# Open eVision Tutorial

# Open eVision 教學手冊

# 目錄


免責聲明 ..... 5

大綱 ..... 6

EasyImage ..... 7

將灰階圖像轉換成二進制圖像(二值化) ..... 8

提取物件輪廓 ..... 11

將灰階圖像轉換成黑白邊緣圖像 ..... 14

如何使用 Harris 角點檢測器檢測物體的角落 ..... 16

使用投影來檢測水平線段或者垂直線段 ..... 18

建立一個自訂的彈性遮罩(flexible mask) ..... 21

使用彈性遮罩計算並統計灰階度 ..... 25

使用 hit-and-miss 轉換來偵測物件的頂角 ..... 27

使用剖面(Profile)功能來提取向量 ..... 30

增強 X 光圖像 ..... 33

修正不均勻亮度 ..... 35

修正剪切效應(shear effect) ..... 37

修正偏斜效應(skew effect) ..... 40

EasyColor ..... 42

圖像的色彩分割 ..... 43

進行色彩分割 ..... 47

EasyObject ..... 49

圖像分割後移除不重要的物體 ..... 50

使用最小 - 最大參考檢測圖像之間的差異 ..... 55

使用遮罩來檢測印刷錯誤 ..... 58

EasyMatch ..... 62
學習樣本(pattern)及新增 EasyMatch 模型(model)檔案 ..... 63
根據模型(model)檔案來匹配樣本(pattern) ..... 64
在 ROI 中根據模型(model)來匹配樣本(pattern) ..... 67
透過使用忽略區域(don't care areas)來改善樣本的匹配分數 ..... 70
EasyFind ..... 74
檢測多個高度衰減圖像中的參考模型 ..... 75
透過可忽略區域(don't care areas)提高實例分數 ..... 79
EasyGauge ..... 85
針對特定座標系統進行點座標定位 ..... 86
量測物件的旋轉角度 ..... 92
測量一個圓的直徑 ..... 95
測量歪斜的矩形 ..... 99
反變形扭曲的圖像 ..... 103
EasyOCR ..... 106
學習字元並建立 EasyOCR 字型 ..... 107
根據字型識別字元 ..... 113
EasyOCV ..... 115
建立 EasyOCV 模型檔 ..... 116
根據模型檢測圖像中的字元 ..... 118
根據模型文件檢查 ROI 中的符號 ..... 120
學習統計模型的使用 (1) ..... 122
學習統計模型的使用 (2) ..... 125
EasyBarCode ..... 128

自動讀取條碼 ..... 129
EasyMatrixCode ..... 133
自動讀取二維條碼 ..... 134
學習二維條碼建立 EasyMatrixCode model 檔 ..... 136
附錄 A ：基本載入與顯示圖像範例程式碼 ..... 138

## 免責聲明

Euresys 應保留硬體和軟體以及 Euresys 商標的所有財產權利、所有權和利益。

本文檔中提及的所有公司和產品名稱可能是其各自所有者的商標。

Euresys 的硬體和軟體，品牌或檔案的許可，使用，租賃，出租，翻譯，複製，複製或修改。包含在本文件中，恕不事先通知。

Euresys 可隨時自行決定修改產品規格或更改本文件中提供的資訊，恕不另行通知。

Euresys 對因使用 Euresys 的硬體和軟體、或因本文件中的遺漏或錯誤所引起的任何種類的收入，利潤，商譽，數據，資訊系統或其他任何特殊的，附帶的，間接的，相應的或懲罰性的損失或損害概不負責。

## 大綱

關於應用程式的基本界面、操作和工作流程，請參閱另外 Help 檔所提供的 Open eVision Studio 入門。 關於 Open eVision 功能概念以及有關 API 類別、方法和屬性的資訊，請參閱 Open eVision 用戶指南。

透過圖形界面通常有很多不同的方式來獲得相同的結果，下面的教學流程只顯示每種情況的其中一種方式。

下面的教學流程中使用的所有圖像預設目錄位於：

C:\Program Files (x86)\Euresys\Open eVision 2.2\Sample Images

由於這個根路徑是不變的，所以在每個教學流程中都不會重複提及。

預設情況下，在主功能表中的 [Activate/Deactivate script code generation] 可以用來開啟按鈕。這會自動在腳本視窗中插入與圖形界面操作所相對應的程式碼。但請注意，直到點擊對話框按鈕顯示執行的相關操作後(如：[Execute], [Find], [Learn]...)，才會將程式碼插入到腳本中 。

## EasyImage


將灰階圖像轉換成二進制圖像(二值化)

提取物件輪廓

將灰階圖像轉換成黑白邊緣圖像

如何使用 Harris 角點檢測器檢測物體的角落

使用投影來檢測水平線段或者垂直線段

建立一個自訂的彈性遮罩(flexible mask)

使用彈性遮罩計算並統計灰階度

使用 hit-and-miss 轉換來偵測物件的頂角

使用剖面(Profile)功能來提取向量

增強 X 光圖像

修正不均勻亮度

修正剪切效應(shear effect)

修正偏斜效應(skew effect)

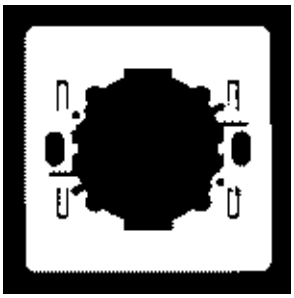
## 將灰階圖像轉換成二進制圖像(二值化)

### 目標

依照下面的教學流程，你將學會如何使用 EasyImage 把灰階圖像轉換成一個二值化圖像。圖像二值化(Thresholding)是將所有灰階像素轉換為黑色或白色像素，取決於它們的像素值是否低於或高於指定的臨界值(threshold)。將圖像二值化可以使圖像更容易進一步地分析，首先，你將需要載入一個圖像 (step 1)，接著設定二值化的參數 (step 2) ，並執行二值化轉換 (step 3)。



**來源灰階圖像**



**來源圖像經過二值化之後的結果**

### Step 1: 載入來源圖像

- 從主選單，點選 [EasyImage]，接著點選 [Threshold]。
- 在 [Simple Threshold] (簡單二值化) 的頁面中，點選 Source Image (來源圖像) 區域中的 [Open] 圖示，然後載入下面的圖檔: **EasyMatch\Switch1.tif**。
- 對新的圖像物件保留預設的變數名稱，然後點選 [OK]。

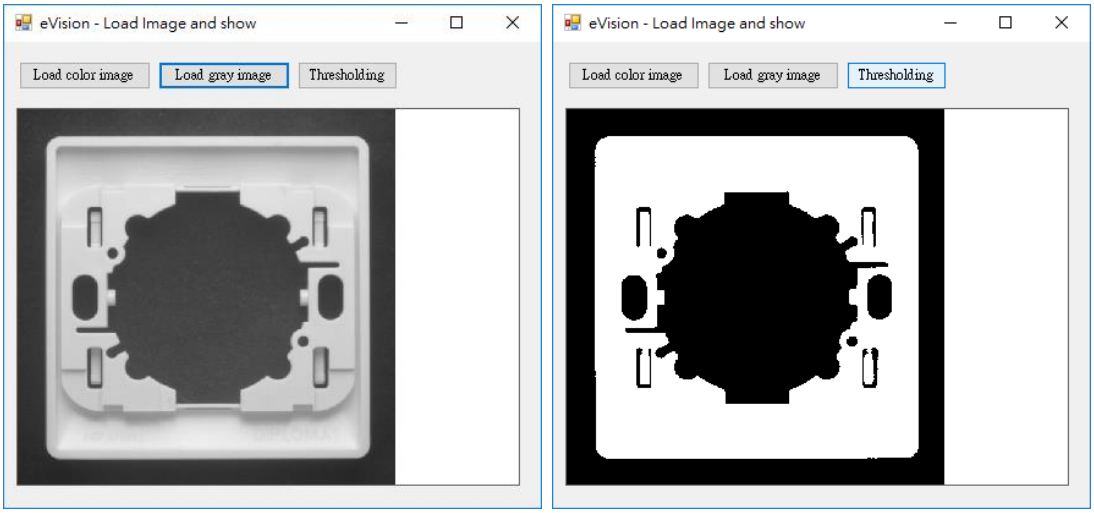
**Step 2: 設定 thresholding 參數**

- 在 Threshold 對話框的右側區域 ，可以用滑鼠上下移動 Slider 來調整 threshold，並且可以直接在來源圖像中預覽二值化的結果。
- 選擇 [Minimum residue] (最小殘留量) 選項來透過預先定義的演算法來自動找出最佳的 threshold 數值。

**Step 3: 執行二值化轉換**

- 點選目標圖像區域中的 [New] 圖示來建立新的目標圖像。
- 對新的圖像物件保留預設的參數，接著點選 [OK]。
- 在 Threshold 對話框中，點選 [Execute] 來執行二值化並將結果顯示在目標圖像中。

範例程式執行畫面




程式碼片段：

```
private void btnThreshold_Click(object sender, EventArgs e)
{
    EBW8Image2.SetSize(EBW8Image1.Width, EBW8Image1.Height); //設定大小跟來源圖像一致
    EasyImage.Oper(EArithmeticLogicOperation.Copy, new EBW8(0), EBW8Image2); //圖像複製過來
    //進行二值化，因為版本變更，預設的設定是MinResidue，所以不用加入工具產生的第三個參數
```

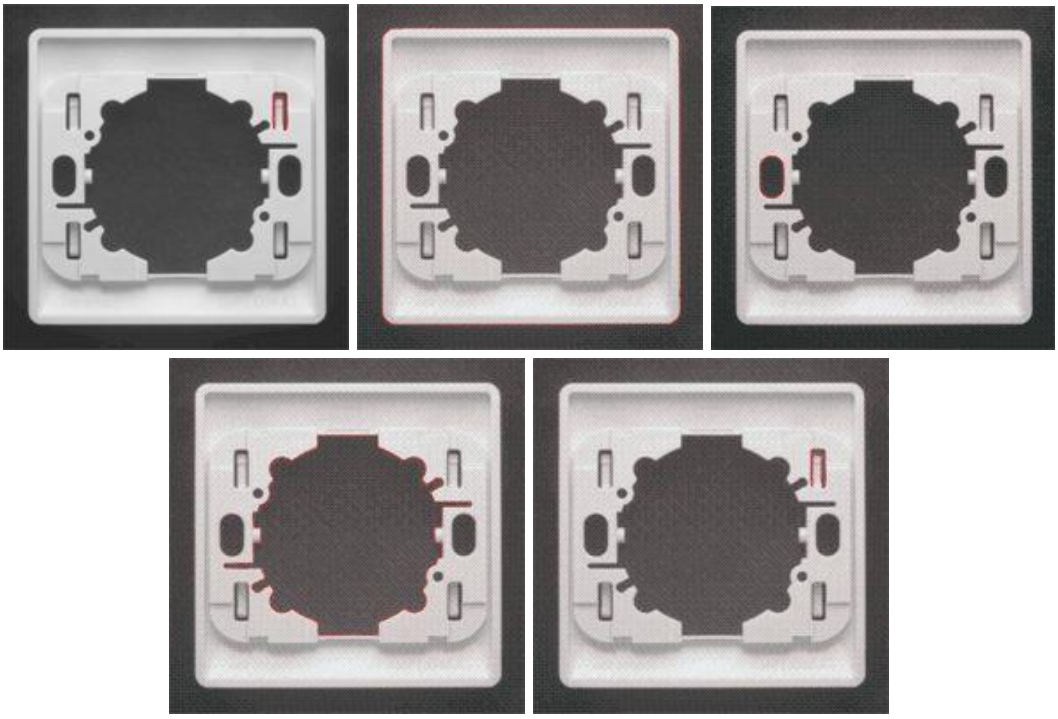
```
    EasyImage.Threshold(EBW8Image1, EBW8Image2); //EThresholdMode.MinResidue
    ShowImage(EBW8Image2, pbImage); //顯示出來
}
```

## 提取物件輪廓

### 目標

依照下面的教學流程，您將學習如何使用 EasyImage 來追蹤灰階圖像中目標物件的輪廓。 輪廓提取允許您取得構成輪廓路徑上所有點的向量，只需點擊目標物件的邊緣即可。

您首先需要載入圖像 (step 1)， 並設置一個包含所有輪廓點的向量 (step 2)。 然後點擊一個物體邊緣，輪廓將自動提取出來 (step 3)。



**從物體邊緣提取輪廓**

### Step 1: 載入來源圖像

- 從主功能表中，點擊 [EasyImage]，接著按下 [Contour]。
- 點擊 [Open] 源圖像區， 並載入圖像文件 **EasyMatch\Switch1.tif**.
- 保留新圖像的預設名稱，並點擊 [OK]。

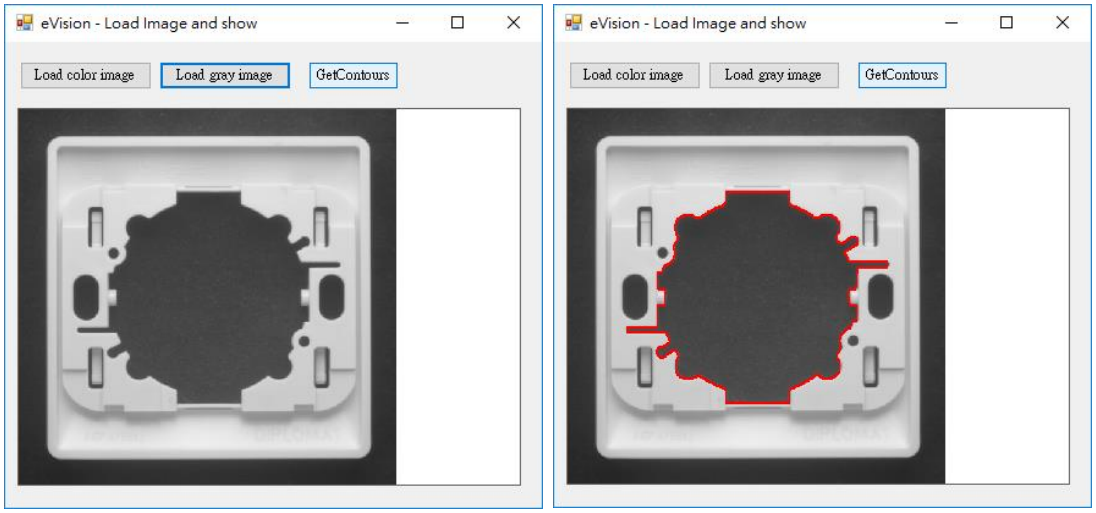
### Step 2: 設置目標向量

- 點擊目標向量(destination vector)區域中的 [New] 圖示。
- 保留新向量的預設設置和變數名稱，並點擊 [OK]。

### Step 3: 提取輪廓

- 將游標移動到圖像上方時，會出現一個箭頭，移動箭頭到欲提取輪廓之物體邊緣上， 並點擊之。從該點擊邊緣開始追蹤物體的輪廓，並在物體周圍出現紅線以標記輪廓，接著目標向量中會依照輪廓順序填入所有構成輪廓的點座標資料。

範例程式執行畫面




程式碼片段：

```
private void btnContours_Click(object sender, EventArgs e)
{
   //原本產生之ContourThreshold要改為EContourThreshold
   EasyImage.Contour(EBW8Image1, EContourMode.ClockwiseContinueIfBorder, 300, 395,
EContourThreshold.Above, 91, EConnexity.Connexity4, EBW8PathVector1, false);
   //EBW8PathVector1.Draw(pbImage.CreateGraphics(), EBW8ScaleRatio); //繪製輪廓在picturebox上面

   Graphics g;
```

```
    g = pbImage.CreateGraphics(); //建立繪圖物件  // 如果我要畫在表單上  Me.CreateGraphics()
    Pen p = new Pen(Color.Red);//, 1);
    int x, y, v;
    for (int i = 0; i < EBW8PathVector1.NumElements; i++)
    {
        x = (int)(EBW8PathVector1.GetElement(i).X*ScalingRatio);
        y = (int)(EBW8PathVector1.GetElement(i).Y*ScalingRatio);
        v = EBW8PathVector1.GetElement(i).Pixel.Value;
        Console.WriteLine(i + "(" + x + "," + y + ") " + v);
        g.DrawLine(p, x , y, x , y+1); //自己畫沒有比較快，還要乘上比例，位置才會正確
    }
}
```

## 將灰階圖像轉換成黑白邊緣圖像

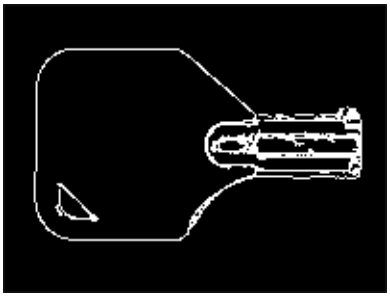
### 目標

依照下面的教學流程，將學習如何使用 EasyImage 把灰諧圖像轉換成二進制圖像，只保留圖像邊緣的部分，此功能使用 Canny 邊緣偵測演算法來達成。

您需載入來源圖檔 (step 1)，並用 Canny 邊緣偵測演算法進行轉換 (step 2)。



**來源圖像**



**經 Canny 邊緣偵測演算法轉換後的圖像**

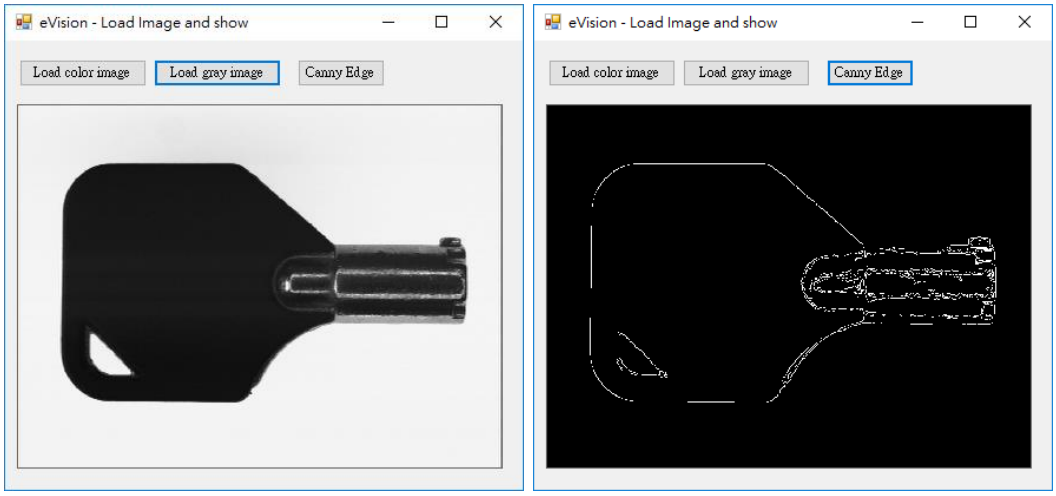
**步驟 1:** 載入來源圖像

- 從主功能表上，按 [EasyImage]， 然後按下 [Canny Edge Detector]。
- 保留預設的變數名稱， 並按下 [OK]。
- 用滑鼠按下來源圖像區域的 [Open] 圖示，並載入下面檔名的圖像：
  **EasyImage\Key1.tif**
- 保留預設的變數名稱， 並按下 [OK]。

### Step 2: 執行 Canny 邊緣偵測演算法

- 點擊目標圖像區域的 [New] 圖示，並建立新的目標圖像。
- 保留預設的設定，並按下 [OK].
- 在 Canny 邊緣檢測對話框中，點擊 [Apply]，邊緣偵測結果會呈現在目標圖像中。

範例程式執行畫面




程式碼片段：

```
private void btnCanny_Click(object sender, EventArgs e)
{
   EBW8Image2.SetSize(EBW8Image1);
   cannyEdgeDetector1.Apply(EBW8Image1, EBW8Image2); //套用Canny邊緣偵測演算法
   ShowImage(EBW8Image2, pbImage);
}
```

## 如何使用 Harris 角點檢測器檢測物體的角落

### 目標

依照下面的教學流程，您將學習如何使用 EasyImage 偵測圖像上的角點，偵測方法是使用哈里斯 (Harris) 角點偵測演算法。

首先，需要載入一個來源圖像 (step 1)，接著執行哈里斯 (Harris) 角點偵測演算法 (step 2)。



**在圖像中偵測到所有的角點**

**Step 1:** 載入來源圖像

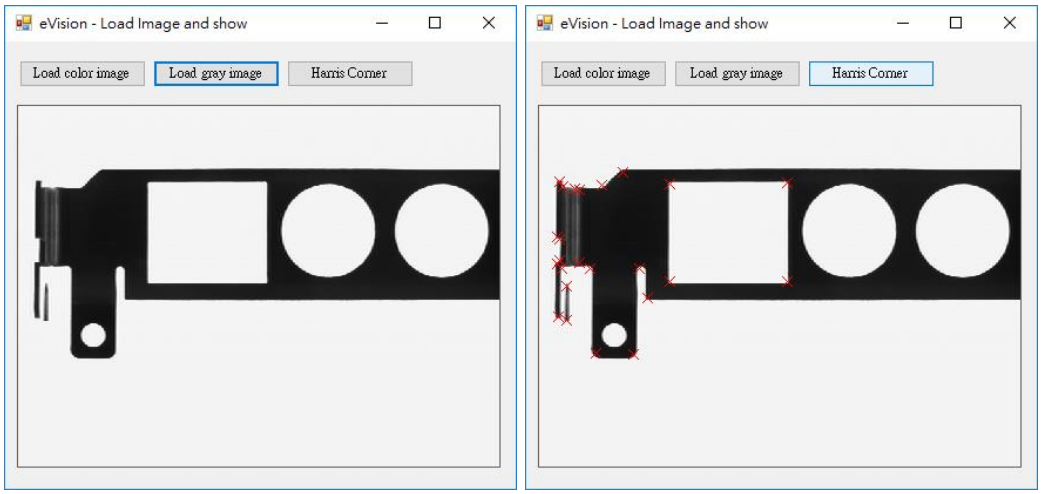
- 從主功能表中，點擊 [EasyImage]，然後點擊 [Harris Corner Detector]。
- 保留預設的變數名稱，然後點擊 [OK]。
- 點擊來源圖像區域的 [open] 圖示，然後載入下面的圖像檔：EasyGauge\ Bracket1.tif
- 保留預設的變數名稱，然後點擊 [OK]。

**Step 2:** 執行哈里斯角點偵測演算法

- 在哈里斯 (Harris) 角點偵測器的對話框中，針對 Scale 屬性輸入 2.3。
- 點擊 [Apply] 來執行角點偵測。
- 點擊 [Results] 來顯示出所有偵測到的角落座標。

- 按下 [Columns] 按鈕可以將其他屬性內容顯示在結果列表中。

## 範例程式執行畫面




## 程式碼片段：

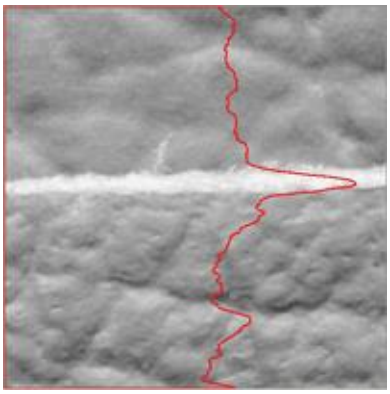
```
private void btnHarris_Click(object sender, EventArgs e)
{
   harrisCornerDetector1.IntegrationScale = 2.30f; //設定整合度參數
   harrisCornerDetector1.Apply(EBW8Image1, harrisCornerDetector1InterestPoints);
   harrisCornerDetector1InterestPoints.Draw(pbImage.CreateGraphics(), ScalingRatio);
   EPoint p; //eVision的點座標物件
   for (int i = 0; i < harrisCornerDetector1InterestPoints.PointCount; i++) //針對每個角點
   {
      p = harrisCornerDetector1InterestPoints.GetPoint(i);
      Console.WriteLine("(" + p.X + "," +p.Y + ")");
   }
}
```

## 使用投影來檢測水平線段或者垂直線段

### 目標

依照下面的教學流程，您將學習如何使用 EasyImage 檢測灰階圖像中的具有水平或者垂直線特徵的缺陷。

首先，載入來源圖像 (step 1)，設置向量 (step 2)，然後檢測缺陷(水平線) (step 3)。



**使用圖像投影可以檢測到的缺陷**

**Step 1:** 載入來源圖像

- 從主選單中， 點選 [EasyImage]， 接著點選 [Projection]。
- 點選來源圖像區域的 [Open] 圖示，然後載入下列的圖像檔：EasyImage\Leather.bmp。
- 針對新圖像保留預設的變數名稱，然後點選 [OK]。

**Step 2:** 設置目標的向量

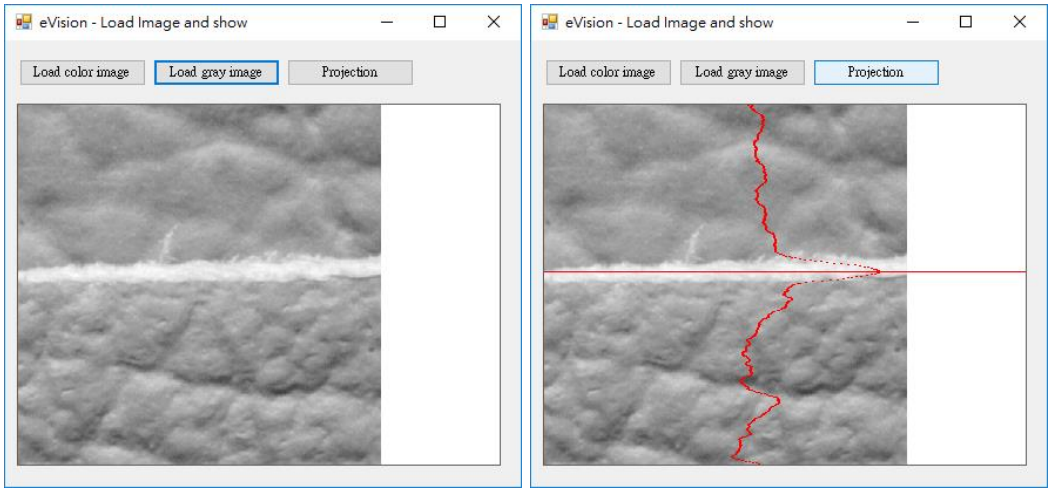
- 點選 [New] 目標向量區域中的圖示。
- 為向量類型選擇 BW32 的選項，然後點選 [OK]。

**Step 3:** 檢測缺陷的部分

- 在圖像投影對話框中，選擇 [column] 的按鈕，然後點選 [Execute] 來執行操作。

- 讓向量和相應曲線圖的結果顯示在目標向量的視窗中，圖形的結果也會出現在圖像上。每個向量值是對應水平行（或垂直列）上所有像素值的總和。這意味著水平（或垂直）缺陷很容易被檢測出來。

