内で確認します。



QRコードを読み取った後の結果例

3. 期待される結果が得られない場合：
   - パラメータを変更してみてください(デフォルト値から始め、一度に1つのパラメータを変更してください) 。
   - 画像が十分でない場合は、で説明されているように画像を強調してみてください。
4. メインOpen eVision Studioウィンドウの左下にある実行タイムバーで実行時間を確認します。



実行時間

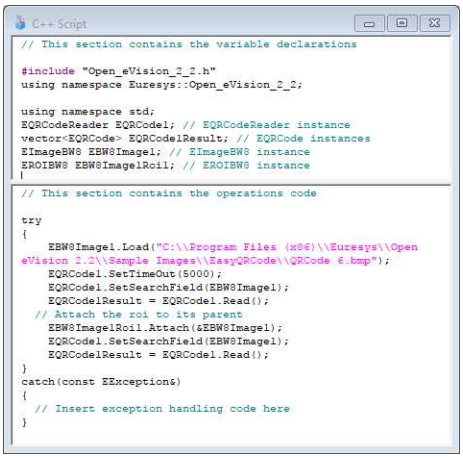
実行時間は、コンピュータで測定された実際の処理時間です。実行時間はお使いのコンピュータのプロセッサ、メモリ、オペレーティングシステム、および実行時のプロセッサの負荷に依存します。したがって、この実行時間はそれぞれわずかに異なります。

5. より代表的な実行時間を割り出すには、**読込**、**検出**、**結果**または**実行**ボタンを数回クリックし、平均実行時間を計算します。
6. アプリケーションで実行時間を短縮する必要がある場合は、以下を試してみてください。
   - ツールのパラメータを変更する
   - 画像に1つまたは複数のROIを追加する
   - 画像を強化する

次のステップは「手順6: 生成されたコードの使用」次ページのです。

手順6: 生成されたコードの使用

既定では、Open eVision Studioはインターフェイスで実行するすべての操作を、次のように選択した言語のコードに変換します。



ツールの結果が合う場合は、この生成されたコードを保存またはコピーして自分のアプリケーションで使用することができます。

アプリケーションにコードをコピー& ペースト

スクリプトウィンドウ:

1. コピーするコードセクションを選択します。
2. このコードを右クリックし、メニューの**コピー**をクリックします。
3. 開発環境ツールに移動し、コードを適切な場所に貼り付けます。

コードの保存

1. **スクリプト**メニューに移動します。
2. **名前を付けてスクリプトを保存**...をクリックします。
3. コードをテキストファイルとして保存するには、ファイル名とパスを入力します。

生成されたコードの管理

スクリプトメニューでは、次の操作を実行できます。

- プログラミング言語を選択します(言語を変更すると、スクリプトウィンドウのコンテンツは自動的に削除されます) 。
- **スクリプトコードの生成**を有効または無効にします。コードとして保存せずにいくつかの操作を実行する場合は、このオプションを無効にします。

4.4. 画像の前処理と保存

画像の前処理はいつするべきですか?

もちろん、最良のタイミングは、画像取得システムを設定して画像を処理しやすくし、Open eVisionがスムーズかつ効率的に実行されるときです。

これが不可能な場合や達成しにくい場合は、画像やROIを前処理を実行することで、Open eVisionツールを強化し準備することができます。

さまざまな機能を使用して、イメージのゲインとオフセットを調整し、畳み込み、閾値、スケール、イメージの回転とホワイトバランス、輪郭の強調など... EasyImageとEasyColor関数を使用して調整できます。

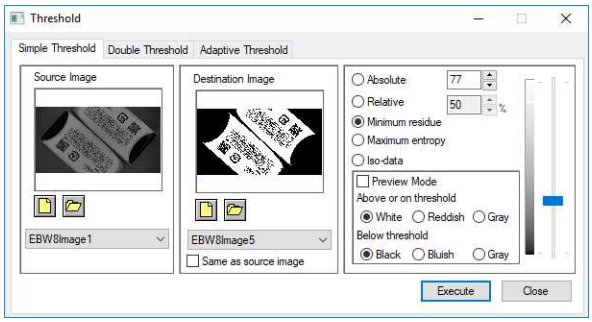
イメージの前処理

イメージの前処理と実行中のツールとの違いは、前処理により新しいイメージを生成され、ツールは主にイメージを変更せずにイメージから抽出して取り出します。

画像やROIを前処理するには:

1. メインメニューバーで、使用するライブラリ(EasyImageまたはEasyColor) をクリックします。
2. 使用する機能をクリックします。

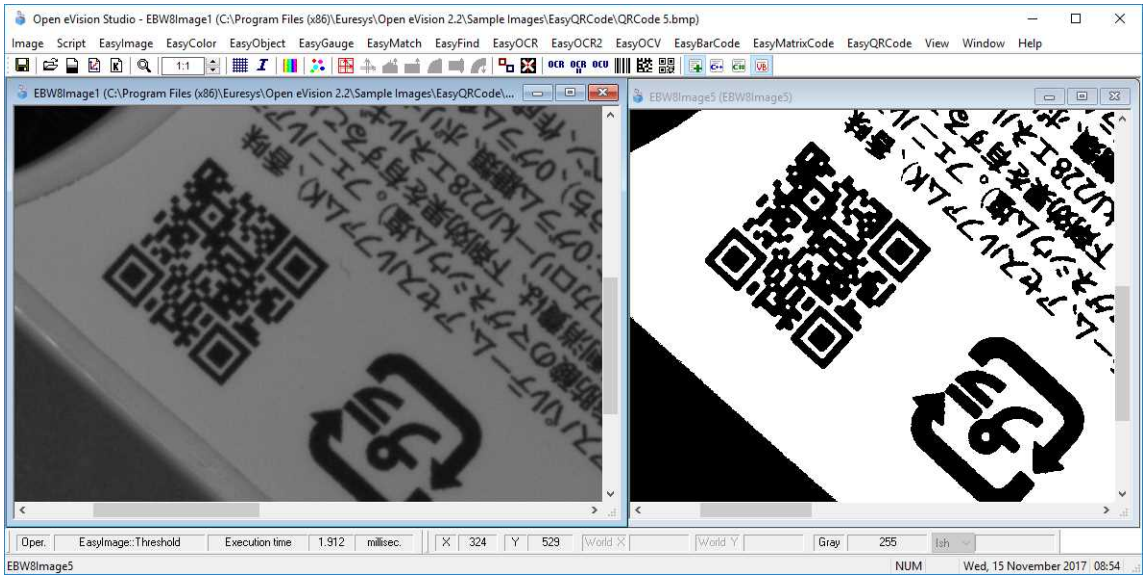
ほとんどの機能ダイアログボックスは、2つの画像選択領域とパラメータ設定領域で以下に示すようなものです。



前処理ダイアログボックスの例(EasyImageの閾値)

3. 選択した機能に複数のバージョンがある場合は、対応するタブを開きます。
4. **ソースイメージ**領域で、ソースイメージを開きます(「ステップ2: 画像を開く」ページの99で説明されているように) 。
5. **宛先イメージ**領域で、新しい宛先イメージを開いたり、作成したりします。
6. パラメータを設定します。
7. **実行**ボタンをクリックします。

前処理された画像は、以下に示すように宛先画像内で利用可能です。



ソースイメージおよび宛先イメージ(EasyImageの閾値)

8. Open eVision Studioの外にある宛先イメージを使用する場合は、以下のように保存します。

画像を保存します

1. 保存する画像をクリックして有効にします。
2. 保存メニューを開くには:
   - 画像を右クリックします
   - または、メインメニュー ＞ **画像**を開きます
3. **名前を付けて保存**をクリックします。
4. ファイル形式(JPEG、JPEG2000、PNG、TIFF、またはビットマップ) を選択します。
5. 名前を入力し、パスを選択します。
6. **保存**ボタンをクリックします。

5. Tutorials

5.1. EasyObject

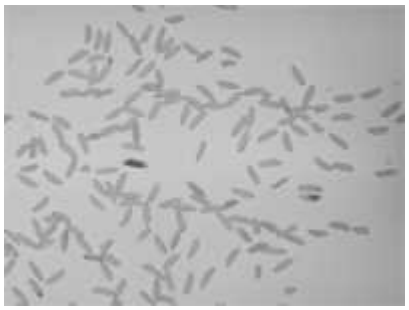
Removing Non-Significant Objects After Image Segmentation

「画像セグメンタ」ページの135

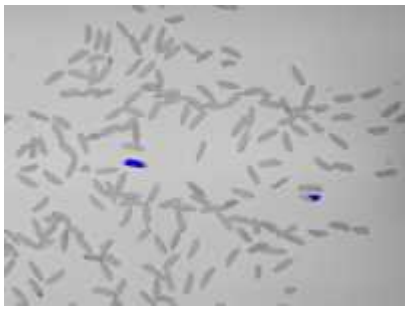
Objective

Following this tutorial, you will learn how to use EasyObject to detect bad rice grains (largely dark) among many normal rice grains (largely light).

You'll need first to load the source image (step 1). Then you'll perform the image segmentation, based on a threshold value (step 2). All the detected objects are dark, but some are too small to be significant. So, you'll set a minimum object area (number of pixels), and remove the smallest objects (step 3). Finally, you'll get only the objects that really represent bad rice grains.



