EzCheck Vision Library
User Manual

Publication Jan, 2012
Ver. 1.0.0

EzCheck Vision Library

User Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Version Record

Version	Author	Date	Description
1.0.0	Clark Tsai	2012/1	Release

Tables of Content

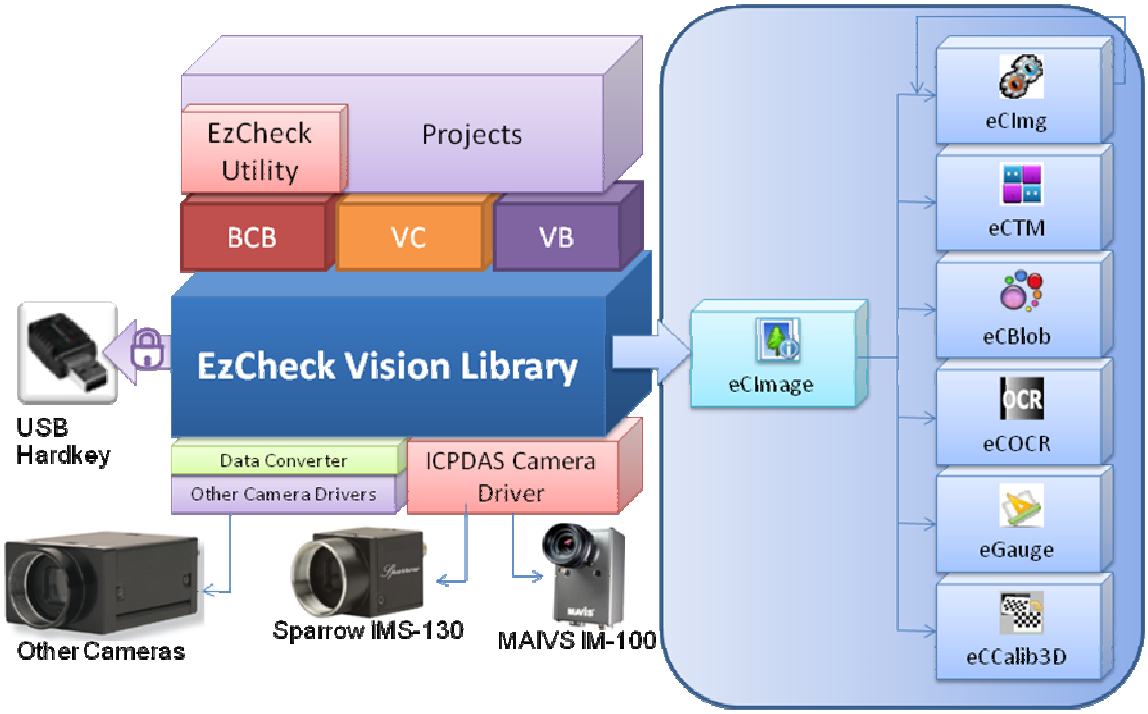
1. Introduction

1.1 Introduction

EzCheck Vision Library is an “Image processing and analysis library” provided by ICP DAS. It helps users to create their own machine vision system easily and fast.

EzCheck Vision Library supports several image file types and provides many basic image processing functions, and some useful advanced functions, such as OCR (Optical character recognition), Image Calibration, Template Matching, Gauge, etc.

Along with Utility and demo programs, Users may realize how to develop systems with EzCheck Vision Library.



1.2 Main Feature

- **Provide seven main components to help users develop their own machine vision system.**
  1. EzCheck Image：eCImage
  2. Image Processing：eCImg Group
  3. Template Matching：eCTM
  4. Blob Analysis：eCBlob
  5. Optical Character Recognition：eCOCR
  6. 3D Calibration：eCCalib3D
  7. Gauge/Measure：ePointGauge, eLinegauge, eCircleGauge, eRectangleGauge。
- **Provide utility software for users to test and verify – EzCheck Utility**
  Users can fully experience the capability of EzCheck Vision Library through the friendly utility software which is developed with EzCheck Vision Library.
- **USB Hardware Key protection and leveling package**
  To protect customer's rights, USB hard key is applied to EzCheck Vision Library to classify the packages of the library and manage the license. Moreover, ICP DAS also provides several packages of EzCheck Vision Library to fit different requirements of different users.

EzCheck Vision Library Grading Packages List						
Packages	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

- **Provide sample programs for multiple development platforms**

  By referencing the open source sample programs, users can learn how to develop applications by EzCheck Vision Library in short time.

  Sample programs of Borland C++ Builder 6.0 and Visual C++ 6.0 are now available.

- **Support the image capture of cameras**

  Users can capture the images of cameras through EzCheck Vision Library. The integration of software and hardware will be very efficient.

2. Use EzCheck Vision Library

In this chapter, users will learn to install and use EzCheck Vision Library step by step. You can read all the files on the case, including User Manual, EzCheck Utility, the necessity of Borland C++ Builder and Visual C++ project.

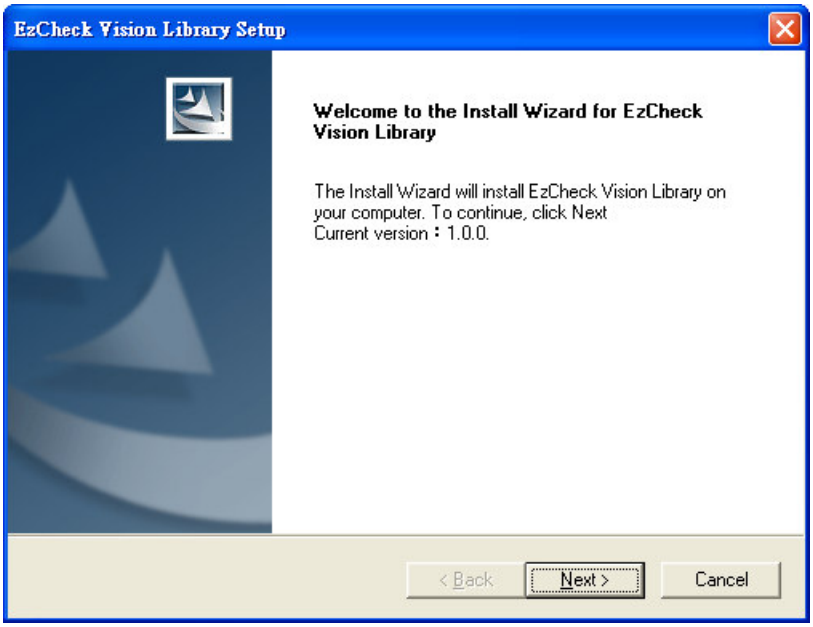
2.1 The installation of EzCheck Vision Library

Step1.Users can gain "EzCheck_Install_v*.*.*.exe" from web or CD. ("*" stands for the version number)

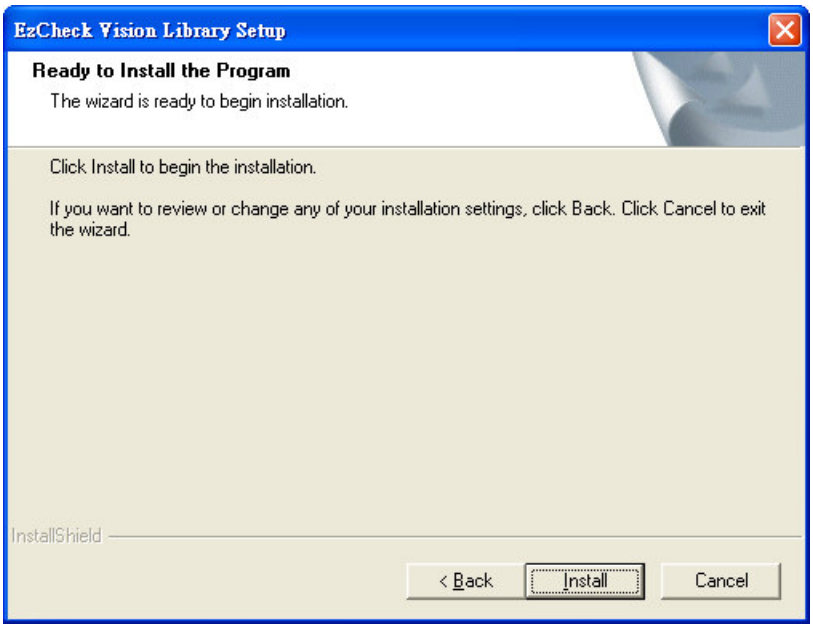
Web link:
http://www.icpdas.com.tw/product/solutions/software/development_tools/ezcheck/ezcheck_introduction.html

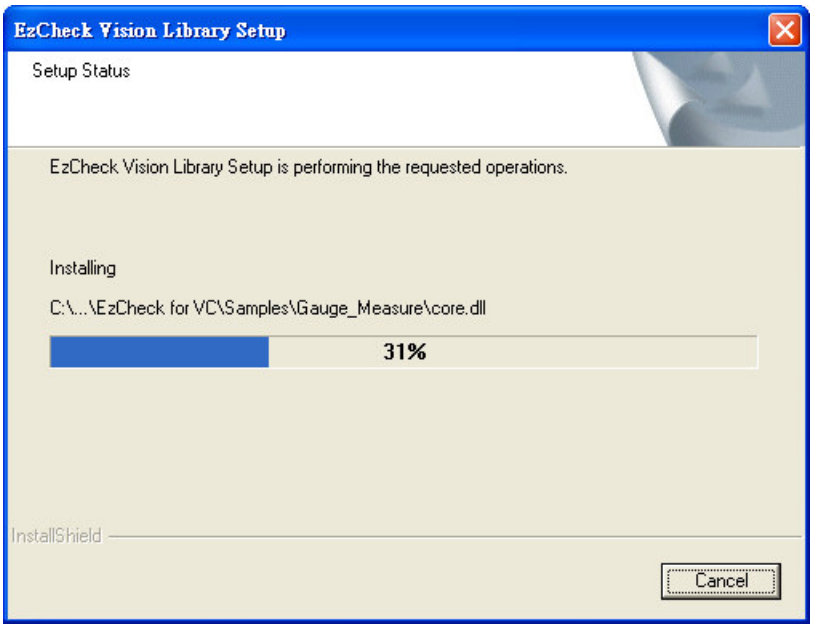
Download:
http://ftp.icpdas.com/pub/cd/EzCheck_cd/

Step2.Execute "EzCheck_Install_v*.*.*.exe" as shown below:

