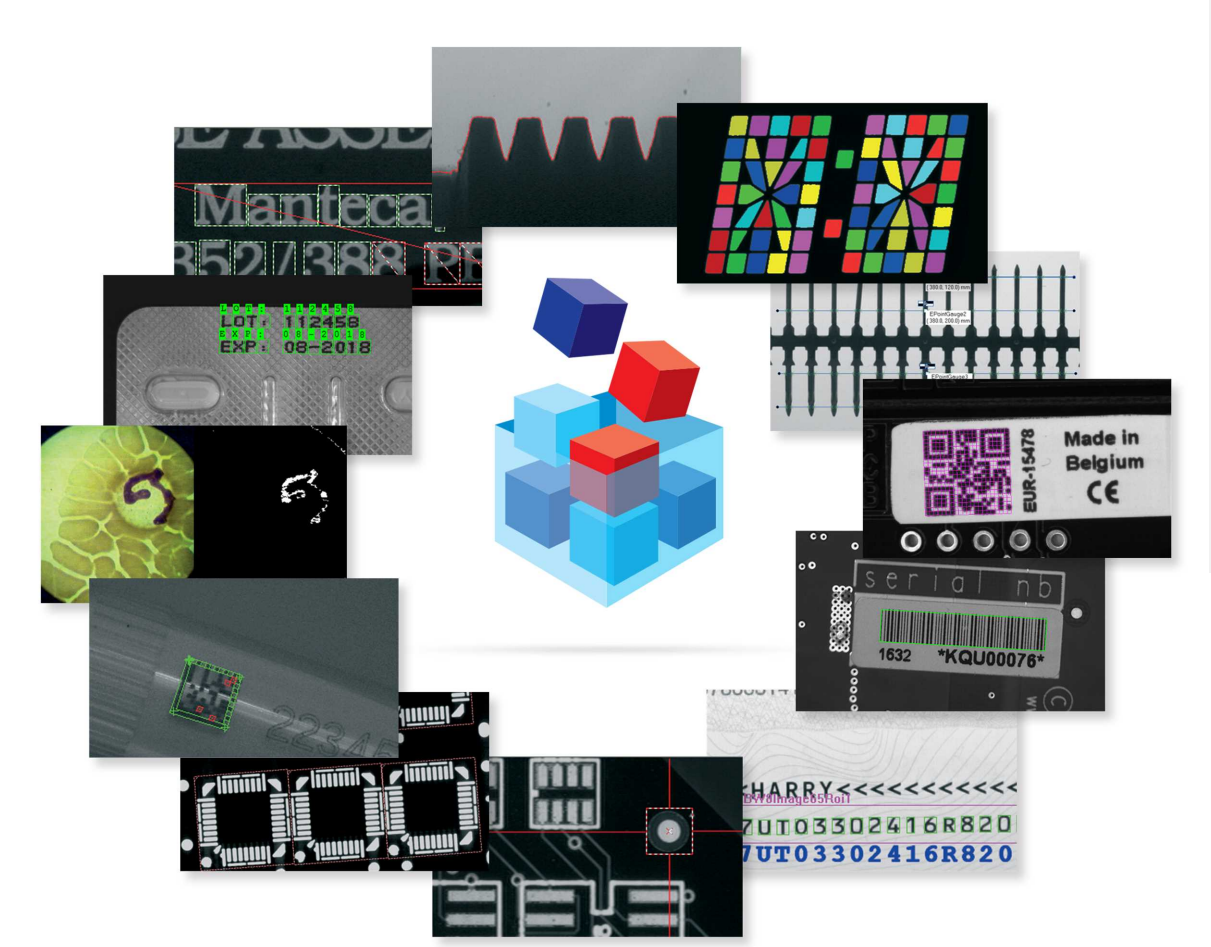
EURESYS™
Excellence in vision

用户指南

Open eVision

.NET 2.3



使用条款

EURESYS s.a. 应保留硬件和软件文档以及 EURESYS s.a. 商标的所有财产权、所有权和利益。

文档中提及的所有公司和产品的名称可能是其各自所有者的商标。

未经事先通知，不得对本书中包含的 EURESYS s.a 的 硬件或软件、品牌或文档进行许可、使用、出租、租赁、翻译、复制、复印或修改。

EURESYS s.a. 可能随时自行修改产品规格或更改本文档中给出的信息，恕不另行通知。

EURESYS s.a. 对于使用其硬件或软件而引起的任何类型的收入、利润、商誉、数据、信息系统损失或损害，或与使用其硬件或软件相关的，或因本文档遗漏或错误造成的其他特殊的、偶然的、间接的、后果性的或惩罚性的损害赔偿，概不负责。

本文档随Open eVision2.3.0(doc build2018-01-19) 提供。

目录

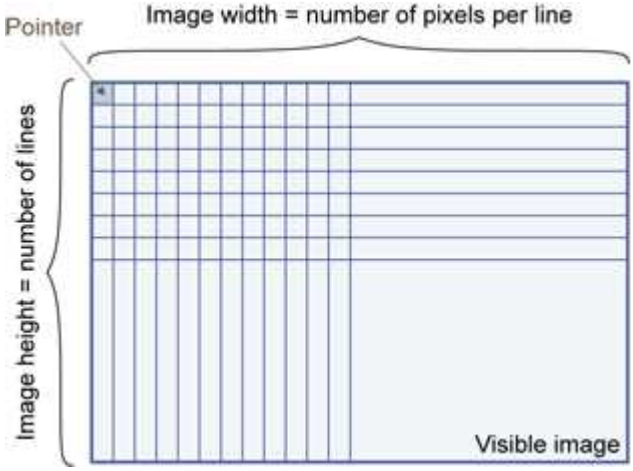
1. 利用像素容器和文件处理

1.1. 像素容器定义

图像

Open eVision 图像对象包含表示矩形图像的图像数据。

每个图像对象都有一个数据缓冲区，可以通过一个指针连续存储像素值。



图像主要参数

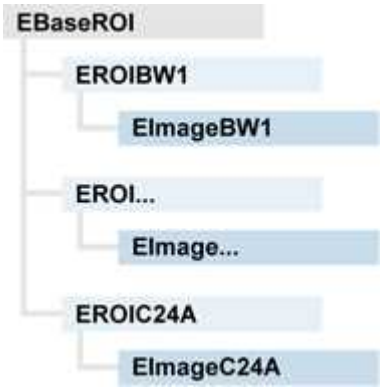
Open eVision 图像对象具有以 `EBaseROI` 参数为特征的矩形矩阵。

- 宽度是每行图像的列数(像素数) 。
- 高度是图像的行数。(Open eVision 32 位中的最大宽度/高度为 32,767(2^{15} -1) ，Open eVision 64 位中则为 2,147,483,647(2^{31} -1) 。)
- 大小是宽度和高度。

平面参数包含颜色分量的数量。灰度图像 = 1。彩色图像 = 3。

类

图像和 ROI 类派生自抽象类 `EBaseROI` ，并继承其所有属性。



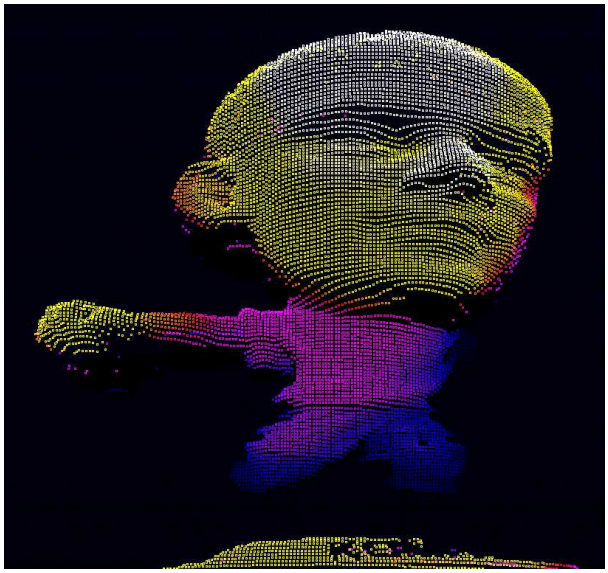
深度图

深度图是使用2D灰度图像表示3D对象的方法，图像中的每个像素表示一个3D点。

像素坐标是点的X和Y坐标的表示，而像素的灰度值是点的Z坐标的表示。

点云

点云(https://en.wikipedia.org/wiki/Point_cloud) 是非结构化的一组3D点，表示对象表面上的离散位置。



3D点云由各种3D扫描技术(如激光三角测量、飞行时间或结构光)产生。

1.2. 像素容器类型

图像

根据像素类型，支持几种图像类型：黑白、灰阶、彩色等。

Easy.GetBestMatchingImageType 返回磁盘上给定文件的最佳匹配图像类型。

BW1	1 位黑白图像(8 个像素存储在 1 个字节中)	EImageBW1
BW8	8 位灰度图像(每个像素存储在 1 个字节中)	EImageBW8
BW16	16 位灰度图像(每个像素存储在 2 个字节中)	EImageBW16
BW32	32 位灰度图像(每个像素存储在 4 个字节中)	EImageBW32
C15	15 位彩色图像(每个像素存储在 2 个字节中)。 兼容 Microsoft®Windows RGB15 彩色图像和 MultiCam RGB15 格式。	EImageC15
C16	16 位彩色图像(每个像素存储在 2 个字节中)。 兼容 Microsoft®Windows RGB16 彩色图像和 MultiCam RGB16 格式。	EImageC16

C24	C24 图像存储 24 位彩色图像(每个像素存储在 3 个字节中)。 兼容 Microsoft®Windows RGB24 彩色图像和 MultiCam RGB24 格式。	EImageC24
C24A	C24A 图像存储 32 位彩色图像(每个像素存储在 4 个字节中)。 兼容 Microsoft®Windows RGB32 彩色图像和 MultiCam RGB32 格式。	EImageC24A

深度图

8和16位深度图值存储在与2D Open eVision图像兼容的缓冲区中。

EDepth8	8位深度图(以1个字节整数存储每个像素)	EDepthMap8
EDepth16	16位深度图(以2个字节定点格式存储每个像素)	EDepthMap16
EDepth32f	32位深度图(以4个字节浮点格式存储每个像素)	EDepthMap32f

点云

点云	点坐标集(存储为浮点数)	E3DPointCloud

1.3. 支持的图像文件类型

类型	描述
BMP	未压缩的图像数据格式(Windows 位图格式)
JPEG	联合图像专家组签发的有损数据压缩标准为 ISO/IEC 10918-1。压缩不可逆地损失图像质量。
JFIF	JPEG 文件交换格式
JPEG-2000	联合图像专家组签发的数据压缩标准为 ISO/IEC 15444-1 和 ISO/IEC 15444-2。Open eVision 仅支持有损压缩格式、文件格式和代码流变体。 - **代码流**描述了图像样本。 - **文件格式**包括元信息，如图像分辨率和颜色空间。
PNG	无损数据压缩方法(便携式网络图形)。

类型	描述
序列化	Euresys 专有图像文件格式，从 Open eVision 图像对象的**序列化**获得。
TIFF	标签图像文件格式目前由 Adobe Systems 控制，并使用 LibTIFF 第三方库处理为 5.0 或 6.0 TIFF 规范编写的图像。 文件**保存**操作是无损的，对所有其他的 1 位二进制像素类型和 LZW 压缩使用 CCITT 1D 压缩。 文件**加载**操作支持 LibTIFF 规范中列出的所有 TIFF 变体。

1.4. 像素和文件类型兼容性

深度图到图像的转换

对于8位和16位的深度图，`AsImage()`方法返回Open eVision的2D处理功能可使用的兼容图像对象(分别为`EImageBW8`和`EImageBW16`)。

像素和文件类型兼容性

像素访问

访问像素的推荐方法是使用 `SetImagePtr` 和 `GetImagePtr` 将图像缓冲区访问嵌入到您自己的代码中。另见图像构造和内存分配和检索像素值。

因为每个函数调用发生的开销，应该限制以下方法的使用：

直接访问

`EROIBW8.GetPixel` 和 SetPixel 方法在所有图像和 ROI 类中实现，以在给定坐标处读取和写入像素值。要扫描图像的所有像素，您可以在 X 和 Y 坐标上运行双循环，并使用每次迭代的 `GetPixel` 或 `SetPixel` ，但不建议这样做。

出于性能原因，当需要处理大量像素时，不应使用这些访问器。在这种情况下 `GetBufferPtr()`，建议使用和迭代指针检索内部缓冲区指针。

C++ 快速访问 BW8 像素

在 BW8 图像中，调用 `EBW8PixelAccessor . GetPixel` 或 `SetPixel` 将比直接 `EROIBW8. GetPixel` 或 `SetPixel` 更快。

支持的结构

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

(*) 这些格式支持 RGB15(5-5-5 位打包)，RGB16(5-6-5 位打包)和 RGB32(RGB + alpha 通道)，但它们在任何处理之前必须使用 `EasyImage.Convert` 转换至/自 EC24。

注意： *eVision 6.5 之前版本的转换应该是无缝的：图像像素类型使用整型类型的 typedef 定义，像素值被视为无符号数，并提供到/从先前类型的隐式转换。*

加载或保存操作期间的像素和文件类型兼容性

类型	BMP	JPEG	JPEG2000	PNG	TIFF	序列化
BW1	好	不可用	不可用	好	好	好
BW8	好	好	好	好	好	好
BW16	不可用	不可用	好	好	好 (***)	好
BW32	不可用	不可用	不可用	不可用	好 (***)	好
C15	好	好 (**)	好 (**)	好 (**)	好 (**)	好
C16	好	好 (**)	好 (**)	好 (**)	好 (**)	好
C24	好	好	好	好	好 (**)	好
C24A	好	不可用	不可用	好	不可用	好
Depth8	好	好	好	好	好	好
Depth16	不可用	不可用	好	好	好 (***)	好
Depth32f	不可用	不可用	不可用	不可用	不可用	好

不可用：不支持。如果使用组合，则会发生异常。

好：图像完整性保留，无数据丢失(除了 JPEG 和 JPEG2000，有损压缩)。

(**) C15 和 C16 格式在保存操作期间自动转换为 C24。

(***) 基准 TIFF 读取器不支持 BW16 和 BW32。

1.5. 颜色类型

EISH: 强度、饱和度、色调色彩系统。

ELAB：CIE 亮度、a*、b* 色彩系统。

ELCH：亮度、色度、色调色彩系统。

ELSH：亮度、饱和度、色调色彩系统。

ELUV：CIE 亮度、u*、v* 色彩系统。

ERGB：NTSC/PAL/SMPTE 红、绿、蓝色色彩系统。

EVSH：值、饱和度、色调色彩系统。

EXYZ：CIE XYZ 色彩系统。

EYIQ：CCIR 明亮度、同相、正交色彩系统。

EYSH：CCIR 明亮度、饱和度、色调色彩系统。

EYUV：CCIR 明亮度、U 色度、V 色度色彩系统。

2. 利用像素容器和文件

2.1. 像素容器文件保存

图像和深度图

Save 方法使用两个参数将图像或深度图的数据保存到文件中：

- 路径：路径、文件名和文件扩展名。
- 图像文件类型。如果省略，则使用文件扩展名。

大于 65,536(宽度或高度) 的图像必须以 Open eVision 专有格式保存。

在以下情况下，保存会引发异常：

- 所请求的图像文件格式与图像像素类型不兼容
- 不支持自动文件类型选择方法和文件扩展名

保存16位深度图时，会丢失定点精度，像素被视为16位整数。

图像文件类型参数

参数	图像文件类型
EImageFileType_Auto(*)	由文件扩展名自动确定。见下文。
EImageFileType_Euresys	Open eVision 序列化。
EImageFileType_Bmp	Windows 位图 - BMP
EImageFileType_Jpeg	JPEG 文件交换格式 - JFIF
EImageFileType_Jpeg2000	JPEG 2000 文件格式/代码流-JPEG2000
EImageFileType_Png	便携式网络图形 - PNG
EImageFileType_Tiff	标签图像文件格式 - TIFF

(*) 默认值。

如果参数为 ImageFileType_Auto 或丢失，则分配图像文件类型

文件扩展名(*)	自动分配的图像文件类型
BMP	Windows 位图格式
JPEG、JPG	JPEG 文件交换格式 - JFIF
JP2	JPEG 2000 文件格式
J2K、J2C	JPEG 2000 代码流
PNG	便携式网络图形
TIFF、TIF	标记图像文件格式

(*) 不区分大小写。

保存 JPEG 和 JPEG2000 有损压缩

SaveJpeg 和 SaveJpeg2K 在保存压缩图像时指定压缩质量。它们有两个参数：

- 路径：包括路径、文件名和文件扩展名的字符串。
- 压缩图像文件的质量，整数值范围[0：100]。
  SaveJpeg 使用 JPEG 文件交换格式 - JFIF 保存图像数据。
  SaveJpeg2K 使用 JPEG 2000 文件格式保存图像数据。

JPEG 压缩值

JPEG 压缩	描述
JPEG_DEFAULT_QUALITY(-1)	默认质量(*)
100	卓越的图像质量，最低的压缩系数
75	良好的画质(*)
50	正常图像质量
25	平均图像质量
10	图像质量差

(*) 默认质量对应于良好的图像质量(75)。

代表性的 JPEG 2000 压缩质量值

JPEG 2000 压缩	描述
-1	默认质量(*)
1	最好图像质量，最低的压缩系数
16	良好图像质量(*)(16：1 比率)
512	最低图像质量，最高的压缩系数

(*) 默认质量对应于良好的图像质量(16:1 比率)。

点云

- 使用Save方法以Open eVision专有文件格式保存点云。
- 使用SavePCD方法将点云保存在与其他软件(如PCL(Point Cloud Library)) 兼容的文件中。

2.2. 像素容器文件加载

图像和深度图

- 使用Load 方法将图像数据加载到图像对象中：
  - 它有一个参数：**路径**：路径、文件名和文件扩展名。
  - 文件类型由文件格式决定。
  - 目标图像会根据磁盘上图像的大小自动调整大小。
- Load 方法会在以下情况下引发异常：
  - 文件类型识别失败
  - 文件类型与图像对象的像素类型不兼容

Open eVision 1.1 及更新版本的序列化图像文件与以前的 Open eVision 版本的序列化图像文件不兼容。

在深度图中加载BW16图像(整数值) 时，保持深度图(默认为0) 中设置的定点精度不变并使用。

点云

- 使用Load方法以Open eVision专有文件格式保存点云。
- 使用LoadPCD方法将点云保存在与其他软件(如PCL(Point Cloud Library)) 兼容的文件中。

2.3. 内存分配

可以使用内部或外部存储器分配来构造图像。

内部内存分配

图像对象动态分配和取消分配缓冲区。内存管理是透明的。
当图像尺寸变化时，会发生重新分配。
当图像对象被破坏时，缓冲区是未分配的。
使用内部内存分配声明图像：

1. 构造一个图像对象，例如 `EImageBW8` ，带宽和高度参数，或使用 `SetSize` 函数。
2. 访问给定像素。有几个函数可以做到这一点。`GetImagePtr` 返回指向给定坐标处像素第一个字节的指针。

外部内存分配

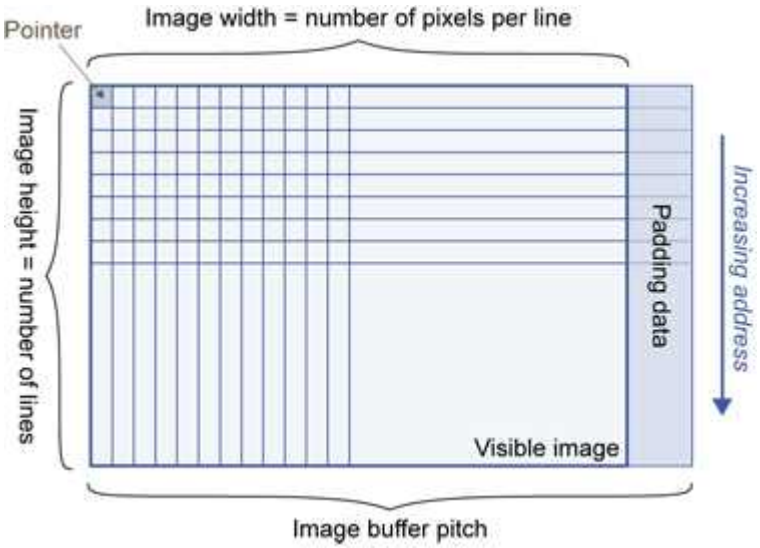
用户控制缓冲区分配或将内存缓冲区中的第三方图像链接到 Open eVision 图像。
必须指定图像大小和缓冲区地址。
当图像对象被破坏时，缓冲区不受影响。
使用外部存储分配声明图像：

1. 声明图像对象，例如 `EImageBW8` 。
2. 创建一个适当大小和对齐的缓冲区(参见图像缓冲区)。
3. 使用 `SetSize` 函数设置图像大小。
4. 使用 `GetImagePtr` 访问缓冲区。另见检索像素值。

2.4. 图像和深度图缓冲区

图像和深度图像素以 Windows 位图格式(从上至下的 DIB [1]) 从左到右依次从顶部到底部存储到相关联的缓冲区。

缓冲区地址是指向缓冲区的起始地址指针，其中包含图像的左上角像素。



图像缓冲区间距

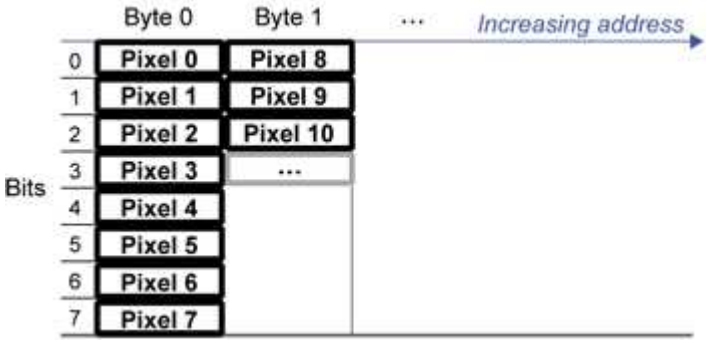
- 对齐必须是 4 个字节的倍数。
- 由于性能原因，Open eVision 从 1.2 版本起默认间距为 32 字节(Open eVision 版本 1.1.5 为 8 字节)。

[1]设备无关位图

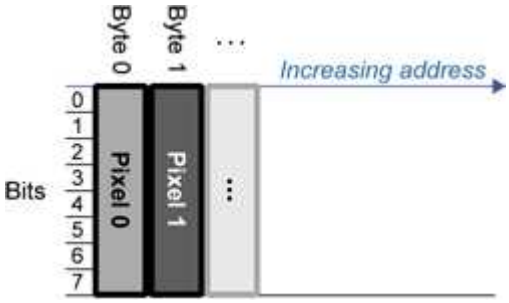
内存布局

- EImageBW1 在一个字节中存储 8 个像素。

  BW1 图像缓冲区的前 2 个像素的示例内存布局：



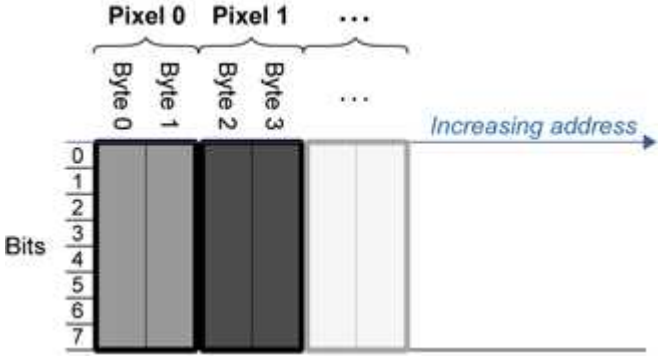
- EImageBW8和EDepthMap8以一个字节存储每个像素。



  BW8 图像缓冲区的第一批像素的示例内存布局：

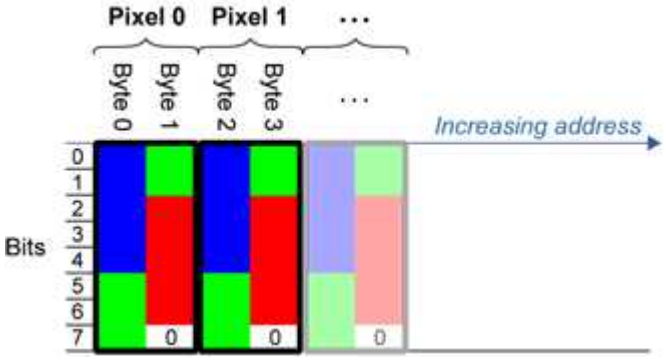
- EImageBW16 将每个像素存储在一个 16 位字(两个字节)中。

  BW16 图像缓冲区的第一批像素的示例内存布局：



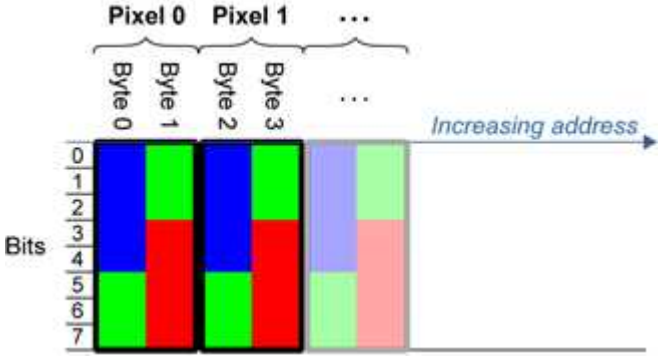
- EImageC15 以 2 个字节存储每个像素。每个颜色分量用 5 位编码。第 16 位未使用。

  C15 图像缓冲区的第一批像素的示例内存布局：



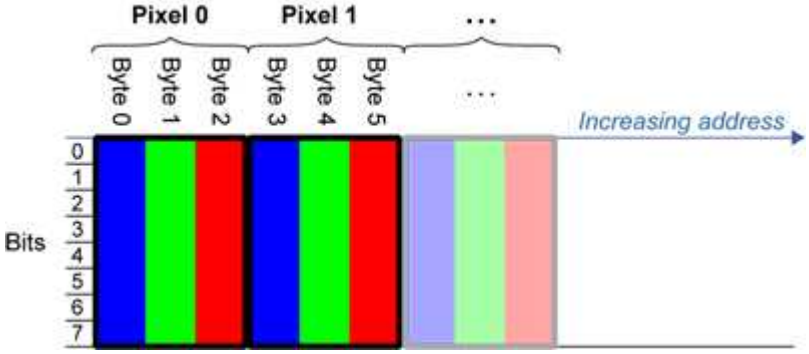
- EImageC16 以 2 个字节存储每个像素。第一和第三个颜色分量用 5 位编码。第二个颜色分量用 6 位编码。

  C16 图像缓冲区的第一批像素的示例内存布局：



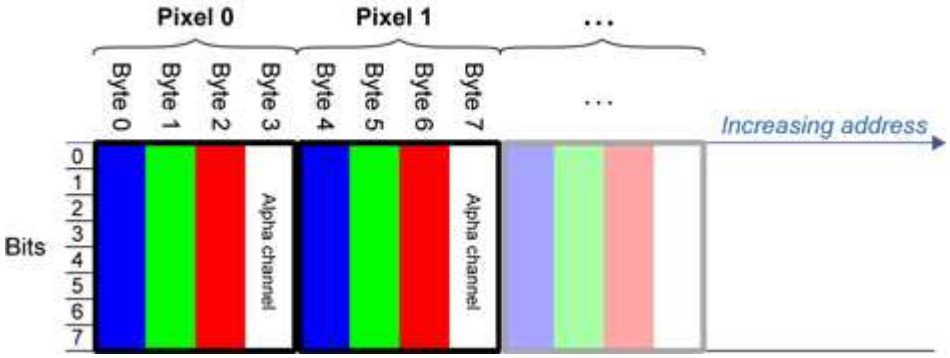
- EDepthMap16以2个字节定点格式存储每个像素。
- EImageC24 以 3 个字节存储每个像素。每个颜色分量用 8 位编码。