ソース画像



Bad rice grains are detected

Step 1: Load the source image

1. From the main menu, click **EasyObject**, then **New EasyObject Tool**.
2. Keep the default variable name for the new object, and click ***OK***.
3. In the **Encoder** tab, click the ***Open*** icon of the Source Image area, and load the image file `EasyObject\Rice.jpg`.
4. Keep the default variable name for the new image, and click ***OK***.

Step 2: Perform image segmentation

1. In the **Encoder** tab, select the **Black Layer** check-box, and unselect the **White Layer** check-box.
2. Click the ... button around the **Threshold** field. In the Threshold dialog box, select ***Absolute***, enter '115', and click ***OK***.
3. Click **Encode** to detect the dark objects. In the image, each object is drawn using a different color.
4. Click **Results** to display the list of all the detected objects. Clicking an object in the image highlights it in the list, and vice versa.

Step 3: Remove the smallest objects

1. In the objects list, click ***Columns***.
2. Tick the ***Area*** check-box, and click ***OK***. In the list view, a new Area column appears, displaying each object area.
3. Click the Area column header to sort the objects. There are many small objects (area < 100) that may be considered as noise.
4. In the ***Selection*** tab, select ***Area*** from the Feature drop-down list. Select '**Less**' from the Mode drop-down list. In the Threshold field, enter '100'.
5. Click **Remove**. All the objects with an area smaller than 100 pixels have been removed from the list. The two remaining objects are the bad rice grains.

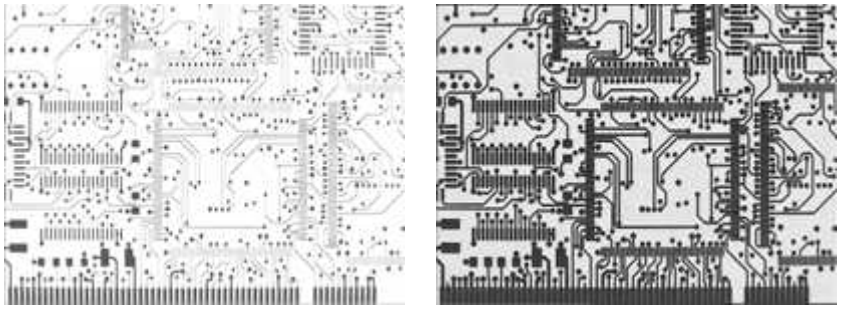
Detecting Differences Between Images Using Min-Max References

「ブロブを選択する / 並び替える」ページの137

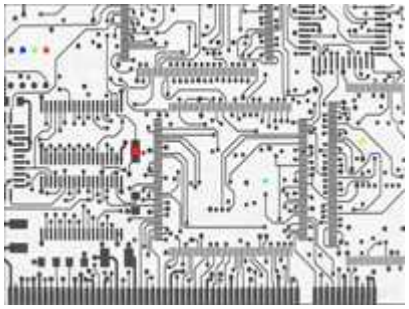
Objective

Following this tutorial, you will learn how to use EasyObject to compare images. In this example, we will check the quality of a PCB film.

You'll need first to load a reliable source image (step 1), from which two reference images (min and max) will be built (step 2). Then you'll load another image to be inspected (step 3), and perform the comparison with the min and max reference images (step 4). The differences will be detected.



High (left) and low (right) threshold reference images



In another image, differences are detected

Step 1: Load the source image

1. From the main menu, click **EasyObject**, then **Make Min Max**.
2. Click the **Open** icon of the Source Image area, and load the image file `EasyObject\FilmOk.png`.
3. Enter 'filmOk' for the name of the new image, and click **OK**.

Step 2: Build min and max reference images

1. Click **Execute**. Min and Max reference images are created, based on the source image.
   - *filmOkMax* is computed by dilating *filmOk* a given number of times ('geometric margin') and adding a constant ('gray level margin') to every pixel. *filmOkMin* is computed by eroding *filmOk* the same number of times and subtracting the same constant to every pixel.
   - The geometric margin can be seen as a position tolerance between the image to be inspected and the reference.
   - The gray level margin introduces a tolerance to lighting variations.

Step 3: Load an image to be inspected

1. From the main menu, click **EasyObject**, then **New EasyObject Tool**.
2. Keep the default variable name for the new object, and click **OK**.
3. In the **Encoder** tab, click the **Open** icon of the Source Image area, and load the image file `EasyObject\FilmBad.png`.
4. Enter '*filmBad*' for the name of the new image, and click **OK**

Step 4: Compare the image with the reference images

1. In the **Encoder** tab, select **ImageRange** in the segmentation method drop-down list.
2. Disable the **White Layer** check-box, and enable the **Black Layer** check-box.
3. Click the ... button around the High Image field. Select **filmOkMax** in the drop-down list, and click **OK**.
4. Click the ... button around the Low Image field. Select **filmOkMin** in the drop-down list, and click **OK**.
5. Click **Encode**. Eight differences are highlighted in the image.
6. Click **Results** to get further information about the detected objects. They may be further filtered and analyzed using object features selection and sorting capabilities of EasyObject.

Detecting Printing Errors Using a Flexible Mask

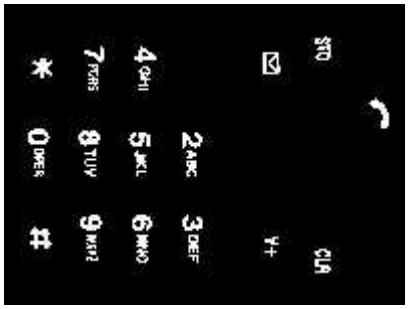
「エンコードされた画像からフレキシブルマスクを作成する」ページの139

Objective

Following this tutorial, you will learn how to use a flexible mask to target and search specific areas in the image.

You'll need first to load a source image (step 1), and a flexible mask image (step 2), that can be automatically applied on the source image to separate do-care areas (that must be considered) and don't-care areas (that should not be considered). Then, you'll perform the inspection only on do-care areas (step 3).

ソース画像



Mask image

Results

Step 1: Load the source image

1. From the main menu, click **EasyObject**, then **New EasyObject Tool**.
2. Keep the default variable name for the new object, and click ***OK***.
3. In the **Encoder** tab, click the ***Open*** icon, and load the image file `EasyObject\Mobile3.jpg`.
4. Keep the default variable name for the new image, and click ***OK***.

Step 2: Load the flexible mask image

1. In the **Encoder** tab, click the ***Open*** icon, and load the flexible mask image file `EasyObject\MobileMask.bmp`.
2. Enter 'mask' for the name of the new image, and click ***OK***. In the **Mask** area of the **Encoder** tab, notice that the mask image is selected from the drop-down list: the mask is automatically applied on the source image, because its name contains 'mask', and because it has been loaded from the coded image dialog box. The source image preview in the dialog box shows (in red diagonal lines) the don't-care area, that is the area that will be not be considered when encoding the source image.

Step 3: Inspect the image

1. In the ***Segmentation*** area of the **Encoder** tab, click the `...` button to display the threshold dialog box.

2. Select Absolute and enter 202 for the threshold. Click **OK** to close the dialog box.
3. Click **Encode** to locate the objects (in the do-care areas only). In the source image, each object is drawn using a different color. Three printing errors can be observed:
   - The digit '7' is partially printed.
   - The '+' sign is missing.
   - The handset is printed on a lighter tone.
4. Click Results to display the statistics on each object.
   - Selecting an object in the list highlights it in the image.
   - Selecting **Columns** and **Drawing** will display more options.

5.2. EasyMatch

Learning a Pattern and Creating an EasyMatch Model File

「検索パターン学習」ページの141

Objective

Following this tutorial, you will learn how to use EasyMatch to learn a model from a reference image, and save it as an EasyMatch model file.

You'll need first to load the reference image (step 1). Then, you'll learn it as the reference model (step 2). And you'll save the model as an EasyMatch model file (step 3).

Reference image

Step 1: Load the reference image

1. From the main menu, click **EasyMatch**, then **New Match Tool**.
2. Keep the default variable name for the new matcher object, and click **OK**.
3. In the **Learning** tab, click the **Open** icon, and load the image file `EasyMatch\FrameModel.tif`.
4. Keep the default variable name for the new image object, and click **OK**.

Step 2: Learn the reference image

- In the **Learning** tab, click **Learn** to acquire the model pattern.

Step 3: Save the model file

1. In the **Learning** tab, click the **Save As...** button.
2. Type a file name for the new EasyMatch model file. Its extension will be `.mch`.
3. Click **Save**.

Matching a Pattern According to a Model File

「パターンマッチングおよび検索結果」ページの142

Objective

Following this tutorial, you will learn how to use EasyMatch to load an EasyMatch model file, and search for occurrences of the pattern in an image.

You'll need first to load the model file, and a source image where the model will be searched for (steps 1-2). Then the pattern matching is fully automatic (step 3).

Occurrences of the model are automatically highlighted

Step 1: Load the model file

1. From the main menu, click **EasyMatch**, then **New Match Tool**.
2. Keep the default variable name for the new matcher object, and click **OK**.
3. In the **Learning** tab, click **Load...** to open the model file `EasyMatch\Switch.MCH`. The model contains all necessary information about the pattern we are searching for.