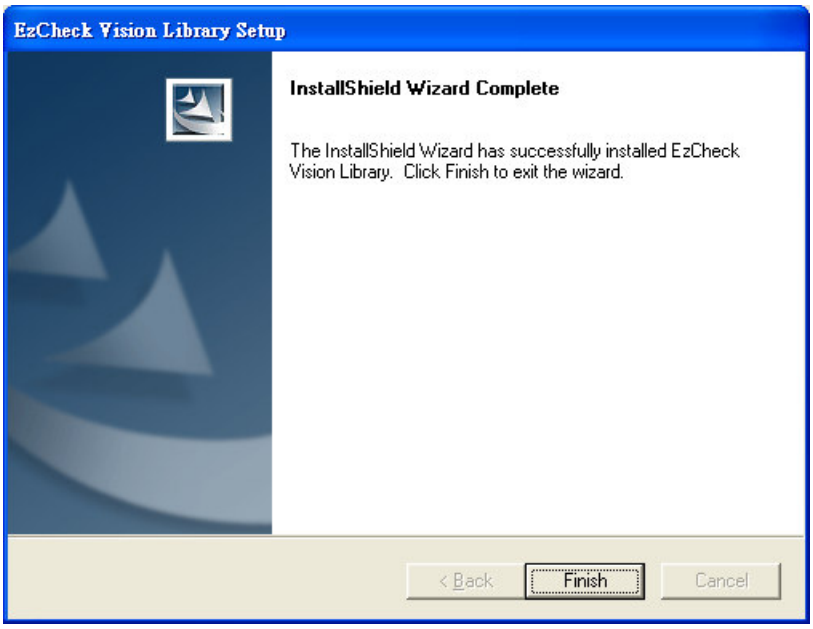


- Click “install”





- Click “Finish” to finish setup.



After installing “EzCheck_Install_v*.*.*.exe”, users can find all the files in C:\ICPDAS\EzCheck Vision Library.

The files of EzCheck Vision Library include EzCheck Utility, EzCheck Document, EzCheck for VC ,and EzCheck for BCB.

- EzCheck for BCB:
  The necessity of Borland C++ Builder:
  - Libs: The necessary LIB files for developing a project.
  - Dlls: The necessary DLL files for developing a project.
  - Includes: The necessary INCLUDE files for developing a project.
  - Sample: Sample programs for BCB.
- EzCheck fot VC:
  The necessity of Visual C++ 6.0 project:
  - Libs: The necessary LIB files for developing a project.
  - Dlls : The necessary DLL files for developing a project.
  - Include: The necessary INCLUDE files for developing a project.
  - Sample: Sample programs for VC.
- Ezcheck Document:
  Include User Manual and Quickstart.
- EzCheck Utility

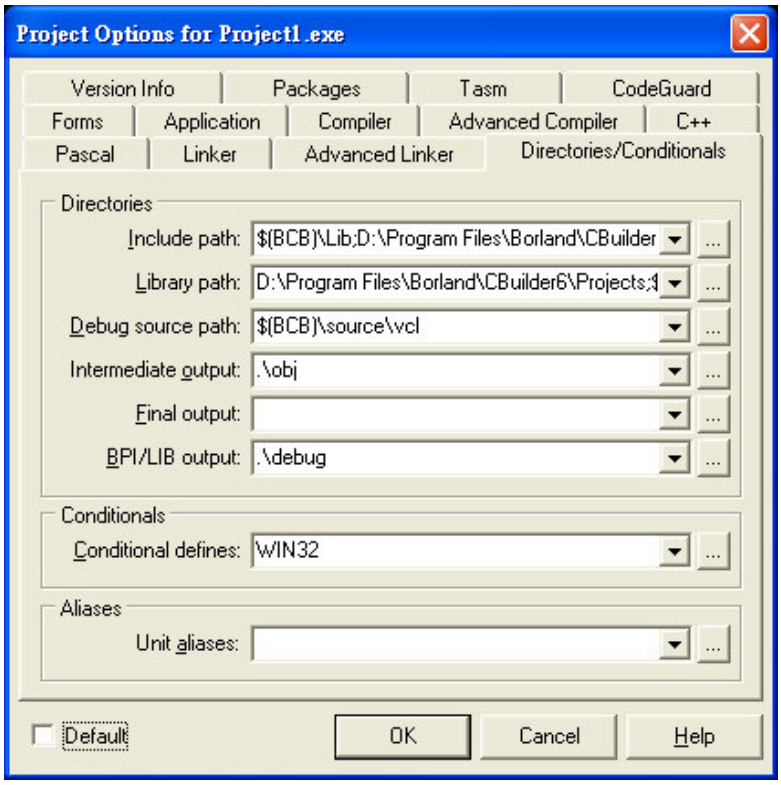
2.2 Operate EzCheck Vision Library in Borland C++ Builder 6.0

At this section, users can learn how to establish EzCheck Vision Library in Borland C++ Builder 6.0 with our step by step guide shown below.

Step1. Create a New Project

Step2. Set Your Path

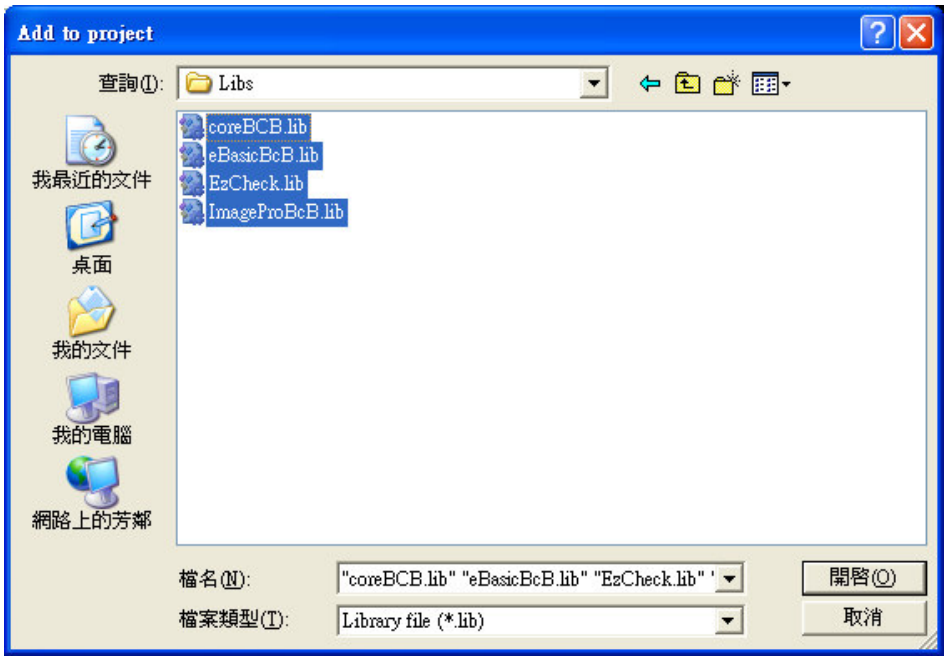
(Project→Options→Directories/Conditionals)



- Include path :
  C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Includes
- Library path :
  C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Libs

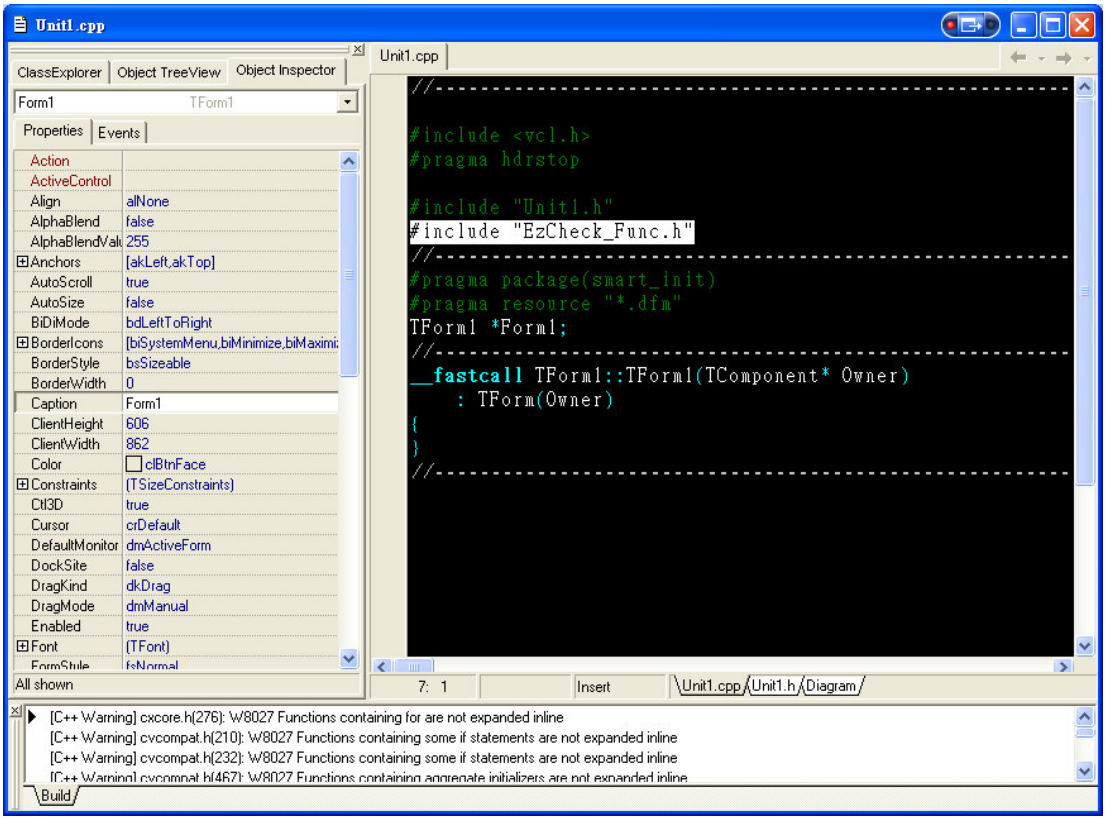
Step3. Add .lib files into the project

Project → Add to Project → C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Libs → Add Library files (*.lib) into the project.