  C24 图像缓冲区的第一批像素的示例内存布局：



- EImageC24A 以 4 个字节存储每个像素。每个颜色分量用 8 位编码。Alpha 通道也用 8 位编码。

  C24A 图像缓冲区的第一批像素的示例内存布局：



- EDepthMap32f以4个字节浮点格式存储每个像素。

2.5. 图像绘制和覆盖

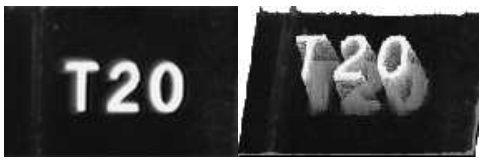
- 绘图使用 Windows GDI [1]系统调用。
  MFC [2]应用程序通常使用 `OnDraw` 事件处理程序绘制，其中指向设备上下文的指针可用。
  Borland/CodeGear 的 OWL 或 VCL 使用 **Paint** 事件处理程序。
- 256 色显示模式的调色板提供最佳渲染效果。可以使用 LUT [3](使用直方图拉伸技术或伪着色) 来改进灰度级图像。
- 缩放可以水平和垂直不同。
- `DrawFrameWithCurrentPen` 方法绘制一个框架。
- **非破坏性覆盖**绘图操作不会改变图像内容，例如 `MoveTo / LineTo` 。
- **破坏性覆盖**绘图操作通过绘制图像内的图像来更改图像内容，例如 `Easy .OpenImageGraphicContext` 。灰度级[彩色]图像只能接收灰度级[彩色]覆盖。

2.6. 2D图像的3D渲染

通过将这些图像先后绕 X 轴、Y 轴旋转来观察它们。

灰色 3D 渲染

`Easy.Render3D` 准备三维渲染，其中灰度值是高度。
可以给出三个方向的放大系数(X = 宽度、Y = 高度和 Z = 深度) 。
渲染的图像显示为独立的点，其大小可以调整以使表面或多或少不透明。



3D 渲染

彩色直方图 3D 渲染

`Easy.RenderColorHistogram`准备了对彩色图像直方图的三维渲染。
在 RGB 空间(而不是 XY 平面) 中绘制像素，以显示 RGB 值的聚类和色散。
该函数可以处理其他颜色系统中的像素(使用 EasyColor 进行转换) ，但原始 RGB 图像需要以常规颜色显示像素。

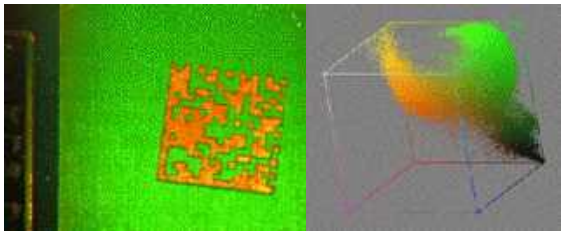
---

[1]图形设备接口

[2]Microsoft Foundation Class

[3]查找表

可以给出所有三个方向(X = 红，Y = 绿，Z = 蓝)的放大系数。

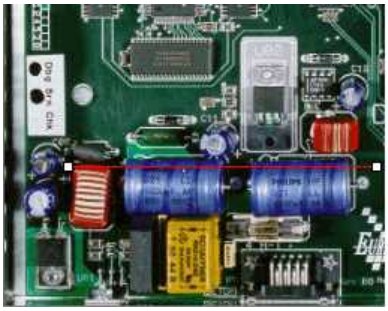


彩色直方图渲染

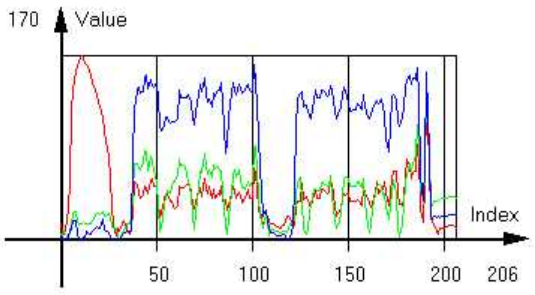
2.7. 矢量类型和主要属性

矢量是像素的一维阵列(取自图像剖面或轮廓)。

EVector 是所有向量的基类。它包含所有非类型特定的方法，主要用于计数元素和序列化。



C24 图像中的剖面



沿着剖面绘制 RGB 值

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 值数组(EC24Vector)

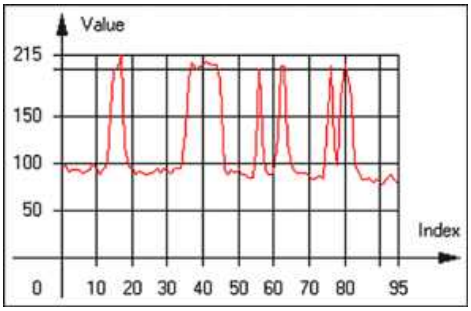
矢量管理元素数组。内存分配是透明的，因此可以动态调整向量。每当函数使用向量时，矢量类型、大小和结构都会自动调整以适应函数需求。

使用向量非常简单：

1. **创建适当类型的向量**，使用其构造函数和预分配元素(如果需要)。

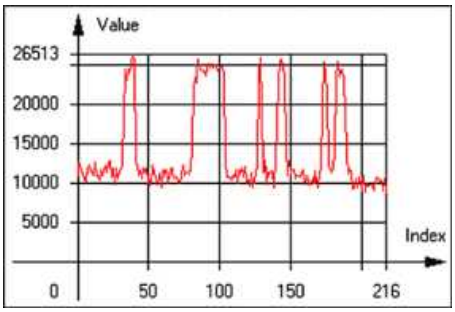
矢量类型

- EBW8Vector：通常从图像剖面提取的一系列灰度像素值(由EasyImage.Lut、EasyImage.SetupEqualize、EasyImage.ImageToLineSegment、EasyImage.LineSegmentToImage、EasyImage.ProfileDerivative...使用)。



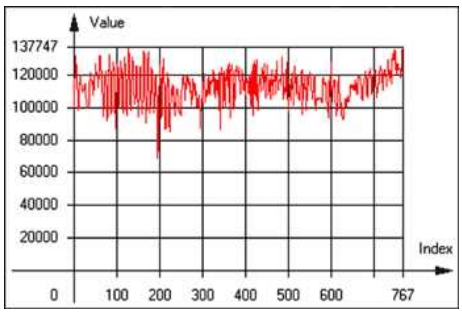
EBW8Vector 的图形表示(参见 Draw 方法)

- EBW16Vector: 使用扩展范围(16 位)的灰度级像素值序列，主要用于中间计算。



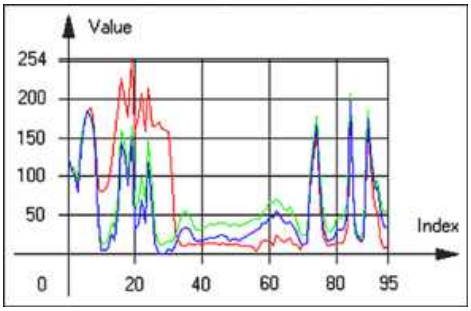
EBW16Vector 的图形表示

- EBW32Vector: 使用扩展范围(32 位)的灰度级像素值序列，主要用于中间计算。(用在 EasyImage.ProjectOnARow, EasyImage.ProjectOnAColumn ...)。



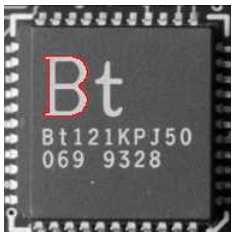
EBW32Vector 的图形表示

- EC24Vector: 通常从图像剖面提取的一系列彩色像素值(由 EasyImage.ImageToLineSegment、EasyImage.LineSegmentToImage、EasyImage.ProfileDerivative ...使用)。



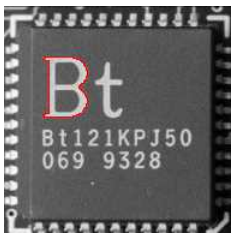
EC24Vector 的图形表示

- EBW8PathVector：从图像剖面或轮廓提取的具有相应像素坐标一系列灰度像素值（由 EasyImage .ImageToPath 、EasyImage .PathToImage...使用）。



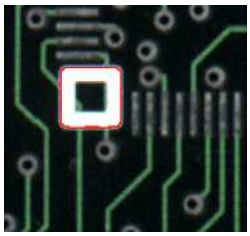
EBW8PathVector 的图形表示(参见 Draw 方法)

- EBW16PathVector：从图像剖面或轮廓提取的具有相应像素坐标一系列灰度像素值（由 EasyImage .ImageToPath 、EasyImage .PathToImage...使用）。



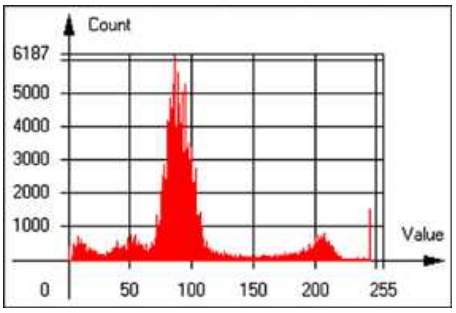
EBW16PathVector 的图形表示(参见 Draw 方法)

- EC24PathVector：从图像剖面或轮廓提取的具有相应像素坐标一系列彩色像素值（由 EasyImage .ImageToPath 、EasyImage .PathToImage...使用）。



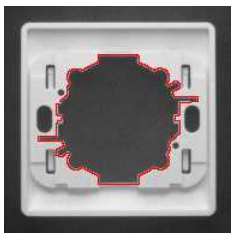
EC24PathVector 的图形表示(参见 Draw 方法)

- EBWHistogramVector：BW8 或 BW16 图像中的一系列像素频率计数
  (由 EasyImage .IsodataThreshold EasyImage .Histogram、EasyImage . AnalyseHistogram、 EasyImage. SetupEqualize ...使用)。



EBWHistogramVector的图形表示(参见 Draw 方法)

- EPathVector：一系列像素坐标。相应的像素不必是连续的
  (由 EasyImage .PathToImage 和 EasyImage .Contour 使用)。



EPathVector 的图形表示(参见 Draw 方法)

- EPeakVector ：在图像剖面中找到的峰值
  (由 EasyImage .GetProfilePeaks 使用)。
- EColorVector ：颜色的描述
  (由 EasyColor .ClassAverages 和 EasyColor .ClassVariances 使用)。

2. **用值**填充向量。首先使用 EVector .Empty成员将其清空，然后通过调用 EC24Vector.AddElement 成员一次性添加元素。您可以通过索引访问任何元素。
3. **访问向量元素**，用于读取或写入。使用括号运算符，例如EC24Vector.operator[]。
4. **确定当前元素数量**，使用成员 EVector .NumElements 。
5. **绘制向量**。
   像素向量是作为元素索引函数的元素值图，因此其图形显示取决于其类型。您可以在窗口中绘制一个矢量。为了易读性，绘图应该出现在中性的背景上。
   在与所需窗口相关联的设备上下文中完成绘制。默认情况下，以蓝色绘制曲线，以黑色绘制注释。可以定义以下参数：graphicContext、width、height、origin、origin、color0、color1、color2。
   EC24Vector 有三条曲线，而不是一条，分别对应一个颜色分量。默认情况下，使用红色、蓝色和绿色笔。

2.8. ROI 主要属性

ROI 由宽度、高度和**原点 x 和 y 坐标**定义。
相对于父图像或 ROI 中的左上角指定原点。
ROI 必须完全包含在其父图像中。
如果OrgX和宽度是 8 的倍数，则 BW1 ROI 的处理/分析时间更快。

保存并加载

您可以将 ROI 保存或加载为单独的图像，以使其像完整图像一样被使用。ROI 完全执行**无内存分配**，并且绝对不会复制其父镜像的部分，父镜像为其提供对其图像数据的访问。

新文件的图像大小必须与加载到其中的 ROI 的大小相匹配。ROI 周围的图像保持不变。

ROI 类

Open eVision ROI 从抽象类EBaseROI继承参数。

根据其像素类型，有几种 ROI 类型。它们具有与相应的图像类型相同的特征。

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

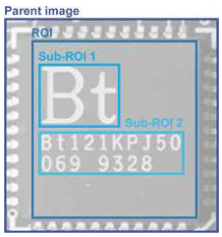
附件

ROI 必须利用设置父级、位置和大小的参数附加到父级(图像/ROI)，并且这些链接被透明更新，避免悬挂指针。
普通图像不能附加到另一个图像或 ROI。

嵌套

Set 和 Get 函数更改或查询 ROI 相对于其直接或最顶层父图像的原点宽度、高度和位置。

图像可以容纳任意数量的 ROI，其可以以分层方式嵌套。移动 ROI 也会相应地移动嵌入式 ROI。图像/ROI 类提供了几种方法来遍历与图像相关联的 ROI 层次结构。



嵌套 ROI: 两个子 ROI 附加到一个 ROI，其本身附加到父图像

裁剪

`CropToImage`裁剪部分不在其图像中的 ROI。调整后的 ROI 永远不会增长。
如果函数试图使用一个具有扩展到父级之外的限制的 ROI，将抛出异常。

> **注意：** *(在 Open eVision 1.0.1 和更早版本中，ROI 被放置在其图像之外时，会被静默地调整大小或重新定位，有时变大。如果 ROI 限制扩展到父级外部，则它们会静默地调整大小以保持在父级限制之内。)*

调整大小和移动

- ROI 可以通过两个函数和拖动手柄轻松调整大小和位置：
  - `EBaseROI.Drag`在光标移动时调整 ROI 坐标。
  - `EBaseROI.HitTest`通知是否在拖动手柄上放置光标。一旦知晓手柄，可以由 `OnSetCursor`MFC 事件处理程序改变光标形状。如果在拖动进行过程中被调用，HitTest 是不可预测的。
    HitTest 可以在 OnSetCursor MFC 事件处理程序中使用，以更改光标形状，或者在拖动操作之前，如 OnLButtonDown，
    (或 Borland/CodeGear 的 OWL 中的 EvSetCursor 和 EvLButtonDown)
    (或 Borland/CodeGear 的 VCL 中的 FormMouseMove 和 FormMouseDown) 。
    在 VB6 中，MouseDown、MouseMove、MouseUp 事件返回当前光标的位置，而不是像素，所以转换是必需的。

2.9. 灵活蒙板

ROI与灵活蒙板

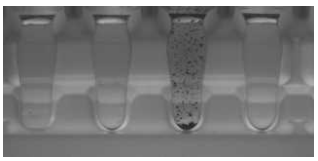
ROI 和蒙板将处理限制到图像的部分：

- "ROI 主要属性" 在上一页适用于所有 Open eVision 函数。通过减少像素数使用感兴趣的区域来加速处理。Open eVision 支持分层嵌套矩形 ROI。
- 建议使用灵活蒙板以处理断开的 ROI 或非矩形形状。它们由一些EasyObject和EasyImage库函数支持。

灵活蒙板

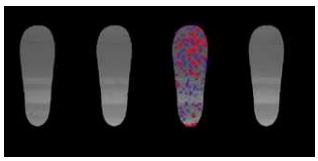
灵活蒙板是与源图像具有相同高度和宽度的 BW8 图像。它包含必须处理和忽略的区域形状(在处理过程中不会考虑):

- 值为 0 的灵活蒙板的所有像素定义了忽略区域。
- 具有任何其他不为 0 值的灵活蒙板的所有像素定义要处理的区域。



源图像

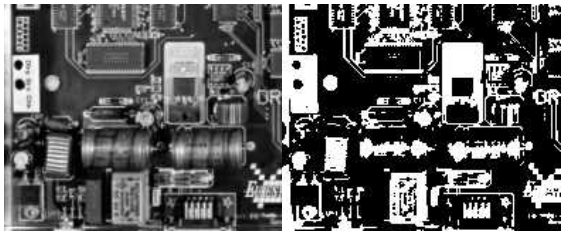
相关蒙板



已处理的蒙板图像

可以通过输出 BW8 图像的任何应用程序和一些 EasyObject 和 EasyImage 函数生成灵活的蒙板。

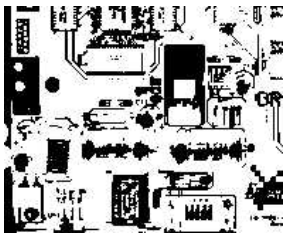
EasyImage 中的灵活蒙板



源图像(左)和蒙板变量(右)

在 EasyImage 中使用灵活蒙板的简单步骤

1. **调用 EasyImage 中将输入蒙板作为参数**的函数。例如,可以在黑色层中评估白色层之后的像素平均值。
2. **显示结果。**



产生的图像

支持灵活蒙板的 EasyImage 函数

- `EImageEncoder .编码`具有 BW1、BW8、BW16 和 C24 源图像的灵活蒙板参数。
- AutoThreshold。
- 直方图(函数 `HistogramThreshold` 没有蒙板参数重载)。

- RmsNoise, SignalNoiseRatio.
- 覆盖(BW8 源图像没有蒙板参数重载)。
- ProjectOnAColumn 、ProjectOnARow(矢量投影)。
- ImageToLineSegment、ImageToPath(矢量剖面)。

EasyObject 中的灵活蒙板

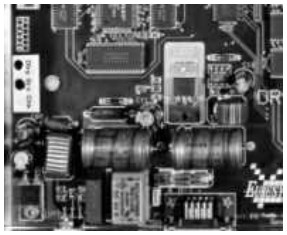
可以通过输出 BW8 图像的任何应用程序或使用 Open eVision 图像处理函数生成灵活的蒙板。
EasyObject 可以使用灵活的蒙板将斑点分析限制到图像的复杂或断开的形状区域。

如果相关对象与图像的其他区域具有相同的灰度级别，则可以使用灵活的蒙板和 `Encode` 函数定义“保持”和“忽略”区域。

灵活蒙板是与源图像具有相同高度和宽度的 BW8 图像。

- 灵活蒙板中的像素值为 0 遮罩对应的源图像像素，使其不出现在编码图像中。
- 灵活蒙板中的任何其他像素值会导致像素被编码。

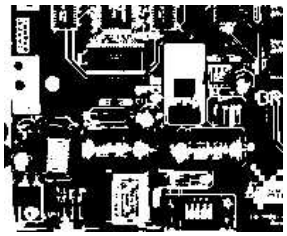
创建灵活蒙板的 EasyObject 函数



源图像

1) ECodedImage2 . RenderMask：从编码图像的层

1. 为了编码和提取灵活蒙板，首先从源图像构造编码图像。
2. 选择一个分割方法(对于上方图像，默认方法`GrayscaleSingleThreshold`是合适的)。
3. 选择要编码的编码图像的层(即 使用最小残留阈值设置的白色和黑色层)。
4. 使用 `mask.SetSize(sourceImage.GetWidth(), sourceImage.GetHeight())` 使蒙板图像成为所需的大小。
5. 利用灵活的蒙板作为`ECodedImage2.RenderMask`的参数。

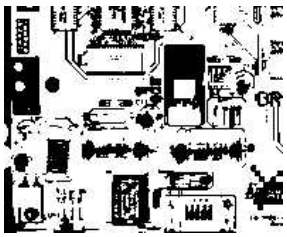


BW8 产生的图像可以用作灵活蒙板

2) ECodedElement . RenderMask：从斑点或洞

1. 选择相关编码元素。

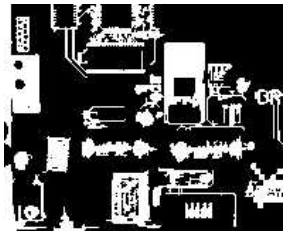
2. 使用ECodedElement.RenderMask，创建一个从编码图像的所选编码元素中提取蒙板的循环。
3. 也可选择根据这些选择的编码元素中的每一个来计算特征值。



BW8 产生的图像可以用作灵活蒙板

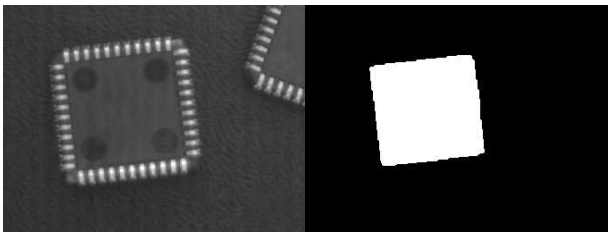
3) EObjectSelection .RenderMask：从选择的斑点

例如，EObjectSelection . RenderMask 可以丢弃由噪声导致的小型对象。



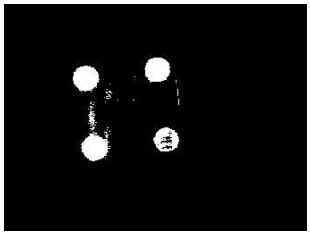
BW8 产生的图像可以用作灵活蒙板

示例：限制 EasyObject 编码的区域



查找四个圆形(左)灵活的蒙板可以隔离中央芯片(右)

1. 声明一个新的 ECodedImage2 对象。
2. 设置变量：首先声明源图像和灵活蒙板，然后加载它们。
3. 声明 EImageEncoder 对象，如果适用，请选择适当的分割器。设置分割器并选择适当的编码层。
4. 编码源图像。然后，只需将灵活蒙板中的区域编码一层就可以很简单。
   我们看到，使用灰度单阈值分割器在黑色层中正确分割了圆圈：



5. 选择编码图像的所有对象。
6. 通过过滤掉太小的对象来选择相关对象。
7. 通过迭代所选对象显示所选特征来显示斑点特征。

2.10. 剖面

剖面抽样

剖面是沿图像中的线/路径/轮廓采样的一系列像素值。

- EasyImage .ImageToLineSegment 将沿给定线段(任意定向并完全包含在图像内)的像素值复制到向量。自动调整矢量长度。该函数支持灵活的蒙板。
- 路径是存储在向量中的一系列像素坐标。
  EasyImage . ImageToPath 将相应的像素值复制到矢量。该函数支持灵活的蒙板。
- **轮廓**是一个闭合或非(连接)路径，形成对象的边界。
  EasyImage .Contour 跟随对象的轮廓，并将其组成的像素值存储在剖面矢量中。

剖面分析

可以处理配置文件以查找峰值或转换：

- 转换对应于对象边缘(黑色到白色或白色到黑色)。可以通过获取信号(将转换(边缘)转换为峰值)的第一个**派生物**并在其中查找峰值来检测。
  EasyImage .ProfileDerivative 计算从灰度图像中提取的剖面的第一个导数。
  EBW8 数据类型仅处理无符号值，因此导数向上移位 128。小于[大于] 128 的值对应于负[正]导数(递减[递减]斜率)。
- 峰值是信号部分，高于或低于给定阈值 - 信号的最大或最小值。这可能对应于白线或黑线或瘦型特征的交叉位置。它由其如下各项定义：
  - 幅度：阈值与最大[或最小]信号值之间的差异。
  - 区域：给定阈值处的信号曲线与水平线之间的表面。

  EasyImage .GetProfilePeaks 检测灰度级剖面中的最大和最小峰值。为了消除由于噪声导致的假峰值，使用两个选择标准。结果存储在峰值矢量中。

剖面插入图像中

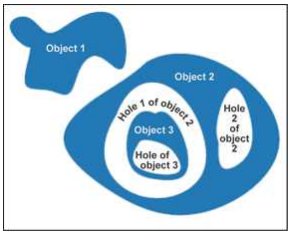
EasyImage.LineSegmentToImage 将像素值从矢量或常量复制到给定线段的像素(任意定向并完全包含在图像内)。

EasyImage .PathToImage 将像素值从矢量或常量复制到给定路径的像素。

3. 检测工具

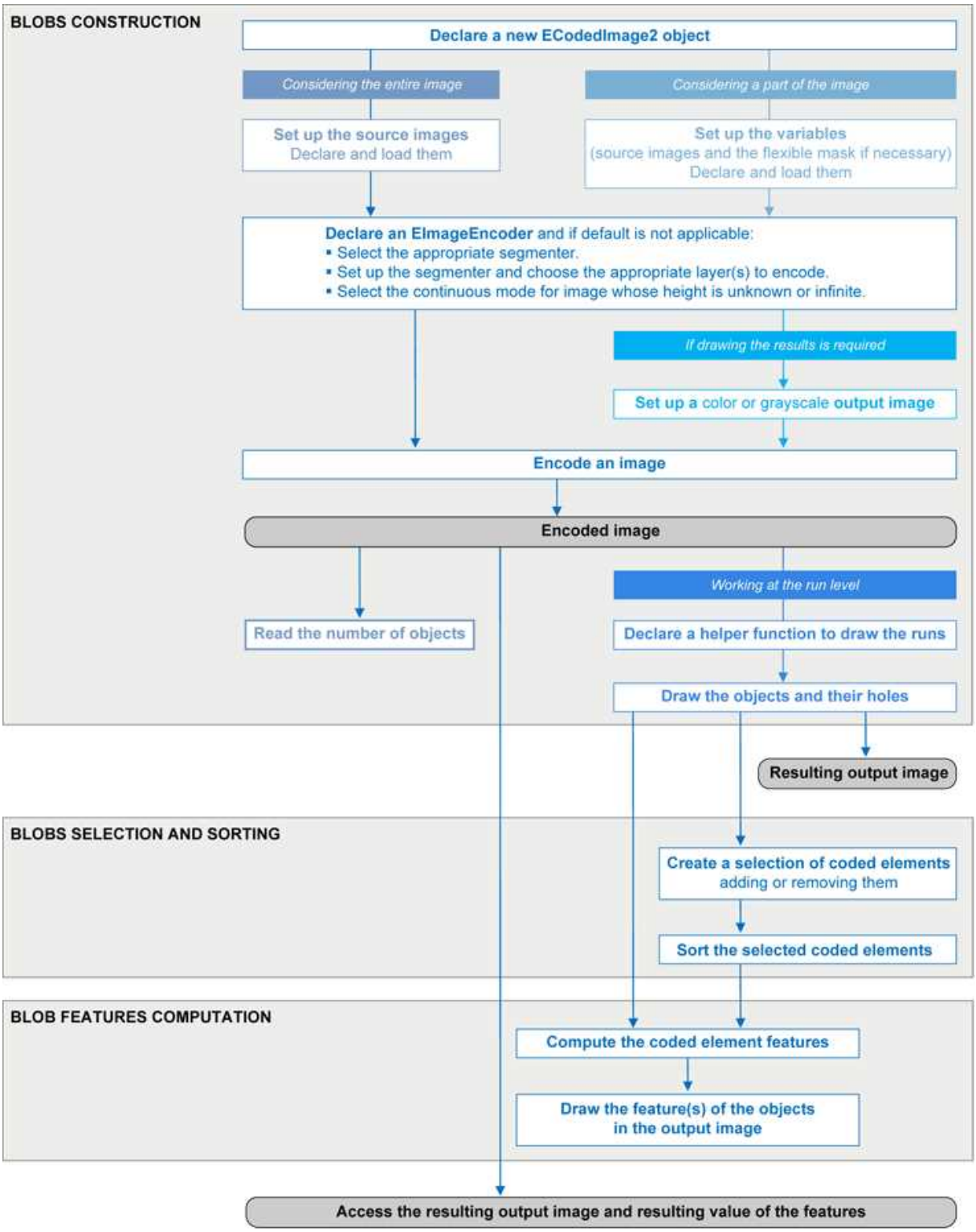
3.1. EasyObject - 分析二进制对象

EasyObject 库通过创建和处理斑点(具有相同灰度级别的对象或孔)来选择图像中的特征。



该库可用于 BW1、BW8、BW16 和 C24 源图像，可从 `ECodedImage2` 类访问，该类具有改进的执行时间，特别是对于具有多个对象的大图像。

工作流


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

斑点定义

斑点是相同灰度级范围的相邻像素的分组。
斑点可以是对象中的对象或孔。EasyObject 函数分析对象和孔。
当构建斑点时，计算孔和对象之间的夹杂关系。

即使孔可能是实际的感兴趣的物体，找到感兴趣的物体更容易，然后用 EasyObject 检测其孔(使用 EasyObject)并测量其特性(使用 EasyGauge 或 EasyObject)。

