EzCheck 影像函式庫

使用說明書

(中文版)

版本: 1.0.0

改版日期: 2011-09-19

EzCheck影像函式庫

使用說明書

產品保固

凡泓格科技股份有限公司產品從購買即日起若無任何材料性缺損保固一年。

免責聲明

凡使用本系列產品除產品質量所造成的損害，泓格科技股份有限公司不承担任何法律責任。泓格科技股份有限公司有義務提供本系列產品可靠而詳盡的資料，但保留修定權利，且不承担使用者非法利用資料對第三方所造成侵害構成的法律責任。

版權

商標

手冊中所涉及所有公司商標，商標名稱以及產品名稱分別屬於該商標或名稱的擁有者所有

版本控管

版本	日期	作者	說明
1.0.0	2011/9/19	Clark Tsai	Release.

目錄

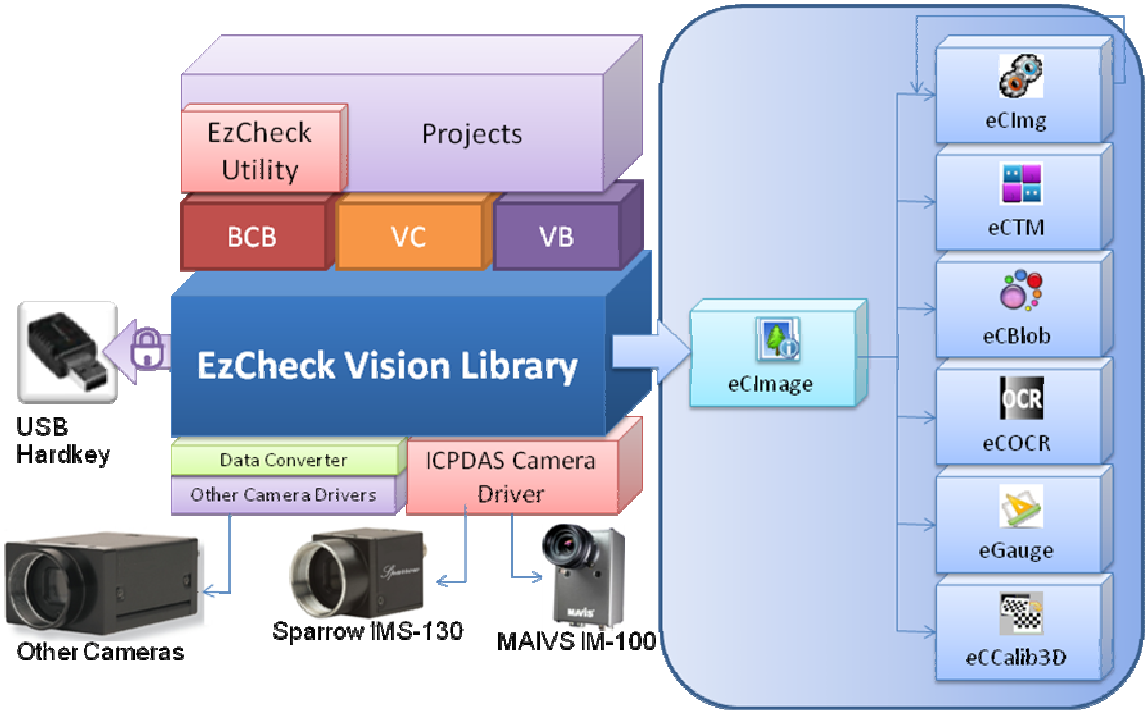
1.EzCheck

1.1. EzCheck 簡介

EzCheck Vision Library 是 ICPDAS 提供之「影像處理與檢測函式庫」，旨在幫助使用者快速簡單的建立自己的機器視覺系統軟體。

EzCheck Vision Library 支援多種影像格式的存取，提供各種常見的基本影像處理及多種可選擇的進階功能，如：光學文字識別、影像校正、樣版比對、測量...等。

配合 EzCheck Vision Library 簡單易懂的功能與相關的工具軟體以及範例介面，客戶將可以很快的學會如何使用 EzCheck Vision Library 來進行開發。



圖一、 EzCheck Viosn Library 架構圖

1.2. EzCheck 主要特點

提供七個主要元件，幫助使用者快速地開發機器視覺系統。

- EzCheck 影像(EzCheck Image)：eCImage。
- 影像處理群組(Image Processing)：eCImg Group。
- 樣板比對(Template Matching)：eCTM。
- Blob 分析(Blob Analysis)：eCBlob。
- 光學文字辨識(Optical character recognition)：eCOCR。
- 立體影像校正(3D Calibration)：eCCalib3D。
- 影像測量(Gauge/Measure)：ePointGauge, eLinegauge, eCircleGauge, eRectangleGauge。

提供工具軟體讓使用者測試與驗證– EzCheck Utility

泓格提供以 EzCheck Vision Library 所開發的工具軟體-EzCheck Utility，藉由這套免費下載使用的工具軟體(使用者不需 USB 硬體鎖也可使用 EzCheck Utility)，使用者可以更完整地了解 EzCheck Vision Library 帶來的功能，也可利用此工具軟體測試評估 EzCheck Vision Library 是否有能力達到應用需求。

USB 硬體鎖保護與分級套件

為了維護購買 EzCheck Vision Library 使用者的權益，每一套 EzCheck Vision Library 都會以 USB 硬體鎖進行保護與控管。只有在安插了 USB 硬體鎖的電腦上，EzCheck Vision Library 才會完成各項任務。

除了 library 的保護與控管之外，USB 硬體鎖也提供套件分級的功能。EzCheck Vision Library 依照功能的劃分，共有五個套件等級讓使用者依照自己的需求進行選擇。

EzCheck Vision Library 套件支援列表

套件	eCImage	eCTM	eCBlob	eCGauge	eCOCR	eCCalib3D
EzCheck-A	V	V	V	V		
EzCheck-B	V	V	V		V	
EzCheck-C	V		V	V		V
EzCheck-D	V	V		V		V
EzCheck-ALL	V	V	V	V	V	V

提供多個平台上的範例程式

為了讓使用者能夠快速的學習如何使用 EzCheck Vision Library 開發機器視覺系統，泓格提供開放原始碼的範例程式供使用者參考。範例程式包含簡易的專案與介面，並且展示 EzCheck Vision Library 各項主要功能的使用方式。

目前已提供 Borland C++ Builder 6.0 以及 Visual C++ 6.0 開發平台上的範例程式。

支援攝影機取像

使用者可以透過 EzCheck Vision Library 直接取得攝影機的影像進行處理，不需要繁瑣的影像轉檔，軟硬體的整合將會更有效率。

2.使用 EzCheck Vision Library

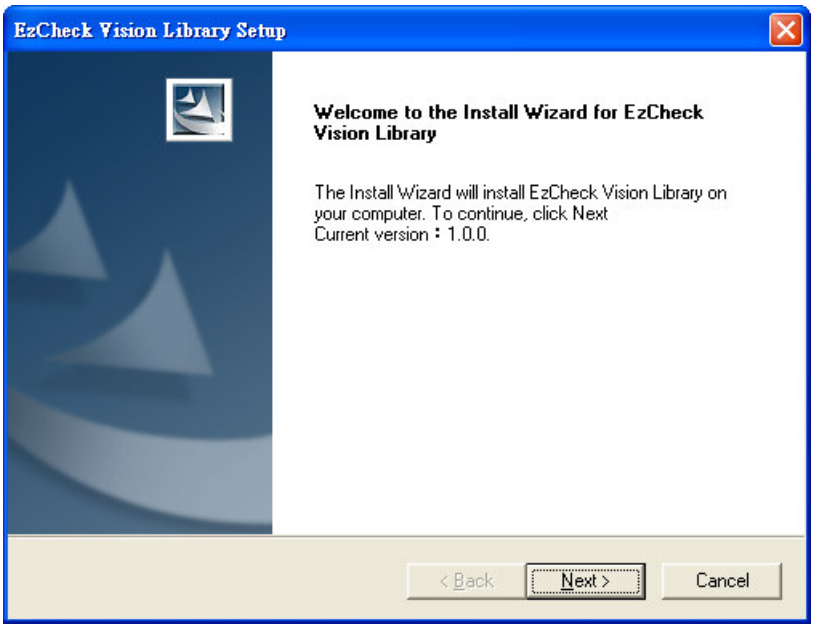
本章節將循序漸進帶領使用者安裝並且使用 EzCheck Vision Library。EzCheck Vision Library 包含開發 Borland C++ Builder 與 Visual C++專案所需的檔案、EzCheck 工具軟體以及 EzCheck Vision Library 使用手冊。

2.1. 安裝 EzCheck Vision Library

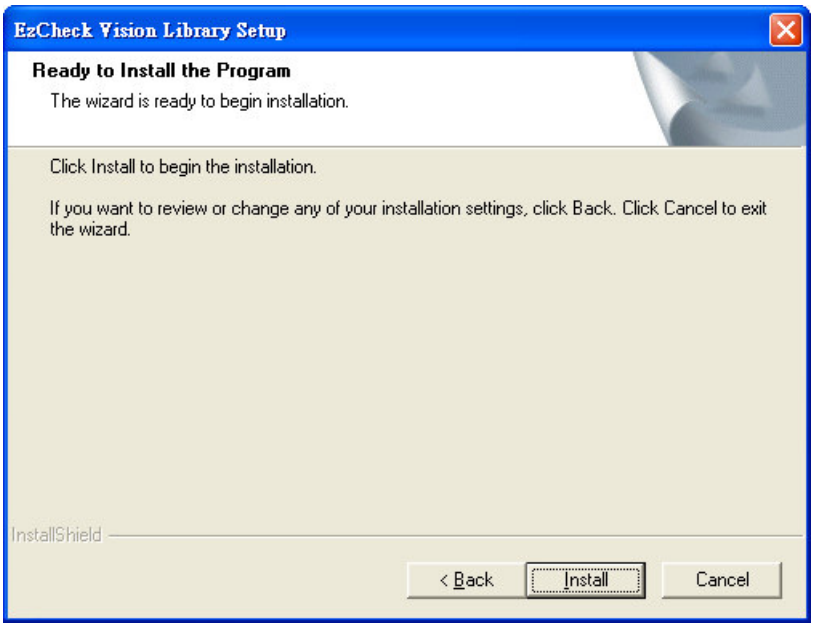
使用者可從網頁上或者光碟中取得 EzCheck Vision Library 的安裝程式：EzCheck_Install_v*.*.*.exe(v*.*.*為版本編號)。

網頁連結：
http://www.icpdas.com.tw/product/solutions/software/development_tools/ezcheck/ezcheck_introduction.html
下載連結：
http://ftp.icpdas.com/pub/cd/EzCheck_cd/

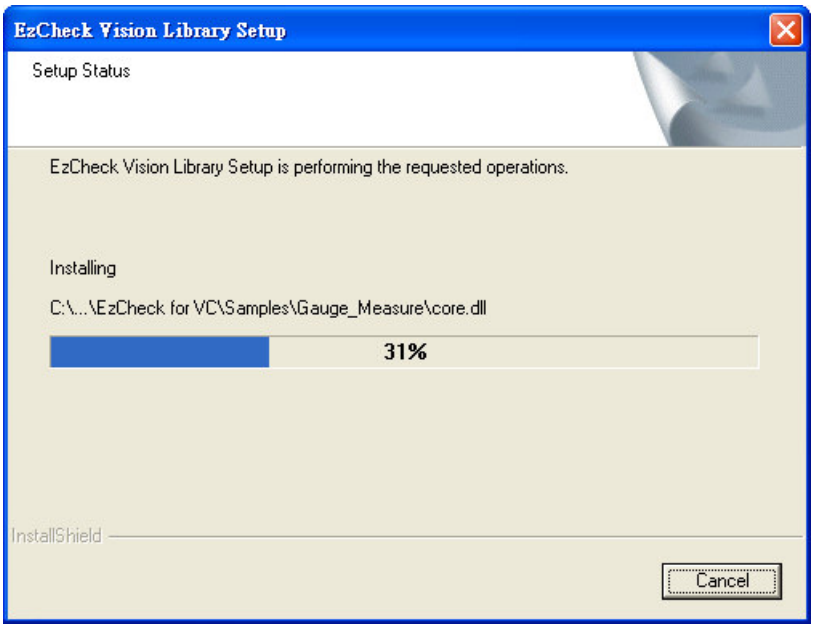
執行畫面如下：



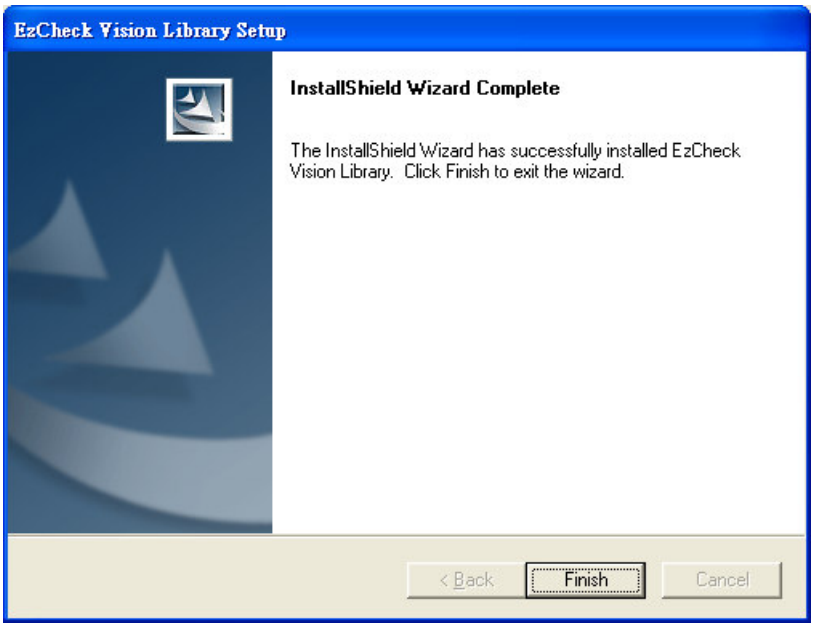
圖二、 EzCheck 安裝歡迎畫面。由畫面中可見，目前版本為 1.0.0



圖三、 確認將要安裝 EzCheck 至電腦中



圖四、 正在將 EzCheck 相關檔案複製至電腦中。預設路徑為 C:\ICPDAS\EzCheck Vision Library。



圖五、 安裝完成

安裝完成後，使用者將可以在 C:\ICPDAS\EzCheck Vision Library 資料夾下找到 EzCheck Vision Library 所有的檔案。

安裝後的資料夾包含：

- EzCheck for BCB。

  以 EzCheck Vision Library 開發 Borland C++ Builder 6.0 專案所需檔案，包含：

  - Libs：開發專案需要的 lib 檔案。
  - Dlls：執行專案需要的 dll 檔案。
  - Includes：開發專案需要的 include 檔案。
  - Sample：BCB 的範例程式。

- EzCheck for VC

  以 EzCheck Vision Library 開發 Visual C++ 6.0 專案所需檔案，包含：

  - Libs：開發專案需要的 lib 檔案。
  - Dlls：執行專案需要的 dll 檔案。
  - Includes：開發專案需要的 include 檔案。
  - Sample：VC 的範例程式。

- EzCheck Document

  包含本使用手冊以及 Quickstart。

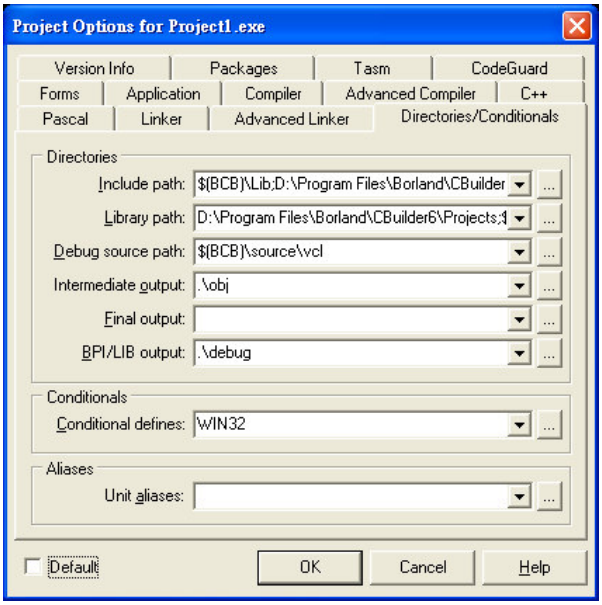
- EzCheck Utility

包含 EzCheck Utility 以及所需檔案。

2.2. 使用 EzCheck Vision Library 於 Borland C++ Builder 6.0

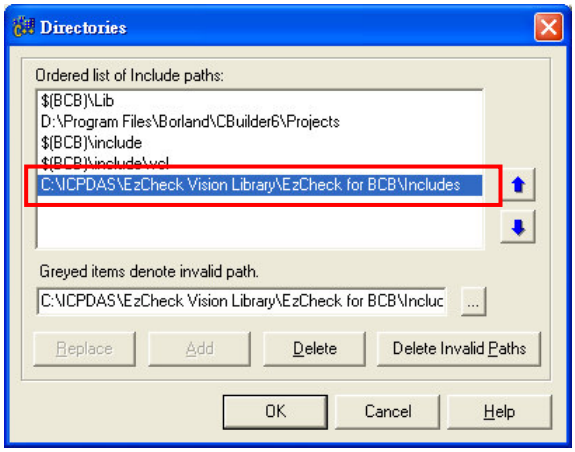
以下流程將帶領使用者建立於 Borland C++ Builder 6.0 開發環境下的 EzCheck Vision Library 應用程式。

(1) 開新專案。
(2) 設定引用檔案路徑。



圖六、 BCB 中設定路徑的介面

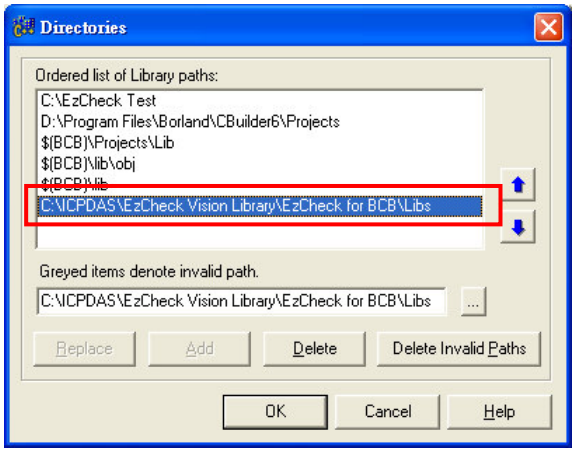
- 設定 header files 的引用路徑：
  Project->Options->Directories/Conditionals->Include path。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Includes 加入專案路徑中。



圖七、 加入 BCB 版本的 Includes 路徑。

➢ 設定 lib files 的引用路徑：

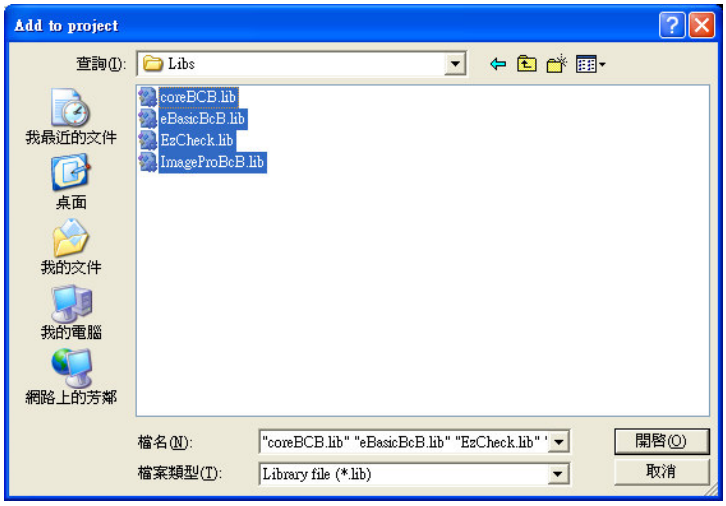
Project->Options->Directories/Conditionals->Library path。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Libs 加入專案路徑中。



圖八、 加入 BCB 版本的 Libs 路徑。

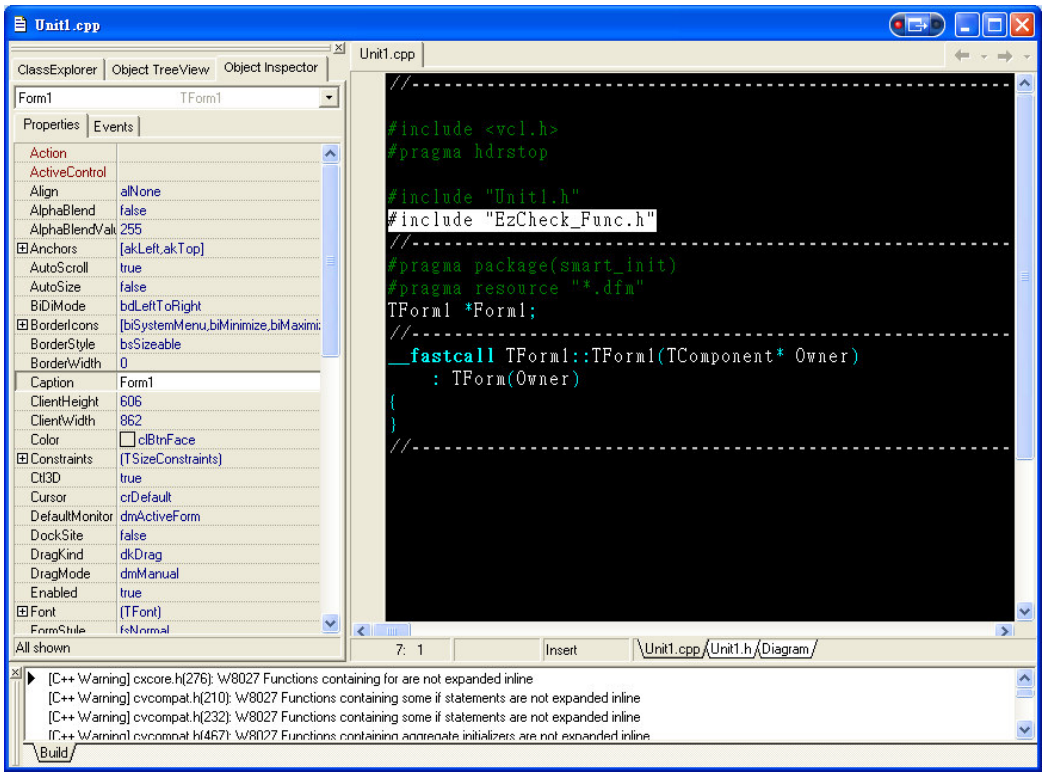
(3) 將 lib 檔案加入專案中。

Project->Add to Project。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Libs 下的四個 lib 檔案加入專案中。



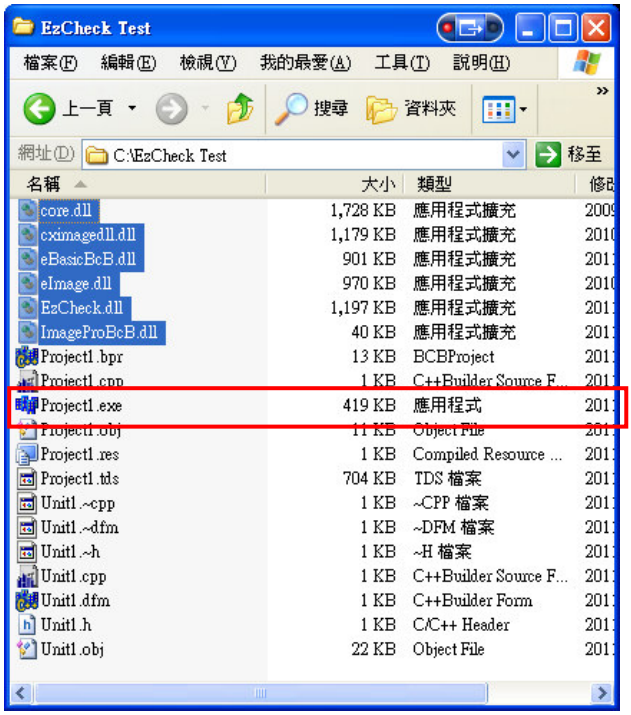
圖九、 加入 BCB 版本的 lib files。

(4) 於專案中加入#include "EzCheck_Func.h"



圖十、 引入 EzCheck_Func.h 即可連結到 EzCheck Vision Library 所有功能。

(5) 將 dll 檔案由 Dlls 資料夾複製至專案執行檔的資料夾下。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Dlls 下的六個 dll 檔案複製到專案執行檔的資料夾。



圖十一、 將六個 dll 檔案與專案執行檔放在同一個資料夾下。

(6) 使用 EzCheck Vision Library 開發機器視覺系統。

2.3. 使用 EzCheck Vision Library 於 Visual C++ 6.0

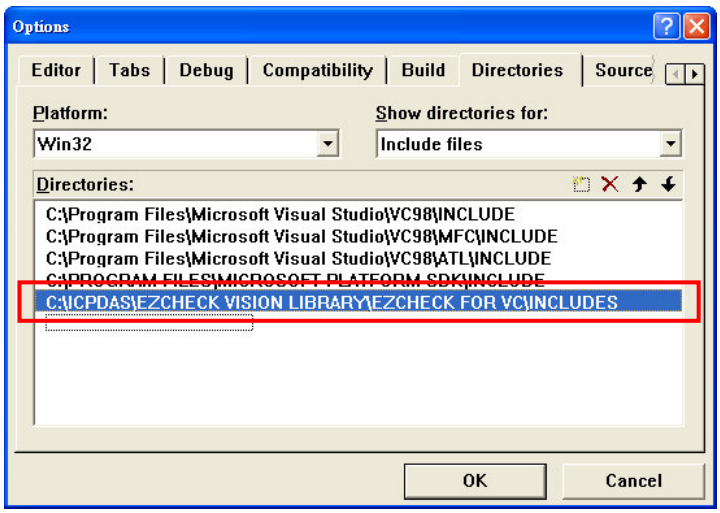
以下流程將帶領使用者建立於 Visual C++ 6.0 開發環境下的 EzCheck Vision Library 應用程式。

(1) 開新專案。

(2) 設定引用檔案路徑。

- 設定 header files 的引用路徑：

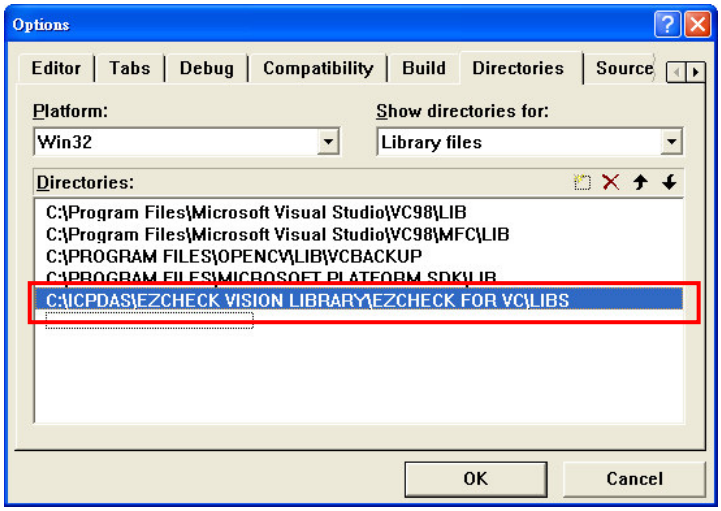
  Tools->Options->Directories->Include filies。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Includes 加入專案路徑中。



圖十二、　　加入 VC 版本的 Includes 路徑。

➢ 設定 lib files 的引用路徑：

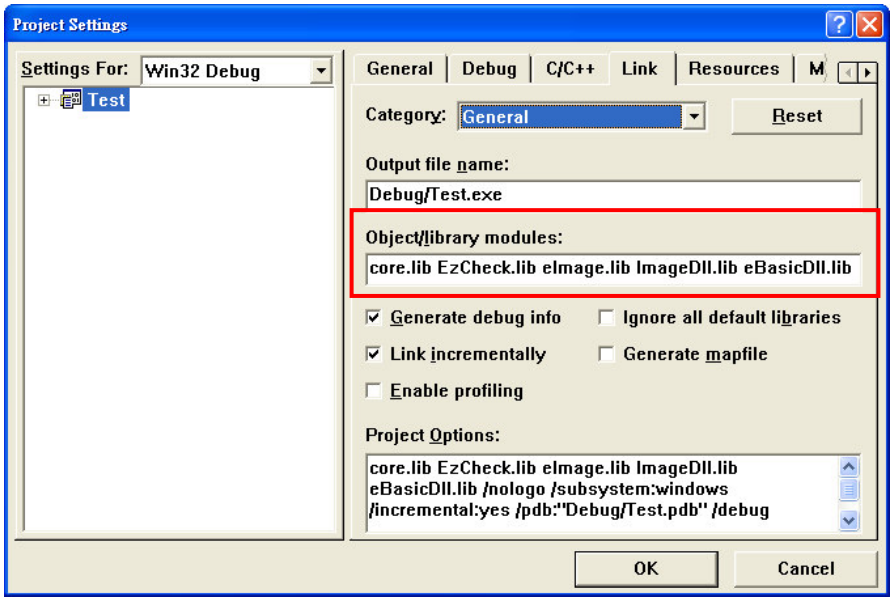
Tools->Options->Directories ->Library files。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Libs 加入專案路徑中。



圖十三、　　加入 VC 版本的 Libs 路徑。

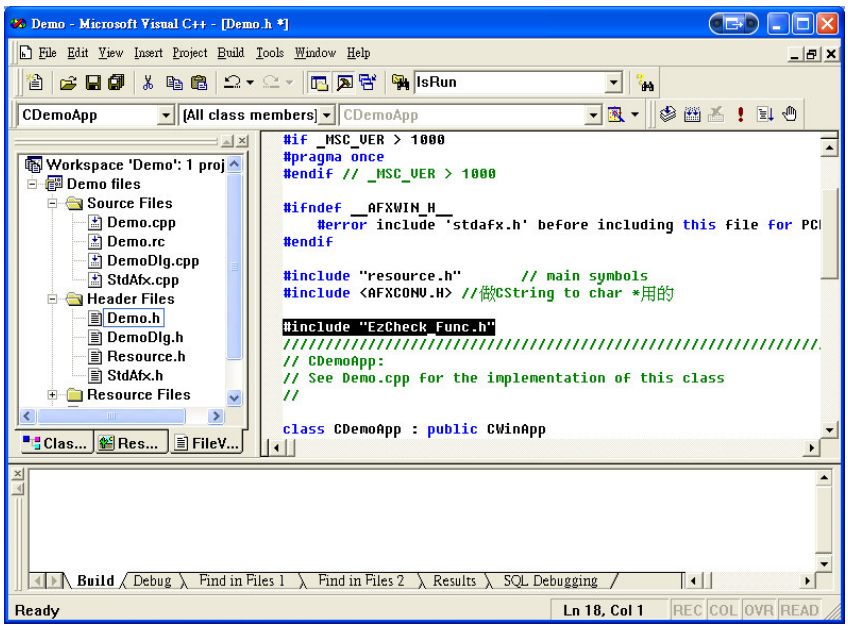
(3) 將 lib 檔案加入專案中。

Project->Setting->Link->Object/library modules。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Libs 下的五個 lib 檔案加入專案中。