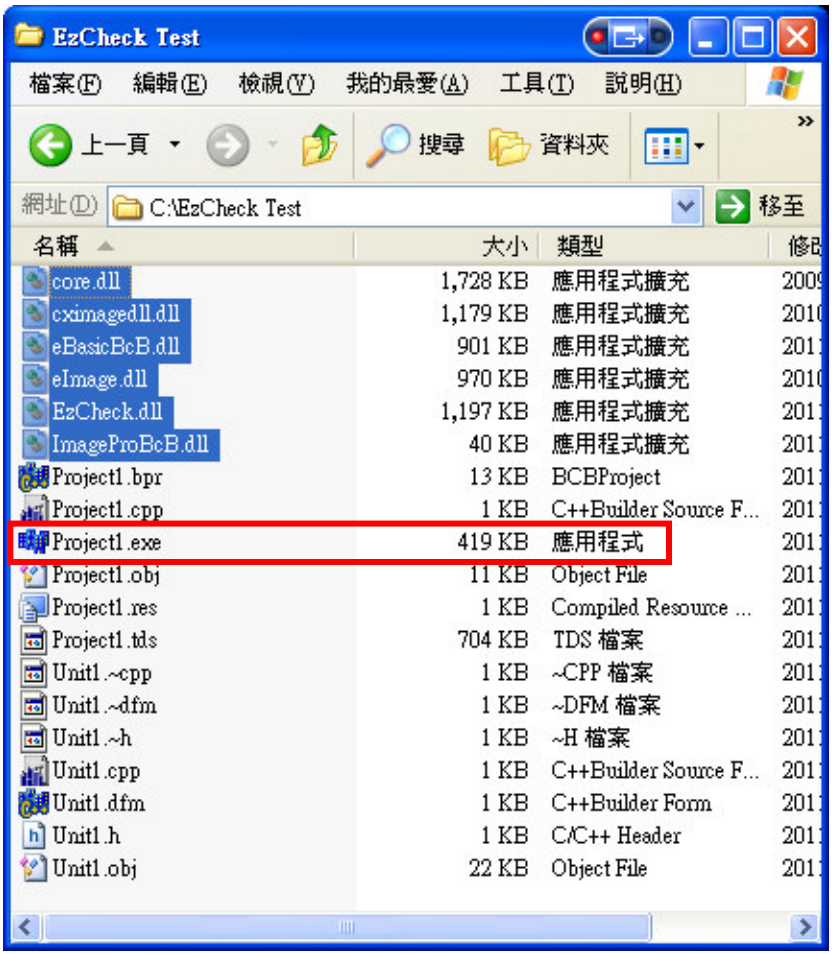
Step4. Enter the source code: #include "EzCheck_Func.h".

Include “EzCheck_Func.h” to link to all the functions of Ezcheck Vision Library.



Step5. Put the six .dll files and the .exe file of project in the same folder.

There are six .dll files in C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Dlls. Copy the six .dll files, and paste them in the same folder with .exe file of project.



Step6. Use EzCheck Vision Library to develop a machine vision system.

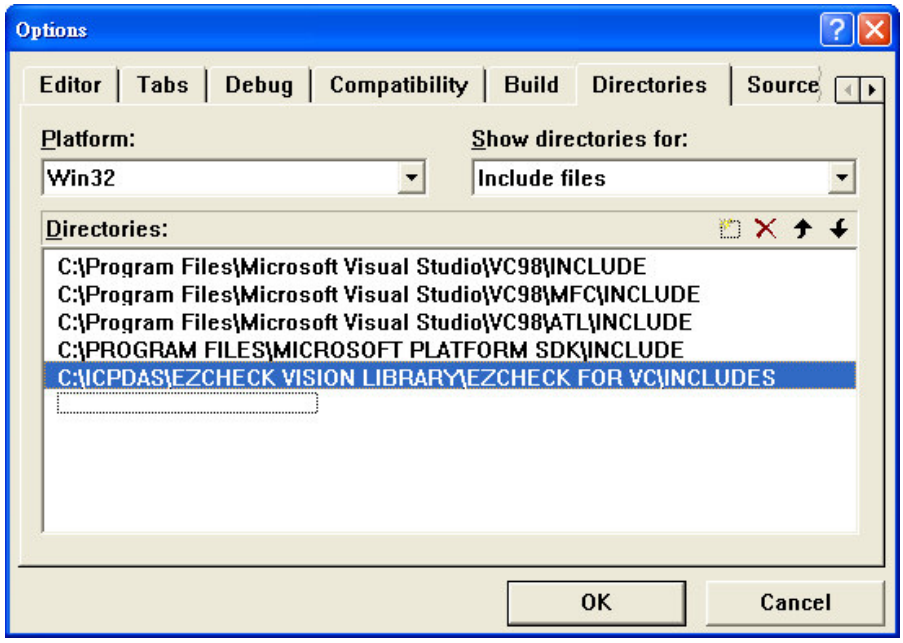
2.3 Operate EzCheck Vision Library in Visual C++ 6.0

At this section, users can learn how to establish EzCheck Vision Library in Visual C++ 6.0 with our step by step guide shown below.

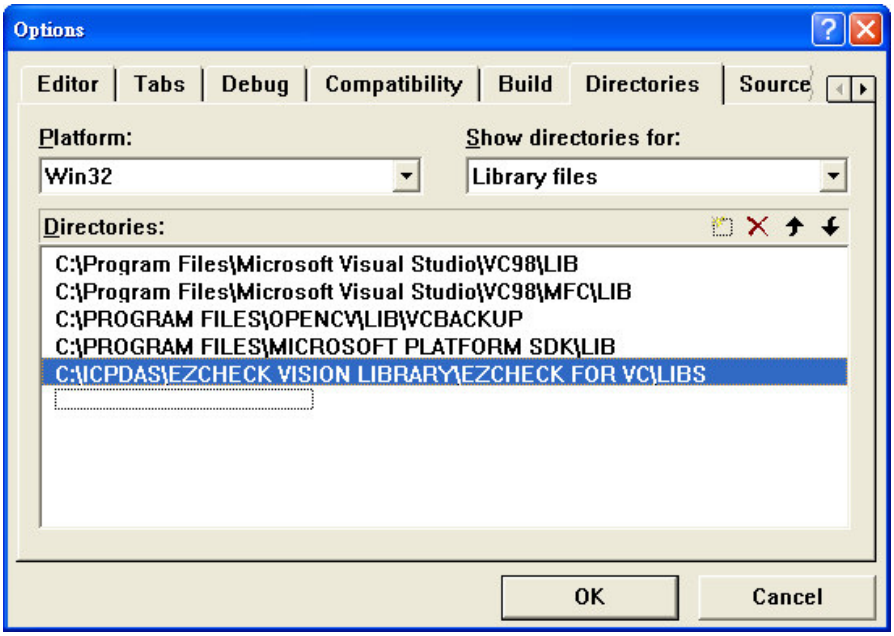
Step1. Create a New Project

Step2. Set Your Path

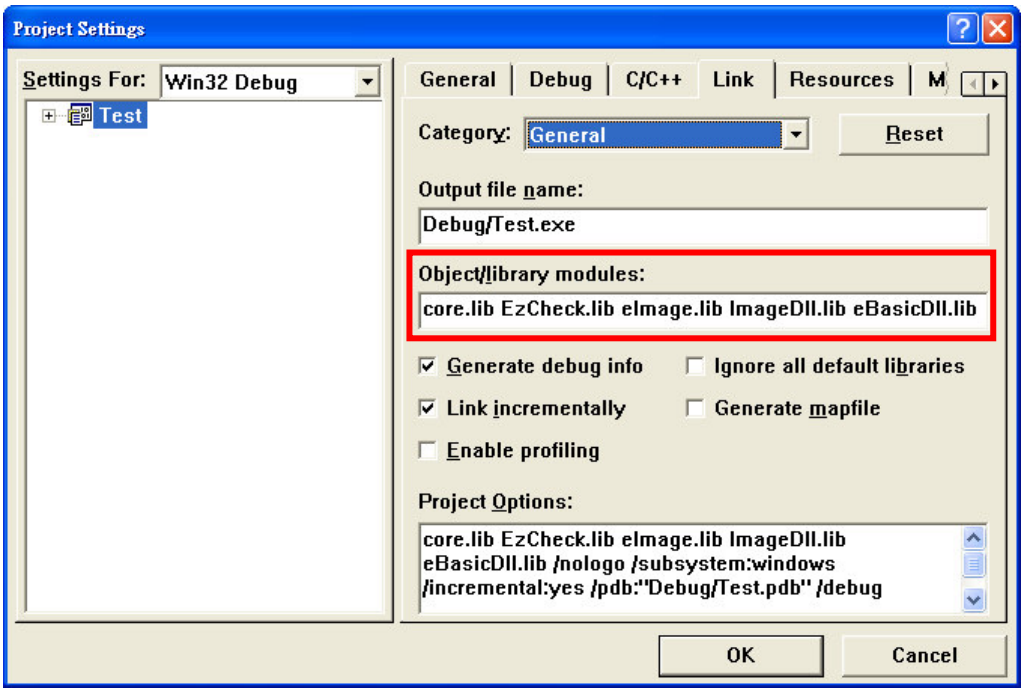
- The path for header files :
  Tools→Options→Directories→Include fillies→Add the path "C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Includes" into "Directories" of Library files



- The path for lib files :
  Tools→Options→Directories→Library files→Add the path "C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Libs" into the project.

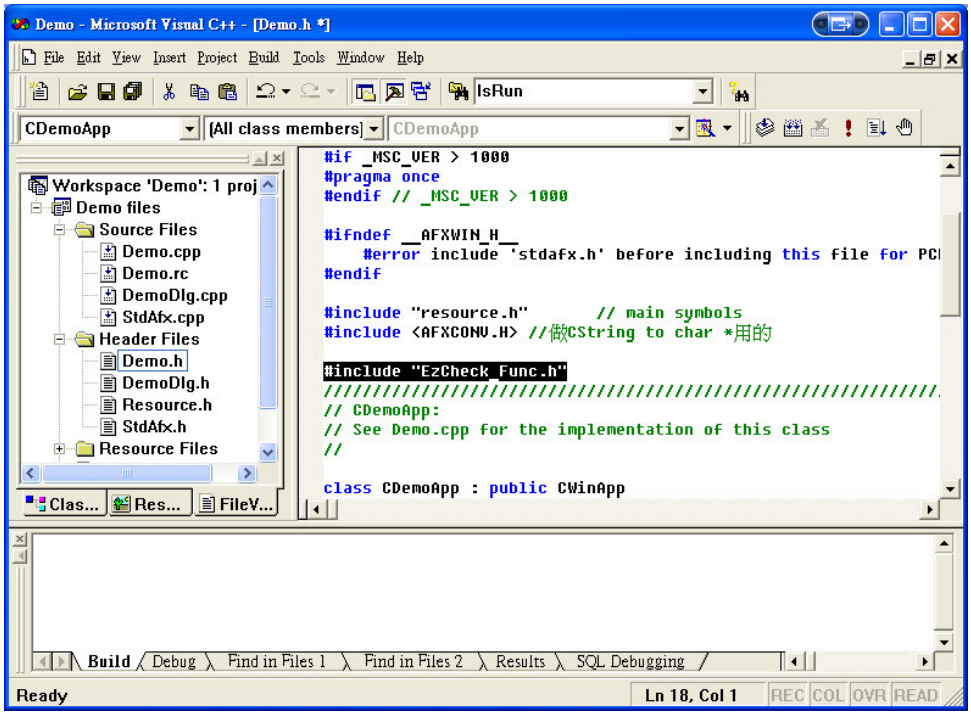


Step3. Add .lib files into the project
Project → Add to Project → C:\ICPDAS\EzCheck Vision Library\EzCheck for BCB\Libs → Add Library files (*.lib) into the project.