将作为独立实体处理：

- 它们可以通过它们所属的层，它们的位置，矩形 ROI 或其计算特征来选择。选择标准可以组合(选择小对象；其中，选择靠近右边缘的...)。
- 它们可以按其几何特征列出和排序：例如惯性的面积、宽度或椭圆。

斑点分析可以限于矩形和嵌套的 ROI 以及使用灵活蒙板的复杂或断开形状的区域。

构建斑点

EasyObject 选择感兴趣的对象并在两个步骤中构建斑点/孔：

1. 分割：对源图像像素进行分类，创建图层并构建运行(运行是一行中相邻像素的序列，共享相同的属性)。
2. 编码：汇编运行以构建每个图层的斑点。
   可以选择保留哪些对象或孔。
   EImageEncoder .编码分析斑点并将结果存储到具有一组叠加的互斥图像层的编码图像中，其中每层的像素具有共同的属性，诸如高于阈值。
   灵活的蒙板可以将编码限制到任意形状的区域。

不需要构建孔，它们在需要时即时构建。

函数

- 分割GetSegmentationMethod 和 SetSegmentationMethod
- 灰度单阈值 EGrayscaleSingleThresholdSegmenter
- 灰度级双阈值 EGrayscaleDoubleThresholdSegmenter
- 彩色单阈值 EColorSingleThresholdSegmenter
- 彩色范围阈值 EColorRangeThresholdSegmenter
- 参考图像 EReferenceImageSegmenter
- 图像范围 EImageRangeSegmenter
- 标记图像 ELabeledImageSegmenter
- 二进制图像 EBinaryImageSegmenter

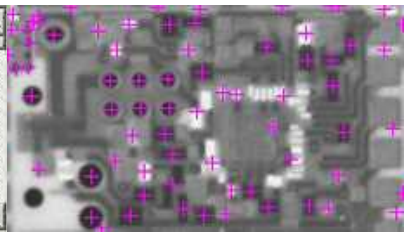
像素聚合(编码器)

- 图层选择
- 对象构造：在对象中运行聚集
- 格孔构造：在对象中运行聚集

提取对象(使用几何参数)

一旦图像被编码,编码的元素(对象或孔)可以通过抽象类 ECodedElement 访问,它提供适用于特定编码元素的大量方法:

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



特征计算和显示

可有效地随机访问目标、孔洞及其特征(通过基于索引的方式)。

图像分割器

有几种方法来分割像素。使用GetSegmentationMethod和SetSegmentationMethod对该方法进行选择。

1. 灰度单阈值(默认)

EGrayscaleSingleThresholdSegmenter 适用于 BW8 和 BW16 灰度图像,并生成具有两层的编码图像:

- **黑色层**(通常是第 0 层)包含灰度值低于阈值的未屏蔽像素.
- **白色层**(通常是第 1 层)包含剩余的未屏蔽像素,即 具有大于或等于阈值灰度值的未屏蔽像素。

EasyObject 提供了 5 种设置阈值方法:

- **绝对**(整数值):表示白色层的第一个灰度值。使用 SetAbsoluteThreshold 方法设置,并使用 GetAbsoluteThreshold 方法。
- **相对**(%):表示属于黑色层的图像像素的分数,它是用户定义的浮点值,范围为 0 到 1。设置 SetRelativeThreshold 方法,并使用 GetRelativeThreshold 方法。
- **最低残值**(默认):阈值是自动计算的值,使源和阈值图像之间的二次方差最小化。
- **最大熵**:自动计算值,使得熵(即 所得阈值图像的信息量)最大化。
- **IsoData**:位于平均暗灰度值(灰度低于阈值)和平均浅灰度值(高于阈值的灰度级)之间的自动计算值。

默认情况下,使用最小残差设置阈值法的灰度单阈值。只有像素具有高于该阈值的对象才被编码。

2. 灰度双阈值

EGrayscaleDoubleThresholdSegmenter适用于 BW8 和 BW16 灰度图像,并生成具有三层

的编码图像：

- **黑色层**(通常是第 0 层)包含灰度值低于低阈值的未屏蔽像素。
- **白色层**(通常是第 2 层)包含灰度值高于或等于高阈值的未屏蔽像素。
- **中性层**(通常是第 1 层)包含剩余的未屏蔽像素。

低阈值和**高阈值**是由 `SetLowThreshold` 和 `SetHighThreshold` 方法设置的用户定义的整数值，与 `GetLowThreshold` 和 `GetHighThreshold` 方法一起产生。

3. 色彩单阈值

`EColorSingleThresholdSegmenter` 适用于 C24 彩色图像；它生成具有两层的编码图像：

- **白色层**(通常是第 1 层)包含属于由阈值点和白点(255,255,255)定义的颜色空间的立方体未屏蔽像素。
- **黑色层**(通常是第 0 层)包含剩余的未屏蔽像素。

颜色阈值是一组三个**用户定义的**整数值，用于指定颜色空间中的颜色，利用 `SetThreshold` 方法设置，并与 `GetThreshold` 方法一起产生。

4. 色彩范围阈值

`EColorRangeThresholdSegmenter`适用于 C24 彩色图像；它生成具有两层的编码图像：

- **白色层**(通常是第 1 层)包含属于由低阈值点和高阈值点定义的颜色空间立方体的未屏蔽像素。
- **黑色层**(通常是第 0 层)包含剩余的未屏蔽像素。

低阈值和高阈值分别是一组三个用户定义的整数值，指定颜色空间中的颜色，利用 `SetLowThreshold` 和 `SetHighThreshold` 方法设置，并与 `GetLowThreshold` 和 `GetHighThreshold` 方法一起产生。

5. 图像范围

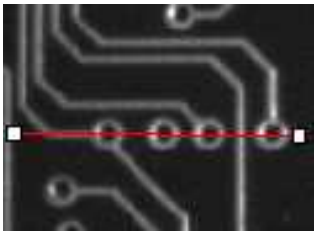
以下情况，需要使用**逐像素设置阈值**进行分割，该设置阈值给出了每个像素的允许范围值：

- 当背景不够统一时，
- 当图像上的照明不均匀时，
- 当仅增加图像和参考图像(理想)之间的差异时，

使用两个图像指定每个像素的允许范围：具有每个像素允许的最小值的低参考图像，具有最大值的高参考图像。因此，参考图像是源图像在整个图像上减去(或加)固定值(假设噪声分布是均匀的和相加的)。
困难在于准备合适的高和低参考图像。

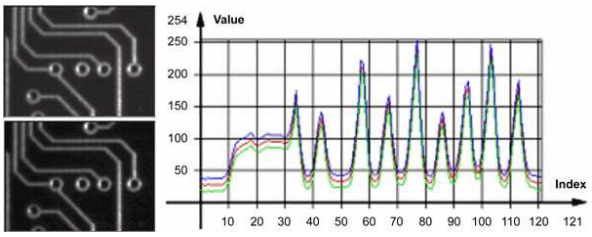
准备高和低参考图像

您可以从没有缺陷的场景图像开始，并在比较之前添加安全限度。



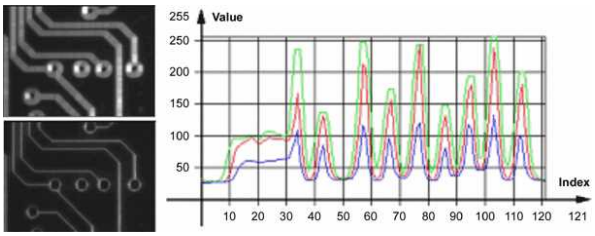
源图像

必须为噪声和照明变化提供灰度公差。



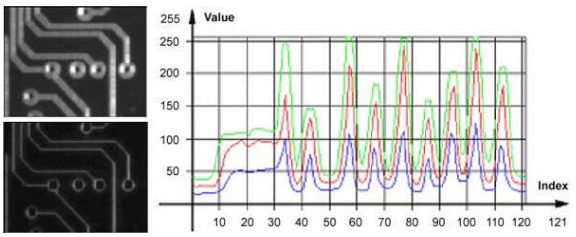
灰度公差限度

图像可能在某些方向上有轻微的变化，这些变化可以通过使用膨胀和腐蚀形态操作扩大明暗区域来校正。该几何公差限度大致与形态滤波器尺寸一样大。



几何公差限度

结合两种公差限度提供最好的结果。



合并限度

图像分割器

EImageRangeSegmenter and EReferenceImageSegmenter适用于 BW8、BW16 和 C24 图像；生成具有两层的编码图像。

通过与源图像相同类型的两个参考图像分别为每个像素定义的低阈值和高阈值：低图像和高图像。参考图像分别定义每个像素的参考阈值。

- 对于**灰度**图像，**白色层**(通常是第 1 层)包含在低图像、高图像或参考图像中相应的未遮罩像素的灰度值定义的范围内具有灰度值的未遮罩像素。
- 对于**彩色**图像，**白色层**(通常是第 1 层)包含未遮罩的像素，其颜色在由低图像、高图像或参考图像中的相应未遮罩像素的颜色定义的颜色空间的立方体内。
- **黑色层**(通常是第 0 层)包含剩余的未屏蔽像素。

可以使用与源图像类型相关联的函数设置或获取**低图像**指针：

- BW8: `SetLowImageBW8GetLowImageBW8`
- BW16: `SetLowImageBW16GetLowImageBW16`
- C24: `SetLowImageC24GetLowImageC24`

可以使用与源图像类型相关联的函数设置或获取**高图像**指针：

- BW8: `SetHighImageBW8GetHighImageBW8`
- BW16 `SetHighImageBW16GetHighImageBW16`
- C24 `SetHighImageC24GetHighImageC24`

可以使用与源图像类型相关联的函数设置或获取**参考图像**的指针：

- BW8: `SetReferenceImageBW8`, `GetReferenceImageBW8`
- BW16:`SetReferenceImageBW16`, `GetReferenceImageBW16`
- C24:`SetReferenceImageC24` , `GetReferenceImageC24`

6. 有标图像

`ELabeledImageSegmenter` 适用于 BW8 和 BW16 灰度图像；它产生具有等于最大灰度值数的层数的编码图像：BW8 图像为 256，BW16 图像为 65536。层 n 包含具有等于 n 的灰度值的所有未遮罩像素。

默认情况下，所有图层都被编码。然而，可以使用 `SetMinLayer` 和 `SetMaxLayer` 函数将编码限制到单个范围的层次，这些函数分别返回索引范围的最低和最高值。

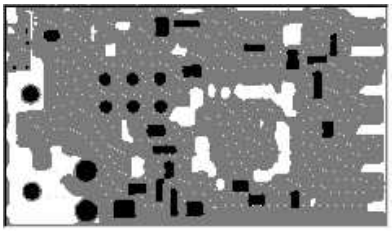
7. 二值图像

`EBinaryImageSegmenter` 适用于 BW1 二进制图像；它生成具有两层的编码图像：

- 黑色层(通常为索引 0)包含二进制值等于零的未遮罩像素。
- 白色层(通常为索引 1)包含剩余的未遮罩像素，即 二进制值等于 1 的未遮罩像素。

图像编码器

表示对象(`EObject`)的类派生于抽象类 `ECodedElement`)。



对象建立

选择要编码的层

分割方法(参见图像分割器)确定默认编码哪个层，并且不对其他图层进行像素编码。

函数 `GetMaxLayerIndex` 返回最高层索引值。它可用于所有分割器。

单独启用/禁用每个图层的图层编码

下表列出了每个层的 Set/Get 函数和默认的启用/禁用值。

两层分割器

层	设置 LayerEncoded 函数	获取 LayerEncoded 函数	默认值
黑色层	`SetBlackLayerEncoded`	`IsBlackLayerEncoded`	**FALSE**
白色层	`SetWhiteLayerEncoded`	`IsWhiteLayerEncoded`	**TRUE**

三层分割器

层	设置 LayerEncoded 函数名称	获取 LayerEncoded 函数名称	默认值
黑色层	`SetBlackLayerEncoded`	`IsBlackLayerEncoded`	**FALSE**
白色层	`SetWhiteLayerEncoded`	`IsWhiteLayerEncoded`	**FALSE**
中性层	`SetNeutralLayerEncoded`	`IsNeutralLayerEncoded`	**TRUE**

手动指定每层的层索引

下表列出了每个层的 Set/Get 函数和默认值。

两层分割器

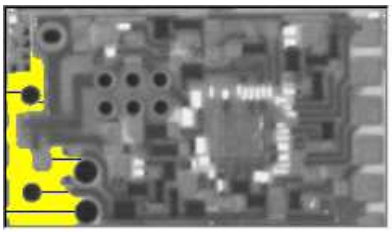
层	设置 LayerEncoded 函数名称	获取 LayerEncoded 函数名称	默认值
黑色层	`SetBlackLayerIndex`	`IsBlackLayerIndex`	**0**
白色层	`SetWhiteLayerIndex`	`IsWhiteLayerIndex`	**1**

三层分割器

层	设置 LayerEncoded 函数名称	获取 LayerEncoded 函数名称	默认值
黑色层	SetBlackLayerIndex	IsBlackLayerIndex	**0**
中性层	SetNeutralLayerIndex	IsNeutralLayerIndex	**1**
白色层	SetWhiteLayerIndex	IsWhiteLayerIndex	**2**

运行

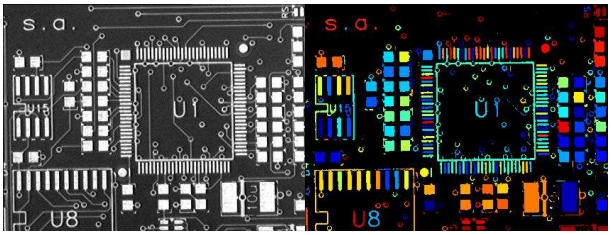
为了计算效率，对象被描述为运行列表。运行是共享相同属性(例如高于给定阈值)的相邻像素序列。这些运行由图像编码器合并在对象中。



具有五个增强运行的单个对象

EasyObject 可以在对象级别和运行级别工作，这样可以在严重的情况下加快处理速度。这对于计算对象上的自定义要素非常有用，然后列出属于给定对象的所有运行，如该运行级别的运行示例所示，输出图像中有彩色运行。

1. **声明一个新的** `ECodedImage2` 对象。
2. **声明** `EImageEncoder`，如果适用，请选择适当的分割器。设置分割器并选择合适的层进行编码。
3. **设置输出图像**。
4. **编码图像**。
5. **在输出图像中对运行进行着色**。通过构造一个循环，然后构造一个`RunsIterator`对象，迭代特定层的对象。该迭代器允许浏览所考虑对象的运行。一旦迭代器完成考虑对象的运行，内部循环将处理由输出图像中的该运行跨越的像素。
6. **选择特定图层**。



源图像(左)呈现白色层(右)

连接

像素可以沿着边缘或角落相互接触。在四个连接中，只有边缘触及的像素被认为是邻

居。角落触及的八个连接(默认)像素也被认为是邻居。

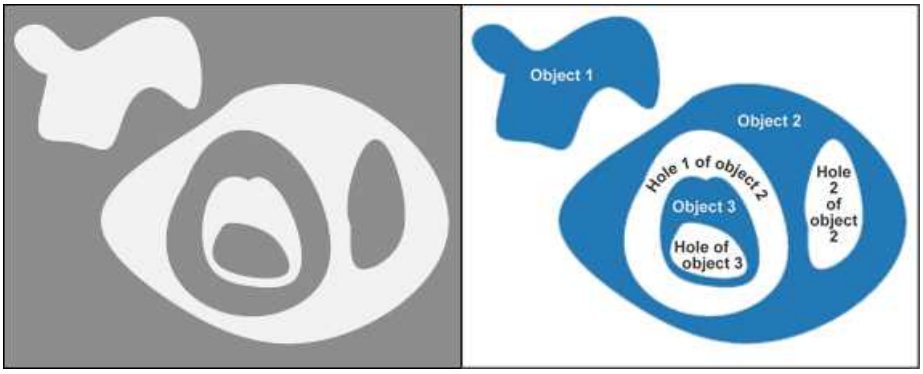
可以在连续模式对多个图像进行编码。

孔构建

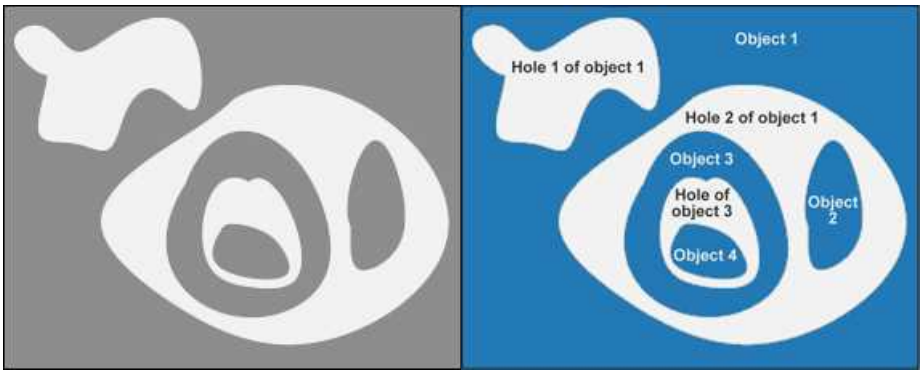
孔是由父对象完全包围的连接像素集合(取决于连接模式的 4 或 8 个像素)。

一个孔没有子级。孔内的对象被认为是单独的对象。

EObject 和 EHole 类均来自 ECodedElement，因此以相同的方式进行管理对象和洞，并共享相同的函数。



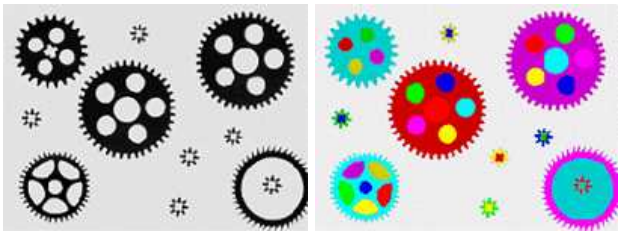
编码白色层(3 个对象和 3 个孔)



编码黑色层(4 个对象和 3 个孔)

如何对孔进行着色

1. **声明一个新的** `ECodedImage2` 对象。
2. **声明** `EImageEncoder`，如果适用，请选择并设置相应的分割器，然后选择适当的编码层。
3. **设置输出图像**。
4. **编码图像**。
5. **声明一个帮助函数来绘制运行效果。**辅助函数(另见“运行级别对象构造/工作”部分)使用例如给定的颜色在输出图像中绘制运行效果。对象和孔可以共享该函数。
6. **在输出图像中绘制物体及其孔。**有必要迭代所选层的对象。
   a. 帮助函数使用特定的颜色绘制每个对象的运行(`DrawRuns`)。
   b. 在当前对象上迭代这些孔，并且绘制孔运行效果。
   c. 全部在全局函数(`GetFadedColor`)中使用不同颜色计算绘制对象的每个孔，会返回一个取决于孔索引的颜色，例如，红色到绿色的渐变。



原始图像(左)构建对象和所有孔(右)

正常模式与连续模式

正常模式(默认)

在正常模式下，图像编码器不会跨越几个连续的图像跟踪斑点。EasyObject 适用于一个图像，而不会在内存中保留斑点。所有的斑点都作为对象返回。

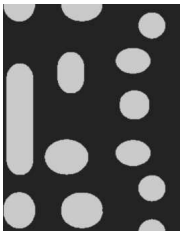
连续模式

在连续模式下，EasyObject 可以处理高度未知或无限大的图像(例如，来自行扫描相机)。图像被分成几个被馈送到图像编码器的块。可以检测到跨越几个连续图像块的对象。

图像编码器仅对不包含运行触及源图像最后一行的任何对象进行编码。触及图像下边缘的对象不会被写入编码图像，因为它们要在后续图像块中继续，但是它们会被保存在存储器中并且在后续图像分析时被处理。

假设大图像被划分成存储在阵列 `EImageBW8` chunk[x] 的数个块中。

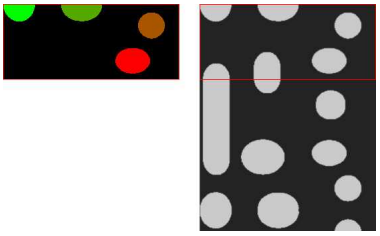
在这个例子中，我们生成了一系列彩色图像，展现了连续块编码的对象



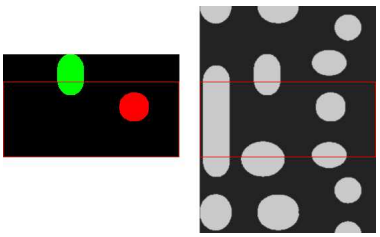
原图

三块图像

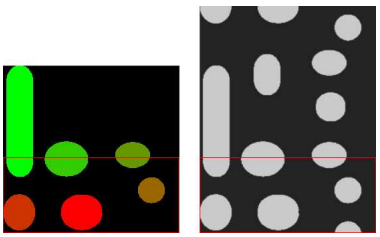
1. **绘制在编码图层中编码的对象。**该代码与“浏览运行”代码片段基本相同。唯一的区别是可以沿 Y 轴施加偏移。
2. **定义绘制图层对象的函数。**如果编码图像包含在先前图像中启动的对象：来自前一图像的该对象的运行被赋予负的 Y 坐标。
   零 Y 坐标是最近编码图像的第一行。约定将最低 Y 坐标分配给编码对象中的最早运行。
   `EImageEncoder . GetStartY` 方法获取最早运行的 Y 坐标。有必要定义显示编码图像层内容的函数。
   每个对象可以用不同的颜色显示(由 `GetFadedColor` 计算)。该函数紧跟着 `DrawRuns` 函数，但是通过考虑**GetStartY** 来适应连续模式。
3. **在**`EImageEncoder.SetContinuousModeEnabled`**属性中启用连续模式。**可以声明附加变量，例如存储连续的编码图像，或者保存输出图像。
4. **分析连续的块。**要对连续的块进行编码，使用`Encode(chunk[count], codedImage)`，然后使用`DrawLayer`。**注意：**变量计数跨越整数 0、1 和 2。当来自块的对象未完成时，它保存在图像编码器的内部存储器中。



当数据等于 0 时，在应用 DrawLayer 之后的 layerImage 内容。
正在考虑的大图像的块。
请注意，不对图像块左下角的两个对象进行编码，
因为它们触到了该块的边框。



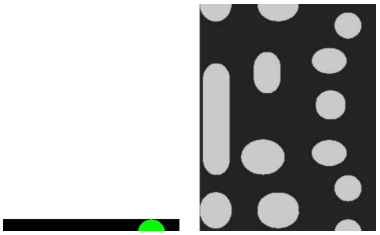
当 count 达到 1 时，这两个对象之一完成，
这导致对以下图像的编码。
此时还有两个其他对象尚未编码。
这是最后一个块的编码结果(count = 2) 。



来自前面的块的三个对象已经被关闭，并且被编码。

冲洗连续模式

在对三个图像块进行编码之后，仍然有一个对象要完成(在大图像的右下角) 。然而，由于没有更多的块，所以有必要明确地关闭该对象，并使用图像编码器的刷新来编码剩余的对象。图像编码器的内部存储器然后为空。



冲洗结果

选择和排序斑点

对象分割过程将任何斑点视为对象，包括显示为微小对象的噪点像素。您可以选择使用 EObjectSelection 类继续使用哪些斑点。

创建/修改选择

您可以使用EObjectSelection Add 和Remove方法以：

- 从选择中添加或删除单个对象、孔或整个图层。
- 根据指定的**功能**添加或删除对象或孔。(参见计算编码元素要素中的功能列表)。
- 根据具体的**位置**添加或删除对象或孔，或者它们是否位于指定的 ROI 矩形内。

这些动作可以在一个选择中随意进行级联和组合。

清除选择

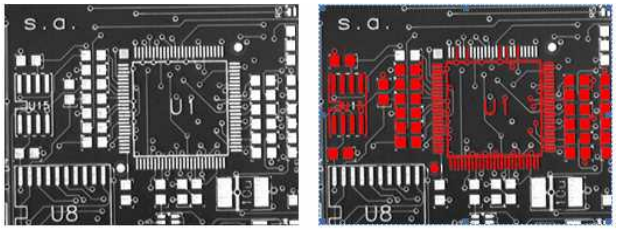
您可以使用 EObjectSelection .Clear来清除以前的选择。

排序选择

您可以根据任何特征对选择内容进行排序。

示例

在这个例子中，我们选择图像中间带的对象，其表面 > 100 像素。



源图像和对象的选择

1. **声明一个新的** ECodedImage2 对象。
2. **声明** EImageEncoder对象，如果适用，选择并设置适当的分割器，并选择适当的编码层。
3. **编码源图像**。
4. **创建对象的选择。**创建 EObjectSelection 类的实例，并将对象添加到该选择，例如通过 EObjectSelection .AddObjects。
5. **基于一个特征值一次删除对象。**可以通过调用 EObjectSelection .Remove方法之一来取消选择对象。

6. **使用**EObjectSelection`RemoveUsingFloatFeature.`，**根据其位置删除对象**。有关详细信息，请参阅“在运行级别工作”。
7. **使用**`EObjectSelection.Sort`对所选对象进行排序。
8. **访问所选对象**。

高级功能

可计算特征

以“**Get** ”为前缀的方法表示惰性求值：结果是在第一次调用方法时计算并缓存。

以“**Compute** ”为前缀的方法表示在每次调用时重新评估函数，从不缓存结果。

位置

限制
(上、下、左、右)
	`ECodedElement.GetTopLimit`
	`ECodedElement.GetBottomLimit`
	`ECodedElement.GetLeftLimit`
	`ECodedElement.GetRightLimit`

重心
(X 和 Y)
	`ECodedElement.GetGravityCenter`
	`ECodedElement.GetGravityCenterX`
	`ECodedElement.GetGravityCenterY`

重量重心 (X 和 Y)	`ECodedElement.ComputeWeightedGravityCenter`

重力中心和重量重心

重力中心返回编码元素重心的横坐标。

重量重心计算一个给定图像在编码元素上的重心。

范围

面积(像素数)	ECodedElement.Area
Feret 框 (中心 X 和 Y、高度、宽度，具有不同的取向角，为 22、45、68 度)	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

边界框 (中心 X 和 Y、高度、宽度)	ECodedElement.GetBoundingBox ECodedElement.GetBoundingBoxCenter ECodedElement.GetBoundingBoxCenterX ECodedElement.GetBoundingBoxCenterY ECodedElement.GetBoundingBoxHeight ECodedElement.GetBoundingBoxWidth
最小封闭矩形 (角度、中心点 X 和 Y 坐标、高度、宽度)	ECodedElement.MinimumEnclosingRectangle ECodedElement.MinimumEnclosingRectangleAngle ECodedElement.MinimumEnclosingRectangleCenter ECodedElement.MinimumEnclosingRectangleCenterX ECodedElement.MinimumEnclosingRectangleCenterY ECodedElement.MinimumEnclosingRectangleHeight ECodedElement.MinimumEnclosingRectangleWidth

Feret 框

Feret 框是个矩形，其中最小表面以指定的包含对象所有像素中心点的角度旋转。

- **边界框**是 0° 的 Feret 框。
- **最小封闭长方形**是具有所有可能角度的最小表面的 Feret 框。
- **FeretBox 矩形**的**宽度**是与 X 轴呈现最小角度的矩形边的长度。这不一定是最小的一边！
- Feret 框矩形的**高度**是矩形另一边的长度。