範例程式執行畫面



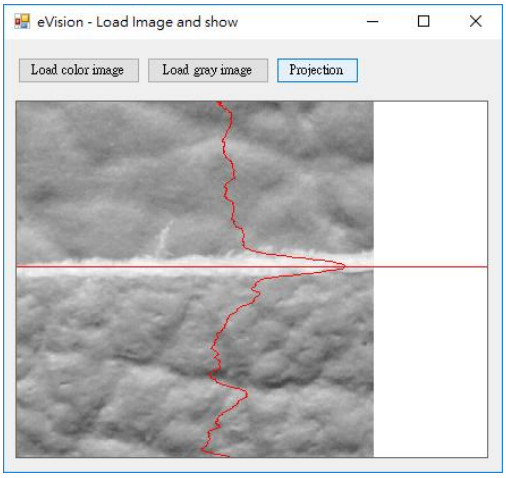

程式碼片段：

```
private void btnProjection_Click(object sender, EventArgs e)
{
   EasyImage.ProjectOnAColumn(EBW8Image1, EBW32Vector1); //作出水平投影
   //EBW32Vector1.Draw(pbImage.CreateGraphics(), pbImage.Width,pbImage.Height);
   //上面是畫出有XY軸的曲線圖，比較不合使用
   //找出最大值
   int max = 0; //紀錄最大值
   int maxi = -1; //紀錄最大值在哪個Y座標
   int v;
   Pen p = new Pen(Color.Red);
   Graphics g = pbImage.CreateGraphics(); //建立繪圖物件
   int x, y;
   for (int i = 0; i < EBW32Vector1.NumElements; i++)
   {
      v = EBW32Vector1.GetElement(i).Value;
      if ( v > max) //比現在的還大
```

```
        {
            max = v;
            maxi = i;
        }
        x = (int)(v * ScalingRatio / 256); //投影統計出來的數值是灰階的總和
        y = (int)(i * ScalingRatio);
        g.DrawLine(p, x, y, x + 1, y);
    }
    g.DrawLine(p, 0, (int)(maxi * ScalingRatio), pbImage.Width, (int)(maxi * ScalingRatio)); //在peak上畫一條水平線
    Console.WriteLine(maxi + " Max=" + max);
}
```

實作練習：

1. 請嘗試改寫程式碼，讓圖中紅色的點可以連結為一個完整曲線，如下圖所示。



## 建立一個自訂的彈性遮罩(flexible mask)

### 目標

依照下面的教學流程，你將學習如何建立一個來源圖像的彈性遮罩 (flexible mask)，這是一個任意形狀的必要區域可以用來限制未來圖像處理的範圍。

彈性遮罩 (flexible masks) 可以以任何方式建立來產生所需的雙層圖像 (bi-level image)。這裡，我們將首先載入來源圖像 (step 1)，然後將顏色反相，並進行二值化 (steps 2-3)。由此產生的圖像就是彈性遮罩 (flexible mask) 將儲存為新的圖像檔 (step 4)，這個新的圖像就是一個雙層圖像(bi-level image)。然而，要將之作為遮罩前仍然需要消除一些黑色區域，可以使用第三方的繪圖軟體來清除不想列入遮罩的區域，例如：小畫家 (Paint)。



**來源圖像**



**彈性遮罩圖像**

### Step 1: 載入來源圖像

- 從主選單中， 點擊 [EasyImage]， 然後選擇 [Arithmetic & logic]。

- 點擊 [Open] 來源圖像 0 區域的圖示， 並載入下列的圖像檔：
  EasyImage\Key1.tif。
- 保留新圖像對象的預設變數名稱， 然後點擊 [OK]。

**Step 2: 反轉圖像**

- 點擊 [New] 目標區域的圖示。
- 保留新圖像對象的預設設置， 然後點擊 [OK]。
- 在 [Operation] 下拉列表， 選擇 [Invert]， 然後點擊 [Execute]。

**Step 3: 閾值圖像**

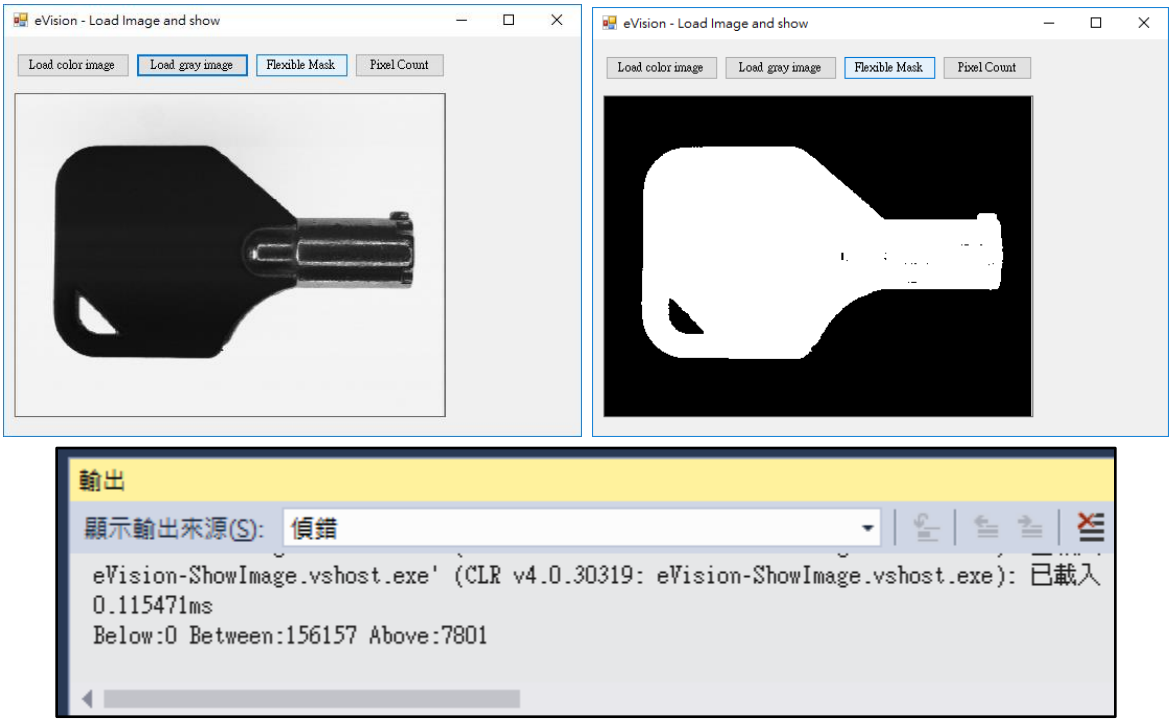
- 從主選單中， 點擊 [EasyImage]， 然後選擇 [Threshold]。
- 在來源圖像區域， 從下拉列表中選擇倒置圖像。
- 點選 [New] 目標區域的圖示。
- 保留新圖像對象的預設設置， 然後點擊 [OK]。
- 選擇絕對選項， 輸入 '46' 作為閾值，然後點擊 [Execute]。

**Step 4: 保存 flexible mask**

- 在目標圖像上點擊鼠標右鍵， 然後選擇 [Save As...]。
- 為新的 flexible mask 檔輸入文件名稱。最後， 點擊 [Save]。

**注意**：現在新的圖像是雙層圖像，然而，仍然需要消除一些不需要的黑色區域，因此，在將圖像做為一個 flexible mask 之前，您可以使用第三方軟體來清除這些不需要的區域，如：小畫家 (Paint)。

範例程式執行畫面（含下一個教學課程）




程式碼片段：

下面是反相的程式碼：

```
private void btnMask_Click(object sender, EventArgs e)
{
   DateTime sTime;
   sTime = DateTime.Now; //紀錄起始時間
   for (int i = 0; i < 100; i++)
   {
      //EBW8Image2.SetSize(512, 512);
      //// Make image black
      //EasyImage.Oper(EArithmeticLogicOperation.Copy, new EBW8(0), EBW8Image2); //建立一張黑色背景的圖，可以不用加入
      EBW8Image2.SetSize(EBW8Image1);
      EasyImage.Oper(EArithmeticLogicOperation.Invert, EBW8Image1, EBW8Image2); //反相
      //EBW8Image3.SetSize(512, 512);
      //// Make image black
      //EasyImage.Oper(EArithmeticLogicOperation.Copy, new EBW8(0), EBW8Image3);
      EBW8Image3.SetSize(EBW8Image2);
```

```
        EasyImage.Threshold(EBW8Image2, EBW8Image3, 46); //使用46作為二值化的臨界值
    }
    TimeSpan ts = DateTime.Now.Subtract(sTime); //經過的時間間隔
    Console.WriteLine(ts.TotalMilliseconds/100.0 + "ms");
    ShowImage(EBW8Image3, pbImage);
}
```

下面是Image Statistics的程式碼：

```
private void btnCount_Click(object sender, EventArgs e)
{
    //EasyImage.PixelCount(EBW8Image1, EBW8Image3, 0, 127, ref pixelsBelow, ref pixelsBetween, ref
pixelsAbove);
    EBW8 low,high; //上面的臨界值需要包裝在EBW8的型態中，才可以傳進去
    low.Value = 0;
    high.Value = 127;
    EasyImage.PixelCount(EBW8Image1, EBW8Image3, low, high, out pixelsBelow, out pixelsBetween, out
pixelsAbove);
    Console.WriteLine("Below:" + pixelsBelow + " Between:" + pixelsBetween + " Above:" + pixelsAbove);
}
```

## 使用彈性遮罩計算並統計灰階度

### 目標

依據下面說明，你將學習如何在任意形狀區域中計算並統計出灰階度。

首先載入一張來源圖像 (step 1)，接著載入遮罩圖像 (step 2)，遮罩圖像是應用在來源圖像上 (step 3)，需要將必要區域 [do-care areas] 與忽略區域 [don't-care areas] 分開，最後，灰階度的統計會計算在必要區域上 (step 4)。



**來源圖像**



**遮罩圖像**

### Step 1: 載入來源圖像

- 從主功能表中，點選 [EasyImage]，接著點選 [Image Statistics]，[Gray Scale]。
- 從來源圖像區域點選 [Open] 圖示，並載入圖像檔：**EasyImage\Key1.tif**。
- 針對新的物件使用預設的參數名稱，接著按下 [OK]。

### Step 2: 載入遮罩圖像

- 從主功能表中，點選 [Image]，接者點選 [Open...]。
- 載入圖像檔案 **EasyImage\Mask2.bmp**.
- 針對新的物件使用預設的參數名稱，接著按下 [OK]。

**Step 3: 應用遮罩圖像在來源圖像上**

- 在遮罩範圍裡的 [Gray Scale Image Pixels Statistics] 對話框，從 Mask 下拉式列表選擇遮罩圖像，來源圖像的忽略區域會在對話框中以紅色對角線來顯示，表示這個區域不會計算到灰階度統計上。

**Step 4: 計算並統計灰階度**

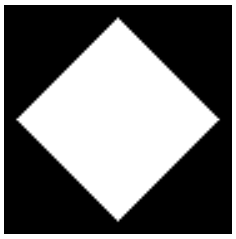
- 將 [Pixel Count] 打勾，並點選 [Execute]，結果將會不同灰階度的像素量顯示在下方。

## 使用 hit-and-miss 轉換來偵測物件的頂角

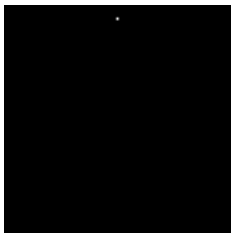
### 目標

依據下面的說明，你可以學習到如何使用 EasyImage 的 hit-and-miss 轉換去偵測出圖像的頂角。

首先，載入一張來源圖像 (step 1)，設置一個核心 (Kernal) 來表示頂角 (step 2)，之後設置一個目標圖像和簡單的執行 hit-and-miss 轉換 (step 3)。



**來源圖像**



**從來源圖像中偵測頂角 (白點)**

### Step 1: 載入來源圖像

- 從主功能表中，點選 [EasyImage]，接者點選 [Hit And Miss]。
- 從來源圖像區域點選 [Open] 圖示，選擇載入下面的圖像檔 EasyImage\Diamond.bmp。
- 針對新的圖像使用預設的參數名稱，接者按下 [OK]。

### Step 2: 設置 hit-and-miss 核心

| | -1 | 0 | 1 |
|---|---|---|---|
| -1 | DontCare | Background | DontCare |
| 0 | Background | Foreground | Background |
| 1 | Foreground | Foreground | Foreground |

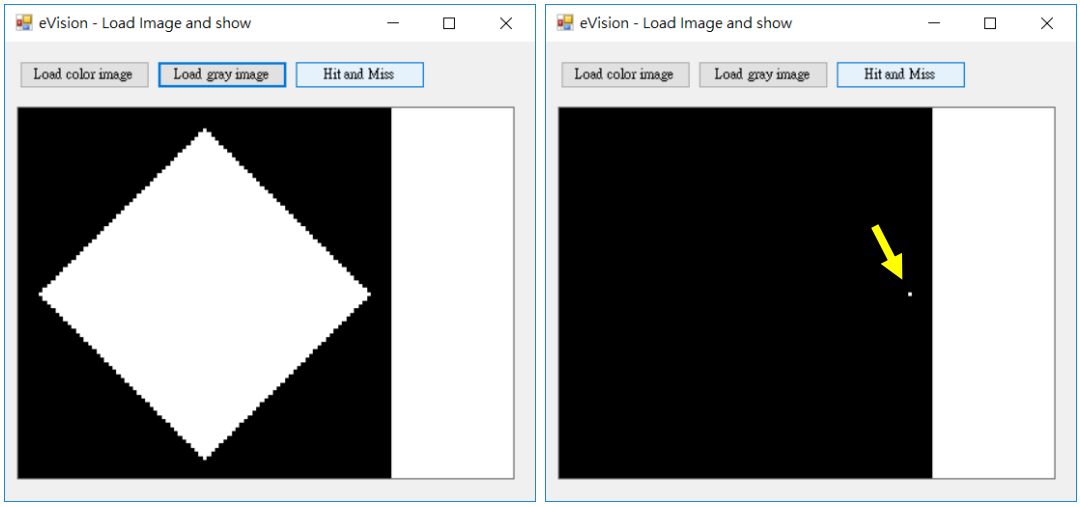
- 在 [Hit And Miss] 對話方塊中，依據以下的值設置核心：

**用於偵測最上方角落的核心設定**

### Step 3: 應用 hit-and-miss 轉換

- 從目標圖像區域點選 [New] 圖示 。
- 針對新的圖像使用預設的參數跟變數名稱，接者按下 [OK]。
- 點選 [Execute] 去執行這項操作，進行頂角(白點)偵測。
- 嘗試其他不同的核心組合去檢測不同的頂角。

範例程式執行畫面



程式碼片段：

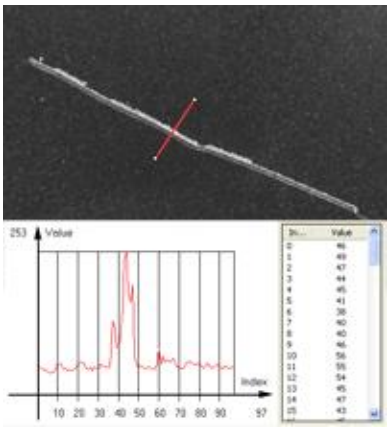
```
private void btnHit_Click(object sender, EventArgs e)
{
   //EHitAndMissKernel hitAndMissKernel11(1,1) = new EHitAndMissKernel(); // EHitAndMissKernel instance
原來的Script有問題
   EHitAndMissKernel hitAndMissKernel11 = new EHitAndMissKernel(1, 1); // EHitAndMissKernel instance
   hitAndMissKernel11.SetValue(-1, -1, EHitAndMissValue.Foreground);
   hitAndMissKernel11.SetValue(0, -1, EHitAndMissValue.Background);
   hitAndMissKernel11.SetValue(1, -1, EHitAndMissValue.DontCare);
   hitAndMissKernel11.SetValue(-1, 0, EHitAndMissValue.Foreground);
   hitAndMissKernel11.SetValue(0, 0, EHitAndMissValue.Foreground);
   hitAndMissKernel11.SetValue(1, 0, EHitAndMissValue.Background);
   hitAndMissKernel11.SetValue(-1, 1, EHitAndMissValue.Foreground);
   hitAndMissKernel11.SetValue(0, 1, EHitAndMissValue.Background);
   hitAndMissKernel11.SetValue(1, 1, EHitAndMissValue.DontCare);
   try
   {
      EBW8Image2.SetSize(EBW8Image1);
      EasyImage.HitAndMiss(EBW8Image1, EBW8Image2, hitAndMissKernel11);
      ShowImage(EBW8Image2, pbImage);
   }
   catch(Exception ex)
   {
      Console.WriteLine(ex.ToString());
   }
}
```

## 使用剖面(Profile)功能來提取向量

### 目標

按照下面的教學流程，您將學習如何使用 EasyImage 檢測刻痕。

首先， 您需要載入圖像 (step 1)，設置目標向量 (destination vector) 並檢測刻痕 (step 2)。



**可以使用剖面(Profile)檢測刻痕**

### Step 1: 載入來源圖像

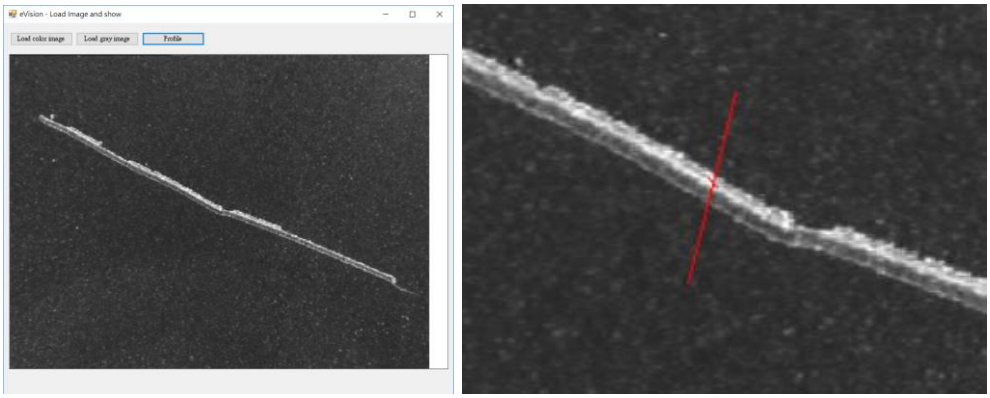
- 從主功能表中，點擊 [EasyImage]，然後點擊 [Profile]。
- 點擊來源圖像區域的 [Open] 圖示，然後載入圖像檔 **EasyImage\Plastic.tif**。
- 保留新圖像預設的變數名稱，然後點擊 [OK]。

### Step 2: 設置目標向量並檢測刻痕

- 滑鼠點擊目標向量 [Destination Vector] 區域中的 [New] 圖示。
- 選擇 BW8 的向量類型，然後點擊 [OK]。

- 圖像上出現一個紅色線段標示的剖面 (profile)。 在目標向量的視窗中，向量值對應於線段上的像素值，向量圖中的急劇變化就可以檢測出刻痕。
- 使用滑鼠來拖拉紅色線段端點的位置，並觀察向量值的變化，只要紅色線段放置在刻痕上，就會出現急劇的數值變化。

範例程式執行畫面



程式碼片段：

```
private void btnProfile_Click(object sender, EventArgs e)
{
   EasyImage.ImageToLineSegment(EBW8Image1, EBW8Vector1, 280, 260, 300, 180); //進行profile
   EBW8 max;
   EBW8 tmp;
   int maxi;
   max= EBW8Vector1.GetElement(0); //第0個當作最大值
   maxi = 0;
   for (int i=1;i < EBW8Vector1.NumElements; i++)
   {
      tmp=EBW8Vector1.GetElement(i); //第i個拿出來
      if (tmp.Value > max.Value)
      {
         max = tmp;
         maxi = i;
      }
   }
```

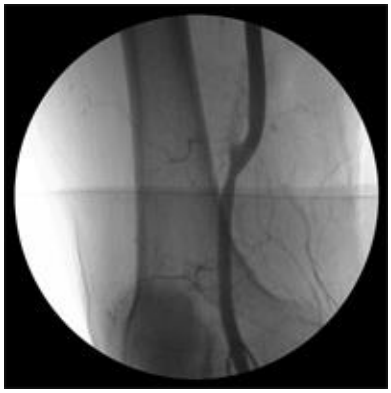
```
    Console.WriteLine("Max=" + max.Value + "(" + maxi + "/" + EBW8Vector1.NumElements+ ")");
    //計算並繪製最大值所在座標
    Point p1, p2;
    int deltaX, deltaY, targetX, targetY;
    p1 = new Point(280, 260);
    p2 = new Point(300, 180);
    if (Math.Abs(p1.X-p2.X)> Math.Abs(p1.Y - p2.Y)) //看X或Y的哪個差值大，maxi就是需要的deltaX或者deltaY
    {
        deltaX = maxi;
        deltaY = (int)(maxi * (p2.Y - p1.Y) / Math.Abs(p1.X - p2.X));
    }
    else
    {
        deltaX = (int)(maxi * (p2.X - p1.X) / Math.Abs(p1.Y - p2.Y));
        deltaY = maxi; //這裡還要判斷正負號問題
    }
    if (p1.X < p2.X) //因為最大值是在兩點中間，所以要判斷兩點座標的位置，然後進行加或者減
        targetX = (int)((p1.X + deltaX) * ScalingRatio);
    else
        targetX = (int)((p1.X - deltaX) * ScalingRatio);
    if (p1.Y < p2.Y)
        targetY = (int)((p1.Y + deltaY) * ScalingRatio);
    else
        targetY = (int)((p1.Y - deltaY) * ScalingRatio);
    Graphics g; //繪製上去
    g = pbImage.CreateGraphics();
    g.DrawLine(new Pen(Color.Red), (int)(p1.X * ScalingRatio), (int)(p1.Y * ScalingRatio), (int)(p2.X *
ScalingRatio), (int)(p2.Y * ScalingRatio));
    g.DrawLine(new Pen(Color.Red), targetX-2, targetY-2, targetX+2, targetY+2);
}
```

## 增強 X 光圖像

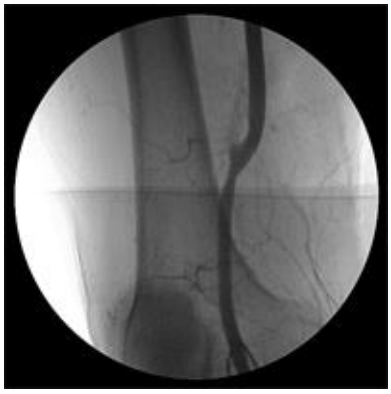
### 目標

按照下面的教學流程，您將學習如何使用 EasyImage 來增強 X 射線圖像。

您需要首先載入圖像 (step 1)， 然後定義卷積 (convolution) 參數以增強圖像 (step 2)。



**來源圖像**



**運用預先定義和使用者自訂的卷積核心來增強圖像效果**

### Step 1: 載入來源圖像

- 在主功能表中，點擊 [EasyImage]， 然後點擊 [Convolution]。
- 點擊來源圖像區域的 [Open] 圖示，然後載入圖檔 **EasyImage\XRay.bmp**。
- 保留新圖像的預設變數名稱，然後點擊 [OK]。
- 點擊目標圖像區域的 [New] 圖示以建立新的目標圖像。

- 保留預設的變數名稱然後點擊 [OK]。

### Step 2: 設置卷積參數 (convolution parameters)

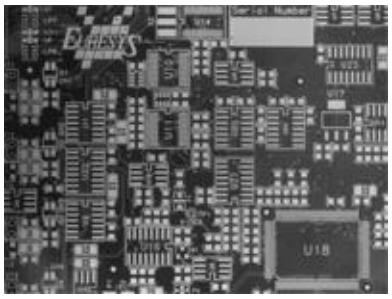
- 從 [Predefined kernels] (預定核心) 下拉列表中選擇 [Highpass2] (高通濾波)，然後點擊 [Execute]，目標圖像不再模糊但成效並不好，因為該濾波將來源圖像中的雜訊也凸顯出來，因此，我們需要自建一個新的卷積核心，採用更軟的高通濾波。
- 點擊 [User defined kernels] (使用者自訂核心)下拉清單旁的 [New] 圖示。
- 保留預設尺寸 (3x3) 和變數名稱，點擊 [OK]。
- 從左到右和從上到下輸入以下核心數值：-1, -1, -1; -1, 15, -1; -1, -1, -1，然後點擊 [Apply] 來套用此核心。
- 點擊 Convolution 對話框中的 [Execute] 以執行此操作，現在圖像看起來更清晰了。

## 修正不均勻亮度

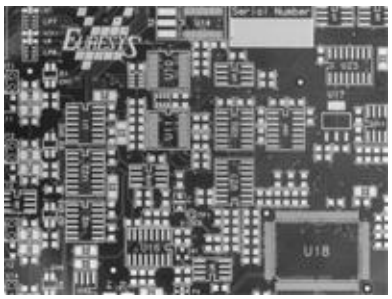
### 目標

按照下面的教學流程，您將學習如何使用 EasyImage 來修正圖像中的不均勻亮度。

首先，您需要載入圖像 (step 1)，載入燈光參考圖像 (step 2)，然後再執行校正 (step 3)。



**原本的圖像，光照不均勻**



**修正後的圖像**

### Step 1：載入來源圖像

- 從主功能表中，點擊 [EasyImage]，然後點擊 [Uniformize]。
- 點擊來源圖像區域的 [Open] 圖示，然後載入圖像檔：**EasyImage\Board (original).tif**。
- 保留新圖像的預設變數名稱，然後點擊 [OK]。

### Step 2：載入參考圖像

- 點擊 [Light Reference] (亮度參照)區域的 [Open] 圖示，並載入圖像檔：**EasyImage\Board (light reference).tif**。 為了獲得光照參考圖像，我們使用了與電路板 (original image) 相同條件下照亮的白色屏幕。
- 保留新圖像的預設變數名稱，然後點擊 [OK]。

**Step 3：執行修正**

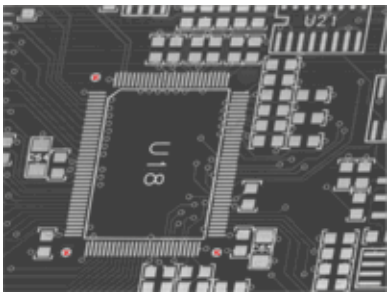
- 點擊 [Destination Image] 區域中的 [New] 圖示以建立新的目標圖像。
- 保留預設值，然後點擊 [OK]。
- 點擊 [Execute] 執行操作。
- 在來源圖像和目標圖像中，分別右鍵點擊並從快顯功能表中選擇 [3D Rendering]。
- 在兩個新的 3D 視窗中，點擊並移動鼠游標旋轉視圖，比較來源與目標的 3D 圖像，就可以清楚看到經過亮度修正過後的圖像有比較平均的 3D 結果。

## 修正剪切效應(shear effect)

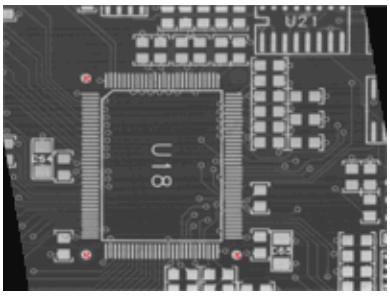
### 目標

依照下面的教學流程，你將學習如何使用 EasyImage 去校正圖像中的剪切效應，以下的圖像是採用線掃描相機 (line-scan camrea)，由於相機感測器發生錯位，造成所謂的剪切效應 (shear effect)。