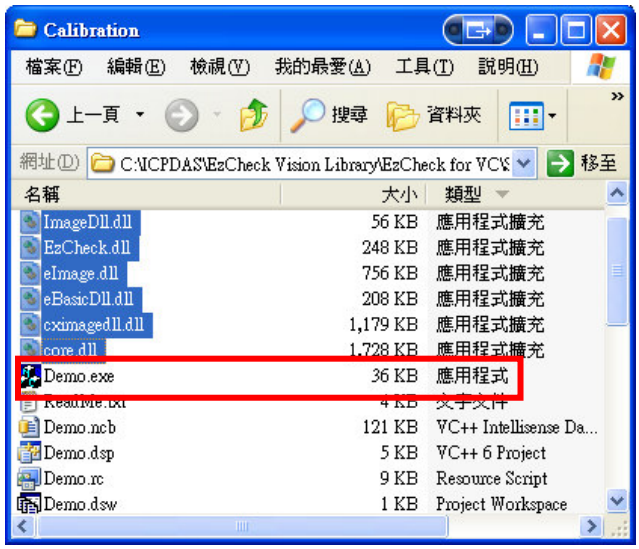
Step4. Enter the source code: #include "EzCheck_Func.h".

Include “EzCheck_Func.h” to link to all the functions of Ezcheck Vision Library.



Step5. Put the six .dll files and the .exe file of project in the same folder.

There are six .dll files in C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Dlls. Copy the six .dll files, and paste them in the same folder with .exe file of project.



Step6. Use EzCheck Vision Library to develop a machine vision system.

3. EzCheck Image (eCImage)

3.1 Main Features of eCImage

- Support Multi-format image file
  - Windows Bitmaps – BMP, DIB
  - JPEGE files – JPEG, JPG, JPE
  - TIFF files – TIFF, TIF
  - Portable Network Graphics – PNG

The eCImage supports Multi-format image files access, and offers a variety of information such as Color Channels, Image Size and PixelPoint. Users can set the ROI (region of interest) of the image they want to work on, or get the image in the ROI. Through eCImage, users can also capture the images from the camera which can be calibrated to many applications.

3.2 Main Functions of eCImage

3.2.1 eCImage

【Meaning】: Constructor.

☛ Function Prototype:

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

☛ Input Value:

n32Width	in	The width of the image.
n32Height	in	The height of the image.
bpp	in	Bit Depth. 8: Gray-Scale image; 24: Color Image.
origin	in	Assign the origin of the coordinates of the image, 0: top left-hand corner; 1: bottom left-hand corner

3.2.2 Create

【Meaning】: Create the Image Object.

☛ Function Prototype:

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

☛ Input Value:

n32Width	in	The width of the image.
n32Height	in	The height of the image.
bpp	in	Bit Depth. 8: Gray-Scale image; 24: Color Image.
origin	in	assign the origin of coordinates to image files, 0: top left-hand corner; 1: bottom left-hand corner

☛ Example Code:

```
eCImage *SrcImage= new eCImage; // Define a new Image Object.
SrcImage→Create(320, 240, 8, 0); // Create a 320*240 Gray-Scale image ; the origin of coordinates is the top left-hand corner.
```

Tips: We suggest users following the steps shown above to define and create the image objects. If users get their image NOT from Load file or eCImage copy, please take this way.

3.2.3 Release

【Meaning】: Release the Object.

☛ Function Prototype:

```
BOOL    Release();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

Tips: We suggest releasing all of the eCImage files after use to avoid wasting memory.

3.2.4 GetCopy

【Meaning】: Copy eCImage objects to get sourceObj, including ROI information.

☛ Function Prototype:

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

☛ Input Value:

sourceObj	in	the source of copy

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example Code:

```
eCImage *SrcImage= new eCImage;
eCImage *DstImage = new eCImage;
SrcImage→Load( “D:\\bitmap1.bmp” );
DstImage→GetCopy(SrcImage); //Copy. DstImage and SrcImage are independent and equivalent eCImage
```

objects.

3.2.5 GetGrayCopy

【Meaning】: Copy eCImage Gray-Scale image.

☛ Function Prototype:

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

☛ Input Value:

sourceObj	in	the source of copy

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example Code:

```
eCImage *SrcImage= new eCImage;
eCImage *DstImage = new eCImage;
SrcImage→Load( “D:\\bitmap1.bmp” );
DstImage→GetGrayCopy(SrcImage); // Get SrcImage Gray-Scale image.
```

Tips: The information of the destination image including Color Channels, Image Size, and ROI setting etc. will be equivalent to the source image in every kinds of copy method.

3.2.6 Load

【Meaning】: Open an image file.

☛ Function Prototype:

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

☛ Input Value:

pszPathName	in	The path and name of the file. Char or Unicode pointer.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

3.2.7 Save

【Meaning】: Save an image file.

☛ Function Prototype:

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

☛ Input Value:

pszPathName	in	The path and name of the file. Char or Unicode bit.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

3.2.8 GetWidth

【Meaning】: Get the width of the image.

☛ Function Prototype:

```
INT32    GetWidth();
```

☛ Return Value:

The width of the image

3.2.9 GetHeight

【Meaning】: Get the height of the image.

☛ Function Prototype:

```
INT32    GetHeight();
```

☛ Return Value:

The height of the image

3.2.10 GetSize

【Meaning】: Get the size of the image(image width*

image height).

☛ Function Prototype:

```
INT32    GetSize();
```

☛ Return Value:

Image Size. (The image size is the multiply value of height and width.)

3.2.11 GetPlanes

【Meaning】: Get the number of Channels: 1~4. Each stands for R,G, B, Alpha. Some file formats increase a 8-bit Alpha Channel to expand 24-bit image to 32-bit.

☛ Function Prototype:

```
INT32    GetPlanes();
```

☛ Return Value:

Channel number of image.

3.2.12 GetBitsPerRow

【Meaning】: Count how many bits per row. (image width*channels*bpp)

☛ Library:

```
INT32    GetBitsPerRow();
```

☛ Return Value:

The number of bits in a row.

3.2.13 GetBitsPerPixel

【Meaning】: Count how many bits per pixel. (channels*bpp)

☛ Function Prototype:

```
INT32    GetBitsPerPixel ();
```

☛ Return Value:

The number of bits in a pixel.

3.2.14 GetImagePtr

【Meaning】: Pointer of the Aligned Image Data. There is some color information of the image in the Aligned Image Data.

☛ Function Prototype:

```
INT32    GetImagePtr();
```

☛ Return Value:

Point at the index of the Aligned Image Data.

3.2.15 IsBlank

【Meaning】:Return the information that whether eCImage object has image data.

☛ Function Prototype:

```
bool    IsBlank();
```

☛ Return Value:

If the return value is TRUE, the contents of image are empty. If the return value is FALSE, the contents of image had recorded.

3.2.16 SetSize

【Meaning】: Set the image size by proportion or absolute size. For example, there is a 640*480 original image, and both of its x_scale and y_scale are 0.5, then we will get the 320*240 miniature. Owing to the change of size, the image will probably lack fidelity.

☛ Function Prototype:

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

☛ Input Value:

x_size	in	Set image width
y_size	in	Set image height
x_scale	in	Set the proportion of width
y_scale	in	Set the proportion of height

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

3.2.17 SetROI

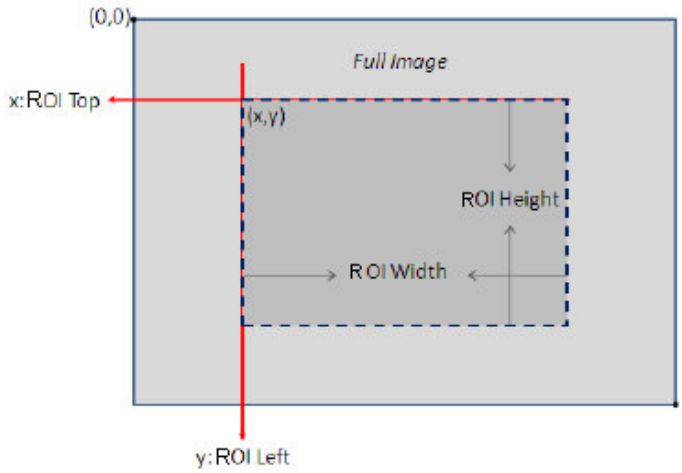
【Meaning】: Set the area of ROI.

There are two methods for setting ROI:
a. Set ROI by eCREC.
b. Set ROI by Original Point or width and height.

☛ eCREC:

```
typedef struct tag_REC
{
```

```
    INT32 org_x ;          // x-coordinate for Original Point
    INT32 org_y ;          // y-coordinate for Original Point
    INT32 width ;          //the width of rectangle
    INT32 height ;         // the height of rectangle
} eCREC;
```



☛ Function Prototype:

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

☛ Input Value:

rec	in	eCREC Data Structures Include the size and original point of ROI.
org_x, org_y	in	The original point of ROI.
Width, height	in	ROI size

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

Tips: The ROI of EzCheck Vision Library has no effect on

image's itself, and all actions process on the image had effect only in the ROI area. Most of the functions in EzCheck Vision Library support ROI operation.

3.2.18 ResetROI

【Meaning】: Release the ROI area. No more ROI information remains on the image

☛ Function Prototype:

```
BOOL    ResetROI();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

3.2.19 GetROI

【Meaning】: Get the ROI area in the image.