圖十四、　　加入 VC 版本的 lib files。

(4) 於專案中加入#include "EzCheck_Func.h"



圖十五、　　引入 EzCheck_Func.h 連結 EzCheck Vision Library 所有功能。

(5) 將 dll 檔案由 Dlls 資料夾複製至專案執行檔的資料夾下。將 C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Dlls 下的六個 dll 檔案複製到專案執行檔的資料夾。

圖十六、 將六個 dll 檔案與專案執行檔放在同一個資料夾下。

(6) 使用 EzCheck Vision Library 開發機器視覺系統。

3.EzCheck 影像－eCImage

3.1. eCImage 主要特色

支援的圖檔格式:

- Windows Bitmaps – BMP, DIB
- JPEGE files – JPEG, JPG, JPE
- TIFF files – TIFF, TIF
- Portable Network Graphics – PNG

eCImage 支援多種影像格式的讀取與寫入，提供各種取得影像資訊，如色彩通道數、尺寸資訊以及各個像素點資訊的存取機制。使用者可對影像進行 ROI 的設定 (ROI 的設定將可以作用於部分的影像處理或分析步驟)、取像。也可透過 eCImage 提供的功能，即時截取來自攝影機的影像資訊，若再結合影像校正的功能，便可以將來自不同視角的影像在校正後供後續處理與分析使用。

3.2. 主要功能介紹

3.2.1. eCImage

建構式

函式原形：

```
eCImage     ( INT32 n32Width, INT32 n32Height, UINT8 bpp, UINT origin );
```

傳入值：

n32Width	**in**	影像寬度
n32Height	**in**	影像高度
bpp	**in**	位元深度。8：灰階影像、24：彩色影像。
origin	**in**	指定影像的原點座標，0： 左上角、1：左下角為原點

3.2.2. Create

宣告影像物件

函式原形：

```
Create (INT32 n32Width.
        INT32 n32Height,
        UINT8 bpp.
        UINT origin);
```

傳入值：

n32Width	**in**	影像寬度
n32Height	**in**	影像高度
bpp	**in**	位元深度。8：灰階影像、24：彩色影像。
origin	**in**	指定影像的原點座標，0： 左上角、1：左下角為原點

Example Code:

```
eCImage *SrcImage= new eCImage; //宣告一個新的 eCImage 物件
SrcImage->Create(320, 240, 8, 0);//宣告 320*240，以左上角為原點的灰階影像
```

Tips：此為 eCImage 的建議宣告方式。若非以 Load(讀檔)或自其他 eCImage 複製影像資訊取得的 eCImage，建議以此方式創建。

3.2.3. Release

釋放物件

函式原形：

```
BOOL   Release();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Tips：建議所有宣告使用的 eCImage，在使用完畢之後將其釋放，避免造成記憶體的浪費。

3.2.4. GetCopy

複製 eCImage 物件。取得 sourceObj 包含 ROI 等所有資訊的複製。

函式原形：

```
BOOL   GetCopy(const eCImage* sourceObj);
```

傳入值：

sourceObj **in** 複製的來源 eCImage。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example Code:

```
eCImage *SrcImage= new eCImage;
eCImage *DstImage = new eCImage;
SrcImage->Load( “D:\\bitmap1.bmp” );
DstImage ->GetCopy(SrcImage);//複製，DstImage 與 SrcImage 將是各自獨立但完全相同的 eCImage 物件
```

3.2.5. GetGrayCopy

複製 eCImage 物件的灰階影像。除複製出的影像為灰階外，其餘資訊皆與 sourceObj 相同。

函式原形：

```
BOOL   GetGrayCopy(const eCImage* sourceObj);
```

傳入值：

sourceObj **in** 複製的來源

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example Code:

```
eCImage *SrcImage= new eCImage;
eCImage *DstImage = new eCImage;
SrcImage->Load( “D:\\bitmap1.bmp” );
DstImage ->GetGrayCopy(SrcImage);// 取得 SrcImage 的灰階影像
```

Tips：各種複製另一 eCImage 的方法，都會將目標 eCImage 的 ROI、尺寸、影像通道數等資訊更新與來源影像相同。

3.2.6. Load

開啟檔案

函式原形：

```
BOOL   Load(const char* pszPathName);
BOOL   Load(const UNICHAR* pszPathName);
```

傳入值：

pszPathName in 檔案路徑與檔名。char 或 Unicode 字元指標。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.7. Save

儲存為檔案

函式原形：

```
BOOL   Save(const char* pszPathName);
BOOL   Save(const UNICHAR* pszPathName);
```

傳入值：

pszPathName in 檔案路徑與檔名。char 或 Unicode 字元指標。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.8. GetWidth

取得影像寬度資訊

函式原形：

```
INT32   GetWidth();
```

回傳值：

影像寬度。

3.2.9. GetHeight

取得影像高度資訊

函式原形：

```
INT32   GetHeight();
```

回傳值：

影像高度。

3.2.10. GetSize

取得影像尺寸 (image width*image height)

函式原形：

```
INT32   GetSize();
```

回傳值：

影像尺寸。影像尺寸為影像寬與影像高的乘積。

3.2.11. GetPlanes

取得影像的通道數 (1~4)
四個通道分別代表 R,G, B,Alpha
有一些檔案格式(例如: TIFF, PNG)會多增加一個 8 位元 Alpha 通道，來將 24 位元的影像擴充到 32 位元

函式原形：

```
INT32   GetPlanes();
```

回傳值：

影像通道數。

3.2.12. GetBitsPerRow

影像的每一列有多少位元 (image width*channels*bpp)

函式原形：

```
INT32   GetPlanes();
```

回傳值：

影像每一列的位元數。

3.2.13. GetBitsPerPixel

影像的每一像素有多少位元 (channels*bpp)

函式原形：

```
INT32   GetPlanes();
```

回傳值：

影像每一像素的位元數。

3.2.14. GetImagePtr

指向影像的*影像對齊資料矩陣*(aligned image data)的指標.
*影像對齊資料矩陣*中存放了一張圖的色彩資訊

函式原形：

```
INT32   GetImagePtr();
```

回傳值：

指向影像的**影像對齊資料矩陣**(aligned image data)的指標。

3.2.15. IsBlank

回傳 eCImage 物件中是否有影像資訊。

函式原形：

```
bool   IsBlank();
```

回傳值：

true 表示影像內容為空，false 表示影像內容已有紀錄。

3.2.16. SetSize

以絕對尺寸或者尺寸比例設定影像大小。例如，原始影像為 640*480，x_scale 與 y_scale 均為 0.5，則會取得原始影像 320*240 的縮圖。影像尺寸的改變會產生一定程度的失真。

函式原形：

```
BOOL   SetSize(INT32 x_size.
               INT32 y_size);
BOOL   SetSize(FLOAT32 x_scale.
               FLOAT32 y_scale);
```

傳入值：

x_size	**in**	設定的影像寬度
y_size	**in**	設定的影像高度
x_scale	**In**	設定的影像寬度縮放比例
y_scale	**in**	設定的影像高度縮放比例

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.17. SetROI

設定影像中的 ROI(Region Of Interest)區域，提供兩種設定 ROI 的方式

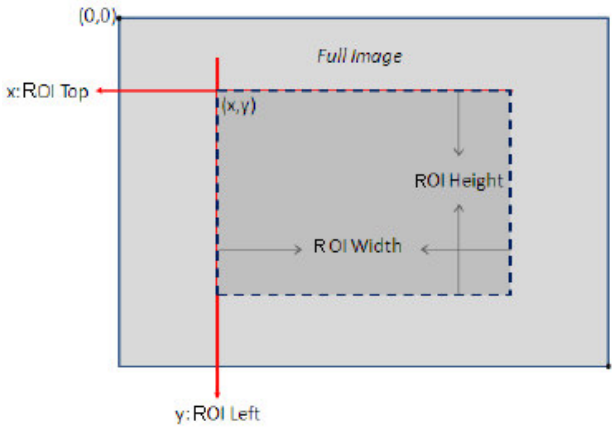
a. 以 eCREC 設定 ROI。

b. 以原點與長寬資訊設定 ROI

eCREC

紀錄 ROI 資訊的資料結構

```
typedef struct tag_REC
{
    INT32 org_x ;        // 原點之 x 座標
    INT32 org_y ;        // 原點之 y 座標
    INT32 width ;        // rectangle 的寬
    INT32 height ;       // rectangle 的高
} eCREC;
```



函式原形：

```
BOOL   SetROI(eCREC rec);
BOOL   SetROI(int org_x.
              int org_y.
              int width.
              int height);
```

傳入值：

rec	**in**	eCREC 資料結構。包含 ROI 的原點與尺寸。
org_x, org_y	**in**	ROI 的原點座標。
Width, height	**in**	ROI 的尺寸。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Tips：EzCheck Vision Library 中的 ROI 將不會影響影像本身，但作用在影像上的功能將只會作用在 ROI 上。大部分對影像進行分析的功能，也僅會對 ROI 的部分進行分析。

3.2.18. ResetROI

釋放影像中的 ROI 區域，該影像不再包含 ROI 資訊

函式原形：

```
BOOL   ResetROI();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.19. GetROI

取得影像中的 ROI 區域

函式原形：

```
eCREC   GetROI();
```

傳入值：

sourceObj **in** 複製的來源

回傳值：

eCREC 資料結構。請參考 eCREC。

3.2.20. GetROIImage

取得影像中 ROI 區域的影像

函式原形：

```
BOOL   GetROIImage(eCImage* &DstImage);
```

傳入值：

DstImage	**out**	目的影像。將會取得來源影像 ROI 中的影像。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example Code:

```
eCImage *OriImage= new eCImage;
eCImage *ROIImage= new eCImage;
OriImage ->Load( “D:\\bitmap1.bmp” );
OriImage->SetROI(100, 100, 100, 100);
OriImage->GetROIImage(ROIImage);   // 取得 ROI 部分的影像
```

3.2.21. GetROIImage

快速取得來源影像中 ROI 的影像，不需在來源影像上設定 ROI 即可直接取得 ROI 影像。若取得的區超出來源影像，會將超出範圍的區域以黑色填滿。

函式原形：

```
BOOL   GetROIImage(eCImage* &DstImage,
                   int ori_x,
                   int ori_y,
                   int roi_width,
                   int roi_height,
                   int Extend);
```

傳入值：

DstImage	**out**	目的影像。將會取得來源影像 ROI 中的影像。
ori_x, ori_y	**in**	ROI 原點
roi_width, roi_height	**in**	ROI 尺寸
Extend	**in**	影像四周延伸尺寸。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

The meaning of Extend:

Example Code:

```
eCImage *OriImage= new eCImage;
eCImage *ROIImage= new eCImage;
OriImage ->Load( “D:\\bitmap1.bmp” );
OriImage->SetROI(100, 100, 100, 100);
OriImage->GetROIImage(ROIImage);   // 取得 ROI 部分的影像
```

Tips：
EzCheck Vision Library 支援兩種 ROI 使用方式：

- 使用 eCImage::GetROIImage 將 ROI 區域直接取出為獨立影像，對此影像進行運算。
- 設定 ROI 後直接對影像進行運算，若運算功能支援 ROI 運算，將只會對 ROI 規範內的區域進行運算或分析。

運算功能是否支援 ROI 運算，會於手冊中註明。

3.2.22. Draw

繪製 eCImage 物件。

函式原形：

```
BOOL   Draw(HDC hDC.
            FLOAT32 f32ZoomX.
            FLOAT32 f32ZoomY);
```

傳入值：

hDC	**in**	輸出標的的索引值
f32ZoomX	**in**	X 方向的縮放比例。
f32ZoomY	**in**	Y 方向的縮放比例。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.23. GetCameraImage

取得攝影機截取出的影像 buffer，將其轉換為可進行處理的影像格式

※目前確定可支援的攝影機有 ICPDAS Sparrow IMS-130 以及 MAVIS IM-100、IM-30。

函式原形：

```
BOOL   GetCameraImage(const char* VideoBuffer,
                      INT32 Width.
                      INT32 Height,
                      UINT8 bpp);
```

傳入值：

VideoBuffer	**in**	檔案路徑與檔名。char 或 Unicode 字元指標。
Width	**in**	影像寬度。需與攝影機取像的資訊相同。
Height	**in**	影像高度。需與攝影機取像的資訊相同。
bpp	**in**	自攝影機取得的影像 buffer

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.24. GetPixel*****----(Gray, Red, Green, Blue)

取得像素點的色彩資訊，輸入像素的座標，取得該像素的灰階、紅色、綠色或是藍色資訊

函式原形：

```
INT32   GetPixelGrav(int x. int v);
INT32   GetPixelRed(int x. int v):
INT32   GetPixelGreen(int x. int v);
INT32   GetPixelBlue(int x, int y);
```

傳入值：

x, y **in** 要取資訊的像素點座標

回傳值：

像素的色彩資訊。取得的資訊為 0~255 的整數。

3.2.25. SetPixel*****----(Gray, Red, Green, Blue)

寫入像素點的色彩資訊，輸入像素的座標與欲寫入的數值

函式原形：

```
BOOL   SetPixelGrav(int x. int v. int value);
BOOL   SetPixelRed(int x. int v. int value):
BOOL   SetPixelGreen(int x. int v. int value);
BOOL   SetPixelBlue(int x, int y, int value);
```

傳入值：

x, y **in** 要取資訊的像素點座標

value **in** 要輸入的像素點資訊。0～255。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

3.2.26. GetErrorCode

取得 eCImage 運作中回報的 ErrorCode。

函式原形：

```
int   GetErrorCode();
```

回傳值：

eCImage 運作中回報的 error code。請參考 ErrorCode 列表。

4.影像處理群組－eCImg Group

4.1. eCImg Group 簡介

基本影像處理泛指: 影像處理上的形態學操作、濾波器、色彩二值化分割、直方圖統計以及影像的幾何變換等等，一般來說在處理一張圖片時，常無可避免的會用到這些處理演算法，因此本文中將這些類別歸類成基本的影像處理方式。本函式庫對這些基本影像處理的函式皆以 eCIma_前贅字開頭命名之。

eCImg Group 包含以下幾個小群組

- 二值化(Threshold)
- 色彩通道轉換(Color Channel)
- 遮罩運算(Filter)
- 形態學運算(Morphology)
- 影像旋轉(Rotation)
- 直方圖運算(Histogram)

Tips：影像處理方法無關好壞，不同的影像處理方法均有其優點與限制，多方嘗試尋找最合適的方法是相當重要的。

4.2. 二值化－Threshold

EzCheck Vision Library 提供多種不同的二值化功能，包含使用者自行輸入臨界值，或者系統依照影像特性計算臨界值的二值化。依照最常使用的狀況，EzCheck Vision Library 提供的二值化，將大於臨界值的像素設為 255(白色)，並將小於臨界值的像素設為 0(黑色)。

Tips：所有二值化功能都支援影像的 ROI 處理。

4.2.1. eCImg_AbsoluteThreshold

由使用者指定一個固定臨界值，做灰階(色彩)二值化

函式原形：

```
BOOL   eClmg_AbsoluteThreshold(eClmage *SrcImage,
                               eClmage *&DstImage,
                               UINT32 threshold);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
threshold	**in**	分割像素強度的臨界值

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.2.2. eClmg_RelativeThreshold

由使用者指定一個比例參數，做灰階(色彩)二值化。

函式原形：

```
BOOL   eClmg_RelativeThreshold(eClmage *SrcImage,
                               eClmage *&DstImage,
                               FLOAT32 f32RelativeThreshold);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
threshold	**in**	分割像素強度的**比例值**。0～1。此參數代表整張畫面內多少比例的像素會被設定為黑(0)，函式中將會自動計算其對應的臨界值大小。例如，比例值設定為 0.8，則此張影像將變為 80% black、20% white

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.2.3. eCImg_DoubleThreshold

由使用者輸入兩個臨界值，做灰階三值化(0, 128, 255)。臨界值 1、2 的設定都必須在 0~255 之間，但使用者無須煩惱兩者的大小關係。小於較小的臨界值的像素將被設定為 0，介於兩者之間的像素設為 128，大於較大的臨界值者則被設為 255。

函式原形：

```
BOOL   eCImg_DoubleThreshold(eCImage *SrcImage,
                             eCImage *&DstImage,
                             UINT32 threshold1,
                             UINT32 threshold2);
```

傳入值：

SrcImage	**in**	來源影像。
DstImage	**out**	目的影像。
threshold1	**in**	臨界值 1。0～255。
threshold2	**in**	臨界值 2。0～255。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.2.4. eCImg_MomentThreshold

使動量守恆(moment preserving)的**自動二值化**。
本函式自動計算出一個像素臨界值，此臨界值能讓原始影像(SrcImage)與二值化影像(DstImage)的動量(Moment)維持守恆

函式原形：

```
BOOL   eCImg_MomentThreshold(eCImage *SrcImage,
                             eCImage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.2.5. eCImg_MinResidueThreshold

使平方差最小的自動二值化。

本函式自動計算出一個像素臨界值，此臨界值能讓原始影像(SrcImage)與二值化影像(DstImage)的每一個像素之相減平方值為最小

函式原形：

```
BOOL   eCImg_MinResidueThreshold(eCImage *SrcImage,
                                 eCImage *&DstImage );
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.2.6. eCImg_MaxEntropyThreshold

使影像之資訊熵(Entropy)最大的二值化。

本函式自動計算出一個像素臨界值，此臨界值能讓二值化影像(DstImage)的 Entropy 為最大

函式原形：

```
BOOL   eCImg_MaxEntropyThreshold(eCImage *SrcImage,
                                 eCImage *DstImage );
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.2.7. eCImg_IsodataThreshold

反覆自我組織資料分析法(ISODATA) 的自動二值化。
本函式自動計算出一個像素臨界值，此臨界值介在亮群之平均灰階值(the average light gray values (i.e. gray levels above the threshold).)跟暗群之平均灰階值(the average dark gray value (i.e. gray levels below the threshold))的中間

函式原形：

```
BOOL   eCImg_IsodataThreshold(eCImage *SrcImage,
                              eCImage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3. 迴旋濾波器－Convolution filter

EzCheck Vision Library 提供多種迴旋濾波器運算，包含影像的模糊(smooth)、銳化(sharp)以及邊緣檢測(edge detection)。使用者不需要額外的影像處理背景知識，僅需依照各個函式的功能提示進行嘗試，找尋最適合的函式處理影像即可。

Tips：所有濾波器功能都支援影像的 ROI 處理。

4.3.1. eCImg_ConvUniform

使影像變得平滑(uniform filter)，並且濾除影像上的雜訊。由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eCImg_ConvUniform( eCImage *SrcImage,
                          eCImage *&DstImage,
                          UINT32 half_width,
                          UINT32 half_height);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬(width=half_width*2+1)
half_height	**in**	遮罩的一半高(height=half_height*2+1)

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.2. eCImg_ConvGaussian

高斯濾波器(Gaussian filtering)。使影像平滑，並濾除影像上的雜訊。由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eCImg_ConvGaussian(eCImage *SrcImage,
                          eCImage *&DstImage,
                          UINT32 half_width,
                          UINT32 half_height );
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
half_height	**in**	遮罩的一半高

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.3. eCImg_ConvHighPass1

高通濾波器(high-pass filtering)。
增強影像的銳利度，改善模糊不清的影像，同時可能使雜訊變的明顯

函式內使用 3*3 濾波器:

0	-1	0
-1	5	-1
0	-1	0

函式原形：

```
BOOL   eCImg_ConvHighPass1(eCImage *SrcImage,
                           eCImage *&DstImage );
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.4. eCImg_ConvHighPass2

高通濾波器(high-pass filtering)。
增強影像的銳利度，改善模糊不清的影像，同時可能使雜訊變的明顯

函式內使用 3*3 濾波器:

-1	-1	-1
-1	9	-1
-1	-1	-1

函式原形：

```
BOOL   eClmg_ConvHighPass2( eCImage *SrcImage,
                            eCImage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像

DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.5. eClmg_ConvHighPass3

高通濾波器(high-pass filtering)。

增強影像的銳利度，改善模糊不清的影像，同時可能使雜訊變的明顯

函式內使用 3*3 濾波器:

-1/8	-1/8	-1/8
-1/8	16/8	-1/8
-1/8	-1/8	-1/8

函式原形：

```
BOOL   eClmg_ConvHighPass3(eCImage *SrcImage,
                           eCImage *&DstImage);
```

傳入值：

SrcImage **in** 來源影像

DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.6. eCImg_ConvLowPass1

低通濾波器(low-pass filtering)用來消除影像上高頻變化的部分，可使影像的變化較均勻(模糊)。

函式內使用 3*3 濾波器:

1	1	1
1	1	1
1	1	1

函式原形：

```
BOOL   eCImg_ConvLowPass1(eCImage *SrcImage,
                          eCImage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.7. eCImg_ConvLowPass2

低通濾波器(low-pass filtering)用來消除影像上高頻變化的部分，可使影像的變化較均勻(模糊)。

函式內使用 3*3 濾波器:

1	1	1
1	0	1
1	1	1

函式原形：

```
BOOL   eCImg_ConvLowPass2(eCImage *SrcImage,
                          eCImage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.8. eCImg_ConvLowPass3

低通濾波器(low-pass filtering)用來消除影像上高頻變化的部分，可使影像的變化較均勻(模糊)。

函式內使用 3*3 濾波器:

1	2	1
2	4	2
1	2	1

函式原形：

```
BOOL   eCImg_ConvLowPass3(eCImage *SrcImage,
                          eCImage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.9. eCImg_ConvGradientX

x 方向的梯度濾波器(Gradient filtering)。
梯度濾波器使用一階導數，可視為一個邊緣(edge)檢測器
函式內使用 3*3 濾波器:

0	0	0
-1	0	1
0	0	0

函式原形：

```
BOOL   eCImg_ConvGradientX(eCImage *SrcImage,
                           eCImage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.10. eCImg_ConvGradientY

y 方向的梯度濾波器(Gradient filtering)。
梯度濾波器使用一階導數，可視為一個邊緣(edge)檢測器

函式內使用 3*3 濾波器:

0	-1	0
0	0	0
0	1	0

函式原形：

```
BOOL   eCImg_ConvGradientY(eCImage *SrcImage,
                           eCImage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.11. eClmg_ConvGradient

等向性的梯度濾波器(Gradient filtering)。
梯度濾波器使用一階導數，可視為一個邊緣(edge)檢測器，等向性梯度濾波器是將 x.y 方向梯度濾波器的結果(絕對值)做相加

函式原形：

```
BOOL   eClmg_ConvGradient(eClmage *SrcImage,
                          eClmage *&DstImage);
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.12. eClmg_ConvPrewittX

x 方向的 Prewitt 濾波器(Prewitt filtering)。
Prewitt 濾波器使用一階導數，可視為一個邊緣(edge)檢測器

函式內使用 3*3 濾波器:

-1	0	1
-1	0	1
-1	0	1

函式原形：

```
BOOL   eClmg_ConvPrewittX(eClmage *SrcImage,
                          eClmage *&DstImage );
```

傳入值：

SrcImage **in** 來源影像

DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.13. eClmg_ConvPrewittY

y 方向的 Prewitt 濾波器(Prewitt filtering)。

Prewitt 濾波器使用一階導數，可視為一個邊緣(edge)檢測器

函式內使用 3*3 濾波器:

-1	-1	-1
0	0	0
1	1	1

函式原形：

```
BOOL   eClmg_ConvPrewittY(eClmage *SrcImage,
                          eClmage *&DstImage);
```

傳入值：

SrcImage **in** 來源影像

DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.14. eClmg_ConvPrewitt

等向性的 Prewitt 濾波器(Prewitt filtering)。
Prewitt 濾波器使用一階導數，可視為一個邊緣(edge)檢測器，等向性 Prewitt 濾波器是將 x.y 方向 Prewitt 濾波器的結果(絕對值)做相加

函式原形：

```
BOOL   eClmg_ConvPrewitt(eClmage *SrcImage,
                         eClmage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.15. eClmg_ConvSobelX

x 方向的 Sobel 濾波器(Sobel filtering)。
Sobel 濾波器使用一階導數，可視為一個邊緣(edge)檢測器

函式內使用 3*3 濾波器:

-1	0	1
-2	0	2
-1	0	1

函式原形：

```
BOOL   eClmg_ConvSobelX(eClmage *SrcImage,
                        eClmage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.16. eClmg_ConvSobelY

y 方向的 Sobel 濾波器(Sobel filtering)。
Sobel 濾波器使用一階導數，可視為一個邊緣(edge)檢測器

函式內使用 3*3 濾波器:

-1	-2	-1
0	0	0
1	2	1

函式原形：

```
BOOL   eClmg_ConvSobelY(eClmage *SrcImage,
                        eClmage *&DstImage);
```

傳入值：

SrcImage **in** 來源影像
DstImage **out** 目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.3.17. eClmg_ConvSobel

等向性的 Sobel 濾波器(Sobel filtering)。
Sobel 濾波器使用一階導數，可視為一個邊緣(edge)檢測器，等向性 Sobel 濾波器是將 x.y 方向 Sobel 濾波器的結果(絕對值)做相加

函式原形：

```
BOOL   eClmg_ConvSobel (eClmage *SrcImage,
                        eClmage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4. 形態學運算 Morphology

EzCheck Vision Library 提供多種形態學運算，包含影像的侵蝕、膨脹以及開、閉等運算，也支援圓形以及方型兩種不同的運算子。使用者不需要額外的影像處理背景知識，僅需依照各個函式的功能提示進行嘗試，找尋最適合的函式處理影像即可。

Tips：所有形態學運算都支援影像的 ROI 處理。

4.4.1. eCImg_Median

平均值濾波器以 Kernel 中所有像素灰階值(包括中心像素本身)之平均取代 Kernel 中所有像素的原本灰階值，也具有平滑影像的效果。

函式原形：

```
BOOL   eCImg_Median(eCImage *SrcImage,
                    eCImage *DstImage,
                    UINT8 size);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
size	**in**	Kernel 的大小(size*size)。size 必須為奇數。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.2. eCImg_OpenBox

以 rectangular kernel 在影像上進行斷開(Open)。

DstImage= Open(SrcImage,kernel) = dilate(erode(SrcImage,kernel)kernel)

由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eCImg_OpenBox(eCImage *SrcImage,
                     eCImage *DstImage,
                     UINT32 half_width,
                     UINT32 half_height,
                     UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
half_height	**in**	遮罩的一半高
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.3. eCImg_OpenDisk

以 elliptic kernel 在影像上進行斷開(Open)。

DstImage= Open(SrcImage,kernel) = dilate(erode(SrcImage,kernel)kernel)

由於遮罩尺寸必須為奇數，因此輸入之寬為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。

函式原形：

```
BOOL   eCImg_OpenDisk(eCImage *SrcImage,
                      eCImage *DstImage,
                      UINT32 half_width,
                      UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.4. eCImg_CloseBox

以 rectangular kernel 在影像上進行閉合(Close)。

DstImage= Close(SrcImage,kernel) = erode(dilate(SrcImage,kernel)kernel)

由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eCImg_CloseBox(eCImage *SrcImage,
                      eCImage *DstImage,
                      UINT32 half_width,
                      UINT32 half_height,
                      UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
half_height	**in**	遮罩的一半高
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.5. eCImg_CloseDisk

以 elliptic kernel 在影像上進行閉合(Close)。

DstImage= Close (SrcImage,kernel) = erode(dilate (SrcImage,kernel)kernel)

由於遮罩尺寸必須為奇數，因此輸入之寬為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。

函式原形：

```
BOOL   eCImg_CloseDisk(eCImage *SrcImage,
                       eCImage *DstImage,
                       UINT32 half_width,
                       UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.6. eCImg_ErodeBox

以 rectangular kernel 在影像上進行侵蝕(Erode)
由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eCImg_ErodeBox(eCImage *SrcImage,
                      eCImage *DstImage,
                      UINT32 half_width,
                      UINT32 half_height,
                      UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
half_height	**in**	遮罩的一半高
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.7. eCImg_ErodeDisk

以 elliptic kernel 在影像上進行侵蝕(Erode)
由於遮罩尺寸必須為奇數，因此輸入之寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。

函式原形：

```
BOOL   eCImg_ErodeDisk(eCImage *SrcImage,
                       eCImage *DstImage,
                       UINT32 half_width,
                       UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.8. eCImg_DilateBox

以 rectangular kernel 在影像上進行膨脹(Dilate)

由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eCImg_DilateBox(eCImage *SrcImage,
                       eCImage *DstImage,
                       UINT32 half_width,
                       UINT32 half_height,
                       UINT8 times    );
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
half_height	**in**	遮罩的一半高
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.9. eClmg_DilateDisk

以 elliptic kernel 在影像上進行膨脹(Dilate)
由於遮罩尺寸必須為奇數，因此輸入之寬為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。

函式原形：

```
BOOL   eClmg_DilateDisk(eClmage *SrcImage,
                        eClmage *DstImage,
                        UINT32 half_width,
                        UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.10. eClmg_MorphGradientBox

以 rectangular kernel 在影像上做形態學梯度計算。
DstImage= MorphGradient (SrcImage,kernel)
= dilate(SrcImage,kernel) - erode(SrcImage,kernel)
由於遮罩尺寸必須為奇數，因此輸入之長寬皆為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。例如：輸入(1, 1)將得到 3x3 的遮罩，輸入(2, 2)則得到 5x5，輸入(1, 2)將得到 3x5。

函式原形：

```
BOOL   eClmg_MorphGradientBox(eClmage *SrcImage,
                              eClmage *DstImage,
                              UINT32 half_width,
                              UINT32 half_height,
```

```
                                        UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
half_height	**in**	遮罩的一半高
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.4.11. eCImg_MorphGradientDisk

以 elliptic kernel 在影像上做形態學梯度計算。

DstImage= MorphGradient (SrcImage,kernel)

= dilate(SrcImage,kernel) - erode(SrcImage,kernel)

由於遮罩尺寸必須為奇數，因此輸入之寬為運算遮罩的一半左右，以確保取得奇數尺寸的遮罩。

函式原形：

```
BOOL   eCImg_MorphGradientDisk(eCImage *SrcImage,
                               eCImage *DstImage,
                               UINT32 half_width,
                               UINT8 times);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
half_width	**in**	遮罩的一半寬 (width=half_width*2+1)
times	**in**	執行侵蝕與膨脹的次數