首先，您要先載入圖像 (step 1)，然後新建一個目標圖像 (step 2)，然後設定軸心 (pivot) 參數去修正 (step 3)。




**原圖像含有剪切效應**




**修正後的圖像**

### Step 1: 載入來源圖像

- 從主功能表中，點選 [EasyImage]，接著點選 [Register]。
- 點選 [Open] 圖示，然後選擇載入下面的圖像檔: **EasyImage\Shear.tif**。
- 針對新的圖像使用預設的參數名稱，接著按下 [OK]，圖像中將會出現 3 個軸點 (pivots)。

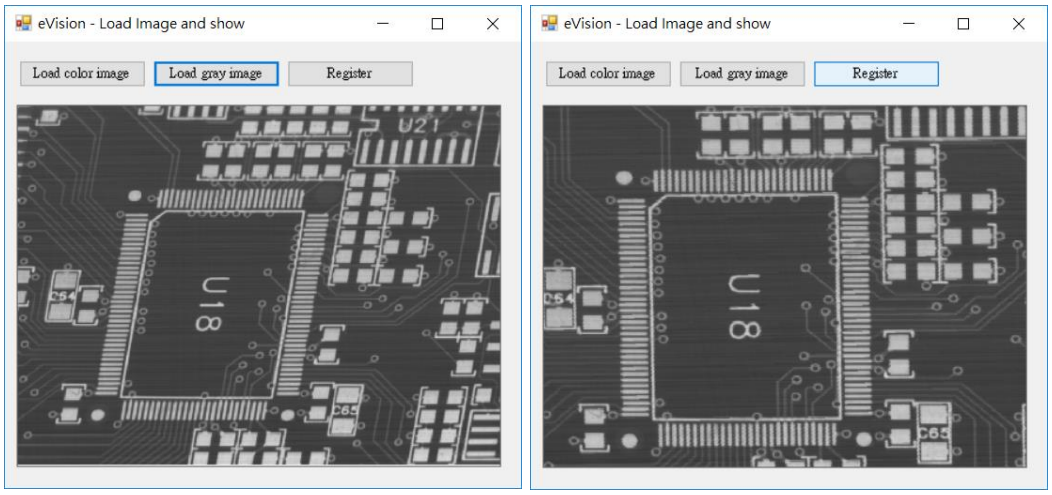
**Step 2: 建立目標圖像**

- 在目標圖像區域中點擊 [New] 圖示。
- 輸入'768'和'576'作為圖像寬度和高度，然後點擊 [OK] 接受預設名稱，圖像中同樣出現 3 個軸點。

**Step 3: 設定軸心參數**

- 在來源圖像中，使用滑鼠游標，拖移每個軸心到基準標記上 (圖像上面 U18 區域周圍的點都可以做為基準標記)，注意這些來源軸點的座標，在註冊 (Register) 對話框中會同步更新。
- 為了修正圖像，請輸入以下目標圖像的軸點座標：
  - X0: 170
  - Y0: 495
  - X1: 470
  - Y1: 495
  - X2: 170
  - Y2: 144
- 點擊 [Execute] 執行操作。

## 範例程式執行畫面




## 程式碼片段：

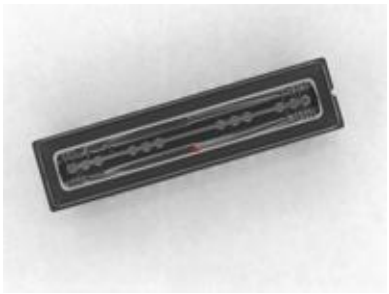
```
private void btnRegister_Click(object sender, EventArgs e)
{
    EBW8Image2.SetSize(768, 576);
    // Make image black
    //EasyImage.Oper(EArithmeticLogicOperation.Copy, new EBW8(0), EBW8Image2);
    EasyImage.Register(EBW8Image1, EBW8Image2, 158.50f, 458.50f, 399.50f, 460.50f, 213.50f, 167.50f,
170.00f, 495.00f, 470.00f, 495.00f, 170.00f, 144.00f, 0);
    ShowImage(EBW8Image2, pbImage);
}
```

## 修正偏斜效應(skew effect)

### 目標

依照下面的教學流程，你將來會學到如何使用 EasyImage 來修正有偏斜效應 (skew effect) 的圖像。

您需要首先載入圖像 (step 1)， 建立一個目標圖像 (step 2)， 然後設置一個正確的角度來執行圖像修正 (step 3)。



**具有偏斜效應的來源圖像**



**修正後的圖像**

### Step 1: 下載來源圖像

- 主選單，點 [EasyImage]， 然後點選 [Scale and Rotate]。
- 點 [Open]來源圖像區域的圖示， 載入圖像檔案路徑：**EasyImage\CCD.tif**。
- 保留新圖像預設的變數名稱，然後點選 [OK]。

### Step 2: 建立一個新圖像

- 點選目標圖像區域中的 [New] 圖示。

- 輸入 '768' 和 '576' 作為圖像的寬和高，然後使用預設的變數名稱，並點選 [OK] 。

**Step 3: 設定一個正確角度**

- 選 [Rotate] 這個選項，然後在角度 (Deg) 區域中輸入 '-16.17'。 (要測量此旋轉角度，請參閱 EasyGauge「**量測物件的旋轉角度**」單元。)
- 從內插位元數 (Interpolation bits) 的下拉清單中選擇 8 位元，可以獲得更好的結果。
- 點選 [Execute] 以執行此操作。

圖像的色彩分割

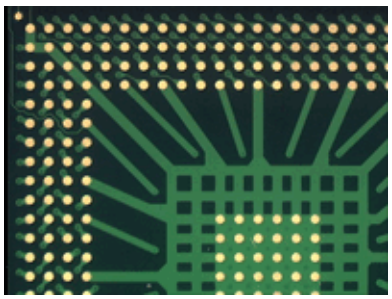
進行色彩分割

## 圖像的色彩分割

### 目標

依照下面的教學流程，您可以學到如何使用 EasyColor 將來源圖像的顏色進行分割，方法是為當前顏色系統的每個顏色分量設置一個閥值。 例如，要取得印刷電路板 (PCB: Printed Circuit Board) 上的焊盤 (solder pads)，可以根據金色像素的色相 (Hue) 執行顏色分割，對亮度(Lightness or Brightness)和飽和度(Saturation)進行顏色分割，以消除表面顏色與亮度的影響。

首先，需要載入彩色的來源圖像，建立一個目標圖像和顏色查找表(lookup table) (steps 1-3)。 接著，設置顏色系統並調整每個色彩成份的容忍值，以獲得一個滿意的焊盤分割結果 (step 4)。



**來源圖像**



**顏色分割後的圖像**

**Step 1: 載入來源圖像**

- 從主功能表中，點選 [EasyColor]，然後點選 [Threshold]。
- 點選來源圖像區域的 [Open] 圖示，並載入下面的圖像檔：EasyColor\BGA Substrate Color.jpg。
- 針對新的圖像使用預設的變數名稱，然後點選 [OK]。
- 停用預覽模式 [Preview Mode]，才能查看來源圖像。

**Step 2: 建立一個目標圖像**

- 點選目標圖像區域的 [New] 圖示。
- 針對新圖像使用預設的變數名稱，然後點選 [OK]。

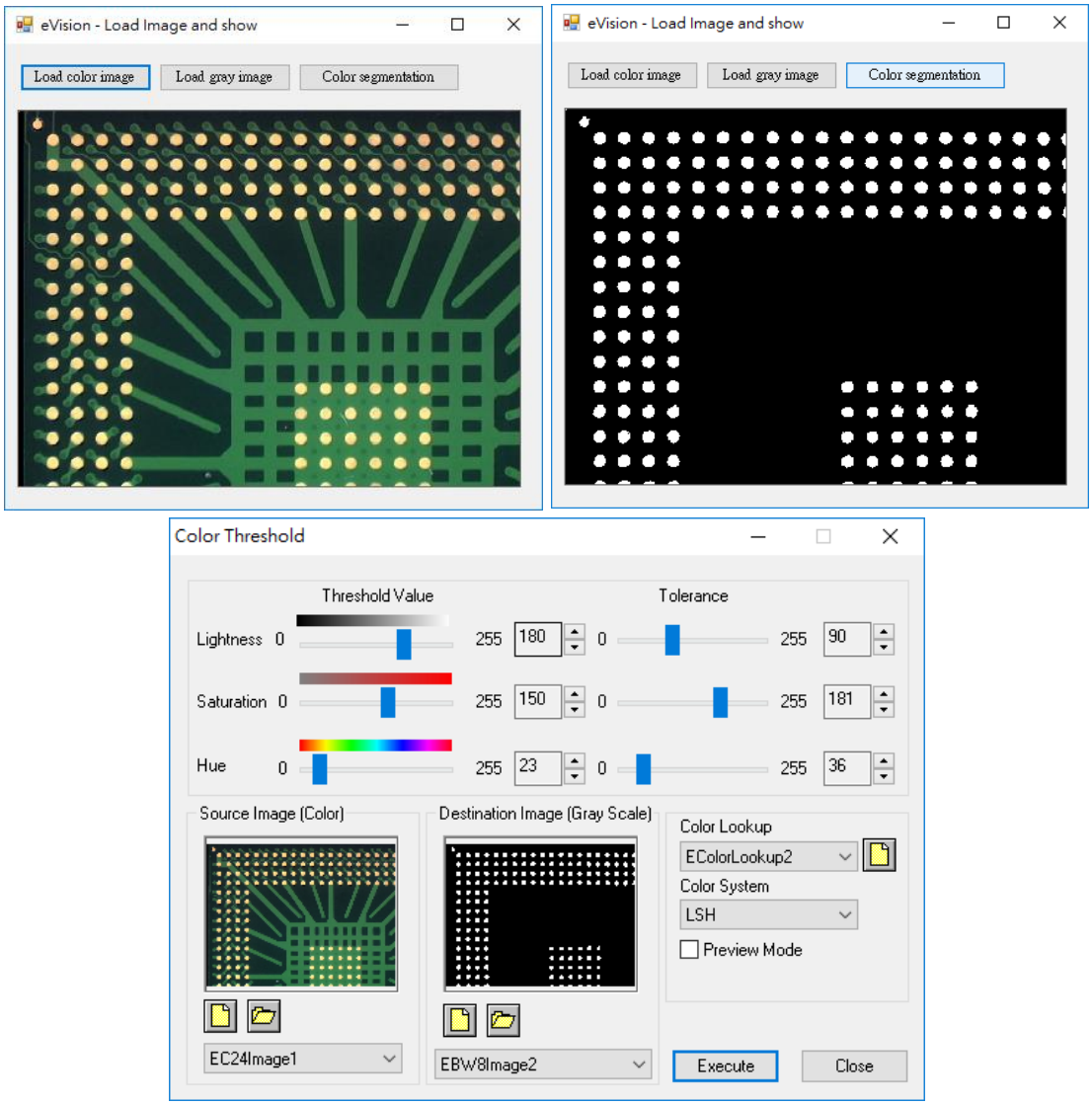
**Step 3: 建立一個顏色查找表 (lookup table)**

- 點選顏色查找區域的 [New] 圖示。
- 使用預設查找表的變數名稱，然後點選 [OK]。

**Step 4: 執行顏色分割**

- 從色彩系統 [Color System] 的下拉清單中選擇 LSH。
- 在來源圖像中，點選焊盤上的某個像素，該像素的亮度，飽和度和色調值會更新到色彩臨界值 [Color Threshold] 的對話框中。
- 調整亮度和飽和度的容忍值以放大像素閾值的範圍，直到在目標圖像中獲得一個滿意的分割結果。

範例程式執行畫面




(上面**臨界值±容忍值**之後的最大值與最小值設定到程式碼中)

程式碼片段：

```
private void btnSegment_Click(object sender, EventArgs e)
{
   //例外處理
   try
   {
      EColorLookup1.ConvertFromRgb(EColorSystem.Lsh); //("EColorSystem_Lsh");
      EBW8Image1.SetSize(EC24Image1.Width, EC24Image1.Height); //需要設定大小，不然下一行會出現例外
```

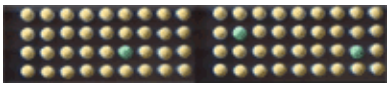
```
        EasyImage.Threshold(EC24Image1, EBW8Image1, new EC24(90, 0, 0), new EC24(255, 255, 59),
EColorLookup1);
        ShowImage(EBW8Image1, pbImage);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

## 進行色彩分割

### 目標

依照下面的教學流程，你會學到如何用 EasyImage 來完成色彩的分割 (segmentation)。

你需要先載入圖像 (step 1)，接著建立色彩查找表 (look-up table) (step 2)，就完成個色彩分割了。



**來源圖像**



**分割後的圖像**

### Step 1: 載入來源圖像

- 從主目錄，點擊 [EasyColor]，接著按 [Threshold]。
- 點擊在 [Source Image] 區域的 [Open] 圖示，然後載入圖檔：**EasyColor\Pills.tif**。
- 為新的圖檔保留預設的變數名稱，然後點擊 [OK]。
- 將 [Preview Mode] 的選項取消 ，才能看到原本的來源圖像。

### Step 2: 建立色彩查找表(look-up table)

- 點擊 [Color Lookup] 的 [New] 圖示。
- 為新的色彩查找表保留預設的變數名稱，然後點擊 [OK]。

### Step 3: 完成色彩分割

- 選擇在 [Color System]下拉列表中的 LSH。

- 在來源圖像中，點擊綠色圓標的中心。在 [Color Threshold] 選擇框中更新像素亮度 (Lightness)，飽和度 (Saturation) 和色調值 (Hue)。
- 將亮度容差增加到 120，將飽和度容差增加到 50。
- 啟用 [預覽模式] 框框查看分割結果。 必要的話，請重新點擊綠色圓標以改善結果。
- 點擊在 [Destination Image] 的 [New] 圖示。
- 為新的圖檔保留預設的變數名稱，然後點擊 [OK]。
- 新圖像會被自動地設置前面步驟所設定好的閾值，點擊 [Execute] 將顏色分割對應的程式碼插入到腳本視窗中。

圖像分割後移除不重要的物體

使用最小 - 最大參考檢測圖像之間的差異

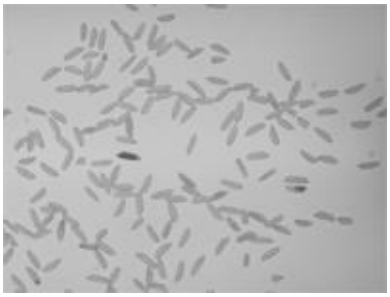
使用彈性遮罩來偵測印刷錯誤

## 圖像分割後移除不重要的物體

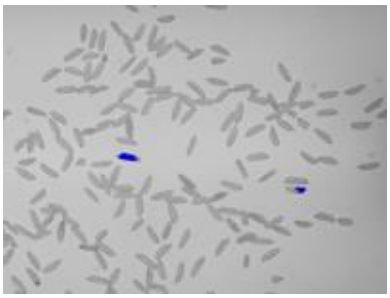
### 目標

依照下面的教學流程，許多正常的米粒中（大部分是淡色）您將學習如何使用 EasyObject 來檢測出不良的米粒（大部分是黑暗色）。

首先，您需要載入來源圖像 (step 1)。然後，您將根據閾值執行圖像分割 (step 2)。所有檢測到的物體都很暗，但有些太小而不重要。因此，您將設置最小的物體區域（像素數量），並刪除最小的物體 (step 3)。最後，你只會得到真正代表不良米粒的物體。



**來源圖像**



**偵測到的不良米粒**

### Step 1: 載入來源圖像

- 從主功能表中，點選 [EasyObject]，然後點選 [New EasyObject Tool]。
- 使用預設變數名稱，然後點選 [OK].
- 在 [Encoder] 的頁面中，點選 [Source Image] 的 [Open] 圖示，然後載入圖像檔：**EasyObject\Rice.jpg**。

- 使用預設的變數名稱，然後點選 [OK]。

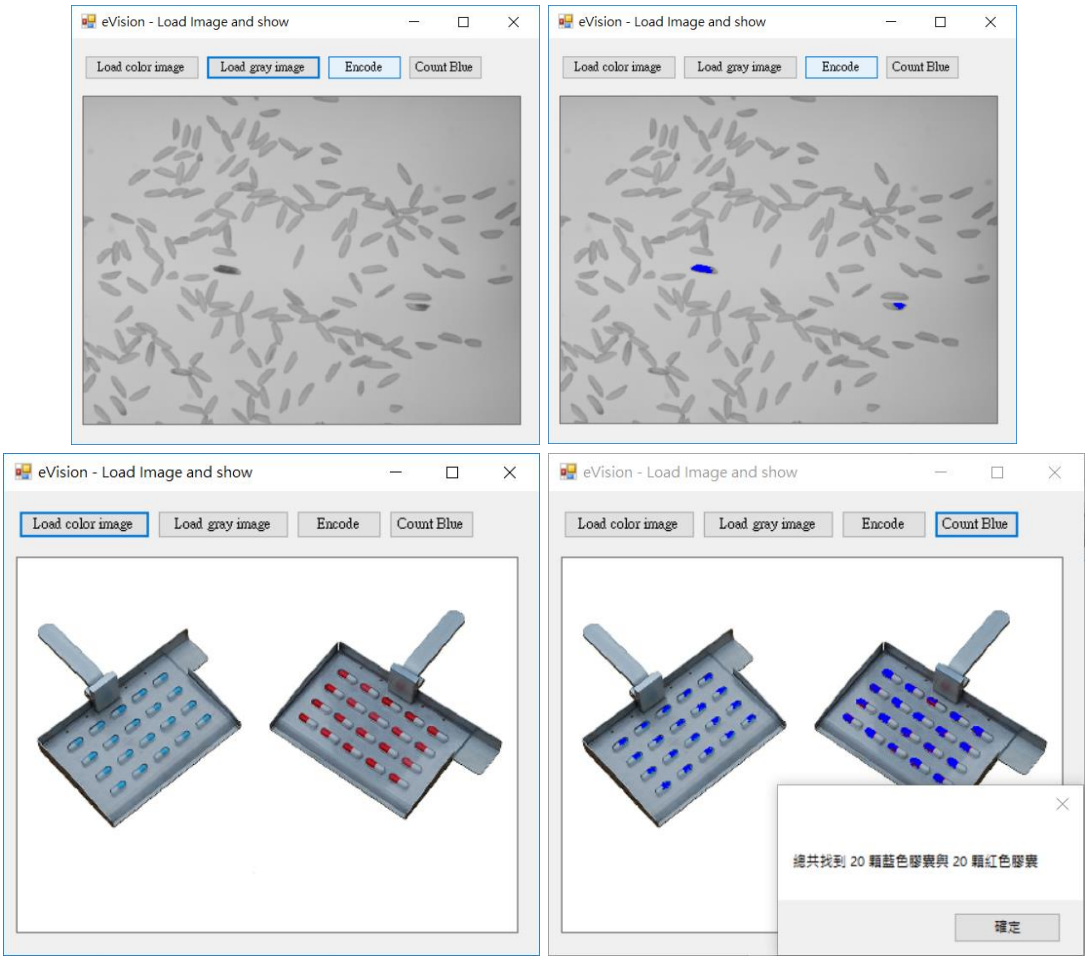
**Step 2: 執行圖像分割**

- 在 [Encoder] 頁面中，選擇 [Black Layer] 複選項，並取消 [White Layer] 的複選項。
- 點選 [Threshold] 右邊的 [...] 按鈕。在閾值對話框中，選擇 [Absolute]，輸入 '115'，然後點選 [OK]。
- 點選 [Encode] 來進行黑暗色物體的檢測，在圖像中每個被檢測到物體都會使用不同的顏色繪製。
- 點選 [Results] 來顯示所有檢測到的物體清單，點擊圖像上的某個物體會將其明顯標示在清單中，反之亦然，點選清單上某個物體時，也會高亮度顯示在圖像上。

**Step 3: 刪除最小的物體**

- 在物體清單中，點選 [Columns]。
- 勾選 [Area] 複選項，然後點選 [OK]，在清單的欄位中，出現一個新的 Area 欄位，顯示每個物體的區域大小。
- 點選 'Area' 的標題可將所有物體進行排序，有許多微小物體 (area<100) 可以將之視為雜訊 (noise)。
- 在 [Selection] 頁面中，從 [Feature] 下拉清單中選擇 'Area'，接著從 [Mode] 下拉清單中選擇 'Less'，然後在 [Threshold] 中輸入 '100'。
- 點選 [Remove] 按鍵，將所有面積小於 100 像素的物體都從清單中刪除，剩下的兩個物體就是期望偵測出的不良米粒。

## 範例程式執行畫面




## 程式碼片段：

```
private void btnEncode_Click(object sender, EventArgs e)
{
   //codedImage1ObjectSelection.FeretAngle = 0.00f; //控制object要標記的邊界框
   codedImage1Encoder.GrayscaleSingleThresholdSegmenter.BlackLayerEncoded = true; //選擇black layer
   codedImage1Encoder.GrayscaleSingleThresholdSegmenter.WhiteLayerEncoded = false;//取消white layer
   codedImage1Encoder.SegmentationMethod = ESegmentationMethod.GrayscaleSingleThreshold;
   codedImage1Encoder.GrayscaleSingleThresholdSegmenter.Mode = EGrayscaleSingleThreshold.Absolute;
   codedImage1Encoder.GrayscaleSingleThresholdSegmenter.AbsoluteThreshold = 115;
   codedImage1Encoder.Encode(EBW8Image1, codedImage1); //產生所有的物件，把原圖上面的物件放入
codedImage1
   codedImage1ObjectSelection.Clear(); //ObjectSelection物件是用來過濾物件屬性的元件
   codedImage1ObjectSelection.AddObjects(codedImage1);
```

```
    codedImage1ObjectSelection.AttachedImage = EBW8Image1; //指定附加的原圖為EBW8Image1
    codedImage1ObjectSelection.RemoveUsingUnsignedIntegerFeature(EFeature.Area, 100,
ESingleThresholdMode.Less);
    Console.WriteLine(codedImage1ObjectSelection.ElementCount); //顯示為2
    //codedImage1.Draw(pbImage.CreateGraphics(), ScalingRatio); //這樣會全部繪製
    codedImage1.Draw(pbImage.CreateGraphics(), codedImage1ObjectSelection, ScalingRatio); //這樣只繪製過
濾出來的結果
}
```

另一個計算藍色與紅色膠囊個數的程式碼：

```
private void btnCountBlue_Click(object sender, EventArgs e)
{
    //EBW8Image2.SetSize(512, 512);
    // Make image black
    int blue, red;
    //計算藍色膠囊
    EBW8Image1.SetSize(EC24Image1);
    EColorLookup1.ConvertFromRgb(EColorSystem.Lsh); //本來是這樣的參數("EColorSystem_Lsh");
    EasyImage.Threshold(EC24Image1, EBW8Image1, new EC24(96, 69, 126), new EC24(140, 255, 158),
EColorLookup1);
    //codedImage1ObjectSelection.FeretAngle = 0.00f;
    codedImage1Encoder.GrayscaleSingleThresholdSegmenter.Mode =
EGrayscaleSingleThreshold.MinResidue;
    codedImage1Encoder.Encode(EBW8Image1, codedImage1);
    codedImage1ObjectSelection.Clear();
    codedImage1ObjectSelection.AddObjects(codedImage1);
    codedImage1ObjectSelection.AttachedImage = EBW8Image1;
    codedImage1ObjectSelection.RemoveUsingUnsignedIntegerFeature(EFeature.Area, 40,
ESingleThresholdMode.Less);
    codedImage1.Draw(pbImage.CreateGraphics(), codedImage1ObjectSelection, ScalingRatio); //這樣只繪製過
濾出來的結果
    blue = codedImage1ObjectSelection.ElementCount;

    //計算紅色膠囊
    EasyImage.Threshold(EC24Image1, EBW8Image1, new EC24(24, 0, 218), new EC24(178, 255, 255),
EColorLookup1);
    codedImage1Encoder.GrayscaleSingleThresholdSegmenter.Mode =
EGrayscaleSingleThreshold.MinResidue;
    codedImage1Encoder.Encode(EBW8Image1, codedImage1);
```

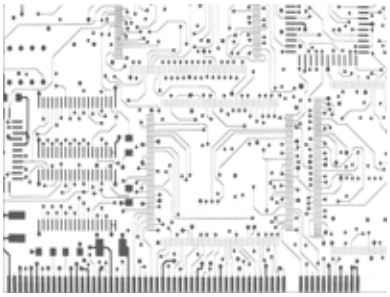
```
    codedImage1ObjectSelection.Clear();
    codedImage1ObjectSelection.AddObjects(codedImage1);
    codedImage1ObjectSelection.AttachedImage = EBW8Image1;
    codedImage1ObjectSelection.RemoveUsingUnsignedIntegerFeature(EFeature.Area, 40,
ESingleThresholdMode.Less);
    codedImage1.Draw(pbImage.CreateGraphics(), codedImage1ObjectSelection, ScalingRatio); //這樣只繪製過
濾出來的結果
    red = codedImage1ObjectSelection.ElementCount;