☛ Function Prototype:

```
eCREC    GetROI();
```

☛ Input Value:

sourceObj	in	the source of copy

☛ Return Value:

eCREC Data Structures
Please refer to the chapter of **eCREC**.

3.2.20 GetROIImage

【Meaning】: Get the ROI image from the source image.

☛ Function Prototype:

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

☛ Input Value:

DstImage	out	Target Image. Will get the ROI image from the source image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

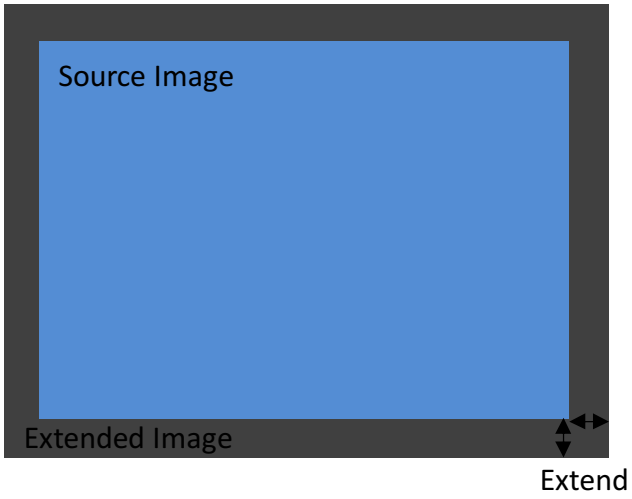
☛ Example Code:

```
eCImage *OriImage= new eCImage;
eCImage *ROIImage= new eCImage;
OriImage →Load( “D:\\bitmap1.bmp” );
OriImage→SetROI(100, 100, 100, 100);
OriImage→GetROIImage(ROIImage);    // Get the ROI image.
```

3.2.21 GetROIImage

【Meaning】: Get the ROI image from the source image.

Users don't have to set ROI in the source image but get the ROI image directly. If the source image is smaller than the acquired area, the system will fill the extended area with black (as shown below).



☛ Function Prototype:

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

☛ Input Value:

DstImage	out	Target Image Will get the ROI image from the image file.
ori_x, ori_y	in	The original point of ROI.
roi_width, roi_height	in	ROI size
Extend	in	The size of the extended erea.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return 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);    // Get the ROI image.
```

Tips:

EzCheck Vision Library support two methods for using ROI :

a. Use eCImage: “GetROIImage” takes the ROI area out, and makes it as an independent image.

b. Process the image directly after setting ROI. If the function supports ROI operation, users can operate or analyze in the ROI area.

Whether function can support ROI operation will be annotated in the user manual.

3.2.22 Draw

【Meaning】: Draw the eCImage objects.

☛ Function Prototype:

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

☛ Input Value:

hDC	out	the index of output mark
f32ZoomX	in	the display proportion of width
f32ZoomY	in	the display proportion of height

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

3.2.23 GetCameraImage

【Meaning】: Get the image buffer from the camera, and transfer the image format to deal with it.

※Support camera: ICPDAS Sparrow IMS-130, MAVIS IM-100,and MAVIS IM-30

☛ Function Prototype:

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

☛ Input Value:

VideoBuffer	in	The path and name of the file. Char or Unicode bit.

Width	in	Image width. (Must be equal to the data in the camera)
Height	in	Image Height. (Must be equal to the data in the camera)
bpp	in	the image buffer from the camera

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

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

【Meaning】: Get the color information about for the pixel.

Input the coordinates of the pixel, get the color information about blue, green, red, or Gray-Scale.

☛ Function Prototype:

```
INT32    GetPixelGray(int x, int y);
INT32    GetPixelRed(int x, int y);
INT32    GetPixelGreen(int x, int y);
INT32    GetPixelBlue(int x, int y);
```

☛ Input Value:

x, y	in	Pixel coordinates which will be taken data.

☛ Return Value:

Get the information about color for Pixel. The value is an integer from 0 to 255.

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

【Meaning】: Write in the color information of the Pixel.

Input the value and coordinates of the Pixel.

☛ Function Prototype:

```
BOOL    SetPixelGray(int x, int y, int value);
BOOL    SetPixelRed(int x, int y, int value);
BOOL    SetPixelGreen(int x, int y, int value);
BOOL    SetPixelBlue(int x, int y, int value);
```

☛ Input Value:

x, y	In	Pixel coordinates which will be taken data.
value	in	Pixel value from 0 to 255

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

3.2.26 GetErrorCode

【Meaning】: Get the ErrorCode returned from the eCImage.

☛ Function Prototype:

```
int    GetErrorCode();
```

☛ Return Value:

The ErrorCode that returned from the eCImage. Please refer to the **ErrorCode List**.

4. Image Processing Group — eCImg Group

4.1 Introduction to eCImg Group

Basic image processing functions contain Color Transform, Histogram, Threshold, Convolution filter, Morphology etc. Generally, we use those methods frequently when we deal with a picture, so they are classified as basic image processing functions in this article. And put the word "eCIma_" in the beginning of their name:

"eCImg Group" includes some groups as shown below:

- Threshold
- Color Channel
- Filter
- Morphology
- Rotation
- Histogram

Tips: Image processing has no good or bad, different image processing has different advantages and defects. Try a variety of methods to find the best way is very important.

4.2 Threshold

"EzCheck Vision Library" offers a variety of Threshold functions. The Threshold Value can be inputted by the users or counted by the system automatically according to the feature of the image.. According to the common situation, the Threshold offered by EzCheck Vision Library that set the Pixel greater than Threshold Value to 255 (White), and set the Pixel smaller than Threshold Value to 0 (Black).

Tip: All Threshold functions support ROI function.

4.2.1 eCImg_AbsoluteThreshold

【Meaning】: Do the Gray-Scale (Color) Threshold by the Threshold Value which assigned by users.

☛ Function Prototype:

```
BOOL    eCImg_AbsoluteThreshold(eCImage *SrcImage,
                                eCImage *&DstImage,
                                UINT32 threshold);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
threshold	in	The Threshold Value of the pixel-strength division.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.2.2 eCImg_RelativeThreshold

【Meaning】: Do the Gray-Scale (Color) Threshold by the proportion parameter which assigned by users.

☛ Function Prototype:

```
BOOL    eCImg_RelativeThreshold(eCImage *SrcImage,
                                eCImage *&DstImage,
                                FLOAT32 f32RelativeThreshold);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
threshold	in	The proportion of the pixel-strength division, from 0 to 1, the parameter stand for what

		percentage of pixel is set to black (0). The system will figure out the Threshold Value automatically. For example, if the proportion is 0.8, then the image will become 80% black and 20% white.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.2.3 eCImg_DoubleThreshold

【Meaning】: The Gray-Scale Triple-Threshold (0, 128, 255) was inputted by users. The Threshold Value 1 and 2 were must between 0 and 255, and users don't have to worry about that which one is greater or smaller. If the pixel value smaller than the smaller Threshold Value, then it will be set to 0; if the pixel value between them, then it will be set to 128; if the value greater than the greater Threshold Value, then it will be set to 255.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
threshold1	in	Threshold Value 1, from 0 to 255.
threshold2	in	Threshold Value 2, from 0 to 255.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.2.4 eCImg_MomentThreshold

【Meaning】: The Auto Threshold that can keep moment preserving balance. The Library counts the Threshold Value of the pixel automatically, and the Threshold Value can keep the Moment of the "SrcImage" and "DstImage" balance.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.2.5 eCImg_MinResidueThreshold

【Meaning】: The Auto Threshold that can make the Difference-of-square lowest. The Library counts the Threshold Value of the pixel automatically, and the Threshold Value can make the Difference-of-square of the "SrcImage" and "DstImage" lowest.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.2.6 eCImg_MaxEntropyThreshold

【Meaning】: The Threshold that makes the image's Entropy largest. The Library counts the Threshold Value of the pixel automatically, and the Threshold Value can make the Entropy of the "DstImage" largest.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.2.7 eCImg_IsodataThreshold

【Meaning】: The ISODATA(Iterative Self-Organizing Data Analysis Techniques Algorithm) Auto Threshold. The Library counts the Threshold Value of the pixel automatically, and the Threshold Value is between the average light gray values (i.e. gray levels above the threshold) and the average dark gray value (i.e. gray levels below the threshold).

☛ Function Prototype:

```
BOOL    eCImg_IsodataThreshold(eCImage *SrcImage,
```

```
                                                    eCImage *&DstImage);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3 Convolution Filter

EzCheck Vision Library provides multiform Convolution filter, contains edge detection, sharp, and smooth. Users don't have to learn the knowledge about dealing with images, just follow the instructions of library functions to find out the best to process the images.

Tips: All Convolution Filters support ROI function.

4.3.1 eCImg_ConvUniform

【Meaning】: It can make the image become uniform, and reduce the image noise. Because the size of mask must be odd numbers, the width and height inputted by users should be the half height of mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask. (height=half_height*2+1)

☛ Return Value:

If operate succeed, return TRUE; otherwise, return

FALSE.

4.3.2 eCImg_ConvGaussian

【Meaning】: A Gaussian filter, it can make the image

smooth, and reduce the image noise. Because the size of mask must be odd numbers, the width and height inputted by users should be the half height of mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask. (height=half_height*2+1)

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.3 eCImg_ConvHighPass1

【Meaning】: A high-pass filter, sharps the images but also makes the noise obvious. The function use 3 x 3 filter:

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

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.4 eCImg_ConvHighPass2

【Meaning】: A high-pass filter, sharps the images but also makes the noise obvious. The function use 3 x 3 filter:

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

☛ Function Prototype:

```
BOOL    eCImg_ConvHighPass2( eCImage *SrcImage,
                             eCImage *&DstImage );
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.5 eCImg_ConvHighPass3

【Meaning】: A high-pass filter, sharps the images but also makes the noise obvious. The function use 3 x 3 filter:

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

☛ Function Prototype:

```
BOOL    eCImg_ConvHighPass3(eCImage *SrcImage,
                            eCImage *&DstImage);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.6 eCImg_ConvLowPass1

【Meaning】: A low-pass filter, it can modify the high frequency part of the image to make it smooth. The function use 3 x 3 filter:

1	1	1
1	1	1
1	1	1

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.7 eCImg_ConvLowPass2

【Meaning】: A low-pass filter, it can modify the high frequency part of the image to make it smooth. The function use 3 x 3 filter:

1	1	1
1	0	1
1	1	1

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.8 eCImg_ConvLowPass3

【Meaning】: A low-pass filter, it can modify the high frequency part of the image to make it smooth. The function use 3 x 3 filter:

1	2	1
2	4	2
1	2	1

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.9 eCImg_ConvGradientX

【Meaning】: X-direction Gradient filter, the Gradient filter takes first derivatives of the image, and we can look upon it as an edge detection method. The function use 3 x 3 filter:

0	0	0
-1	0	1
0	0	0

- Function Prototype:

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

- Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

- Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.10 eCImg_ConvGradientY

【Meaning】: Y-direction Gradient filter, the Gradient filter takes first derivatives of the image, and we can look upon it as an edge detection method. The function use 3 x 3 filter:

0	-1	0
0	0	0
0	1	0

- Function Prototype:

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

- Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.11 eCImg_ConvGradient

【Meaning】: Gradient filter, it sums up the output of

x-direction Gradient filter and y-direction Gradient filter (absolute value).