Step 2: Load a source image

1. In the **Matching** tab, click the **Open** icon, and load the image file `EasyMatch\Switch1.tif`.

2. Keep the default variable name for the new image object, and click **OK**.

Step 3: Perform the pattern matching

1. The pattern matching is automatically performed on the source image, and the matching occurrences are highlighted. Clicking **Execute** will insert the corresponding code into the script windows.
2. Further information about each occurrence can be found by clicking **Results**.
3. Click in a row to see the corresponding occurrence highlighted in the image.

Learning a Pattern According to an ROI

「検索パターン学習」ページの141

Objective

Following this tutorial, you will learn how to use EasyMatch to learn a model from an ROI in a source image, and to perform pattern matching on the same image.

You'll need first to load the source image, and define an ROI inside (steps 1-2). Then, you'll have to learn the model, using this ROI (step 3). Finally, you'll perform pattern matching in the source image (step 4), and will find additional occurrences of the model.

ROI that will be learned

Occurrences matching the model ROI

Step 1: Load the source image

1. From the main menu, click **Image**, then **Open**.
2. Load the image file `EasyMatch\BOARD.JPG`.
3. Keep the default variable name for the new image object, and click **OK**.

Step 2: Define an ROI

1. Right-click in the image, and select **New ROI...** from the contextual menu.
2. Keep the default variable name for the new ROI object, and click **OK**. A default ROI is placed over the image (blue rectangle with handles).

The ROI management dialog box is opened.

3. Resize the ROI and move it around one of the blue capacitors at the lower left part of the image.

Step 3: Learn a model from the ROI

1. From the main menu, click **EasyMatch**, then **New Match Tool**.
2. Keep the default variable name for the new matcher object, and click **OK**.
3. In the **Learning** tab of the matcher dialog box, select the ROI object from the Source Image drop-down list, and click **Learn** to acquire the model pattern.

Step 4: Match the pattern

1. In the **Matching** tab, increase the **Max Occurrences** field to 2.
2. Select the image object from the Source Image drop-down list.
3. Click **Execute**. The occurrences of the learned model are highlighted in the source image.
4. Further information about each occurrence can be found by clicking **Results**.
5. Click in a row to see the corresponding occurrence highlighted in the image.

Improving the Score of Matching Instances by Using "Don't Care Areas"

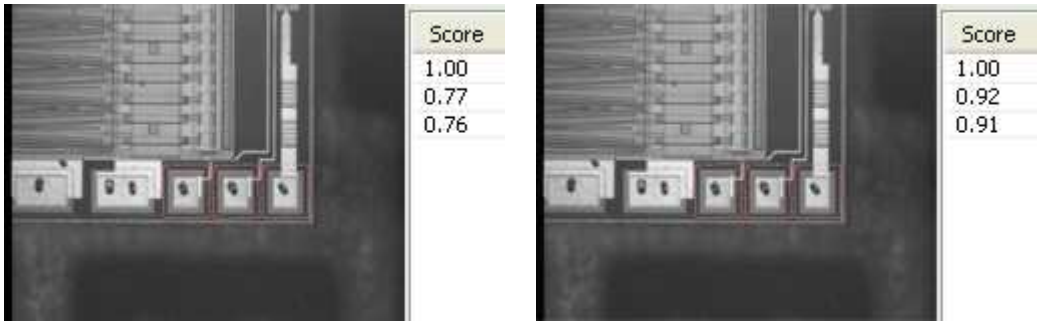
「検索パラメータの設定」ページの141

Objective

Following this tutorial, you will learn how to use EasyMatch to handle "don't care areas". "Don't care areas" help to define in the image the meaningful features only, by masking the areas that might change from image to image.

You'll need first to load a reference image and learn the reference model (steps 1-2). Then you'll perform automatic pattern matching of instances of the reference model, without using "don't care areas" (step 3). As the matching scores of the found instances are not high enough, you'll define a "don't care area" on the reference model, and restart the detection. The matching scores are much better (steps 4-5).

Reference model



Found instances and matching scores, without (left) and with (right) using "don't care areas"

Step 1: Load the reference image

1. From the main menu, click **EasyMatch**, then **New Match Tool**.
2. Keep the default variable name for the new Matcher object, and click ***OK***. The Matcher management dialog box is opened.
3. In the **Learning** tab, click the ***Open*** icon, and load the image file `EasyMatch\Die Pad Model 1.bmp`.
4. Keep the default variable name for the new Image object, and click ***OK***.

Step 2: Learn the reference model

- Click **Learn** to acquire the model pattern.

Step 3: Detect instances of the reference model without "don't care areas"

1. In the **Matching** tab, click the ***Open*** icon, and load the image file `EasyMatch\Die Pad 1.bmp`.
2. Keep the default variable name for the new Image object, and click ***OK***. An instance matching the reference model is highlighted.
3. Increase the ***Max Occurrences*** field to 3. Enable the **Sub-Pixel Interpolate** check-box to increase the matching precision.

4. Enter '-10.0' as the Minimum Angle (Deg). (Check that the angle is displayed in degrees. If not, select the angles unit from **View** > **Option** menu.)
5. Enter '10.0' as the Maximum Angle (Deg).
6. Click **Execute**. The pattern locations are highlighted in the source image.
7. Click **Results** to see the matching score of each found instance. The last two scores are rather bad. This is mainly due to the bright rectangle on the upper part of the reference image we have learned. We can improve the scores by using a "don't care area" to mask this bright area.

Step 4: Define the "don't care area"

1. In the **Don't Care Areas** tab, select the **Rectangle** radio button from the **Blacken Inside** group.
2. In the reference image, move your mouse pointer on the lower left corner of the bright rectangle, left-click and drag the don't care area (rectangle with red stripes) to the upper right corner of the bright rectangle to mask out this area.
3. In the **Don't Care Areas** tab, click **Learn**.

Step 5: Detect instances of the reference model with "don't care areas"

1. In the **Matching** tab, click **Execute**. The instances matching the reference model are still highlighted.
2. Click **Results** to compare the new matching scores. They are much better.

5.3. EasyGauge

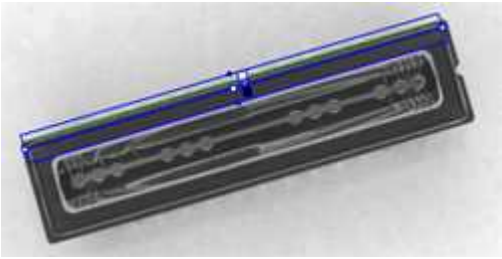
Measuring the Rotation Angle of an Object

「ラインフィッティング」ページの145

Objective

Following this tutorial, you will learn how to use EasyGauge to measure the rotation angle of a CCD sensor package. As we only need to retrieve an angle value, it's not required to work in a calibrated field of view. All geometrical parameters and results will be express as numbers of pixels.

You'll need to load the source image (step 1), and attach a line fitting gauge (step 2). The inspection is automatically performed (step 3).



Line fitting gauge

Step 1: Load the source image

1. From the main menu, click **EasyGauge**, then **New World Shape**.
2. Keep the default variable name, and click **OK**.
3. From the **Gauges** tab, click the **Open** icon, and load the image file `EasyImage\CCD.tif`.
4. Keep the default variable name for the new image object, and click **OK**.

Step 2: Attach a line gauge to the image

1. In the **Gauges** tab of the world shape dialog box, right-click the world shape icon, select **New** > **Line Gauge** from the contextual menu.
2. Keep the default variable name, and click **OK**. The line location gauge appears on the image. It consists of the following elements:
   - A blue line segment along which the transitions search is carried out.
   - Five white handles, allowing the user to move and rotate the segment.
   - A gray arrow, indicating in which direction the segment is traversed.
   - Black and white rectangles, indicating which kind of transition is searched for.
   - Green line, indicating the transition points found (if any).
3. Using the handles, move, rotate and extend the line gauge, so that it is positioned on the upper edge of the CCD package, with the gray arrow pointing downwards.
4. In the **Measurement** tab of the line gauge dialog box, choose 'White to Black' from the Type dropdown list and 'From Begin' from the Choice dropdown list.

Step 3: Perform the inspection

1. The image is automatically inspected. However, clicking **Process** in the world shape dialog box will insert the corresponding code into the script window.
2. Click the **Results** tab to retrieve the measured angle value.
3. To see the individual fitting points, select the **Points** checkbox under the Draw Samples area.

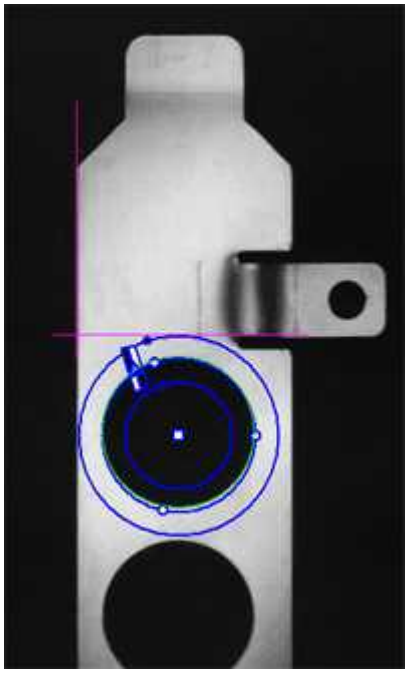
Measuring the Diameter of a Circle

「円形フィッティング」ページの146

Objective

Following this tutorial, you will learn how to use EasyGauge to measure the diameter of a circle in an image.