    MessageBox.Show("總共找到 " + blue + " 顆藍色膠囊與 "+ red + " 顆紅色膠囊");
}
```

## 使用最小 - 最大參考檢測圖像之間的差異

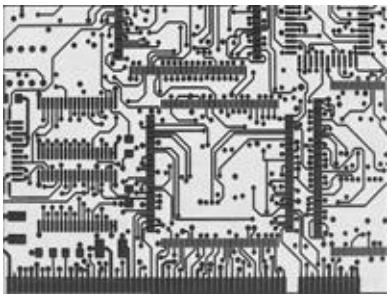
### 目標

依照下面的教學流程，您將學習如何使用 EasyObject 來比對圖像，在這個實例中，我們將檢查 PCB (Printed Circuit Board: 印刷電路板) 底片 (film) 的品質。

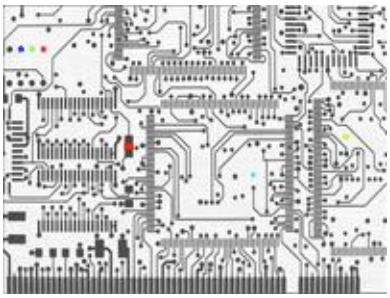
首先，你載入一個來源圖像 (step 1)，從中將構建兩個參考圖像 (最小和最大) (step 2)。然後，再載入另一張要檢查的圖像 (step 3)，並與最小和最大參考圖像進行比較 (step 4)，兩張圖像的差異將被檢測出來。



**高閥值參考圖像**



**低閥值參考圖像**



**檢測到的差異結果**

**Step 1: 載入來源圖像**

- 從主功能表中，點選 [EasyObject]，接者點選 [Make Min Max]。
- 點選來源圖像區域的 [Open] 圖示，然後載入下面的圖像檔：**EasyObject\FilmOk.png。**
- 輸入'filmOk'作為新圖像的名稱，然後點選 [OK]。

**Step 2: 建立最小和最大參考圖像**

- 點選 [Execute]，以來源圖像分別建立最小和最大的參考圖像。
  - 'filmOkMax'是通過擴展 'filmOk' 指定的次數 (幾何範圍: geometric margin) 並向每個像素加上指定的常數 (灰階度範圍: gray level margin)來計算的，反之，通過對 'filmOk' 進行相同的次數並將相同的常數減去每個像素來計算 'filmOkMin'。
  - 幾何範圍主要是針對欲檢測的圖像和參考圖像之間的位置容忍值。
  - 灰階值範圍主要是針對欲檢測的圖像和參考圖像之間光照變化的容忍值。

**Step 3: 載入要檢測的圖像**

- 從主功能表中，點選 [EasyObject]，接著點選 [New EasyObject Tool]。
- 保留新物件的預設變數名稱，然後點選 [OK]。
- 在 [Encoder] 頁面中，點選 [Open] 的圖示，然後選擇載入下面的圖像檔名:**EasyObject\FilmBad.png。**
- 輸入'filmBad'作為新圖像的名稱，然後點擊 [OK]。

**Step 4: 比較原圖像和參考圖像**

- 在 [Encoder] 頁面中的 [Segmentation] 區域，從下拉清單中點選 'ImageRange'。

- 將 [White Layer] 取消勾選，再把 [Black Layer] 打勾。
- 點選 [High Image] 的 [...] 按鈕，在下拉列表中選擇 'filmOkMax'，然後點選 [OK]。
- 點選 [Low Image] 的 [...] 按鈕，在下拉列表中選擇 'filmOkMin'，然後點選 [OK]。
- 點選 [Encode]，在圖像中顯示出了八個差異之處。
- 點選 [Results] 可以獲得有關檢測結果的相關訊息，可以進一步使用 EasyObject 的物體特徵 (Feature) 選擇和排序功能對它們進行進一步過濾和分析。

## 使用遮罩來檢測印刷錯誤

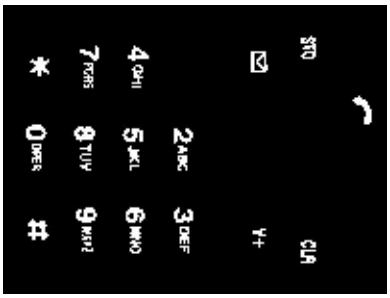
### 目標

依照下面的教學流程，你可以學習到如何使用自訂的彈性遮罩 (flexible mask) 來鎖定與搜尋圖像中的特定區域。

首先，你載入一個來源圖像 (step 1)，一個以及遮罩圖像 (step 2)，有時候，你會需要在來源圖像中設定需要處理的必要區域 (do-care area) 或者不需要處理的忽略區域 (don't care area)。然後，檢測功能只會針對必要區域來進行運作 (step 3)。



**來源圖像**



**遮罩圖像**



**檢測結果**

**Step 1: 載入來源圖像**

- 從主功能表中，點選 [EasyObject]，接者點選 [New EasyObject Tool]。
- 針對新的物件使用預設的參數名稱，接著按下 [OK]。
- 在 [Encoder] 頁面中，點選 [Open] 圖示，然後選擇載入下面的圖像檔名：**EasyObject\Mobile3.jpg**
- 針對新的圖像使用預設的參數名稱，接著按下 [OK]。

**Step 2: 載入遮罩圖像**

- 請在 [Encoder] 頁面中，點選 [Open] 圖示，並且選擇載入下面檔名的遮罩圖像： **EasyObject\MobileMask.bmp**。
- 輸入 'mask' 作為新圖像的變數名稱，接著按下 [OK]。在 [Encoder] 頁面中的 [Mask] 區域中，注意一下遮罩圖像已經從下拉式清單中被選取，該遮罩圖像已經自動被應用到來源圖像上面，由於變數名稱包含 'mask' 這個字並且已經從編碼圖像 (coded image) 對話方塊中被載入。對話方塊的來源圖像預覽中，將忽略區域以紅色交叉線來標示，表示當來源圖像進行編碼時將不會考慮這些區域。

**Step 3: 進行圖像偵測**

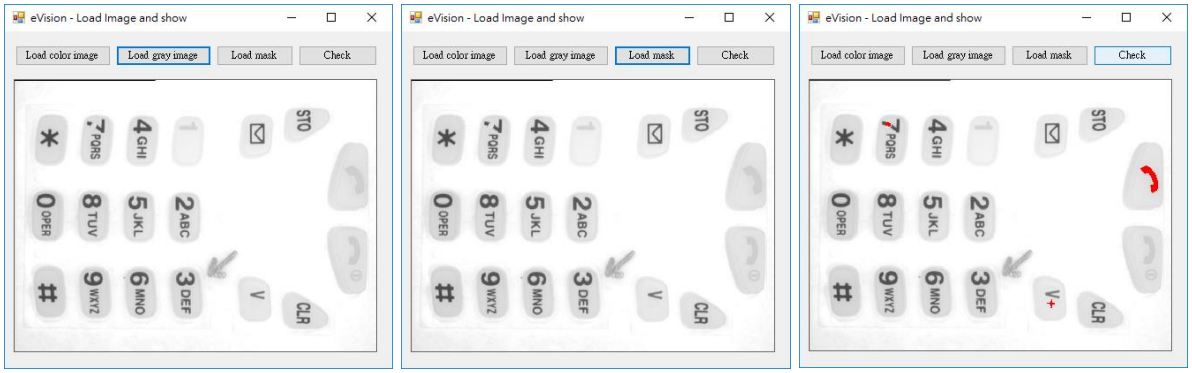
- 在 [Encoder] 頁面中的 [Segmentation] 區域，點選 [...] 按鍵來顯示臨界值 (threshold ) 的對話方塊。
- 選擇絕對方式 (Absolute) 並輸入數值 202 以作為臨界值，接著按下 [OK] 後關閉對話方塊。
- 按下 [Encode] 來找出所有的物件 (只針對必要區域 do-care areas). 在來源圖像中，每個物件都會以不同顏色來進行標示，這裡可以清楚地看到圖像上面有三個列印錯誤：
  - 數字 '7' 只有部分被印出來

- '+' 號不見了
- 手機符號的印刷色調太淺

▪ 點選 [Results] 可以顯示每個物件的統計資料。

- 在清單中選擇某個物件可以在圖像中凸顯這個物件。
- 選擇 [Columns] 以及 [Drawing] 可以顯示更多的選項 。

範例程式執行畫面




(載入原始圖像) (載入遮罩) (執行結果)

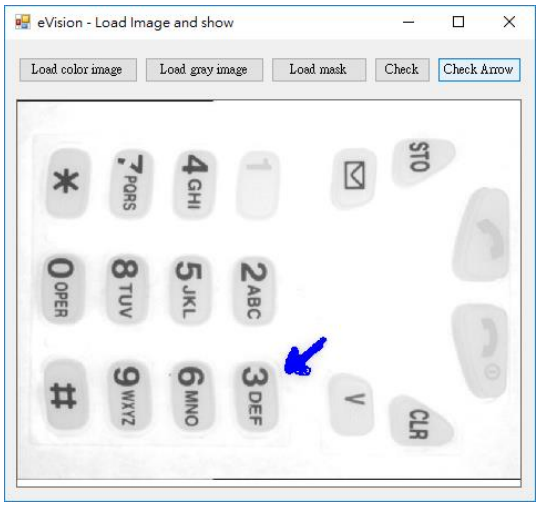
程式碼片段：

```
ECodedImage2 codedImage1 = new ECodedImage2(); // ECodedImage2 instance
EImageEncoder codedImage1Encoder = new EImageEncoder(); // EImageEncoder instance
EImageBW8 mask = new EImageBW8(); //eVision的灰階圖像物件
```

```
private void btnCheck_Click(object sender, EventArgs e)
{
   codedImage1Encoder.GrayscaleSingleThresholdSegmenter.Mode = EGrayscaleSingleThreshold.Absolute;
   codedImage1Encoder.GrayscaleSingleThresholdSegmenter.AbsoluteThreshold = 202; //設定臨界值
   codedImage1Encoder.Encode(EBW8Image1, mask, codedImage1); //進行編碼，找出物件
   Console.WriteLine(codedImage1.GetObjCount()); //取得物件的數量
   codedImage1.Draw(pbImage.CreateGraphics(), ScalingRatio); //進行繪製
}
```

實作練習：

1. 請嘗試針對範例圖像中的箭頭物件，設計遮罩並且偵測標示出來，實現下圖的結果。



## EasyMatch

學習樣本(pattern)及新增 EasyMatch 模型(model)檔案

根據模型(model)檔案來匹配樣本(pattern)

在 ROI 中根據模型(model)來匹配樣本(pattern)

透過使用忽略區域(don't care areas)來改善樣本的匹配分數

## 學習樣本(pattern)及新增 EasyMatch 模型(model)檔案

### 目標

依照下面的教學流程，您將學習到如何使用 EasyMatch 將參考圖像學習 (Learn) 成一個模型 (model)，並輸出儲存成 EasyMatch 模型檔。

您需要先載入一個參考圖像作為樣本 (step 1)，接著您將對它做學習並產生一個參考模型 (step 2)，最後您可以將此模型儲存為 EasyMatch 模型檔 (step 3)。



**參考圖像**

### Step 1: 載入參考圖像

- 從主功能表中點選 [EasyMatch]，接著點選 [New Match Tool]。
- 針對新的 EMatcher 物件使用預設變數名稱，並點選 [OK]。
- 在 [Learning] 標籤中點選 [Open] 圖示，接著載入圖像檔：**EasyMatch\FrameModel.tif**。
- 使用預設的變數名稱來命名此圖像物件，並點選 [OK]。

### Step 2: 學習參考圖像

- 在 [Learning] 標籤中點選 [Learn] 以進行學習並建立模型。

### Step 3: 儲存模型檔

- 在 [Learning] 標籤中點擊 [Save As...] (另存新檔) 按鈕，接著使用者輸入模型檔的名稱，其附檔名將會是 .mch，最後點選 [Save] 完成此操作。

## 根據模型(model)檔案來匹配樣本(pattern)

### 目標

按照下面的教學流程，您將學習如何使用 EasyMatch 來載入 EasyMatch 模型檔案，並搜尋圖像中出現的樣本 (pattern)。

你需要先載入模型檔案，並從模型中搜尋來源圖像 (step 1-2)，這整個樣本的匹配是全自動的 (step 3)。



**模型的變化會自動顯示**

### Step 1: 載入模型檔案

- 從主功能表中點選 [EasyMatch]，接著點選 [New Match Tool]。
- 針對新的物件使用預設的變數名稱，並點選 [OK]。
- 在 [Learning] 頁面中，點選 [Load...] 來開啟模型檔案：**EasyMatch\Switch.MCH**。該模型檔包含所有有關我們所要搜索樣本之必要資訊。

### Step 2: 載入來源圖像

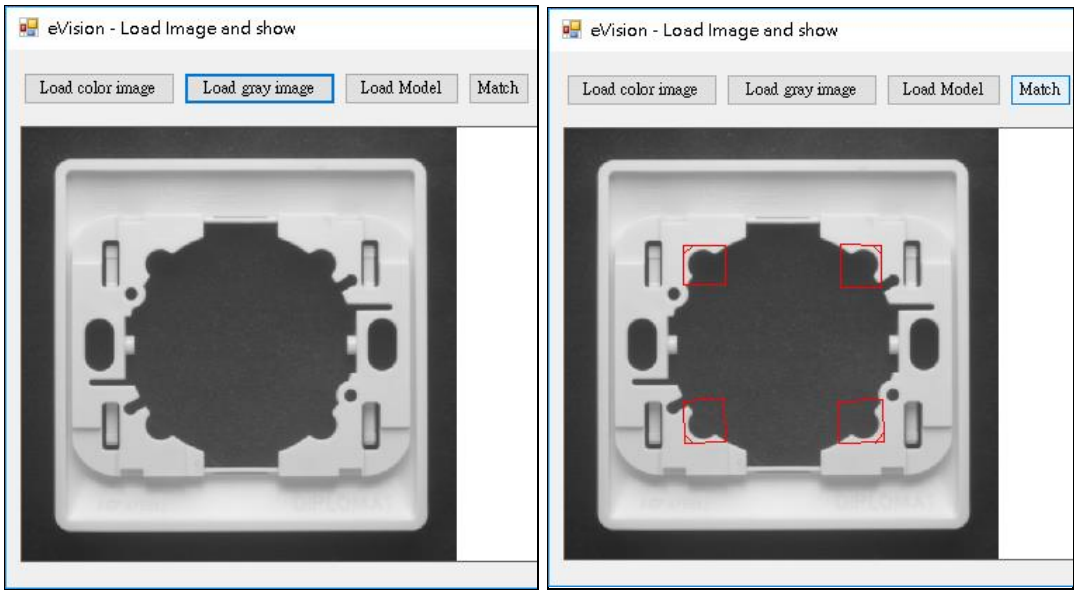
- 在 [Matching] 頁面中，點選 [Open...] 來載入圖像檔案： **EasyMatch \Switch1.tif** 。

- 針對新的圖像物件使用預設的變數名稱，並點選 [OK]。

**Step 3: 執行樣本匹配**

- 樣本匹配的動作會在來源圖像上自動執行，並在圖像上顯示出這些樣本的位置，點選 [Execute] 會將相應的程式碼插入到腳本視窗中。
- 點擊 [Results] 可以列出這些樣本匹配結果的相關資訊，點擊圖像上某個樣本的匹配結果就可以查看其對應的資訊。

範例程式執行畫面




程式碼片段：

```
private void btnLoadModel_Click(object sender, EventArgs e)
{
   openFileDialog1.Filter = "MCH(*.mch)|*.mch";
   if (openFileDialog1.ShowDialog()== DialogResult.OK)
```

```
    {
        EMatcher1.Load(openFileDialog1.FileName); //載入model檔
        //MessageBox.Show("Model file is loaded successfully.");
    }
}

private void btnMatch_Click(object sender, EventArgs e)
{
    EMatcher1.MaxAngle = 180.00f;
    EMatcher1.MinAngle = -180.00f; //設定最大與最小的容許旋轉角度
    EMatcher1.MaxPositions = 4; //設定最多找出幾個
    EMatcher1.MinScore = 0.95f; //設定最低的匹配分數
    EMatcher1.Match(EBW8Image1);
    EMatchPosition pos; //回傳的位置物件
    for (int i = 0; i < EMatcher1.NumPositions; i++)
    {
        pos = EMatcher1.GetPosition(i); //先取出該位置物件
        Console.WriteLine(i + ":(" + pos.CenterX+ "," + pos.CenterY + ")" + pos.Angle); //輸出相關結果
    }
    EMatcher1.DrawPositions(pbImage.CreateGraphics(),true, ScalingRatio); //true:表示要畫出一個角落的斜線
標示
}
```

## 在 ROI 中根據模型(model)來匹配樣本(pattern)

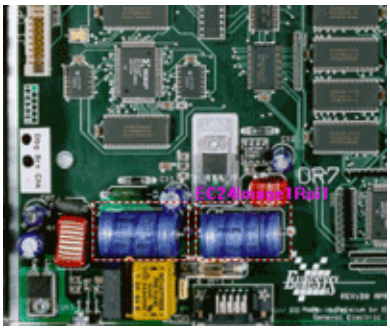
### 目標

依照下面的教學流程，你將學習到如何使用 EasyMatch 並從一張來源圖像的 ROI (感興趣區域)中，去訓練一個模型，藉由某些圖像去完成圖案的匹配。

首先你必須載入一張來源圖像，並設定一個 ROI (steps 1-2)。然後，使用剛才設定好的 ROI，訓練出一個模型 (step 3)。最後，你就能在來源圖像裡進行樣本匹配，並且能得到匹配結果的相關資訊。



**作為模型學習的 ROI 區域**



**以某個 ROI 所建立模型之多個匹配結果**

### Step 1:載入來源圖像

- 在主選單，點擊 [Image]，再點擊 [Open]。
- 載入下列的來源圖像檔： EasyMatch\BOARD.JPG。

- 針對新的圖像物件使用預設的變數名稱，接著點擊 [OK]。

**Step 2: 定義一個 ROI**

- 在圖像上按下滑鼠右鍵，在右鍵功能表中選擇 [New ROI...]。
- 使用預設的 ROI 變數名稱，並點擊 [OK]。你將會看見一個 ROI 的區域標示出現在來源圖像上。
- 使用滑鼠來調整 ROI 大小，並將其移至圖像右下角的藍色電容上。

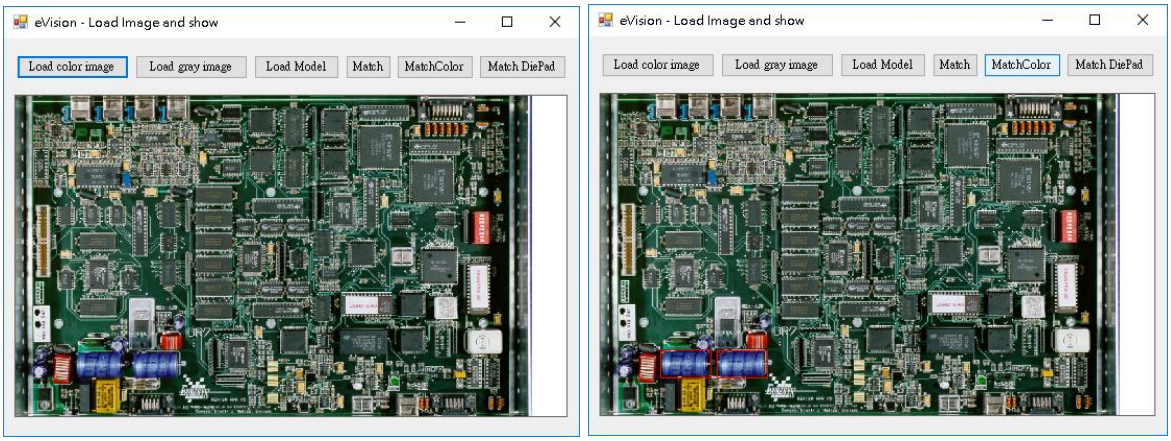
**Step 3: 從 ROI 來訓練模型**

- 在主功能表中，點擊 [EasyMatch]，再選擇 [New Match Tool]。
- 使用預設的變數名稱，並點擊 [OK]。
- 在 EMatcher 的對話方塊上，選擇 [Learning] 的頁面，接著從 Source Image 的下拉式選單中選擇前面步驟所建立的 ROI，並點擊 [Learn] 來建立樣本。

**Step 4: 進行樣板匹配**

- 在 [Matching] 的頁面中，將 [Max Occurrences] (最多出現次數) 這個欄位的數值遞增到 2。
- 在下拉式選單中，選擇同一張來源圖像。
- 點擊 [Execute] 來執行樣本匹配的動作。
- 使用者可以點擊 [Results] 來檢視樣本匹配結果的相關資訊。

## 範例程式執行畫面




## 程式碼片段：

```
private void btnMatchColor_Click(object sender, EventArgs e)
{
   //EMatcher1.MaxAngle = 180.00f;
   //EMatcher1.MinAngle = -180.00f; //設定最大與最小的容許旋轉角度
   EMatcher1.MaxPositions = 4; //設定最多找出幾個
   EMatcher1.MinScore = 0.7f; //設定最低的匹配分數
   EMatcher1.Match(EC24Image1);
   EMatchPosition pos; //回傳的位置物件
   for (int i = 0; i < EMatcher1.NumPositions; i++)
   {
      pos = EMatcher1.GetPosition(i); //先取出該位置物件
      Console.WriteLine(i + ":(" + pos.CenterX + "," + pos.CenterY + ")" + pos.Angle); //輸出相關結果
   }
   EMatcher1.DrawPositions(pbImage.CreateGraphics(), true, ScalingRatio); //true:表示要畫出一個角落的斜線
標示
}
```

## 透過使用忽略區域(don't care areas)來改善樣本的匹配分數

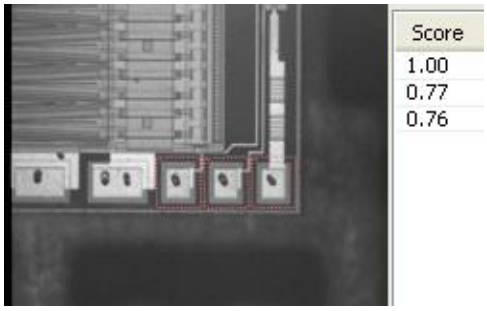
### 目標

依照下面的教學流程，你將學習如何使用 EasyMatch 處理忽略區域 (don't care areas)。忽略區域的設定有助於在圖像中定義出有意義的特徵，藉此遮蔽圖像中可能發生變化的區域。

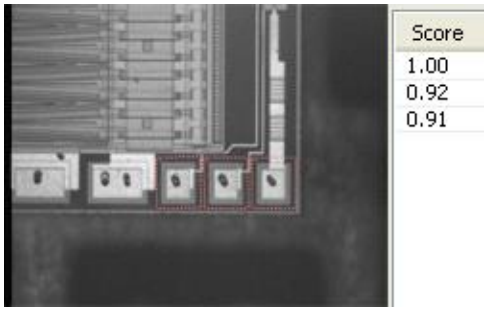
首先你需要載入參考圖像並且學習參考模型 (steps 1-2)。然後，你將執行參考模型範例的自動匹配模式，不使用忽略區域 (don't care areas) (step 3)。 由於找到的範例的匹配分數不夠高，你會定義一個忽略區域 (don't care area) 在參考模型上，並重新啟動檢測，接著會發現匹配分數會改善很多 (steps 4-5)。



**參考模型**




**未使用忽略區塊 (don't care areas)的樣本匹配分數**




**使用忽略區塊(don't care areas)的樣本匹配分數**

### Step 1: 載入參考圖像

- 從主功能表中，點選 [EasyMatch]，然後按下 [New Match Tool]
- 保留預設的變數名稱，然後按下 [OK]，把 EMatcher 對話框開啟。
- 請在 [Learning] 頁面中，點擊 [Open] 圖示，並載入圖像檔案：**EasyMatch\Die Pad Model 1.bmp**。
- 保留新圖像預設的變數名稱，然後點選 [OK]。

**Step 2: 學習參考模型 046**

- 點擊 [Learn] 來學習參考模型。

**Step 3: 在沒有忽略區域(don't care area)的狀況下檢測參考模型**

- 在 [Matching] 頁面中，點擊 [Open] 圖示，載入圖像檔案： **EasyMatch\Die Pad 1.bmp**。
- 保留新圖像物件預設的變數名稱，然後點擊 [OK]，這時會顯示與參考模型的匹配結果 。
- 將最大出現次數 (Max Occurrences) 增加到 3，勾選子像素內插 (Sub-Pixel Interpolate) 以提高匹配精準度。
- 輸入 '-10.0' 作為最小角度(Minimum Angle (Deg))。 (檢查角度的單位是否為度(degrees))。如果不是，請從 [View] > [Option]的功能表中選擇角度(degrees)做為單位。)
- 輸入 '10.0' 最為最大角度 (Maximum Angle (Deg))。
- 點擊 [Execute]，模型被匹配到的位置將在來源圖像中標記出來。
- 點擊 [Results] 查看每個找到實例的匹配分數，會發現最後兩個分數非常糟糕，這是由參考模型上半部的亮色矩形所導致的，我們可以透過將這個亮色的區域設為忽略區域 (don't care area) 來改善分數。

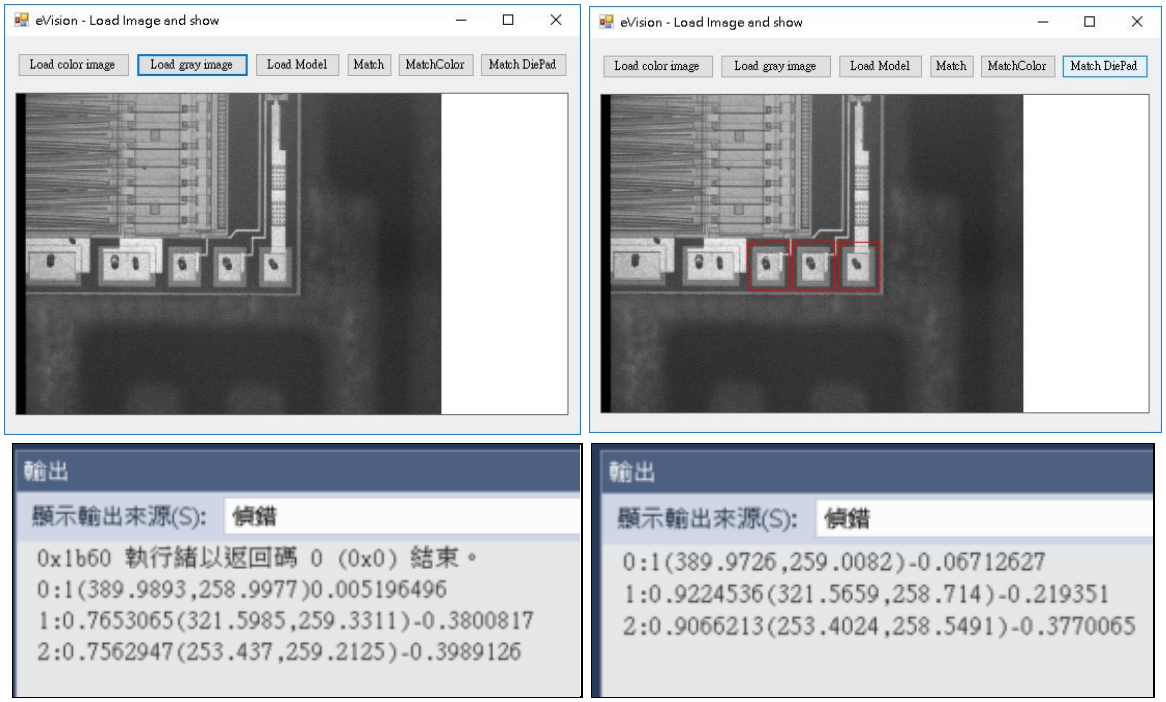
**Step 4: 定義忽略區域(don't care area)**

- 在忽略區域 (don't care area) 分頁中，從 [Blacken Inside] 中選擇單選按鈕 [Rectangle]。
- 在參考圖像中，將滑鼠指標移動到右上角亮色矩形的一角，壓住滑鼠左鍵移動至其對角，使忽略區域(帶有紅色條紋的矩形)完全籠罩在亮色矩形上。
- 在 [Don't Care Areas] 頁面中，點擊 [Learn]。

**Step 5: 使用忽略區域(don't care areas)來檢測參考模型的範例**

- 在 [Matching] 頁面中，點擊 [Execute]，參考模型的匹配範例依舊有被標示出來。
- 點擊 [Results] 比較新的匹配分數，分數提高很多。

範例程式執行畫面



程式碼片段：