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5. Color Transform 色彩通道轉換

EzCheck Vision Library 提供色彩通道的轉換。使用者可將彩色影像轉為灰階，或者分割為單一色彩的影像。也可取得貼近人類對色彩認知的 HIS(色調、飽和度、亮度)影像通道。

4.5.1. eClmg_RGB2GRAY

取得彩色 RGB 影像的灰階影像。若影像本身為灰階影像，則只會取得原始影像的複製影像。
轉換公式為 Gray = 0.212671*R + 0.715160*G + 0.072169*B

函式原形：

```
BOOL   eClmg_RGB2GRAY(eClmage *SrcImage,
                      eClmage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5.2. eClmg_RGB2Red

取得彩色 RGB 影像的紅色通道影像。若影像本身為灰階影像，則只會取得原始影像的複製影像。

函式原形：

```
BOOL   eClmg_RGB2Red(eClmage *SrcImage,
                     eClmage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5.3. eCImg_RGB2Green

取得彩色 RGB 影像的綠色通道影像。若影像本身為灰階影像，則只會取得原始影像的複製影像。

函式原形：

```
BOOL   eCImg_RGB2Green(eCImage *SrcImage,
                       eCImage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5.4. eCImg_RGB2Blue

取得彩色 RGB 影像的藍色通道影像。若影像本身為灰階影像，則只會取得原始影像的複製影像。

函式原形：

```
BOOL   eCImg_RGB2Blue(eCImage *SrcImage,
                      eCImage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5.5. eCImg_RGB2HSI_H

取得彩色 RGB 影像的 HIS 影像格式之色調(Hue)影像，將色調以 0~255 表示。若影像本身為灰階影像，則只會取得原始影像的複製影像。

函式原形：

```
BOOL   eCImg_RGB2HSI_H(eCImage *SrcImage,
                       eCImage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5.6. eCImg_RGB2HSI_S

取得彩色 RGB 影像的 HIS 影像格式之飽和度(Saturation)影像，將飽和度以 0~255 表示。若影像本身為灰階影像，則只會取得原始影像的複製影像。

函式原形：

```
BOOL   eCImg_RGB2HSI_S(eCImage *SrcImage,
                       eCImage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.5.7. eClmg_RGB2HSI_I

取得彩色 RGB 影像的 HIS 影像格式之亮度(Intensity)影像，將亮度以 0~255 表示。若影像本身為灰階影像，則只會取得原始影像的複製影像。

函式原形：

```
BOOL   eClmg_RGB2HSI_I(eClmage *SrcImage,
                       eClmage *&DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.6. Histograms 直方圖運算

EzCheck Vision Library 提供對影像的色彩、亮度分布進行統計以及對影像的直方圖進行等化等功能。

eClmg_Histogram

計算一張影像的色彩強度直方圖。統計整張影像在各通道與色彩強度上的像素點數目。

函式原形：

```
BOOL   eClmg_Histogram(eClmage *SrcImage,
                       int rHist[256],
                       int gHist[256],
                       int bHist[256] );
```

傳入值：

SrcImage	**in**	來源影像
rHist	**out**	紅色的強度直方圖，強度範圍從 0 至 255
gHist	**out**	綠色的強度直方圖，強度範圍從 0 至 255
bHist	**out**	藍色的強度直方圖，強度範圍從 0 至 255

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

---

eCImg_Equalize

使影像變成等化的色彩強度。Equalize 可讓影像的對比度提升，一般來說，當想增強影像的黑暗細節處時，使用 Equalize 可使黑暗處的細節較明顯。

函式原形：

```
BOOL   eCImg_Equalize( eCImage* SrcImage,
                       eCImage* DstImage);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

4.7. Rotation 影像旋轉

EzCheck Vision Library 提供四種不同特性的影像轉換。當影像以中心點為旋轉中心旋轉後，影像是否要隨之改變大小，旋轉後留下的空白區域以何種方式填滿都可以依照需求進行運算。

4.7.1. eCImg_Rotation1

旋轉影像。旋轉留下的空缺以黑色填滿，旋轉後不改變影像大小。

函式原形：

```
BOOL   eCImg_Rotation1(eCImage *SrcImage,
                       eCImage *&DstImage,
                       FLOAT32 Angle);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
FLOAT32	**in**	旋轉角度

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：

圖十七、　eCImg_Rotation1 運算結果

4.7.2. eCImg_Rotation2

旋轉影像。旋轉留下的空缺以黑色填滿，旋轉後改變影像大小。

函式原形：

```
BOOL   eCImg_Rotation2(eCImage *SrcImage,
                       eCImage *&DstImage,
                       FLOAT32 Angle);
```

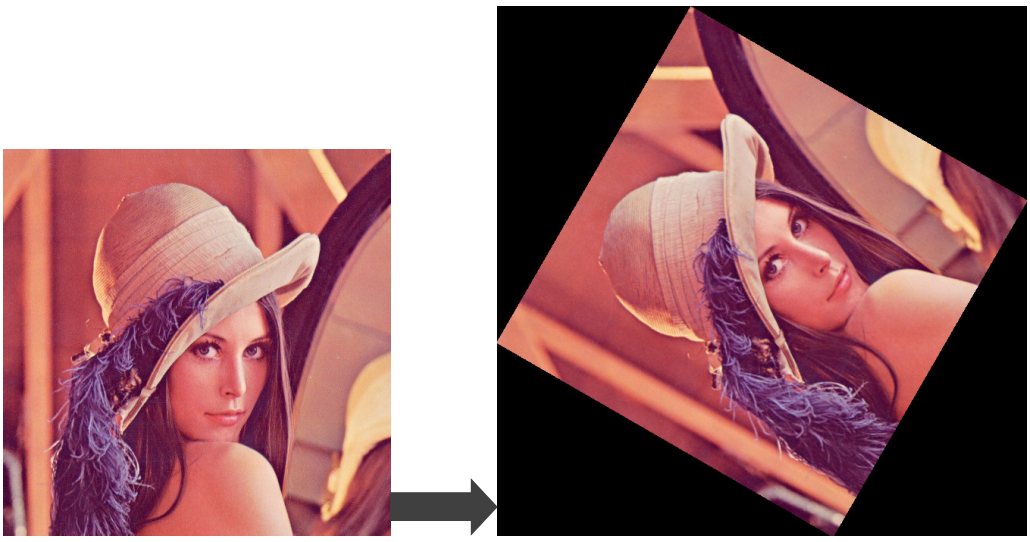
傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
FLOAT32	**in**	旋轉角度

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：



圖十八、　eCImg_Rotation2 運算結果

4.7.3. eCImg_Rotation3

旋轉影像。旋轉留下的空缺以影像邊緣延展，旋轉後不改變影像大小。

函式原形：

```
BOOL   eCIma  Rotation3(eCImage *SrcImage,
                        eCImage *&DstImage,
                        FLOAT32 Angle);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
FLOAT32	**in**	旋轉角度

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：

圖十九、　eCImg_Rotation3 運算結果

4.7.4. eCImg_Rotation4

旋轉影像。旋轉留下的空缺以影像邊緣延展，旋轉後改變影像大小。

函式原形：

```
BOOL   eCIma  Rotation4(eCImage *SrcImage,
                        eCImage *&DstImage,
                        FLOAT32 Angle);
```

傳入值：

SrcImage	**in**	來源影像
DstImage	**out**	目的影像
FLOAT32	**in**	旋轉角度

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：

圖二十、　eCImg_Rotation4 運算結果

4.8. 取得 ErrorCode

傳入的影像或者參數設定錯誤，都可能使影像處理函式無法正常動作。當影像處理函式未如預期的執行，或者回傳 FASLE 時，可將錯誤代碼輸出進行除錯。

4.8.1. eCImg_GetErrorCode

取得最後一個影像處理函式回傳的 error code。

函式原形：

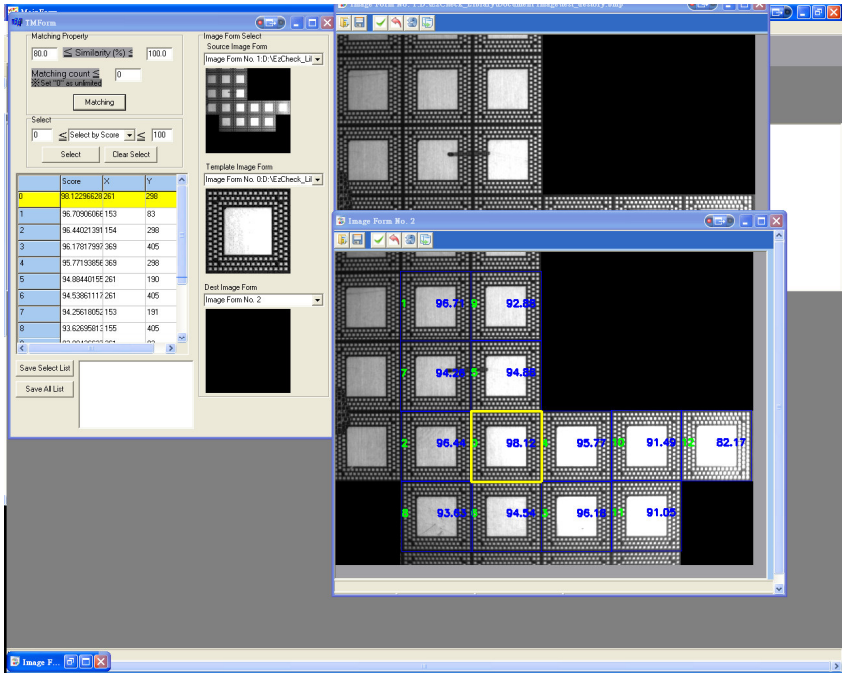
```
INT32   eCImg_GetErrorCode();
```

回傳值：

運作中回報的 ErrorCode。請參考 ErrorCode 列表。

5.樣板比對－eCTM

eCTM 提供在影像上找尋特定相似區域的功能。使用者可指定一個做為樣板的影像，並且在另一張影像上尋找與之相似的區域，取得其相似度與位置。內建多種標記相似區域的顯示影像，從影像即可取得比對資訊。



圖二十一、　以 eCTM 對影像進行樣板比對

5.1.　eCTM 主要特色

- **使用 Correlation coefficient matching 演算法，尋找指定相似度區間的相似區域**

  事先限制相似度區間，將可以減少相似度過低的不必要資訊。

- **可依照相似區間的相似度、座標過濾或排序比對結果**

  每個相似區域都包含相似度、與 XY 座標資訊，利用這些資訊進行相似區域的過濾，尋找到真正重要的資訊。

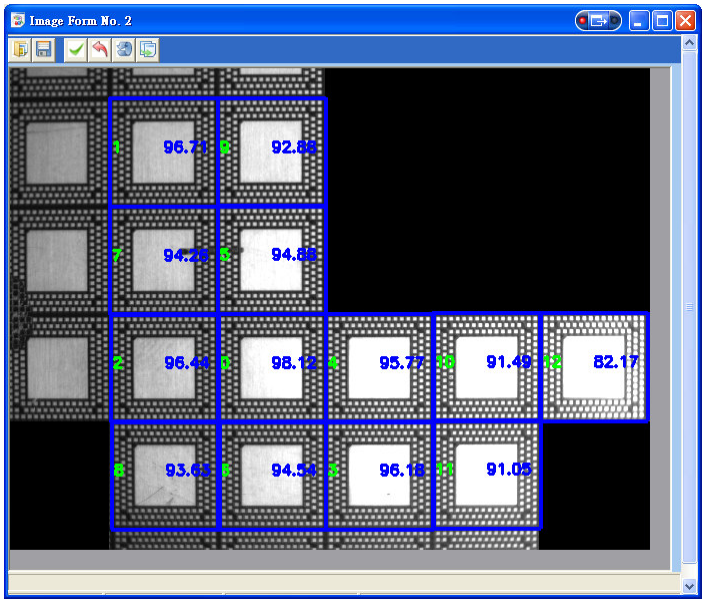
- **可將相似區域結果資訊輸出為文件**

  將相似區間結果輸出為文件檔，幫助事後的驗證與檢查。

- **可取得包含相似度與相似度排名的影像**

  可透過 eCTM，輸出帶有簡易標記介面的影像。影像上將框選出相似區域的

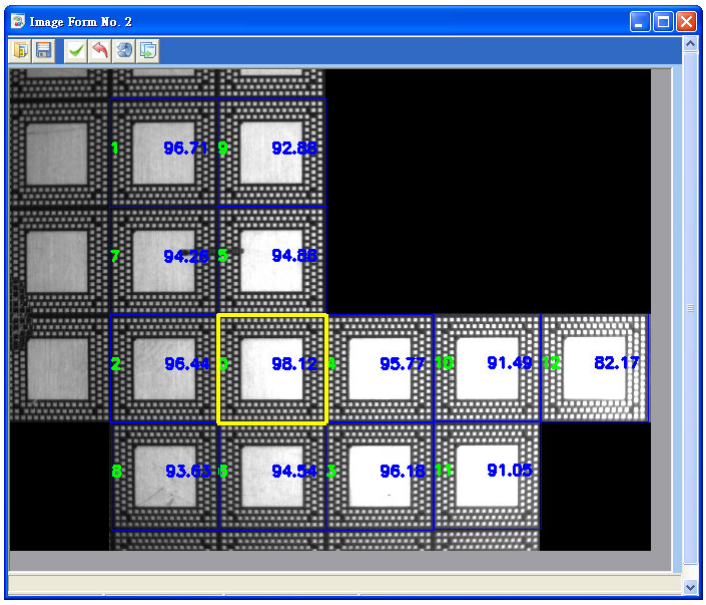
位置，並且標明其排名以及相似度。



圖二十二、 標記出所有相似區域的影像

➢ **可取得單一相似區域的標記影像**

可透過 eCTM 輸出帶有簡易標記介面的影像。可指定僅標記出單一相似區域，或者以強調的方式標記。



圖二十三、 標記單一相似區域的影像。保留其他相似區域資訊標記，但以強調方式標記其一。

5.2. eCTM 主要功能介紹

5.2.1. struct _MATCH

eCTM 的比對相似區域結果型態

```
struct   MATCH
{
     int x. v:
     double Score;
}
```

x, y　　相似區域的中心點座標

Score　　相似度

5.2.2. SelectMatching

存放相似區域結果的結構陣列，使用者可自由存取。所有篩選以及排序都會由此結構陣列輸出。

```
_MATCH *SelectMatching
```

5.2.3. eCTM

eCTM 建構式。

```
eCTM ( );
```

5.2.4. Release

eCTM 資源釋放。eCTM 內包含若干運行時需要的資料結構，建議使用者在使用完畢後釋放 eCTM。

函式原形：

```
BOOL   Release( );
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.5. LoadTemplateImage

自檔案讀取樣版影像。

函式原形：

```
BOOL   LoadTemplateImage(const char* pszPathName);
BOOL   LoadTemplateImage(const UNICHAR* pszPathName);
```

傳入值：

pszPathName in 影像檔案路徑與名稱

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.6. SetTemplateImage

自 eCImage 設定樣版影像。

函式原形：

```
BOOL   SetTemplateImage(eCImage *SrcImage);
```

傳入值：

SrcImage in 設定為樣板影像的來源影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.7. DoTemplateMatching

eCTM 核心功能，進行樣板比對。當 eCTM 已經成功讀取或者設定樣版影像後方可運作。

函式原形：

```
BOOL   DoTemplateMatching(eCImage *SrcImage);
```

傳入值：

SrcImage	**in**	要進行比對的影像。DoTemplateMatching 將會在此影像上尋找與樣版影像相似的區域。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example code:

```
eCTM* TM_Sample = new eCTM;
int First_Match_Score = 0;
TM_Sample ->SetTemplateImage(TemplateImage);
TM_Sample ->DoTemplateMatching(SrcImage);
First_Mathc_Score = TM_Sample-> SelectMatching[0].Score;//取得最大相似度區域的相似度
```

5.2.8. SetProperty

設定樣板比對的屬性。事先設定樣板比對的屬性有助於加快樣板比對的速度，並且避免過多不必要的資訊。

函式原形：

```
BOOL   SetProperty(float Min.
                   float Max.
                   int MaxCount);
```

傳入值：

Min	**in**	樣板比對的最小相似度(0.0~1.0)。小於此相似度的將不採記
Max	**in**	樣板比對的最大相似度(0.0~1.0)。大於此相似度的將不採記
MaxCount	**in**	樣板比對的最大採記數目。自相似度最大的區域開始，取得 MaxCount 個相似區域資訊 ※Min 預設為 0.8，Max 預設為 1.0，MaxCount 預設為 100

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example code:

```
eCTM* TM_Sample = new eCTM;
TM_Sample->SetProperty(0.6, 1.0, 10);//取得 60%~100%相似度，相似度前 10 的相似區域資訊
```

5.2.9. GetPropertyMin

取得樣板比對屬性的相似度最小值。

函式原形：

```
flaot   GetPropertyMin();
```

回傳值：

樣板比對時的相似度下限，與 SetProperty 相呼應。

5.2.10. GetPropertyMax

取得樣板比對屬性的相似度最大值。

函式原形：

```
flaot   GetPropertyMax();
```

回傳值：

樣板比對時的相似度上限，與 SetProperty 相呼應。

5.2.11. GetPropertyMaxCount

取得樣板比對屬性的樣板比對的最大採記數目。

函式原形：

```
flaot   GetPropertyMaxCount();
```

回傳值：

回傳樣板比對時最大採樣數目，與 SetProperty 相呼應。

5.2.12. GetAllMatchingImage

取得標記所有比對結果的影像。

函式原形：

```
BOOL   GetAllMatchingImage(eCImage *&DstImage);
```

傳入值：

DstImage **out** 接收所有比對結果影像的 eCImage

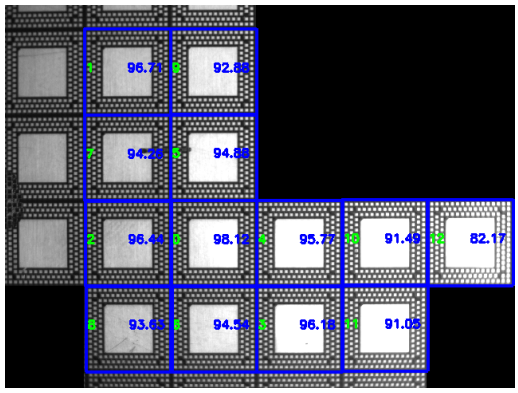
回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example code:

```
//Template Matching 完成後
eCImage *dstImage = new eCImage;
TM_Sample->GetAllMatchingImage(dstImage);
```

運作結果：



5.2.13. GetSelectMatchingImage

取得標記篩選後比對結果的影像。進行篩選之前，輸出結果與 GetAllMatchingImage 相同。

函式原形：

```
BOOL   GetSelectMatchingImage(eCImage *&DstImage);
```

傳入值：

DstImage　　**out**　接收所有比對結果影像的 eCImage。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.14. GetSingleMatchingImage

取得標記單一比對結果的影像。

函式原形：

```
BOOL   GetSingleMatchingImage(eCImage *&DstImage,
                              int Number);
```

傳入值：

DstImage　　**out**　接收所有比對結果影像的 eCImage。

Number　　**in**　欲標記的比對結果編號，以當時排序為準。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.15. GetRemarkMatchingImage

取得強調單一比對結果的影像。

函式原形：

```
BOOL   GetRemarkMatchingImage(eCImage *&DstImage,
                              int Number);
```

傳入值：

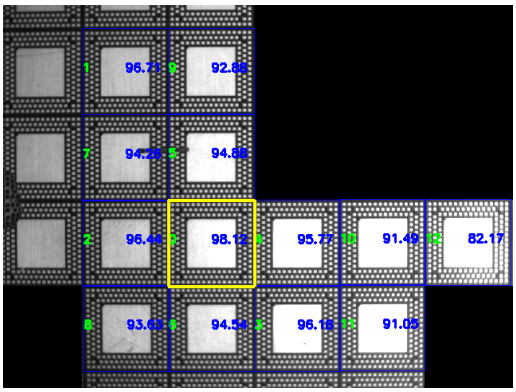
DstImage　　**out**　接收所有比對結果影像的 eCImage。

Number　　**in**　欲強調的比對結果編號，以當時排序為準。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：



Tips：所有 eCTM 中的 Get***Image 都必須在 DoTemplateMatching 正確完成後才會回傳正確的影像。

5.2.16. SelectMatchByX

以 X 座標篩選比對結果，並取得篩選後的比對結果數目。

函式原形：

```
int   SelectMatchBvX(int MaxValue,
                     int MinValue);
```

傳入值：

MaxValue	**in**	篩選的 X 座標上限。忽略 X 座標大於此數值的相似區域。
MinValue	**in**	篩選的 X 座標下限。忽略 X 座標小於此數值的相似區域。

回傳值：

篩選後留下的相似區域數量。

5.2.17. SelectMatchByY

以 Y 座標篩選比對結果，並取得篩選後的比對結果數目。

函式原形：

```
int   SelectMatchByY(int MaxValue,
                     int MinValue);
```

傳入值：

MaxValue	**in**	篩選的 Y 座標上限。忽略 Y 座標大於此數值的相似區域。
MinValue	**in**	篩選的 Y 座標下限。忽略 Y 座標小於此數值的相似區域。

回傳值：

篩選後留下的相似區域數量。

5.2.18. SelectMatchByScore

以相似度篩選比對結果，並取得篩選後的比對結果數目。

函式原形：

```
int   SelectMatchByScore(int MaxValue,
                         int MinValue);
```

傳入值：

MaxValue	**in**	篩選的相似度上限。忽略相似度大於此數值的相似區域。
MinValue	**in**	篩選的相似度下限。忽略相似度小於此數值的相似區域。

回傳值：

篩選後留下的相似區域數量。

5.2.19. ClearSelect

將比對結果重置為 DoTemplateMatching 後的最初比對結果。

函式原形：

```
BOOL   ClearSelect();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.20. SortMatch

排序比對結果。使用者可選擇以何種特徵以及何種排列順序進行排序。

函式原形：

```
BOOL   SortMatch(int SortType,
                 int Order);
```

傳入值：

SortType	**in**	排序的特徵。0：相似度、1：X 座標、2：Y 座標
Order	**in**	排序的方向。0：由大至小、1：由小至大

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.21. SaveAllMatchingList

將所有比對結果存出為文件檔。

函式原形：

```
BOOL   SaveAllMatchingList(char *FileName);
```

傳入值：

FileName	**in**	輸出文件檔的完成檔名路徑。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.22. SaveSelectMatchingList

將篩選後的比對結果存出為文件檔。

函式原形：

```
BOOL   SaveSelectMatchingList (char *FileName);
```

傳入值：

FileName **in** 輸出文件檔的完成檔名路徑。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

5.2.23. GetErrorCode

取得 eCTM 回報的 error code。

函式原形：

```
INT32   GetErrorCode ();
```

回傳值：

運作中回報的 ErrorCode。請參考 ErrorCode 列表。

6.斑點分析－eCBlob

eCBlob 提供對影像連通區域或獨立區間進行分析的功能。使用者可以藉由各個連通區域的資訊，包括位置、重心、面積...等資訊進行過濾，也可以將相鄰的區域進行整合。

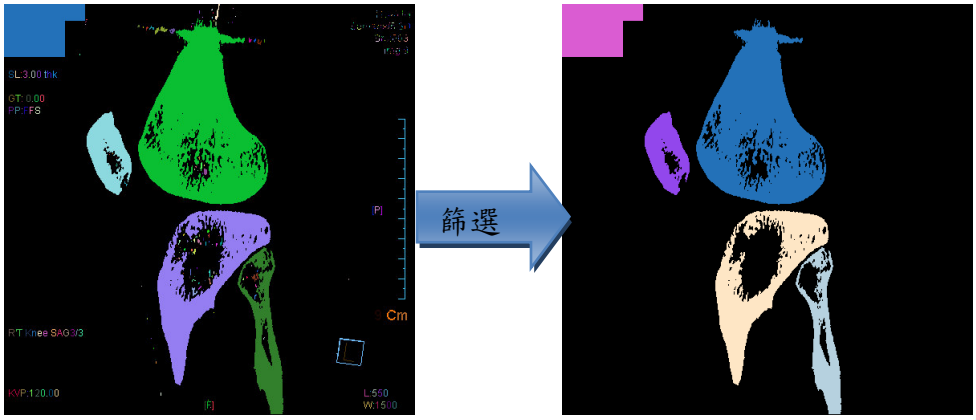
6.1. eCBlob 主要特色

- **分析出影像上所有的連通區域，並以多色區塊表示不同的連通區域**

不同的連通區域將會以不同的顏色表示，未檢測為連通區塊的部分則顯示為黑色。

- **支援分析結果過濾篩選以及排序功能**

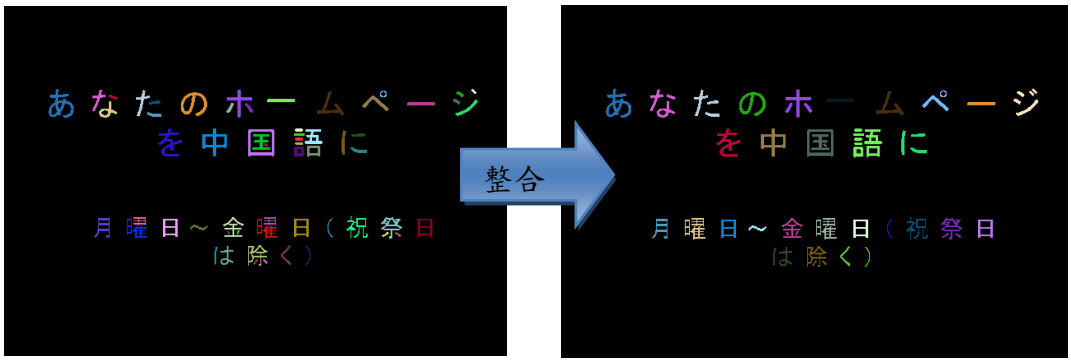
可依據區域面積、區域位置等資訊，對分析結果進行篩選或者排序，去除破碎的小面積區塊或者其他不需要的資訊。



圖二十四、 eCBlob 支援的篩選功能，可過濾不需要的 blob

- **支援區塊整合功能，將接近的獨立區塊整合為同一區塊**

除了可以將區塊篩選過濾之外，還可以將特定範圍內的小區塊合併為一同一個區塊，在文字辨識的前處理上尤其重要。



圖二十五、 eCBlob 支援的整合功能，可整合鄰近的 blob

- **可取得所有連通區域的分析影像，或篩選過的分析影像**

可將記錄連通區塊的多色顯示影像輸出為影像檔案，也可將僅包含篩選結果的影像輸出，利於其他應用或驗證。

6.2. eCBlob 主要功能介紹

eCBlob 主要功能包含自影像上分析 blob 的核心功能之外，還包含取得 blob 相關資訊，或者將 blob 輸出為影像檔案等功能。自影像上分析的所有 blob，可透過篩選或合併，取得其選擇性的子集合，稱為 Select blob。大部分的功能都會對所有 blob 與 select blob 分別提供函式支援。

6.2.1. eCBlob

eCBlob 建構式。

函式原形：

```
eCBlob();
```

6.2.2. ~eCBlob

eCBlob 解構式。

函式原形：

```
~eCBlob();
```

6.2.3. DoBlobAnalysis

對來源影像進行 blob analysis。

函式原形：

```
BOOL   DoBlobAnalysis (eCImage* SrcImage,
                       UINT8 ClassType);
```

傳入值：

SrcImage	**in**	要進行 blob 分析的來源影像
ClassType	**in**	分析的 blob 類型。1:白底黑 blob、2:黑底白 blob、3:黑白 blob、4:較暗 blob、5:亮度中等的 blob、6:較亮 blob

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example code:

```
eCImage *SrcImage = new eCImage;
eCBlob Blob;//宣告一個 eCBlob 物件
SrcImage->Load("C:\\Test_Image.bmp");
eCImg_AbsoluteThreshold(SrcImage, SrcImage, 100);//對 SrcImage 做臨界值為 100 的二值化，並且直接修改 SrcImage
Blob.DoBlobAnalysis(SrcImage, 1);//對 SrcImage 上的白底黑 Blob 進行 blob analysis
```

Tips：建議進行 DoBlobAnalysis 之前先進行影像的二值化，並且使用黑底白 blob 或者白底黑 blob，可得到較好的 blob analysis 結果。

6.2.4. SelectBlobUsingFeature

在 DoBlobAnalysis 完成後，對 blob 進行篩選。完成 Select blob 後，與 Select 相關之函式方可運作。

函式原形：

```
BOOL   SelectBlobUsingFeature(INT32 FeatureType,
                              INT32 OptionType,
                              INT32 MinTHValue,
                              INT32 MaxTHValue);
```

傳入值：

FeatureType	篩選的特徵。請參考 SELECTFEATURE。
OptionType	篩選後所要留下的 blob。 1:大於等於 MinTHValue、 2:小於等於 MaxTHValue、 3:介於 MaxTHValue 與 MinTHValue 之間、 4:大於 MaxTHValue 並且小於 MinTHValue。
MinTHValue	篩選的特徵值。作用方式見 OptionType。
MaxTHValue	篩選的特徵值。作用方式見 OptionType。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example Code:

```
//After DoBlobAnalysis
Blob.SelectBlobUsingFeature(1, 1, 200, 0);//取得 area 大於 200 的 blob
```

6.2.5. SortBlobUsingFeature

在 DoBlobAnalysis 完成後，對 blob 進行排序。

函式原形：

```
BOOL   SortBlobUsingFeature(INT32 FeatureType,
                            INT32 SortType);
```

傳入值：

FeatureType　排序的特徵。請參考 SELECTFEATURE。

SortType　排序種類。1:升冪、2:降冪、3:初始值

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example Code:

```
//After DoBlobAnalysis
Blob. SortBlobUsingFeature (1, 1);//blob 以 area 數值進行升冪排序
```

6.2.6. GetBlobParameter

在 DoBlobAnalysis 完成後，取得該次 blob analysis 的資訊。

函式原形：

```
PBA_ROI   GetBlobParameter ();
```

回傳值：

PBA_ROI。Blob analysis 的資訊，請參考 BLOBANALYSIS_TAG。

6.2.7. CalculateAdvancedFeature

在 DoBlobAnalysis 完成後，計算 blob 的進階特徵。運作成功將回傳 TRUE，否則回傳 FALSE。

函式原形：

```
BOOL   CalculateAdvancedFeature(eCImage* SrcImage,
                                INT32 FeatureType);
```