You'll first load an image for calibration —a dot grid— (step 1), and calibrate the field of view (step 2). Then you'll load the source image for inspection (step 3), and attach a circle fitting gauge (step 4). The inspection is automatically performed (step 5). All measurement results are expressed in physical units..



Measuring the diameter of a circle

Step 1: Load the calibration image

1. From the main menu, click **EasyGauge**, then **New World Shape**.
2. Keep the default variable name, and click **OK**.
3. In the **Dot Grid Calibration** tab, click the **Open** icon, and load the image file `EasyGauge\Dot Grid 1.tif`.
4. Keep the default variable name for the new image object, and click **OK**.

Step 2: Calibrate the field of view

- Click **Calibrate**. From now on, the field of view is calibrated, and all results will be expressed in physical units.

Step 3: Load the source image

1. From the **Gauges** tab, click the Open icon, and load the image file `EasyGauge\Bracket 6.tif`.
2. Keep the default variable name, and click **OK**.

Step 4: Attach a circle gauge to the image

1. In the **Gauges** tab of the world shape dialog box, right-click the frame shape icon, select **New** > **Circle Gauge** from the contextual menu.
2. Keep the default variable name, and click **OK**.
3. The circle location gauge appears on the image. It consists of the following elements:
   - A blue ring in which the circle is searched for.
   - A blue nominal circle.
   - Six white handles, allowing the user to move and rotate the segment.
   - A gray arrow, indicating in which direction the segment is traversed.
   - Black and white rectangles, indicating which kind of transition are searched for.
   - A green arc of circle, indicating the circle found (if any).
4. Using the handles, drag the circle fitting gauge around the upper bracket hole. Adjust the nominal circle on the hole edge and extend the searching area if necessary.
5. In the **Measurement** tab of the circle gauge dialog box, select 'From Begin' from the Choice dropdown list.

Step 5: Perform the inspection

1. The image is automatically inspected. However, clicking **Process** in the world shape dialog box will insert the corresponding code into the script window.
2. Click the **Results** tab to retrieve the measured diameter. All measurement results are expressed in physical units.

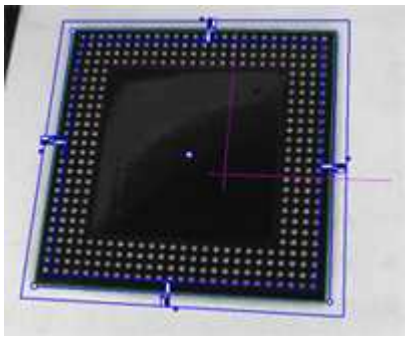
Measuring a Distorted Rectangle

「長方形のフィッティング」ページの146

Objective

Following this tutorial, you will learn how to use EasyGauge to perform measurements on a distorted rectangle component.

To obtain measurement results in physical units, we need to work in a calibrated field of view. You'll need first to load an image for calibration —a dot grid— (step 1), and calibrate the field of view (step 2). Then you'll load the distorted image (step 3), and attach a rectangle fitting gauge (step 4). The inspection is automatically performed (step 5). All measurement results are expressed in physical units.



Measuring a distorted rectangle

Step 1: Load the calibration image

1. From the main menu, click **EasyGauge**, then **New World Shape**.
2. Keep the default variable name, and click **OK**.
3. In the **Dot Grid Calibration** tab, click the **Open** icon, and load the image file `EasyGauge\Dot Grid 5.tif`. This dot grid has been acquired in the same conditions and has the same distortion as the image we want to inspect.
4. Keep the default variable name for the new image object, and click **OK**.

Step 2: Calibrate the field of view

- Click **Calibrate**. From now on, the field of view is calibrated, and all results will be expressed in physical units.

Step 3: Load the distorted image

1. From the **Gauges** tab, click the **Open** icon, and load the image file `EasyGauge\Distorted Component.tif`.
2. Keep the default variable name, and click **OK**.

Step 4: Attach a rectangle gauge to the image

1. In the **Gauges** tab, right-click the world shape icon, and select **New** > **Rectangle Gauge** from the contextual menu.
2. Keep the default variable name, and click **OK**. The rectangle gauge dialog box is opened, and the rectangle gauge is drawn on the image. It consists of the following elements:
   - A blue rectangular ring in which the rectangle is searched for.
   - A blue nominal rectangle.

- Eleven white handles, allowing the user to move and extend the search area.
- Gray arrows, indicating in which direction segments are examined.
- Black and white rectangles, indicating which kind of transition are searched for.
- A green rectangle, indicating the rectangle found (if any).

3. Due to the perspective effect, the rectangle gauge doesn't look like a rectangle. Using the central handle, move the rectangle gauge in the image and observe the rectangle deformation. Due to the calibration, the rectangle gauge shape adapts to the field of view deformation. Extend the searching area, and adjust the nominal rectangle on the component edges.
4. In the **Measurement** tab of the rectangle gauge dialog box, select 'White To Black' from the Type dropdown list and 'From Begin' from the Choice dropdown list.

Step 5: Perform the inspection

1. The image is automatically inspected. However, clicking **Process** in the world shape dialog box will insert the corresponding code into the script window.
2. Click the **Results** tab to retrieve the measured X and Y sizes. All measurement results are expressed in physical units.

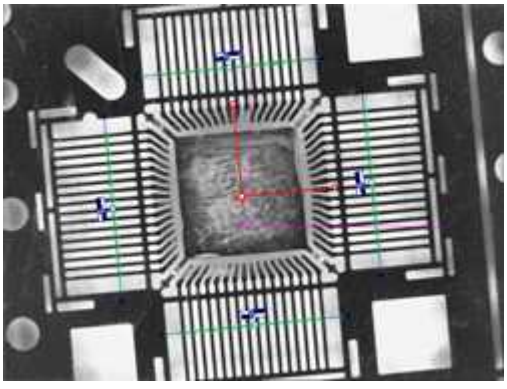
Locating Points Regarding to a Coordinate System

「点の位置」ページの145

Objective

Following this tutorial, you will learn how to use EasyGauge to perform lead frames inspection. This operation determines the dimension, position, curvature, size, angle or diameter of the lead frames with an excellent accuracy. Robustness is ensured by powerful edge-point selection mechanisms that are intuitive and easy to tune, allowing measurement in cluttered images.

You'll first load an image for calibration —a dot grid— (step 1), and calibrate the field of view (step 2). Then you'll load the lead frame image (step 3), and set a coordinate system (a frame shape). Regarding to this coordinate system, you can define point gauges (steps 5-6). Finally, you'll load another lead frame image, that has a slight angle deviation, so the coordinate system has to be rotated (steps 7-8). The inspection is then automatically performed (step 9). All measurement results are expressed in physical units.



Four point gauges over four sets of leads

Step 1: Load the calibration image

1. From the main menu, click **EasyGauge**, then **New World Shape**.
2. Keep the default variable name, and click **OK**.
3. In the **Dot Grid Calibration** tab, click the **Open** icon, and load the image file `EasyGauge\Dot Grid 2.tif`.
4. Keep the default variable name for the new image object, and click **OK**.

Step 2: Calibrating the field of view

1. Click **Calibrate**. From now on, the field of view is calibrated, and all results will be expressed in physical units.

Step 3: Loading a lead frame image

1. From the **Gauges** tab, click the **Open** icon, and load the image file `EasyGauge\Lead Frame 1.tif`.
2. Keep the default variable name, and click **OK**.

Step 4: Setting a coordinate system

1. In the **Gauges** tab of the world shape dialog box, right-click the world shape icon, select **New** > **Frame Shape** from the contextual menu.
2. Keep the default variable name, and click **OK**. The frame shape icon appears in the world shape dialog box.
3. Drag the frame shape center approximately to the square center of the image.

Step 5: Attaching a point gauge to the frame shape

1. In the **Gauges** tab of the world shape dialog box, right-click the frame shape icon, select **New** > **Point Gauge** from the contextual menu.
2. Keep the default variable name, and click **OK**. The point location gauge appears on the image. It consists of the following elements:
   - A blue line segment along which the transitions search is carried out.
   - Three white handles, allowing the user to move and rotate the segment.
   - A gray arrow, indicating in which direction the segment is traversed.
   - Black and white rectangles, indicating which kind of transition is searched for.
   - Green crosses, indicating the transition points found (if any).
3. Place the point location gauge over a set of leads: in the **Position** tab of the point gauge dialog box, set the Center Y property to 7, and the Tolerance property to 5.

Step 6: Attaching other point gauges to the frame shape

1. From the **Gauges** tab of the world shape dialog box, create three other point gauges (refer to step 5).
2. Place these point location gauges over the remaining sets of leads :
   - Center Y = -7, Tolerance = 5
   - Center X = 7, Tolerance = 5, Angle = 90
   - Center X = -7, Tolerance = 5, Angle = 90

Step 7: Loading another lead frame image

1. From the **Gauges** tab, click the Open icon, and load the image file `EasyGauge\Lead Frame 2.tif`.
2. Keep the default variable name for the new image, and click **OK**.