```
private void button1_Click(object sender, EventArgs e)
{
   pbImage.Refresh();
   EMatcher1.MaxAngle = 10.00f;
   EMatcher1.MinAngle = -10.00f; //設定最大與最小的容許旋轉角度
   EMatcher1.MaxPositions = 4; //設定最多找出幾個
   EMatcher1.MinScore = 0.7f; //設定最低的匹配分數
   EMatcher1.Match(EBW8Image1);
   EMatchPosition pos; //回傳的位置物件
   for (int i = 0; i < EMatcher1.NumPositions; i++)
   {
      pos = EMatcher1.GetPosition(i); //先取出該位置物件
      Console.WriteLine(i + ":"+ pos.Score +"(" + pos.CenterX + "," + pos.CenterY + ")" + pos.Angle); //輸出相關結果
   }
   EMatcher1.DrawPositions(pbImage.CreateGraphics(), true, ScalingRatio); //true:表示要畫出一個角落的斜線標示
}
```

檢測多個高度衰減圖像中的參考模型

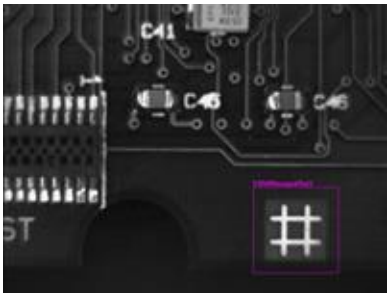
透過可忽略區域(don't care areas)提高實例分數

## 檢測多個高度衰減圖像中的參考模型

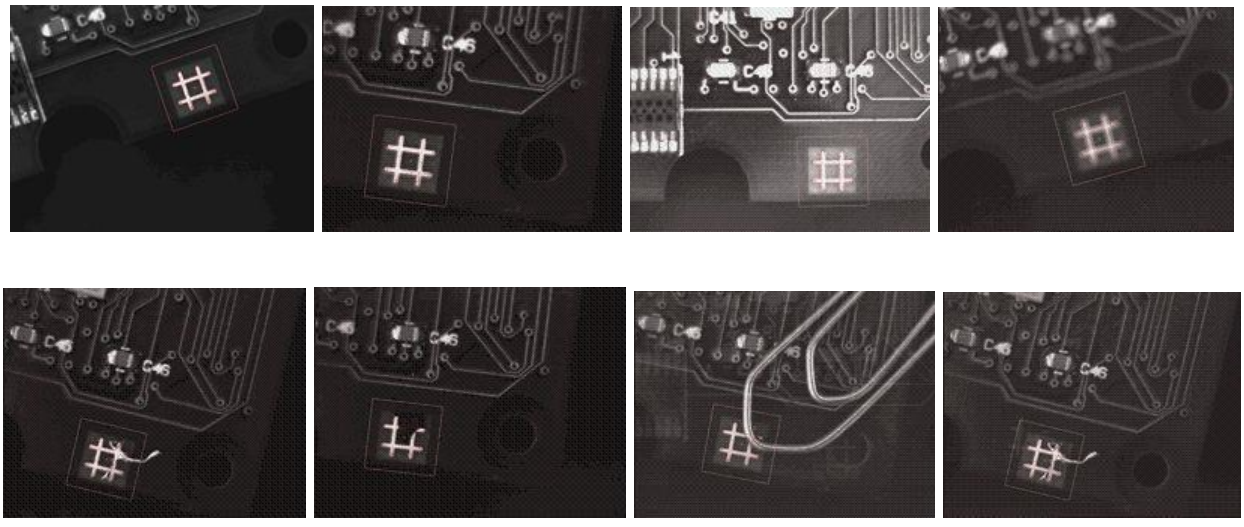
### 目標

依照下面的教學流程，你將會學習使用 EasyFind 在多個圖檔具有嚴重衰減的狀況下(highly-degraded)，仍可以偵測出圖像中的參考模型。衰減的原因可能是由於雜訊、模糊、遮蔽、部份缺失或不穩定的照明條件所造成。

首先，你必須先載入參考圖像，定義 EasyFind 將學習之參考模型的 ROI，並為偵測設置旋轉和縮放的容忍範圍 (steps1-4)。然後你就可以開啟多個圖像檔，並自動偵測圖像上的參考模型 (steps 5-6) ，可以發現即使圖像具有高度衰減的狀況，仍可以偵測出來。



**參考模型**



**偵測展示範例：即使圖像出現各種衰減狀況，仍可以搜尋到該參考模型**

**Step 1: 載入參考圖像**

- 在主功能表中，點擊 [EasyFind]，然後選取 [New EasyFind Tool]。
- 針對新的 PatternFinder 物件使用預設的變數名稱，並點擊 [OK]。接著開啟 PatternFinder 的對話方塊。
- 在 [Model] 區域中， 點擊 [Open] 圖示，並載入圖像檔案： **EasyFind\Fiducial 1.tif**。
- 保留新圖像物件的預設變數名稱，然後點擊 [OK]。.

**Step 2: 在參考圖像上建立 ROI 以定義參考模型**

- 在圖像上點擊滑鼠右鍵，接著從快顯功能表中選擇 [New ROI...]。
- 保留新 ROI 物件的預設變數名稱，然後點擊 [OK]。 預設的 ROI 在圖像上會出現一個藍色的長方形框框，接著開啟 ROI 管理功能的對話方塊。
- 將 ROI 拖放到參考模型上，並直接在圖像上調整其大小。也可以在 ROI 對話方塊中，直接輸入以下坐標：OrgX、OrgY、Width 和 Height 分別為 500, 365, 170, 170，然後點擊 [Close]。

**Step 3: 學習參考模型**

- 在 PatternFinder 的 [Model] 頁面中，從來源圖像下拉列表中選擇新增的 ROI，然後點擊 [Learn]，這時參考模型可以被完美的偵測出來 (如 ROI 中的綠色邊緣所示)。
- 在 PatternFinder 的 [SearchField] 頁面中，從來源圖像的下拉清單中選擇圖像對象，並勾選 [Draw Features] 的複選項，這時來源圖像上的模型位置和特徵點會被突顯示來。

**Step 4: 設置旋轉和縮放公差**

- 在 PatternFinder [Allowances] 頁面中，將角度和縮放的容許值設定為 25。

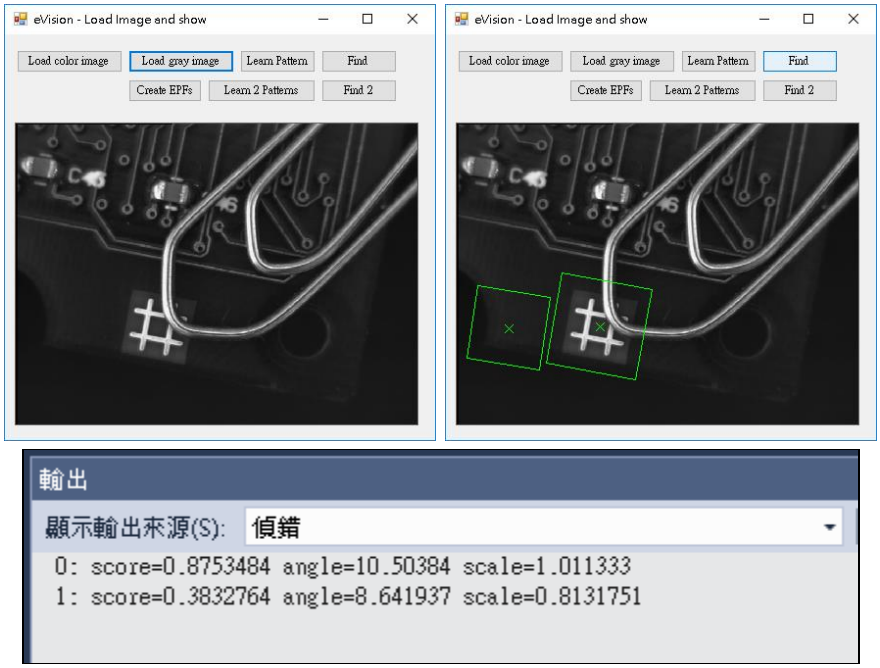
**Step 5: 選擇多個圖像**

- 在 PatternFinder 的 [SearchField] 頁面中，點擊 [Open] 圖示， 接著選擇圖像檔：EasyFind\Fiducial2.tif 到 Fiducial8.tif ( 可以使用 Shift 鍵來選擇多個圖檔)，然後點擊 [Open]。
- 保留新圖像物件的預設變數名稱，然後點擊 [OK]，當最後一個圖像出現時，即使有高度衰減的狀況，也可以偵測到該圖像上的參考模型。
- 系統會自動執行參考模型的偵測，一旦開啟新圖像，也不需要點擊 [Find]，因為偵測是自動執行的。不過，點擊 [Find] 的話會將相應的程式碼插入到腳本視窗中。
- 點擊 [Results] 來檢視偵測結果的相關資訊。

**Step 6: 瀏覽多個圖像**

- 在 PatternFinder 的 [Search Field] 頁面中，點擊 [Load Next] 或 [Load Previous] 圖示，可以瀏覽不同圖像的偵測結果。

範例程式執行畫面



程式碼片段：

```
private void btnLearn_Click(object sender, EventArgs e)
{
   EBW8Image1.Load("Fiducial 1.tif"); //直接載入參考圖像
   EBW8Image1Roi1.SetPlacement(500, 365, 170, 170);//設定ROI的範圍
   EBW8Image1Roi1.Attach(EBW8Image1); //attach:附加，將ROI附加在此圖上
   EPatternFinder1.Learn(EBW8Image1Roi1); //學習這個ROI
}

private void btnFind_Click(object sender, EventArgs e)
{
   EPatternFinder1.AngleTolerance = 25.00f; //角度容忍值
   EPatternFinder1.ScaleTolerance = 0.25f;  //縮放容忍值
   EPatternFinder1.MaxInstances = 10; //針對第二個例子增加為4個
   EPatternFinder1FoundPatterns = EPatternFinder1.Find(EBW8Image1);
   EFoundPattern found;
   for (int i= 0;i< EPatternFinder1FoundPatterns.Length; i++)
   {
      found = EPatternFinder1FoundPatterns.ElementAt(i); //把第i個取出來
      //found.DrawFeaturePoints = true; //把特徵點繪製出來，畫得很慢
      found.Draw(pbImage.CreateGraphics(), ScalingRatio);
      Console.WriteLine(i + ": score=" + found.Score + " angle=" + found.Angle + " scale=" +found.Scale);
   }
}
```

## 透過可忽略區域(don't care areas)提高實例分數

### 目標

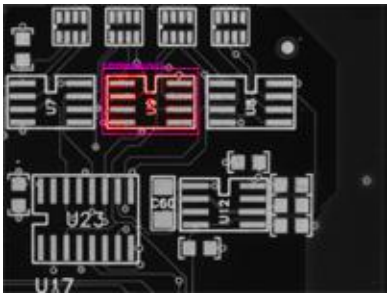
依照下面的教學流程，你將會學到如何在幾何圖案匹配裡使用 EasyFind 去處理忽略區域 ("don't care areas") 。

忽略區域有助於定義圖像裡有意義的特徵，藉由忽略圖像之間可能變化的區域，像是圖像或字元。

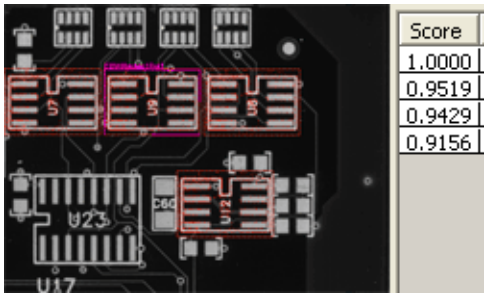
你需要先讀取參考圖像，定義一個 ROI，EasyFind 將在其中學習參考模型，並為要搜索的預期的情況設置旋轉容忍範圍 (steps 1-4)。

然後，你就可以自動檢測參考模型的實例，而無需使用忽略區域 (step 5)。

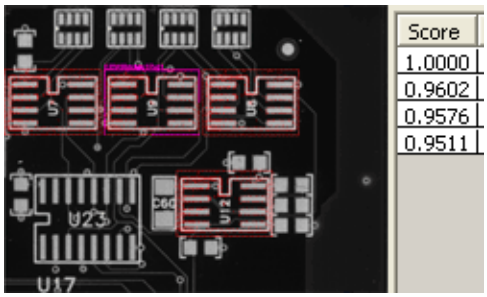
由於找到的實例的匹配分數不夠高，你將會在參考模型中定義一個忽略區域，然後重新啟動檢測，匹配分數會稍好一些 (steps 6-7)。



**參考模型**




**不使用忽略區域所找到的實例以及匹配分數**




**設定忽略區域後所找到的實例以及匹配分數**

**Step 1: 載入參考圖像**

- 在主功能表中， 按下 [EasyFind]， 然後按下 [New EasyFind Tool]。
- 保留新的 PatternFinder 物件的預設名稱，然後按下 [OK]，打開 PatternFinder 對話框。
- 在 [Model] 頁面中，按下 [Open] 圖示，然後載入圖像檔：

  **EasyFind\Solder Pad 1.tif**。
- 保留新圖像的預設名稱，然後點擊保留新圖像按下的預設名稱，然後按下 [OK]。

**Step 2: 新增 ROI 以定義出圖像上的參考模型**

- 在圖像上，用滑鼠右鍵點擊並從快顯功能表中選擇 [New ROI ...]。
- 保留新 ROI 物件的預設名稱，然後按下 [OK]。 預設的 ROI 會出現在圖像上（藍色長方形），並且出現 ROI 管理對話框。
- 將 ROI 拖放到欲搜尋之參考模型上，並調整其大小。 或者直接在 ROI 對話框中輸入以下坐標： OrgX、OrgY、Width、Height 分別為 200, 130, 190, 130，然後點擊 [Close]。

**Step 3: 學習參考模型**

- 在 [Model] 頁面中，從來源圖像下拉列表中選擇設定好的 ROI，然後點擊 [Learn]，將參考模型的相關輪廓特徵學習起來。

- 在 [Search Field] 頁面中，從來源圖像下拉列表中選擇圖像對象，勾選 [Draw Features] 複選框，這時模型位置和特徵點會在來源圖像中突顯出來。

**Step 4: 設置旋轉容忍範圍**

- 在 [Allowances] 頁面中，將角度容忍值設置為 5.0。

**Step 5:在沒有"忽略區域"的情況下檢測參考模型的實例**

- 在 [Search Field] 頁面中，設定 [Max Instances] 為 4，並點擊 [Find]，這時可以匹配到參考模型的實例都被明顯地標示出來。
- 點擊 [Results] 來查看每個實例的匹配分數，即使分數很高，我們仍然可以透過使用忽略區域的遮罩來遮蔽已學習樣本 (pattern) 上的文字，從而稍微改善匹配分數 。

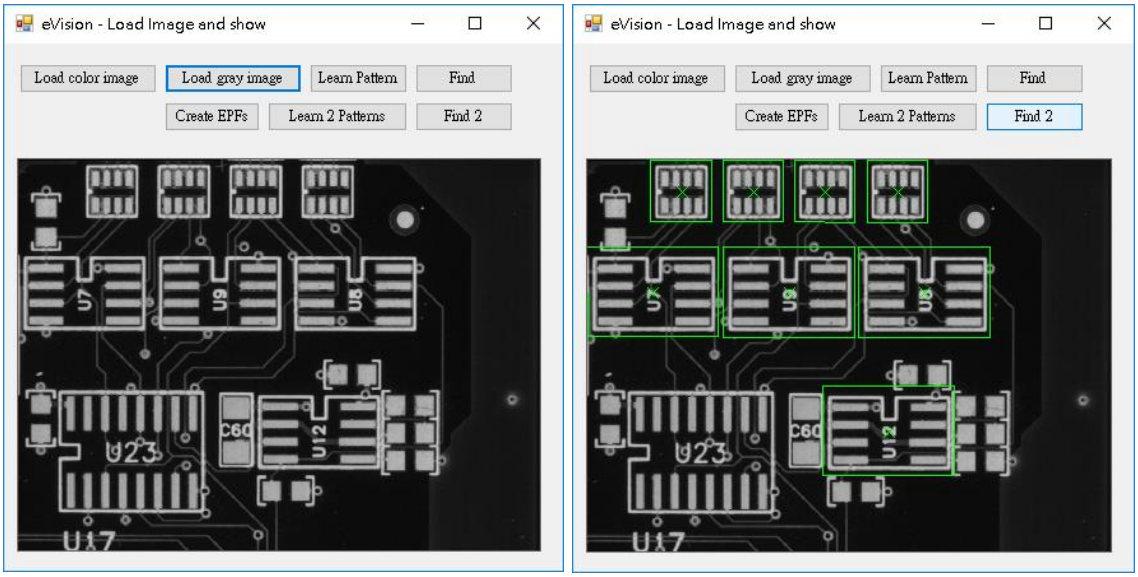
**Step 6: 定義 忽略區域**

- 在 [Don't Care Areas] 頁面中，從 [Blacken Inside] 群組中點選 [Rectangle] 的單選按鈕。
- 在定義參考模型的 ROI 中，將游標指針移動到文字 "U9" 的左上角，按下左鍵並拉出一個忽略區域（帶紅色條紋的矩形）以遮蔽該文字。

**Step 7: 檢測參考模型含忽略區域的實例**

- 在 [Don't Care Areas] 頁面中，點選 [Learn]，然後在點選 [Find]，參考模型中被匹配到的實例仍然會被標示出來，但是該文字已經不再被列入搜尋的範圍。
- 點擊 [Results] 來檢視新的匹配分數，可以發現比前面的結果更好一些。

## 範例程式執行畫面




## 程式碼片段：

(宣告)

```
EPatternFinder EPatternFinder1 = new EPatternFinder(); // EPatternFinder instance
EPatternFinder EPatternFinder2 = new EPatternFinder(); // EPatternFinder instance
ArrayList EPFs=new ArrayList(0); //放置EPatternFinder物件的ArrayList
EROIBW8 EBW8Image1Roi1 = new EROIBW8(); // EROIBW8 instance
EFoundPattern[] EPatternFinder1FoundPatterns; // EFoundPattern instances
```

(Create EPFs按鍵)

```
private void button1_Click(object sender, EventArgs e)
{
   //動態建立EPatternFinder物件
   for (int i = 0; i < 2; i++) {
      EPatternFinder tmpEPF = new EPatternFinder();
      tmpEPF.AngleTolerance = 25.00f; //角度容忍值
      tmpEPF.ScaleTolerance = 0.25f;  //縮放容忍值
      tmpEPF.MaxInstances = 4; //針對第二個例子增加為4個
      EPFs.Add(tmpEPF);
   }
}
```

(Find 2按鍵)

```
private void btnFind2_Click(object sender, EventArgs e)
{
   //原本沒有使用ArrayList的寫法
   //EPatternFinder1.AngleTolerance = 25.00f; //角度容忍值
   //EPatternFinder1.ScaleTolerance = 0.25f;  //縮放容忍值
   //EPatternFinder1.MaxInstances = 4; //針對第二個例子增加為4個
   //EPatternFinder1FoundPatterns = EPatternFinder1.Find(EBW8Image1);
   //EFoundPattern found;
   //for (int i = 0; i < EPatternFinder1FoundPatterns.Length; i++)
   //{
   //    found = EPatternFinder1FoundPatterns.ElementAt(i); //把第i個取出來
   //    //found.DrawFeaturePoints = true; //把特徵點繪製出來，畫得很慢
   //    found.Draw(pbImage.CreateGraphics(), ScalingRatio);
   //    Console.WriteLine(i + ": score=" + found.Score + " angle=" + found.Angle + " scale=" + found.Scale);
   //}

   //EPatternFinder2.AngleTolerance = 5.00f; //角度容忍值
   //EPatternFinder2.MaxInstances = 4; //針對第二個例子增加為4個
   //EPatternFinder1FoundPatterns = EPatternFinder2.Find(EBW8Image1);
   //for (int i = 0; i < EPatternFinder1FoundPatterns.Length; i++)
   //{
   //    found = EPatternFinder1FoundPatterns.ElementAt(i); //把第i個取出來
   //    //found.DrawFeaturePoints = true; //把特徵點繪製出來，畫得很慢
   //    found.Draw(pbImage.CreateGraphics(), ScalingRatio);
   //    Console.WriteLine(i + ": score=" + found.Score + " angle=" + found.Angle + " scale=" + found.Scale);
   //}

   //使用ArrayList的寫法
   EPatternFinder tmpEPF;
   for (int idx=0;idx < EPFs.Count; idx++)
   {
      tmpEPF = (EPatternFinder)EPFs[idx]; //依序取出EPatternFinder物件
      EPatternFinder1FoundPatterns = tmpEPF.Find(EBW8Image1);
      EFoundPattern found;
      for (int i = 0; i < EPatternFinder1FoundPatterns.Length; i++)
      {
         found = EPatternFinder1FoundPatterns.ElementAt(i); //把第i個取出來
         //found.DrawFeaturePoints = true; //把特徵點繪製出來，畫得很慢
         found.Draw(pbImage.CreateGraphics(), ScalingRatio);
```

```
            Console.WriteLine(i + ": score=" + found.Score + " angle=" + found.Angle + " scale=" + found.Scale);
        }
    }
}
```

針對特定座標系統進行點座標定位

量測物件的旋轉角度

測量一個圓的直徑

測量歪斜的矩形

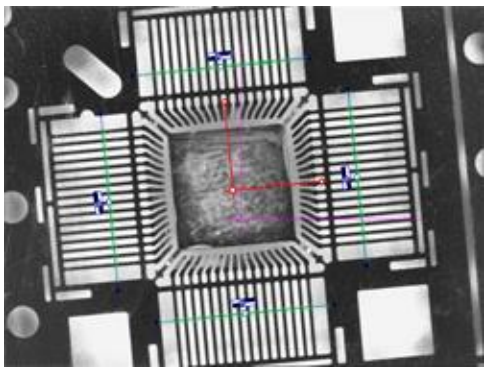
反變形扭曲的圖像

## 針對特定座標系統進行點座標定位

### 目標

依照下面的教學流程，你將學會如何使用 EasyGauge 來檢測導線架 (lead frames)。本操作可以準確地取得導線架的尺寸、位置、曲率、大小、角度及直徑，EasyGauge 提供強大的邊緣座標檢測功能，直覺化且容易調整的操作介面，並且可以測量複雜的圖像。

首先，選取一張基準圖像，由網格點 (dot grid) 形成的圖像 (step 1)，接著調整你需要校正的區域 (step 2)。校正完畢後，載入一張導線架的圖像 (step 3) 並設置座標系統(和導線架相同形狀)，在此座標系統中，你可以定義點座標量測的位置 (point gauges) (step 5-6)。最後，你可以開啟另一張導線架的圖像，而此圖會有微小的角度偏差，所以，座標系統需要進行旋轉 (step 7-8)，此量測功能也會自動調整 (step 9)，讓所有的量測結果皆可以用一般物理單位來表示(mm)。



**四個 point gauges 至於四邊的引線上.**

### Step 1:讀取基準圖像

- 從主選單點選 [EasyGauge]，然後點選 [New World Shape]。
- 針對新的 EWorldShape 物件保留預設的變數名稱，並點選 [OK]。
- 在 [Dot Grid Calibration] (網格點校正) 頁面中，點選 [Open] 圖示，然後讀取下列的圖像檔：**EasyGauge\Dot grid 2.tif**。

- 針對新物件保留預設的名稱，並點選 [OK]。

**Step 2: 調整需要的區域**

- 點選 [Calibrate] (校正)，按下之後，目前影像的可視範圍皆已校正好，所有結果都會以一般物理單位來表示(mm)。

**Step 3: 讀取導線架(lead frames)的圖像**

- 從 [Gauges] 頁面中，點選 [Open] 圖示，載入圖像檔： **EasyGauge\Lead Frame 1.tif**。
- 保留圖像預設的變數名稱，並點選 [OK]。

**Step 4: 設置座標系統**

- 在 [Gauges] 右邊的對話視窗中，使用滑鼠右鍵點擊 EWorldShape1，從快顯功能表中選擇 [New] > [Frame Shape]。
- 保留預設的變數名稱，並點選 [OK]，這時 EWorldShape1 下方會出現一個 EFrameShape1 的形狀框架 。
- 拖曳該框架到圖像中方形的中心處。

**Step 5: 附加一個點量測 (point gauge) 到導線架上**

- 在 [Gauges] 右邊的對話視窗中，使用滑鼠右鍵點擊 EFrameShape1，從快顯功能表中選擇 [New] > [Point Gauge]。
- 保留預設的變數名稱，並點選 [OK]。一個點量測工具顯示在圖像上，由以下幾個部分組成：
  - 藍色線段標示欲搜尋邊緣轉換的位置
  - 使用者可以使用三個白色調整點來移動或者旋轉該量測位置
  - 灰色箭頭指示該點量測的走訪方向
  - 白色與黑色的方形指示欲搜尋的轉換類型

- 綠色十字指示所搜尋到的轉換點(如果有找到的話)

- 將 Point Gauge 放置在導線上，在 EPointGuage1 對話方塊的 [Position] 中設定 Center Y 屬性的數值為 7 以及 Tolerance 屬性的數值為 5。

**Step 6: 在形狀框架(Frame Shape)中附加測量點(Point Gauges)**

- 在 [Gauges] 頁面中的 EWorldShape1 底下的 EFrameShape1，增加三個點測量 (Point Gauge)（請參考 Step 5）。
- 將剛建立的三個 Point Gauge 中的屬性改為以下設定：
  - Center Y = -7, Tolerance = 5
  - Center X = 7, Tolerance = 5, Angle = 90
  - Center X = -7, Tolerance = 5, Angle = 90

**Step 7: 讀取另一張圖像**

- 在 [Gauges] 的頁面中， 點選 [Open] 的圖示，載入下面的圖像檔：**EasyGauge\Lead Frame 2.tif**。
- 使用預設的圖像名稱， 點選 [OK] 後載入該圖像。

**Step 8: 調整座標系統**

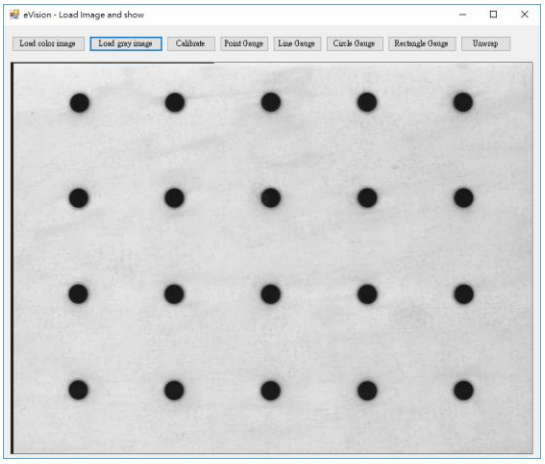
- 在形狀框架 (Frame Shape) 的屬性設定框中，將角度 (Angle) 設定為 5.8。
- 拖曳圖像上的形狀框架 (Frame Shape) 至圖像的正中心，形狀框架內的測量點都會自動調整到對應正確的位置。

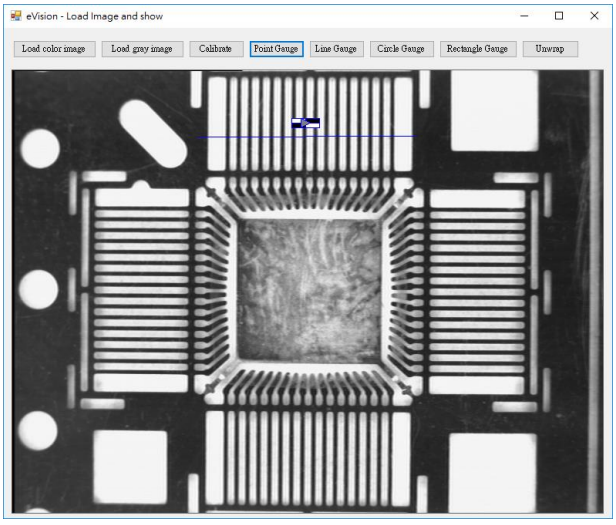
**Step 9: 執行檢測**

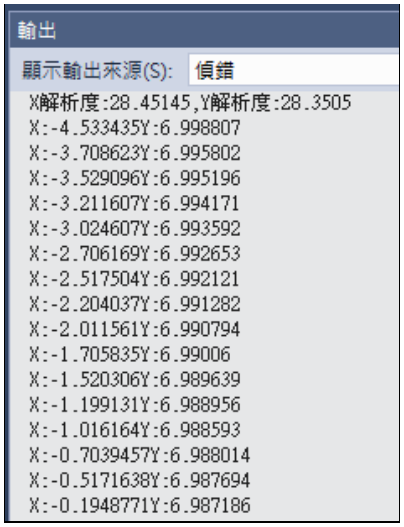
- 上述步驟設定好的時候，圖像都會被自動檢測，但是點選 WorldShape 對話框內的 [Process] 按鈕時，才會自動產生程式碼至腳本視窗中。

- 從點測量的對話框內點選 [Results] 頁面，可以看到所有檢測到的點位置座標，所有的檢測結果都是以一般物理單位來表示(mm)，其數值是參照所設定的形狀框架的座標系統 (frame shape system)。

範例程式執行畫面




程式碼片段：