傳入值：

SrcImage　計算特徵的參考影像。請傳入與 DoBlobAnalysis 傳入值相同的影像。

FeatureType　欲計算的進階特徵。進階特徵請參考 SELECTFEATURE

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.8. GetBlobBasicFeature

在 DoBlobAnalysis 完成後，取得所有 blob 的基本特徵。基本特徵定義於 DataFile.h。

函式原形：

```
PBLOB_BASICFEATURE   GetBlobBasicFeature ();
```

回傳值：

PBLOB_BASICFEATURE。紀錄 blob 基本特徵，請參考 _BLOBBASICFEATURE_TAG。

Example code:

```
//After DoBlobAnalysis.
BLOB_BASICFEATURE *sB;//宣告存放 blob 基本特徵的結構陣列
sB = Blob. GetBlobBasicFeature ();
int Blob_Area = sB[0].Area; //取得第 0 個 blob 的面積
```

6.2.9. GetBlobAdvancedFeature

在 DoBlobAnalysis 以及 CalculateAdvancedFeature 完成後，取得所有 blob 的進階特徵。進階特徵定義於 DataFile.h。

函式原形：

```
PBLOB_ADVANCEDFEATURE   GetBlobAdvancedFeature ();
```

回傳值：

PBLOB_ADVANCEDFEATURE。存放 blob 進階特徵，請參考 _BLOBADVANCEDFEATURE_TAG。

Example code:

```
//After DoBlobAnalysis.
BLOB_ADVANCEDFEATURE *sA;//宣告存放 blob 基本特徵的結構陣列
sA = Blob. GetBlobAdvancedFeature ();
FLOAT32 Blob_SigmaX = sB[0]. SigmaX; //取得第 0 個 blob 的 SigmaX
```

6.2.10. GetBlobConvexHull

在 DoBlobAnalysis 完成後，取得所有 blob 的 Convex hull 資訊。Convex hull 定義於 DataFile.h。

函式原形：

```
PConvex_RESULT   GetBlobConvexHull ();
```

回傳值：

PConvex_RESULT。存放 blob 的 Convex hull 資訊，請參考 _CONVEXHULL_TAG。

6.2.11. GetSelectBlobParameter

在 DoBlobAnalysis 與 SelectBlobUsingFeature 完成後，取得 select blob 的資訊。

函式原形：

```
PBA_ROI   GetSelectBlobParameter ();
```

回傳值：

PBA_ROI。取得 select blob 的資訊。請參考 _BLOBANALYSIS_TAG。

6.2.12. CalculateSelectBlobAdvancedFeature

在 DoBlobAnalysis 與 SelectBlobUsingFeature 完成後，計算篩 select blob 的進階資訊。

函式原形：

```
BOOL   CalculateSelectBlobAdvancedFeature(eCImage* SrcImage,
                                          INT32 FeatureType);
```

傳入值：

SrcImage 計算特徵的參考影像。請傳入與 DoBlobAnalysis 傳入值相同的影像。

FeatureType 欲計算的進階特徵。進階特徵請見 SELECTFEATURE

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.13. GetSelectBlobBasicFeature

在 DoBlobAnalysis 與 SelectBlobUsingFeature 完成後，取得所有 select blob 的基本特徵。基本特徵定義於 DataFile.h。

函式原形：

```
PBLOB_BASICFEATURE   GetSelectBlobBasicFeature();
```

回傳值：

PBLOB_BASICFEATURE。Blob 基本特徵，請參考 BLOBBASICFEATURE_TAG。

Example code:

```
//After DoBlobAnalysis and SelectBlobUsingFeature.
BLOB_BASICFEATURE *sB;//宣告存放 blob 基本特徵的結構陣列
sB = Blob. GetSelectBlobBasicFeature ();
int Blob_Area = sB[0].Area; //取得第 0 個 select blob 的面積
```

6.2.14. GetSelectBlobAdvancedFeature

在 DoBlobAnalysis、SelectBlobUsingFeature 以及 CalculateAdvancedFeature 完成後，取得所有 blob 的進階特徵。進階特徵定義於 DataFile.h。

函式原形：

```
PBLOB_ADVANCEDFEATURE   GetSelectBlobAdvancedFeature();
```

回傳值：

PBLOB_ADVANCEDFEATURE。blob 進階特徵，請參考 _BLOBADVANCEDFEATURE_TAG。

Example code:

```
//After DoBlobAnalysis, SelectBlobUsingFeature, and
CalculateSelectBlobAdvancedFeature.
BLOB_ADVANCEDFEATURE *sA;//宣告存放 blob 基本特徵的結構陣列
sA = Blob. GetSelectBlobAdvancedFeature ();
FLOAT32 Blob_SigmaX = sB[0]. SigmaX; //取得第 0 個 select blob 的 SigmaX
```

6.2.15. GetSelectBlobConvexHull

在 DoBlobAnalysis 與 SelectBlobUsingFeature 完成後，取得 select blob 的 Convex hull 資訊。Convex hull 定義於 DataFile.h。

函式原形：

```
PConvex_RESULT   GetSelectBlobConvexHull();
```

回傳值：

PConvex_RESULT。存放 blob 的 Convex hull 資訊，請參考 _CONVEXHULL_TAG。

6.2.16. SaveBlobImage/ SaveSelectBlobImage

儲存以多色區塊標記 blob/select blob 的影像。

函式原形：

Borland C++ Builder version:

```
BOOL   SaveBlobImage(AnsiString FileName);
BOOL   SaveBlobImage(WideString FileName);
BOOL   SaveSelectBlobImage(AnsiString FileName);
BOOL   SaveSelectBlobImage(WideString FileName);
```

Visual C++ version:

```
BOOL   SaveBlobImage(CString FileName);
BOOL   SaveSelectBlobImage(CString FileName);
```

傳入值：

FileName **in** 存檔路徑檔名

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.17. SaveSingleBlob/ SaveSingleSelectBlob

儲存單一blob/select blob的影像，指定該blob的方式有用編號與座標，座標為絕對座標，存出的檔案有可有二值化與灰階圖2種結果可選。儲存成功將回傳TRUE，否則回傳FALSE。

函式原形：

Borland C++ Builder version:

```
BOOL   SaveSingleBlob(eCImage* SrcImage,
                      INT32 X  axis.
                      INT32 Y  axis.
                      WideString SaveName,
                      UINT8 ProcMode);
BOOL   SaveSingleBlob(eCImage* SrcImage,
                      UINT16 NstBlob.
                      WideString SaveName,
                      UINT8 ProcMode);
BOOL   SaveSingleBlob(eCImage* SrcImage,
                      INT32 X  axis.
                      INT32 Y  axis.
                      AnsiString SaveName,
                      UINT8 ProcMode);
BOOL   SaveSingleBlob(eCImage* SrcImage,
                      UINT16 NstBlob.
                      AnsiString SaveName,
                      UINT8 ProcMode);
BOOL   SaveSingleSelectBlob(eCImage* SrcImage,
                            INT32 X  axis.
                            INT32 Y  axis.
                            WideString SaveName,
                            UINT8 ProcMode);
BOOL   SaveSingleSelectBlob(eCImage* SrcImage,
                            UINT16 NstBlob.
                            WideString SaveName,
                            UINT8 ProcMode);
BOOL   SaveSingleSelectBlob(eCImage* SrcImage,
                            INT32 X  axis.
                            INT32 Y  axis.
                            AnsiString SaveName,
                            UINT8 ProcMode);
BOOL   SaveSingleSelectBlob(eCImage* SrcImage,
                            UINT16 NstBlob.
                            AnsiString SaveName,
                            UINT8 ProcMode);
```

Visual C++ version:

```
BOOL   SaveSingleBlob(eCImage* SrcImage,
                      INT32 X  axis.
                      INT32 Y  axis.
                      CString SaveName,
                      UINT8 ProcMode);
BOOL   SaveSingleBlob(eCImage* SrcImage,
                      UINT16 NstBlob.
                      CString SaveName,
                      UINT8 ProcMode);
BOOL   SaveSingleSelectBlob(eCImage* SrcImage,
                            INT32 X  axis.
                            INT32 Y  axis.
                            CString SaveName,
                            UINT8 ProcMode);
BOOL   SaveSingleSelectBlob(eCImage* SrcImage,
                            UINT16 NstBlob.
                            CString SaveName,
                            UINT8 ProcMode);
```

傳入值：

SrcImage	**in**	來源影像。請傳入與 blob analysis 相同，或者未二值化影像。
X_axis, Y_axis	**in**	Blob 的絕對位置座標。輸入該 blob 範圍內的座標即可。
NstBlob	**in**	Blob 的編號。
SaveName	**in**	存檔的路徑與檔名
ProcMode	**in**	存出的圖檔為黑白或灰階。請參考 PROCESSMODE

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.18. SaveAllBlob/SaveAllSelectBlob

分別儲存**所有**單一 blob/select blob 的影像，存出的檔案有可有二值化與灰階圖 2 種結果可選。將所有 blob 的影像依序個別存出，並以其編號作為檔案名稱。儲存成功將回傳 TRUE，否則回傳 FALSE。

函式原形：

Borland C++ Builder version:

```
BOOL   SaveAllBlob(eCImage* SrcImage,
                   WideString SavePath,
                   AnsiString ImageType,
                   UINT8 ProcMode);
BOOL   SaveAllBlob(eCImage* SrcImage,
                   AnsiString SavePath,
                   AnsiString ImageType,
                   UINT8 ProcMode);
BOOL   SaveAllSelectBlob(eCImage* SrcImage,
                         WideString SavePath,
                         AnsiString ImageType,
                         UINT8 ProcMode);
BOOL   SaveAllSelectBlob(eCImage* SrcImage,
                         AnsiString SavePath,
                         AnsiString ImageType,
                         UINT8 ProcMode);
```

Visual C++ version:

```
BOOL   SaveAllBlob(eCImage* SrcImage,
                   CString SavePath,
                   CString ImageType,
                   UINT8 ProcMode);
BOOL   SaveAllSelectBlob(eCImage* SrcImage,
                         CString SavePath,
                         AnsiString ImageType,
                         UINT8 ProcMode);
```

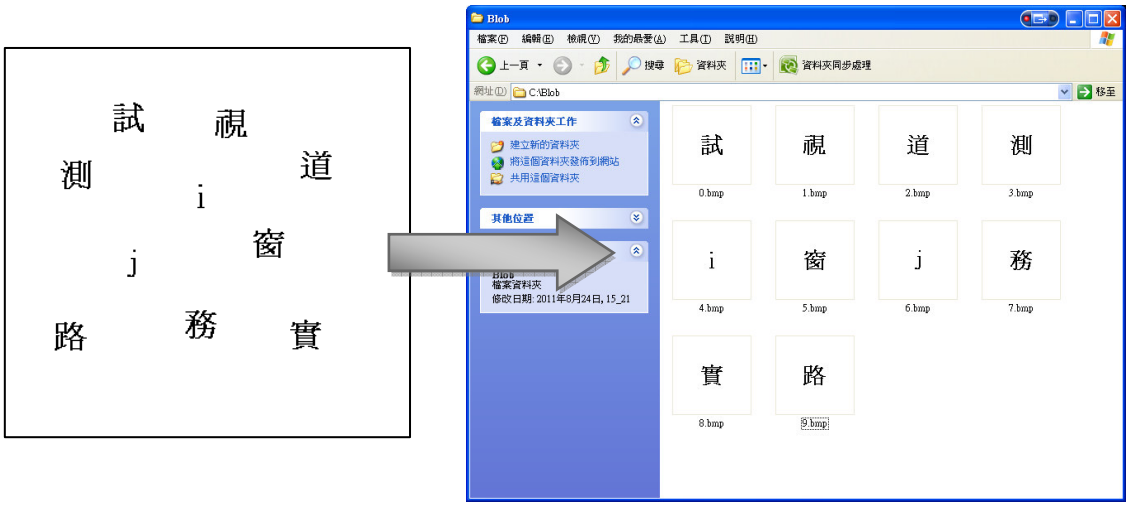
傳入值：

SrcImage	**in**	來源影像。請傳入與 blob analysis 相同，或者未二值化影像。
SavePath	**in**	存檔的路徑。
ImageType	**in**	存檔的副檔名。建議使用".bmp"
ProcMode	**in**	存出的圖檔為黑白或灰階。請參考 PROCESSMODE

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：



圖二十六、 SaveAllBlob 結果。將所有 blob 的影像個別存出，並且以自動編號設定檔案名稱。

Tips：SaveBlobImage 等功能，可幫助使用者直接截取 blob 的影像，這在當使用者需要將這些 blob 的影像轉換為後續 OCR 的資料庫時相當實用。

6.2.19. AutoMerge

自動合併臨近的 blob。運算成功將回傳 TRUE，否則回傳 FALSE。

函式原形：

```
BOOL   AutoMerge(eCImage* SrcImage,
                 INT32 SetWidth,
                 INT32 SetHeight);
```

傳入值：

SrcImage	**in**	來源影像。請傳入與 blob analysis 相同的影像。
SetWidth	**in**	合併 blob 的寬。
SetHeight	**in**	合併 blob 的高。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.20. Merge

合併 Merge List 中輸入的 blob。Merge List 相關函式如後述。

函式原形：

```
BOOL   Merge(eCImage* SrcImage);
```

傳入值：

SrcImage **in** 來源影像。請傳入與 blob analysis 相同的影像。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.21. AddMergeList

將要合併的 blob 加入 Merge List 中。可用 blob 編號或者座標加入特定 blob。運算成功將回傳 TRUE，否則回傳 FALSE。Merge List 中的 blob 將以 eCBlob::Merge 合併在一起。

函式原形：

```
BOOL   AddMergeList (UINT16 NstBlob);
BOOL   AddMergeList (INT32 X_axis.
                     INT32 Y_axis);
```

傳入值：

NstBlob **in** Blob 編號。

X_axis, Y_axis **in** ~~Blob 位置~~。Blob 的位置座標

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.22. CleanMergeList

清除 MergeList 內的資訊。

函式原形：

```
BOOL   CleanMergeList ();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Tips：Merge 等功能的合併結果，可透過 GetSelectBlobBasicFeature、SaveSelectBlobImage…等 select blob 相關函式取得。

6.2.23. SaveTxt

將 blob 資訊存為文字檔，可選擇自行輸入檔案名稱，或者由系統自動儲存一個”BlobAnalysisDLL.txt”於當下使用的資料夾中。

函式原形：

Borland C++ Builder version:

```
BOOL   SaveTxt ();
BOOL   SaveTxt (AnsiString TxtName);
```

Visual C++ version:

```
BOOL   SaveTxt ();
BOOL   SaveTxt(CString TxtName);
```

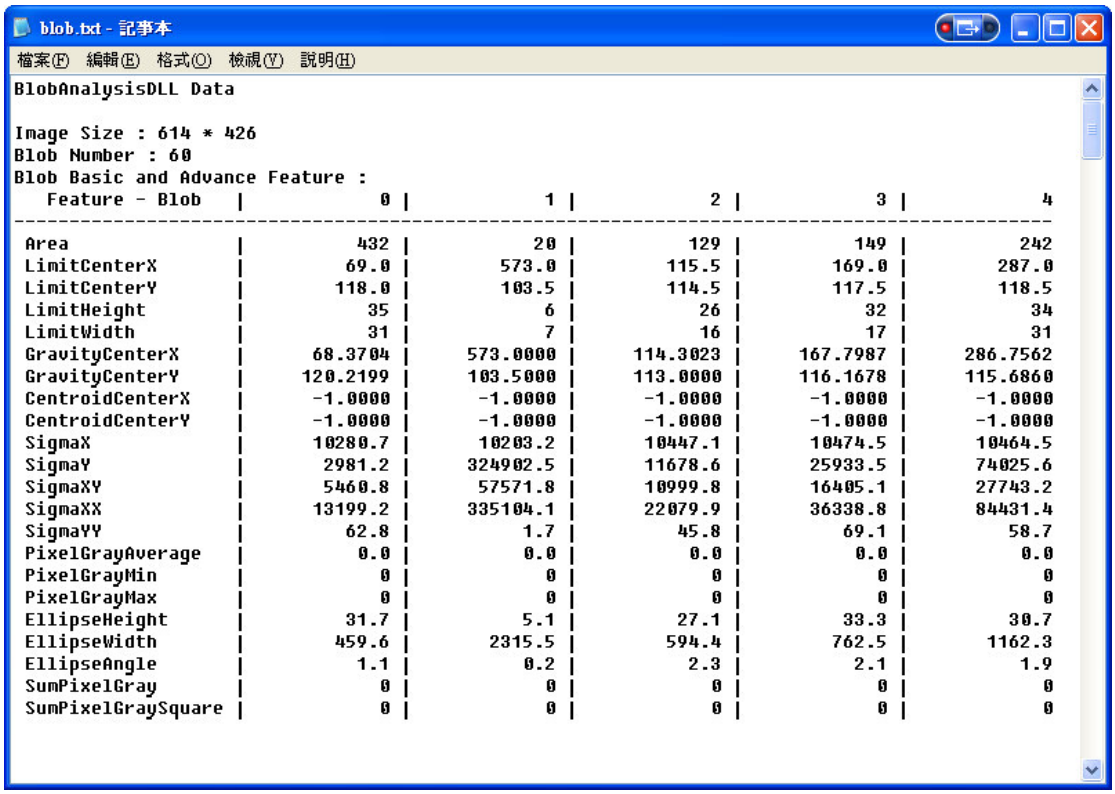
傳入值：

TxtName in 存文字檔的路徑檔名

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

運作結果：



圖二十七、 SaveTxt 輸出之 blob 資訊

6.2.24. CleanBuffer

清除所有 blob analysis 的資訊。運算成功將回傳 TRUE，否則回傳 FALSE。

函式原形：

```
BOOL   CleanBuffer();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.25. CleanSelectBuffer

將 select blob 的資訊重置為 blob analysis 後原始的 blob 資訊。

函式原形：

```
BOOL   CleanSelectBuffer();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.26. SingleBlobFree

消去某特定 blob，可依照位置或者編號指定要消去的 blob。

函式原形：

```
BOOL   SingleBlobFree(UINT16 NstBlob);
BOOL   SingleBlobFree(UINT16 X_axis.
                      UINT16 Y_axis);
```

傳入值：

NstBlob	**in**	Blob 的編號
X_axis, Y_axis	**out**	Blob 的位置座標

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

6.2.27. GetErrorCode

取得 eCBlob 回報的 error code。請參考 ErrorCode 列表。

函式原形：

```
INT32   GetErrorCode ();
```

回傳值：

eCBlob 運作中回報的 error code。請參考 ErrorCode 列表。

6.3. eCBlob 資料結構

以下列出 eCBlob 中使用的專用資料格式。相關資料格式宣告於 DataFile.h 中。

6.3.1. struct _BLOBANALYSIS_TAG

blob analysis 的資訊

宣告內容：

```
struct  BLOBANALYSIS_TAG
{
    INT32       StartX:
    INT32       StartY:
    INT32       Width:
    INT32       Height;
    UINT16      Count:
};
```

內容簡介：

StartX	該次 blob analysis 的起始點 X 座標。
StartY	該次 blob analysis 的起始點 Y 座標。
Width	該次 blob analysis 的寬。
Height	該次 blob analysis 的高。
Count	該次 blob analysis 取得的 blob 總數。

6.3.2. struct _BLOBBASICFEATURE_TAG

blob 的基本特徵

宣告內容：

```
struct   BLOBBASICFEATURE  TAG
{
      UINT32    Area:
      FLOAT32    LimitCenterX:
      FLOAT32    LimitCenterY;
      UINT16    LimitHeight;
      UINT16    LimitWidth:
      FLOAT32    GravitvCenterX:
      FLOAT32    GravitvCenterY:
      FLOAT32    CentroidCenterX:
      FLOAT32    CentroidCenterY:
      FLOAT32    RectangleXY[4][2];
};
```

內容簡介：

Area	blob 的面積。
LimitCenterX	blob 的限制中心 X 座標。
LimitCenterY	blob 的限制中心 Y 座標。
LimitHeight	限制中心區域的高。
LimitWidth	限制中心區域的寬。
GravityCenterX	blob 的重心 X 座標。
GravityCenterY	blob 的重心 Y 座標。
CentroidCenterX	blob 的質心 X 座標。
CentroidCenterY	blob 的質心 Y 座標。
RectangleXY[4][2]	包圍 blob 方型區域的四個頂點座標。

Tips:

- 限制中心：在 0、22、45、68 度時繪製出的邊框範圍。
- 重心：最小外接圓之中心。
- 質心：所有像素座標平均。

6.3.3. struct _BLOBADVANCEDFEATURE_TAG

blob 的進階特徵

宣告內容：

```
struct  BLOBADVANCEDFEATURE_TAG
{
    FLOAT32   SigmaX:
    FLOAT32   SigmaY:
    FLOAT32   SigmaXY:
    FLOAT32   SigmaXX:
    FLOAT32   SigmaYY:
    FLOAT32   PixelGravAverage;
    UINT8     PixelGravMin:
    UINT8     PixelGravMax;
    FLOAT32   EllipseHeight;
    FLOAT32   EllipseWidth:
    FLOAT32   EllipseAngle:
    UINT32    SumPixelGrav:
    UINT32    SumPixelGravSquare:
};
```

內容簡介：

SigmaX	慣性橢圓(參數 D)
SigmaY	慣性橢圓(參數 E)
SigmaXY	慣性橢圓(參數 2B)
SigmaXX	慣性橢圓(參數 A)
SigmaYY	慣性橢圓(參數 C)
PixelGrayAverage	平均灰階值
PixelGrayMin	最小灰階值
PixelGrayMax	最大灰階值
EllipseHeight	慣性橢圓高
EllipseWidth	慣性橢圓寬
EllipseAngle	慣性橢圓角度
SumPixelGray	灰階值總和
SumPixelGraySquare	灰階值平方和

Tips:慣性橢圓為以限制中心為中心之外接橢圓。其方程式為：

$Ax^2 + 2Bxy + Cy^2 + Dx + Ey = 0$

6.3.4. struct _CONVEXHULL_TAG

Blob 的 Convex hull 資訊

宣告內容：

```
struct  CONVEXHULL_TAG
{
    UINT16 **X:
    UINT16 **Y:
    UINT8 *ConvexCount;
};
```

內容簡介：

X	該 blob 進行 Convex hull 計算後的 x 座標
Y	該 blob 進行 Convex hull 計算後的 y 座標
ConvexCount	該 blob 進行 Convex hull 後的數量

Tips:Convex Hull 為完整包圍目標物的最小面積的凸多邊形。

6.4. eCBlob 列舉型態

以下列出 eCBlob 中使用的列舉型態。

6.4.1. enum PROCESSMODE

Blob 的存檔模式

宣告內容：

```
enum PROCESSMODE
{
    Binarv_Mode = 1,
    Gray_Mode,
};
```

內容簡介：

Binary_Mode	設定黑白 2 色模式
Gray_Mode	設定灰階模式

6.4.2. enum SELECTFEATURE

用以篩選 blob 的特徵

宣告內容：

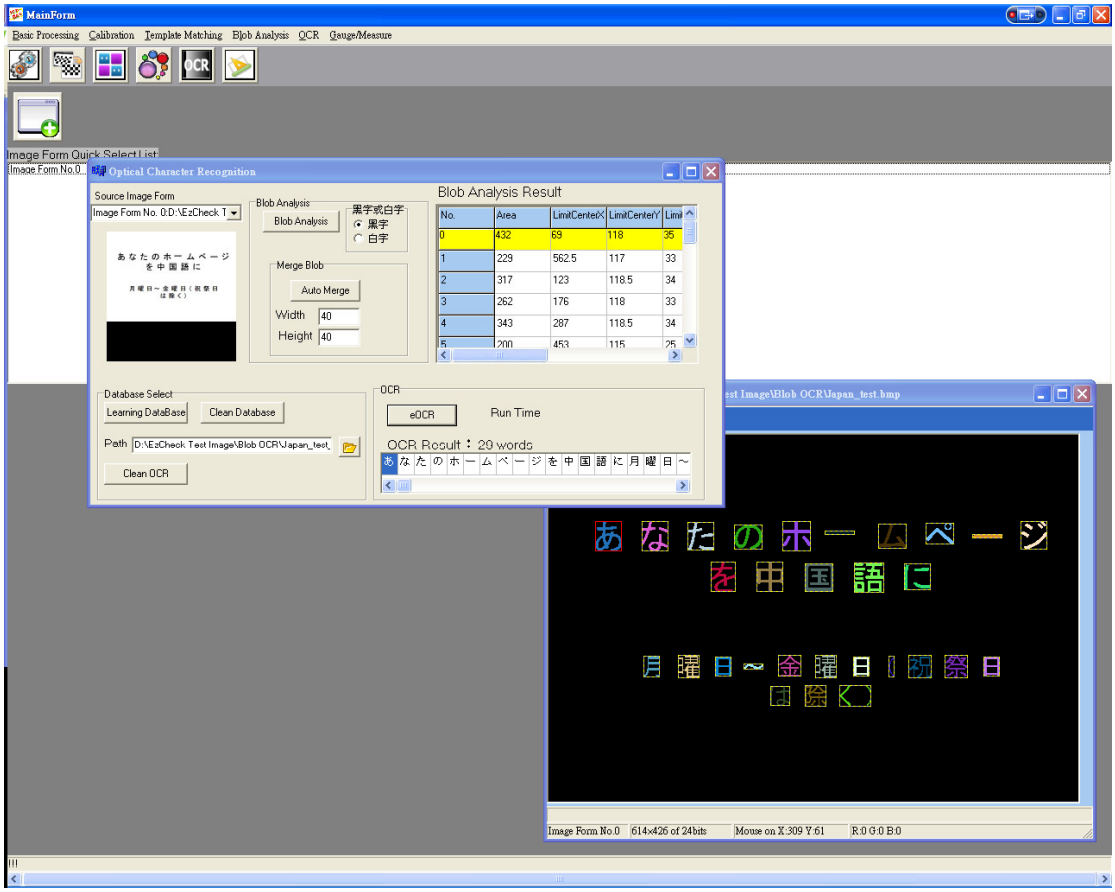
```
enum SELECTFEATURE
{
	SelectFeature  Area = 1.
	SelectFeature  LimitCenterX,
	SelectFeature  LimitCenterY,
	SelectFeature  LimitHeight,
	SelectFeature  LimitWidth.
	SelectFeature  GravityCenterX.
	SelectFeature  GravityCenterY.
	SelectFeature  CentroidCenterX,
	SelectFeature  CentroidCenterY,
	SelectFeature  SigmaX,
	SelectFeature  SigmaY.
	SelectFeature  SigmaXY.
	SelectFeature  SigmaXX,
	SelectFeature  SigmaYY.
	SelectFeature  PixelGravAverage,
	SelectFeature  PixelGravMin.
	SelectFeature  PixelGravMax.
	SelectFeature  SumPixelGrav.
	SelectFeature  SumPixelGravSquare,
	SelectFeature  EllipseHeight,
	SelectFeature  EllipseWidth.
	SelectFeature  EllipseAngle.
	SelectFeature  Convex_Hull,
	SelectFeature_All,
};
```

內容簡介：

各宣告之定義與 blob 基本特徵以及 blob 進階特徵相同。請參考 _BLOBBASICFEATURE_TAG **與** _BLOBADVANCEDFEATURE_TAG。

7. 光學文字辨識－eCOCR

eCOCR 提供文字辨識的功能。使用者可透過資料庫的建構，分析包括數字、中文(簡繁)、韓文、日文、英文...等多種語言的文字。



圖二十八、 以 eCOCR 對日文影像進行文字辨識

7.1. 主要特色

- **支援各種語言、文字與數字的辨識**

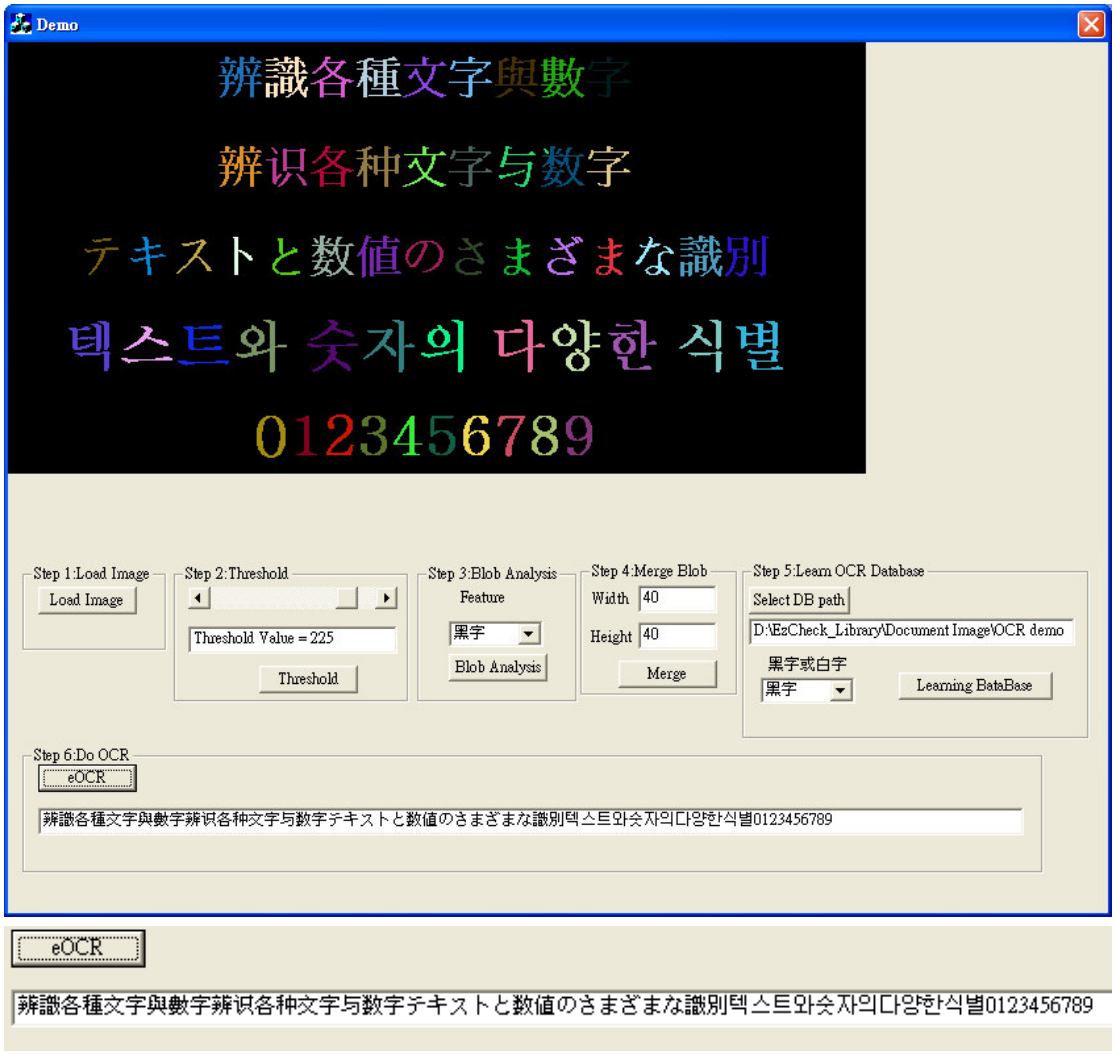
跨越各國不同型態與特性的文字的界限，無論是日文、英文、數字、簡體中文、繁體中文或甚至是韓文，只要作業系統支援顯示的語言，就可以透過 eCOCR 進行辨識。

- **資料夾管理辨識資料庫**

可利用資料夾將各種不同用途的文字資料庫進行分類，辨識時也可以透過只定資料夾的方式，增加辨識的準確度。

➢ **支援結果的輸出、排序**

辨識後，可透過內建函式取得辨識結果相關資訊，如座標或者文字。也可對結果進行排序或者存出為文字檔方便後續的驗證。



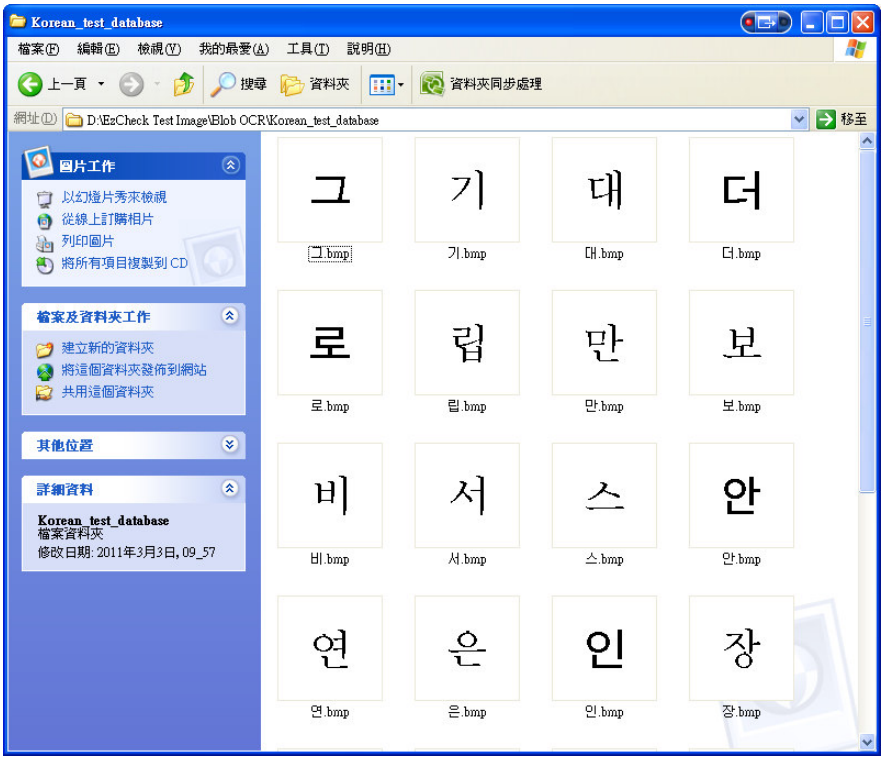
圖二十九、 eCBlob 可支援多國語言的文字辨識

7.2. eCOCR 主要功能介紹

eCOCR 以原始影像與 eCBlob 的結果為辨識依據。對影像進行二值化等影像前處理，並完成必要的篩選與合併功能的 eCBlob 後，eCOCR 將會依照 blob 的資訊以及另外建置的資料庫進行文字辨識。只要作業系統與編譯器支援顯示的文字、數字，都可透過 eCOCR 進行辨識。

資料庫的建置可利用 eCBlob 中 SaveSingleBlob 或者 SaveAllBlob 分別取

得各個 blob 的檔案，並且賦予各個檔案與文字內容相符的檔名。



圖三十、　韓文資料庫示意圖

7.2.1. eCOCR

建構式。

函式原形：