Step 8: Tuning the coordinate system

1. In the frame shape dialog box, set the Angle property to 5.8.
2. Drag the frame shape center approximately to the square center of the image. All point location gauges automatically follow.

Step 9: Performing the inspection

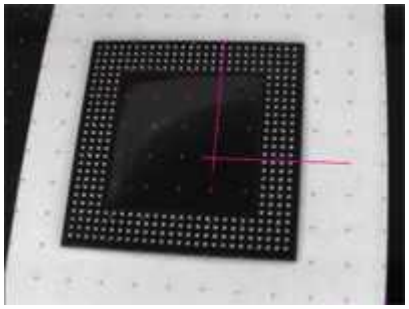
1. The image is automatically inspected. However, clicking Process in the world shape dialog box will insert the corresponding code into the script window.
2. From the point gauge dialog boxes, click the Results tab to retrieve the located points coordinates. All measurement results are expressed in physical units. The values refer to the frame shape system.

Unwarping a Distorted Image

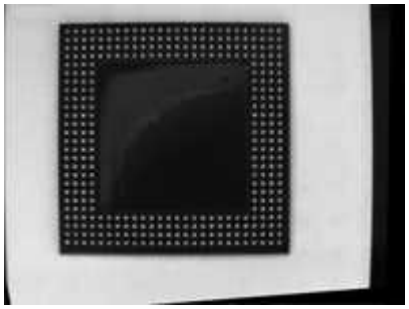
Objective

Following this tutorial, you will learn how to use EasyGauge to perform grid calibration, and unwarp a distorted image.

You'll first load an image for calibration —a dot grid— (step 1), and calibrate the field of view (step 2). Then you'll load the distorted image (step 3), and perform the unwarping operation (step 4).



Distorted image



Unwarped image

Step 1: Load the calibration image

1. From the main menu, click **EasyGauge**, then **New World Shape**.
2. Keep the default variable name, and click **OK**.
3. In the **Dot Grid Calibration** tab, click the **Open** icon, and load the image file `EasyGauge\Dot Grid 5.tif`. This dot grid has been acquired in the same conditions and has the same distortion as the image we want to unwarp.
4. Keep the default variable name for the new image object, and click **OK**.

Step 2: Calibrate the field of view

- Click **Calibrate**.

Step 3: Load the distorted image

1. From the **Unwarping** tab, click the **Open** icon, and load the image file `EasyGauge\Distorted component.tif`.
2. Keep the default variable name, and click **OK**.

Step 4: Unwarp the distorted image

1. In the Destination Image area, click **New** icon to create a new image.
2. Keep the default image settings, and click **OK**.

3. Select **Interpolate** checkbox to improve resulting image quality.
4. Click **Unwarp**. In the destination image, all distortions are corrected.

5.4. EasyFind

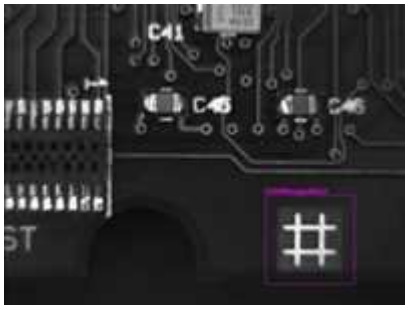
Detecting Highly-Degraded Occurrences of a Reference Model in Multiple Files

「パターンの検索および検索結果」ページの144

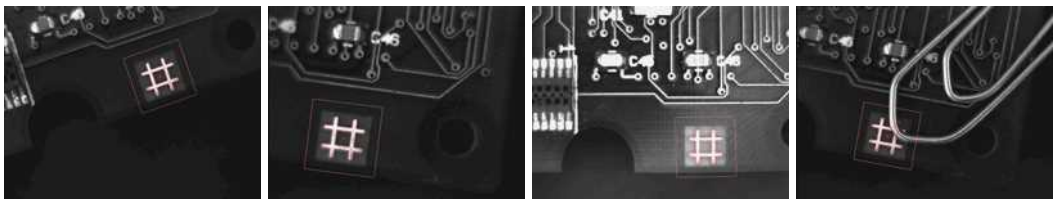
Objective

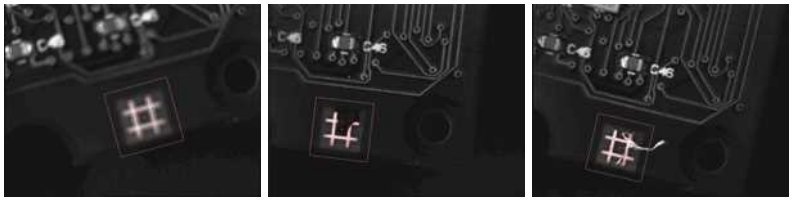
Following this tutorial, you will learn how to use EasyFind to detect in multiple images highly-degraded occurrences of a reference model. The degradation can be due to noise, blur, occlusion, missing parts or unstable illumination conditions.

You'll need first to load a reference image, define an ROI where EasyFind will learn the reference model, and set rotation and scaling tolerances for the expected occurrences to search for (steps 1-4). Then you're ready to open multiple files, and perform automatic detection of occurrences (even highly-degraded) of the reference model (steps 5-6).



Reference model





Occurrences of the reference model are found, even if highly-degraded

Step 1: Load the reference image

1. From the main menu, click **EasyFind**, then **New EasyFind Tool**.
2. Keep the default variable name for the new PatternFinder object, and click ***OK***. The PatternFinder management dialog box is opened.
3. In the **Model** tab, click the ***Open*** icon, and load the image file `EasyFind\Fiducial 1.tif`.
4. Keep the default variable name for the new image object, and click OK.

Step 2: Create an ROI to define the reference model on the reference image

1. In the image, right-click and select **New ROI**... item from the menu.
2. Keep the default variable name for the new ROI object, and click ***OK***. A default ROI is placed over the image (blue rectangle with handles). The ROI management dialog box is opened.
3. Drag the ROI over the reference model and resize it using its handles. Alternatively, enter the following coordinates in the ROI dialog box: 500, 365, 170, 170 for OrgX, OrgY, Width, and Height respectively, and click ***Close***.

Step 3: Learn the reference model

1. In the PatternFinder **Model** tab, select the ROI object from the source image drop-down list, and click **Learn**. The reference model is perfectly detected (green edges).
2. In the PatternFinder ***Search Field*** tab, select the Image object from the source image drop-down list. Tick the Draw Features check-box.

The model location and feature points are highlighted in the source image.

Step 4: Set rotation and scaling tolerances

- In the PatternFinder ***Allowances*** tab, set both angle tolerance and scale tolerance to 25.0.

Step 5: Select multiple images

1. In the PatternFinder ***Search Field*** tab, click the ***Open*** icon. Select the images files `EasyFind\Fiducial 2.tif` to `Fiducial 8.tif` (use the shift key to select multiple files), and click ***Open***.

2. Keep the default variable name for the new Image object, and click **OK**. The last image appears. The reference model is found, even if highly-degraded.
3. Detection of the reference model is automatically performed. It is not necessary to click Find once a new image appears, as inspection is automatically performed. However, clicking **Find** will insert the corresponding code into the script windows.
4. Click **Results** to find more information about the found instance.

Step 6: Browse multiple images

- In the PatternFinder **Search Field** tab, click the **Load Next** or **Load Previous** icons.

The image files appear, and the reference model is automatically detected.

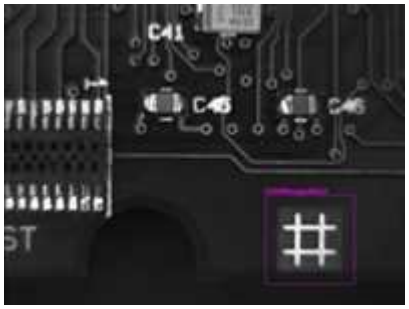
Improving the Score of Found Instances by Using "Don't Care Areas"

「検索パラメータの設定」ページの143

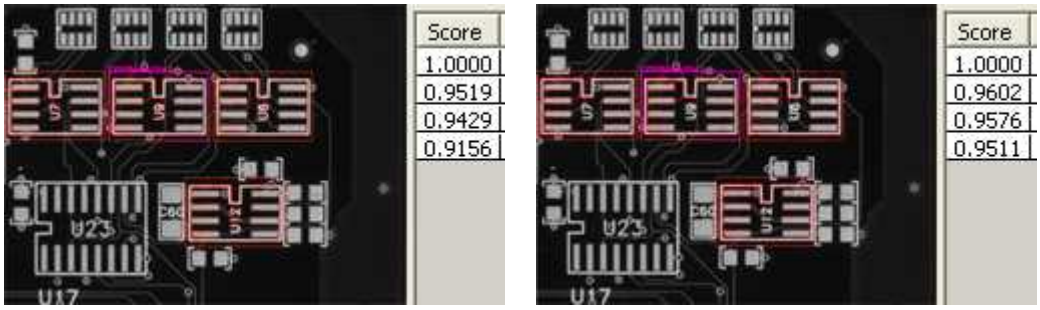
Objective

Following this tutorial, you will learn how to use EasyFind to handle "don't care areas" in geometric pattern matching. "Don't care areas" help to define in the image the meaningful features only, by masking the areas that might change from image to image, such as text and numbers.