其他

对象轮廓的起始点 (X 和 Y)	ECodedElement.GetContour ECodedElement.GetContourX ECodedElement.GetContourY
最大运行	ECodedElement.GetLargestRun
运行数	ECodedElement.GetRunCount
对象数量 (索引)	ECodedElement.GetLayerIndex ECodedElement.GetElementIndex
像素灰度值 (平均值、偏差、方差)	ECodedElement.ComputePixelGrayAverage ECodedElement.ComputePixelGrayDeviation ECodedElement.ComputePixelGrayVariance
像素灰度值 (最小和最大)	ECodedElement.ComputePixelMax ECodedElement.ComputePixelMin

惯性椭圆

惯性椭圆离心率	ECodedElement.Eccentricity
矩	ECodedElement.GetCentralMoment ECodedElement.GetMoment ECodedElement.GetNormalizedCentralMoment
椭圆 (角度、高度、宽度)	ECodedElement.GetEllipseAngle ECodedElement.GetEllipseHeight ECodedElement.GetEllipseWidth
二阶几何矩 (Sigma：X, XX, XY, Y, YY)	ECodedElement.GetSigmaX ECodedElement.GetSigmaXX ECodedElement.GetSigmaXY ECodedElement.GetSigmaY ECodedElement.GetSigmaYY

注意：*可以通过使用轮廓方法跟踪对象轮廓并对像素进行计数来间接测量对象周长。*

从标准几何特征可以推导出其他特征。例如，对象伸长率计算为大小椭圆轴或最大高度最大宽度的比率。对象圆定义为平方的周长比除以四倍π,再乘以对象面积。

注意：*注意。公式(N =面积)：*

$$\sigma_X = I_X = \frac{1}{N}\sum(Y_i - \bar{Y})^2$$

$$\sigma_Y = I_Y = \frac{1}{N}\sum(X_i - \bar{X})^2$$

$$\sigma_{XX} = I_A = \frac{I_X + I_Y}{2} + \sqrt{\left(\frac{I_X + I_Y}{2}\right)^2 + I_X I_Y + I_{XY}^2}$$

$$\sigma_{XY} = I_{XY} = \frac{1}{N}\sum(X_i - \bar{X})(Y_i - \bar{Y})$$

$$\sigma_{YY} = I_B = \frac{I_X + I_Y}{2} - \sqrt{\left(\frac{I_X + I_Y}{2}\right)^2 + I_X I_Y + I_{XY}^2}$$

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

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

$$ANGLE = arccotan\left(\frac{I_x - I_a}{I_w}\right)$$

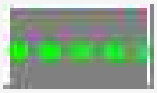
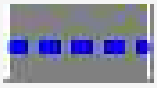
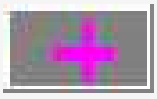
凸包

形状凸包是指完整包围一个目标的最小区域的凸多边形。凸包可用于表现目标足迹和观察凹形。假设凸包的顶点数是可变的，它们被存储在 `EPathVector` 容器中。

相应的函数是 `ECodedElement . ComputeConvexHull`。

图形表征

可以通过 `ECodedImage2 . Draw` 将对象绘制到源图像上。还可以通过 `ECodedImage2 . DrawFeature` 的方式绘制的图形表示以下特征。

目标	图形
边界框	
凸包	
椭圆	
Feret 框	
角度为 22°的 Feret 框	
角度为 45°的 Feret 框	
角度为 68°的 Feret 框	
重心	

最小封闭长方形	
加权重心	

协调系统和规则

坐标系

EasyObject 使用像素坐标系，其原点通常位于图像左上角像素的左上角。因此，像素中心坐标的小数部分为“.5”。该规则最适合于子像素坐标的表示。

角

因此，根据数学规则，这些角度现在反向计数：正角度使 X 轴在 Y 轴上。

评估特征

每个特征有一个属性，通过**枚举**无需访问该特征。

绘制编码元素

一旦图像被编码，可以通过抽象类 `ECodedElement` 和大量的方法来访问编码的元素(对象或孔)：

绘制**编码元素**

1. **声明一个新的** `ECodedImage2` 对象。
2. **声明** `EImageEncoder`对象，如果适用，选择并设置适当的分割器，并选择适当的编码层。
3. **创建输出图像**：如果所得特征图形必须着色,将(灰度)源图像逐个像素复制到(彩色)输出图像。
4. **编码源图像**。
5. **绘制图层**中每个对象的特征。
6. 读取结果，可以向下舍入。可以创建一个特定的图形来标记特征(例如，绘制重心的目标)。

要生成灵活的蒙板，请使用`ECodedElement.RenderMask`。

可有效地随机访问目标、孔洞及其特征(通过基于索引的方式)。

EasyObject 中的灵活蒙板

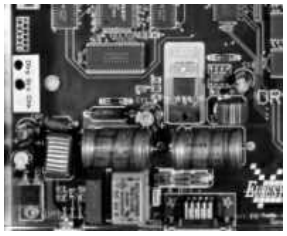
可以通过输出 BW8 图像的任何应用程序或使用 Open eVision 图像处理函数生成灵活的蒙板。
EasyObject 可以使用灵活的蒙板将斑点分析限制到图像的复杂或断开的形状区域。

如果相关对象与图像的其他区域具有相同的灰度级别，则可以使用灵活的蒙板和 `Encode` 函数定义“保持”和“忽略”区域。

灵活蒙板是与源图像具有相同高度和宽度的 BW8 图像。

- 灵活蒙板中的像素值为 0 遮罩对应的源图像像素，使其不出现在编码图像中。
- 灵活蒙板中的任何其他像素值会导致像素被编码。

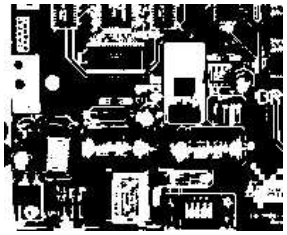
创建灵活蒙板的 EasyObject 函数



源图像

1) ECodedImage2 . RenderMask：从编码图像的层

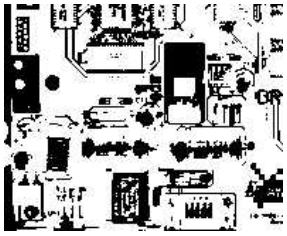
1. 为了编码和提取灵活蒙板，首先从源图像构造编码图像。
2. 选择一个分割方法(对于上方图像，默认方法`GrayscaleSingleThreshold`是合适的)。
3. 选择要编码的编码图像的层(即 使用最小残留阈值设置的白色和黑色层)。
4. 使用 `mask.SetSize(sourceImage.GetWidth(), sourceImage.GetHeight())` 使蒙板图像成为所需的大小。
5. 利用灵活的蒙板作为`ECodedImage2.RenderMask`的参数。



BW8 产生的图像可以用作灵活蒙板

2) ECodedElement . RenderMask：从斑点或洞

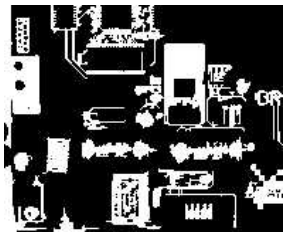
1. 选择相关编码元素。
2. 使用`ECodedElement.RenderMask`，创建一个从编码图像的所选编码元素中提取蒙板的循环。
3. 也可选择根据这些选择的编码元素中的每一个来计算特征值。



BW8 产生的图像可以用作灵活蒙板

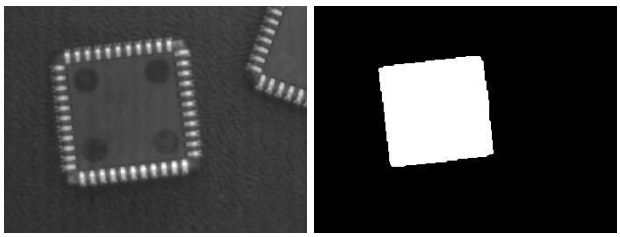
3) EObjectSelection .RenderMask：从选择的斑点

例如，`EObjectSelection . RenderMask` 可以丢弃由噪声导致的小型对象。



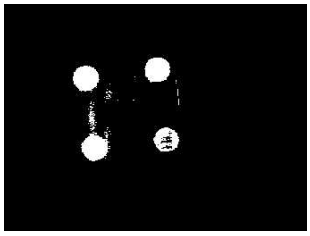
BW8 产生的图像可以用作灵活蒙板

示例：限制 EasyObject 编码的区域



查找四个圆形(左)灵活的蒙板可以隔离中央芯片(右)

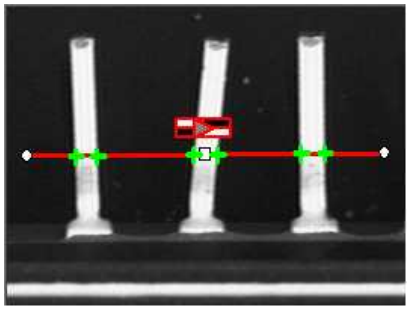
1. 声明一个新的 `ECodedImage2` 对象。
2. 设置变量：首先声明源图像和灵活蒙板，然后加载它们。
3. 声明 `EImageEncoder` 对象，如果适用，请选择适当的分割器。设置分割器并选择适当的编码层。
4. 编码源图像。然后，只需将灵活蒙板中的区域编码一层就可以很简单。
   我们看到，使用灰度单阈值分割器在黑色层中正确分割了圆圈：



5. 选择编码图像的所有对象。
6. 通过过滤掉太小的对象来选择相关对象。
7. 通过迭代所选对象显示所选特征来显示斑点特征。

3.2. EasyGauge - 向下测量至子像素

EasyGauge 库控制维度。它准确地确定各部分的位置、方向、曲率和尺寸。它可以以图形方式进行交互，以便放置和分类量规，将它们组合在分组的层次结构中，并使用其所有参数存储和检索它们。



工作流

量规模型可以编程或在图形编辑器中构建，然后在最终应用程序中“播放”。
选择与您的模型复杂性相匹配的工作流程以及所需的准确性：未校准、校准或分组。

未校准测量：用于一个简单的模型

EasyGauge的基本用法很简单。

1. 创建一个对应于所需测量的量规对象。
2. 更改默认值不合适的参数。
3. 调用所需的测量函数。
4. 读取生成的位置参数。

未校准测量易于实现，但有几个缺点：

- 测量以像素为单位执行，而不是毫米。
- 测量模型不可移植：如果观察条件发生变化，量规位置和尺寸必须重新处理。

- 光学失真或透视造成不准确的测量。

校准测量：用于一个或两个简单的测量点

校准测量更准确，并且独立于观察条件测量被检查的部分。
所有测量都在校准单元中进行，任何失真都会被隐含补偿。请参阅校准，了解如何掌握视场校准。

1. 创建校准器对象。
2. 将其放在被检查的场景上。
3. 调整校准参数。
4. 附上量规。

复杂测量

量规可以分组(见量规操作流程)并附加到另一个项目：

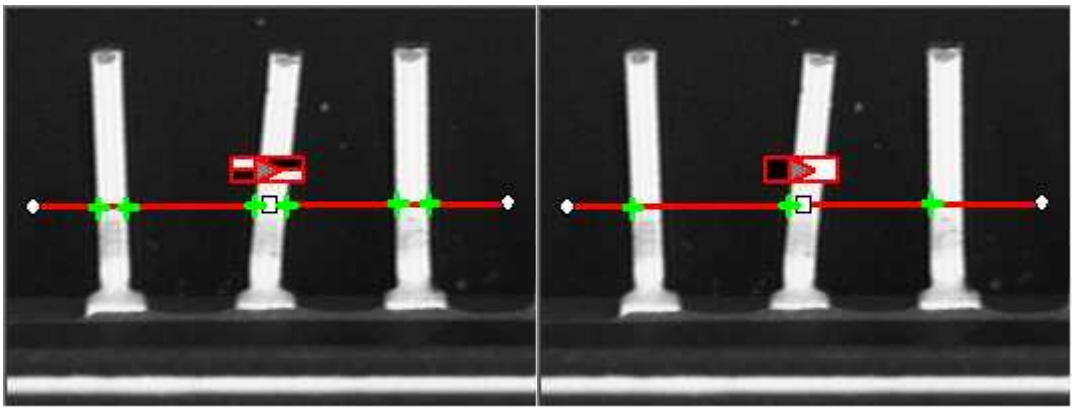
- **将量规连接到**EFrameShape对象上，使用框架(平移和/或旋转)移动量规，应用程序必须调整框架位置以跟踪被检查部件。
- **将量规连接到另一个量规**，根据支撑量规的测量位置移动量规。例如，如果量规连接到正在检测零件轮廓的普通矩形量规，则当矩形轮廓拟合时，所有量规会自动跟踪部件。

如果使用多个测量站点，您可以在单个文件中保存完整的模型、校准模式、系数和附加的量规。

量规定义

点量规

您可以沿着跨越一个或多个对象边缘的线段探针选择最相关的转换点。可以激活横向和纵向滤波以降噪。



点位置。基于对比的选择

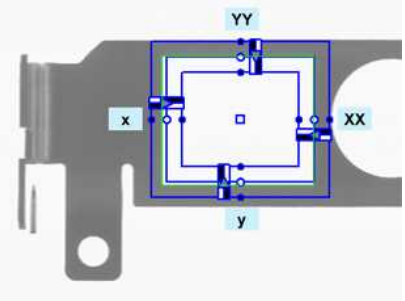
矩形量规

矩形量规的放置由其标称位置(由其中心的坐标给定)，其标称尺寸及其旋转角度。

矩形的每一边可以有自己的转换检测参数，并且可以使用 ActiveEdges 属性设置为活动或非活动。当一边活跃时：

- 设置参数的值仅适用于当前活动的边。
- 只有当该属性的值对于所有活动边相同时，获取参数的值才会产生结果。
- 只有活动边用于测量和模型拟合。

这些规则允许为不同的边设置不同的参数，并且测量平行边或角点而不是整个矩形。四边分别用字母“x”、“y”、“XX”和“YY”表示。



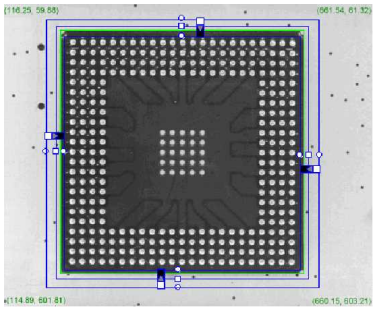
矩形量规各边的命名约定

使用

定义和定位量规，然后使用 Measure 来适应线条。
要获取矩形属性，请将 ActualShape 设置为 **TRUE** 以返回拟合线(**TRUE** 值)(默认值为 **FALSE**)。

或者，MeasuredRectangle 将结果提供为 ERectangle 对象。

例如，您可以使用矩形拟合量规精确定位矩形的四个角(地标)。



找到矩形的角落

楔形量规

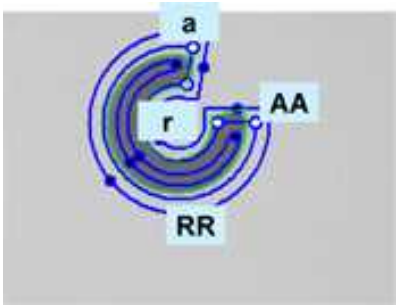
楔形量规的放置由其标称位置(由其中心的坐标给出)、其标称内和外半径(内和外径)、其宽度(半径之间的差)、其扩展的角位置及其角振幅确定。

设定的成员可以区分完整的环、扇区和圆盘。

楔形的每一边可以有自己的转换检测参数，并且可以使用 ActiveEdges 属性设置为活动或非活动。当一边活动时，这意味着：

- 设置参数的值仅适用于当前活动的边；
- 只有当该属性的值对于所有活动边相同时，获取参数的值才会产生结果；
- 只有活动边用于测量和模型拟合。

所以不同的边可以有不同的参数，你可以测量平行弧或斜边或角点，而不是整个楔形。四边分别用字母“a”、“r”、“AA”和“RR”表示。



楔形量规各边的命名约定

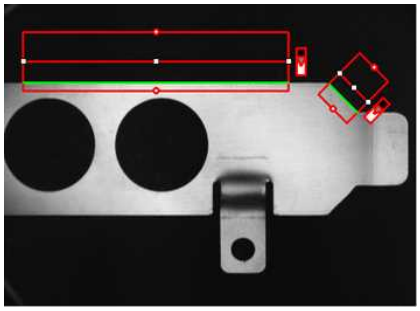
使用

定义和定位量规，然后使用 Measure 来适应线条。
要获得楔形属性，请将 ActualShape 属性设置为 **TRUE** 以返回拟合线(TRUE 值)(而不是标称行位置 **FALSE** 值，默认值)。

或者，MeasuredWedge 将结果作为 EWedge 对象提供。

线规

线规的放置由其相对于 X 轴的中心坐标、长度和角度定义。要限制线斜率值，请设置 Angle 和 KnownAngle。



线配

使用

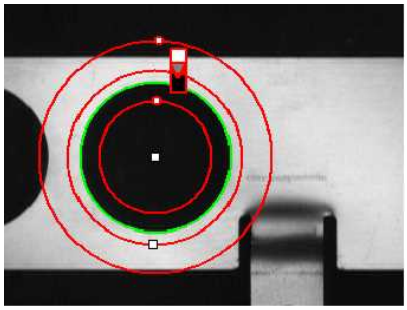
定义和定位量规，然后使用 Measure 来适应线条。要获取行属性，请将 ActualShape 属性设置为 **TRUE** 以返回拟合行(**TRUE** 值)(而不是名义行位置 **FALSE** 值，默认值)。

或者，MeasuredLine 将结果提供为 ELine 对象。

圆形量规

圆形量规的放置由其标称位置(由其中心的坐标给出)、其标称直径(或半径)、其扩展角位置及其角度振幅确定。

设定成员可以区分整圆和圆弧(必须指定圆弧振幅)。



圆拟合

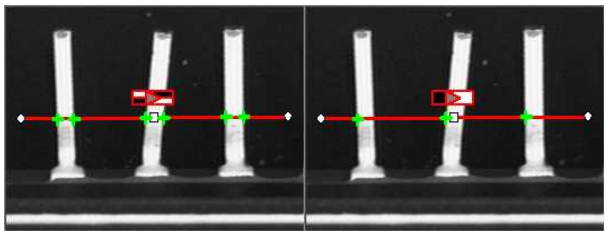
使用

一旦定义并定位量规，使用测量触发圆拟合操作。要获得测量结果，请将 ActualShape 模式设置为 **TRUE** 。ActualShape 模式确定询问是返回拟合圆(**TRUE** 值)还是名义圆位置(**FALSE** 值，默认值)。然后通过圆属性检索所请求的信息。

或者，MeasuredCircle 将结果提供为 ECircle 对象。

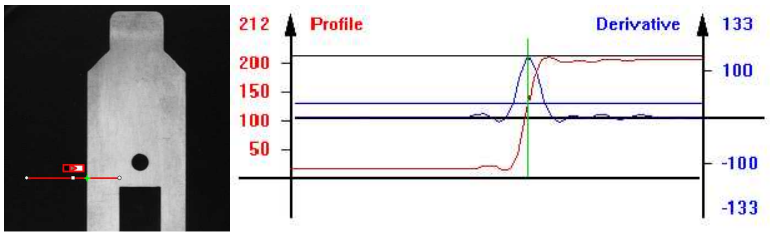
使用峰值分析查找转换点

查找跨越一个或多个对象边缘的线段探针的所有转换点的位置，并允许选择最相关的。可以激活横向和纵向滤波以降噪。



点位置。基于对比的选择

点位置原理



点位置原理(左)和 S 形曲线及其衍生(右)

在从图像提取的线性轮廓上，边缘显示为从深到浅的转换(反之亦然)。沿着量规绘制像素值时，该转换显示为 S 形曲线。该曲线的一阶导数在转折点附近出现峰值。对比度越好，转换越清晰，峰值越高。

EasyGauge 提取沿曲线(红色曲线)的像素值，然后使用峰值分析来确定转换位置。导数曲线和水平用户定义阈值水平之间的峰值区域[1]计算转换位置。

- 只有在转换被至少 2 个像素宽的几乎均匀的区域包围时，子像素精度才可能实现。
- BWB[2]转换的轮廓曲线增加，峰值呈正值。否则，曲线减小，峰值负向延伸。
- 您不能正常使用默认阈值(20)检测峰值，因为 BWB 或 WBW 转换基于沿 EPointGauge(或采样路径)的灰度分布图的峰值分析，而不是其一阶导数。

EPointGauge包含所有点测量参数，其默认值可检测合适的对比度边缘。

EPointGauge 参数

中心：标称点位置(测量前后通常不同)。
公差：公差值和量规方向。
TransitionType、TransitionChoice、TransitionIndex：峰值选择策略。
阈值：抗噪声。
MinAmplitude、MinArea：峰值强度。
厚度、平滑：局部滤波器宽度。
RectangularSamplingArea将采样区域(默认为矩形)设置为横向过滤模式。
测量：测量对象。
- 在单次转换模式下，当找到适当的点时，Valid返回 True。要获得测量结果，请将ActualShape设置为 True，以便Center返回定位的点。(假默认值返回标称点位置)。
- 在多重转换模式下，NumMeasuredPoints返回找到的点数，
GetMeasuredPoint返回一个包含位置信息的EPoint对象。
必须传递 0 和 GetNumMeasuredPoints-1 之间的整数索引。
GetMeasuredPeak：返回EPeak包含峰值的面积和幅度，以及沿探测段(开始、长度和中心值)的定界坐标。

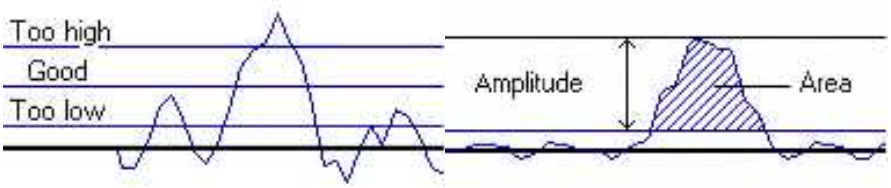
选择峰值以提高边缘精度

阈值水平非常重要：

[1]中的所有像素值用于
[2]黑色/白色/黑色

- 太高可能导致遗漏明显的峰值，并且像素值不足以达到良好的精度。
- 太低，由于噪声可能导致假峰值。

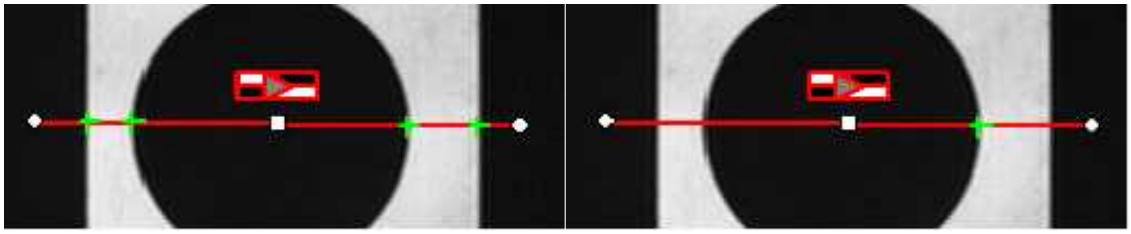
为了解决这个困难，EasyGauge 峰值选择机制可以抑制低对比度或虚假边缘：通过峰值*幅度*和*面积*测量转换强度。每个边缘测量确定峰值幅度和面积。如果任一值低于最小幅度或最小面积，则忽略峰值，并且在该位置不设置任何点。



阈值水平选择(左)和峰值幅度和面积(右)

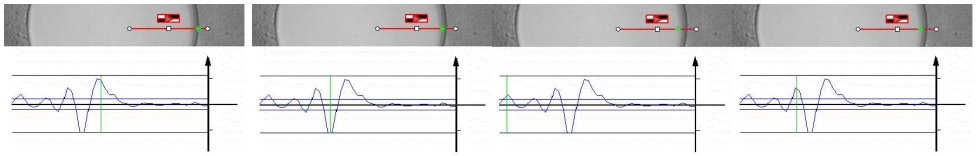
多次与单一转换

EasyGauge 可以单次测量多个边缘点，然后在*多重转换模式*中检索所有结果。



多次转换(左)与单次转换(右)

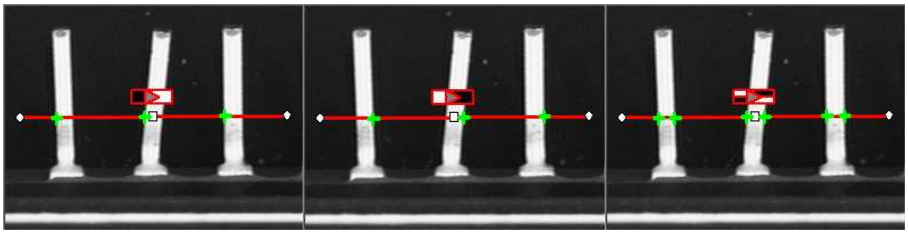
您可以根据 4 个条件选择最相关的转换：最高峰，最大面积的峰值，最接近测量中心的峰，或从量规的一个尖端开始遇到的第 N 个峰。



最佳面积(第一幅图像)和最佳幅度选择(第二幅图像)，最接近(第三幅图像)和从开始起的第三个图像(第四幅图像)

正或负峰选择

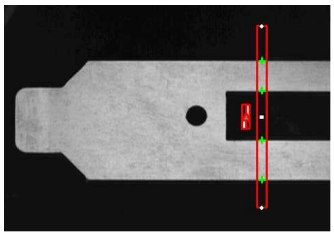
也可以通过选择转换极性对峰值选择进行细化：白色至黑色或黑色至白色(即 正或负峰)或无极性。



黑色至白色，白色至黑色或无极性

预滤波

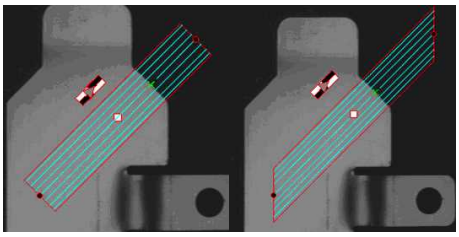
在本地对图像进行预滤波可以降低噪声影响。
横向(纵向)滤波对采样图像时的几条平行线进行平均。
纵向(横向)均匀滤波也可应用于所得轮廓曲线。



过滤厚度计

横向过滤

横向过滤将平行线段放置在平行四边形或矩形(默认)中。该行为可以切换。
如果角度接近 0°或 90°，或厚度小于5，则平行四边形模式比矩形快。如厚度 = 1，则两种模式之间不存在差异。
厚度确定并行数。
采样区是包含所有并行行的最小区域段。



矩形采样区(左)和平行四边形采样区(右)

点探针位置

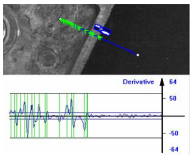
点量规的预期**标称**位置由**中心**、方位**角度**相对于X轴和长度**公差**指定，点位置可以变化。

结果是定位点的坐标(**实际**位置)和转换强度(幅度和面积)。
低值表示弱边缘，可能对应于不可靠或不准确的测量。

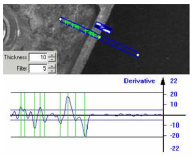
为不清晰边缘调整点测量参数

EasyGauge 的默认参数和工作模式适合清晰的边缘。更复杂的情况可能需要参数调整。

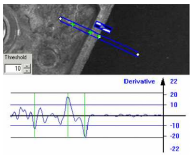
1. 设置量规点位置和公差。
根据样本图像或坐标考虑很容易决定中心位置和方向。公差取决于边缘位置的变化。较大的公差增加了触及边缘的可能性，但它可能是假边缘或无关特征。



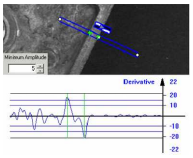
2. 决定是否需要降噪。将量规放置在所需位置上，并观察轮廓曲线及其导数(使用过滤参数，同时查看绘制曲线)。曲线规则性给出了灰度值的扩展指示。
当设置这些系数时，灰度级别的配置文件将不再变化。



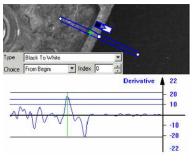
3. 将阈值设置为足够低，以使有效部分的峰值覆盖足够的像素(以获得更好的子像素精度)，但不要低于环境图像噪声。