```
eCOCR ();
```

7.2.2. SetDataBaseCharacterType

設定要辨識的文字是白色或黑色。運算成功將回傳 TRUE，否則回傳 FALSE。

函式原形：

```
BOOL    SetDataBaseCharacterType (UINT8 Type);
```

傳入值：

Type **in** 辨識文字種類。請參考 CharacterType

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

7.2.3. DistanceTransformDataBase

讀取資料庫。運算成功將回傳 TRUE，否則回傳 FALSE。

函式原形：

Borland C++ Builder version:

```
BOOL   DistanceTransformDataBase(AnsiString DBFolder);
BOOL   DistanceTransformDataBase(WideString DBFolder);
```

Visual C++ version:

```
BOOL   DistanceTransformDataBase(CString DBFolder);
```

傳入值：

DBFolder **in** 資料庫路徑

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

7.2.4. DistanceTransformRecognition /

7.2.5. SelectDistanceTransformRecognition

eCOCR 核心功能，文字辨識。需要自來源影像與 eCBlob 結果取得相關資訊，也可對所有 blob 或者 select blob 進行辨識。

函式原形：

```
BOOL   DistanceTransformRecognition(eCImage * SrcImage,
                                    eCBlob Blob);
BOOL   SelectDistanceTransformRecognition(eCImage * SrcImage,
                                          eCBlob Blob);
```

傳入值：

SrcImage	**in**	來源影像。請輸入與 blob analysis 相同之影像。
Blob	**in**	提供辨識資訊的 eCBlob。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

7.2.6. GetSortResult

取得文字辨識的結果。取得的結果將是排序好的。

函式原形：

```
PPRecognition   GetSortResult();
```

回傳值：

PRecognition。紀錄字辨識結果的結構陣列。請參考 PRecognition。

7.2.7. GetRecognitionCount

取得文字辨識結果的數量。

函式原形：

```
UINT16   GetRecognitionCount();
```

回傳值：

該次文字辨識結果的數量。

Example code:

```
eCBlob BlobOCR;
eCOCR OCR;
UINT16 OcrCount = 0;//The number of ocr results.
WideString OcrResult;//A string to get ocr results. Use CString if using VC++
PRecognition *SortPR = NULL;
//After blob analysis finished.
OCR.SetDataBaseCharacterType(BlackCharacterWhiteBackground);
OCR.DistanceTransformDataBase("C:\\OCR_DB");
OCR.DistanceTransformRecognition(SrcImage, BlobOCR);
OcrCount = OCR.GetRecognitionCount();
```

```
if(SortPR != NULL)//Make sure that SorPR really have information.
{
    for (int i = 0; i < OcrCount; i++)//Survey all of the ocr results.
    {
        OcrResult += WideString(SortPR[i].str);//Get the string of ocr results.
    }
}
```

7.2.8. GetOCRTime

取得文字辨識運算的時間。

函式原形：

```
FLOAT64    GetOCRTime();
```

回傳值：

以毫秒(ms)為單位之運算時間。不包含進行排序的時間。

7.2.9. CleanBuffer

清空 eCOCR 暫存的運算結果。

函式原形：

```
BOOL    CleanBuffer();
```

回傳值：

運算成功將回傳 TRUE，否則回傳 FALSE。

7.2.10. CleanDataBaseBuffer

清空 eCOCR 讀取的資料庫資料。

函式原形：

```
BOOL    CleanDataBaseBuffer();
```

回傳值：

運算成功將回傳 TRUE，否則回傳 FALSE。

7.2.11. SaveTxt

將 OCR 結果輸出為文字檔。可選擇自訂路徑檔名，也可讓系統自行產生。

函式原形：

Borland C++ Builder version:

```
BOOL   SaveTxt():
BOOL   SaveTxt(AnsiString TxtName);
```

Visual C++ version:

```
BOOL   SaveTxt():
BOOL   SaveTxt(CString TxtName);
```

傳入值：

TxtName in 使用者自訂的路徑檔名。

回傳值：

運算成功將回傳 TRUE，否則回傳 FALSE。

7.2.12. GetErrorCode

取得 eCOCR 運作中回報的 error code。

函式原形：

```
INT32   GetErrorCode();
```

回傳值：

eCOCR 運作回傳的 error code。請參考 ErrorCode 列表。

7.3. eCOCR 資料結構

7.3.1. PPRecognition

記錄文字辨識結果的資料結構。結果已經過左上至右下的排序。

宣告內容：

```
typedef struct
{
    UINT16 x;
    UINT16 y;
    wchar_t *str;
}PRecognition, *PPRecognition;
```

內容簡介：

x, y	文字中心點座標。
str	文字辨識結果。

7.4. eCOCR 列舉型態

7.4.1. CharacterType

eCOCR 辨識文字的參考。

宣告內容：

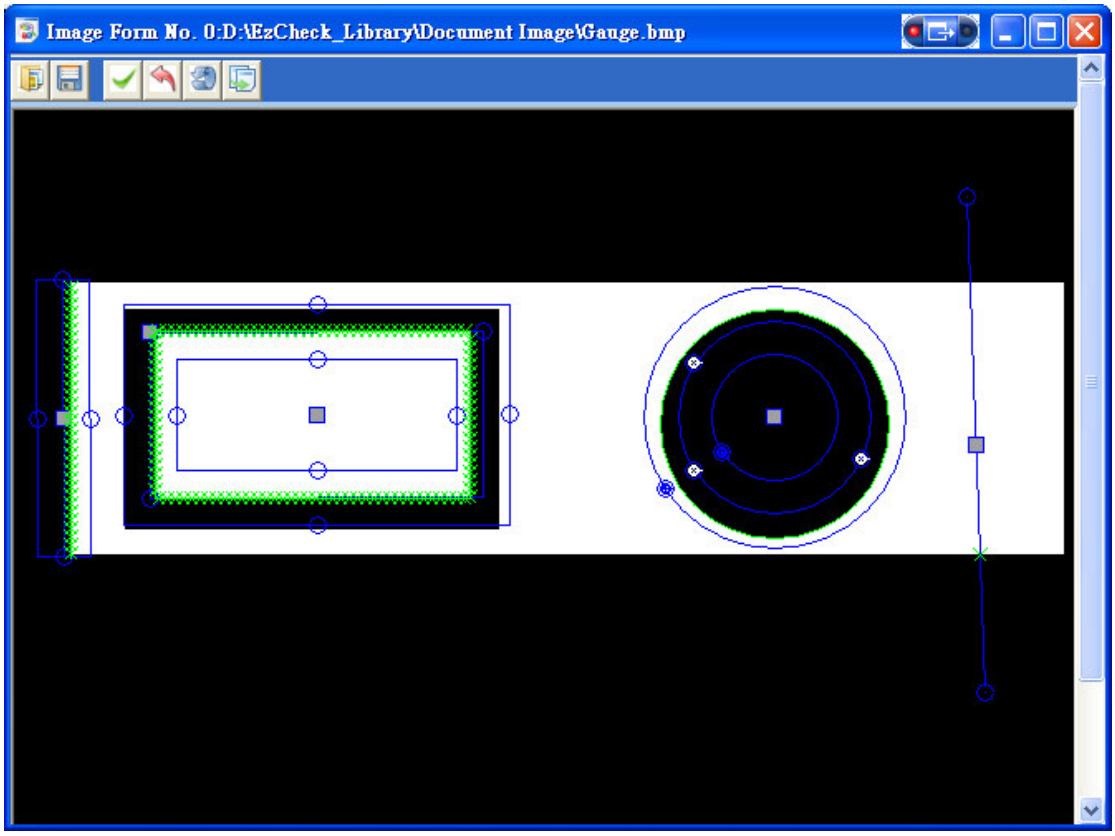
```
enum CharacterType
{
    BlackCharacterWhiteBackground = 1,   //黑字白底
    WhiteCharacterBlackBackground = 2    //白字黑底
};
```

內容簡介：

BlackCharacterWhiteBackground	黑底白字
WhiteCharacterBlackBackground	白底黑字

8. 測量-eGauge

eGauge 提供物件導向的測量工具，利用影像上的邊緣資訊，測量分析多種形狀與特徵。透過物件導向的設計，各個測量工具的檢測參數設定都可輕易完成。



圖三十一、　多個 eGauge 測量工具同時對影像進行測量

8.1.　主要特色

- **物件導向的工具設計**

  eGauge 中的各個工具均以物件導向的方式包裝而成，無須過多程式技巧。

- **適用於多種形狀的工具**

  eGauge 提供點、線、圓形、方型等四種測量工具以符合不同的測量應用需求。

- **提供各種不同的設定參數**

  各種不同特性的影像，均可透過 eGauge 參數的調整完成測量。無論工具的位置角度、邊界明暗方向與數值的定義，或是測量結果的選擇性輸出。

- **多個不同形狀的工具可同時存在**

  同一張影像可能會有多個不同的測量需求，eGauge 提供多個不同形狀、設定的測量工具同時對同一張影像進行測量。

- **提供內建的影像介面**

  eGauge 提供內建的影像顯示，使用者可從影像的輸出上觀察到各項設定所產生的影響。

- **提供開發介面的 API**

  搭配 eGauge 內建 API，可快速開發互動的影像介面。

- **支援次像素的結果輸出**

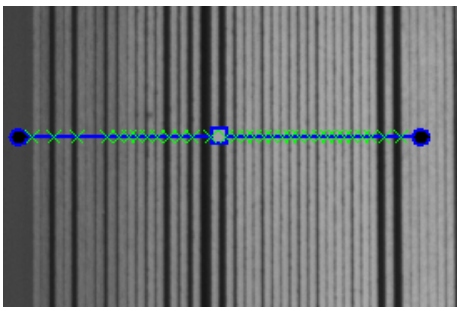
  測量結果精確度達次像素等級，幫助使用者進行更精確的測量。

8.2. 測量工具：

➢ ePointGauge：

測量一個線段區間有多少邊界相交的特徵點，取得各點的位置資訊。

- 特徵點點數
- 各特徵點座標



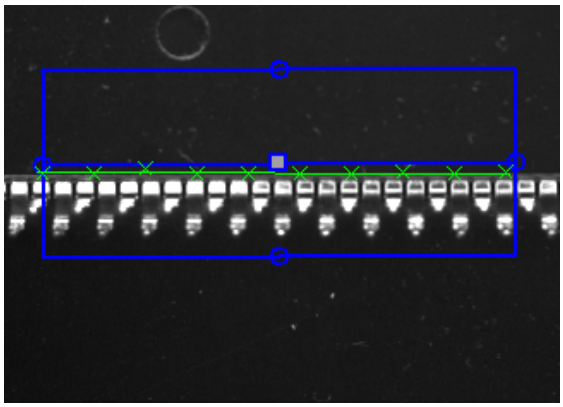
圖三十二、 ePointGauge 範例。ePointGauge 可計算影像上的邊界數量

➢ eLineGauge：

測量一個區間中包含的邊緣點所形成的直線，取得直線的位置與角度資訊。

測量結果：

- 線段中心點座標
- 直線角度

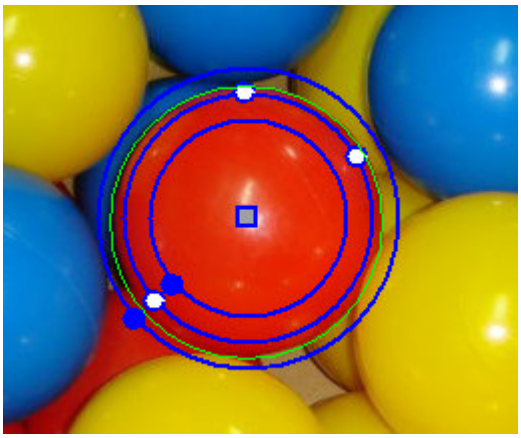


圖三十三、 eLineGauge 範例。eLineGauge 可取得影像上的直線資訊

➢ eCircleGauge：

測量一個區間中包含的圓形，取得圓形的圓心與直徑。

- 圓心座標
- 圓形直徑

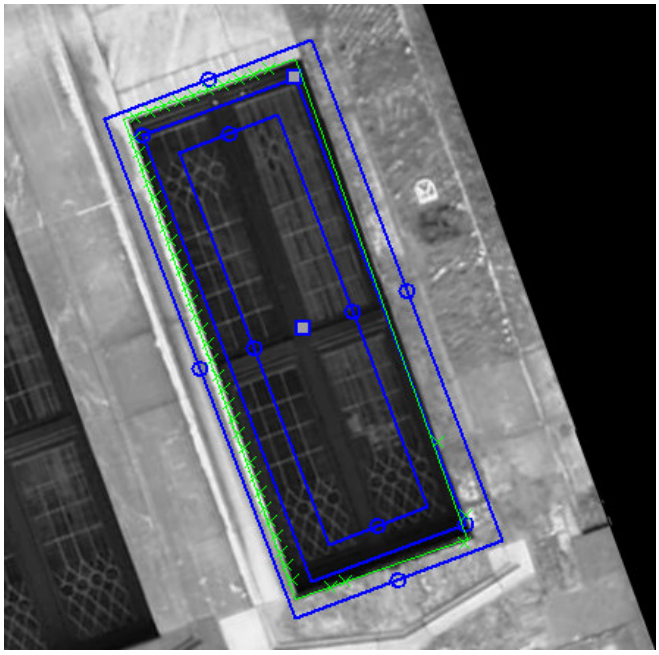


圖三十四、 eCircleGauge 範例。eCircleGauge 可取得影像上的圓形資訊

➢ eRectangleGauge：

測量一個區間中包含的方形，取得方形的中心點位置、長寬尺寸以及傾斜角度。

- 方形中心點座標
- 方形長寬尺寸
- 方形傾斜角度



圖三十五、 eRectangleGauge 範例。eRectangleGauge 可取得影像中方型窗戶的資訊

8.3. eTransition 類別

eTransition 類別是 ePointGauge, eLineGauge, eCircleGauge, eRectangleGauge 類別的基礎(父)類別，所有子類別皆繼承了 eTransistion 的函式與變數。eTransistion 的主要目的是讓使用者設定一些參數，在一個連續或明顯產生灰階變化的影像區塊中，找到使用者認定的最佳灰階轉折點，此轉折點被認定為物體的邊緣(edge) ，將所有偵測的邊緣點連接起來後，計算是否有 Point,Line,Circle ,Rectangle 等形狀。

後面章節將不再贅述每個 Gauge 中屬於 eTransistion 的函式與成員變數。

8.3.1. SetTolerance

1.ePointGauge: 設定 ePointGauge 物件的一半長度
2. eLineGauge: 設定 eLineGauge 物件的垂直寬度
3. eCircleGauge: 設定 eCircleGauge 物件的垂直寬度
4. eRectangleGauge: 設定 eRectangleGauge 的垂直寬度

函式原形：

```
Void   SetTolerance(FLOAT32 f32Tolerance,
                    FLOAT32 Angle);
```

傳入值：

f32Tolerance	**in**	長度/寬度
Angle	**in**	只有 ePointGauge 需要設定此參數

8.3.2. GetTolerance

1.ePointGauge:取得 ePointGauge 物件的一半長度

2. eLineGauge:取得 eLineGauge 物件的垂直寬度

3. eCircleGauge: 取得 eCircleGauge 物件的垂直寬度

4. eRectangleGauge: 取得 eRectangleGauge 的垂直寬度

函式原形：

```
FLOAT32   GetTolerance();
```

回傳值：

1.ePointGauge：ePointGauge 物件的一半長度

2. eLineGauge：eLineGauge 物件的垂直寬度

3. eCircleGauge：eCircleGauge 物件的垂直寬度

4. eRectangleGauge：eRectangleGauge 的垂直寬度

8.3.3. GetToleranceAngle

取得 ePointGauge 物件的旋轉角度。

函式原形：

```
FLOAT32   GetToleranceAngle();
```

回傳值：

ePointGauge 物件的旋轉角度。

8.3.4. SetThickness

加入更多的灰階資訊來輔助 edge 的偵測,適當的 Thickness 參數可以增強 edge 落點的準確性以及排除雜訊干擾,相反的,過大的 Thickness 參數可能會讓 edge 的落點失真

函式原形：

```
Void   SetThickness( UINT32 un32Thickness);
```

傳入值：

Un32Thickness in ePointGauge：設定平行於 ePointGauge 物件的灰階資訊參考範圍之寬度。
eLineGauge：設定平行於 eLineGauge 物件之取樣線的參考範圍之寬度。
eCircleGauge：設定 eCircleGauge 物件之取樣線的參考範圍之寬度。
eRectangleGauge：設定 eRectangleGauge 之取樣線的參考範圍之寬度。

8.3.5. GetThickness

加入更多的灰階資訊來輔助 edge 的偵測,適當的 Thickness 參數可以增強 edge 落點的準確性以及排除雜訊干擾,相反的,過大的 Thickness 參數可能會讓 edge 的落點失真

函式原形：

```
UINT32   GetThickness();
```

回傳值：

eGauge 元件的取樣線參考範圍寬度。請參考 SetThickness 之**傳入值**。

8.3.6. SetTransitionType

設定灰階的 Transition 型態

函式原形：

```
void   SetTransitionType(enum GGE_TRANSITION_TYPE
                                                    eTransitionType);
```

傳入值：

eTransitionType in 測量物件的 Transition 型態。請參考 enum GGE_TRANSITION_TYPE

8.3.7. GetTransitionType

取得灰階的 Transition 型態

函式原形：

```
enum GGE_TRANSITION_TYPE GetTransitionType();
```

回傳值：

測量物件的 Transition 型態。請參考 enum GGE_TRANSITION_TYPE。

8.3.8. SetTransitionChoice

設定灰階的 Choice 型態

函式原形：

```
void SetTransitionChoice(enum GGE_TRANSITION_CHOICE
                                                    eTransitionChoice);
```

傳入值：

eTransitionChoice in 測量物件決定 edge 的方式。請參考 enum GGE_TRANSITION_CHOICE

8.3.9. GetTransitionChoice

取得灰階的 Choice 型態

函式原形：

```
enum GGE_TRANSITION_CHOICE    GetTransitionChoice();
```

回傳值：

測量物件決定 edge 的方式。請參考 enum GGE_TRANSITION_CHOICE

8.3.10. SetTransitionIndex

當 enum GGE_TRANSITION_CHOICE 是 GGE_NTH_FROM_BEGIN 或者 GGE_NTH_FROM_END 模式時，設定從某方向数來的第 n 個結果點(邊緣點)

函式原形：

```
void    SetTransitionIndex(UINT32 un32TransitionIndex);
```

傳入值：

un32TransitionIndex   in   指定結果點，從零開始。

8.3.11. GetTransitionIndex

當 enum GGE_TRANSITION_CHOICE 是 GGE_NTH_FROM_BEGIN 或者 GGE_NTH_FROM_END 模式時，取得從某方向数來的第 n 個結果點(邊緣點)設定。

函式原形：

```
UINT32    GetTransitionIndex();
```

回傳值：

指定結果點，從零開始。

8.3.12. SetThreshold

設定 Threshold 來決定哪些灰階變化程度可視為 edge

函式原形：

```
Void   SetThreshold(UINT32 un32Threshold);
```

傳入值：

un32Threshold **in** Threshold 越小可得到越多 edge 點，Threshold 越大就得到越少 edge 點，視情況調整以讓量測到的 edge 具有可信度。

8.3.13. GetThreshold

取得 Threshold 來決定哪些灰階變化程度可視為 edge 的設定

函式原形：

```
UINT32   GetThreshold();
```

回傳值：

參考**傳入值**。

8.3.14. SetMinAmplitude

設定最小的灰度振幅

函式原形：

```
void   SetMinAmplitude( UINT32 un32MinAmplitude);
```

傳入值：

un32MinAmplitude **in** 指定最小的灰度振幅值。

8.3.15. GetMinAmplitude

取得最小的灰度振幅設定

函式原形：

```
UINT32   GetMinAmplitude();
```

回傳值：

最小的灰度振幅值。

8.3.16. SetMinArea

設定最小的灰度面積

函式原形：

```
void   SetMinArea(UINT32 un32MinArea);
```

傳入值：

un32MinArea in 指定最小的灰度面積值。

8.3.17. SetMinArea/GetMinArea

取得最小的灰度面積設定

函式原形：

```
UINT32   GetMinArea();
```

回傳值：

最小的灰度面積設定。

8.3.18. GetNumMeasuredPoints

量測後找到的 Point 結果數量，此函式專屬於 ePointGauge 使用

函式原形：

```
UINT32   GetNumMeasuredPoints();
```

回傳值：

量測後找到的 Point 結果數量。

8.3.19. GetMeasuredPoint

量測後找到的 Point 之座標資料。可直接用於 ePointGauge，eLineGauge、eCircleGauge、eRectangleGauge 類別的 GetMeasuredPoint ( )要搭配 MeasureSample(index)使用。

函式原形：

```
ePoint   GetMeasuredPoint(UINT32 un32Index);
ePoint   GetMeasuredPoint();
```

傳入值：

un32Index **in** 指定 Point 的序號(從 0 開始)，此參數專屬於 ePointGauge 使用。

回傳值：

ePoint 類別。紀錄點的相關資訊。請參考 eCGeametry.h 中的定義。例如：GetX 與 GetY 可以取得該點的座標。

Example Code:

```
CircleGauge.MeasureSample(3); // 指定 CircleGauge 上的第三條 sample path
If(CircleGauge.GetValid )//假使第三條 sample path 上有 edge point 存在
{
    Float x= CircleGauge.GetMeasuredPoint().GetX();
    Float y= CircleGauge.GetMeasuredPoint().GetY();
}
```

8.3.20. GetMeasuredPeak

取得 Gauge 的向量資料。

函式原形：

```
ePeak   GetMeasuredPeak(UINT32 un32Index);
```

傳入值：

un32Index **in** 指定 m_Peaks 的序號 (from 0 on)

回傳值：

ePeak 資料。請參考 m_Peaks。

8.3.21. GetValid

該 Gauge 元件是否有效。

函式原形：

```
BOOL   GetValid();
```

回傳值：

ePointGauge：若 ePointGauge 有測量到點，回傳 TRUE，否則回傳 FALSE。
eLineGauge：若 eLineGauge 有測量到點，回傳 TRUE，否則回傳 FALSE。
以最後一次進行 MeasureSample(index)取得的資訊為準。
eCircleGauge：同 eLineGauge。
eRectangleGauge：同 eLineGauge。

8.3.22. SetTransitionRectangularSamplingArea

設定 Gauge 物件的外觀為直角矩形或平行四邊形

函式原形：

```
void   SetTransitionRectangularSamplingArea
                                    (BOOL bRectangularSamplingArea);
```

傳入值：

bRectangularSamplingArea	**in**	TRUE 表示直角矩形(rectangle)，FALSE 表示平行四邊形(parallelogram)

8.3.23. GetTransitionRectangularSamplingArea

取得 Gauge 物件的外觀為直角矩形或平行四邊形設定

函式原形：

```
BOOL   GetTransitionRectangularSamplingArea ( )
```

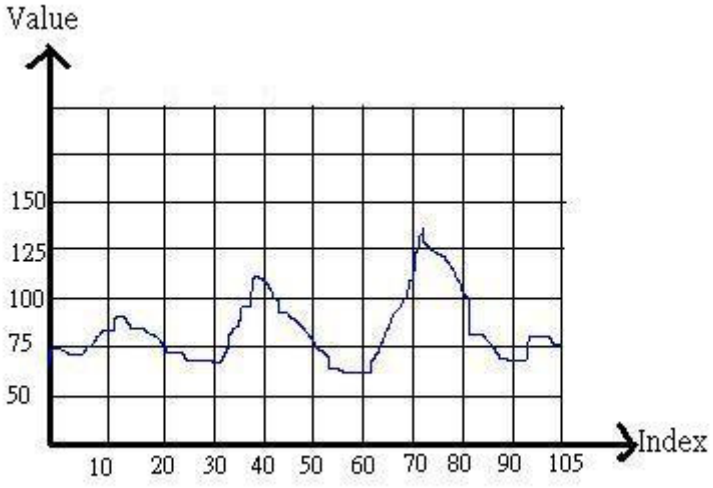
回傳值：

Gauge 物件的外觀為直角矩形或平行四邊形的設定。

8.3.24. m_Profile

eTransition 類別中的 **eProfileVector** 型態成員變數。一張影像中，關於連續路徑的灰階值以一維陣列 **eProfileVector** 紀錄之。此資料陣列可用一連續曲線剖面圖來圖示化，如下圖

1. Index: 像素位於連續路徑上的序號
2. Value: 像素的灰階值



圖三十六、 Vector 示意圖

8.3.25. m_Derivative

eTransition 類別中的 **eDerivativeVector** 型態成員變數。計算連續路徑上的灰階值之**一階導數**值，以一維陣列 **eDerivativeVector** 紀錄之。此資料陣列可用一連續曲線剖面圖來圖示化，如上圖所示。

1. Index: 像素位於連續路徑上的序號
2. Value: 像素的一階導數值

8.3.26. m_Peaks

eTransition 中的 **ePeaksVector** 型態成員變數。Peak 的定義為：**一段連續高於(或低於)Threshold 的一階導數值**。以一維陣列 **ePeaksVector** 紀錄之。

8.4. eTransition 列舉型態與資料結構

8.4.1. enum GGE_TRANSITION_TYPE

灰階的 Transition 型態

宣告內容：

```
enum GGE_TRANSITION_TYPE
{
  GGE_BW.
  GGE_WB.
  GGE_BW_OR_WB.
  GGE_BWB.
  GGE_WBW.
  GGE_UNKNOWN_TRANSITION_TYPE   = 0xFFFFFFFF
};
```

內容簡介：

GGE_BW	灰階的變化是由黑到白
GGE_WB	灰階的變化是由白到黑
GGE_BW_OR_WB	灰階的變化是由黑到白，或白到黑
GGE_BWB	灰階的變化是由黑到白到黑
GGE_WBW	灰階的變化是由白到黑到白
GGE_UNKNOWN_TRANSITION_TYPE	= 0xFFFFFFFF

8.4.2. enum GGE_TRANSITION_CHOICE

決定回傳為邊緣(edge)的方式與選擇。

宣告內容：

```
enum GGE_TRANSITION_CHOIC
{
  GGE_NTH_FROM_BEGIN,
  GGE_NTH_FROM_END.
  GGE_LARGEST_AMPLITUDE,
  GGE_LARGEST_AREA,
  GGE_CLOSEST,
  GGE_ALL.
  GGE_UNKNOWN_TRANSITION_CHOICE   = 0Xffffffff
};
```

內容簡介：

GGE_NTH_FROM_BEGIN	檢查方向從 Gauge 的結束控制點(end handle)開始，只回傳此方向上的第一個 egde 點
GGE_NTH_FROM_END	只回傳取像路徑上最大 amplitude 的 edge 點
GGE_ LARGEST_AMPLITUDE	只回傳取像路徑上最大 area 的 edge 點
GGE_ LARGEST_AREA	只回傳路徑上最靠近中心控制點(center handle)的 edge 點
GGE_ CLOSEST	回傳全部的 edge 點
GGE_ ALL	檢查方向從 Gauge 的結束控制點(end handle)開始，只回傳此方向上的第一個 egde 點
GGE_ UNKNOWN_TRANSITION_CHOICE	= 0xFFFFFFFF

8.4.3. struct ePeak

記錄數值起伏的資料結構

宣告內容：