You'll need first to load a reference image, define an ROI where EasyFind will learn the reference model, and set a rotation tolerance for the expected instances to search for (steps 1-4). Then you're ready to perform automatic detection of instances of the reference model, without using "don't care areas" (step 5). As the matching scores of the found instances are not high enough, you'll define a "don't care area" on the reference model, and restart the detection. The matching scores are slightly better (steps 6-7).



Reference model



Found instances and matching scores, without (left) and with (right) using "don't care areas"

Step 1: Loading the reference image

1. From the main menu, click **EasyFind**, then **New EasyFind Tool**.
2. Keep the default variable name for the new PatternFinder object, and click ***OK***. The PatternFinder management dialog box is opened.
3. In the **Model** tab, click the ***Open*** icon, and load the image file `EasyFind\Solder Pad 1.tif`.
4. Keep the default variable name for the new Image object, and click ***OK***.

Step 2: Creating an ROI to define the reference model on the reference image

1. In the image, right-click and select **New ROI...** item from the menu.
2. Keep the default variable name for the new ROI object, and click ***OK***. A default ROI is placed over the image (blue rectangle with handles). The ROI management dialog box is opened.
3. Drag the ROI over the reference model and resize it using its handles. Alternatively, enter the following coordinates in the ROI dialog box: 200, 130, 190, 130 for OrgX, OrgY, Width, and Height respectively, and click ***Close***.

Step 3: Learning the reference model

1. In the PatternFinder **Model** tab, select the ROI object from the source image drop-down list, and click **Learn**. The reference model is detected.
2. In the PatternFinder **Search Field** tab, select the Image object from the source image drop-down list. Tick the **Draw Features** check-box. The model location and feature points are highlighted in the source image.

Step 4: Setting a rotation tolerance

1. In the PatternFinder ***Allowances*** tab, set the angle tolerance to 5.0.

Step 5: Detecting instances of the reference model without "don't care areas"

1. In the PatternFinder **Search Field** tab, set **Max Instances** to 4, and click **Find**. The instances matching the reference model are highlighted.
2. Click **Results** to see the matching score of each found instance. Even though the scores are good, we can still improve them slightly by using a "don't care area" to mask the text appearing in the learned pattern.

Step 6: Defining the "don't care area"

1. In the PatternFinder **Don't Care Areas** tab, select the **Rectangle** radio button from the **Blacken Inside** group.
2. In the ROI defining the reference model, move your mouse pointer on the top-left corner of the text "U9", left-click and drag the don't care area (rectangle with red stripes) to mask out the text.

Step 7: Detecting instances of the reference model with "don't care areas"

1. In the PatternFinder **Don't Care Areas** tab, click **Learn**, and then click **Find**.

The instances matching the reference model are still highlighted, but the text is not found anymore.

2. Click **Results** to compare the new matching scores.

They are slightly better.

6. Code Snippets

6.1. 基本的な型

画像の読み込みと保存

```
////////////////////////////////////////////////////////////
// This code snippet shows how to load and save an image. //
////////////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// Load an image file
srcImage.Load("mySourceImage.bmp");
```

```
// ...
```

```
// Save the destination image into a file
dstImage.Save("myDestImage.bmp");
```

```
// Save the destination image into a jpeg file
// The default compression quality is 75
dstImage.Save("myDestImage.jpg");
```

```
// Save the destination image into a jpeg file
// set the compression quality to 50
dstImage.SaveJpeg("myDestImage50.jpg", 50);
```

第三者の画像のインターフェイス

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to link an Open eVision image //
// to an externally allocated buffer. //
////////////////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
```

```
// Size of the third-party image
int sizeX;
int sizeY;
```

```
//Pointer to the third-party image buffer
EBW8* imgPtr;
```

```
// ...
```

```
// Link the Open eVision image to the third-party image
// Assuming the corresponding buffer is aligned on 4 bytes
srcImage.SetImagePtr(sizeX, sizeY, imgPtr);
```

ピクセル値の取得

```
////////////////////////////////////////////////////////////
```

```
// This code snippet shows the recommended method (fastest)  //
// to access the pixel values in a BW8 image                 //
///////////////////////////////////////////////////////////////
```

```
EImageBW8 img;
```

```
OEV_UINT8* pixelPtr;
OEV_UINT8* rowPtr;
OEV_UINT8  pixelValue;
OEV_UINT32 rowPitch;
OEV_UINT32 x, y;
```

```
rowPtr = reinterpret_cast <OEV_UINT8*>(img.GetImagePtr());
rowPitch = img.GetRowPitch();
```

```
for (y = 0; y < height; y++)
{
    pixelPtr = rowPtr;

    for (x = 0; x < width; x++)
    {
        pixelValue = *pixelPtr;

        // Add your pixel computation code here

        *pixelPtr = pixelValue;
        pixelPtr++;
    }

    rowPtr += rowPitch;
}
```

ROIの設置

```
///////////////////////////////////////////////////////////////
// This code snippet shows how to attach an ROI to an image //
// and set its placement. //
///////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 parentImage;
```

```
// ROI constructor
EROIBW8 myROI;
```

```
// ...
```

```
// Attach the ROI to the image
myROI.Attach(&parentImage);
```

```
//Set the ROI position
myROI.SetPlacement(50, 50, 200, 100);
```

ベクトル管理

```
///////////////////////////////////////////////////////////////
// This code snippet shows how to create a vector, fill it //
// and retrieve the value of a given element. //
///////////////////////////////////////////////////////////////
```

```
// EBW8Vector constructor
EBW8Vector ramp;
```

```
// Clear the vector
ramp.Empty();
```

```
// Fill the vector with increasing values
for(int i= 0; i < 128; i++)
{
        ramp.AddElement((EBW8)i);
}
```

```
// Retrieve the 10th element value
EBW8 value= ramp[9];
```

例外管理

```
//////////////////////////////////////////
// This code snippet shows how to manage //
// Open eVision exceptions. //
//////////////////////////////////////////
```

```
try
{
// Image constructor
        EImageC24 srcImage;
```

```
// ...
```

```
// Retrieve the pixel value at coordinates (56, 73)
        EC24 value= srcImage.GetPixel(56, 730);
}
```

```
catch(Euresys::Open_eVision_1_1::EException exc)
{
// Retrieve the exception description
        std::string error = exc.What();
}
```

6.2. EasyObject

ブロブの構 築

画 像 エンコーダ

```
////////////////////////////////////////////////////////////
// This code snippet shows how to build blobs belonging to //
// the white layer according to the minimum residue method //
// and how to build blobs belonging to the black layer //
// according to an absolute threshold. //
////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Build the blobs belonging to the white layer,
// the segmentation is based on the Minimum Residue method
encoder.Encode(srcImage, codedImage);
```

```
// Build the blobs belonging to the black layer,
// the segmentation is based on an absolute threshold (110)
Segmenters::EGrayscaleSingleThresholdSegmenter& segmenter=
encoder.GetGrayscaleSingleThresholdSegmenter();
segmenter.SetBlackLayerEncoded(true);
segmenter.SetWhiteLayerEncoded(false);
```

```
segmenter.SetMode(EGrayscaleSingleThreshold_Absolute);
segmenter.SetAbsoluteThreshold(110);
```

```
encoder.Encode(srcImage, codedImage);
```

画 像 セグメンタ

```
////////////////////////////////////////////////////////////
// This code snippet shows how to build blobs according to //
// a user-defined image segmenter. //
////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...

// Set the segmentation method to GrayscaleDoubleThreshold
encoder.SetSegmentationMethod(ESegmentationMethod_GrayscaleDoubleThreshold);

// Retrieve the segmenter object
Segmenters::EGrayscaleDoubleThresholdSegmenter& segmenter=
encoder.GetGrayscaleDoubleThresholdSegmenter();

// Set the high and low threshold values
segmenter.SetHighThreshold(150);
segmenter.SetLowThreshold(50);

// Specify the layers to be encoded (neutral layer only)
segmenter.SetBlackLayerEncoded(false);
segmenter.SetNeutralLayerEncoded(true);
segmenter.SetWhiteLayerEncoded(false);

// Encode the image
encoder.Encode(srcImage, codedImage);
```

ホール抽出

```
////////////////////////////////////////////////////////
// This code snippet shows how to retrieve blobs' holes. //
////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage;

// Image encoder
EImageEncoder encoder;

// Coded image
ECodedImage2 codedImage;

// ...

// Encode the image
encoder.Encode(srcImage, codedImage);

// Retrieve holes for all the blobs
for (unsigned int blobIndex = 0; blobIndex < codedImage.GetObjCount(); blobIndex++)
{
        EObject& blob = codedImage.GetObj(blobIndex);

// Browse the holes of the current object
for (unsigned int holeIndex = 0; holeIndex < blob.GetHoleCount(); holeIndex++)
        {
 // Retrieve a given hole
 EHole& hole = blob.GetHole(holeIndex);
        }
}
```

連続モード

```
////////////////////////////////////////////////
// This code snippet shows how to build blobs //
// in the continuous mode context. //
```

```
////////////////////////////////////////////////
// Image constructor
EImageBW8 srcImage;

// Image encoder
EImageEncoder encoder;

// Coded image
ECodedImage2 codedImage;

// ...