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.12 eCImg_ConvPrewittX

【Meaning】: X-direction Prewitt filter, it takes first

derivatives of the image, and we can look upon it as an edge detection method. The function use 3 x 3 filter:

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

☛ Function Prototype:

```
BOOL    eCImg_ConvPrewittX(eCImage *SrcImage,
```

```
                                        eCImage *&DstImage );
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.13 eCImg_ConvPrewittY

【Meaning】: Y-direction Prewitt filter, it takes first derivatives of the image, and we can look upon it as an edge detection method. The function use 3 x 3 filter:

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

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.14 eCImg_ConvPrewitt

【Meaning】: Prewitt filter, it sums up the output of x-direction Prewitt filter and y-direction Prewitt filter (absolute value).

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.15 eCImg_ConvSobelX

【Meaning】: X-direction Sobel filter, it takes first derivative of the image, and we can look upon it as an edge detection method. The function use 3 x 3 filter:

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

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.16 eCImg_ConvSobelY

【Meaning】: Y-direction Sobel filter, it takes first

derivatives of the image, and we can look upon it as an edge detection method. The function use 3 x 3 filter:

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

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	DestinationImage.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.3.17 eCImg_ConvSobel

【Meaning】: Sobel filter, it sums up the output of

x-direction Sobel filter and y-direction Sobel filter (absolute value).

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4 Morphological Operations

"EzCheck Vision Library" offers a variety of Morphological Operations, includes image erosion, expansion, close and open, and provides square and circle operations. Users don't have to learn the knowledge about dealing with images, just follow the instructions of library function to find out the best library to deal with images.

Tips: All Morphological Operations support ROI function.

4.4.1 eCImg_Median

【Meaning】: Average value filter replaced gray scale pixel values (including the center pixel) by the average gray scale pixel value, and had smooth impression on image.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
size	in	Kernel size, it must be an odd number.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.2 eCImg_OpenBox

【Meaning】: Implement the Open Operation on the image by rectangular kernel.
DstImage= Open(SrcImage,kernel) = dilate(erode(SrcImage,kernel)kernel)
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask.
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.3 eCImg_OpenDisk

【Meaning】: Implement the Open Operation on the image by elliptic kernel.
DstImage= Open(SrcImage,kernel) = dilate(erode(SrcImage,kernel)kernel)
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.4 eCImg_CloseBox

【Meaning】: Implement the Close Operation on the image by rectangular kernel.
DstImage= Close(SrcImage,kernel) =
erode(dilate(SrcImage,kernel)kernel)
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask.
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.5 eCImg_CloseDisk

【Meaning】: Implement the Close Operation on the image by elliptic kernel.
DstImage= Close (SrcImage,kernel) = erode(dilate (SrcImage,kernel)kernel)
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.6 eCImg_ErodeBox

【Meaning】: Implement the Erode Operation on the image by rectangular kernel.
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if

users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask.
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.7 eCImg_ErodeDisk

【Meaning】: Implement the Erode Operation on the

image by elliptic kernel.
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.8 eCImg_DilateBox

【Meaning】: Implement the Dilate Operation on the image by rectangular kernel.
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask.

times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.9 eCImg_DilateDisk

【Meaning】: Implement the Dilate Operation on the image by elliptic kernel.
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.10 eCImg_MorphGradientBox

【Meaning】: Do the Morphology Gradient on the image by rectangular kernel.
DstImage= MorphGradient (SrcImage,kernel)
= dilate(SrcImage,kernel) - erode(SrcImage,kernel)
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size. For example, if users input (1, 1), then they will get 3 x 3 mask; if users input (2, 2), then they will get 5 x 5 mask; if users input (1, 2), then they will get 3 x 5 mask.

☛ Function Prototype:

```
BOOL   eCImg_MorphGradientBox(eCImage *SrcImage,
                              eCImage *DstImage,
                              UINT32 half_width,
                              UINT32 half_height,
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
half_height	in	The half height of mask.
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.4.11 eCImg_MorphGradientDisk

【Meaning】: Do the Morphology Gradient on the image by elliptic kernel.
Because the size of mask must be odd numbers, the width and height inputted by users should be the half height or the half width of the mask to get an odd number mask size.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
half_width	in	The half width of mask. (width=half_width*2+1)
times	in	The times of image erode and dilate.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5 Color Transform

EzCheck Vision Library provides Color Transform. Users can transform the color images to gray-scale images or divide into single color images; moreover, it can also get HIS image channel (Intensity, Saturation and Hues), the color knowledge closer to human.

4.5.1 eCImg_RGB2GRAY

【Meaning】: Get the gray-scale image of the RGB image.

If the image is a gray-scale image originally, then it will just get the copy of the SrcImage.
Formula:
Gray = 0.212671*R + 0.715160*G + 0.072169*B

☛ Function Prototype:

```
BOOL    eCImg_RGB2GRAY(eCImage *SrcImage,
                       eCImage *&DstImage);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5.2 eCImg_RGB2Red

【Meaning】: Get the red channel image of the RGB image.

If the image is a gray-scale image originally, then it will just get the copy of the SrcImage.

☛ Function Prototype:

```
BOOL    eCImg_RGB2Red(eCImage *SrcImage,
                      eCImage *&DstImage);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5.3 eCImg_RGB2Green

【Meaning】: Get the green channel image of the RGB image. If the image is a gray-scale image originally, then it will just get the copy of the SrcImage.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5.4 eCImg_RGB2Blue

【Meaning】: Get the blue channel image of the RGB image. If the image is a gray-scale image originally, then it will just get the copy of the SrcImage.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5.5 eCImg_RGB2HSI_H

【Meaning】: Get the Hue channel of the HIS image format, and it will be shown as 0~255. If the image is not a RGB image but a gray-scale image originally, then it will just get the copy of the SrcImage.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5.6 eCImg_RGB2HSI_S

【Meaning】: Get the Saturation channel of the HIS image format, and it will be shown as 0~255. If the image is not a RGB image but a gray-scale image originally, then it will just get the copy of the SrcImage.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.5.7 eCImg_RGB2HSI_I

【Meaning】: Get the Intensity channel of the HIS image format, and it will be shown as 0~255. If the image is not a RGB image but a gray-scale image originally, then it will just get the copy of the SrcImage.

☛ Function Prototype:

```
BOOL    eCImg_RGB2HSI_I(eCImage *SrcImage,
                        eCImage *&DstImage);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return

FALSE.

4.6 Histogram Operations

EzCheck Vision Library offers the statistics of the image color and intensity distribution, and the equalization of the histogram.

4.6.1 eCImg_Histogram

【Meaning】: The histogram shows the color intensity (the strength of a color) of an image. It figures out the amount of pixel in every channels and color intensity of all image.

☛ Function Prototype:

```
BOOL    eCImg_Histogram(eCImage *SrcImage,
                        int rHist[256],
                        int gHist[256],
                        int bHist[256] );
```

☛ Input Value:

SrcImage	in	Source Image.
rHist	out	It's the histogram of the red intensity, and the intensity range is from 0 to 255.
gHist	out	It's the histogram of the green intensity, and the intensity range is from 0 to 255.
bHist	out	It's the histogram of the blue intensity, and the intensity range is from 0 to 255.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.6.2 eCImg_Equalize

【Meaning】: Equalize the color and intensity of the image.

The Equalize function can increase image contrast. Generally, if you want to enhance the detail of the black part in the image, you can use this function.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.7 Image Rotation

EzCheck Vision Library provides four different characters Image Rotation functions. According to user requirements to decide whether the image change the size after rotation, and how to fill the empty part after rotation.

4.7.1 eCImg_Rotation1

【Meaning】: Rotate the image. After rotation, fill the empty part with black, and keep the original size.

As shown below:

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
FLOAT32	in	The angle of rotation.

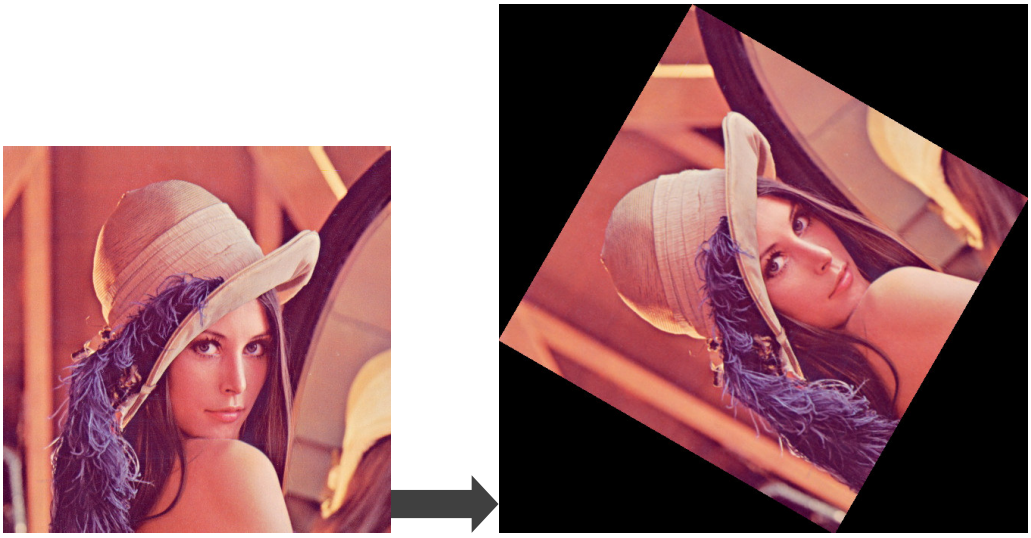
☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.7.2 eCImg_Rotation2

【Meaning】: Rotate the image. After rotation, fill the empty part with black, and change the size.

As shown below:



☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
FLOAT32	in	The angle of rotation.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.7.3 eCImg_Rotation3

【Meaning】: Rotate the image. After rotation, extend the edge of the image to fill the empty part., and keep the original size.

As shown below:

☛ Function Prototype:

```
BOOL    eCImg_Rotation3(eCImage *SrcImage,
                        eCImage *&DstImage,
                        FLOAT32 Angle);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
FLOAT32	in	The angle of rotation.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.7.4 eCImg_Rotation4

【Meaning】: Rotate the image. After rotation, extend the edge of the image to fill the empty part, and change the size.