```
struct ePeak
{
   UINT32 m  un32Start. m  un32Lenath;
   FLOAT32 m  f32Center:
   INT32 m_n32Amplitude, m_n32Area;
};
```

內容簡介：

m_un32Start	Peak 的起始位置
m_un32Length	Peak 的長度
m_f32Center	Peak 的中心位置
m_n32Amplitude	Peak 的振幅
m_n32Area	Peak 的面積

8.5. eGauge

eGauge 包含 ePointGauge、eLineGauge、eCircleGauge 與 eRectangleGauge 四個類別，其多數基本函式的運作、使用方法與原理都類似。

8.5.1. ePointGauge

ePointGauge 建構式

函式原形：

```
ePointGauge   (FLOAT32 f32CenterX.
               FLOAT32 f32CenterY) :
ePointGauge   (const ePointGauge& otherInstance):
ePointGauge&   operator= (const ePointGauge& otherInstance)
```

傳入值：

f32CenterX, **in** Gauge 的中心點座標

f32CenterY

otherInstance **in** 另一個 ePointGauge 物件

8.5.2. eLineGauge

eLineGauge 建構式

函式原形：

```
eLineGauge    (const eLineGauge& otherInstance):
eLineGauge&   operator= (const eLineGauge& otherInstance)
```

傳入值：

otherInstance **in** 另一個 eLineGauge 物件

8.5.3. eCircleGauge

eCircleGauge 建構式

函式原形：

```
eCircleGauge    (const eCircleGauge& otherInstance):
eCircleGauge&   operator= (const eCircleGauge& otherInstance)
```

傳入值：

otherInstance **in** 另一個 eCircleGauge 物件

8.5.4. eRectangleGauge

eRectangleGauge 建構式

函式原形：

```
eRectangleGauge    () :
eRectangleGauge    (const eRectangleGauge& otherInstance);
eRectangleGauge&   operator= (const eRectangleGauge&
                                                    otherInstance);
```

傳入值：

otherInstance **in** 另一個 eRectangleGauge 物件

8.5.5. SetCenter

指定 Gauge 的中心點座標

函式原形：

```
Void   SetCenter(ePoint Point)
Void   SetCenter(FLOAT32 f32CenterX.
                 FLOAT32 f32CenterY);
```

傳入值：

f32CenterX, **in** Gauge 的中心點座標

f32CenterY

Point **in** 點資訊類別。可用 SetX、SetY 設定座標。

8.5.6. Rescale

縮小或放大 Gauge 的尺寸

函式原形：

```
void   Rescale(FLOAT32 f32Factor);
```

傳入值：

f32Factor **in** 百分比例

8.5.7. SetActive

指定此 Gauge 可否被操作

函式原形：

```
void   SetActive(BOOL bActive);
```

傳入值：

bActive　　**in**　TRUE: 可被操作 ， FALSE: 不可操作(變更)狀態

8.5.8. SetZoom

指定此 Gauge 的運作與顯示比例。目前僅 ePointGauge 支援此功能。

函式原形：

```
void   SetZoom(FLOAT32 f32ZoomX,
               FLOAT32 f32ZoomY);
```

傳入值：

f32ZoomX　　**in**　X 方向的縮放比例

f32ZoomY　　**in**　Y 方向的縮放比例

8.5.9. SetSelected

指定此 Gauge 目前是否被選擇。被選擇的 Gauge 元件將會以較明顯的方式繪製出來。

函式原形：

```
void   SetSelected(BOOL bSelected);
```

傳入值：

bSelected　　**in**　元件是否被選擇

8.5.10. Measure

ePointGauge 核心功能，執行 Point 量測。

函式原形：

```
void    Measure(eCImage* pSrc);
```

傳入值：

pSrc　**in**　被測量的來源影像。

8.5.11. GetMeasuredPoint

回傳指定序號的 Point。對 ePointGauge 而言可取得最終結果，對其他 eGauge 元件而言則能取得測量過程中賴以判斷形狀的點。

函式原形：

```
ePoint    GetMeasuredPoint(UINT32 un32Index);
```

傳入值：

un32Index　**in**　所有被檢測出的 Points 會依照 Gauge 檢測方向排序，因此使用者要以序號去得到他所要的 Point 座標

回傳值：

ePoint，請參考**傳入值**。

Example Code:

```
ePointGauge* PointGauge = new ePointGauge();//創建一個新的 ePoingGauge 元件
int Point_Number = 0;
PointGauge->SetCenter(100, 100);//將 PointGauge 元件放在(100, 100)
PointGauge->Measure(SrcImage);//對 SrcImage 做測量
Point_Number = PointGauge->GetNumMeasuredPoints();//取得測量到的點數
PointGauge->GetMeasuredPoint(0).GetX();//取得第 0 個點的 X 座標
PointGauge->GetMeasuredPoint(0).GetY();//取得第 0 個點的 Y 座標
```

8.5.12. GetMeasuredLine

回傳 eLineGauge 測量到的線段資訊。

函式原形：

```
eLine   GetMeasuredPoint();
```

回傳值：

eLine。紀錄線段資訊的類別。可利用 GetCenterX, GetCenterY 與 GetAngle 等函式取得線段資訊。

Example Code:

```
eLineGauge* LineGauge = new eLineGauge();//創建一個新的 LineGauge 元件
LineGauge->Measure(SrcImage);//對 SrcImage 做測量
LineGauge ->GetMeasuredLine().GetCenterX();//取線段中心點 X 座標
LineGauge -> GetMeasuredLine().GetCenterY();//取線段中心點 Y 座標
LineGauge->GetMeasuredLine().GetAngle()/(PI/180);//取得線段角度，PI 為圓周率
```

8.5.13. GetMeasuredCircle

回傳 eLineGauge 測量到的圓形資訊。

函式原形：

```
eCircle   GetMeasuredCircle();
```

回傳值：

eCircle。紀錄圓形資訊的類別。可利用 GetCenterX, GetCenterY, GetDiameter 等功能取得圓形的資訊。

Example Code:

```
eCircleGauge * CircleGauge = new eCircleGauge;//創建一個新的
eCircleGauge 元件
CircleGauge ->Measure(SrcImage);//對 SrcImage 做測量
CircleGauge->GetMeasuredCircle().GetCenterX();//取得圓形 X 座標
CircleGauge->GetMeasuredCircle().GetCenterY();//取得圓形 X 座標
CircleGauge->GetMeasuredCircle().GetDiameter();//取得圓形直徑
```

8.5.14. GetMeasuredRectangle

回傳 eRectangleGauge 測量到的矩形資訊。

函式原形：

```
eRectangle    GetMeasuredRectangle ();
```

回傳值：

eRectangle。紀錄矩形資訊的類別。可用 GetCenter, GetCenterY, GetSizeX 等功能取得矩形的資訊。

Example Code:

```
eRectangleGauge * RectangleGauge = new eRectangleGauge;//創建一個新的 eRectangleGauge 元件
CircleGauge ->Measure(SrcImage);//對 SrcImage 做測量
RectangleGauge->GetMeasuredRectangle().GetCenterX();//取得矩形的中心 X 座標
RectangleGauge->GetMeasuredRectangle().GetCenterY();//取得矩形的中心 Y 座標
RectangleGauge->GetMeasuredRectangle().GetSizeX();//取得矩形的寬
RectangleGauge->GetMeasuredRectangle().GetSizeY();//取得矩形的高
RectangleGauge->GetMeasuredRectangle().GetAngle()/(PI/180));//取得矩形傾斜角度
```

8.5.15. GetType

回傳此 Gauge 物件的 Shape 類型

函式原形：

```
enum INS_SHAPE_TYPES    GetType();
```

回傳值：

enum INS_SHAPE_TYPES。請參考 enum INS_SHAPE_TYPES。

8.5.16. Draw

畫出 Gauge 的樣貌

函式原形：

```
void   Draw( HDC hDC.
             enum INS_DRAWING_MODES eDrawingMode);
```

傳入值：

hDC	**in**	輸出標的的索引值
eDrawingMode	**in**	Gauge 物件的繪圖設定。 請參考 enum INS_DRAWING_MODES

8.5.17. HitTest

詢問滑鼠是否擊落在 eGauge 的 Handle 範圍中

函式原形：

```
BOOL   HitTest ();
```

回傳值：

若滑鼠擊落在 eGauge 的 Handle 範圍中，回傳 TRUE，否則回傳 FALSE。

8.5.18. Drag

用滑鼠游標將 eGauge 元件拖曳到新的座標上

函式原形：

```
void   Drag ( INT32 n32CursorX.
              INT32 n32CursorY);
```

傳入值：

n32CursorX,	**in**	指定的新座標。
n32CursorY		

8.6. eGauge 列舉型態

8.6.1. enum INS_SHAPE_TYPES

用來描述形狀的列舉型態

宣告內容：

```
enum INS_SHAPE_TYPES
{
      INS_NO_SHAPE             = 1 << 0,
      // Base shapes
      INS_POINT_SHAPE          = 1 << 2,
      INS_LINE_SHAPE           = 1 << 3,
      INS_CIRCLE_SHAPE         = 1 << 4,
      INS_WEDGE_SHAPE          = 1 << 5,
      INS_RECTANGLE_SHAPE = 1 << 6,
      INS_FRAME_SHAPE          = 1 << 7,
      INS_WORLD_SHAPE          = 1 << 8,
      INS_ANY_SHAPE            = INS_POINT_SHAPE | INS_LINE_SHAPE |
                                 INS_CIRCLE_SHAPE | INS_WEDGE_SHAPE |
                                 INS_RECTANGLE_SHAPE |
                                 INS_FRAME_SHAPE | INS_WORLD_SHAPE,
     // Gauging probes
      INS_POINT_GAUGE          = 1 << 9,
      INS_LINE_GAUGE           = 1 << 10,
      INS_CIRCLE_GAUGE         = 1 << 11,
      INS_RECTANGLE_GAUGE = 1 << 12,
      INS_WEDGE_GAUGE          = 1 << 17,
      INS_ANY_GAUGE            = INS_POINT_GAUGE | INS_LINE_GAUGE |
                                 INS_CIRCLE_GAUGE | INS_WEDGE_GAUGE|
                                 INS_RECTANGLE_GAUGE,
      // Any
      INS_ANY_TYPE             = ~1,
      INS_SHAPE_UNKNOWN        = 0xFFFFFFFF
};
```

8.6.2. enum INS_DRAGGING_MODES

當滑鼠按住 Gauge 物件的 Handle 進行拖曳時，Gauge 物件改變的模式。
例如:

1. eLineGauge: 在 INS_DRAG_STANDARD 模式下，拉動 INS_HANDLE_ORG 時,INS_HANDLE_END 的座標也同步改變,改變方向與 INS_HANDLE_ORG 相反。
   而在 INS_DRAG _TO_EDGES 模式下,單邊拉動 INS_HANDLE_ORG 時，INS_HANDLE_END 不會跟著改變。
2. eCircleGauge: 在 INS_DRAG_STANDARD 模式下,控制點 INS_HANDLE_ORG/MID/END 中僅剩 INS_HANDLE_ORG 可以操作。當拉動 INS_HANDLE_ORG 時，整個 Circle 以圓心為依據而放大縮小 Circle 尺寸。
   在 INS_DRAG _TO_EDGES 模式下，可以單邊拉動 INS_HANDLE_ORG/MID/END，整個 Circle 依據這三個 Hanle 形成一個圓。

宣告內容：

```
enum INS_DRAGGING_MODES
{
    INS_DRAG_STANDARD,
    INS_DRAG_TO_EDGES,
    INS_DRAG_UNKNOWN
};
```

內容簡介：

INS_DRAG_STANDARD	Gauge 的大小與形狀,呈現同步變動的方式
INS_DRAG_TO_EDGES	Gauge 的大小與形狀,呈現單邊變動的方式
INS_DRAG_UNKNOWN	0xFFFFFFFF

8.6.3. enum INS_HANDLES

使用者透過圖形介面操作 Gauge 物件的控制點(Handle)

宣告內容：

```
enum INS_HANDLES
{
   INS_HANDLE_NONE,
   INS_HANDLE_CENTER,
   INS_HANDLE_X_AXIS,
   INS_HANDLE_Y_AXIS,
   INS_HANDLE_ORG,
   INS_HANDLE_MID,
   INS_HANDLE_END,
   INS_HANDLE_INNER_ORG,
   INS_HANDLE_INNER_MID,
   INS_HANDLE_INNER_END,

   INS_HANDLE_TOL_0,
   INS_HANDLE_TOL_1,

   INS_HANDLE_TOL_x0,
   INS_HANDLE_TOL_x1,
   INS_HANDLE_TOL_y0,
   INS_HANDLE_TOL_y1,
   INS_HANDLE_TOL_X0,
   INS_HANDLE_TOL_X1,
   INS_HANDLE_TOL_Y0,
   INS_HANDLE_TOL_Y1,

   INS_HANDLE_OUTER_ORG        = INS_HANDLE_ORG,
   INS_HANDLE_OUTER_MID       = INS_HANDLE_MID,
   INS_HANDLE_OUTER_END       = INS_HANDLE_END,

   INS_HANDLE_TOL_a0          = INS_HANDLE_TOL_x0,
   INS_HANDLE_TOL_a1          = INS_HANDLE_TOL_x1,
   INS_HANDLE_TOL_A0          = INS_HANDLE_TOL_y0,
   INS_HANDLE_TOL_A1          = INS_HANDLE_TOL_y1,
   INS_HANDLE_TOL_r0          = INS_HANDLE_TOL_X0,
   INS_HANDLE_TOL_r1          = INS_HANDLE_TOL_X1,
   INS_HANDLE_TOL_R0          = INS_HANDLE_TOL_Y0,
   INS_HANDLE_TOL_R1          = INS_HANDLE_TOL_Y1,

   INS_HANDLE_EDGE_x          = 0x100,
```

```
INS_HANDLE_EDGE_X           = 0x200,
INS_HANDLE_EDGE_y           = 0x400,
INS_HANDLE_EDGE_Y           = 0x800,

INS_HANDLE_CORNER_xy        = 0x1000,
INS_HANDLE_CORNER_Xy        = 0x2000,
INS_HANDLE_CORNER_xY        = 0x4000,
INS_HANDLE_CORNER_XY        = 0x8000,

INS_HANDLE_EDGE_a           = INS_HANDLE_EDGE_x,
INS_HANDLE_EDGE_A           = INS_HANDLE_EDGE_X,
INS_HANDLE_EDGE_r           = INS_HANDLE_EDGE_y,
INS_HANDLE_EDGE_R           = INS_HANDLE_EDGE_Y,

INS_HANDLE_CORNER_ar        = INS_HANDLE_CORNER_xy,
INS_HANDLE_CORNER_Ar        = INS_HANDLE_CORNER_Xy,
INS_HANDLE_CORNER_aR        = INS_HANDLE_CORNER_xY,
INS_HANDLE_CORNER_AR        = INS_HANDLE_CORNER_XY,

INS_EDGE_x                  = INS_HANDLE_EDGE_x,
INS_EDGE_X                  = INS_HANDLE_EDGE_X,
INS_EDGE_y                  = INS_HANDLE_EDGE_y,
INS_EDGE_Y                  = INS_HANDLE_EDGE_Y,

INS_CORNER_xy               = INS_HANDLE_CORNER_xy,
INS_CORNER_Xy               = INS_HANDLE_CORNER_Xy,
INS_CORNER_xY               = INS_HANDLE_CORNER_xY,
INS_CORNER_XY               = INS_HANDLE_CORNER_XY,

INS_ALL_RECTANGLE_EDGES     = INS_EDGE_x | INS_EDGE_X |
                              INS_EDGE_y | INS_EDGE_Y,
INS_ALL_RECTANGLE_CORNERS = INS_CORNER_xy |
                              INS_CORNER_Xy |
                              INS_CORNER_xY |
                              INS_CORNER_XY |,

INS_EDGE_a                  = INS_HANDLE_EDGE_x,
INS_EDGE_A                  = INS_HANDLE_EDGE_X,
INS_EDGE_r                  = INS_HANDLE_EDGE_y,
INS_EDGE_R                  = INS_HANDLE_EDGE_Y,

INS_CORNER_ar               = INS_HANDLE_CORNER_xy,
INS_CORNER_Ar               = INS_HANDLE_CORNER_Xy,
INS_CORNER_aR               = INS_HANDLE_CORNER_xY,
INS_CORNER_AR               = INS_HANDLE_CORNER_XY,

INS_ALL_WEDGE_EDGES         = INS_EDGE_a | INS_EDGE_A |
                              INS_EDGE_r | INS_EDGE_R,
```

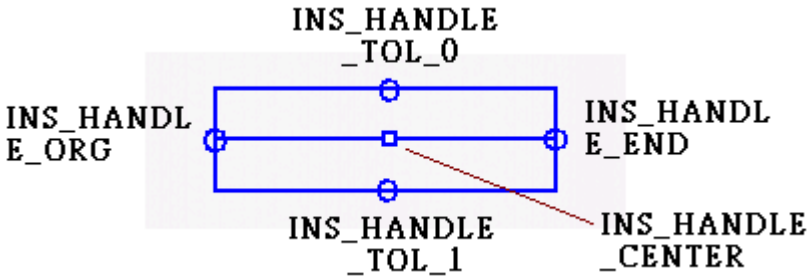
```
  INS_ALL_WEDGE_CORNERS       = INS_CORNER_ar |
                                INS_CORNER_Ar |
                                INS_CORNER_aR |
                                INS_CORNER_AR,

  INS_HANDLE_UNKNOWN          = 0xFFFFFFFF
};
```

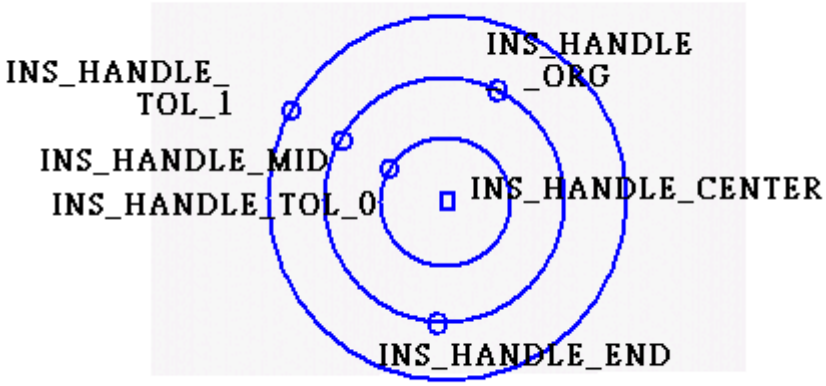
內容簡介：

請參考 ePointGauge Handle 示意圖、eLineGauge Handle 示意圖、eCircleGauge Handle 示意圖與 eRectangleGauge Handle 示意圖。

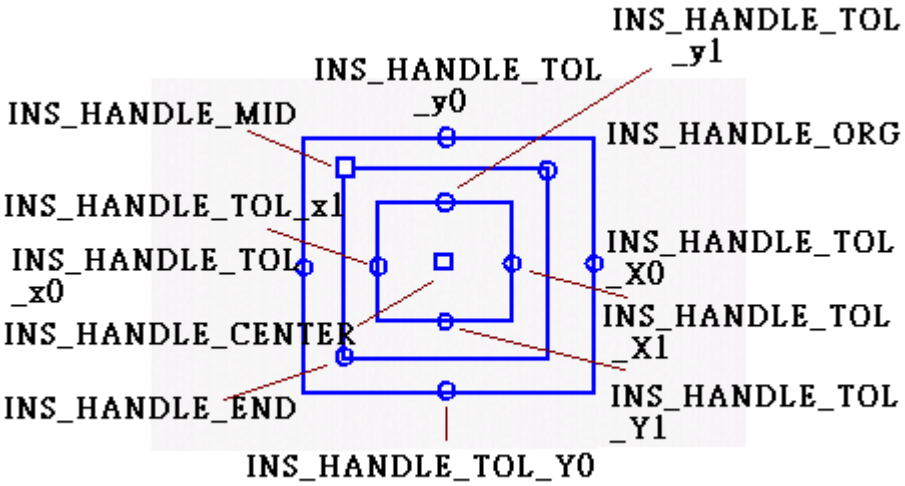
圖三十七、 ePointGauge Handle 示意圖



圖三十八、 eLineGauge Handle 示意圖



圖三十九、 eCircleGauge Handle 示意圖



圖四十、 eRectangleGauge Handle 示意圖

8.6.4. enum INS_DRAWING_MODES

Gauge 物件的繪圖設定

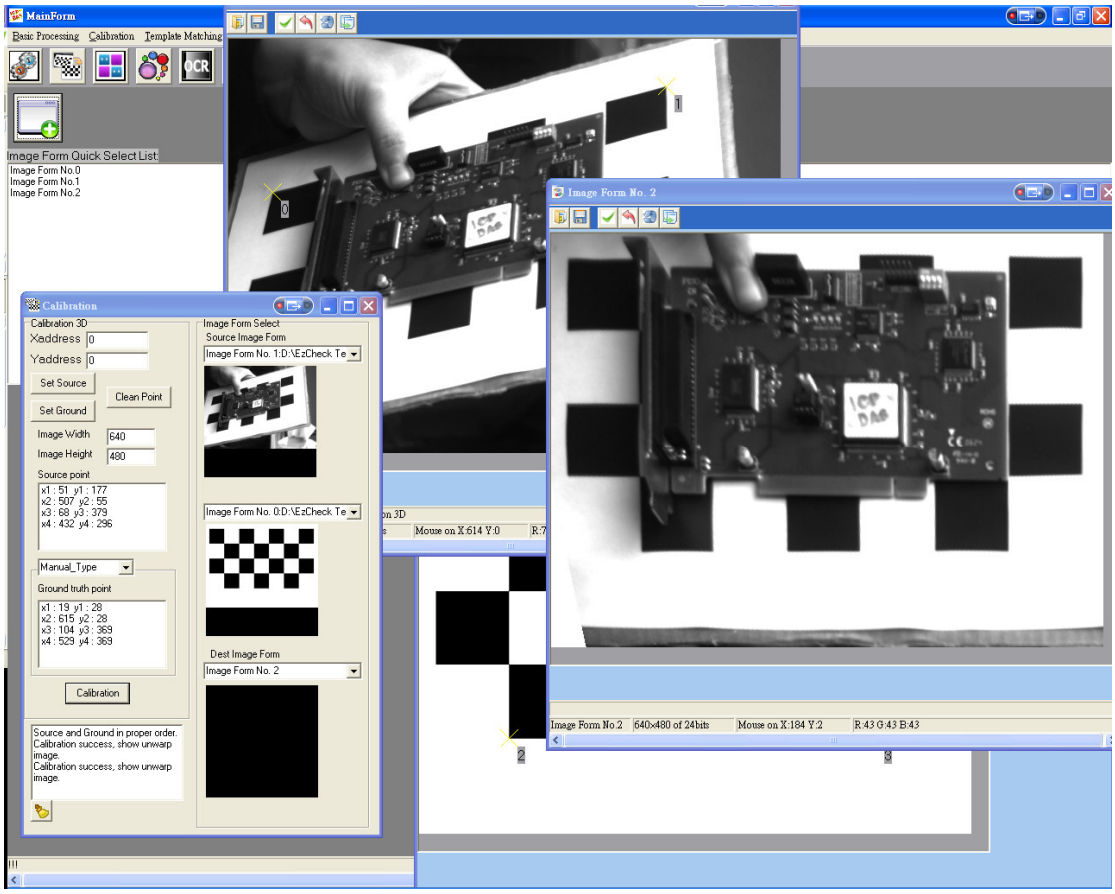
宣告內容：

```
enum INS_DRAWING_MODES
{
   INS_DRAW_NOMINAL = 0x01,   // 畫出 Gauge 本身以及 Gauge 的控制點
   INS_DRAW_ACTUAL, //畫出 Gauge 量測結果(edge 或 Line、Circle、
                              Rectangle)
   INS_DRAW_TOLERANCE,        //畫出 Tolerance 範圍
   INS_DRAW_SAMPLED_PATHS,   //畫出所有取樣路徑
   INS_DRAW_SAMPLED_PATH,    // 畫出指定的取樣路徑
   INS_DRAW_POINTS_IN_SKIP_RANGE,
   INS_DRAW_SAMPLED_POINTS, //畫出所有 edge point
   INS_DRAW_SAMPLED_POINT,   //畫出指定的 edge point
   INS_DRAW_INVALID_SAMPLED_POINTS,
   INS_DRAWING_MODE_UNKNOWN   = 0xFFFFFFFF
};
```

9.eCCalib3D

eCCalib3D 提供將傾斜扭曲的影像進行校正的功能，將因為拍攝角度傾斜而產生扭曲的影像，校正為正面拍攝的影像，幫助後續的測量或是辨識運算更有意義。

9.1. eCCalib3D 主要特色

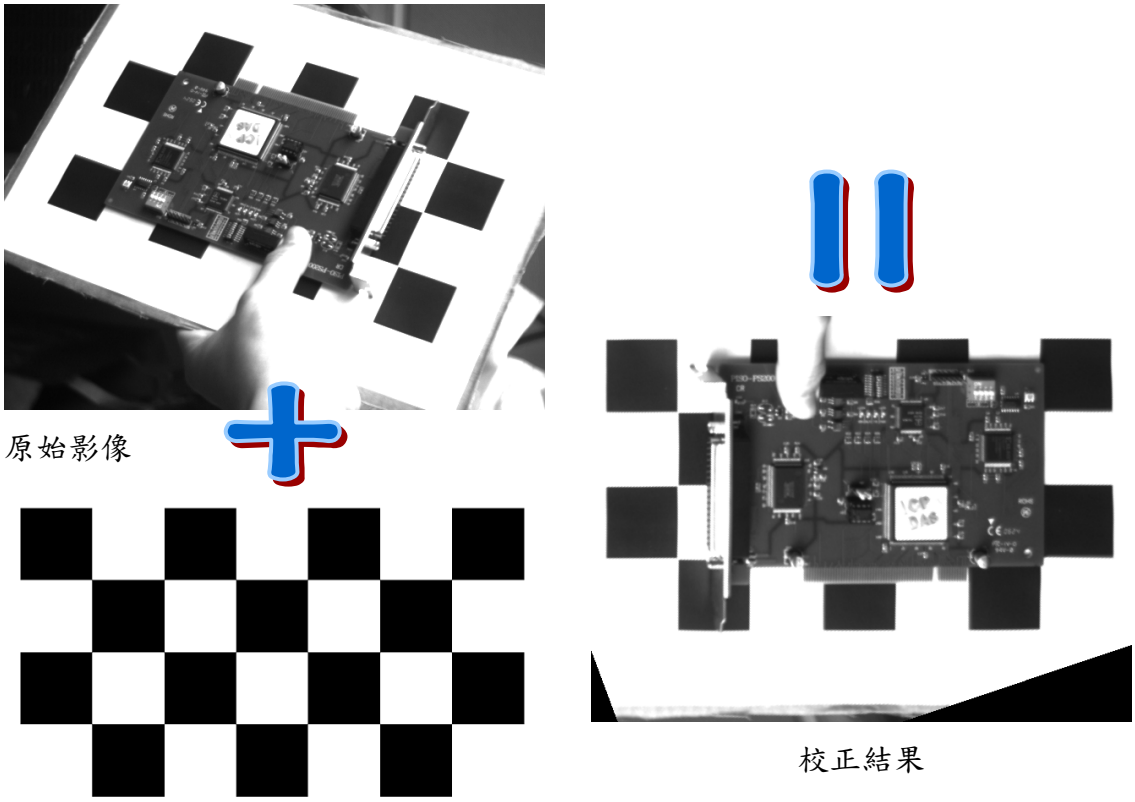


- **不受角度與維度限制的影像校正**

  只要分別輸出來源影像以及樣版影像的**四組對應特徵點**，各種傾斜、旋轉、扭曲的影像都可完成校正。

- **支援半自動流程**

  可對特徵點座標進行自動排序，並依照指定參數進行較正。



圖四十一、 eCCalib3D 運作示意圖

9.2. eCCalib3D 主要功能

eCCalib3D 以四組對應點計算影像的歪斜扭曲，並且依此進行校正。使用時應輸入在來源影像(需要被校正的影像)，以及校正樣版(希望校正後的狀況，或稱為 ground truth)，上的各四個校正參考點。目前來源影像上的校正參考點需要以手動的方式輸入，而樣版上的參考點則能以手動、半手動以及自動的方式取得。

9.2.1. eCCalib3D

建構式

函式原形：

```
eCCalib3D       ();
```

9.2.2. ~eCCalib3D

解構式。eCCalib3D 類別內含暫存影像與矩陣，不需使用時將其釋放以避免記憶體堆疊。

函式原形：

```
~eCCalib3D ();
```

9.2.3. SrcPT

eCCalib3D 類別中之開放成員，記錄四個來源影像上的校正參考點座標。使用者可直接存取之。

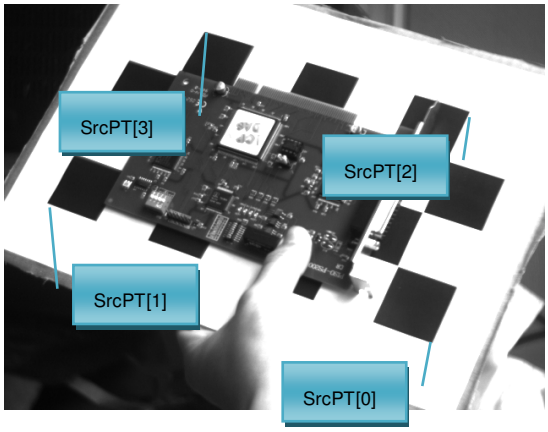
```
eCPoint   SrcPT[4];
struct eCPoint
{
    int x;
    int y;
};
```

9.2.4. GroundPT

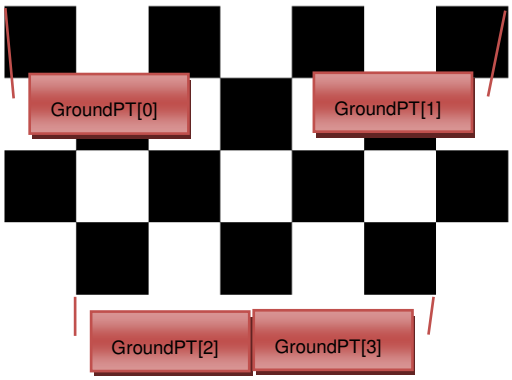
eCCalib3D 類別中之開放成員，記錄四個校正樣版上的校正參考點座標。使用者可直接存取之。