// Enable the continuous mode
encoder.SetContinuousModeEnabled(true);

// Loop to acquire the different chunks
for (int count = 0; count < MAX_COUNT ; count++)
{
// Store the new chunk into srcImage
// ...

// Encode the current chunk
        encoder.Encode(srcImage, codedImage);
}

// Flush the continuous mode
encoder.FlushContinuousMode(codedImage);
```

ブロブの特徴の計算

```
////////////////////////////////////////////////////////////
// This code snippet shows how to retrieve blobs' features. //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage;

// Image encoder
EImageEncoder encoder;

// Coded image
ECodedImage2 codedImage;

// ...

// Encode the source image
encoder.Encode(srcImage, codedImage);

for (unsigned int index = 0; index < codedImage.GetObjCount(); index++)
{
// Retrieve the selected blob gravity center
        EObject& blob = codedImage.GetObj(index);
float centerX = blob.GetGravityCenter().GetX();
float centerY = blob.GetGravityCenter().GetY();
}
```

ブロブを選択する / 並び替える

```
////////////////////////////////////////////////////////
```

```
// This code snippet shows how to build blobs, select //
// some of them and sort the selected ones. //
//////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Encode the source image
encoder.Encode(srcImage, codedImage);
```

```
// Create a blob selection
EObjectSelection selection;
selection.AddObjects(codedImage);
```

```
// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature_Area, 100, ESingleThresholdMode_
Less);
```

```
// Retrieve the number of remaining blobs
unsigned int numBlobs= selection.GetElementCount();
```

```
// Sort the remaining blobs based on their area
selection.Sort(EFeature_Area, ESortDirection_Ascending);
```

```
// Retrieve the selected blobs
for (unsigned int index = 0; index < numBlobs; index++)
{
 float centerX= selection.GetElement(index).GetGravityCenterX();
 float centerY= selection.GetElement(index).GetGravityCenterY();
}
```

フレキシブルマスクを使 用 する

ブロブの構 築

```
//////////////////////////////////////////////////////
// This code snippet shows how to build blobs inside //
// a region defined by a flexible mask. //
//////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Encode the source image regions
// corresponding to the mask do care areas
encoder.Encode(srcImage, mask, codedImage);
```

エンコードされた画像からフレキシブルマスクを作成する

```
////////////////////////////////////////////////////////
// This code snippet shows how to generate a flexible //
// mask from an encoded image. //
////////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Encode the source image
encoder.Encode(srcImage, codedImage);
```

```
// The source image and the mask must have the same size
mask.SetSize(&srcImage);
```

```
// Create the mask based on the white layer
// of the coded image
codedImage.RenderMask(mask, 1);
```

ブロブ選択からフレキシブルマスクを作成する

```
////////////////////////////////////////////////////////
// This code snippet shows how to generate a flexible //
// mask from a selection of blobs. //
////////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Encode the source image
encoder.Encode(srcImage, codedImage);
```

```
// The source image and the mask must have the same size
mask.SetSize(&srcImage);
```

```
// Create a blob selection
EObjectSelection selection;
selection.AddObjects(codedImage);
```

```
// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature_Area, 100, ESingleThresholdMode_
Less);
```

```
// Create the mask based on the blob selection
selection.RenderMask(mask);
```

```
// Sort the remaining blobs based on their area
selection.Sort(EFeature_Area, ESortDirection_Descending);
```

```
// Create the mask corresponding to the largest blob
selection.GetElement(0).RenderMask(mask);
```

6.3. EasyMatch

検索パターン学習

```
////////////////////////////////////////////////////
// This code snippet shows how to learn a pattern //
// defined by a region of interest (ROI). //
////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// ROI constructor
EROIBW8 pattern;
```

```
// EMatcher constructor
EMatcher matcher;
```

```
// ...
```

```
// Attach the ROI to the source image
// and set its position
pattern.Attach(&srcImage);
pattern.SetPlacement(214, 52, 200, 200);
```

```
// Learn the pattern
matcher.LearnPattern(&pattern);
```

検索パラメータの設定

```
//////////////////////////////////////////////////////////
// This code snippet shows how to tune pattern matching //
// search parameters and save them into a file. //
//////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 pattern;
```

```
// EMatcher constructor
EMatcher matcher;
```

```
// ...
```

```
// Learn the pattern
matcher.LearnPattern(&pattern);
```

```
// Set the maximum number of occurrences
matcher.SetMaxPositions(5);
```

```
// Set the rotation tolerances
matcher.SetMinAngle(-20.f);
matcher.SetMaxScale(20.f);
```

```
// Enable sub-pixel accuracy
matcher.SetInterpolate(true);
```

```
// Set the minimum score
matcher.SetMinScore(0.70f);
```

```
// Save the matching context into a model file
matcher.Save("myModel.mch");
```

パターンマッチングおよび検索結果

```
//////////////////////////////////////////////////////
// This code snippet shows how to perform pattern //
// matching operations and retrieve the results. //
//////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EMatcher constructor
EMatcher matcher;
```

```
// ...
```

```
// Load a model file
matcher.Load("myModel.mch");
```

```
// Perform the matching
matcher.Match(&srcImage);
```

```
// Retrieve the number of occurrences
int numOccurrences= matcher.GetNumPositions();
```

```
// Retrieve the first occurrence
EMatchPosition myOccurrence= matcher.GetPosition(0);
```

```
// Retrieve its score and position
float score= myOccurrence.Score;
float centerX= myOccurrence.CenterX;
float centerY= myOccurrence.CenterY;
```

6.4. EasyFind

検索パターン学習

```
////////////////////////////////////////////////////
// This code snippet shows how to learn a pattern //
// defined by a region of interest (ROI). //
////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// ROI constructor
EROIBW8 pattern;
```

```
// EPatternFinder constructor
EPatternFinder finder;
```

```
// ...
```

```
// Attach the ROI to the source image
// and set its position
pattern.Attach(&srcImage);
pattern.SetPlacement(214, 52, 200, 200);
```

```
// Learn the pattern
finder.Learn(&pattern);
```

検索パラメータの設定

```
//////////////////////////////////////////////////////////
// This code snippet shows how to tune pattern finding //
// search parameters and save them into a file. //
//////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 pattern;
```

```
// EPatternFinder constructor
EPatternFinder finder;
```

```
// ...
```

```
// Learn the pattern
finder.Learn(&pattern);
```

```
// Set the maximum number of occurrences
finder.SetMaxInstances(5);
```

```
// Set the rotation tolerances
finder.SetAngleTolerance(20.f);
```

```
// Set the minimum score
finder.SetMinScore(0.70f);
```

```
// Save the finding context into a model file
finder.Save("myModel.fnd");
```

パターンの検索および検索結果

```
////////////////////////////////////////////////////
// This code snippet shows how to perform pattern //
// finding operations and retrieve the results. //
////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EPatternFinder constructor
EPatternFinder finder;
```

```
// EFoundPattern constructor
std::vector<EFoundPattern> foundPattern;
```

```
// ...
```

```
// Load a model file
finder.Load("myModel.fnd");
```

```
// Perform the pattern finding
foundPattern= finder.Find(&srcImage);
```

```
// Retrieve the number of instances
int numInstances= foundPattern.size();
```

```
// Retrieve the score and the
// position of the first instance
float score= foundPattern[0].GetScore();
float centerX= foundPattern[0].GetCenter().GetX();
float centerY= foundPattern[0].GetCenter().GetY();
```

6.5. EasyGauge

点の位置

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to create a point location tool, //
// adjust the transition parameters, set the nominal gauge //
// position, perform the measurement and retrieve the result. //
////////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EPointGauge constructor
EPointGauge pointGauge;
```

```
// Adjust the transition parameters
pointGauge.SetTransitionType(ETransitionType_Wb);
pointGauge.SetTransitionChoice(ETransitionChoice_Closest);
```

```
// Set the gauge nominal position
pointGauge.SetCenterXY(256.f, 256.f);
```

```
// Set the gauge length to 100 units and the angle to 45°
pointGauge.SetTolerance(100.f, 45.f);
```

```
// Measure
pointGauge.Measure(&srcImage);
```

```
// Get the measured point coordinates
float measuredX = pointGauge.GetMeasuredPoint().GetX();
float measuredY = pointGauge.GetMeasuredPoint().GetY();
```

```
// Save the point gauge measurement context
pointGauge.Save("myPointGauge.gge");
```

ラインフィッティング

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to create a line measurement tool, //
// adjust the transition parameters, set the nominal gauge //
// position, perform the measurement and retrieve the result. //
////////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// ELineGauge constructor
ELineGauge lineGauge;
```

```
// Adjust the transition parameters
lineGauge.SetTransitionType(ETransitionType_Bw);
lineGauge.SetTransitionChoice(ETransitionChoice_NthFromEnd);
lineGauge.SetTransitionIndex(2);
```

```
// Set the line fitting gauge position,
// length (50 units) and orientation (20°)
```

```
EPoint center(256.f, 256.f);
ELine line(center, 50.f, 20.f);
lineGauge.SetLine(line);
```