4. 使用峰值幅度和面积列表去除弱边或虚边。绘制这些值以及良好和无关的峰值可以帮助找到合适的峰值抑制极限。

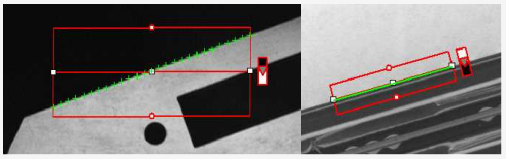
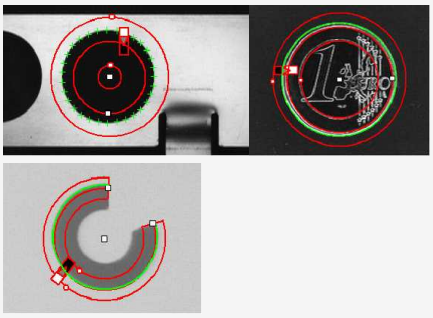


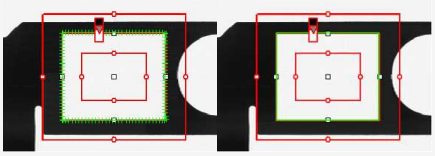
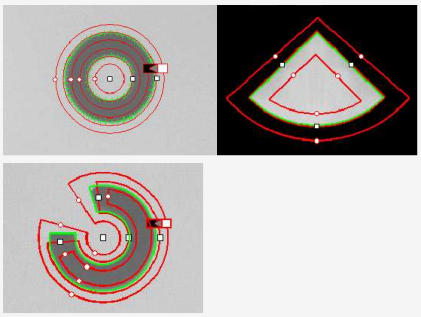
5. 选择是否需要所有转换点或是最相关的。如果都需要，可以一个接一个地查询。否则，应根据强度、顺序或转换极性(黑色至白色和/或相反)选择点选择策略。



使用几何模型查找形状

ELINEGauge、ECircleGauge、ERectangleGauge或EWedgeGauge预定义的几何模型可以适合对象的边缘。必须定义目标边缘，并以规则间隔的点测量计在其上采样点。可以应用最小二乘的拟合模型。

行:测量直边的位置和方向。	
圆圈:测量圆圈或圆弧的位置和曲率。	

矩形：测量矩形的位置、方向和大小。	
楔形：测量环/圆盘扇区/曲线矩形的位置、方向和大小。	

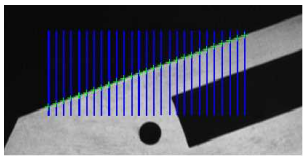
所有量规类型具有如下常见功能：

点抽样

沿着边缘放置点量规，并且在规则间隔的点进行点测量，可以在矩形和楔形量规中对每侧分别进行调节。所有点测量参数和操作模式都可用。

`SamplingStep`沿着模型的点位置设置量规的间距。

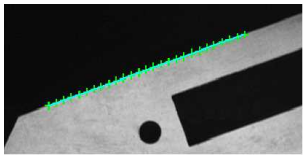
`NumSamples`返回模型拟合操作期间采样的点数。



采样路径和采样点

模型拟合

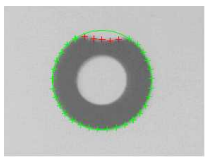
调整模型以最小化误差残余，并提供最佳边缘参数估计。矩形和楔形具有平行度和同心度约束。图像显示采样点和拟合线。



异常排斥

模型拟合后，某些点与拟合模型相距太远，可能会损害定位精度。EasyGauge 可以使用`FilteringThreshold`属性将其标记为异常值并忽略。

使用`NumFilteringPasses`可以重复数次异常值消除过程。
模型拟合操作后剩余的有效采样点数保存在`NumValidSamples`中。
这些点与拟合模型的平均距离由`AverageDistance`返回。



量规操作：绘图、拖动、绘制、组合

EasyGauge 提供了以图形方式与量规相互作用的方法来放置和缩放尺寸，将它们组合为分组项目的层次结构，并使用模型文件处理它们以及所有工作参数的存储/检索。

绘图

绘图给出了量规的图形表示。在与所需窗口相关联的设备上下文中，用当前的笔完成绘

制。根据操作，可能会显示手柄。

拖动

操作员可以在图像上交互拖动量规。有几个拖动手柄可用。

- HitTest确定鼠标光标在句柄上方。如果出现鼠标光标，光标的形状应该改变以进行反馈，并可以进行拖动。
- 拖动相应地移动手柄和相应的量规。

绘制

EasyGauge 可以沿着采样路径绘制灰度值和/或其导数 - 对参数调整很有用。
调用Measure后，点量规可绘制。
在MeasureSample调用**0**和**GetNumSamples-1**(包括)之间的索引参数后，模型拟合量规可以绘制。
要查看相应的采样路径，请使用Draw方法，模式为**EDrawingMode_SampledPath**。

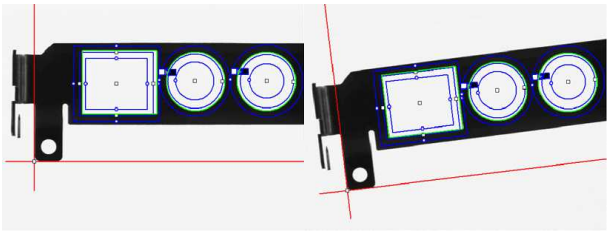
组合

量规可以分组(它们的相对位置保持固定)，以形成可以在计算测量之前移动(平移和旋转)跟踪所检查项目/探针的移动的专用工具。

Attach将量规与母量规或 EFrameShape 对象相关联。

NumDaughters、GetDaughter或Mother检索与附属的子级或父级相关的信息。

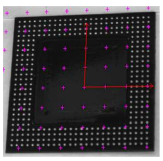
Detach、DetachDaughters将量规或子级从父级分开。



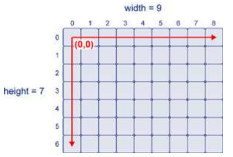
校准和转换

视场校准

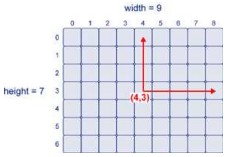
校准建立了真实点坐标与图像像素之间的关系。简单的校准模型计算速度更快，可重复的部件位置更容易定位。



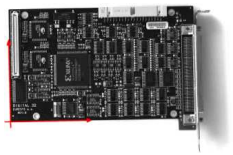
原始传感器坐标系从左上方开始，向右和向下延伸。
横坐标的范围是**0**到**width-1**，纵坐标的范围是**0**到**height-1**，其中整数坐标值对应于像素中心。



中心传感器坐标系从原始系统的中心开始([width-1]/2，[height-1]/2)，并向右和向上延伸。



在连接到参考平面的 2D 参考框架中定义真实现实世界的 3D 坐标。轴的原点和方向通常与检查部件的主要特征对齐。



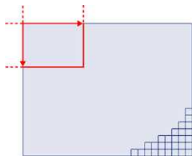
在现实世界到传感器变换之前

在从现实世界转换为传感器坐标之前，应该消除变形源：

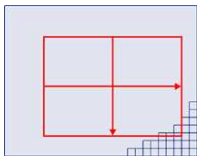
- 调整扫描频率或扫描速度以避免非方形像素。
- 调整光学对准以尽量减少透视效果。视场应与传感器平面平行。
- 使用长焦距和高品质镜头，以尽量减少光学失真。
- 根据镜头放大倍率、观察距离和对焦度，使用适当的比例因子。
- 通过安全夹具和部件移动/采集触发同步，最小化偏斜和平移效应。

现实世界到传感器变换的影响

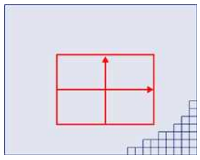
- **无校准**。现实世界和传感器坐标是相同的。



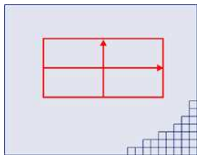
- **转换校准**：可以移动坐标原点。现实世界坐标对应于像素单位。



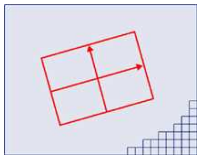
- **各向同性缩放**（方形像素）。比例因子将像素值转换为物理测量。



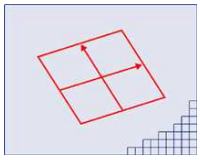
- **各向异性标度**（非方形像素）。使用像素长宽比（X/Y）在[-4/3，-3/4]（或[3/4，4/3]）范围内的两个比例因子。像素始终显示为正方形，因此图像呈现拉伸状态。



- **缩放和偏斜**（方形像素）。使用平移、旋转和缩放，实际轴与旋转检查部分对齐。

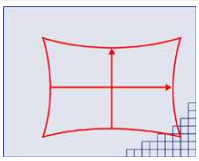


- **缩放和倾斜**（非方形像素）。扭曲是显而易见的。当相机扫描速度与像素间距不匹配时发生。

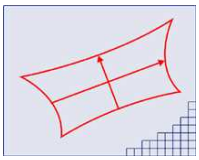


- **透视失真**导致更远的物体看起来更小；线条保持直线，但不保留角度。

- **光学畸变**造成矩形缓冲或筒状外观。



- **组合失真**导致从现实现实世界到传感器空间的复杂的，非线性的变换。



使用 EWorldShape 进行校准

如果光学设置被修改，EWorldShape 对象可以校准整个视场(在给定的成像条件下，利用固定的摄像机放置和镜头放大)。
EWorldShape计算适当的校准系数并转换与之相关的量规。
它可以设置现实世界到传感器的变换参数，从坐标系到坐标系进行转换，确定未知的校准参数，并保存给定变换的参数以供以后重用。

校准后EWorldShape可以使用SensorToWorld和WorldToSensor对任意点执行坐标变换以：

- 测量非正方形像素和旋转坐标轴。
- 校正视角和光学失真，确保没有性能损失。

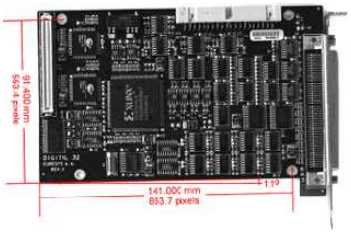
有几种方法来获得校准系数：

估计(如果不需要失真校正并且精度要求低，则可行)

要估计校准系数，可以定位视场的极限，并将图像分辨率除以视场大小，或者使用以下步骤：

1. 拍摄要检查的部分或校准目标(例如矩形)的照片。
2. 找到特征点，例如图像中的角(通过眼睛)，并以像素为单位确定其坐标--假设为(i，j)。
3. 使用欧氏距离公式推导校准系数：$C = \frac{\sqrt{(i_1 - i_0)^2 + (j_1 - j_0)^2}}{D}$
   其中C是校准系数，单位为像素，D为单位之间的对应点之间的现实世界距离。
4. 对于非方形像素，对水平和垂直点对重复该操作。

要估计歪斜角度，请将该公式应用于现实世界系统 X 轴上的两个点：$\theta = \arctan \frac{j_1 - j_0}{i_1 - i_0}$



估计比例因子和偏斜角

当校准系数可用时，请使用SetSensor调整它们并设置校准模式，或使用以下方法单独设置：SetSensorSize、SetFieldSize、SetResolution、SetCenter、SetAngle。

将一组参考点(地标)传递给校准函数

找到至少 4 个地标，并在传感器(使用图像处理)和现实世界坐标系(实际测量)中获得坐标。更多地标提供更准确的校准。

所得到的像素长宽比(X 分辨率/Y 分辨率)必须在[-4/3，-3/4](或[3/4，4/3])的范围内。

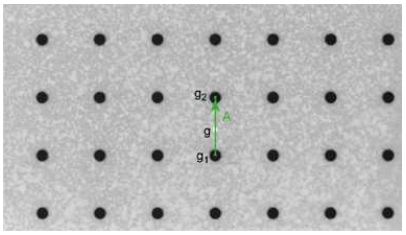
分析校准目标

可以自动分析校准目标，以获得适当的地标集。只要提供适当的程序来提取所需的地标点坐标，这是实现自动校准的简单方法。

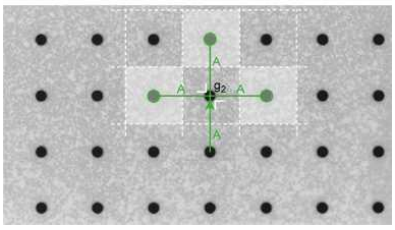
Open eVision 依赖于使用一个特定的对象，包含一个对称点(任意形状)矩形网格，网格上没有其他对象。

基于点网格的校准示例

1. 抓住校准目标的图像，使其覆盖整个视场(或将视图的图像限制在只有点可见的 ROI)。
2. 应用斑点分析来提取点的中心坐标，可以通过 EasyObject 完成。
3. 将所有检测到的点传递给 AddPoint(仅传感器坐标)。
4. 调用 RebuildGrid 重建网格，以使用计算每个点的现实世界坐标的迭代算法校准视场。
   a. 选择最接近网格点重心(g)的网格点(g_1和g_2)以形成长度为 A 的第一个参考定向段。



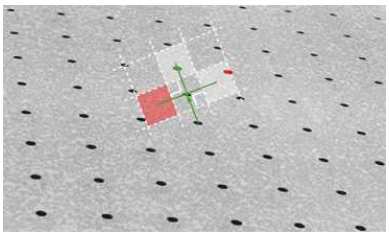
   b. 从参考段(g_2)的末端开始，该算法在垂直方向上确定 3 个公差区域(图中的白色正方形)。公差区域在距离(g_2)A(参考段的长度)的中心位置。它们是正方形，边长为 A。
   该算法在三个公差区域的每一个中都搜索一个相邻点。
   如果每个公差区域包含一个相邻点，则网格将被正确校准。



   c. 三个垂直段是下一次迭代搜索的参考。算法返回步骤 2。
5. 调用 Calibrate 这一地标方法。

如果网格出现过多的失真，不能按预期进行网格重建。可能会发生以下错误：

1. 公差区域不包含相邻点(图中的红色方块)。
2. 公差区域包含多个相邻点。
3. 公差区域中的点不正确。例如，该点可能是对角线连接的(图中的红点)。



高级功能

可以使用以下参数调整视场校准模型：

传感器宽度和高度

传感器宽度和传感器高度给出了以像素(始终为整数)计算的逻辑图像大小。

视场宽度和高度

视场(f-o-v)宽度和高度以长度单位给出实际图像大小，即 对应于现实世界空间中的图像边缘的矩形大小。这些值通过以下等式与像素分辨率相关：

```
f-o-v 宽度 = 像素宽度 * 传感器宽度
f-o-v 高度 =像素高度 * 传感器高度
```

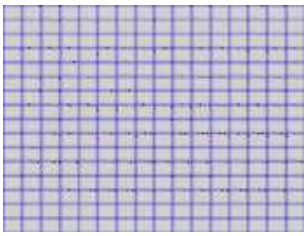
或

```
传感器宽度 = f-o-v 宽度 * 水平分辨率
传感器高度 = f-o-v 高度 * 垂直分辨率
```

默认情况下，未指定像素高度，像素假设为正方形(像素宽度 = 像素高度)。

比例

比率



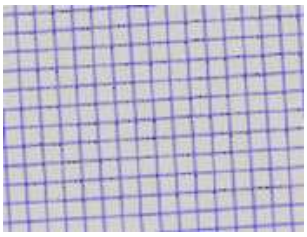
各向异性纵横比

中心横坐标和纵坐标

中心横坐标(x)和纵坐标(y)表示图像原点(现实世界坐标(0,0))。默认是图像中心。

倾斜角

倾斜角是由真实现实世界参考框架(X轴)和图像边缘(水平)形成的角度。默认是没有偏移。

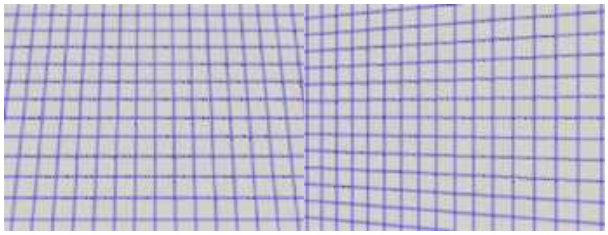


倾斜角

注意： *当像素不是正方形时，`EWorldShape`对象可以转换现实世界和传感器空间之间的角度。*

X 和 Y 倾斜角度

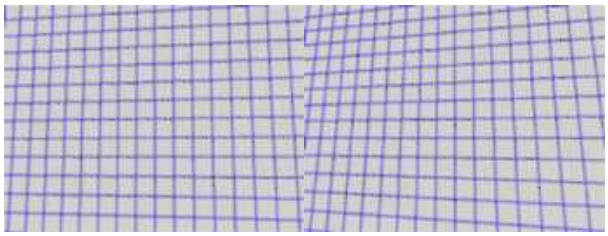
X和Y倾斜角描述了观察平面方向。它们对应于使 Z 轴平行于光轴的 X 轴和 Y 轴周围所需的旋转。



倾斜 X 和倾斜 Y 角度

透视力

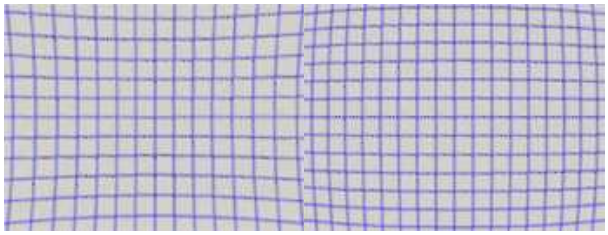
透视力给出了透视效果的相对度量。焦距越短，值越大。



强和弱透视

变形强度

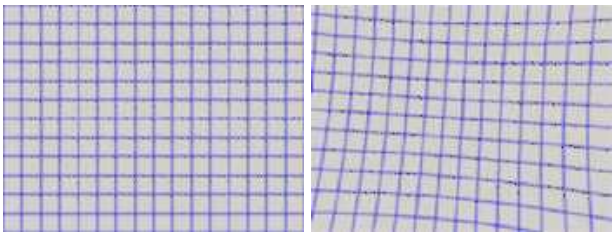
变形强度和`GetDistortionStrength2`给出了图像角中的径向失真的相对度量，即 图像对角线长度有无失真的比例。



正负变形

可以通过CalibrationMode访问校准模式，表示为选项的组合。

校准系数的影响



无校准系数：所有系数组合。

反变形图像

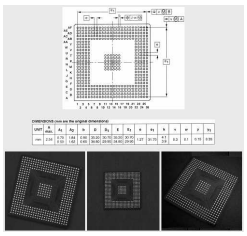
EWorldShape对象管理视场校准上下文。这样的对象能够表示现实世界坐标(物理单位)和传感器坐标(像素)之间的关系，并且说明图像形成过程中固有的变形。

图像校准是定量测量应用中的重要过程。它确定了被检查项目中图像中的点位置(像素索引)与现实世界中的那些点的实际位置之间的关系。

可以通过提供校准模型的显式校准参数或一组已知点(地标)或校准目标来设置校准。

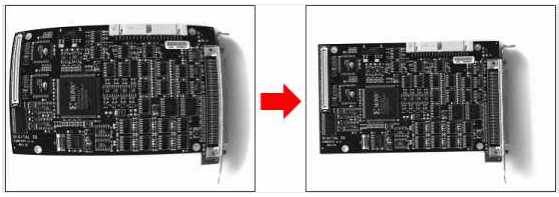
校准的目标是双重的：

- 为了在观看条件(视场中部件放置，透镜放大率，传感器分辨率，...)中获得独立性，让您使用绝对测量方式描述所有检测项目。



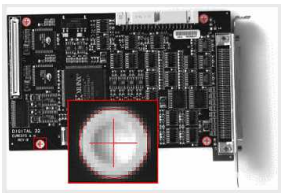
单一模型与多重观察条件

- 纠正与成像过程相关的一些失真(透视效果，光学像差，...)。



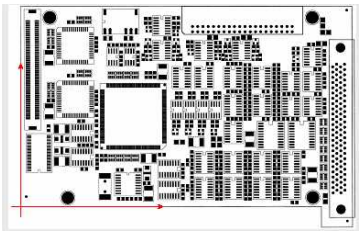
去除图像失真

图像中的像素索引通常是整数，但是当使用子像素方法时，可能会发生分数值。它们通常通过处理图像和定位已知特征点来获得。这些值称为传感器坐标。



传感器空间中的特征点

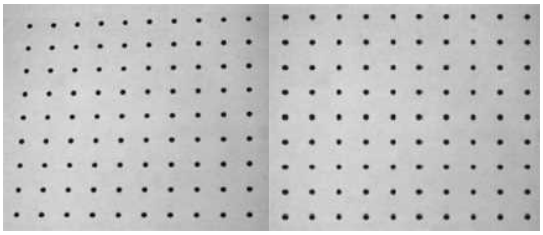
现实世界坐标描述检查项目上的点的位置以适当的长度测量单位表示。
现实世界坐标是实际尺寸，通常从设计图纸或机械测量中收集。
它们需要定义参考框架。



现实世界空间参考框架

Unwarp

使用Unwarp、SetupUnwarp**和UnwarpAfterSetup重新对图像进行扭曲校正。**
在反变形之前使用查找表可能会加速进程。

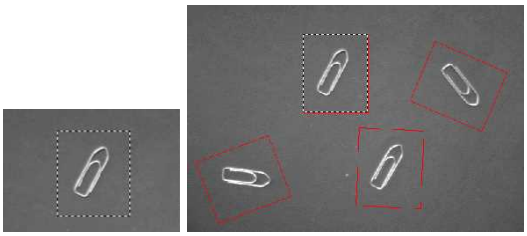


扭曲与反变形图像

3.3. EasyMatch——匹配区域图案

EasyMatch 学习一个图案，并找到完全匹配：

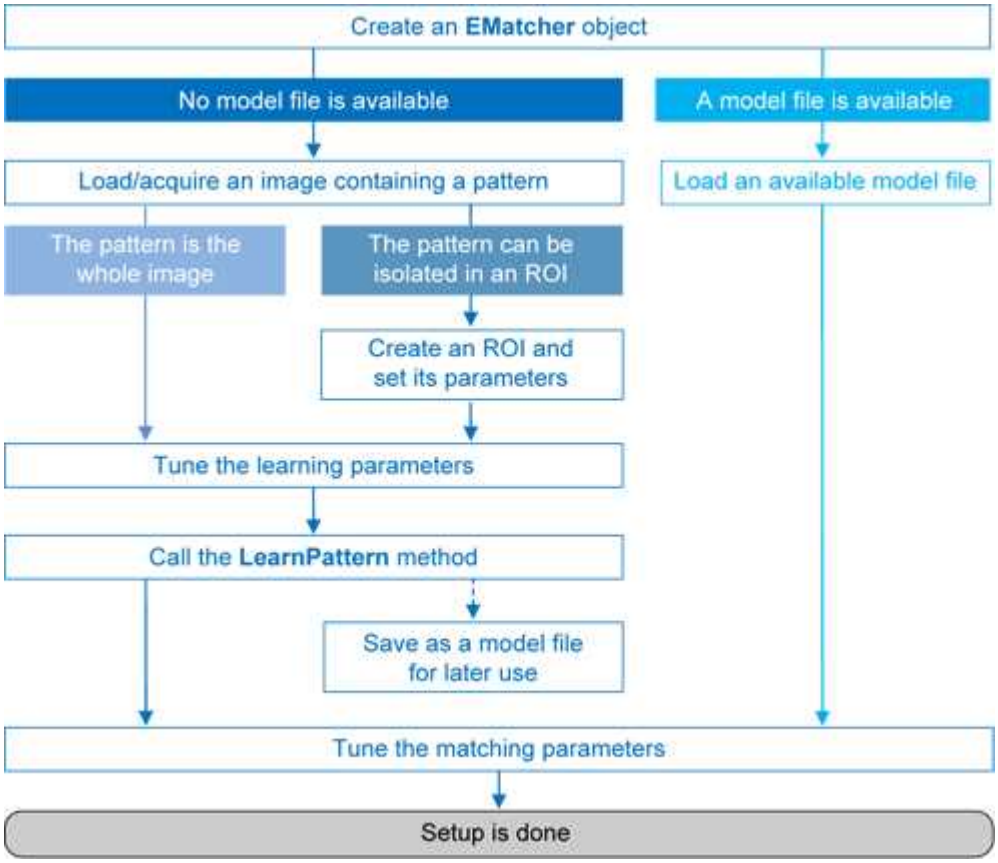
1. 通过定义包含要匹配的对象的 ROI 来学习图案。
   该 ROI 是从包含对象的几个图像迭代学习后创建的。
2. 调整参数以确保可靠地找到图案。
3. 现在可以搜索图像的一个或多个出现的图案，可以进行转换、旋转或缩放。



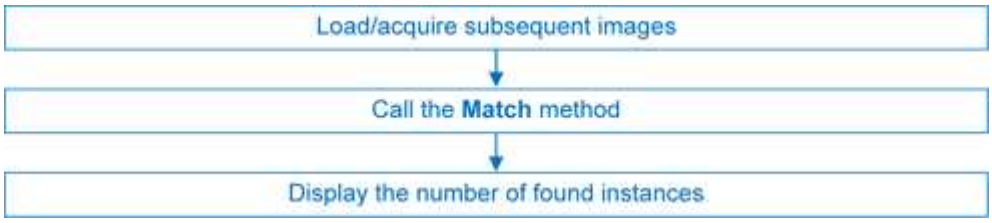
学习和匹配图案

工作流

学习工作流程



匹配工作流程



学习过程

选择包含要搜索的模式/ROI 的图像，并调用LearnPattern。

所得到的图案可以保存为模型供以后使用。您可以重复此过程来搜索并保存多个图案。

最佳图案特点

- **可重复**，您需要知道是否可以转换或旋转或缩放。
- **代表要定位的对象。**

  应该：

  - 保持相同的外观，无论照明条件如何。

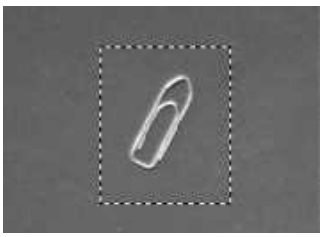
- 保持在相对于该部分的固定位置。
- 坚硬而不变形。

- **在大小比例上表现出良好的对比度。**它应该从远处清晰可见，在缩小的图像上也是如此。
- **在被测量的自由度下不是不变的。**例如，黑白水平条纹的图案不能检测水平转换；齿轮无法帮助测量大尺度旋转。
- **有中性的背景。**如果 ROI 中图案周围的对象可能发生变化，则应通过“无关”像素或蒙板来中和该区域。
- **围绕对象对余量进行了对比**，以便看到前景和背景强度。

自定义参数

可以调整参数以最小化处理时间，但是与整个选定区域匹配时，仍然需要比 EasyFind 更长的时间。

- `DontCareThreshold`：如果需要“无关”区域，相应的像素必须保持低于 `DontCareThreshold` 的值。
  如果所有背景都可以忽略，只需将 `DontCareThreshold` 调整到正确的设置阈值值即可。
  否则，当“无关”区域与阈值图案图像无关时，`DontCareThreshold` 应设置为 1，属于“无关”区域的所有像素应设置为黑色(值 0)。
- `MinReducedArea`：为了实现可接受的时间性能，EasyMatch 对图案进行了部分取样。该参数规定要保留图案图像的最小像素数。值越小，匹配过程越快，但可能会产生不可靠的结果。默认值(64)通常是一个很好的折中。
- `FilteringMode`：如果图像具有尖锐的灰度级转换，最好选择低通内核而不是通常的均匀内核。