```
private void btnCalibrate_Click(object sender, EventArgs e)
{
   EWorldShape1.SetSensorSize(EBW8Image1.Width, EBW8Image1.Height); //直接設定為網格點的寬高
```

```
    //EWorldShape1.Process(EBW8Image1, true); //原本產生的程式碼是"TRUE"(此方法已整併，不用呼叫)
    EWorldShape1.AutoCalibrateDotGrid(EBW8Image1, 5.00f, 5.00f, false); //5.00f = 5mm
    if (EWorldShape1.CalibrationSucceeded()) //如果有校正成功
    {
        MessageBox.Show("Calibration finished.");
        //取出校正後的相關係數
        Console.WriteLine("X解析度:" + EWorldShape1.XResolution + ",Y解析度:" + EWorldShape1.YResolution);
    }else
    {
        MessageBox.Show("Calibration fail.");
    }
}
```

```
private void btnPointGauge_Click(object sender, EventArgs e)
{
    //EWorldShape1.Process(EBW8Image1, true);
    EFrameShape1.Attach(EWorldShape1);
    EFrameShape1.SetCenterXY(0, 2);//依據工具上面調整位置後的數值
    EPointGauge1.Attach(EFrameShape1);
    EPointGauge1.SetCenterXY(0, 7);
    EPointGauge1.SetTolerances(5);

    EPointGauge2.Attach(EFrameShape1);
    EPointGauge2.SetCenterXY(0, -7);
    EPointGauge2.SetTolerances(5);

    EPointGauge3.Attach(EFrameShape1);
    EPointGauge3.SetCenterXY(7, 0);
    EPointGauge3.SetTolerances(5);
    EPointGauge3.Angle = 90;

    EPointGauge4.Attach(EFrameShape1);
    EPointGauge4.SetCenterXY(-7, 0);
    EPointGauge4.SetTolerances(5);
    EPointGauge4.Angle = 90;

    EPointGauge1.Measure(EBW8Image1); //進行實際的量測
    EPointGauge1.Draw(pbImage.CreateGraphics());
```

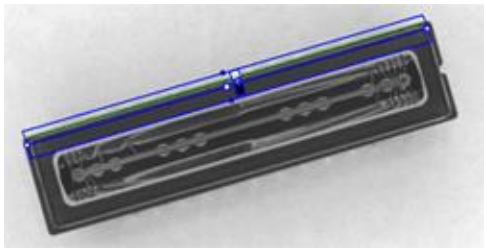
```
    //取出量測的結果
    for (int i = 0; i < EPointGauge1.NumMeasuredPoints; i++)
    {
        measuredPoint = EPointGauge1.GetMeasuredPoint(i); //取出第i個點
        Console.WriteLine("X:" + measuredPoint.X + "Y:" + measuredPoint.Y);
    }
}
```

## 量測物件的旋轉角度

### 目標

依照下面的教學流程，您將會學習到如何使用 EasyGauge 來測量一個待測物 (CCD sensor package) 的旋轉角度。因此我們只需取得角度補償數據 (這並不需要在已校正的區域中使用)，最後所有的幾何參數的結果將被以 pixel 為單位來表示。

首先，您需要讀取來源圖像 (step 1)，再附加一個合適的直線測量 (line fitting gauge) (step 2)，最後將會自動執行檢測以量測出角度 (step 3)。



**線擬合量測的結果 (Line fitting gauge)**

### Step 1: 載入來源圖像

- 從主選單點選 [EasyGauge] 下的 [New World Shape]。
- 使用預設的變數名稱，並點選 [OK]。
- 在 [Gauges] 頁面上點選 [Open] 按鈕，並且讀取圖像檔：**EasyImage\CCD.tif**
- 針對新的圖像物件使用預設的名稱，並點選 [OK]。

### Step 2: 在圖像上附加一個直線測量 (line gauge)

- 在 [Gauges] 頁面上的對話框裡，對 World Shape 圖示點選滑鼠右鍵，在右鍵選單裡點選 [New]，並點選 [Line Gauge]。
- 使用預設的變數名稱，並點選 [OK]，直線測量 (line gauge) 會顯示在圖像上，其包含以下元素：
  - 藍色框線用來標示搜尋到的邊緣轉換區域

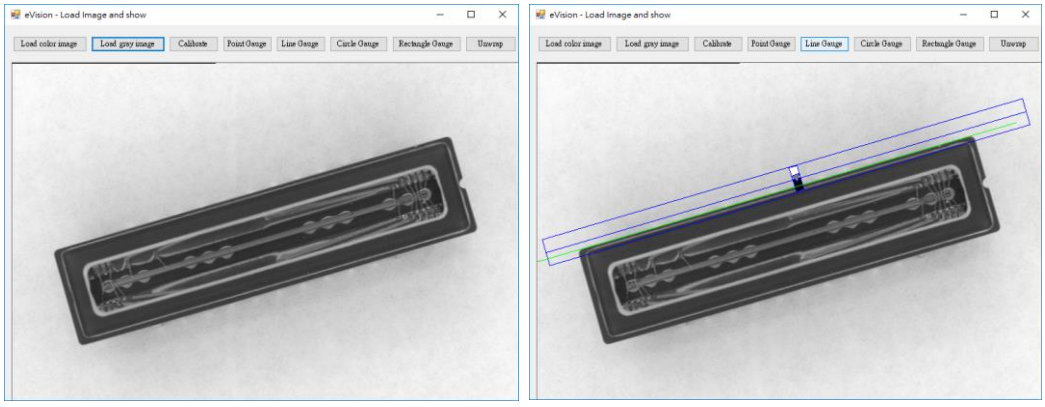
- 使用者可以使用五個白色調整點來移動或者旋轉該量測區域
- 灰色箭頭指示該區域的走訪 (traverse) 方向
- 白色與黑色的方形指示期望搜尋到的轉換類型
- 綠色線指示所搜尋到的轉換點 (如果有找到的話)

- 使用抓取點來移動、轉動和擴大直線測量的區域， 所以請把它放置在待測物的上方邊緣，且把灰色箭頭設定朝下。
- 在 Line Gauge 對話框的 [Measurement] 頁面的 Type 的下拉式選單選擇 'White to Black'，表示要搜尋從白色轉換為黑色的邊緣。

**Step 3: 執行檢查**

- 圖像會被自動檢測，然後在 World Shape 對話框中點選 [Process]，程式碼將會插入到程式腳本視窗中。
- 點選 [Results] 頁面取得量測到的角度值。
- 可以在繪製樣本 (Draw Sample) 中勾選 [Points] 來檢視直線上個別匹配到的點。

範例程式執行畫面



程式碼片段：

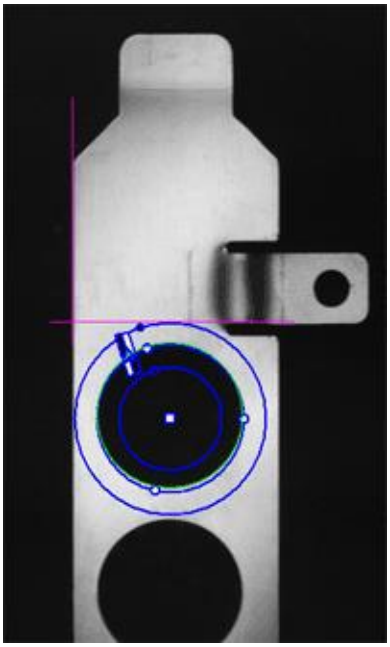
```
private void btnLineGauge_Click(object sender, EventArgs e)
{
   ELineGauge1.Attach(EWorldShape1); //只有依附在WorldShape上，以pixel為單位
   ELineGauge1.SetCenterXY(364, 174);//依照工具上面的設定
   ELineGauge1.Tolerance = 20;//左右寬度
   ELineGauge1.Length = 732;  //長度
   ELineGauge1.Angle = -16.35f; //角度
   ELineGauge1.TransitionType = ETransitionType.Wb; //轉換型態 White to Black
   ELineGauge1.Measure(EBW8Image1); //進行line gauge
   ELineGauge1.Draw(pbImage.CreateGraphics()); //畫出量測的區域
   ELineGauge1.Draw(pbImage.CreateGraphics(),EDrawingMode.Actual); //畫出實際的結果
   measuredLine = ELineGauge1.MeasuredLine; //直接取出檢測到的直線
   Console.WriteLine("Angle:" + measuredLine.Angle);
}
```

## 測量一個圓的直徑

### 目標

依照下面的教學流程，您將學習如何使用 EasyGauge 來測量圖像中圓形物件的直徑。

首先載入用於校正的圖像 - 點網格 (step 1)，校準可視範圍 (step 2)。 然後，您將載入來源圖像以進行檢查(step 3)，然後附上一個圓形測量功能 (circle gauge)(step 4)，進行圓形測量 (step 5)。所有測量結果均以物理單位表示。



**測量一個圓的直徑**

### Step1：載入校正圖像

- 從主功能表中，點擊 [EasyGauge]，然後點擊 [New World Shape]。
- 保留預設的變數名稱，然後點擊 [OK]。
- 在 [Dot Grid Calibration] 頁面中，點擊 [Open] 圖示，並載入圖像檔：
  **EasyGauge\Dot Grid 1.tif**
- 保留新圖像物件預設的變數名稱，然後點擊 [OK]。

**Step2：校正可視範圍**

- 點擊 [Calibrate]。 從現在起，可視區域已經被校準，所有結果將以物理單位表示。

**Step3：載入來源圖像**

- 從 [Gauges] 分頁中點擊 [Open] 圖示，然後載入圖像檔：**EasyGauge\ Bracket 6.tif**
- 保留預設的變數名稱，然後點擊 [OK]。

**Step4：在圖像上附加一個圓形量測功能**

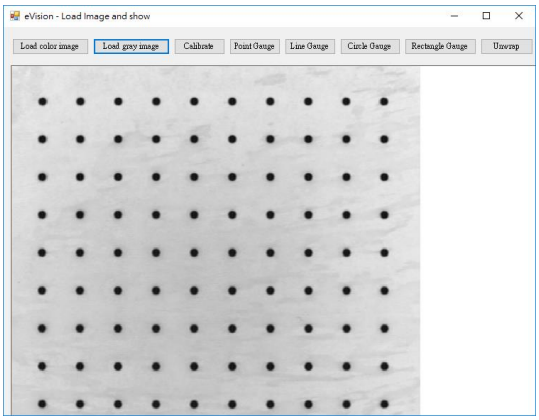
- 在 WorldShape 對話框的 [Gauges] 頁面中，右鍵點擊框架形狀圖示，從快顯功能表中選擇 [New] > [Circle Gauge]。
- 保留預設的變數名稱，然後點擊 [OK]。
- 圓形量測工具出現在圖像上，它由以下元素組成：
    - 一個藍色的環，用於搜索該環狀範圍內的圓形。
    - 一個預計搜尋圓形的藍色圓圈。
    - 六個白點，用於移動和旋轉分段。
    - 一個灰色的箭頭，指示圓形邊緣的搜尋方向。
    - 黑色和白色矩形，指示搜索哪種類型的邊緣轉換。
    - 綠色的圓弧或圓形，表示找到的圓圈（如果有找到的話）。
- 使用手柄，將圓形量測工具拖放的支架孔上，將預計搜尋的藍色圓形放置在圓孔邊緣上，並在必要時擴大搜尋區域。
- 在圓形量測對話框的 [Measurement] 頁面中，從下拉式選單中選擇 “From Begin”。

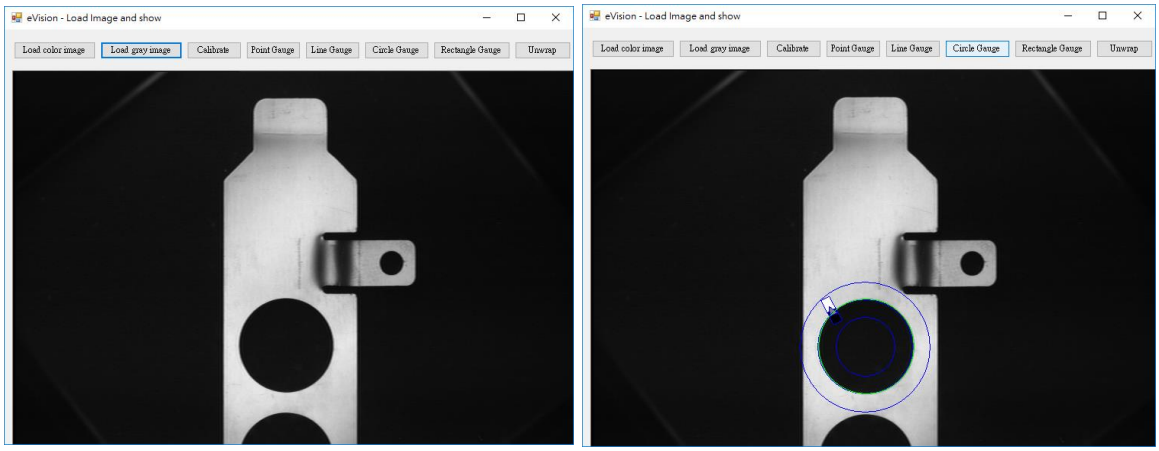
**Step5：進行量測**

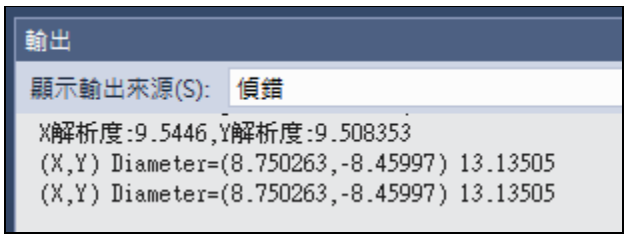
- 圖像將會被自動進行圓形量測，點擊 World Sqhape 對話框中的 [Process] ，將在腳本視窗中插入相應的程式碼。

- 點擊 [Results] 頁面檢視測量的圓形直徑， 所有測量結果均以物理單位表示。

範例程式執行畫面




程式碼片段：

```
private void btnCircleGauge_Click(object sender, EventArgs e)
{
   ECircleGauge1.Attach(EWorldShape1);
   ECircleGauge1.SetCenterXY(8.6f, -8.5f);//依照工具上面的設定
   ECircleGauge1.Tolerance = 2.46f;//左右寬度
```

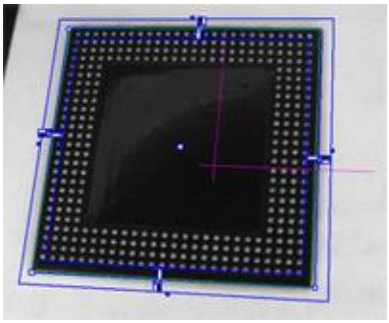
```
    ECircleGauge1.Diameter = 13;  //長度
    ECircleGauge1.Angle = -1.8f;
    ECircleGauge1.Amplitude = 360;
    ECircleGauge1.TransitionType = ETransitionType.Bw;
    ECircleGauge1.Measure(EBW8Image1);
    ECircleGauge1.Draw(pbImage.CreateGraphics()); //畫出偵測區域
    ECircleGauge1.Draw(pbImage.CreateGraphics(), EDrawingMode.Actual); //畫出偵測結果
    measuredCircle = ECircleGauge1.MeasuredCircle;
    Console.WriteLine("(X,Y) Diameter=(" + measuredCircle.X+"," + measuredCircle.Y+") "
+measuredCircle.Diameter);


}
```

## 測量歪斜的矩形

### 目標

依照下面的教學流程，您將學習如何使用 EasyGauge 在不規則的矩形組件上執行測量。

為了取得在真實工部上的測量結果，我們需要在校準的可視範圍中進行量測。 首先您需要載入圖像進行校準—使用網格點 (Dot Grid) 圖像— (step 1)， 並校準可視範圍 (step 2)。然後載入歪斜的圖像 (step 3)，並附上一個合適此矩形的測量範圍 (step 4)。自動執行檢驗測量 (step 5)，所有測量結果均以實際的物理單位表示。



**測量扭曲的矩形**

### Step 1: 載入校準圖像

- 在主功能表上點選 [EasyGauge] 之後點選 [New World Shape] 。
- 保留預設的變數名稱，然後點擊 [OK]。
- 在 [Dot Grid Calibration] 頁面中，點擊 [Open] 圖示，並載入圖像檔：EasyGauge\Dot Grid 5.tif
- 這個點網格圖像是在相同條件下所拍攝下來，並且與我們想要量測的圖像具有相同的扭曲狀況。
- 保留新圖像物件預設的變數名稱，然後點擊 [OK]。

### Step 2: 校準可視範圍

- 點擊 [Calibrate] 功能，按下之後，可視範圍已完成校正，所有量測結果將以物理單位表示。

**Step 3: 載入歪斜的圖像**

- 從 [Gauges] 頁面中點擊 [Open] 圖示，然後載入圖像檔：EasyGauge \Distorted Component.tif
- 保留預設的變數名稱，然後點擊 [OK]。

**Step 4: 將矩形量規附加到圖像上**

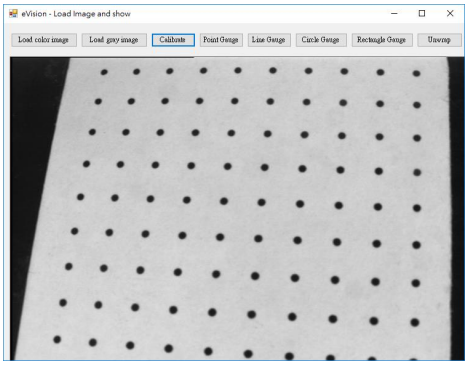
- 在 [Gauges] 頁面中，右鍵點擊 World Shape 圖示，然後從下拉式選單中選擇 [New] > [Rectangle Gauge]。
- 保留預設的變數名稱，然後點擊 [OK]，開啟矩形量測對話框，在圖像上繪製矩形量表。 它由以下元素組成：
  - 一個藍色矩形環，用來搜尋範圍中的矩形物件。
  - 一個預期要搜尋到的藍色矩形。
  - 十一個白色座標，允許使用者移動和延伸搜尋區域。
  - 灰色箭頭指示要邊緣搜尋的方向。
  - 黑色和白色矩形，指示搜索哪種類型的邊緣轉換。
  - 一個綠色的矩形，表示找到的矩形（如果有找到的話）。
- 由於視覺透視效果，矩形量測的結果看起來不像標準矩形。
- 可以移動圖像中的矩形量測工具的中心點，並觀察量測到矩形的變形。由於圖像已經經過校正，矩形測量工具的形狀也會依據可視範圍調整變形，使用者也可以擴展矩形搜尋區域，並調整預期搜尋到的藍色矩形放置在圖像的矩形邊緣上。

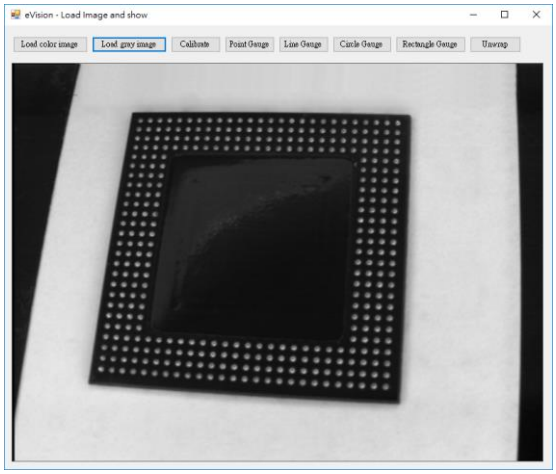
- 在 ERectangleGauge 對話框的 [Measurement] 的選項中，從 Type (類型)下拉列表中選擇 “White to Black”，表示要搜尋從白色轉換為黑色的邊緣，然後從 Choice (選擇) 下拉清單中選擇 “From Begin” (表示選擇一開始找到的邊緣)。

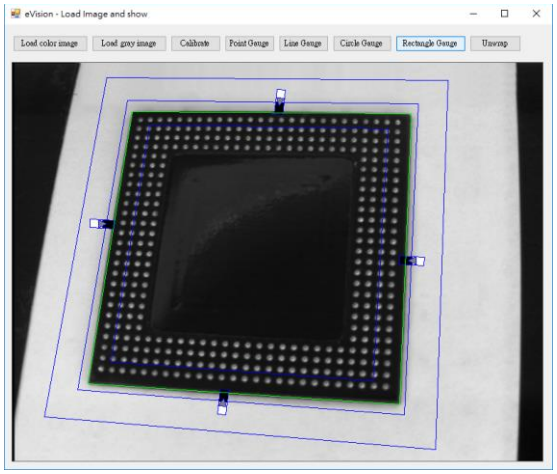
**Step 5: 執行量測**

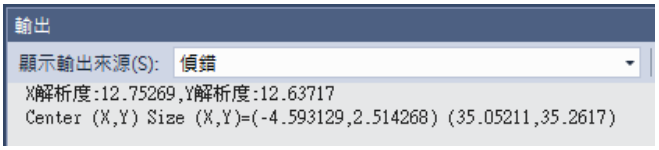
- 圖像會被自動量測，接著點擊在 World Shape 對話方塊中的 [Process] ，會在腳本視窗中插入對應的程式碼。
- 點擊 [Results] 來檢視測量的 X 和 Y 尺寸，所有測量結果均以實際物理單位表示。

範例程式執行畫面



程式碼片段：

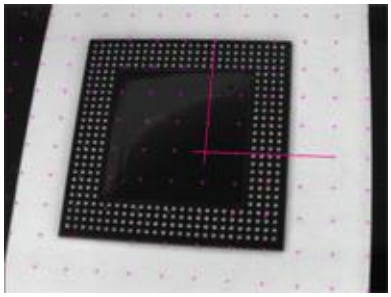
```
private void btnDistoredRec_Click(object sender, EventArgs e)
{
    ERectangleGauge1.Attach(EWorldShape1);
    ERectangleGauge1.SetCenterXY(-4.77733f, 2.81394f); //設定好再工具上面調整好的相關參數
    ERectangleGauge1.SetSize(36.797f, 37.7268f);
    ERectangleGauge1.Angle = -0.389911f;
    ERectangleGauge1.Tolerance = 3.54711f;
    ERectangleGauge1.TransitionChoice = ETransitionChoice.NthFromBegin;
    ERectangleGauge1.TransitionType = ETransitionType.Wb;
    ERectangleGauge1.Measure(EBW8Image1); //進行量測
    ERectangleGauge1.Draw(pbImage.CreateGraphics()); //畫出偵測區域
    ERectangleGauge1.Draw(pbImage.CreateGraphics(), EDrawingMode.Actual); //畫出偵測結果
    measuredRectangle = ERectangleGauge1.MeasuredRectangle;  //再把量測到的Rectangle拿出來，如果沒有找到CenterXY跟Size會等於設定值
    Console.WriteLine("Center (X,Y) Size (X,Y)=(" + measuredRectangle.X + "," + measuredRectangle.Y + ") (" + measuredRectangle.SizeX + "," + measuredRectangle.SizeY + ")");
}
```

# 反變形扭曲的圖像

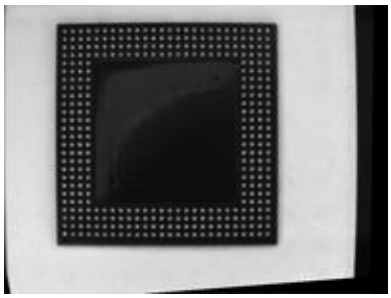
## 目標

依照下面的教學流程，您將會學到如何利用 EasyGauge 去進行網格校正 (grid calibration)，並且反變形 (unwarp) 來轉正一個扭曲的圖像。

首先載入一張圖像進行校正 —網格點校正— (step 1)，並且調整圖像的可視範圍 (step 2)，然後再載入一張扭曲的圖像 (step 3)，來執行反變形(unwarping)的操作 (step 4)。



**扭曲變形的圖像**



**透過校正進行反變形轉正後的圖像**

### Step 1: 載入校正圖像

- 從主功能表中，點擊 [EasyGauge] 中的 [New World Shape]。
- 保留預設的變數名稱，點擊 [OK]。

- 在 [Dot Grid Calibration] 頁面中，點擊 [Open]，並且載入圖像檔：EasyGauge\Dot Grid 5.tif。這個網格點是在相同的拍攝條件下所取得的，並且與我們想要變形的圖像具有相同的扭曲狀況。
- 保留新圖像預設的變數名稱，然後點擊 [OK]。

**Step 2: 校正可視範圍**

- 點擊 [Calibrate] 進行校正。

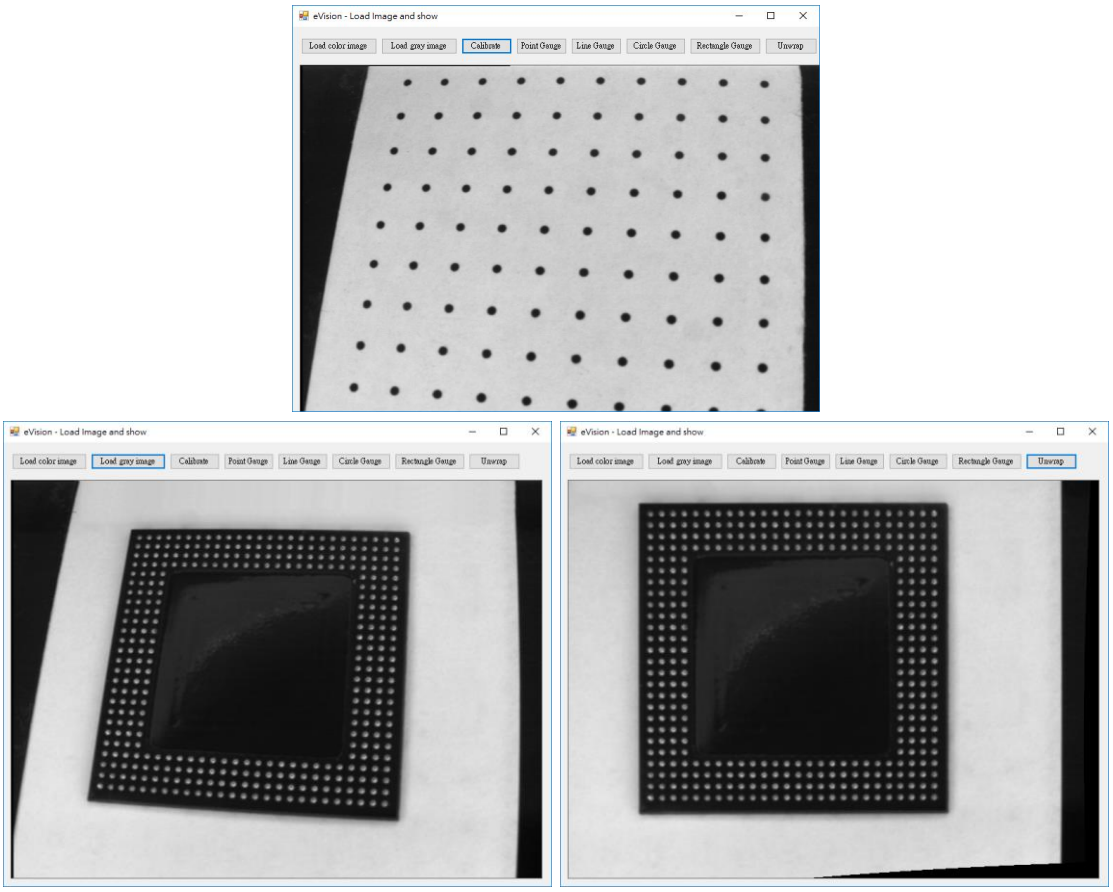
**Step 3: 載入扭曲的圖像**

- 從 [Unwarping] 選項中，點擊 [Open]，並且載入圖像檔：EasyGauge\Distorted component.tif.
- 保留預設的變數名稱，然後點擊 [OK]。

**Step 4: 反變形扭曲的圖像**

- 在目標圖像區域中，點擊 [New] 建立一個新圖像。
- 保持預設的圖像設定，然後點擊 [OK]。
- 選擇 [Interpolate] 複選框來提高產生圖像的品質。
- 點擊 [Unwarp] 進行反變形，可以看到在目標圖像中，所有扭曲的狀況都已經被校正了。

## 範例程式執行畫面




## 程式碼片段：