As shown below:

☛ Function Prototype:

```
BOOL    eCImg_Rotation4(eCImage *SrcImage,
                        eCImage *&DstImage,
                        FLOAT32 Angle);
```

☛ Input Value:

SrcImage	in	Source Image.
DstImage	out	Destination Image.
FLOAT32	in	The angle of rotation.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

4.8 Get ErrorCode

We may take some mistakes no matter the input of images or set the parameter to cause the failure in the library operation. It can output the Error Code for error checking and correction, when the operation underperform or return FALSE.

4.8.1 eCImg_GetErrorCode

【Meaning】: Get the error code returned from the last image processing.

☛ Function Prototype:

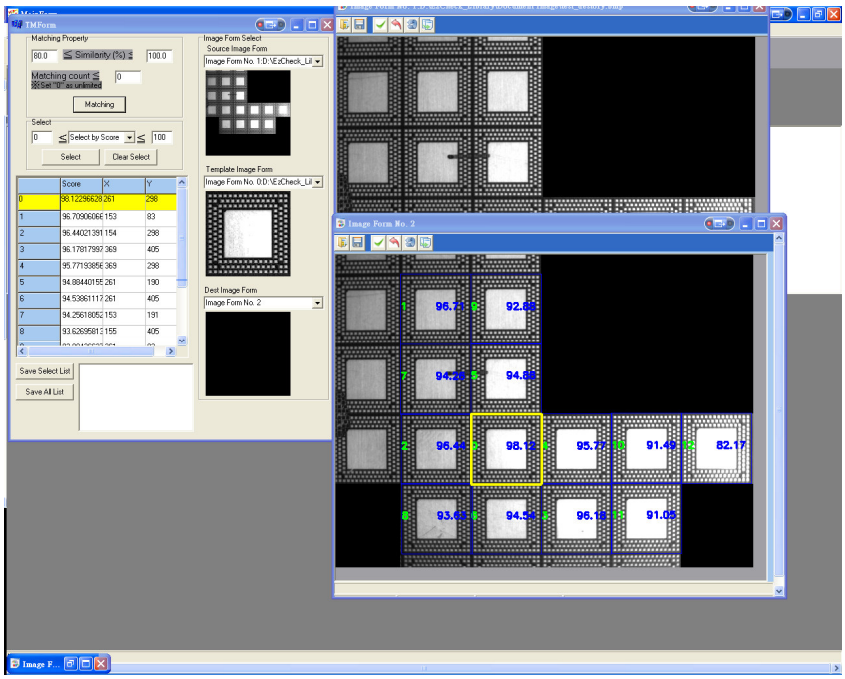
```
INT32    eCImg_GetErrorCode();
```

☛ Return Value:

The ErrorCode returned from the processing.
Please refer to the **ErrorCode List**.

5. Template Matching – eCTM

The eCTM is a gray-level template matching tool which can search similar matching areas of a specific template on the image. Users can assign an image as template, search for its matching areas on another image, and then obtain the location and similarity of the "matchings". Through the built-in functions Users can get images with matching information marked on it.

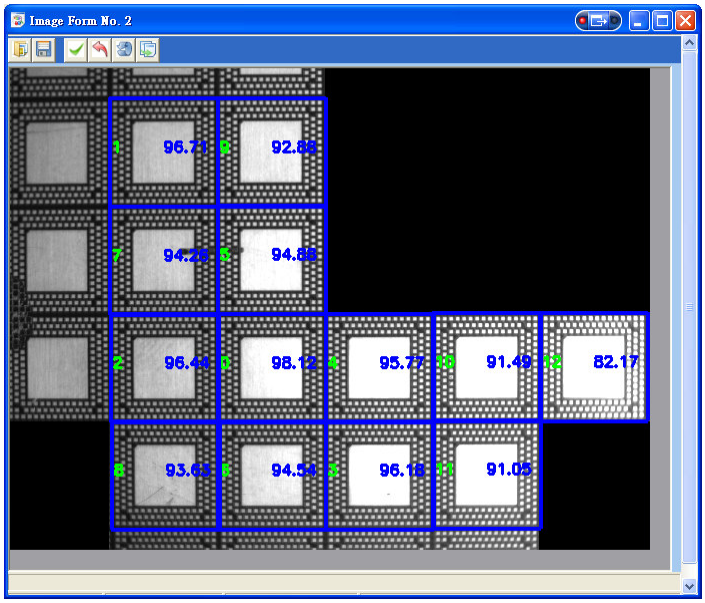


5.1 Main Features of eCTM

- **The matching is based on "Correlation coefficient matching" algorithm. Users can search the matching areas which have assigned similarity range.**
  Users can assign the similarity range before matching to reduce unnecessary information with low similarity.
- **The matching results can be sorted and selected by similarity or coordinates.**
  Every matching result contains similarity and coordinates. Users can sort and select the matching results to get necessary information.
- **Output the matching results into the text files.**
  Users can output the matching results into the text files for later validation and check.
- **Output the image with similarity value and rank marking**

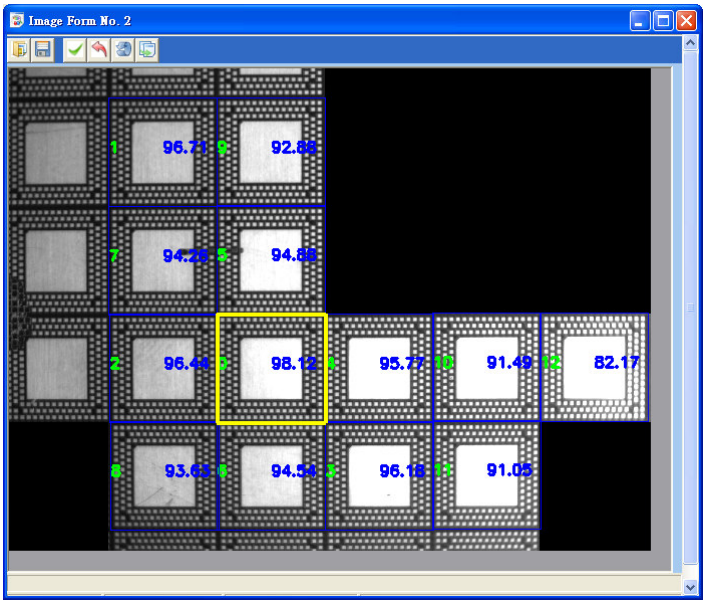
on it.

Through the eCTM, Users can get the image marked with matching results, such as the similarity value and rank as well as the location of similar sections on it.



❖ **Output the image with single matching marking on it.**

Through the eCTM, Users can output the image with markings of single matching result such as frame effects, similarity value and similarity place on it.



5.2 Main Functions of eCTM

5.2.1 struct _MATCH

【Meaning】:The eCTM matching results.

☛ Function Prototype:

```
struct _MATCH
{
    int x, y;
    double Score;
}
```

x,y	in	The central point of the similar area.
Score	in	Similarity.

5.2.2 SelectMatching

【Meaning】: The structure array that saves the matching results. Users can easily access the data. All siftings and sorts are outputted from here.

☛ Function Prototype:

```
_MATCH *SelectMatching
```

5.2.3 eCTM

【Meaning】: Constructor.

☛ Function Prototype:

```
eCTM ( );
```

5.2.4 Release

【Meaning】: Release the resources of the eCTM. There are some necessary data structures in the eCTM, we advise

users release the eCTM after using.

☛ Function Prototype:

```
BOOL    Release( );
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.5 LoadTemplateImage

【Meaning】: Load an image file and assign it to be the template image.

☛ Function Prototype:

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

☛ Input Value:

pszPathName	in	The path and name of the file.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.6 SetTemplateImage

【Meaning】: Assign an eCImage to be the template image.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	The template's source image.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.7 DoTemplateMatching

【Meaning】: Do the template matching. It's the core function of the eCTM. User must assign the template image before process.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	The source image DoTemplateMatching will match the image with template to find out the similar area on it.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return 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; // Get most similar area.
```

5.2.8 SetProperty

【Meaning】: Set the property of matching to encourage the matching speed and avoid the unnecessary data.

☛ Function Prototype:

```
BOOL    SetProperty(float Min,
                    float Max,
```

```
int MaxCount);
```

☛ Input Value:

Min	in	The minimum similarity of matching (0.0~1.0). If the similarity is smaller than the minimum, then it won't be recorded.
Max	in	The maximum similarity of matching (0.0~1.0). If the similarity is larger than the maximum, then it won't be recorded.
MaxCount	in	The maximum of recording. Get the MaxCount begin at maximum.

- The default setting of Min: 0.8
- The default setting of Max: 1.0
- The default setting of MaxCount: 100

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example code:

```
eCTM* TM_Sample = new eCTM;
TM_Sample->SetProperty(0.6, 1.0, 10);// Get the similarity area information that they are top 10 and 60%~100% similarity.
```

5.2.9 GetPropertyMin

【Meaning】: Get the minimum similarity setting of Property.

☛ Function Prototype:

```
flaot    GetPropertyMin();
```

☛ Return Value:

The minimum similarity of the matching and it will respond with SetProperty.

5.2.10 GetPropertyMax

【Meaning】: Get the maximum similarity setting of Property.

☛ Function Prototype:

```
flaot    GetPropertyMax();
```

☛ Return Value:

The maximum similarity of the matching and it will respond with SetProperty.

5.2.11 GetPropertyMaxCount

【Meaning】: Get the highest recorded number of the matching.