学习图案

匹配过程

对于每个新图像，搜索一个或多个出现的图案，允许它使用单个函数调用进行转换、旋转或缩放：

- 匹配：接收目标图像/ROI 作为其参数，并定位图案的期望出现位置。

您可以设置这些参数：

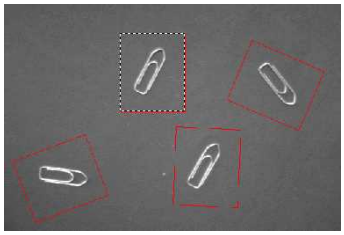
- 旋转范围：`MinAngle`、`MaxAngle`。

- 缩放范围：MinScale、MaxScale。
- 异向性缩放范围：MinScaleX、MaxScaleX、MinScaleY、MaxScaleY。

以下函数返回匹配结果：

- NumPositions 返回找到的良好匹配数。一个良好匹配被定义为得分高于规定值（MinScore阈值）。
- GetPosition 返回第 N 个良好匹配的坐标。按照分数逆序进行排序。

如果要对同一图像匹配多个图案，请为每个图案创建 EMatcher 对象。



匹配图案

高级功能

加快该过程的最佳方法是最小化旋转和缩放，并限制搜索的出现次数。

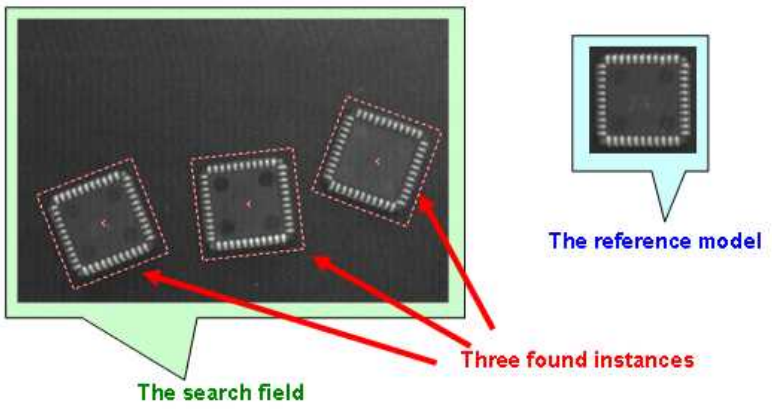
- 学习时间：
  - 优化搜索次数：搜索所有位置需要太长时间，所以按各种尺度（缩小）执行一系列搜索。最粗略的减少是快速和近似。随后的减少发生在邻近地区，以改善位置，大大减少了被尝试的位置数量。位置精度由 2^K给出，其中 K 是减少数。
  - MinReducedArea。表示粗略位置的图案可以有多小。
- 匹配时间：
  - 相关模式（比较图案和图像的方式）：CorrelationMode. **可以是标准、偏移归一化、增益归一化和完全归一化：**在连续色调值上计算相关性。归一化处理可变光条件，在比较之前自动调整图案的对比度和/或强度。
  - 对比度模式（处理对比度反转的方法）：ContrastMode. 照明效果可能导致对象以反向对比度出现，您可以选择是否保留反转实例，以及是仅匹配正出现次数，还是仅匹配负出现次数或两者兼顾。
  - 最大位置（预期匹配数）：MaxPositions、MaxInitialPositions。您可以使用 MaxInitialPositions 参数（在最后阶段逐渐减少达到MaxPositions），让 EasyMatch 粗略地考虑比需要更多的实例。
  - 最低分数（根据该值判断哪个匹配被视为假并将其丢弃）：MinScore、InitialMinScore 。
  - 亚像素精度：插值。图像测量的精度是可选的（精度越差，速度越快）。默认情况下，以像素的精度计算每个自由度的位置参数。可以执行较低的精度。最高精度可达十分之一像素。

- 减少步数：FinalReduction。当粗略位置足够时，可以加快匹配，范围[0 ...NumReductions-1]。
- 非正方形像素：GetPixelDimensions、 SetPixelDimensions。当所获取的图像包含非正方形像素时，旋转的对象会发生偏斜。考虑像素比可弥补这一影响。
- “无关”像素(忽略相关性分数)低于 DontCareThreshold 值。在矩形 ROI 中内切图像时，可将像素值设置为低于阈值，从而忽略某些部分的 ROI。如果不同样本之间的模板零件发生变化，亦可使用此功能。

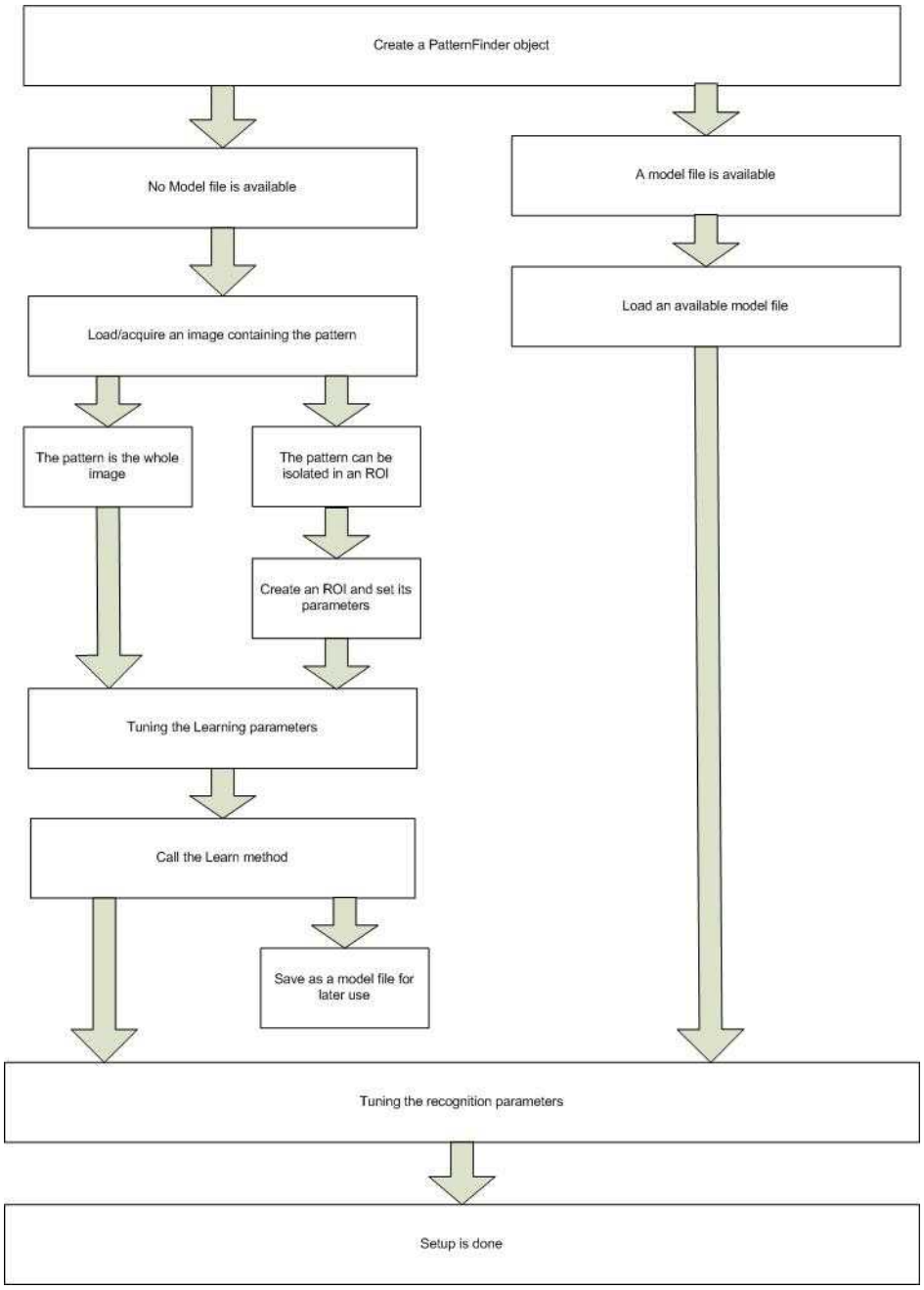
3.4. EasyFind - 匹配几何图案

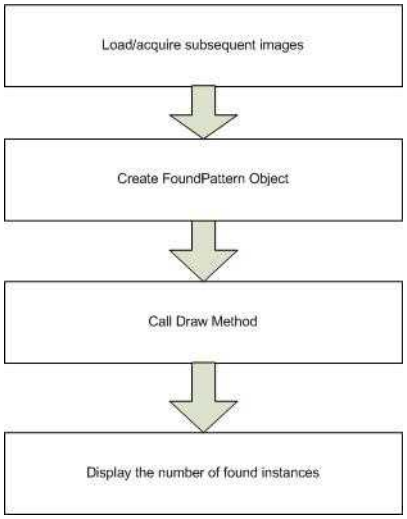
工作流

EasyFind 从一个图案学习一个参考模型，用于在其他图像中查找类似的图案，并且重新获取有关这些实例的信息。
它快速、可靠，对噪声声、模糊、遮挡、缺少部分和照明变化的容错能力强。



工作流

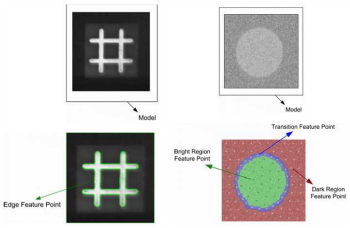

Create a PatternFinder object
No Model file is available
A model file is available
Load/acquire an image containing the pattern
Load an available model file
The pattern is the whole image
The pattern can be isolated in an ROI
Create an ROI and set its parameters
Tuning the Learning parameters
Call the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



特征点定义

特征点是一对坐标(X，Y)和一个类型(边缘、转换或区域)。

EasyFind 使用特征点在搜索字段中查找实例。

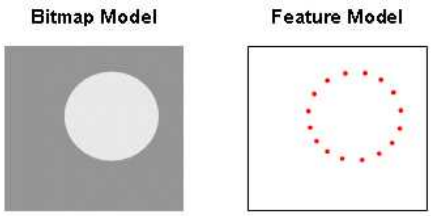


- **边缘特征点**：两个区域之间的灰度级的突然变化表示搜索区域中的该位置处的边缘。
- **转换特征点**：两个区域之间的灰度平滑变化表示其邻域中的转换区域(由上述示例的蓝色区域中的点表示)。可以修改邻域的大小。
- **区域特征点**：识别大致均匀灰度级别的 2 个区域：
  - 黑色区域(由上述示例的红色区域中的一系列点表示)，
  - 明亮区域(由绿色区域中的一系列点组成)。

学习过程

EasyFind 支持各种图案类型(一致边、薄型结构或对比区域)。
在学习过程中，EasyFind 自身计算一个特征模型，该模型是从图案的位图表示中提取出的所有特征点集合。

EasyFind 仅需要该功能模型才能启动查找函数，但您可以创建自己的最佳模型。



最优模型取决于正在搜索的图案类型。

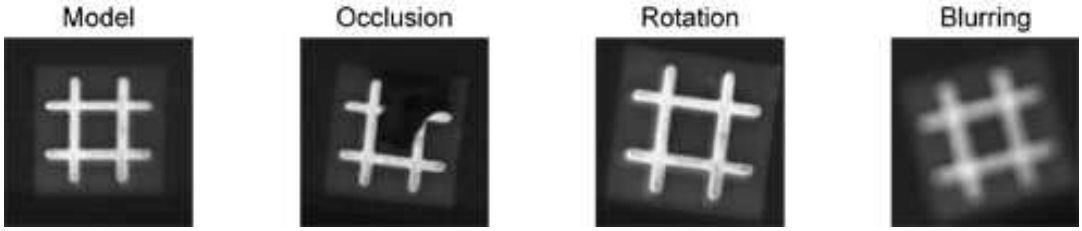
一致边缘

模型必须与锋利的边缘形成良好的对比。它们应该与其余的预期搜索字段大不相同。可以缩放或旋转，能够很好地对抗：模糊、噪声、遮挡、照明变化(逐点得分提高了发现阶段的鲁棒性和计算时间)。

适合具有一致边缘，锐利对比度转换的模型，其区域由(所有搜索字段中每个实例大致位置相同)清晰边缘定界。

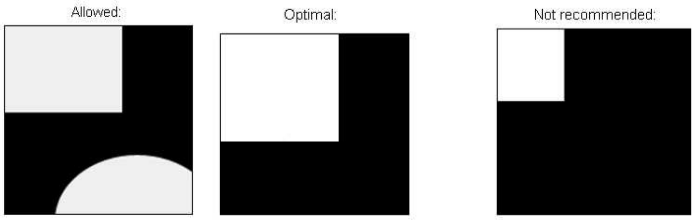
使用`EPatternFinder.ContrastMode`属性选择类似的点：

- PointByPointNormal：如果点具有相同的对比度极性。
- PointbyPointInverse：如果点显示相反的对比度极性。
- PointByPointAny: 不管它们各自的对比度极性。



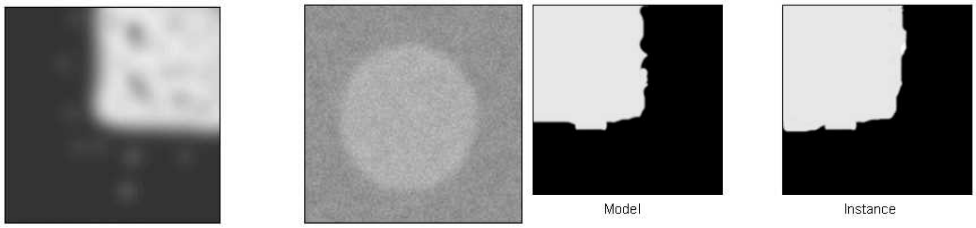
对比区域(由区域特征点和转换特征点定义)

理想情况下，模型应为 50% 亮区和 50% 暗区。可以有多个黑暗和/或明亮的区域。



无法缩放或旋转，能够对抗：模糊、噪声、照明变化(但不遮挡)。

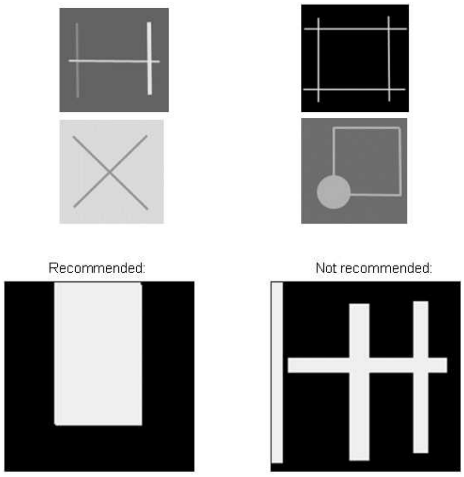
适合具有不一致转换或边缘的模型，或者有多个由转换或边缘定界的区域。



薄型结构(由边缘特征点定义)

可以缩放或旋转，能够对抗：模糊、噪声、遮挡、照明变化。边缘必须在薄型元素和区域之间保持一致，每个薄型元素的对比度应该相同。

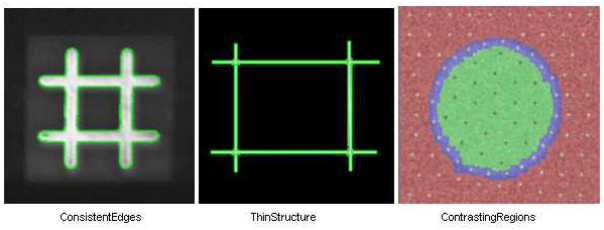
适合包含薄型元素的模型。



检查学习的模型是否正确

EasyFind 可以使用`PatternFinder`对象的`DrawModel`方法在模型上绘制提取的特征点。

在示例中，边缘特征点显示为一致边和薄型结构的绿点，以及对比度区域的交叉线。

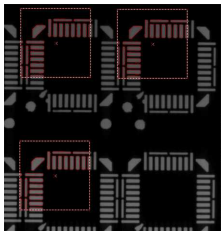


查找过程

您可以通过在配置文件中设置和保存一些参数来优化查找过程。您只需加载该文件并运行以查看 EasyFind 在报告信息中找到的内容。

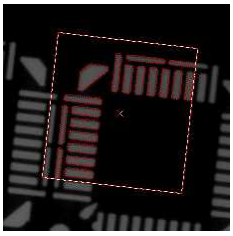
预期实例的最大数量

设置 EasyFind 应返回的最大实例数。在这个例子中，这个数字是三个。



角度和比例(薄型结构和一致边缘图案类型)

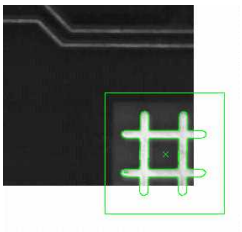
范围包含偏差和公差。例如，对于 20°的角度偏差和 5°的角度公差，EasyFind 返回与学习模型(20°±5°)相关的角度在 15°和 25°之间的实例。



高级功能

查找部分图案

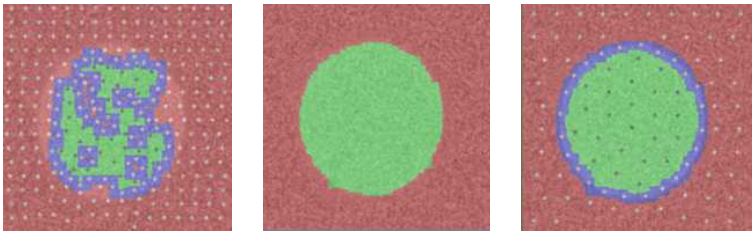
如果搜索字段的扩展设置为 >0 像素，EasyFind 可以找到部分超出搜索范围的薄型结构和一致边缘实例。



调整参数

可以调整所有模型的这些参数：

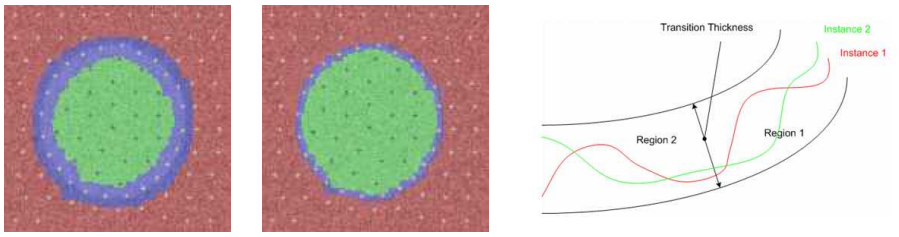
- 使用模型图来调整光平衡以预览模型，使其符合图案的有用部分，然后再次学习模型。



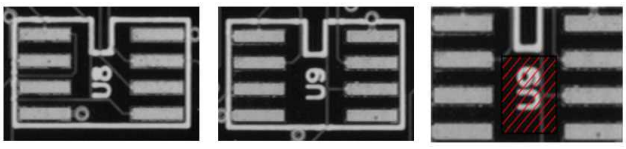
劣质提取(左) - 调整光平衡(中心) - 优质提取(右)

- 设置灰度阈值(这将覆盖光平衡)并再次学习该模型。
- 将枢轴移动到模型中的特定位置，如角落或孔。枢轴是 EasyFind 在找到一个实例时返回的位置(默认为中心)。

- 薄型结构可以从调整这些参数中受益：
  - 自动(与相邻区域之间具有相同对比度的薄型元素)
  - 薄型元素比邻域更暗
  - 薄型元素比邻域更亮
- 对比区域可以从调整这些参数中受益：
  - 转换厚度：转换特征点在*转换带*(下面的示例中的蓝色区域)，这是发生转变的地方。转换厚度参数应允许不同实例的边界保持在转换带内，因此建议使其等于实例之间最大的变化。以下是两个厚度示例以及推荐的厚度：



  - **忽略区域**。零值表示忽略的区域。255 个值表示考虑到的区域。例如：如果模型中心的文本与实例不同，您可以指出 EasyFind 不能从模型的这一部分提取特征点。



3.5. 黄金模板验证 (EChecker)

EChecker需要两个处理步骤，培训和检查。这两个操作完全独立，可以在单独的应用程序中进行编程。

培训涉及预处理参考图像集，以计算每个像素的接受范围，并将其存储在两个阈值图像中，以便与 EasyObject 一起使用。您可以测试“黄金模板”是否良好，如有必要，请调整学习过程参数。培训可以一次性完成，结果存档在黄金模板文件中供以后使用。

检查涉及处理图像，重新对准和规范化，检测落在接受间隔之外的像素，检查质量，然后在必要时调整检查参数。

以下部分介绍了在培训和检查步骤中使用的相关 API 函数。

培训

参考图像进行预处理以计算像素接受范围，并将其存储在两个阈值图像中，以便与 EasyObject(旧版) 一起使用。培训结果存档在模型文件中供以后使用。

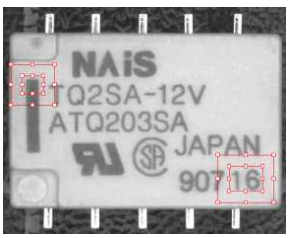
根据参考图像是存储在磁盘上还是即时获取和处理，提供了两种操作模式。

在线操作时不能使用这些操作。

要定义一个模型，按照以下顺序执行几个操作：

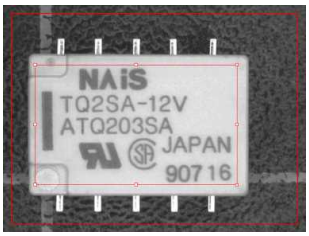
1. 在第一个参考图像上，放置一个或两个 ROI 来定义位置图案(基准标记或地标) 。
   这些 ROI 被其他两个包围以定义可能的运动自由度，并且被用作图案匹配的搜索区域。可以使用拖动手柄交互地放置 ROI。
   EChecker管理拖动操作。
   -Draw以图形方式呈现 ROI 和手柄。
   -HitTest检测其中一个手柄上是否存在光标。
   -Drag移动句柄。
   这些函数可以对所有三种 ROI(图案、搜索和检查) 进行操作。您会很快注意到：

- 拖动图案 ROI 会使相应的搜索 ROI 自动调整，以使公差保持不变。
- 拖动搜索 ROI 会使其大小相对于图案对称地进行调整。
- 调整搜索区域也将检查的 ROI 设置为图像中最大的可用空间。



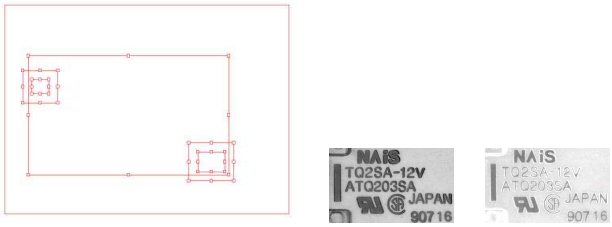
图案 ROI 和搜索 ROI

- 如果培训图像在磁盘上，则可以注册文件名列表，以后可以在单个命令(*批量学习*)中执行学习，而不是*即时学习*)。
- 定义另一个 ROI 来界定要检查的区域。该区域只能包含刚性部分的像素(与基准标记一起移动)，而不是背景。



检查 ROI

- 使用统计培训处理所有图像并进行平均。它使用重新对齐来处理视场中被检测部分的位移，以及灰度规范化以处理全局照明变化。
- 理想情况下，学习传递中应使用至少 16 张图像，以创建低阈值和高阈值图像。用户可以使用全局公差参数来加强或松开接受间隔。调用 `Learn` (ELearningMode_Ready)。`RelativeTolerance`属性调整接受范围。
- 该模型可以保存到包含所有相关信息的单个文件，即 各种 ROI 的放置、基准图案图像、灰度规范化参数和阈值图像。



Echecker 模型的组件

检查参考图像的可靠性。

在对齐 ROI 设置完毕后，应检查其基准位置的可靠性。最好的方法是通过成员函数 `Register`来加载培训图像，显示它们并定位其中的图案。如果定位失败，可以采取不同的纠正措施，具体取决于问题：

- 图案的选择很差。用更稳定的内容定义其他 ROI。
- 搜索区域太紧，从而沿着边缘发现图案。放大搜索区域。
- 图像代表性不足(有缺陷)。最好将其从学习集中撤出。

注意：*首次调用成员`Register`会调用对齐 EasyMatch 上下文的`Learn`成员，即 对图案培训进行了存档。除非调用`Learn` (ELearningMode_Reset)，否则这些图案将用于所有后续对齐操作。要使用的第一个图像可以同时作为定义对准图案和对比度测量的参考。它被称为父级图像。*

如果学习图像保存在磁盘上：每次文件成功加载并被接受为参考图像时，EChecker对象可以将文件路径名添加到列表中。之后，可以在一个命令中处理所有文件。

更多学习

在执行 ROI 放置和图案学习（Register操作）之后，培训仍需要两次传递：

- **“平均”传递**需要计算理想的无噪声图像，显示部分图像的集中趋势。
  对于每个图像，重新对齐和规范化（Register）。如果操作成功（良好的图案定位），请调用Learn（ELearningMode_Average）立即处理（即时学习），或
  AddPathName用于延迟处理（批量学习）。
- **“偏差”传递**衡量平均图像周围的变化。
  对于每个图像，重新对齐和规范化（Register）。如果操作成功，请调用
  Learn（ELearningMode_RmsDeviation）（增强大偏差）或
  Learn（ELearningMode_AbsDeviation）用于即时处理（功能更强大，在大多数情况下推荐）。

原则上，在这两次传递中显示的图像应该是相同的。
如果您不想存档，可以使用两组不同的图像（即时学习）。这两组图像甚至不需要相同的尺寸。

建议图像的学习组大小至少为 16 张。

或者，调用BatchLearn将对添加到文件列表的所有图像执行两个必需的传递。

调整阈值

Learn（ELearningMode_Ready）. 可以调整属性RelativeTolerance以调整接受范围。

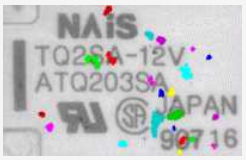
检查

检查很简单：样本图像与模型文件重新对齐，对灰度规范化。

> **注意：***检查的 ROI 必须位于父级图像上。它能够以与图案和搜索区域相同的方式进行交互式定位。可以与其他 ROI 同时设置该 ROI。更改搜索公差将被检查的 ROI 重置为最大可用区域。*
> *EChecker可以在检查时调整全局RelativeTolerance属性，从而改变 EasyObject（旧版）使用的下限和上限阈值图像。*
> *这是唯一可以在学习后更改的EChecker参数。*


被检查的图像，...


... 高和低阈值图像，...