```
private void btnWrap_Click(object sender, EventArgs e)
{
   EBW8Image2.SetSize(EBW8Image1.Width, EBW8Image1.Height); //先準備一張相同大小的圖像
   EWorldShape1.Unwarp(EBW8Image1, EBW8Image2); //反變形
   ShowImage(EBW8Image2, pbImage);
}
```

## EasyOCR

[學習字元並建立 EasyOCR 字型]

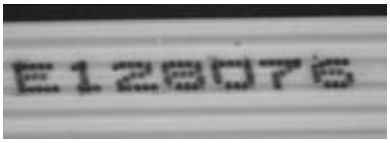
[根據字型識別字元]

## 學習字元並建立 EasyOCR 字型

### 目標

依照下面的操作流程，您將學習如何使用 EasyOCR 來學習新的字元，並將它們保存在 EasyOCR 的字型檔中。

首先，您必須先載入來源圖像 (step 1) ，然後，您將設定分割參數，以分割出每個字元 (step 2)，每一個字元都必須學習到 (step 3)，最後將所有學習到的字元保存在一個字型檔 (step 4)， 如果需要的話，您還可以將新的字元添加到現有字型檔中 (step 5)。



**來源圖像**



**分割圖像以檢測所有字元**



**已學習所有的字元**

### Step 1: 載入來源圖像

- 從主功能表中，點擊 [EasyOCR]，然後選取 [New OCR Tool]。
- 保留新 OCR 物件的預設變數名稱，並且點擊 [OK]。
- 在來源圖像的選項中，點擊來源圖像區域的 [Open] 圖示，然後載入圖像檔：**EasyOCR\FlatCable\FlatCable1.tif**
- 保留新圖像物件的預設名稱，並且點擊 [OK]。

**Step 2: 設定分割參數**

- 選擇 [Segmentation Parameters]，然後將圖像中的紅色方框移動到字元上方。
- 調整每個字元的屬性以使得分割後的綠色邊界框可以框住每個字元：
  參考閾值 (threshold value) = 113
  字元顏色 (characters color) = 白底黑字 (Black on White)
  最小寬度(min width) = 36
  最小高度(min height) = 31
  間距(spacing) = 4
  最大寬度 (max width) = 98
  最大高度 (max height) = 72
  雜訊區域 (noise area) = 9

**Step 3: 學習新的字元**

- 選擇 [Learn]，然後點擊來源圖像中的字元 E，隨後會提示您辨識字元以及其類型，在字元段落中輸入 E，並選擇 'EOcrClass_Uppercase' 的類型，點擊 [OK]，每當一個字元被添加到目前的字型中，其邊框就會變成黃色。
- 然後點擊來源圖像中的字元 1。在字元段落中輸入 1，並選擇 'EOcrClass_Digit' 的類型，點擊 [OK]。
- 依據上面步驟來繼續處理其他的字元。

**Step 4: 保存 EasyOCR 字型**

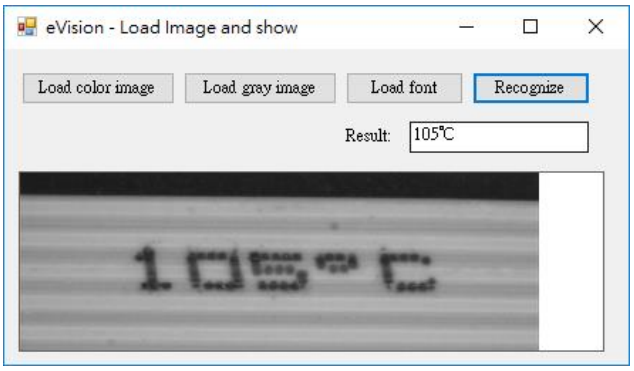
- 在選項 [Font File] 中點擊 [Save As...] 的按鈕，輸入新的 EasyOCR 字型檔的檔名，它的副檔名為 .ocr，最後點擊 [Save]。

**Step 5: 將字元添加到現有字體中**

- 在來源圖像的選項中，點擊 [Open] 的圖示，然後載入圖像檔：
  **EasyOCR\FlatCable\FlatCable2.tif**
- 保留新圖像物件預設的變數名稱，並且點擊 [OK]。

- 在 [Recognition] 頁面中，點擊 [Execute]，字元 2 和 8 被正確讀取，但 A，W 和 G 並沒有最低信心分數 (low confidence score)，並且它們不屬於在現有的字型檔中。
- 在選項 [Learn] 頁面中，學習字元 A，W 和 G (請參閱 step 3)。
- 再次將學習結果儲存到字型檔中 (請參閱 step 4)，這些新的字元就會被加入字型檔中。

執行畫面




程式碼片段：

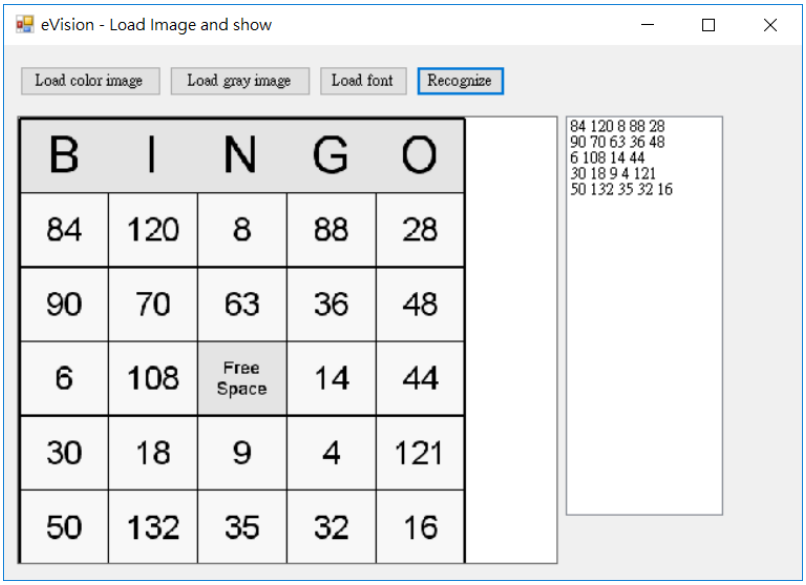
```
EOCR EOCR1 = new EOCR(); // EOCR instance
string recognizedChars;
```

```
private void btnLoadFont_Click(object sender, EventArgs e)
{
   openFileDialog1.Filter = "OCR file(*.OCR)|*.OCR|All files(*.*)|*.*";
   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      EOCR1.Load(openFileDialog1.FileName);
   }
}
```

```
private void btnRecognize_Click(object sender, EventArgs e)
{
```

```
    recognizedChars = EOCR1.Recognize(EBW8Image1,
20,(int)((int)EOCRClass.Digit+EOCRClass.UpperCase));
    //txtResult.Text = recognizedChars; //原本直接輸出，會變成"105蚓"
    //需要轉成bytes陣列，在轉回char，然後串成字串，才會出現正常的結果
    Byte[] bytes = System.Text.Encoding.Default.GetBytes(recognizedChars); //先把字串轉成bytes陣
列,GetString
    String str = "";
    for (int i = 0; i < bytes.Length; i++) //將每個byte顯示出來
    {
        Console.WriteLine((char)(bytes[i])+"("+bytes[i]+")"); //輸出到Console
        str += (char)(bytes[i]); //逐一串成字串
    }
    txtResult.Text = str;

}
```

練習題執行畫面




程式碼片段：

```
private void btnFont_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "*OCR file(.OCR)|*.OCR";
```

```
    //使用者選取OCR檔案
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        EOCR1.Load(openFileDialog1.FileName); //載入字型檔
    }
}

private void btnRecognize_Click(object sender, EventArgs e)
{
    //原本的Script: recognizedChars = EOCR1.Recognize(EBW8Image4, "256", "ocrClasses");

    recognizedChars = EOCR1.Recognize(EBW8Image1, 256, (int)EOCRClass.Digit); //256:最多辨識出幾個字元,
    //EOCR1.DrawChars(pbImage.CreateGraphics(),ScalingRatio); //每個辨識出來的字元繪製紅色框框

    int tmpCode,tmpX=0,tmpY,preX=0,preY=0;
    float tmpScore;
    int startI=0;
    String tmps = "";
    for (int i = 0; i < recognizedChars.Length; i++)
    {
        tmpCode = EOCR1.GetFirstCharCode(i); //回傳ASCII碼, '8'會回傳56, '0'是48
        tmpScore = EOCR1.GetConfidenceRatio(i); //取得信心指數
        tmpX = EOCR1.CharGetOrgX(i);
        tmpY = EOCR1.CharGetOrgY(i);
        if (i > 0 && Math.Abs(tmpX - preX) > (EOCR1.MaxCharWidth+EOCR1.CharSpacing)) //字元分開距離過大
        {
            if (Math.Abs(preY - tmpY) <= EOCR1.MaxCharHeight) //如果是同一行
            {
                tmps += recognizedChars.Substring(startI, i - startI) + " "; //只是加空白，串在同一行
                startI = i;
            }
            else
            {
                tmps += recognizedChars.Substring(startI, i - startI) + " "; //目前的不是同一行，也就是前一行最後一個數字，所以串在最後面
                listBox1.Items.Add(tmps); //加入listBox
                tmps = "";
                startI = i;
```

```
            }
        }
        Console.WriteLine("Code:" + (Char)tmpCode + " Distance:" + tmpScore);
        preX = tmpX; //把前一個字元X位置記錄下來，為了分辨數字是否分開
        preY = tmpY;
    }
    tmps += recognizedChars.Substring(startI, recognizedChars.Length - startI); //最後一個數字
    listBox1.Items.Add(tmps);
}
```

## 根據字型識別字元

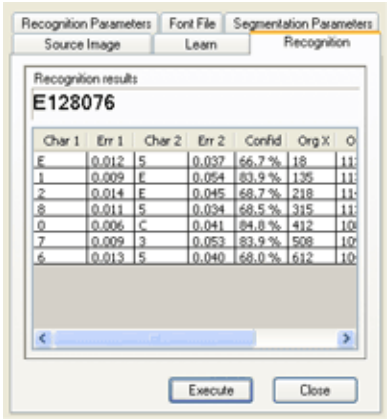
### 目標

依照下面的教學流程，您將會學習到如何使用 EasyOCR 來對於特定的字型進行字元辨識。

首先你需要載入來源圖像 (step 1) 以及 EasyOCR 的字型檔 (step 2)。 然後執行字元的識別 (step 3)。



**自動檢測與字型匹配的字元**




**顯示識別後的結果**

### Step 1: 載入來源圖像

- 在主功能表中，點擊 [EasyOCR]， 然後選擇 [New OCR Tool]。
- 保留 OCR 物件預設的變數名稱， 然後點擊 [OK]。
- 在來源圖像的標籤中， 點選來源圖像區域的 [Open] 圖示 ，然後載入來源圖像檔：**EasyOCR\FlatCable\FlatCable1.tif**

- 保留新圖像物件的預設名稱， 然後點擊 [OK]。

**Step 2: 載入字型檔**

- 在 [Font File] 頁面中，點擊 [Load]，然後選取下面的字型檔：**EasyOCR\FlatCable\FlatCable.ocr**，在圖像中，被檢測到的字元會以綠色進行標記。

**Step 3: 辨識字元**

- 在 [Recognition] 頁面中，點擊 [Execute] 觸發對檢測字元的識別，識別到的字元會出現在識別結果區域中，有關每個字元的其他資訊都列在表格中。

## EasyOCV

建立 EasyOCV 模型檔

根據模型檢測圖像中的字元

根據模型文件檢查 ROI 中的符號

學習統計模型的使用 (1)

學習統計模型的使用 (2)

## 建立 EasyOCV 模型檔

### 目標

依據下面的教學流程，您可以學到如何建立 EasyOCV 模型。首先，您需要載入一張圖像 (step 1)，接著建立模型 (step 2)，以及儲存這個模型 (step 3)。




**學習模型 (Learnt model)**

**Step 1: 載入一張圖像**

- 在主功能表中，點選 [EasyOCV]，接著點選 [New OCV Tool]。
- 保留預設的變數名稱，接著點選 [OK]。
- 在 [Learn] 頁面中點選 [Open] 的圖示，載入下列圖像檔：**EasyOCV\T20\T20 A.bmp**
- 保留預設的圖像物件變數名稱，接著點選 [OK]。

**Step 2: 建立模型**

- 在 [Learn] 頁面中，點擊 [Next] 三次，接著按下 [Finish]，便完成了建立模型的程序。

**Step 3: 儲存模型**

- 在 [Inspect] 頁面中，點選 [Save As...]，輸入一個新的 EasyOCV 模型名稱。EasyOCV 的模型將存成 .ocv 格式，最後點選 [Save] 來完成儲存動作。

## 根據模型檢測圖像中的字元

### 目標

根據下面的教學流程，您可以學習到如何載入 EasyOCV 模型，以及檢測圖像。

首先你需要載入一張圖像 (step 1) 以及載入模型 (step 2)，最後檢測並標記出字元 (step 3)。




**多個圖像上進行標記檢測的範例**

### Step 1: 載入一張圖像

- 在主功能表中，點選 [EasyOCV]，接著點選 [New OCV Tool]。
- 保留預設的變數名稱，接著點選 [OK]。
- 在 [Inspect] 頁面中，點選 [Open] 圖示，載入下列幾個圖像檔：
  **EasyOCV\T20\T20 A.bmp** 到 **T20 E.bmp**，按住 shift 可以選擇多個檔案。
- 保留預設的變數名稱，接著點選 [OK]。

### Step 2: 載入模型

- 在 [Inspect] 頁面中，點選 [Load] 按鈕，選擇下列圖像檔：**EasyOCV\T20\T20.ocv**

### Step 3: 標記檢測出的字元

- 在 [Inspect] 頁面中，點選 [Execute] 來開始檢測，檢測到的字元將被標示出來。
- 在 [Parameters and results] 頁面中，點選 [Character Overview] 或 [Text Overview] 來顯示該字元的詳細資料，點選 [Quality] 按鈕，可以選擇圖像中的字元來顯示品質指標。
- 在 [Inspect] 頁面中，點選 [Load Next] 或 [Load Previous] 的圖示來瀏覽多張圖像，在圖像 T20 E 中，可以看到字元 2 有印刷不全 (underprinted)、而字元 0 則是有重複印刷 (overprinted) 的狀況，都可以被檢驗出來。

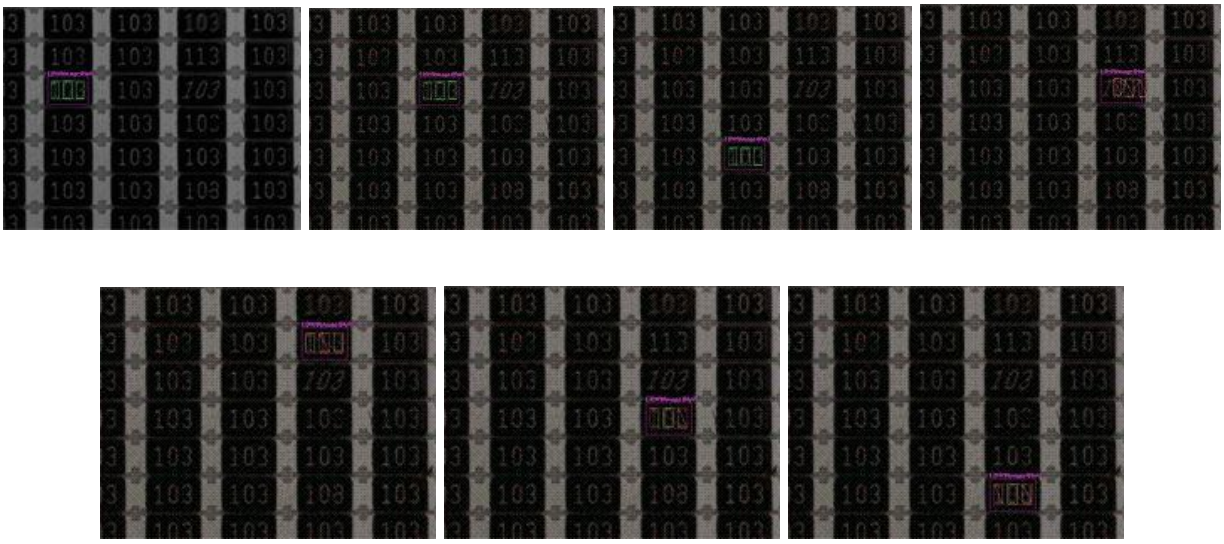
## 根據模型文件檢查 ROI 中的符號

### 目標

依據下面的教學流程，您將會學習如何載入一個 EasyOCV 模型文件，並在來源圖像的 ROI 中執行標記檢查。

你首先將要載入來源圖像，並限定 ROI 在哪裡進行標記檢查(steps 1-2)。

然後，你將載入模型文件(step 3)。最後，你將要執行這個標記文件(step 4)。



**在多個 ROIs 中進行檢查**

### Step 1: 載入來源圖像

- 從主選單點擊 [EasyOCV]，然後點選 [New OCV Tool]。
- 保留預設的變數名稱，然後點擊 [OK]。
- 在 [Inspect] 頁面中，點擊來源圖樣區域中的 [Open] 圖示，

  然後載入下面圖像檔：**EasyOCV\103\103.bmp**。
- 保留預設的變數名稱，然後點擊 [OK]。

**Step 2: 定義 ROI 感興趣區域**

- 使用滑鼠右鍵點擊圖像，然後從快顯功能表中選擇 [New ROI...]。
- 保留預設的變數名稱，然後點擊 [OK]。在圖像上放置預設的 ROI (藍色框)，也會顯示一個 ROI 的管理對話框。
- 分別輸入以下的座標到 ROI 的管理對話框：118、181、104、74 到 OrgX、OrgY、Width、和 Height。

**Step 3: 載入模型檔案**

- 在 [Inspect] 頁面中，點擊 [Load] 按鈕，然後選擇下列模型檔：
  **EasyOCV\103\103.ocv**

**Step 4: 執行標記檢查**

- 在 [Inspect] 頁面中，從來源圖像的下拉式表單裡選擇 ROI 物件，然後點擊 [Execute] 來觸發標記檢查，這時在來源圖像中的字元和符號都會被凸顯出來。
- 在 [Parameters and results] 頁面中，點擊 [Character Overview] 和 [Text Overview] 顯示出更多有關於字元和符號的資訊，點擊 [Quality] 按鈕，然後在圖像裡選擇符號來顯示對應的品質指標。
- 變更 ROI 位置 (OrgX, OrgY)，點擊 [Execute Inspection] ，在下面不同的位置上重複觸發檢查：
  - (288, 181): OK。
  - (288, 349): OK。
  - (462, 181): 數字傾斜。
  - (462, 95): 數字 0 是錯誤的。
  - (462, 266): 數字 3 破掉了。
  - (462, 438): 數字 3 變形了。

## 學習統計模型的使用 (1)

### 目標

依照下面的教學流程，您將學習如何使用 EasyOCV 學習 EasyOCV 模型，並使用統計資訊對其進行調整。

您需要首先載入參考圖像 (step 1)， 並讀取模型 (step 2)。 然後您將在模型位置上添加值 (step 3)，並使用多個圖像的統計數據調整模型 (step 4)。最後， 您將對多個圖像執行標記檢查 (step 5)。




**在多個圖像中標記檢測**

### Step 1: 載入參考圖像

- 從主功能表中， 點擊 [EasyOCV]，然後點擊 [New OCV Tool]。

- 保留預設的變數名稱， 然後點擊 [OK]。
- 在 [Learn] 標籤， 點擊來源圖像區域中的 [Open] 圖示，並載入圖像檔：EasyOCV\ISE\ISE 01.bmp。
- 保留預設的變數名稱，然後點擊 [OK]。

**Step 2: 學習模型**

- 在 [Learn] 標籤， 點擊 [Next] 編輯分段對象。字元 "ASS"， "MB" 和 "an" 編成一組 (用相同的顏色分段)，他們需要被分隔開。
- 在圖像中， 畫線以分隔出這些個別字元。
- 點擊 [Next] 兩次，然後點擊 [Finish]，表示學習過程已完成。

**Step 3: 為模型添加容忍值**

- 在 [Parameters and results] 標籤， 點擊在 “Selected texts parameters”區域中的 [Position] 。
- 爲了提供位移量的自由度，請在 Tolerance(容忍值)輸入 80 和 60 作為 ShiftX 和 ShiftY 的容忍值。
- 點擊 [Close]，實現基本的 EasyOCV 模型的構建。

**Step 4: 使用統計來調整模型**

- 在 [Inspect] 標籤，點擊 [Open] 來源圖像區域的圖示，使用 shift 鍵選擇多個檔案後載入下列圖像檔：EasyOCV\ISE\ISE 01.bmp ~ ISE 05.bmp。
- 保留預設變數名稱並點擊 [OK]。
- 在 [Inspect] 標籤，點擊 [Add to statistics]。點擊 [Load Next] 圖示，然後點擊 [Add to statistics]，然後如此在所有的圖像上。
- 在 [Parameters and results] 標籤，點擊 [Quality] 統計公差調整區域的按鈕。
- 保持提示框中的預設值。點擊 [OK]。

**Step 5: 檢查多個圖像**

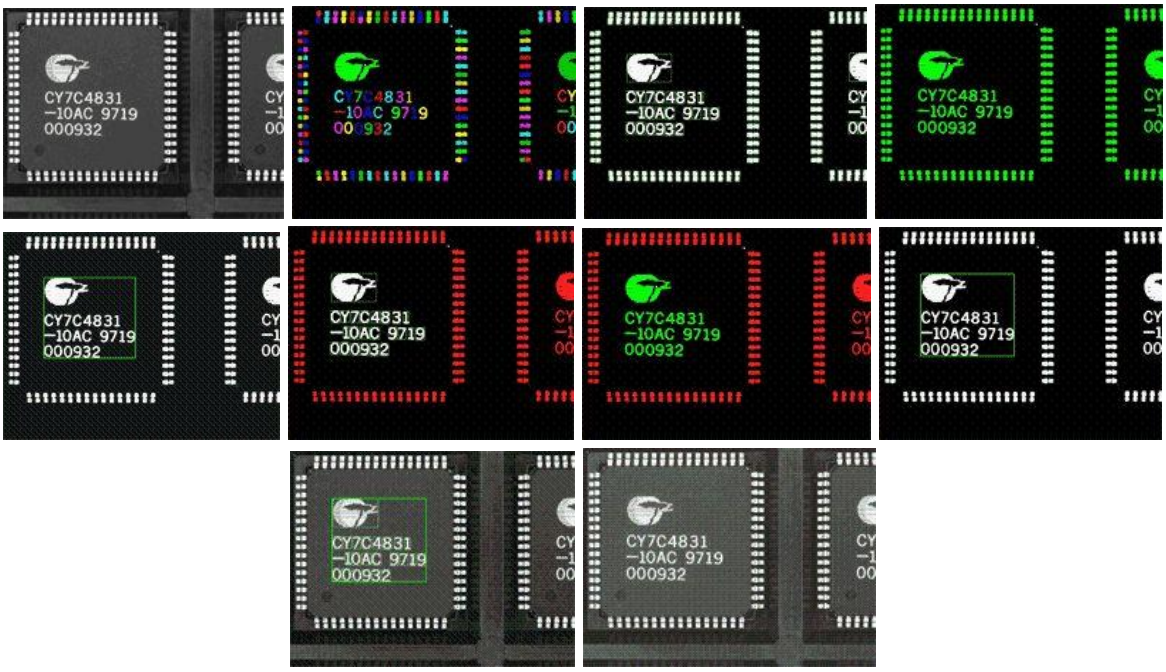
- 在 [Inspect] 標籤，點擊 [Open] 源圖像區域的圖示，使用 shift 鍵選擇多個文件後載入下列圖像檔：**EasyOCV\ISE\ISE 01.bmp ~ ISE 14 (bad character).bmp**。
- 保留新圖像物件的預設變數名稱，然後點擊 [OK]。
- 在 [Inspect] 標籤，點擊 [Load Next] 和 [Load Previous] 圖示瀏覽多個圖像。注意以下觀察結果。
  - 在圖像 ISE 01 到 ISE 05 的都沒有缺陷。
  - 在圖像 ISE 06 中，PBGA 打印不全。
  - 在圖像 ISE 07 中，標記變形。
  - 在圖像 ISE 08 中，標記模糊不清。
  - 在圖像 ISE 09 中，字元 5 和 2 不正確。
  - 在圖像 ISE 10 中，字元打印過度以及不良。
  - 在圖像 ISE 11 中，字元 388 打印顏色太淺。
  - 在圖像 ISE 12 中，字元 ISE 被打印過度，顏色太深。
  - 在圖像 ISE 13 中，缺少 CA 兩個字元。
  - 在圖像 ISE 14 中， A 字元打印破損。

## 學習統計模型的使用 (2)

### 目標

依照下面的教學流程，您將學習到如何使用 EasyOCV 的統計特徵和旋轉容差來構建模型文件。

你需要先載入範例圖像 (step 1)，並學習模型 (text and logo) (step 2)。 然後在模型位置上添加公差 (step 3)，並使用多個圖像的統計數據來調整模型 (step 4)。 最後，您將對多個圖像執行標記檢查 (step 5)。




**學習字元和標誌**

### Step 1: 載入範例圖像

- 從主選單上點選 [EasyOCV] 然後點選 [New OCV Tool]。
- 保持預設名稱並按下 [OK]。
- 在 [Learn] 選項中，點選圖像來源區域的 [Open] 圖示，並載入圖檔 **EasyOCV\CY7C\CY7C 01.jpg**。

- 保持預設名稱並按下 [OK]。

**Step 2: 學習模型**

- 在 [Learn] 選項中， 點選 [Next] 兩次後，將會出現字元對話方塊。
- 點選 [Undo Char(s)]，接下來標誌和 PING 腳將會特別明顯標示為綠色。
- 使用你的滑鼠游標，點擊並拖曳矩形到字元和徽標的上面。
- 點選 [Form Auto Chars]。
- 點選 [Next]，然後點選 [Finish]。此學習處理已經完成。

**Step 3: 為模型添加公差**

- 在 [Parameters and results] 選項中，點選在 "Selected texts parameters" 區域中的 [Position]。
- 為了能自由的平移翻轉，輸入 90 和 80 作為 ShiftX 和 ShiftY 的公差。
- 為了平移和旋轉，請輸入 5 作為斜度容差，並輸入 11 作為斜度/數量。
- 要加快定位過程，請輸入 7 作為 ShiftX 和 ShiftY 步幅/計數。
- 點擊[close]。在這個階段，建立基本的 OCV 模型已經實現。

**Step 4: 使用統計來調整模型**

- 在 [Inspect] 選項中，點選圖像來源區域的 [Open] 圖示，使用 shift 鍵選取多個檔案來載入下列檔案： **EasyOCV\CY7C\CY7C 01.jpg ~ CY7C 15.jpg**。
- 保持預設名稱並按下 [OK]。
- 點擊 [Add to statistics]。點擊 [Load Next] 圖示，然後點選 [Add to statistics]，並等待所有的圖像。
- 在 [Parameters and results] 選項中，點擊統計公差調整區域中的 [Quality] 按鈕。
- 將安全係數設置為 3.5，然後點擊 [OK]。

**Step 5: 檢查多個圖像**

- 在 [Inspect] 選項中， 點選圖像來源區域的 [Open] 圖示，使用 shift 鍵選取多個檔案來載入下列檔案：**EasyOCV\CY7C\CY7C 01.jpg ~ CY7C 27.jpg**。
- 將新的圖像物件保留預設名稱，以及點選 [OK]。
- 在 [Inspect] 選項中，點擊 [Load Next] 和 [Load Previous]圖示來瀏覽多個圖像，觀察檢測到的預設值。

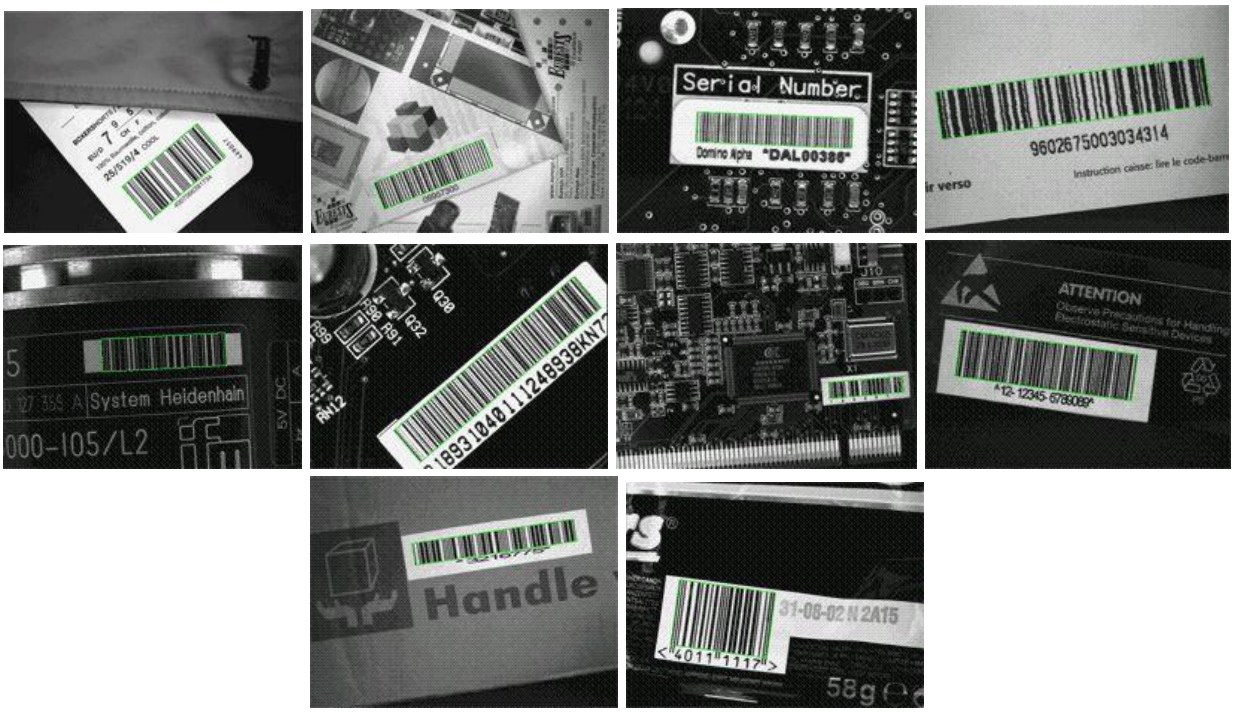
自動讀取條碼

# 自動讀取條碼

## 目標

依照下面的教學流程 ，您將學習如何進行多個一維條碼(bar codes)的讀取。

首先，需要先載入多個包含條碼的來源圖像 (step 1)，接著，自動執行每個圖像的條碼讀取 (step 2)。




**每個條碼都會被自動檢測並解碼**

### Step 1: 載入來源圖像

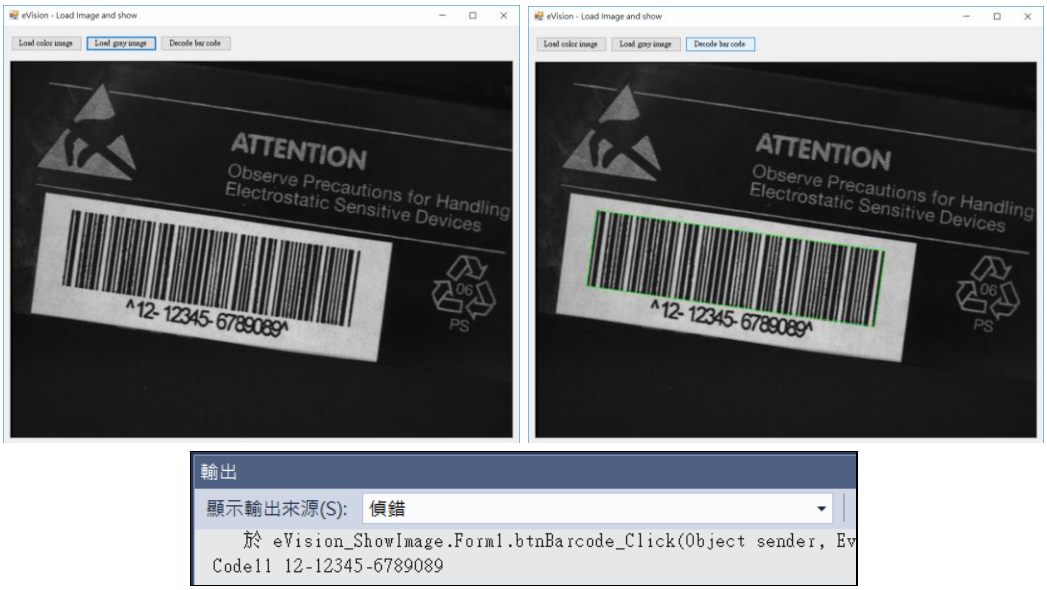
- 從主選單選取 [EasyBarCode] 下的 [New BarCode Tool]。
- 使用預設的變數名稱，直接按下 [OK]。

- 在 [AutoRead] 選項中點選 Source Image 區域中的 [Open] ，可以使用 Shift 鍵來選擇多個圖檔，將下列圖檔全部載入：**EasyBarCode\Barcode 01.tif ~ Barcode 10.tif 。**
- 使用預設的物件名稱以及點選 [OK]，就會出現這些來源圖像。

**Step 2: 自動讀取一維條碼**

- 條碼會自動檢測並解碼，結果會出現在圖像上，而條碼格式與相應的數值將顯示在 Decoded Symbology (解碼符號系統) 區域中 ，一旦出現新圖像，不需要點擊 [Read]。 但是，點擊 [Read] 可以在腳本視窗中插入相應的程式碼 。
- 在 [Results] 選項中，查看更多有關條碼的資訊，由於條碼在不同的符號系統下可能會有不同的意思，因此為了避免混淆的情況下列出可能的結果供使用者參考。
- 在 [AutoRead] 選項中，點擊 [Load Next] 和 [Load Previous] 圖示來瀏覽圖像 01 至 06。這時會出現對應圖像，並自動檢測與解碼每個條碼，並且更新條碼的屬性。
- 要解碼其他的條碼圖像，我們必須啟用其他的符號系統。
- 在 [Symbologies ] 的頁面中，點擊額外附加區域的 [Toggle All] 按鈕。
- 在 [AutoRead] 標籤中，點擊 [Load Next] 和 [Load Previous] 圖示來瀏覽其它圖像的條碼讀取結果。

## 範例程式執行畫面




## 程式碼片段：