```
// Measure
lineGauge.Measure(&srcImage);
```

```
// Get the origin and end point coordinates of the fitted line
EPoint originPoint = lineGauge.GetMeasuredLine().GetOrg();
EPoint endPoint = lineGauge.GetMeasuredLine().GetEnd();
```

```
// Save the point gauge measurement context
lineGauge.Save("myLineGauge.gge");
```

円形フィッティング

```
//////////////////////////////////////////////////////////////////////
// This code snippet shows how to create a circle measurement tool, //
// adjust the transition parameters, set the nominal gauge //
// position, perform the measurement and retrieve the result. //
//////////////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// ECircleGauge constructor
ECircleGauge circleGauge;
```

```
// Adjust the transition parameters
circleGauge.SetTransitionType(ETransitionType_Bw);
circleGauge.SetTransitionChoice(ETransitionChoice_LargestAmplitude);
```

```
// Set the Circle fitting gauge position, diameter (50 units),
// starting angle (10°), and amplitude (270°)
EPoint center(256.f, 256.f);
ECircle circle(center, 50.f, 10.f, 270.f);
circleGauge.SetCircle(circle);
```

```
// Measure
circleGauge.Measure(&srcImage);
```

```
// Get the center point coordinates and the radius of the fitted circle
float centerX = circleGauge.GetMeasuredCircle().GetCenter().GetX();
float centerY = circleGauge.GetMeasuredCircle().GetCenter().GetY();
float radius = circleGauge.GetMeasuredCircle().GetRadius();
```

```
// Save the point gauge measurement context
circleGauge.Save("myCircleGauge.gge");
```

長方形のフィッティング

```
//////////////////////////////////////////////////////////////////////
// This code snippet shows how to create a rectangle measurement tool, //
// adjust the transition parameters, set the nominal gauge position, //
// perform the measurement and retrieve the result. //
//////////////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// ERectangleGauge constructor
ERectangleGauge rectangleGauge;
```

```
// Adjust the transition parameters
rectangleGauge.SetTransitionType(ETransitionType_Bw);
rectangleGauge.SetTransitionChoice(ETransitionChoice_LargestAmplitude);
```

```
// Set the rectangle fitting gauge position,
// size (50x30 units) and orientation (15°)
EPoint center(256.f, 256.f);
ERectangle rectangle(center, 50.f, 30.f, 15.f);
rectangleGauge.SetRectangle(rectangle);
```

```
// Measure
rectangleGauge.Measure(&srcImage);
```

```
// Get the size and the rotation angle of the fitted rectangle
float sizeX = rectangleGauge.GetMeasuredRectangle().GetSizeX();
float sizeY = rectangleGauge.GetMeasuredRectangle().GetSizeY();
float angle = rectangleGauge.GetMeasuredRectangle().GetAngle();
```

```
// Save the point gauge measurement context
rectangleGauge.Save("myRectangleGauge.gge");
```

ウェッジフィッティング

```
////////////////////////////////////////////////////////////////////
// This code snippet shows how to create a wedge measurement tool, //
// adjust the transition parameters, set the nominal gauge //
// position, perform the measurement and retrieve the result. //
////////////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EWedgeGauge constructor
EWedgeGauge wedgeGauge;
```

```
// Adjust the transition parameters
wedgeGauge.SetTransitionType(ETransitionType_Bw);
wedgeGauge.SetTransitionChoice(ETransitionChoice_NthFromBegin);
wedgeGauge.SetTransitionIndex(0);
```

```
// Set the wedge fitting gauge position, diameter (50 units),
// breadth (-25 units), starting angle (0°) and amplitude (270°)
EPoint center(256.f, 256.f);
EWedge wedge(center, 50.f, -25.f, 0.f, 270.f);
wedgeGauge.SetWedge(wedge);
```

```
// Measure
wedgeGauge.Measure(&srcImage);
```

```
// Get the inner and outer radius of the fitted wedge
float innerRadius = wedgeGauge.GetMeasuredWedge().GetInnerRadius();
float outerRadius = wedgeGauge.GetMeasuredWedge().GetOuterRadius();
```

```
// Save the point gauge measurement context
wedgeGauge.Save("myWedgeGauge.gge");
```

ゲージのグループ化

ゲージ階層

```
////////////////////////////////////////////////////////////
// This code snippet shows how to create a gauge hierarchy //
// and save it into a file. //
////////////////////////////////////////////////////////////

// EWorldShape constructor
EWorldShape worldShape;

// Gauges constructor
ERectangleGauge rectangleGauge;
ECircleGauge circleGauge1, circleGauge2;

// ...

// Attach the rectangle gauge to the EWorldShape
rectangleGauge.Attach(&worldShape);

// Attach the circle gauges to the rectangle gauge
circleGauge1.Attach(&rectangleGauge);
circleGauge2.Attach(&rectangleGauge);

// Set the first circle gauge name
circleGauge1.SetName("myCircleGauge1");

// ...

// Save worldShape together with its daughters
worldShape.Save("myWorldShape.gge", true);
```

複雑な測定

```
////////////////////////////////////////////////////////////
// This code snippet shows how to trigger the measurement //
// of a whole gauge hierarchy and retrieve the results. //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage;

// EWorldShape constructor
EWorldShape worldShape;

// Load the EWorldShape together with its daughters
worldShape.Load("myWorldShape.gge", true);

// Retrieve the number of worldShape's daughters
int numDaughters= worldShape.GetNumDaughters();

// ...

// Trigger the measurement of all the
// gauges attached to the EWorldShape
worldShape.Process(&srcImage, true);
```

```
// Retrieve the measurement result of
// the first daughter (a rectangle gauge)
ERectangleGauge* rectangleGauge= (ERectangleGauge*)worldShape.GetDaughter(0);
float sizeX= rectangleGauge->GetMeasuredRectangle().GetSizeX();
```

```
// Retrieve the measurement result of a
// daughter gauge called "myCircleGauge1"
ECircleGauge* circleGauge= (ECircleGauge*)worldShape.GetShapeNamed
("myCircleGauge1");
EPoint center= circleGauge->GetMeasuredCircle().GetCenter();
```

EWorldShapeを用いたキャリブレーション

推測によるキャリブレーション

```
////////////////////////////////////////////////////////
// This code snippet shows how to perform a calibration //
// by guesswork. //
////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// ...
```

```
// Compute the calibration coefficients
// Field of view: 32x24 mm
worldShape.SetSensor(srcImage.GetWidth(), srcImage.GetHeight(), 32.f, 24.f);
```

```
// Retrieve the spatial resolution
float resolutionX= worldShape.GetXResolution();
float resolutionY= worldShape.GetYResolution();
```

ランドマークベースのキャリブレーション

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a landmark-based //
// calibration. //
////////////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// ...
```

```
// Reset the calibration context
worldShape.EmptyLandmarks();
```

```
// Loop on the landmarks
for(int index= 0; index < numLandmarks; index++
{
// Get the I-th landmark as a pair of EPoint(x, y)
        EPoint sensorPoint, worldPoint;
```

```
// Retrieve and store the relevant data into worldPoint and sensorPoint
// ...
```

```
// Add the I-th pair
        worldShape.AddLandmark(sensorPoint, worldPoint);
}
```

```
// Perform the calibration
worldShape.Calibrate(ECalibrationMode_Skewed);
```

ドット グリッド ベースのキャリブレーション

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a dot grid-based //
// calibration. //
////////////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// ...
```

```
// Reset the calibration context
worldShape.EmptyLandmarks();
```

```
// Loop on the dots
for(int index= 0; index < numDots; index++)
{
// Get the I-th dot as an EPoint(x, y)
        EPoint dotPoint;
```

```
// Retrieve and store the relevant data into dotPoint
// ...
```

```
// Add the I-th dot
        worldShape.AddPoint(dotPoint);
}
```

```
// Reconstruct the grid topology
// pitch X and Y = 5 units
worldShape.RebuildGrid(5, 5);
```

```
// Perform the calibration
// the calibration modes are computed automatically
worldShape.AutoCalibrate(true);
```

座標変換

```
////////////////////////////////////////////////////////////
// This code snippet shows how to convert coordinates from //
// the Sensor space to the World space and conversely. //
////////////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// EPoint constructor
EPoint sensor;
EPoint world;
```

```
// ...
```

```
// Perform the calibration
worldShape.Calibrate(ECalibrationMode_Scaled | ECalibrationMode_Skewed);
```

```
// Retrieve the world coordinates of a point, knowing its sensor coordinates
world= worldShape.SensorToWorld(sensor);
```

```
// Retrieve the sensor coordinates of a point, knowing its world coordinates
sensor= worldShape.WorldToSensor(world);
```

画像アンワーピング

```
//////////////////////////////////////////////////////
// This code snippet shows how to unwarp an image based //
// of the computed calibration coefficients. //
//////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// Lookup table constructor
EUnwarpingLut lut;
```

```
// ...
```

```
// Perform the calibration
worldShape.Calibrate(ECalibrationMode_Tilted | ECalibrationMode_Radial);
```

```
// Setup the lookup table for unwarping
worldShape.SetupUnwarp(&lut, &srcImage, true);
```

```
// Perform the image unwarping
worldShape.Unwarp(&lut, &srcImage, &dstImage, true);
```