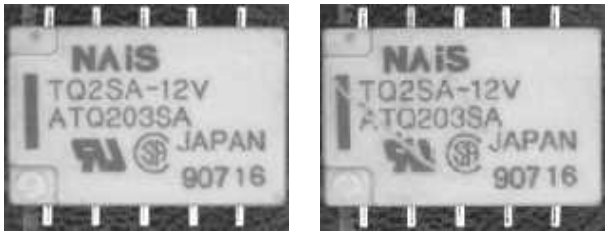

... 和检测到的斑点。

生成的图像被传递给ECodedImage对象进行斑点分析；它们组成相邻像素以形成斑点，丢弃较小的斑点（通常是噪声），测量几何特征（位置、大小、方位、...）以及其他，请参阅EasyObject。

然后将该图像与下参考图像和上参考图像进行比较(在样本和模板之间逐像素地进行比较以检测落在接受间隔之外的像素)。由标准 EasyObject 函数处理缺陷像素。

使用 Echecker 进行比较

EChecker发现模板和样本之间的明显差异，并将其报告在缺陷图中，突出显著差异。EasyObject Blob 分析工具可以定位缺陷，并根据程度、方向、亮度等进行限定。当检测项目为刚性(形状不变)且照明均匀时，确保图像的可重复视觉外观，使得点对点图像比较有意义，是理想的选择。

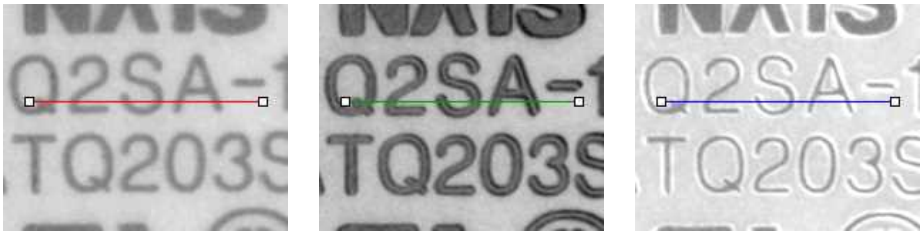


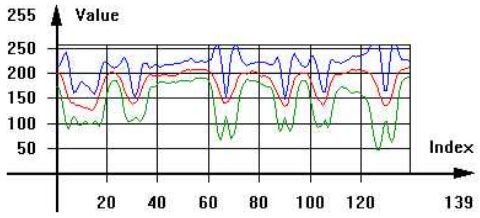
参考图像与具有缺陷的样本图像

统计培训

抓住几个参考图像可以优化正常灰度变化和验收间隔的评估：

- 没有任何变化的相同部分的连续图像(静态测试)给出了对应于噪声分布的灰度分布。
- 不同无缺陷部件的连续图像显示由于部件造成的变化。





接受的灰度范围

图像比较

比较图像的最佳方法是使用EChecker组合一组图像，并确定每个像素的可接受值范围。

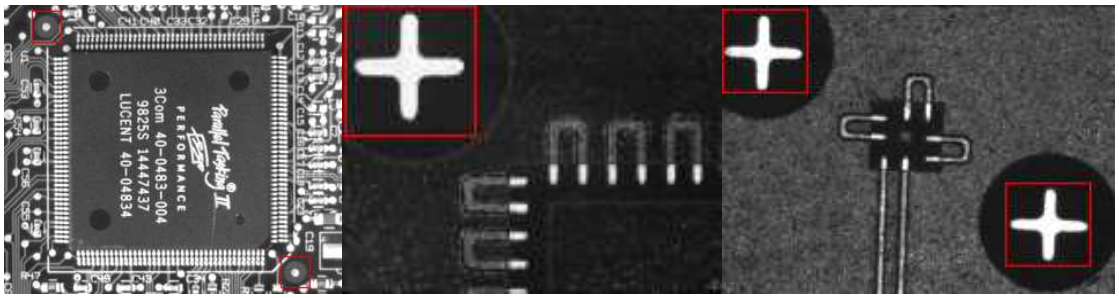
也可以通过减去 2 个图像(使用算术和逻辑函数)来获得绝对值，然后对该差异进行阈值处理，以突出显示图像不同的非零像素来进行比较。

Echecker 执行：

校准

图案匹配测量被检部分在平移、旋转和/或缩放方面的移动(如EasyMatch章节所述)。
当图案被定位时，图像被重新对准，使得被检查部分被带到与参考图像中相同的位置。
一个匹配图案很容易处理平移。
两个匹配的图案提供了更好的旋转测量精度。

当基准标记可用时，它们可以用作准确和可重复位置的地标。
EChecker内部承载了一个或两个匹配的上下文。

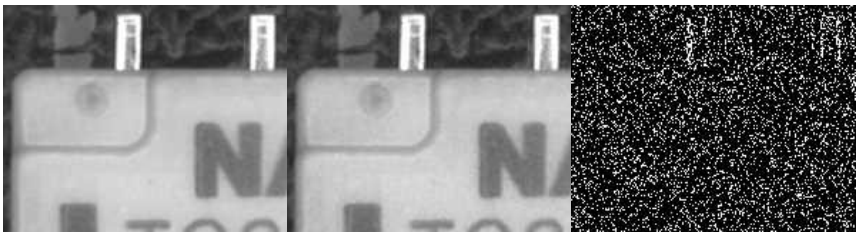


使用基准标记重新排列

该校准方法有三个缺点：

1) 不可避免的变化

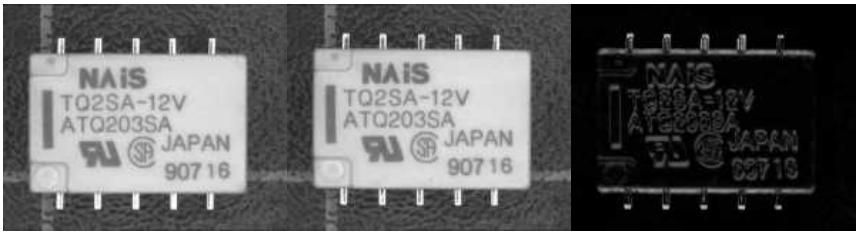
必须引入公差，因为噪声可以使相同的图像在颜色或形状上有轻微的变化。



比较两个噪声图像是否严格相等。
结果图像的黑色像素是不相等的像素。

2) 检查部件的放置很少重复

稍微的错位意味着比较的像素不再对应，所产生的效果在边缘周围特别明显。



比较不对齐的图像

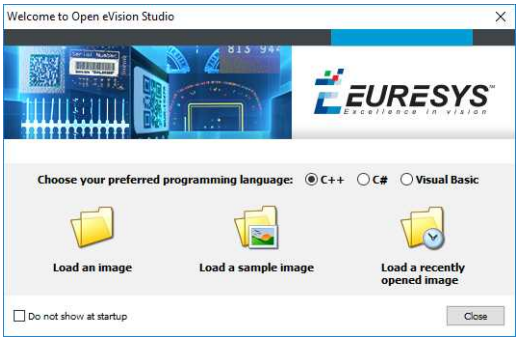
3) 照明不能与部件分离

比较不同照明条件下的图像

4. 使用Open eVision Studio

4.1. 选择您的编程语言

当您第一次启动Open eVision Studio时，显示以下欢迎界面：



1. 选择您的编程语言。

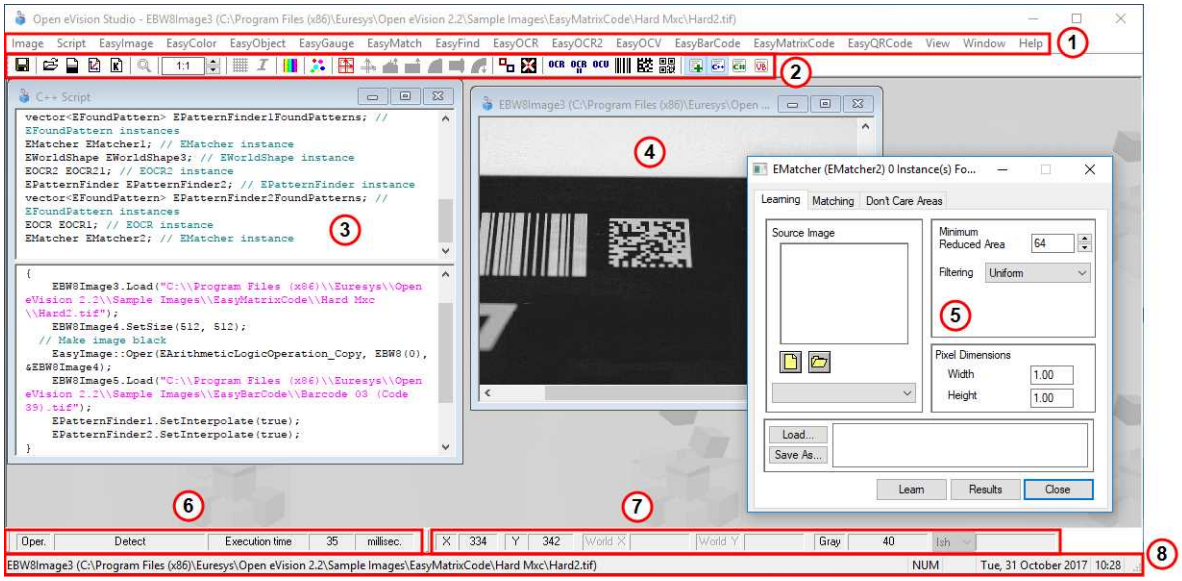
您的选择已保存，下次启动Open eVision Studio时，会自动选择您的编程语言。

> **注意：***当您更换编程语言时，描述窗口中的所有脚本都会被自动删除，窗口内容会重置。*

2. 点击**Load**(加载)按钮，加载一个或多个图片，便于以后处理。
3. 选择 **Do not show at startup**(不要在启动时显示)框，下次启动Open eVision Studio时就会隐藏该欢迎界面。

*如果要在任何时间打开该欢迎界面，更改该设置，进入**Help**(帮助)>**Welcome Screen**(欢迎界面)菜单。*

4.2. 导航界面



Open eVision Studio图形用户界面(GUI)如下所示：

1. *main menu bar* (主菜单栏) 提供所有库的功能和工具快捷方式。

Open eVision Studio无需任何许可证，即可测试所有的库。当然，如果您从您自有应用中的Open eVision Studio拷贝代码，但是没有必需的许可证，那么在运行时您会收到"缺少许可证"的提示信息。

2. *main toolbar*(主工具栏) 可以让您快速访问Open eVision对象，例如图片、形状、标记、条形码、矩阵码等。

3. *script window* (脚本窗口) 显示您在Open eVision Studio中执行的操作所对应的代码，按照您先前所选择的编程语言方式显示。您可以随时将这些代码保存或复制到您的应用程序中。

4. *image windows*(图像窗口) 显示您使用库和工具处理的已打开图片。

5. *tool windows*(工具窗口) 确保您能够轻松配置所有现有的工具。相应的设置会自动添加到脚本窗口中，便于重复使用。

大多数工具窗口是浮动的，您可以轻松将其移动到Open eVision Studio 主窗口外，以便合理利用窗口界面。

6. *execution time bar*(执行时间栏) 显示所选功能在计算机上的精确执行时间(以毫秒或微妙计算) 。这种精确度量帮助您评估应用程序的性能。

7. *color toolbar*(颜色工具栏) 显示诸如图片上曲线X轴和Y轴及对应像素值等当前信息。

8. *status bar* (状态栏) 显示应用程序的基本信息，例如动态图片的文件路径等

4.3. 对图片使用工具

第1步：选择一个工具

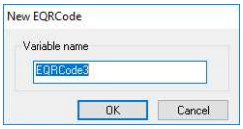
使用Open eVision Studio时，一般第一步时选择您想对该图片使用的库和工具。

要实现这个目的：

1. 在主菜单栏中，点击您想使用的库。
2. 点击您想使用的工具：

*所有库(除EasyImage、EasyColor and EasyGauge之外)只显示一个名为**加载下一个(新Xxx工具)**的工具。这里面的有些库还有其他的功能。*

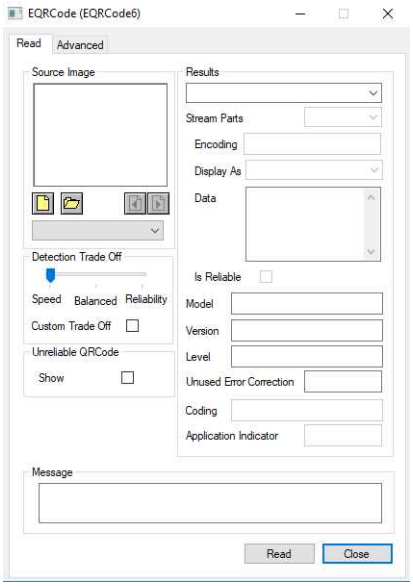
3. 在对话框中，为自动创建且包含处理结果的变量输入一个**Variable name**(变量名)。



EasyQRCode变量创建对话框示例

4. 点击**OK**。

所选择的工具对话框打开。



EasyQRCode变量创建对话框示例

下一步是"第2步：打开图片"在相对页面上。

第2步：打开图片

一旦选择了您的库和工具后，您需要打开要使用该工具的图片。

在所选工具对话框的**Source Image**（源图像）区：

1. 打开一个图片：
   - 点击  **Open an Image**（打开图像）按钮，使用SHIFT和CTRL从计算机中选择一个或多个图片。
   - 或从下拉列表中选择已经打开的图片（或ROIs）。

   注意：*您可以只选择适合格式（JPG、PNG、TIGG或BMP），8位和/或24位（根据库的不同）的图片。*

2. 如果您选择了多个图片，使用  **Load Previous**（加载上一个）  **Load Next**（加载下一个）来激活一个。

该工具会自动应用于任何加载的图片上，此时，结果会按照工具默认的设置显示。

下一步是"第3步：管理ROIs" 低于。

第3步：管理ROIs

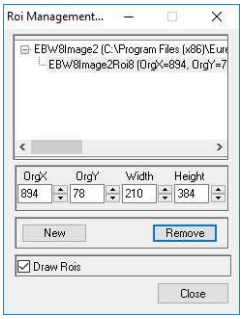
在某些情况下，您往往减少处理时间或挑出您想阅读的对象，而不想处理的整体图像，只处理该图像中一个或多个选定矩形部分，或ROIs（感兴趣区域）。

在Open eVision中，ROIs附在一个图像上，只在父图像存在时显示。

创建ROI

1. 打开图像：
   - 如果图像已经打开，激活相应的图像窗口。
   - 如果如未打开，转到主菜单：**Image**（图像）> **Open**...（打开），打开一张图片。
2. 要创建ROI，进入主菜单：**Image**（图像）> **ROI Management**...（ROI管理）。

ROI Management（ROI管理）窗口显示如下所示。

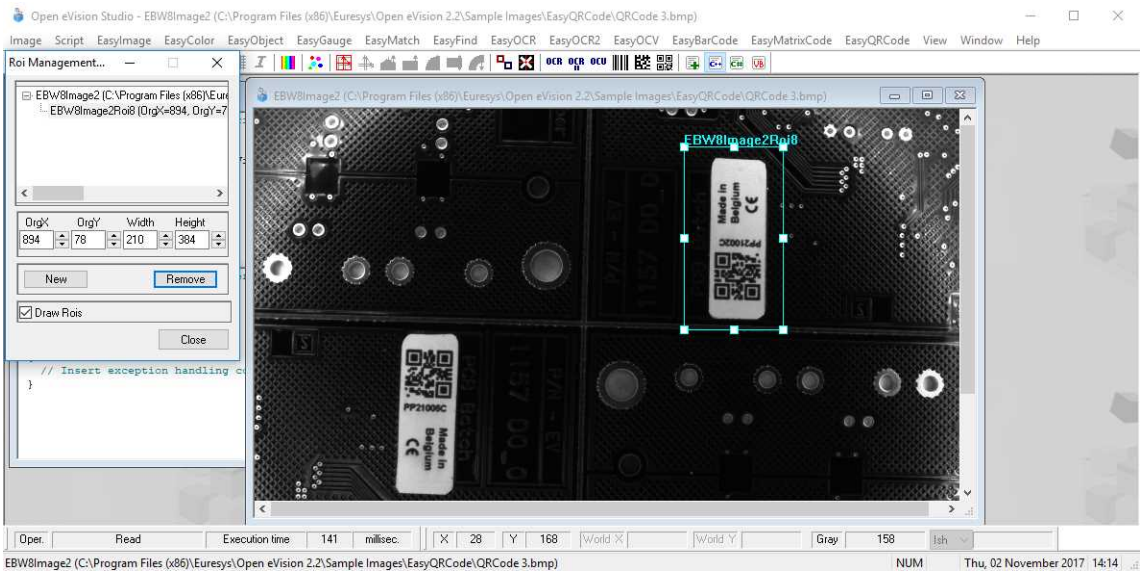


3. 在路径树中选择图像。

4. 点击**New**(新建)按钮。

5. 在对话框中，输入新ROI的**Variable name**(变量名)。

ROI在图像上以矩形形式显示出来，如下所示。



6. 拖动ROI的一角和侧边，将其莫移动到规定的位置。

7. 点击**Close**(关闭)按钮，关闭**ROI Management**(ROI管理)窗口。

下一步是"第4步：配置工具" 在相对页面上。

管理ROIs

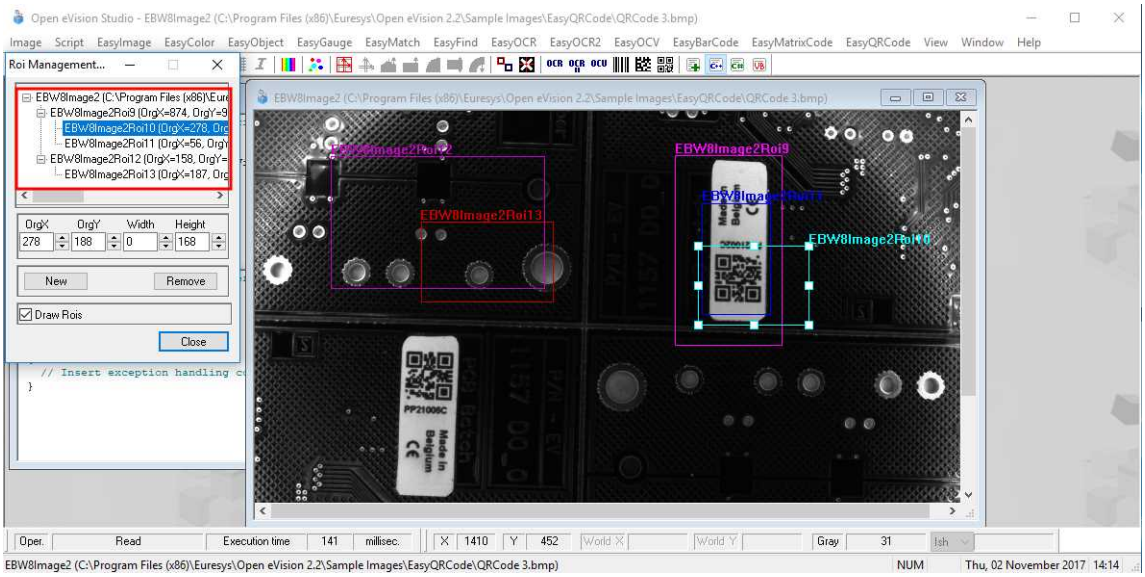
您可以添加、修改和移动ROIs。

一张图像可以有多个ROIs。每个ROI都可以直接附着到图像上(意味着其位置相对于图像是相对的的)或其他ROI上(意味着其位置相对于其"父"ROI)是相对的。

1. 要管理ROIs，进入主菜单：**Image** >(图像) **ROI Management**...(ROI管理)。

ROI Management(ROI管理)窗口带ROI关系树显示，如下所示。

如果**Draw Rois**选项卡已选定，所有的ROIs都已不同的颜色显示在图像上。



2. 从ROI关系树上选择一个ROI。
3. 拖动ROI一角或侧边，改变选定ROI的位置和尺寸(以及所有附着在其上的ROI的位置)。
4. 点击**New**(新建)按钮，在选定ROI上新增ROI。

在ROI关系树顶部选择图像，直接将ROI附着到图像上。

5. 点击**Remove**(删除)按钮，删除选定的ROI(即附着在其上的所有ROI)。
6. 点击**Close**(关闭)按钮，关闭**ROI Management**(ROI管理)窗口。

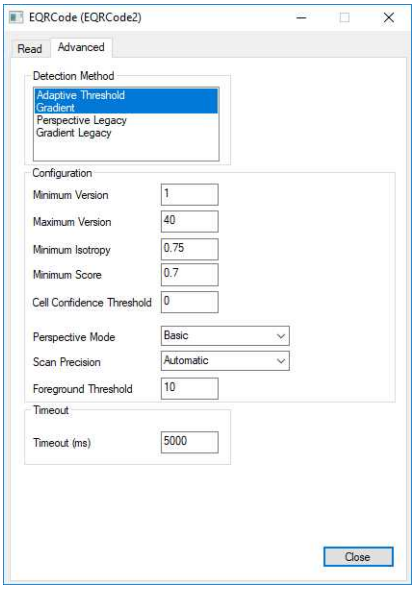
第4步：配置工具

当您的图像(包括您创建的ROI)已经准备好，您需要配置工具。

在工具窗口：

1. 打开各种选项卡。

在创建新的工具时，所有的参数都设置为默认值。



EasyQRCode工具的参数选项卡示例

2. 在每个选项卡中，设置期望的参数值。

请参考"Functional Guide"(功能指南)和"Reference Manual"(参考手册)，查看参数的详细信息、功能和默认值。

具体行为，例如学习和使用量具，请参考"Functional Guide"(功能指南。)

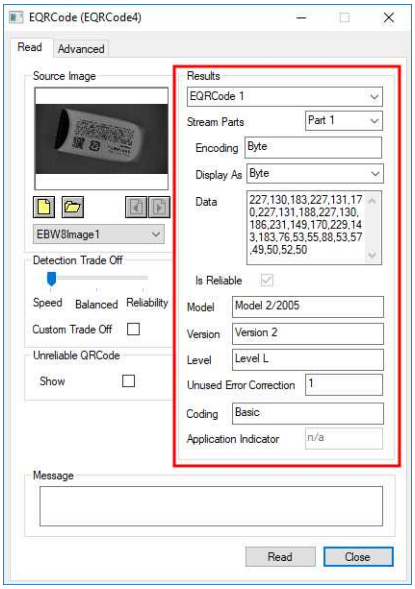
3. 运行工具并分析下一步"第5步：运行工具和检查执行时间"低于中描述的结果。

第5步：运行工具和检查执行时间

一旦您设置了工具参数后，运行您的工具，如有需要，检查在您计算机上执行时间。

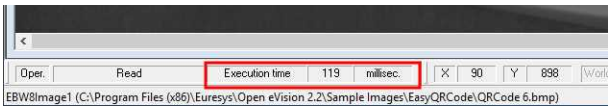
在工具窗口：

1. 根据库功能，点击**Read**(读取)、**Detect**(删除)、**Results**(结果)或**Execute**(执行)按钮，在选定图像上执行工具。
2. 检查图像上、结果栏或结果区里的结果，如图所示。



读取二维码后的结果示例

3. 如果您未得到期望结果：
   - 尝试更改参数(开始用默认值，然后一次修改一个参数)。
   - 如果您的图像不够好，尝试按照所述进行增强。
4. 检查在主Open eVision Studio窗口左下部执行时间栏中的执行时间。



执行时间

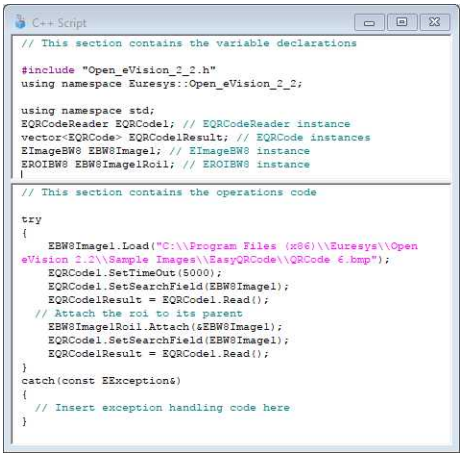
执行时间是您计算机上测量的实际处理时间。它取决于您的计算机处理器、内存、操作系统等，当然还有执行时的处理器负载。因此，该执行时间在每次执行时都不尽相同。

5. 要获得更具代表性的执行时间，请单击**Read**(“读取”)、**Detect**(“检测”)、**Results**(“结果”)或**Execute**(“执行”)按钮几次，计算平均执行时间。
6. 如果您的应用要求减小执行时间，请尝试：
   - 修改工具参数；
   - 在您的图像上增加一个很火多个ROI；
   - 增强图像。

下一步是"第6步：使用生成的代码"低于。

第6步：使用生成的代码

默认时，Open eVision Studio将所有的界面操作转化为选定的语言代码，如下所示。



如果您觉得工具结果适合您，您可以保持或复制生成的代码，在您自己的应用程序中使用。

复制粘贴代码到您的应用中

在脚本窗口中：

1. 选择您要复制的代码段。
2. 在该代码上鼠标右击，在菜单中点击**Copy**（复制）。
3. 进入开发环境工具中，然后将代码粘贴进去。

保存代码

1. 进入**Script**（脚本）菜单。
2. 点击**Save Script As...**（保存脚本为）。
3. 输入文件名和路径，保存代码为文本文件。

管理生产的代码

在**Script**（脚本）菜单中，您可以：

- 选择编程语言（请注意，如果您更改语言，脚本窗口的内容会自动删除）。
- 激活或停用**Script Code Generation**（脚本代码生成）。如果您想执行某些操作，单不保存为代码，则停用该选项。

4.4. 预处理和保存图片

您什么时候应该预处理图片？

当然，最好的方法是建立您自己的图像采集系统，优质简单处理图像，这样Open eVision工具才能灵活高效的运行。

如果没办法或不容易实现，您可以预处理您的图片或您的ROIs，为您要运行的Open eVision工具增强和准备图片。

您可以采用多种现有功能，调整图像的增益和偏置值，领用EasyImage和EasyColor功能，执行回旋、阈值、缩放、旋转和白平衡，增强等高校等。

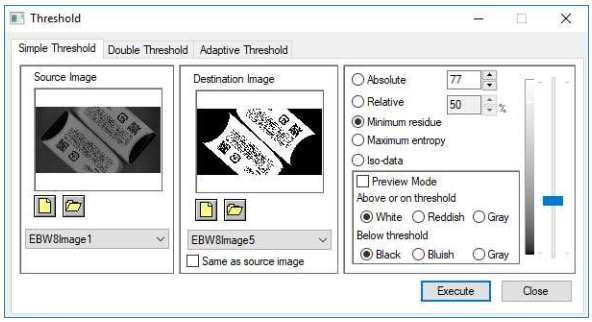
预处理图片

预处理图片和实用工具的不同点在于，预处理生成一张新图片，而工具主要是提取和检索信息，不会改变图片。

预处理图片或ROI：

1. 点击主菜单栏中您想要使用的库 (EasyImage或EasyColor)。
2. 点击您想使用的功能。

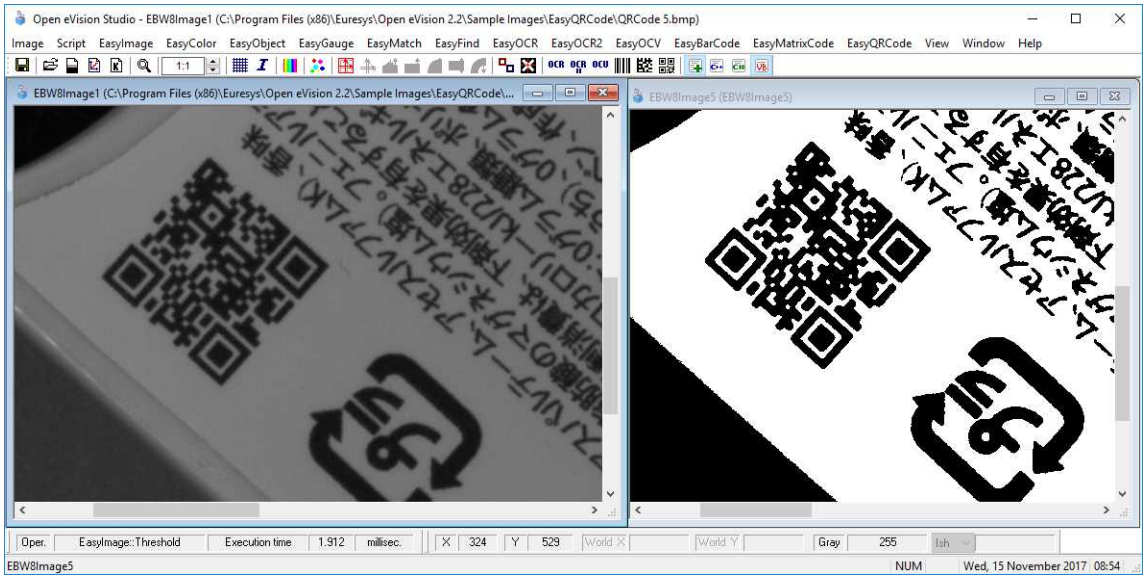
大多数功能对话框类似于下图所示，有2个图片选择区和一个参数设置区。



已处理对话框举例(使用EasyImage阈值)

3. 如果您选择的功能有多个版本，打开对应的选项卡。
4. 打开**Source Image**(源图像)区中的源图片(如"第2步：打开图片" 于页面91所述)。
5. 在**Destination Image**(目标图像)中打开或创建新的目标图片。
6. 设置参数。
7. 点击**Execute**(执行)按钮。

如下图所示，预处理的图片就已经在目标图片中。



源图像和目标图像(带easyimage阈值)

8. 如果要使用Open eVision Studio外的目标图片，请按如下所述保存。

保存图片

1. 单击要保存的图片，并激活。
2. 要打开保存菜单，可以：
   - 鼠标右键点击图片
   - 或者打开主菜单>**Image**(图像)
3. 点击**Save as**...(另存为)。
4. 选择文件格式(JPEG、JPEG2000、PNG、TIFF或Bitmap)。
5. 输入文件名称，选择路径。
6. 点击**Save**(保存)按钮。

5. Tutorials

5.1. EasyObject

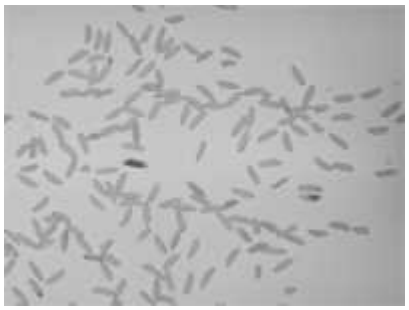
Removing Non-Significant Objects After Image Segmentation

"Image Segmenter" 于页面127

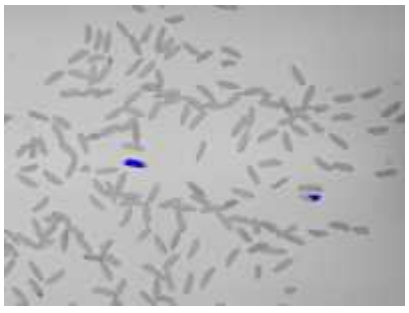
Objective

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.