```
private void btnBarcode_Click(object sender, EventArgs e)
{
   try
   {
      EBarCode1.AdditionalSymbologies += (int)ESymbologies.Code11; //指定增加code11進行解碼
      EBarCode1.AdditionalSymbologies += (int)ESymbologies.Code25Datalogic;
      //一般standard的解碼寫法:
      EBarCode1.Detect(EBW8Image1); //進行條碼偵測
      string str = EBarCode1.Decode(EBarCode1.GetDecodedSymbology(0)); //依據偵測到的符號系統進行解碼
      Console.WriteLine(EBarCode1.GetDecodedSymbology(0) + " " + str); //輸出結果
      EBarCode1.Draw(pbImage.CreateGraphics());//畫出結果
   }
   catch (Exception ex)
   {
      Console.WriteLine("Unable to decode. "+ ex.ToString());
   }
}
```

自動讀取二維條碼

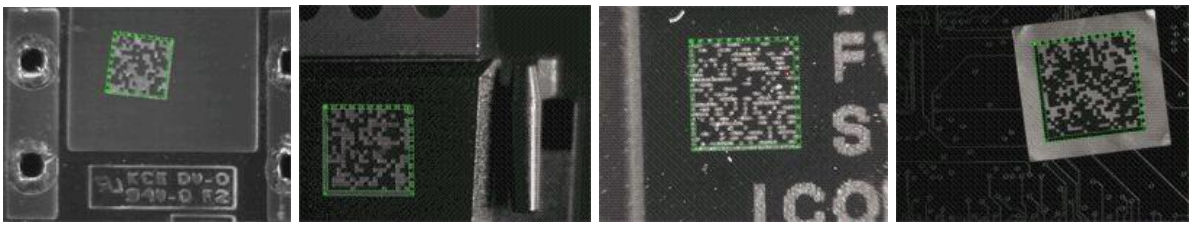
學習二維條碼建立 EasyMatrixCode model 檔

# 自動讀取二維條碼

## 目標

依照下面的教學流程，您將學會如何使用 EasyMatrixCode 從多個檔案中檢測並自動地解碼 Data Matrix 的資料。

首先，載入多個來源圖像 (step 1)，然後自動執行每個圖像的解碼 (step 2)，你還可為每個二維條碼的印刷品質分級 (step 3)。



**每個 Data Matrix 代碼會自動地檢測並解碼**

### Step 1: 載入來源圖像

- 從主功能表中，點選 [EasyMatrixCode] 接著點選 [New MatrixCode Tool]。
- 使用預設的變數名稱，然後點選 [OK]。
- 在 [Read] 頁面中，點選來源圖像區域的 [Open] 圖示，然後載入下列圖檔，可以使用 Shift 鍵選取多個檔案：

  **EasyMatrixCode\AutoRead\AutoRead 01.tif~AutoRead 04.tif**
- 使用圖像預設的變數名稱，然後點選 [OK]，圖像就會出現在畫面上。

### Step 2: 自動讀取二維條碼

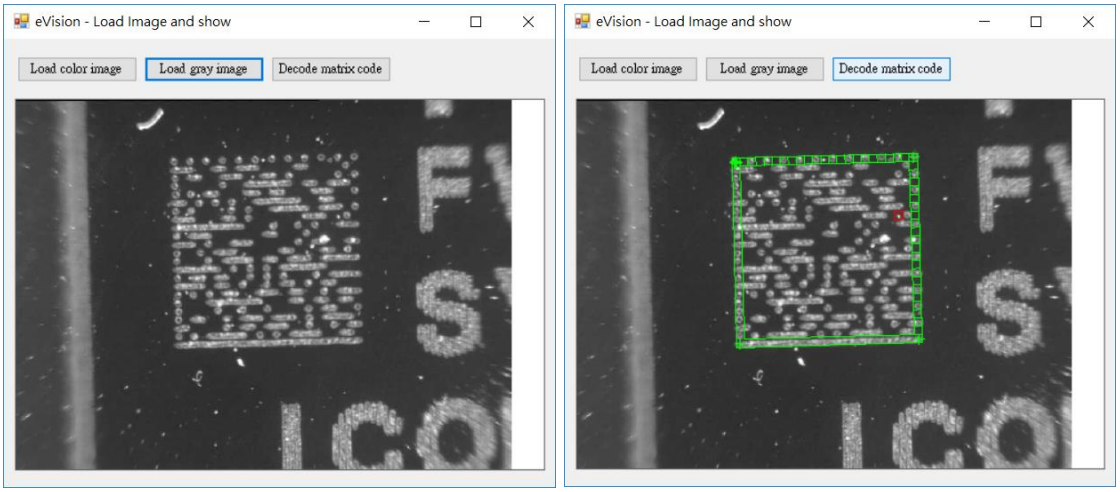
- 二維條碼會自動檢測並解碼，從十字架所標記的地方設定二維條碼的亮度，一旦出現新圖像就不需要點選 [Read]。點選 [Read] 時也會插入對應的程式碼在腳本視窗內。
- 在畫面右邊 [Results] 區域中可以找到更多二維條碼的相關訊息，例如：解碼出來的字串。

- 在 [Read] 頁面中，點選 [Load Next] 及 [Load Previous] 圖示，圖像會進行切換，然後每個圖像會自動檢測並解碼，二維條碼的屬性也會進行更新，如果圖像中沒有二維條碼，則訊息欄會顯示錯誤訊息。

**Step 3: 為二維條碼的品質進行分級** (Open eVision Studio 2.2.2 版已移除)

- 在 [Print Quality] 頁面中，選取 [Compute Grading] 的複選框，然後點選 [Apply]。對於每個二維條碼的品質都會顯示相應的值及等級， A 級代表高品質，而 F 級則代表低品質。

範例程式執行畫面




程式碼片段：

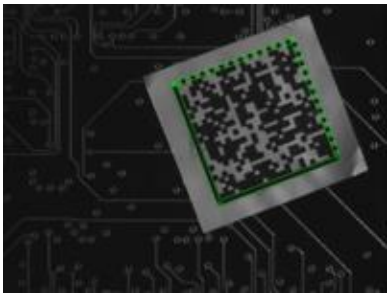
```
private void btnDecode_Click(object sender, EventArgs e)
{
   EMatrixCodeReader1Result=EMatrixCodeReader1.Read(EBW8Image1); //讀取二維條碼
   EMatrixCodeReader1Result.Draw(pbImage.CreateGraphics(), ScalingRatio); //有支援指定縮放比例的繪製
   EMatrixCodeReader1Result.DrawErrors(pbImage.CreateGraphics(), ScalingRatio); //將有錯誤的地方標示出來
   Console.WriteLine(EMatrixCodeReader1Result.DecodedString); //輸出結果
}
```

## 學習二維條碼建立 EasyMatrixCode model 檔

### 目標

依照下面的教學流程，您將可以了解如何使用 EasyMatrixCode 來學習二維條碼 (Data Matrix)，並且將之儲存在 EasyMatrixCode 的 model 檔中。

首先，您需要載入一個來源圖像 (step 1)，接著學習圖像上的二維條碼 (step 2)。然後，你將會儲存所學習到的二維條碼到一個 model 檔中 (step 3)。如果需要的話，你也可以新增一個或多個學習好的二維條碼到現在的 model 檔中 (step 4)。



**已經學習好的二維條碼**

### Step 1: 載入來源圖像

- 從主功能表中， 按下 [EasyMatrixCode]，接著按下 [New MatrixCode Tool]。
- 使用預設的變數名稱，然後按下 [OK]。
- 在 [Learn] 頁面中，按下 Source Image 區域中的 [Open] 圖示，並載入下面的圖檔： **EasyMatrixCode\Label\Label 4.tif**
- 使用預設的圖像物件變數名稱，然後按下 [OK]。

### Step 2: 學習二維條碼

- 在 [Learn] 的頁面中，按下 [Learn]，圖像中的二維條碼會被偵測並且正確無誤地解碼，圖形結果會顯示在圖像上面，學習好二維條碼的屬性都會更新到對話方塊上。.

**Step 3: 儲存 model 檔**

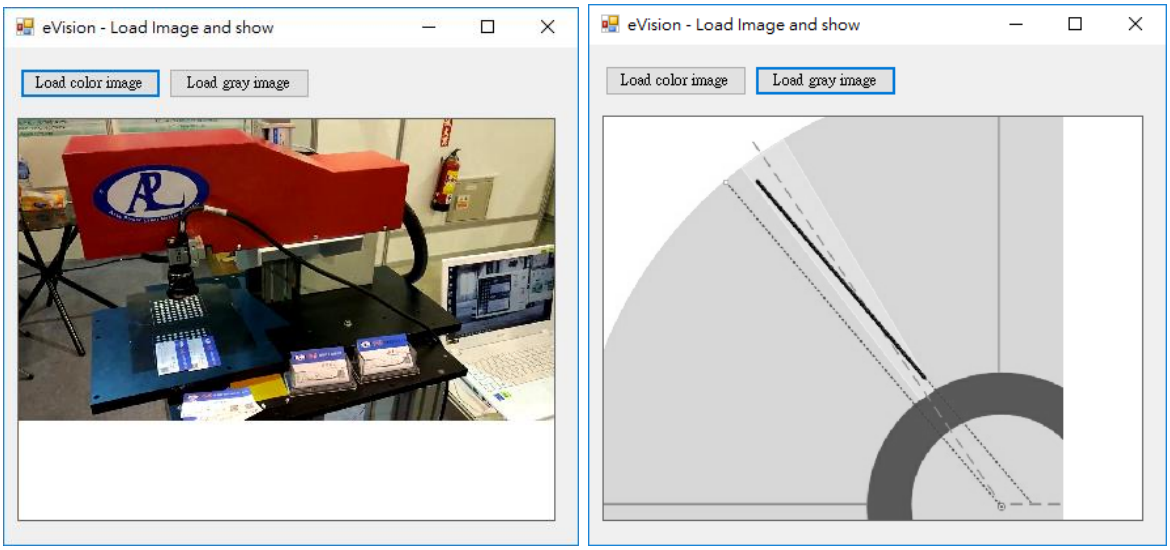
- 在 [Learn] 頁面中，按下 [Save As...] 按鍵，輸入一個 EasyMatrixCode model 的新檔名，副檔名會是 .mx2，最後，按下 [Save]。

**Step 4: 學習更多的二維條碼**

- 在 [Read] 頁面中，按下 Source Image 區域中的 [Open] 圖示，並載入下面的圖檔：**EasyMatrixCode\PCB Code\PCB Code 3.jpg**
- 使用預設的圖像物件變數名稱，然後按下 [OK]。
- 當 [Read] 頁面中的 Message 區域顯示一個錯誤訊息時，表示該二維條碼無法讀取，主要是因為目前讀取器使用來自於 "Label 4" 圖像的 model。你需要學習 "PCB Code 3" 的二維條碼，並加入 model 後才能正確讀取。
- 在 [Learn] 頁面中，按下 [Learn More]，圖像中的二維條碼會被偵測並且正確無誤地解碼，圖形結果會顯示在圖像上面，學習好二維條碼的屬性都會更新到對話方塊上。
- 使用 [Learn More] 而不使用 [Learn] 的話，會導致 "Label 4" model 不會被 "PCB Code 3" model 所取代，而且兩個 models 會同時被包含在相同的 model 檔中。在 [Read] 頁面中，"PCB Code 3" 的二維條碼可以正確的讀取出來。在來源圖像的下拉式選單中，選擇 "Label 4" 圖像，"Label 4"的二維條碼仍然可以正確無誤地讀取，這樣就表示兩個學習好的二維條碼都有被保存在 model 檔中。
- 最後，再次儲存此 model (請參考 step 3)，這樣新的二維條碼就被加入 model 檔中了。

## 附錄Ａ：基本載入與顯示圖像範例程式碼

本範例提供載入彩色圖像或者灰階圖像，並且顯示到 PictureBox 上的 C#範例，可以作為開發相關應用的基本專案。




程式碼：

```
//eVision-Show Image Sample
//Function: 1. Load Image
//          2. convert Bitmap to EImageC24
//          3. Show EImageC24 Image to picturebox
//Enviornment: VS 2012, eVision 2.2.2
//Caution: using eVision need to run [Visual Studio] or [compiled exe] as administrator, if not it
will not able to get the grant from license dongle.
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Euresys.Open_eVision_2_2;

namespace eVision_ShowImage
{
    public partial class Form1 : Form
    {
```

```
        Bitmap bitmap = null;
        BitmapData bmpData = null;
        EImageC24 EC24Image1 = new EImageC24(); //eVision的彩色圖像物件
        EImageBW8 EBW8Image1 = new EImageBW8(); //eVision的灰階圖像物件
        float ScalingRatio = 0; //Picturebox與原始影像大小的縮放比例
        float EBW8ScaleRatio;

        public Form1()
        {
            InitializeComponent();
        }

        private void button_load_Click(object sender, EventArgs e)
        {
            //使用者選取Bitmap檔案
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                //第一種載入作法:
                ////讀取檔案的Bitmap (一般這一行也可以改成從相機的影像進行讀取)
                //Bitmap bitmap_source = (Bitmap)Image.FromFile(openFileDialog1.FileName);
                ////當bitmap尚未創建記憶體空間時直接使用Clone複製來源影像
                //if (bitmap == null)
                //    bitmap = (Bitmap)bitmap_source.Clone(); //複製完整的記憶體空間
                ////指派影像
                //bitmap = bitmap_source;

                ////防止來源影像為空則不處理
                //if (bitmap == null)
                //    return;
                ////Bitmap影像轉換eVision格式影像
                ////EC24Image1 = BitmapToEImageC24(ref bitmap);

                //第二種載入作法:
                EC24Image1.Load(openFileDialog1.FileName); //直接載入

                //顯示影像於Picturebox
                ShowImage(EC24Image1, pbImage);

                //因Bitmap容易造成記憶體外洩，因此將資料從記憶體中Dispose，並且進行GC
                //bitmap.Dispose(); //第一種作法要記得取消記憶體的空間
                System.GC.Collect();
                System.GC.WaitForPendingFinalizers();
            }
        }

        private void btnGray_Click(object sender, EventArgs e)
        {
            //使用者選取Bitmap檔案
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                EBW8Image1.Load(openFileDialog1.FileName); //直接載入
```

```
            //顯示影像於Picturebox
            ShowImage(EBW8Image1, pbImage);
        }
    }

    /// <summary>
    /// Bitmap影像轉換eVision格式影像
    /// </summary>
    /// <param name="bitmap">來源Bitmap影像</param>
    /// <returns>轉換後的EImageC24影像</returns>
    private EImageC24 BitmapToEImageC24(ref Bitmap bitmap)
    {
        EImageC24 EC24Image1 = null;
        try
        {
            EC24Image1 = new EImageC24();

            Rectangle rect = new Rectangle(0, 0, bitmap.Width, bitmap.Height);
            bmpData =
                bitmap.LockBits(rect, System.Drawing.Imaging.ImageLockMode.ReadWrite,
                bitmap.PixelFormat);

            EC24Image1.SetImagePtr(bitmap.Width, bitmap.Height, bmpData.Scan0);
            bitmap.UnlockBits(bmpData);

        }
        catch (EException e)//EException為eVision的例外處理
        {
            Console.WriteLine(e.ToString());
        }
        return EC24Image1;
    }

    /// <summary>
    /// 顯示影像於Picturebox
    /// </summary>
    /// <param name="img">顯示的eVision影像</param>
    /// <param name="pb">目標Picturebox</param>
    private void ShowImage(EImageC24 img, PictureBox pb)
    {
        try
        {
            Bitmap bmp;
            bmp = new Bitmap(pb.Width, pb.Height);

            //計算Picturebox與顯示影像的比例，以便將影像縮放並且完整呈現到picturebox上。
            float PictureBoxSizeRatio = (float)pb.Width / pb.Height;
            float ImageSizeRatio = (float)img.Width / img.Height;
            if (ImageSizeRatio > PictureBoxSizeRatio)
                ScalingRatio = (float)pb.Width / img.Width;
            else
                ScalingRatio = (float)pb.Height / img.Height;
```

```
                //委派
                if (pb.InvokeRequired)
                {
                    pb.Invoke(new MethodInvoker(delegate() { img.Draw(Graphics.FromImage(bmp),
ScalingRatio); pb.BackgroundImage = bmp; }));
                }
                else
                {
                    //先將EImageC24畫在bitmap模式的圖檔
                    img.Draw(Graphics.FromImage(bmp), ScalingRatio);
                    //再由Picturebox取得bitmap
                    pb.BackgroundImage = bmp;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
        private void ShowImage(EImageBW8 img, PictureBox pb)
        {
            try
            {
                Bitmap bmp;
                bmp = new Bitmap(pb.Width, pb.Height);

                //計算Picturebox與顯示影像的比例，以便將影像縮放並且完整呈現到picturebox上。
                float PictureBoxSizeRatio = (float)pb.Width / pb.Height;
                float ImageSizeRatio = (float)img.Width / img.Height;
                if (ImageSizeRatio > PictureBoxSizeRatio)
                    ScalingRatio = (float)pb.Width / img.Width;
                else
                    ScalingRatio = (float)pb.Height / img.Height;
                EBW8ScaleRatio = ScalingRatio; //記錄此比例供繪製輪廓軌跡使用
                //委派
                if (pb.InvokeRequired)
                {
                    pb.Invoke(new MethodInvoker(delegate () { img.Draw(Graphics.FromImage(bmp),
ScalingRatio); pb.BackgroundImage = bmp; }));
                }
                else
                {
                    //先將EImageC24畫在bitmap模式的圖檔
                    img.Draw(Graphics.FromImage(bmp), ScalingRatio);
                    //再由Picturebox取得bitmap
                    pb.BackgroundImage = bmp;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
```

```
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            openFileDialog1.InitialDirectory = Application.StartupPath;
        }
    }
}
```