```
eCPoint   GroundPT [4];
```

Tips:SrcPT 與 GroundPT 中的點應該在影像上依序具有對應關係，以確保後續校正結果的正確性。
Example：



來源影像



校正樣版

9.2.5. SeFourPoint

設定校正的特徵點。可選擇自動設點以及手動設點。

函式原形：

```
BOOL   SetFourPoint(UINT16 X  axis.
                    UINT16 Y  axis.
                    UINT8 WhichSet);
BOOL   SetFourPoint (UINT16 X  axis.
                    UINT16 Y  axis.
                    UINT8 WhichSet .
                    UINT8 PointNumber);
```

傳入值：

X_axis	**in**	校正特徵點 x 座標
Y_axis	**in**	校正特徵點 y 座標
WhichSet	**in**	輸入的校正點為何組。0:輸入到 SrcPT、1:輸入到 GroundPT
PointNumber	**in**	校正點編號。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example code:

```
eCCalib3D cali3D;//宣告 eCCalib3D 物件
cali3D.SetFourPoint(100, 100, 0);//寫入(100, 100)至 SrcPT[0]，自動跳號
cali3D.SetFourPoint(200, 100, 0); //寫入(500, 100)至 SrcPT[1]，自動跳號
cali3D.SetFourPoint(100, 200, 0, 2); //寫入(100, 500)至 SrcPT[2]，指定編號
cali3D.SetFourPoint(200, 200, 0, 3); //寫入(500, 500)至 SrcPT[3]，指定編號
```

Tips:使用者可將座標資訊直接寫入特定的特徵點資訊中。若選擇不自行輸入校正點編號，則會有內建的計數器記錄輸入點的順序。當輸入的點數大於四點，所輸入的第五個點座標資訊將會更新第一點的資訊，而輸入的第六個座標點則會更新第二點的資訊，依此類推。

9.2.6. CleanPoint

清除所有輸入的 SrcPT 與 GroundPT。

函式原形：

```
BOOL    CleanPoint();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

9.2.7. CleanSourcePoint

清除所有輸入的 SrcPT。

函式原形：

```
BOOL    CleanSourcePoint();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

9.2.8. CleanGroundPoint

清除所有輸入的 GroundPT。

函式原形：

```
BOOL   CleanGroundPoint();
```

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

9.2.9. SetSampleSize

設定要校正的目標在校正後的尺寸。

函式原形：

```
BOOL   SetSampleSize(UINT16 Width,
                     UINT16 Height);
```

傳入值：

Width	**in**	目標物寬
Height	**in**	目標物高
Sample size	**in**	僅在自動校正時需要進行設定，未設定的 sample size 將與來源影像的尺寸相同。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

9.2.10. GetSampleWidth

取得目前設定的目標物寬。

函式原形：

```
INT32   GetSampleWidth();
```

回傳值：

目標物寬。

9.2.11. GetSampleHeight

取得目前設定的目標物高。回傳目標物高。

函式原形：

```
INT32   GetSampleHeight();
```

回傳值：

目標物高。

9.2.12. SetImageSize

設定要校正後影像的尺寸。

函式原形：

```
BOOL   SetImageSize(UINT16 Width,
                    UINT16 Height);
```

傳入值：

Width	**in**	校正後影像寬
Height	**in**	校正後影像高 ※使用者若未輸入影像尺寸，則會以來源影像為輸出影像尺寸。

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

9.2.13. GetImageWidth

取得目前設定的輸出影像寬。回傳輸出影像寬。

函式原形：

```
INT32   GetImageWidth();
```

回傳值：

輸出影像寬。

9.2.14. GetImageHeight

取得目前設定的輸出影像高。回傳輸出影像高。

函式原形：

```
INT32   GetImageHeight();
```

回傳值：

輸出影像高。

9.2.15. Calibration

eCCalib3D 主要功能，根據設定的相關對應點校正傳入的來源影像，並在完成校正後回傳 TRUE；若函式運行中無法取得所需的資訊，將會設定 error code，並且回傳 FALSE。

使用之前必須確認已完成 SrcPT 的設定以及 GroundPT 的設定，若使用者選擇讓系統自行決定 GroundPT，則需要設定 Sample size。自動校正會將「Sample」校正至「Image」中央，根據設定的 Image size 以及 Sample size 計算出 GroundPT 的四個點，並將 SrcPT 設定的點進行排序以符合 GroundPT 的順序並且進行校正。自動校正僅適用當影像歪斜角度為+/-90 度時。

函式原形：

```
BOOL    Calibration(eCImage *SrcImage,
                    bool GD);
```

傳入值：

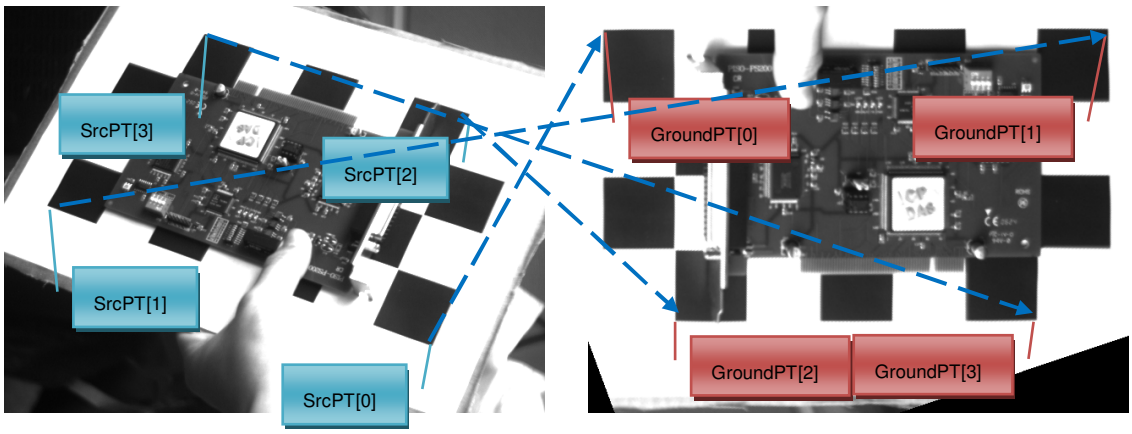
SrcImage	**in**	來源影像。
GD	**in**	使用者是否自行設定 GroundPT。 true:以 SrcPT 與 GroundPT 共四組對應點進行校正 false:以設定的 sample size 與 image size 產生 GroundPT 進行校正

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

Example code 1(手動校正):

```
eCCalib3D cali3D;//宣告 eCCalib3D 物件
//設定 SrcPT 與 GroundPT，編號與順序必須一致
cali3D.SetImageSize(640, 480); //設定影像尺寸為 640*480
cali3D.Calibration(simage, true);//以手動設定的四組對應點進行校正
```

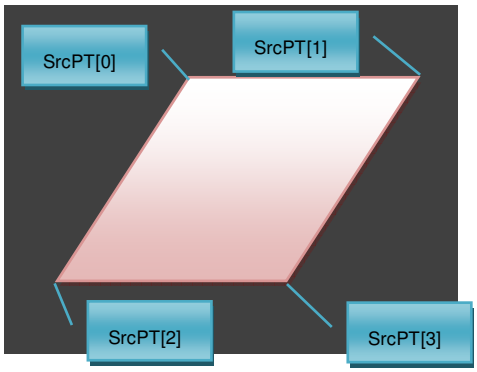


來源影像　　校正結果

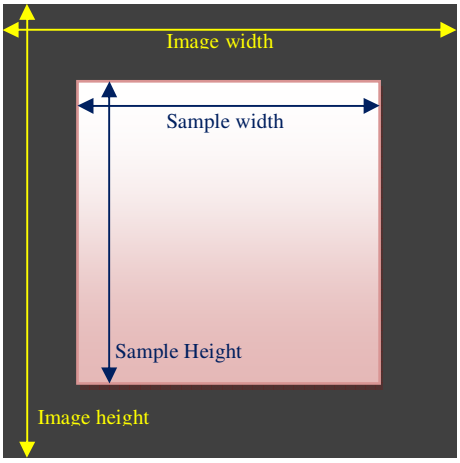
圖四十二、　手動校正示意圖

Example code 2(自動校正):

```
eCCalib3D cali3D;//宣告 eCCalib3D 物件
//設定 SrcPT，無關對應順序。
cali3D. SetSampleSize(100, 100);//校正後的目標物尺寸為 400*400
cali3D. SetImageSize(300, 300);//校正後影像尺寸為 600*600
cali3D.Calibration(simage, false);//自動校正
```



來源影像與自動排序之 SrcPT



校正結果

圖四十三、 自動校正示意圖

Tips:當 Calibration 完成後，會將該次運算的矩陣暫存。若使用者不再使用 SetFourPoint 或者 SetSampleSize 等函式改變運算資訊，之後的運算將會直接套用該轉換矩陣直接進行校正，在校正大量影像時可加速運算。

9.2.16. GetUnwarpImage

在校正後取得校正影像。

函式原形：

```
BOOL   GetUnwarpImage(eCImage *&DstImage);
```

傳入值：

DstImage **out** 校正完成的影像

回傳值：

運作成功將回傳 TRUE，否則回傳 FALSE。

9.2.17. GetErrorCode

取得 eCCalib3D 運作中回報的 error code。

函式原形：

```
INT32   GetErrorCode();
```

回傳值：

eCCalib3D 運作中回報的 error code，定義於 ErrorCode.h。請參考 ErrorCode 列表。

10. 錯誤代碼－ErrorCode

10.1. ErrorCode 簡介

ErrorCode 包含 EzCheck Vision Library 運作中可能出現的問題或者例外狀況。當函式在使用上回傳 FALSE 或者無法得到預期結果時，可透過 GetErrorCode 相關功能將 errorcode 輸出查看，快速找到問題。

10.2. ErrorCode 列表

10.2.1. enum GENERAL_ERRORS

宣告內容：

```
enum GENERAL_ERRORS
{
	ERROR_OK = 0,

	//General Error
	ERROR_INVALID_FILE_NAME = 100,
	ERROR_INVALID_EXTENSION_NAME = 101,
	ERROR_INVALID_IMAGE_FORMAT = 102,
	ERROR_INVALID_BPP = 103,
	ERROR_INVALID_DATA_STRUCTURE = 104,
	ERROR_INVALID_IMPORT_PARAMETER = 105,
	ERROR_INVALID_CLEAN = 106,
	ERROR_INVALID_FOLDER = 107,
	ERROR_RESULT_ZERO = 108,

	//eCImage Error
	ERROR_ECIMAGE_SOURCE_ERROR = 109,
	ERROR_ECIMAGE_CREATE_ERROR = 110,
	ERROR_ECIMAGE_DIDNOT_NEW_ERROR = 111,
	ERROR_ECIMAGE_IS_NULL = 112,
	ERROR_ECIMAGE_LOAD_ERROR = 113,
	ERROR_ECIMAGE_COPY_ERROR = 114,
	ERROR_ECIMAGE_SETTING_ERROR = 115,
	ERROR_ECIMAGE_NOFILENAME_ERROR = 116,
	ERROR_ECIMAGE_16BITIMAGE_ERROR = 117,
	ERROR_ECIMAGE_ROIRANGE_ERROR = 118,
	ERROR_ECIMAGE_CAMERA_SIZE = 119,

	//eCBlob Error
	ERROR_INVALID_BLOBANALYSIS_MODE = 120,
	ERROR_BLOB_LIMIT = 121,
```

```
    ERROR_INVALID_BLOBANALYSIS = 122,
    ERROR_INVALID_BASIC_FEATURE = 123,
    ERROR_INVALID_IMAGE_DATA = 124,
    ERROR_INVALID_BLOB_NUMBER = 125,
    ERROR_INVALID_PROCESS_MODE = 126,
    ERROR_INVALID_FEATURE_TYPE = 127,
    ERROR_INVALID_OPTION_TYPE = 128,

    ERROR_INVALID_SORT_TYPE = 129,
    ERROR_INVALID_ADVANCED_FEATURE = 130,
    ERROR_INVALID_ADD_MERGE_LIST = 131,
    ERROR_INVALID_MANUAL_MERGE = 132,
    ERROR_INVALID_THRESHOLD_VALUE = 133,

    //eCOCR Error
    ERROR_INVALID_DATABASE_TYPE = 134,
    ERROR_INVALID_OCR_LEARNING = 135,
    ERROR_YET_LEARNING_DATABASE = 136,
    ERROR_INVALID_OCR_COUNT = 137,
    ERROR_INVALID_OCR_TYPE = 138,
    ERROR_YET_OCR = 139,
    ERROR_INVALID_ROW_LIMIT = 140,
    ERROR_INVALID_CLEAN_DATABASE = 141,
    ERROR_INVALID_RECOGNITION_DATA = 142,

  //eCImg Shared error
  ERROR_EZCHECK_FUNC_ERROR = 143,
  ERROR_SOURCE_IMAGE_ERROR = 144,
  ERROR_DEST_IMAGE_ERROR = 145,
  ERROR_MASK_INFO_ERROR = 146,
  ERROR_COLOR_UNCHANGED = 147,

  //Calibration 3D error
  ERROR_CALIB3D_POINT_ERROR = 148,
  ERROR_CALIB3D_UNWARP_IMAGE_ERROR = 149,
  ERROR_CALIB3D_UNWARP_NOT_READY = 150,

  //Template matching error
  ERROR_TM_NODATA_ERROR = 151,
  ERROR_TM_SORT_INPUT_ERROR = 152,