Source image



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.

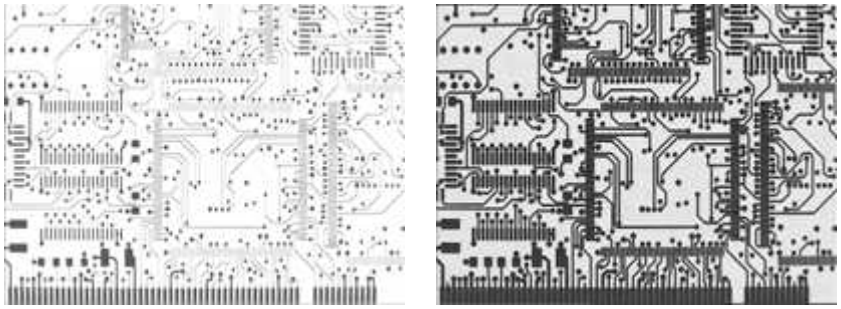
Detecting Differences Between Images Using Min-Max References

"Selecting and Sorting Blobs" 于页面130

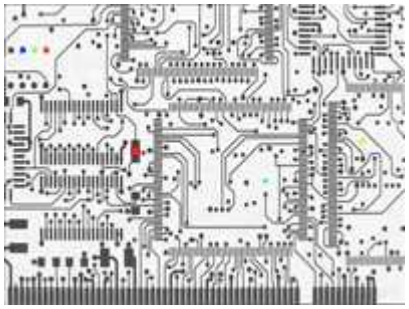
Objective

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

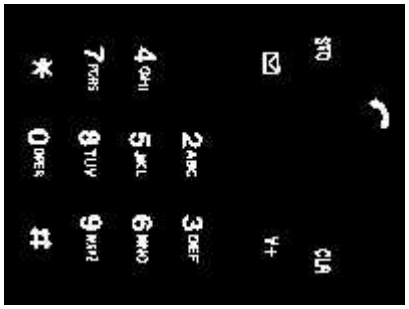
"Generating a Flexible Mask from an Encoded Image" 于页面131

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).

Source image



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

"Pattern Learning" 于页面133

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

"Pattern Matching and Retrieving Results" 于页面134

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

"Pattern Learning" 于页面133

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"

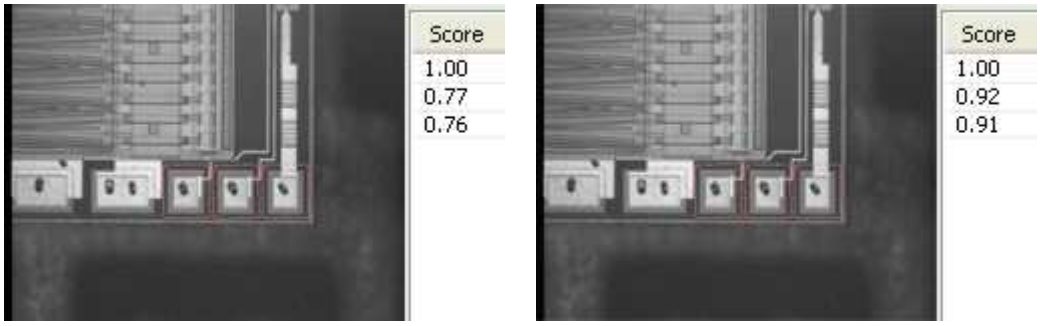
"Setting Search Parameters" 于页面133

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

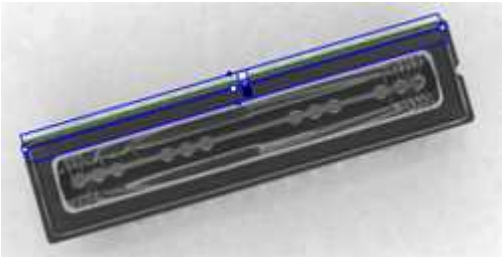
Measuring the Rotation Angle of an Object

"Line Fitting" 于页面137

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.

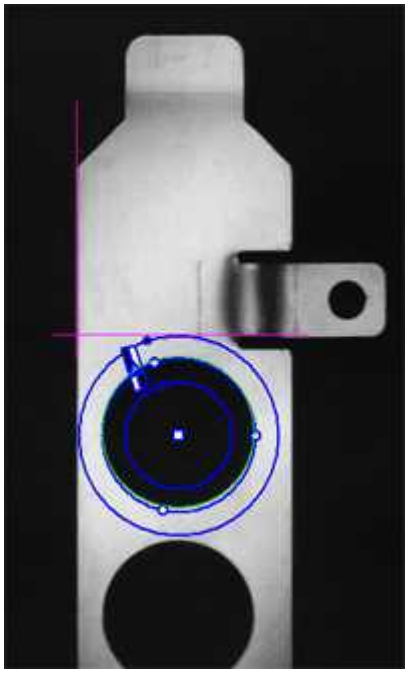
Measuring the Diameter of a Circle

"Circle Fitting" 于页面 138

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.

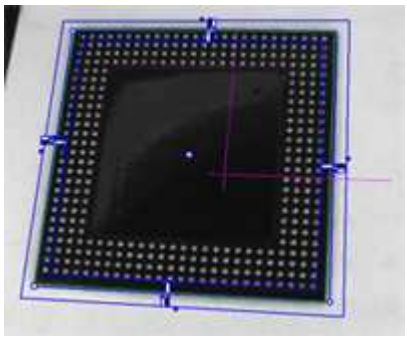
Measuring a Distorted Rectangle

"Rectangle Fitting" 于页面138

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.

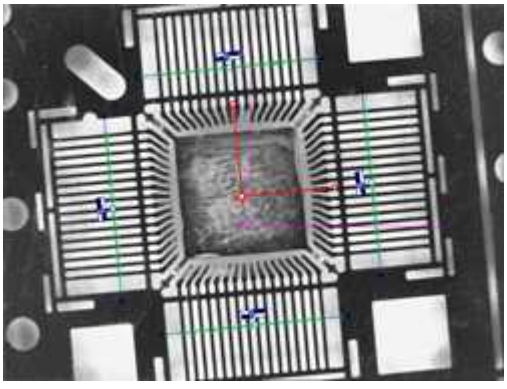
Locating Points Regarding to a Coordinate System

"Point Location" 于页面137

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

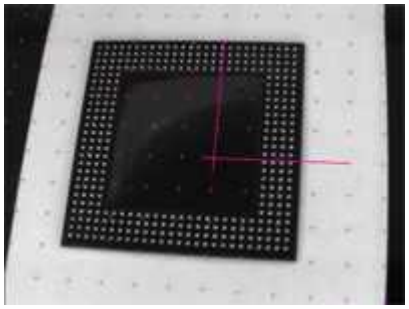
1. The image is automatically inspected. However, clicking Process in the world shape dialog box will insert the corresponding code into the script window.
2. From the point gauge dialog boxes, click the Results tab to retrieve the located points coordinates. All measurement results are expressed in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

Unwarping a Distorted Image

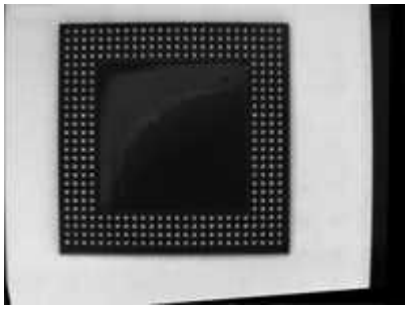
Objective

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

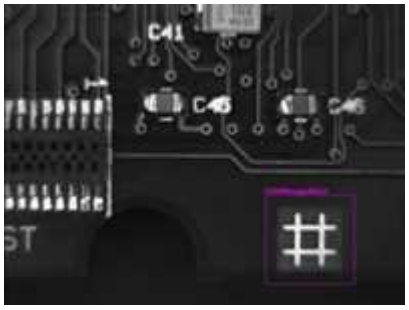
Detecting Highly-Degraded Occurrences of a Reference Model in Multiple Files

"Pattern Finding and Retrieving Results" 于页面136

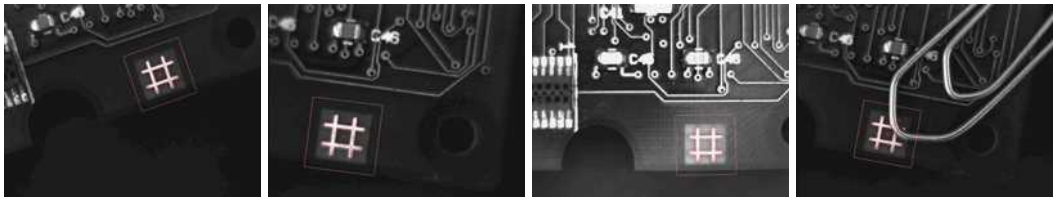
Objective

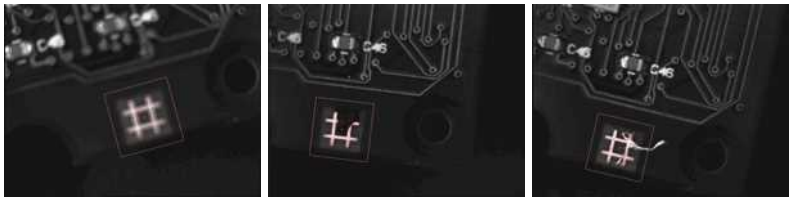
Following this tutorial, you will learn how to use EasyFind to detect in multiple images highly-degraded occurrences of a reference model. The degradation can be due to noise, blur, occlusion, missing parts or unstable illumination conditions.

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.

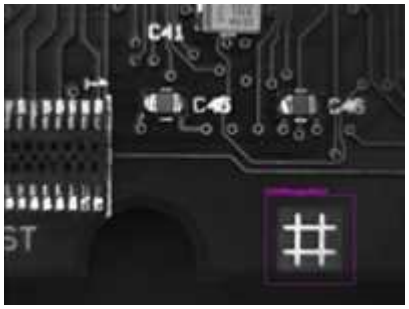
Improving the Score of Found Instances by Using "Don't Care Areas"

"Setting Search Parameters" 于页面135

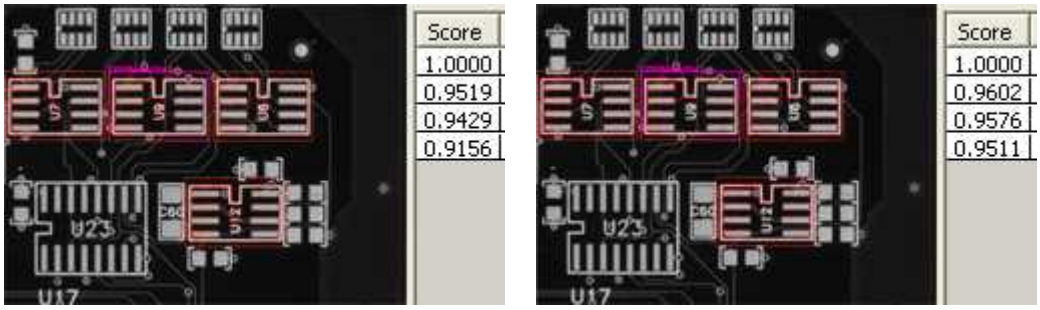
Objective

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. Basic Types

Loading and Saving Images

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

```
// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();
```

```
// 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);
```

Interfacing Third-Party Images

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

```
// Images constructor
EImageBW8 srcImage= new EImageBW8();
```

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

```
//Pointer to the third-party image buffer
IntPtr imgPtr = bufferPointer;
```

```
// ...
```

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

Retrieving Pixel Values

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

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

```
using System.RunTime.InteropServices;
```

```
IntPtr pixAddr;
byte pix;
```

```
//...
```

```
for(int y = 0; y < height; ++y)
 pixAddr = bw8Image.GetImagePtr(0,y)
 for(int x = 0; x < width; ++x)
 pix = Marshal.ReadByte(pixAddr,x)
```

ROI Placement

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

```
// Image constructor
EImageBW8 parentImage= new EImageBW8();
```

```
// ROI constructor
EROIBW8 myROI= new EROIBW8();
```

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

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

Vector Management

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

```
// EBW8Vector constructor
EBW8Vector ramp= new EBW8Vector();
EBW8 bw8 = new EBW8();
```

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

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

```
 bw8.Value = (byte)i;
 ramp.AddElement(bw8);
}
```

```
// Retrieve the 10th element value
EBW8 value = ramp.GetElement(9);
```

Exception Management

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

```
try
{
 // Image constructor
 EImageC24 srcImage= new EImageC24();
```

```
 // ...
```

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

```
catch(EException exc)
{
```

```
 // Retrieve the exception description
 string error = exc.What();
}
```

6.2. EasyObject

Constructing the Blobs

Image Encoder

```
////////////////////////////////////////////////////////////
// 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= new EImageBW8();
```

```
// Image encoder
EImageEncoder encoder= new EImageEncoder();
```

```
// Coded image
ECodedImage2 codedImage= new ECodedImage2();
```

```
// ...
```

```
// 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)
Euresys.Open_eVision_1_1.Segmenters.EGrayscaleSingleThresholdSegmenter segmenter=
encoder.GrayscaleSingleThresholdSegmenter;
segmenter.BlackLayerEncoded= true;
segmenter.WhiteLayerEncoded= false;
```

```
segmenter.Mode= EGrayscaleSingleThreshold.Absolute;
segmenter.AbsoluteThreshold= 110;
```

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

Image Segmenter

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// Image encoder
EImageEncoder encoder= new EImageEncoder();
```

```
// Coded image
ECodedImage2 codedImage= new ECodedImage2();
```

```
// ...
```

```
// Set the segmentation method to GrayscaleDoubleThreshold
encoder.SegmentationMethod= ESegmentationMethod.GrayscaleDoubleThreshold;
```

```
// Retrieve the segmenter object
Euresys.Open_eVision_1_1.Segmenters.EGrayscaleDoubleThresholdSegmenter segmenter=
encoder.GrayscaleDoubleThresholdSegmenter;
```

```
// Set the high and low threshold values
segmenter.HighThreshold= 150;
segmenter.LowThreshold= 50;
```

```
// Specify the layers to be encoded (neutral layer only)
segmenter.BlackLayerEncoded= false;
segmenter.NeutralLayerEncoded= true;
segmenter.WhiteLayerEncoded= false;
```

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

Holes Extraction

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// Image encoder
EImageEncoder encoder= new EImageEncoder();
```

```
// Coded image
ECodedImage2 codedImage= new ECodedImage2();
```

```
// ...
```

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

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

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

Continuous Mode

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// Image encoder
EImageEncoder encoder= new EImageEncoder();
```

```
// Coded image
ECodedImage2 codedImage= new ECodedImage2();
```

```
// ...
```

```
// Enable the continuous mode
encoder.ContinuousModeEnabled= true;
```

```
// Loop to acquire 50 different chunks
for (int count = 0; count < 50 ; count++)
{
// Store the new chunk into srcImage
// ...
```

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

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

Computing Blobs Features

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// Image encoder
EImageEncoder encoder= new EImageEncoder();
```

```
// Coded image
ECodedImage2 codedImage= new ECodedImage2();
```

```
// ...
```

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

```
for (int index = 0; index < codedImage.GetObjCount(); index++)
{
// Retrieve the selected blob gravity center
        EObject blob = codedImage.GetObj(index);
float centerX = blob.GravityCenter.X;
```

```
float centerY = blob.GravityCenter.Y;
}
```

Selecting and Sorting Blobs

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

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

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

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

// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature.Area, 100,
ESingleThresholdMode.Less);

// Retrieve the number of remaining blobs
int numBlobs= selection.ElementCount;

// Sort the remaining blobs based on their area
selection.Sort(EFeature.Area, ESortDirection.Ascending);

// Retrieve the selected blobs
for (int index = 0; index < numBlobs; index++)
{

 float centerX= selection.GetElement(index).GravityCenterX;
 float centerY= selection.GetElement(index).GravityCenterY;
}
```

Using Flexible Masks

Constructing Blobs

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

```
// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask = new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

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

Generating a Flexible Mask from an Encoded Image

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

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// 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);
```

Generating a Flexible Mask from a Blob Selection

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

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();
```

```
// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// 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= new EObjectSelection();
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

Pattern Learning

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// ROI constructor
EROIBW8 pattern= new EROIBW8();
```

```
// EMatcher constructor
EMatcher matcher= new EMatcher();
```

```
// ...
```

```
// 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);
```

Setting Search Parameters

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

```
// Image constructor
EImageBW8 pattern= new EImageBW8();
```

```
// EMatcher constructor
EMatcher matcher= new EMatcher();
```

```
// ...
```

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

```
// Set the maximum number of occurrences
matcher.MaxPositions= 5;
```

```
// Set the rotation tolerances
matcher.MinAngle= -20.0f;
matcher.MaxAngle= 20.0f;
```

```
// Enable sub-pixel accuracy
matcher.Interpolate= true;
```

```
// Set the minimum score
matcher.MinScore= 0.70f;
```

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

Pattern Matching and Retrieving Results

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// EMatcher constructor
EMatcher matcher= new EMatcher();
```

```
// ...
```

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

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

```
// Retrieve the number of occurrences
int numOccurrences= matcher.NumPositions;
```

```
// 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

Pattern Learning

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// ROI constructor
EROIBW8 pattern= new EROIBW8();
```

```
// EPatternFinder constructor
EPatternFinder finder= new EPatternFinder();
```

```
// ...
```

```
// 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);
```

Setting Search Parameters

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

```
// Image constructor
EImageBW8 pattern= new EImageBW8();
```

```
// EPatternFinder constructor
EPatternFinder finder= new EPatternFinder();
```

```
// ...
```

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

```
// Set the maximum number of occurrences
finder.MaxInstances= 5;
```

```
// Set the rotation tolerances
finder.AngleTolerance= 20.0f;
```

```
// Set the minimum score
finder.MinScore= 0.70f;
```

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

Pattern Finding and Retrieving Results

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

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// EPatternFinder constructor
EPatternFinder finder= new EPatternFinder();
```

```
// EFoundPattern constructor
EFoundPattern[] foundPattern= null;
```

```
// ...
```

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

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

```
// Retrieve the number of instances
int numInstances= foundPattern.Length;
```

```
// Retrieve the score and the
// position of the first instance
float score= foundPattern[0].Score;
float centerX= foundPattern[0].Center.X;
float centerY= foundPattern[0].Center.Y;
```

6.5. EasyGauge

Point Location

```
////////////////////////////////////////////////////////////////
// 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= new EImageBW8();
```

```
// EPointGauge constructor
EPointGauge pointGauge= new EPointGauge();
```

```
// Adjust the transition parameters
pointGauge.TransitionType= ETransitionType.Wb;
pointGauge.TransitionChoice= ETransitionChoice.Closest;
```

```
// Set the gauge nominal position
pointGauge.SetCenterXY(256.0f, 256.0f);
```

```
// Set the gauge length to 10 units and the angle to 45°
pointGauge.SetTolerances(10.0f, 45.0f);
```

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

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

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

Line Fitting

```
////////////////////////////////////////////////////////////////
// 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= new EImageBW8();
```

```
// ELineGauge constructor
ELineGauge lineGauge= new ELineGauge();
```

```
// Adjust the transition parameters
lineGauge.TransitionType= ETransitionType.Bw;
lineGauge.TransitionChoice= ETransitionChoice.NthFromEnd;
lineGauge.TransitionIndex= 2;
```

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

```
EPoint center= new EPoint(256.0f, 256.0f);
ELine line= new ELine(center, 50.0f, 20.0f);
lineGauge.SetLine(line);
```

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

```
// Get the origin and end point coordinates of the fitted line
EPoint originPoint = lineGauge.MeasuredLine.Org;
EPoint endPoint = lineGauge.MeasuredLine.End;
```

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

Circle Fitting

```
////////////////////////////////////////////////////////////////////
// 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= new EImageBW8();
```

```
// ECircleGauge constructor
ECircleGauge circleGauge= new ECircleGauge();
```

```
// Adjust the transition parameters
circleGauge.TransitionType= ETransitionType.Bw;
circleGauge.TransitionChoice= ETransitionChoice.LargestAmplitude;
```

```
// Set the Circle fitting gauge position, diameter (50 units),
// starting angle (10°), and amplitude (270°)
EPoint center= new EPoint(256.0f, 256.0f);
ECircle circle= new ECircle(center, 50.0f, 10.0f, 270.0f);
circleGauge.SetCircle(circle);
```

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

```
// Get the center point coordinates and the radius of the fitted circle
float centerX = circleGauge.MeasuredCircle.Center.X;
float centerY = circleGauge.MeasuredCircle.Center.Y;
float radius = circleGauge.MeasuredCircle.Radius;
```

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

Rectangle Fitting

```
////////////////////////////////////////////////////////////////////
// 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= new EImageBW8();
```

```
// ERectangleGauge constructor
ERectangleGauge rectangleGauge= new ERectangleGauge();
```

```
// Adjust the transition parameters
rectangleGauge.TransitionType= ETransitionType.Bw;
rectangleGauge.TransitionChoice= ETransitionChoice.LargestAmplitude;
```

```
// Set the rectangle fitting gauge position,
// size (50x30 units) and orientation (15°)
rectangleGauge.SetCenterXY(256.0f, 256.0f);
rectangleGauge.SetSize(50.0f, 30.0f);
rectangleGauge.Angle = 15.0f;
```

```
// Measure
rectangleGauge.Measure(srcImage);
```

```
// Get the size and the rotation angle of the fitted rectangle
float sizeX = rectangleGauge.MeasuredRectangle.SizeX;
float sizeY = rectangleGauge.MeasuredRectangle.SizeY;
float angle = rectangleGauge.MeasuredRectangle.Angle;
```

```
// Save the point gauge measurement context
rectangleGauge.Save("myRectangleGauge.gge");
```

Wedge Fitting

```
////////////////////////////////////////////////////////////////////
// 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= new EImageBW8();
```

```
// EWedgeGauge constructor
EWedgeGauge wedgeGauge= new EWedgeGauge();
```

```
// Adjust the transition parameters
wedgeGauge.TransitionType= ETransitionType.Bw;
wedgeGauge.TransitionChoice= ETransitionChoice.NthFromBegin;
wedgeGauge.TransitionIndex= 0;
```

```
// Set the wedge fitting gauge position, diameter (50 units),
// breadth (-25 units), starting angle (0°) and amplitude (270°)
EPoint center= new EPoint(256.0f, 256.0f);
EWedge wedge= new EWedge(center, 50.0f, -25.0f, 0.0f, 270.0f);
wedgeGauge.SetWedge(wedge);
```

```
// Measure
wedgeGauge.Measure(srcImage);
```

```
// Get the inner and outer radius of the fitted wedge
float innerRadius = wedgeGauge.MeasuredWedge.InnerRadius;
float outerRadius = wedgeGauge.MeasuredWedge.OuterRadius;
```

```
// Save the point gauge measurement context
wedgeGauge.Save("myWedgeGauge.gge");
```

Gauge Grouping

Gauge Hierarchy

```
////////////////////////////////////////////////////////////
// This code snippet shows how to create a gauge hierarchy //
// and save it into a file. //
////////////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();
```

```
// Gauges constructor
ERectangleGauge rectangleGauge= new ERectangleGauge();
ECircleGauge circleGauge1= new ECircleGauge();
ECircleGauge circleGauge2= new ECircleGauge();
```

```
// ...
```

```
// 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.Name= "myCircleGauge1";
```

```
// ...
```

```
// Save worldShape together with its daughters
worldShape.Save("myWorldShape.gge", true);
```

Complex Measurement

```
////////////////////////////////////////////////////////////
// This code snippet shows how to trigger the measurement //
// of a whole gauge hierarchy and retrieve the results. //
////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();
```

```
// Load the EWorldShape together with its daughters
worldShape.Load("myWorldShape.gge", true);
```

```
// Retrieve the number of worldShape's daughters
int numDaughters= worldShape.NumDaughters;
```

```
// ...
```

```
// 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.MeasuredRectangle.SizeX;
```

```
// Retrieve the measurement result of a
// daughter gauge called "myCircleGauge1"
ECircleGauge circleGauge= (ECircleGauge)worldShape.GetShapeNamed("myCircleGauge1");
EPoint center= circleGauge.MeasuredCircle.Center;
```

Calibration using EWorldShape

Calibration by Guesswork

```
//////////////////////////////////////////////////////
// This code snippet shows how to perform a calibration //
// by guesswork. //
//////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();
```

```
// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();
```

```
// ...
```

```
// Compute the calibration coefficients
// Field of view: 32x24 mm
worldShape.SetSensor(srcImage.Width, srcImage.Height, 32.0f, 24.0f);
```

```
// Retrieve the spatial resolution
float resolutionX= worldShape.XResolution;
float resolutionY= worldShape.YResolution;
```

Landmark-Based Calibration

```
//////////////////////////////////////////////////////
// This code snippet shows how to perform a landmark-based //
// calibration. //
//////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();
```

```
// ...
```

```
// 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
 sensorPoint = myIthLandmark_Sensor;
 worldPoint = myIthLandmark_World;

 // Add the I-th pair
        worldShape.AddLandmark(sensorPoint, worldPoint);
}
```

```
// Perform the calibration
worldShape.Calibrate((int)ECalibrationMode.Skewed);
```

Dot Grid-Based Calibration

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a dot grid-based //
// calibration. //
////////////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();
```

```
// ...
```

```
// 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
 dotPoint = myIthDot;
```

```
// 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);
```

Coordinates Transform

```
////////////////////////////////////////////////////////////
// This code snippet shows how to convert coordinates from //
// the Sensor space to the World space and conversely. //
////////////////////////////////////////////////////////////
```

```
// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();
```

```
// EPoint constructor
EPoint sensor= new EPoint();
EPoint world= new EPoint();

// ...

// Perform the calibration
worldShape.Calibrate((int)ECalibrationMode.Scaled | (int)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);
```

Image Unwarping

```
////////////////////////////////////////////////////////
// This code snippet shows how to unwarp an image based //
// of the computed calibration coefficients. //
////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// Lookup table constructor
EUnwarpingLut lut= new EUnwarpingLut();

// ...

// Perform the calibration
worldShape.Calibrate((int)ECalibrationMode.Tilted | (int)ECalibrationMode.Radial);

// Setup the lookup table for unwarping
worldShape.SetupUnwarp(lut, srcImage, true);

// Perform the image unwarping
worldShape.Unwarp(lut, srcImage, dstImage, true);
```