  //USBHK Error
  ERROR_CHECK_USBHK = 153,
};
```

內容簡介：

ErrorCode	No.	Meaning
ERROR_OK	0	無錯誤回報或運作正常。
General Error		
ERROR_INVALID_FILE_NAME	100	檔案路徑錯誤或該檔案不存在，路徑錯誤請注意路徑是否有空白，因空白不明顯示，容易被遺忘
ERROR_INVALID_EXTENSION_NAME	101	該檔案無 Extension Name(副檔名)
ERROR_INVALID_IMAGE_FORMAT	102	非圖檔格式
ERROR_INVALID_BPP	103	本功能現階段版本只支援 24 與 8 位元灰階圖，請注意是否為支援格式
ERROR_INVALID_DATA_STRUCTURE	104	該類別中不存著影像資料，請檢查類別中的 data structure 是否存在著資料
ERROR_INVALID_IMPORT_PARAMETER	105	請檢查導入的參數是否正確合理
ERROR_INVALID_CLEAN	106	資料不存在，無法清除
ERROR_INVALID_FOLDER	107	資料夾不存在
ERROR_RESULT_ZERO	108	結果為 0，請檢查輸入資料與設定條件是否正確
eCImage Error		
ERROR_ECIMAGE_SOURCE_ERROR	109	傳入 eCImage 的 source image 有問題
ERROR_ECIMAGE_CREATE_ERROR	110	傳入創建 eCImage 時有問題
ERROR_ECIMAGE_DIDNOT_NEW_ERROR	111	eCImage 未經過建構子初始化
ERROR_ECIMAGE_IS_NULL	112	eCImage 為空
ERROR_ECIMAGE_LOAD_ERROR	113	讀檔案時出錯
ERROR_ECIMAGE_COPY_ERROR	114	GetCopy 時出錯
ERROR_ECIMAGE_SETTING_ERROR	115	設定 eCImage 時出錯
ERROR_ECIMAGE_NOFILENAME_ERROR	116	檔名錯誤
ERROR_ECIMAGE_16BITIMAGE_ERROR	117	無法顯示 16bit 影像
ERROR_ECIMAGE_ROIRANGE_ERROR	118	ROI 範圍與原始影像一樣
ERROR_ECIMAGE_CAMERA_SIZE	119	eCImage 與攝影機取像 buffer 不合
eCBlob Error		
ERROR_INVALID_BLOBANALYSIS_MODE	120	無效的分析模式，請參考 datafile.h 中 ANALYSISCLASS enum 的描述
ERROR_BLOB_LIMIT	121	超過 blob 的限定值，現在支援到 16bits，65535

		個 blob 的數量限制
ERROR_INVALID_BLOBANALYSIS	122	分析結果為 0,請確認導入參數是否正確與資料是否存在
ERROR_INVALID_BASIC_FEATURE	123	不存在基本特徵，請檢查是否進行過 Blob Analysis 或 Blob Analysis 是否執行無誤，其函式中會計算出 Basic Feature
ERROR_INVALID_IMAGE_DATA	124	使用與 Blob Analysis 時用的影像大小不符
ERROR_INVALID_BLOB_NUMBER	125	不存在 blob analysis 後的 blob 編號
ERROR_INVALID_PROCESS_MODE	126	無效的處理模式，請參考 PROCESSMODE enum 的描述
ERROR_INVALID_FEATURE_TYPE	127	無效的處理模式，請參考 SELECTFEATURE enum 的描述
ERROR_INVALID_OPTION_TYPE	128	無效的處理模式，請參考 SELECTOPTIONS enum 的描述
ERROR_INVALID_SORT_TYPE	129	無效的處理模式，請參考 SORTOPTIONS enum 的描述
ERROR_INVALID_ADVANCED_FEATURE	130	不存在進階特徵，請檢查是否進行過進階特徵計算
ERROR_INVALID_ADD_MERGE_LIST	131	該 blob 編號已存在 mergelist 中
ERROR_INVALID_MANUAL_MERGE	132	mergelist 中必須存在 2 個 blob 編號以上
ERROR_INVALID_THRESHOLD_VALUE	133	Select 函式導入之臨界值錯誤，請檢查最大值與最小值臨界值是否符合定義
eCOCR Error		
ERROR_INVALID_DATABASE_TYPE	134	database 字體顏色設定錯誤或未設定 database 字體顏色，請參考 CharacterType enum 的描述
ERROR_INVALID_OCR_LEARNING	135	learning 該 database 的數量為 0，注意該 database 中圖檔格式與是否存在圖檔檔案
ERROR_YET_LEARNING_DATABASE	136	尚未 learning database
ERROR_INVALID_OCR_COUNT	137	要識別的數量為 0
ERROR_INVALID_OCR_TYPE	138	識別的顏色錯誤，請參考 ANALYSISCLASS enum 的描述
ERROR_YET_OCR	139	尚未進行 OCR 識別
ERROR_INVALID_ROW_LIMIT	140	排序超過上限 255 行的數量

ERROR_INVALID_CLEAN_DATABASE	141	清除 database 失敗，請確定是否有進行過 learning 的動作
ERROR_INVALID_RECOGNITION_DATA	142	Source 與 Blob 影像 size 不相同
eCImg Shared Error		
ERROR_EZCHECK_FUNC_ERROR	143	其他的例外狀況，請回報使用過程的完整步驟給開發者
ERROR_SOURCE_IMAGE_ERROR	144	來源影像錯誤
ERROR_DEST_IMAGE_ERROR	145	目的影像錯誤
ERROR_MASK_INFO_ERROR	146	傳入的遮罩資訊有誤，所以無法進行運算
ERROR_COLOR_UNCHANGED	147	傳入的影像本身是灰階，無法進行通道轉換
eCCalib3D Error		
ERROR_CALIB3D_POINT_ERROR	148	傳入的特徵點座標超過影像尺寸
ERROR_CALIB3D_UNWARP_IMAGE_ERROR	149	傳入的特徵點座標超過影像尺寸
ERROR_CALIB3D_UNWARP_NOT_READY	150	還沒做過校正就試圖讀取校正影像
eCTM Error		
ERROR_TM_NODATA_ERROR	151	還沒做過 Template matching 就 access data
ERROR_TM_SORT_INPUT_ERROR	152	還沒做過 Template matching 就 access data
USB Hardware Key Error		
ERROR_CHECK_USBHK	153	未檢查到相對應的 USB hardkey

11. 工具軟體(EzCheck Utility)

ICPDAS 提供工具軟體讓使用者進行 EzCheck Vision Library 的使用與評估。在工具軟體上看得到的功能，使用者就能藉由 EzCheck Vision Library 發展出來。

11.1. 主要特點

- **以 EzCheck Vision Library 進行開發**

工具軟體中所有的影像存取、處理以及分析均由 EzCheck Vision Library 所提供的功能完成。

- **EzCheck Vision Library 各主要功能一目了然**

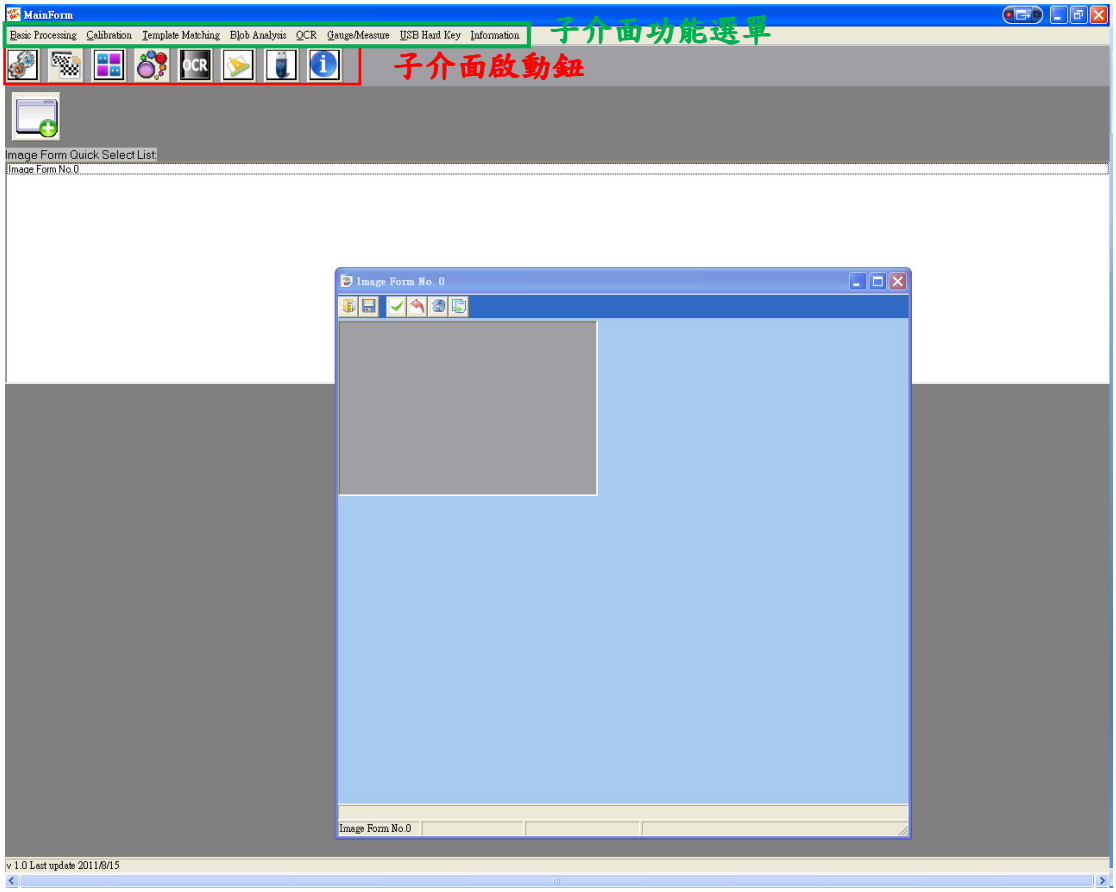
工具軟體包含所有 EzCheck Vision Library 主要功能的實際應用介面，讓使用者能快速了解 EzCheck Vision Library。

- **實用的功能介面**

工具軟體為 EzCheck Vision Library 各主要功能設計了簡單實用的功能介面。多種圖形化的操作除了可以用來進行影像評估實驗，還可以體驗 EzCheck Vision Library 在開發上帶來的方便。

11.2. EzCheck Utility 主畫面與功能簡介

EzCheck Utility 主畫面包含各個子介面的啟動鈕或選單，以及新增影像視窗的功能鍵。各子介面的啟動鈕與選單功能相同，點選啟動鈕或者選單都可以開啟相對應的子介面，同時也會關閉其他的子界面(USB 硬體鎖介面與資訊介面除外)。



圖四十四、 EzCheck Utility 主畫面

11.3. 影像視窗

影像視窗主要進行影像的開檔、存檔、更新與顯示，並且輔助各個主要功能的影像介面使用。使用者可同時開啟若干個影像視窗，以及多張影像，進行多張影像的交互運算時更加便利。只要點選主畫面上的圖示即可開啟一個新的影像視窗。

11.3.1. 影像與檔案管理

影像視窗上有六個按鈕，分別是**開啟檔案**、**儲存檔案**、**更新檔案**、**復原檔案**、**重置檔案**與**複製影像視窗**。

(1) 開啟檔案：自檔案開啟影像，並顯示至影像視窗上。最初讀取的影像將會以一個固定的影像 buffer，作為**重置影像**的依據。

(2) 儲存檔案：儲存最後更新原始影像為檔案。請在儲存檔案之前先進行**更新影像**。

(3) 更新影像：EzCheck Utility 中許多功能在運算後會將結果輸出至暫存影像上。因此影像視窗上顯示的影像有時僅是暫存影像，在進行其他運算後不會保留，建議使用者在進行完各個步驟後，進行影像更新，將暫存影像與來源影像同步，保存剛完成的處理。

(4) 復原影像：復原上一步尚未更新的動作。通常是還未更新過的來源影像。

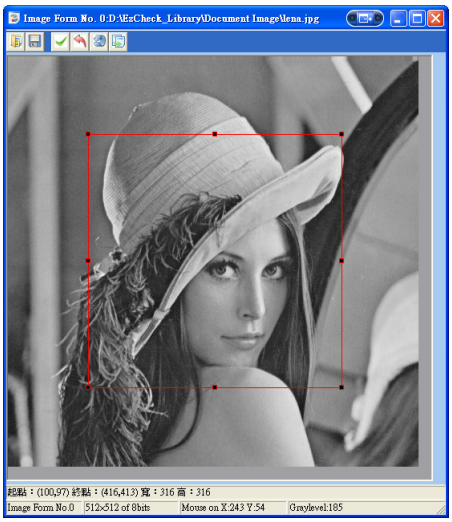
(5) 重置影像：將影像重置。通常是將所有影像 buffer 回復為最初的讀取影像。

(6) 複製影像視窗：新增一個影像視窗，並且將所有檔案與類別資訊複製至新的視窗中。

圖四十五、 影像視窗與按鈕

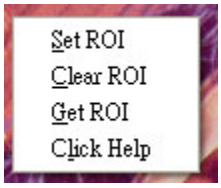
11.3.2. ROI 管理

另外使用者也可直接在畫面上以拖曳的方式繪製 ROI 框架，框架建立後也會同時將 ROI 設定。框架上的八個控制點以及框架內都可以利用拖曳的方式調整 ROI 的大小與位置。點選框架以外的地方可重置 ROI。



圖四十六、 影像視窗與 ROI

在影像視窗上點選滑鼠右鍵可開啟右鍵選單。包括 Set ROI, Clear ROI, Get ROI 以及 Click Help，其中 Get ROI 可將 ROI 框選範圍擷取出來。



圖四十七、 影像視窗右鍵選單

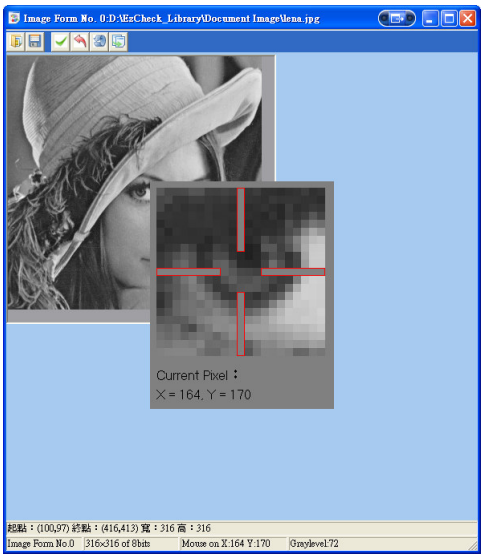
圖四十八、 影像視窗與 Get ROI。

Tips:若要保留擷取後的 ROI 範圍影像，請記得進行影像更新。

11.3.3. 點選輔助 Click Help

EzCheck Utility 中部分功能需要進行較精確的畫面點選，如：影像校正。點

選輔助提供類似放大鏡的功能，幫助使用者點進行游標的微調。在影像視窗上點選滑鼠右鍵開啟選單，並選擇 Click Help，即可開啟 Click Help 介面。再次點選右鍵選單上的 Click Help 即可關閉此介面。

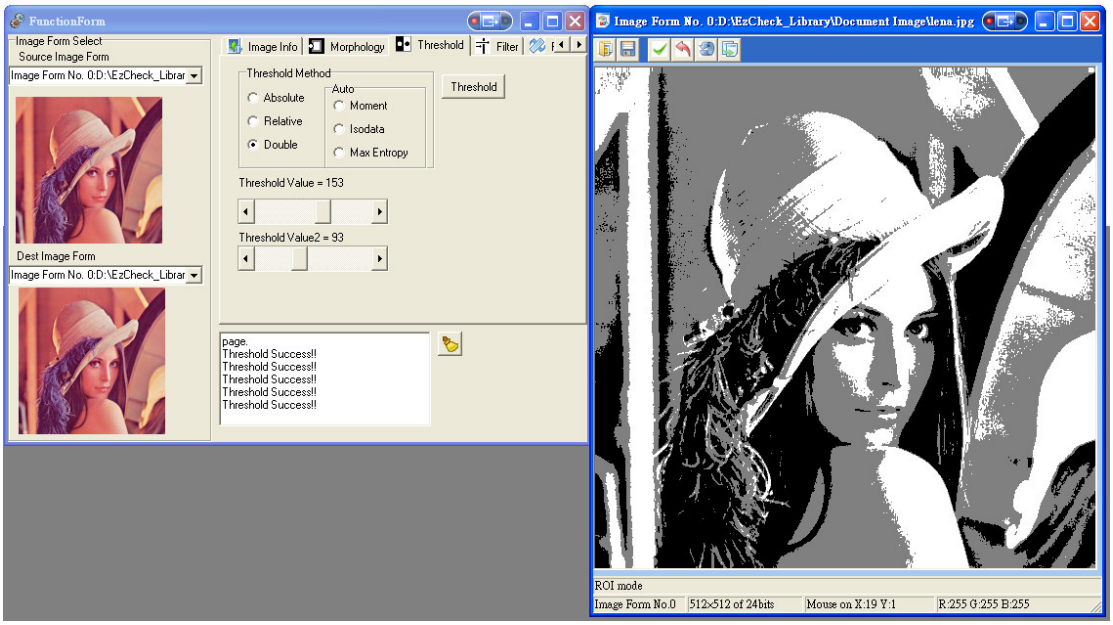


圖四十九、 影像視窗與點選輔助介面。

11.4. 基礎影像處理功能介面

提供二值化、色彩通道轉換、型態學運算...等於影像前處理中常見的演算法。在此將影像進行前處理後，使用其他功能進行分析，效果更佳。

使用者必須要先選擇 Source Image Form 以及 Dest Image Form 才能開啟功能介面。基礎影像處理功能介面上的所有運算都僅會輸出至影像視窗中的 Temp Image，方便使用者嘗試各種影像處理的功能。因此當使用者完成某步驟並且確認該次的運算是成功的，**請務必在進行下一個運算之前進行影像更新**。



圖五十、　基礎影像處理功能介面

11.5. 樣板比對功能介面

搭配影像視窗互動，提供 eCTM 功能展示。包含限制相似度的樣版比對、結果的篩選與顯示。在結果影像或者結果表格上點選，使用者將更能清楚地了解比對結果。

11.5.1. 影像視窗選擇

在樣板比對介面中，使用者將需要開啟至少兩個影像視窗，一個讀取要進行樣板比對的檔案，選為 Source Image Form，另一個讀取要比對的樣板，選為 Template Image Form。使用者同樣必須指定 Dest Image Form 以輸出影像，此影像視窗可由上述兩者兼任，也可開啟第三個影像視窗來做為目的影像的視窗。三個影像視窗指定完成方可進行樣板比對。

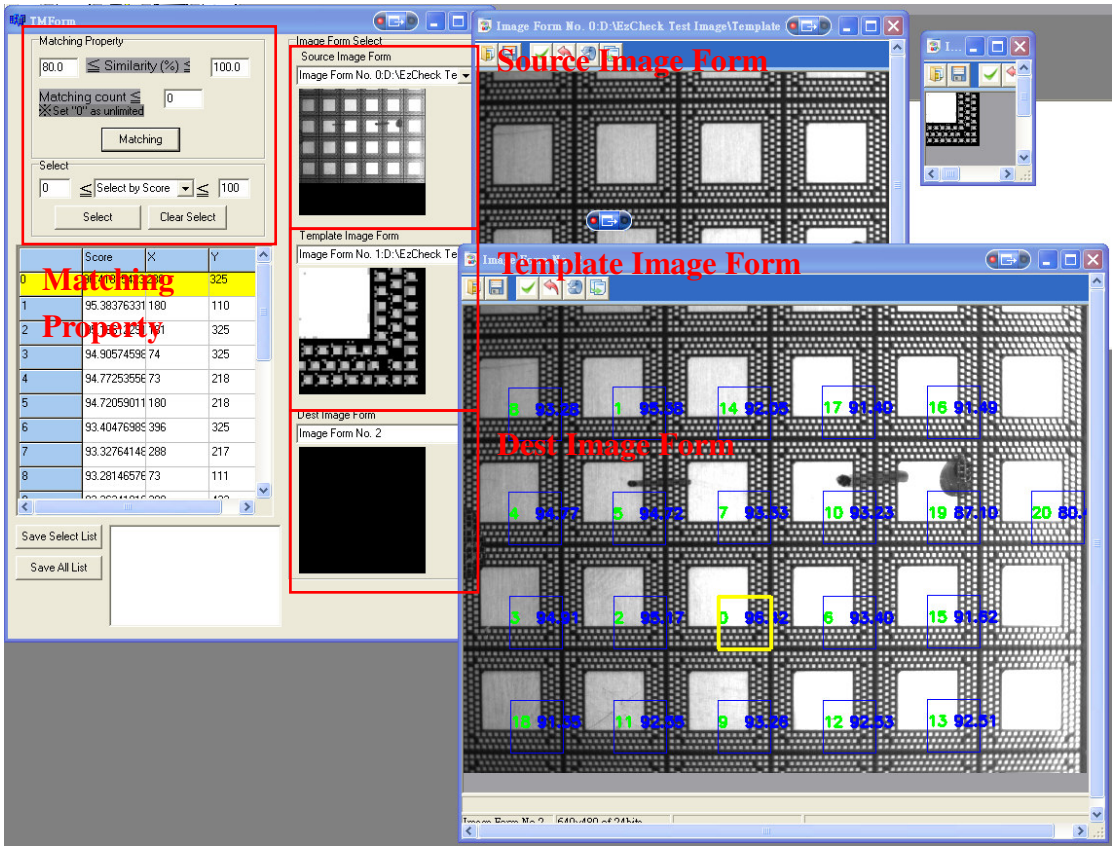
11.5.2. 樣板比對與相似區域資訊

影像視窗指定完成後，可在 Matching Property 中進行比對參數的調整，並且進行樣板比對。比對完成後，各個相似區域的結果將會顯示於表格中，相似區域的影像也會顯示於畫面上。於此介面上可進行篩選的設定，對相似區域資訊進行篩選並將篩選結果同時顯示於表格與影像上，點選表格第一列則會對相似區域

資訊進行排序。

11.5.3. 樣板比對與影像視窗互動

點選 Dest Image Form 上樣板比對的輸出影像，會自動標記出該相似區域於表格中的資訊；點選表格中的資訊，也會標記出該資訊於影像視窗上的位置。



圖五十一、 樣版比對介面

11.6. Blob 分析功能介面

搭配影像視窗互動，提供 eCBlob 功能展示。包含黑 blob、白 blob 或者黑白 blob 的分析，當然也包含各種 blob 特性的篩選。與樣板比對功能相似 Blob 也支援結果表格與影像視窗的互動，使用者將能更快的找到所需的 blob 資訊。

11.6.1. 影像視窗選擇

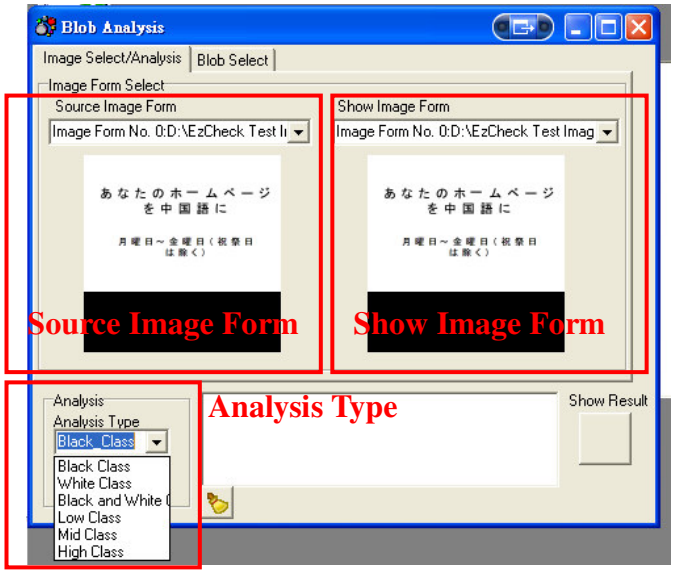
在 Blob 分析功能介面中，使用者需要開啟至少一個影像視窗，當作 Source Image Form。使用者可指定任一影像視窗作為 Show Image Form 來輸出 blob 分析的結果。

Tips:建議使用者先將影像進行二值化，使 blob 分析的結果更理想。

11.6.2. Blob 分析與 Blob 資訊

影像視窗指定完成後，指定 Analysis Type，對影像進行 blob 分析。分析完成後會將分析出的各個 blob 標示於畫面上，同時相關資訊也會顯示於表格中。

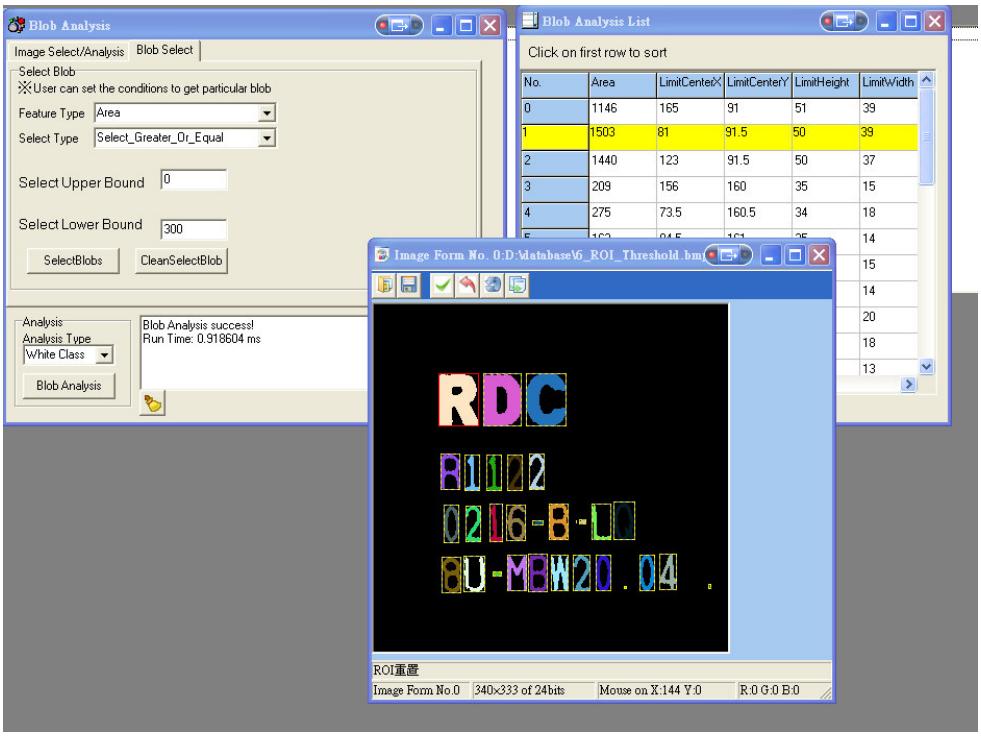
於 Blob Select 頁面中，使用者可根據 blob 的不同特徵，進行各種不同方式的篩選。篩選後的結果將會即時顯示於表格資訊以及影像資訊上。



圖五十二、 Blob 分析介面與說明

11.6.3. Blob 分析與影像視窗互動

點選 Show Image Form 上 blob 分析的輸出影像，會自動標記出該相似區域於表格中的資訊；點選表格中的資訊，也會標記出該資訊於影像視窗上的位置。



圖五十三、 Blob 分析介面

11.7. OCR 功能介面

提供 eCOCR 功能展示。對影像進行 Blob 分析之後，使用者可定義將一定範圍的 Blob 資訊進行整合，並且透過學習指定資料庫完成文字辨識。

11.7.1. 影像視窗選擇

在 OCR 功能介面中，使用者需要開啟一個影像視窗，當作 Source Image Form。

Tips:建議使用者先將影像進行二值化，使 blob 分析以至於 OCR 的結果更理想。

11.7.2. Blob 分析與 Blob 資訊處理

影像視窗指定完成後，指定黑字或白字，對影像進行 blob 分析。分析完成後會將分析出的各個 blob 標示於畫面上，同時相關資訊也會顯示於表格中。

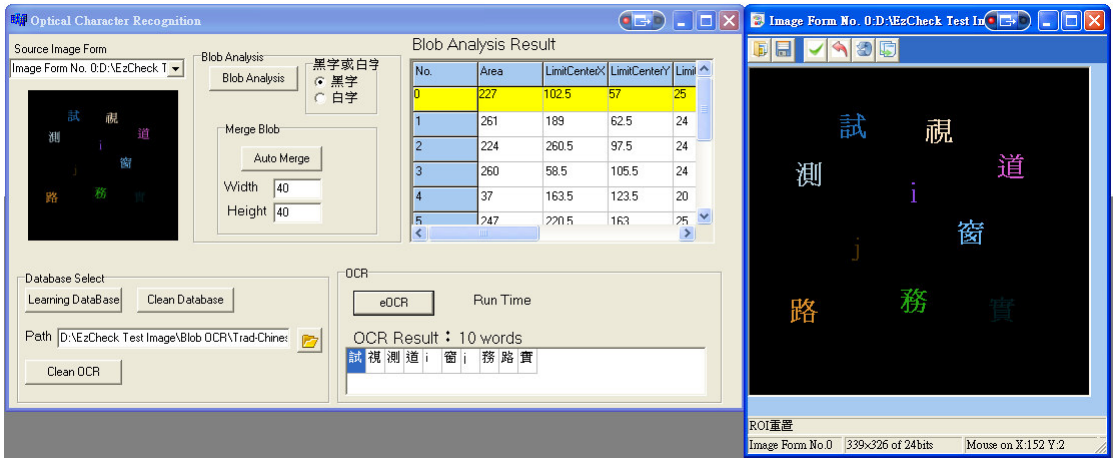
使用者可利用 Select 以及 Merge，留下需要的 blob，使後續 OCR 的結果

更佳。

11.7.3. 資料庫讀取與 OCR

使用者可點選資料路徑按鈕選擇路徑，也可自行輸入路徑。路徑輸入完成後點選 Learning DataBase 將 OCR 影像資料庫讀入。資料庫讀取完成後，即可進行 OCR，並且於下方的表格中看到 OCR 結果。

Tips:由於開發環境的限制(不支援某些字型的顯示)，部分字型可能無法顯示，如韓文。但若輸出的結果數量是正確的，表示函式庫本身是有正常輸出該文字的。



圖五十四、 OCR 介面

11.8. 測量功能介面

搭配影像視窗互動，提供 eGauge 功能展示。在指定的影像視窗上創建 eGauge 工具後，可直接在影像視窗上以點選拖曳的方式改變 eGauge 元件的大小與位置，將 eGuage 元件的使用變得相當便利。

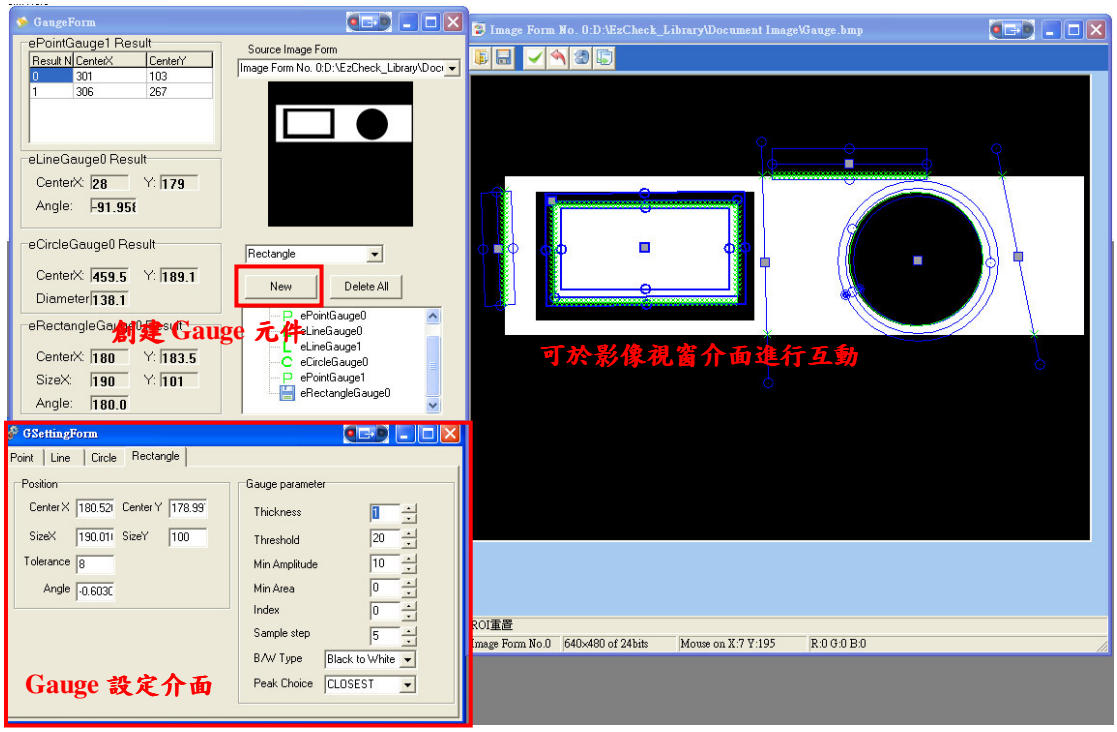
11.8.1. 影像視窗選擇

在 eGauge 功能介面中，使用者需要開啟一個影像視窗，當作 Source Image Form。

11.8.2. eGauge 元件創建與設定

影像視窗指定完成後，使用者可選擇希望創建的 eGauge 元件，包括 Point、Line、Circle、Rectangle。於 ComboBox 中選擇後，按下 New 即可將該元件創建於影像視窗上，同時也會顯示參數設定介面。若參數設定介面被關閉，於下方的樹狀圖上雙擊所要調整的物件也可以開啟參數設定介面。

使用者可透過畫面上各個特徵點的拖曳改變 eGauge 元件的位置、容許值、角度等資訊，也可於參數設定介面上輸入數值，兩者的參數是彼此同步且會即時更新的。


GaugeForm
ePointGauge1 Result
Result N
CenterX
CenterY
0
301
103
1
306
267
eLineGauge0 Result
CenterX: 28
Y: 179
Angle:
eCircleGauge0 Result
CenterX: 459.5
Y: 189.1
Diameter: 138.1
eRectangleGauge0 Result
CenterX: 180
Y: 183.5
SizeX: 190
Y: 101
Angle: 180.0
Source Image Form
Image Form No. 0:D:\EzCheck_Library\Docu
Rectangle
New
Delete All
ePointGauge0
eLineGauge0
eCircleGauge0
ePointGauge1
eRectangleGauge0
創建 Gauge 元件
GSettingForm
Point
Line
Circle
Rectangle
Position
Center X
Center Y
SizeX
SizeY
Tolerance
Angle
Gauge parameter
Thickness
Threshold
Min Amplitude
Min Area
Index
Sample step
B/W Type
Black to White
Peak Choice
CLOSEST
Gauge 設定介面
Image Form No. 0:D:\EzCheck_Library\Document Image\Gauge.bmp
可於影像視窗介面進行互動
ROI重置
Image Form No.0
640x480 of 24bits
Mouse on X:7 Y:195
R:0 G:0 B:0

圖五十五、 測量功能介面

11.9. 影像校正功能介面

搭配影像視窗互動，提供 eCCalib3D 功能展示。使僅需在指定為來源影像以及樣版影像的影像視窗上各點四個對應點，即可將扭曲傾斜的影像校正為正規影像，不需要額外的參數輸入。影像視窗也支援放大鏡功能，幫助使用者進行點選。

11.9.1. 影像視窗選擇

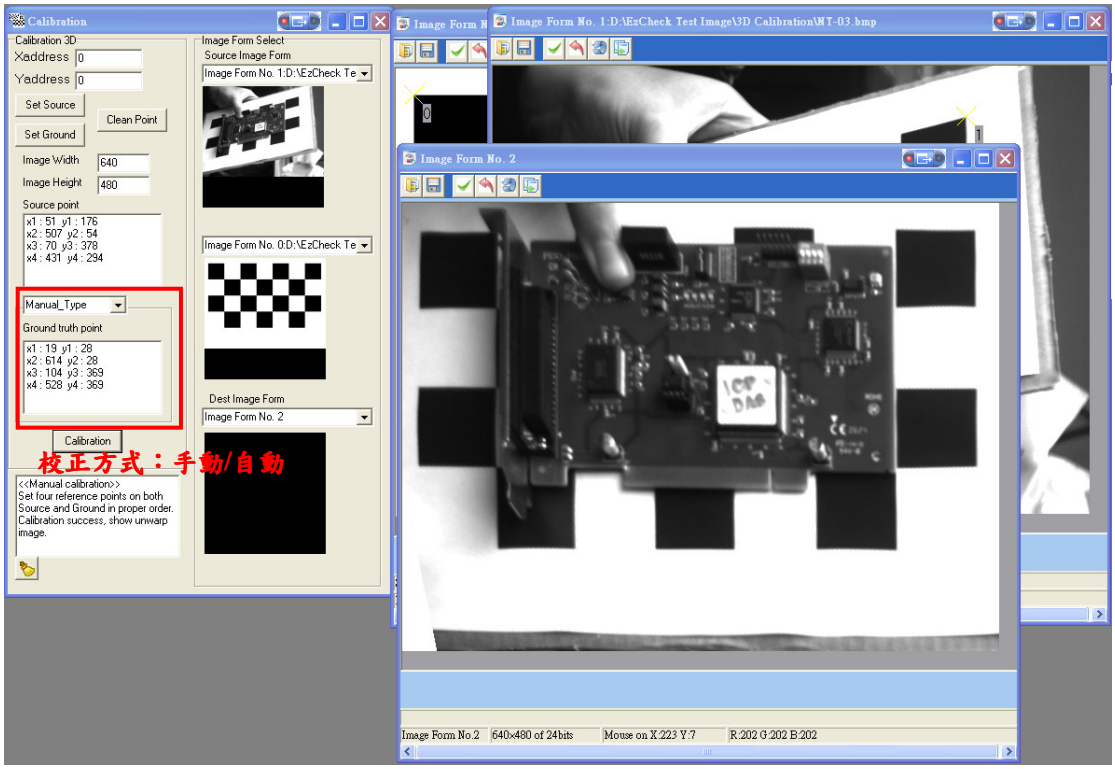
在 eCCalib3D 功能介面中，使用者需要開啟至少一至二個影像視窗，當作 Source Image Form 讀取要進行校正的影像以及 Ground Truth Image Form 讀取作為校正樣板的影像(若使用者選用自動校正則不需要)。使用者也需要指定任意一個影像視窗作為 Dest Image Form。

11.9.2. 影像視窗點選

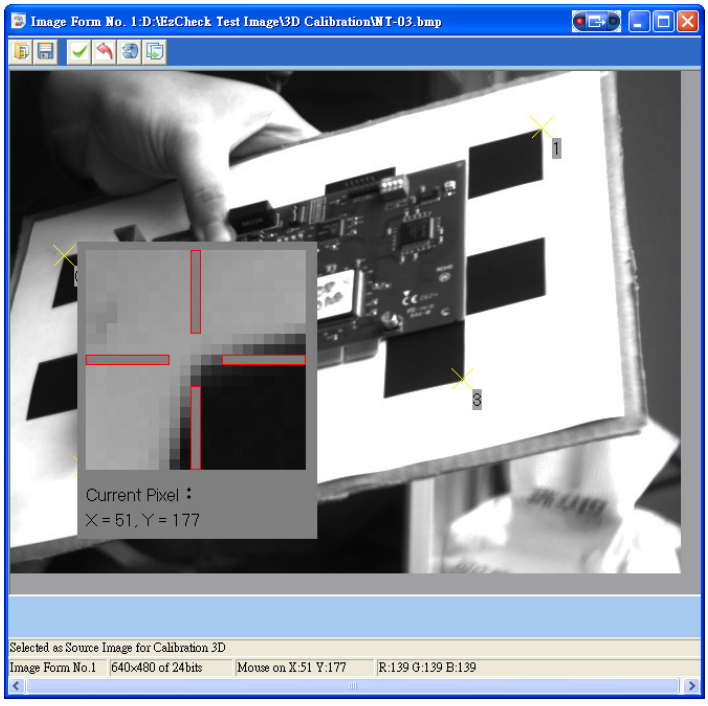
若使用手動校正(Manual_Type)，則使用者會需要在 Source Image Form 與 Ground Truth Image Form 兩個影像視窗上點選各四個校正參考點。點選完參考點後即可進行校正。

若使用自動校正(Auto_Type)，使用者需要在 Source Image Form 上點選四個參考點，輸入 Sample Width 與 Sample Height，並進行校正。

點選畫面時可開啟「點選輔助」，更精確的畫面點選，將可以得到更佳的校正結果。



圖五十六、 影像校正功能介面



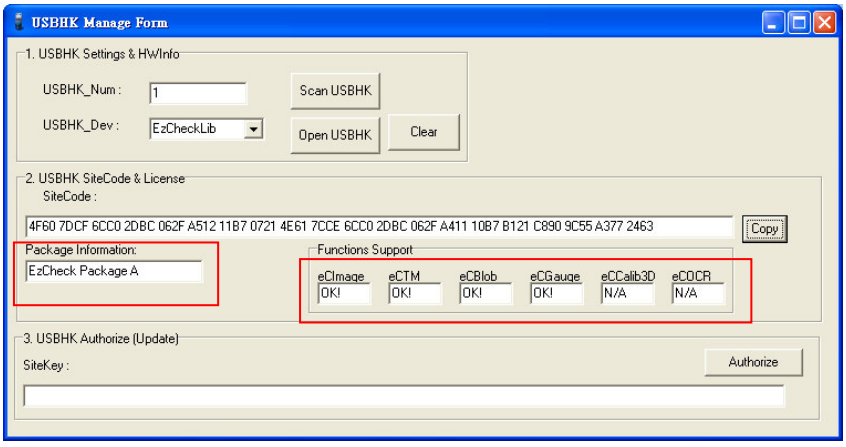
圖五十七、 「Click help」介面可幫助使用者更準確的點選影像

12. USB 硬體鎖(USB Hardware Key) 與套件(Package)

12.1. EzCheck USB 硬體鎖與套件簡介

為維護購買 EzCheck Vision Library 的使用者權益，每一套 EzCheck Vision Library 都將會以 USB 硬體鎖進行保護。對於不同的功能需求，泓格提供五種 EzCheck Vision Library 的套件來配合。

EzCheck Vision Library 套件支援列表						
套件	eCImage	eCTM	eCBlob	eCGauge	eCOCR	eCCalib3D
EzCheck-A	ˇ	ˇ	ˇ	ˇ		
EzCheck-B	ˇ	ˇ	ˇ		ˇ	
EzCheck-C	ˇ		ˇ	ˇ		ˇ
EzCheck-D	ˇ	ˇ		ˇ		ˇ
EzCheck-ALL	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ



圖五十八、 EzCheck Utility 硬體鎖管理介面。可看到該硬體鎖為套件 A，支援項目為 eCImage、eCTM、eCBlob、eCGauge。

13. FAQs

13.1. Unicode 支援問題

13.1.1. Borland C++ Builder

EzCheck Vision Library 中的 eCOCR 為多國語言文字辨識，只要 OS 支援顯示的文字，就可以透過 eCOCR 輸出。但由於 Borland C++ Builder(後簡稱 BCB)本身並無支援 unicode 的顯示，因此部分語言的文字在 BCB 下可能無法正常顯示，如：日文、韓文。但使用者仍可透過加裝 BCB 的 unicode 的外掛解決多國文字顯示的問題。

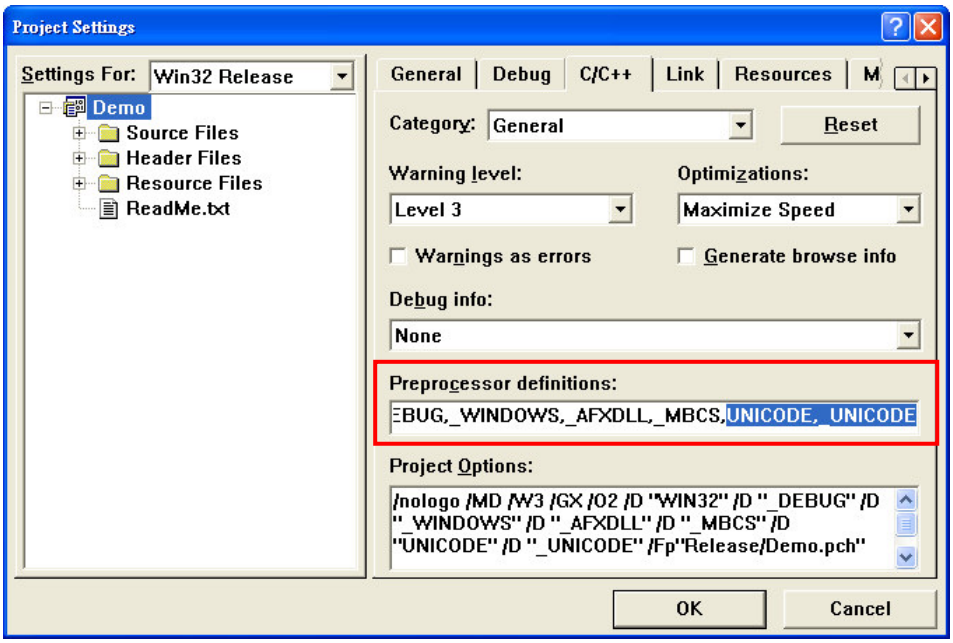
使用者可參考 http://www.tmssoftware.com/site/tmsuni.aspTMS Unicode Component Pack，或是其它支援 BCB unicode 的外掛軟體，將 unicode 相關支援元件掛載於 BCB 下。專案開發中，則可利用外掛元件進行多國語言的顯示。

13.1.2. VC

Visual C++本身就有提供 Unicode 的支援，只要進行 unicode 的相關設定，以及程式的撰寫即可。以下以 Visual C++ 6.0 為例，進行 unicode 的設定。

- 設定 1：

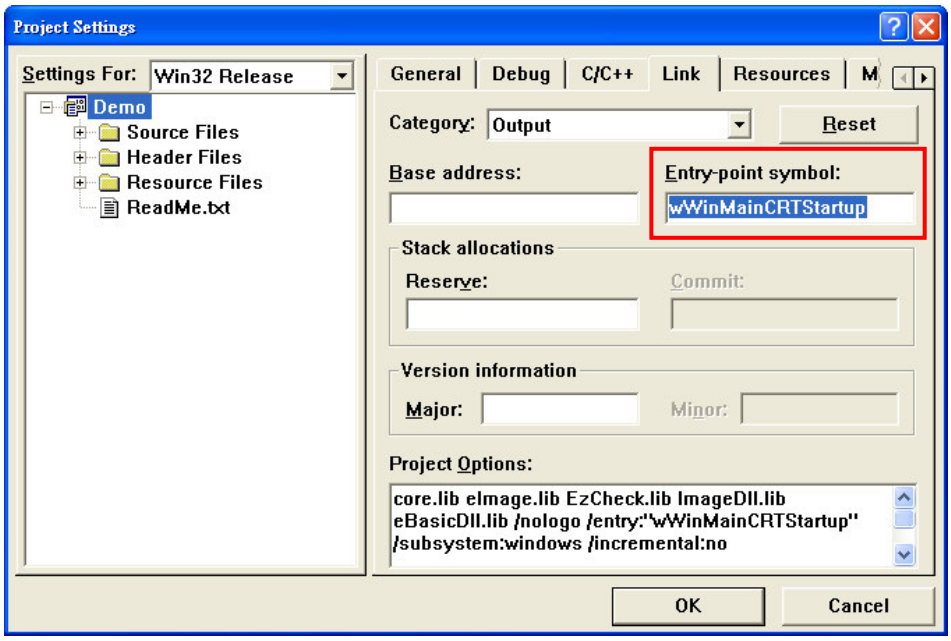
  Project->Setting->C/C++->Proprocessor definitions。加入 UNICODE 與 _UNICODE。



圖五十九、 Visual C++ 6.0 unicode 設定 1

➢ 設定 2：

Project->Setting->Link->Category 選擇 Output。於 Entry-point symbol 加入 wWinMainCRTStartup。



圖六十、 Visual C++ 6.0 unicode 設定 2

程式撰寫中的字串處理也必須遵照 unicode 的方式來進行。詳細內容可參考 C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Sample\Blob OCR 的範例程式。