☛ Function Prototype:

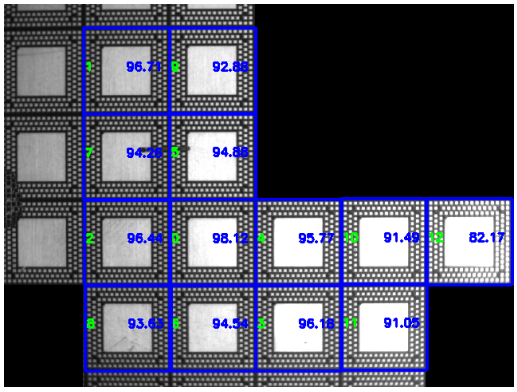
```
flaot    GetPropertyMaxCount();
```

☛ Return Value:

Return the highest recorded number of the matching and it will respond with SetProperty.

5.2.12 GetAllMatchingImage

【Meaning】: Get the result image with all similar areas marked on it. The outcome as shown below:



☛ Function Prototype:

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

☛ Input Value:

DstImage	out	Receive the eCImage of the all matching outcome.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example code

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

5.2.13 GetSelectMatchingImage

【Meaning】: Get the result image with selected similar areas marked on it. Before selecting, the outcome looks the same as GetAllMatchingImage.

☛ Function Prototype:

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

☛ Input Value:

DstImage	out	Receive the eCImage of the all matching outcome.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.14 GetSingleMatchingImage

【Meaning】: Get the result image with a specific similar area marked on it.

☛ Function Prototype:

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

☛ Input Value:

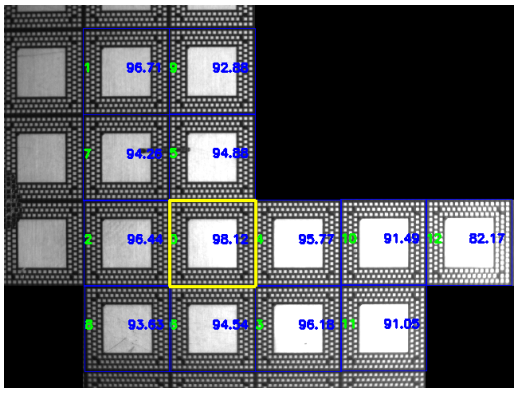
DstImage	out	Receive the eCImage of the all matching outcome.
Number	in	The serial numbers of matching output.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.15 GetRemarkMatchingImage

【Meaning】: Get the image emphasized the single similar area. The outcome as shown below:



☛ Function Prototype:

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

☛ Input Value:

DstImage	out	Receive the eCImage of the all matching outcome.
Number	in	The serial numbers of matching output.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

Tips: All Get***Image of eCTM were not return correct image unless complete DoTemplateMatching correctly.

5.2.16 SelectMatchByX

【Meaning】: Select similar areas output by x-coordinate, and get the number of outcome.

☛ Function Prototype:

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

☛ Input Value:

MaxValue	in	The ceiling of x-coordinate select. Ignore the similarity area that is larger than this numeric.
MinValue	in	The minimum of x-coordinate select. Ignore the similarity area that is smaller than this numeric.

☛ Return Value:

The number of similarity area after selecting.

5.2.17 SelectMatchByY

【Meaning】: Select similar areas output by y-coordinate, and get the number of outcome.

☛ Function Prototype:

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

☛ Input Value:

MaxValue	in	The ceiling of y-coordinate select. Ignore the similarity area that is larger than this numeric.
MinValue	in	The minimum of y-coordinate select. Ignore the similarity area that is smaller than this

		numeric.

☛ Return Value:

The number of similarity area after selecting.

5.2.18 SelectMatchByScore

【Meaning】: Select the similar areas output by similarity, and get the number of which is selected.

☛ Function Prototype:

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

☛ Input Value:

MaxValue	in	The maximum similarity of selecting. Ignore the similarity area that is larger than this numeric.
MinValue	in	The minimum similarity of selecting. Ignore the similarity area that is smaller than this numeric.

☛ Return Value:

The number of similarity area after selecting.

5.2.19 ClearSelect

【Meaning】: Reset the similar areas output to DoTemplateMatching initial similar areas output.

☛ Function Prototype:

```
BOOL    ClearSelect();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.20 SortMatch

【Meaning】: Sort the similar areas results. Users can choose the type and order of sort.

- Function Prototype:

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

- Input Value:

SortType	in	The type of sort. 0: similarity, 1: x-coordinate, 2: y-coordinate
Order	in	The order of sort. 0: from large to small, 1: from small to large

- Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.21 SaveAllMatchingList

【Meaning】: Save all similar areas results to document file.

- Function Prototype:

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

- Input Value:

FileName	in	File's name.

- Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.22 SaveSelectMatchingList

【Meaning】: Save selected similar area results to

document file.

- Function Prototype:

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

- Input Value:

FileName	in	File's name.

- Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

5.2.23 GetErrorCode

【Meaning】: Get the error code of eCTM.

- Function Prototype:

```
INT32    GetErrorCode ();
```

- Input Value:

FileName	in	File's name.

- Return Value:

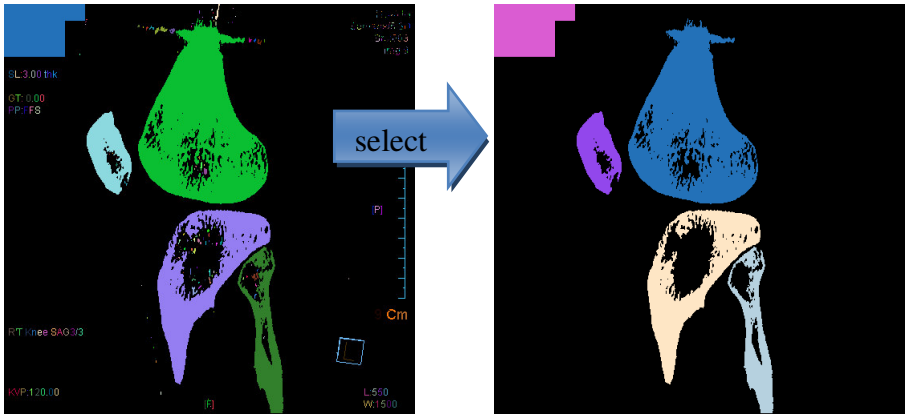
The ErrorCode returned from the processing. Please refer to the **ErrorCode List**.

6. Blob Analysis — eCBlob

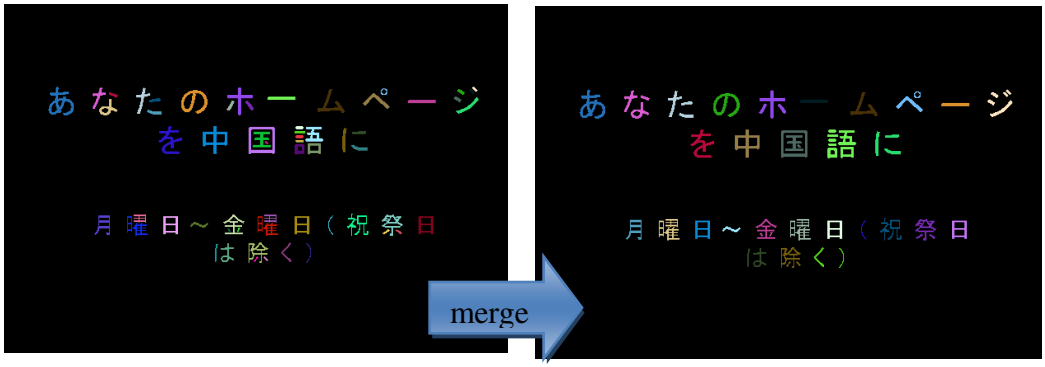
The eCBlob is a blob analysis library that can analyze isolated areas and connected areas which we called them “blob” on the image. Users can sort or select blob information such as location, gravity center, measure of area, etc, and even merge the blobs nearby each other.

6.1 Main Features of eCBlob

- **Analyze the blobs on the image and mark them by different color.**
  Each blob will be marked by unique color while the non-blob areas are shown in black color.
- **Support the selection and sorting to the result of Blob Analysis**
  Every blob contains information such as measure of area and location, etc. Users can sort and select the blob by those features to get rid of unnecessary information.



- **Support the function of merging nearby blobs**
  Except selecting the blobs, Users can also merge them together. It is very important to merge the blobs before OCR which will be introduced later.



❖ **Get image of connected or selected blobs**
Users can output the images with connected or selected blobs which can be useful to other application and validation.

6.2 Main Functions of eCBlob

The main functions of eCBlob include analyze the blobs on the image, get the information about the blobs, and turn the blobs to image file etc..Users can through select or merge to analyze all blobs on the image, and get the optional subclass--Select Blob. The eCBlob supports processing both Blob and Select Blob by individually functions.

6.2.1 eCBlob

【Meaning】: Constructor.

☛ Function Prototype:

```
eCBlob();
```

6.2.2 ~eCBlob

【Meaning】: Destructor.

☛ Function Prototype:

```
~eCBlob();
```

6.2.3 DoBlobAnalysis

【Meaning】:Do blob analysis for the source image.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	The source image that will be done blob analysis.
Class Type	in	Analysis the type of blob. 1: Black blob with white background. 2: White blob with black background. 3. Black-and-White blob. 4: Darker blob. 5: Well situated brightness. 6: Brighter blob.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example code:

```
eCImage *SrcImage = new eCImage;
eCBlob Blob;//Define a new eCBlob Object.
SrcImage->Load("C:\\Test_Image.bmp");
eCImg_AbsoluteThreshold(SrcImage, SrcImage,
100);//Alter the SrcImage directly, and the Threshold
Value was set 100.
Blob.DoBlobAnalysis(SrcImage, 1);// Do blob analysis for
the black blob with white background of SrcImage.
```

Tips: We advise doing Threshold before doing DoBlobAnalysis, and use the white blob with black background or black blob with white background to get the better output.

6.2.4 SelectBlobUsingFeature

【Meaning】: Select blob after **DoBolbAnalysis**. The functions about **Select Blob** will be available after finishing **SelectBlobUsingFeature**.

☛ Function Prototype:

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

☛ Input Value:

FeatureType	in	Select features. Please refer to **SELECTFEATURE**.
OptionType	in	Blob left after selecting: 1: Larger than MinTHValue 2. Smaller than MaxTHValue 3. Between MaxTHValue and MinTHValue. 4. Larger than MaxTHValue and smaller than MinTHValue.
MinTHValue	in	Select feature values. Please refer to **OptionType**.
MaxTHValue	in	Select feature values. Please refer to **OptionType**.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example code:

```
//After DoBlobAnalysis
Blob.SelectBlobUsingFeature(1, 1, 200, 0);//Get the blob that the area larger than 200.
```

6.2.5 SortBlobUsingFeature

【Meaning】: Sort blobs after DoBlobAnalysis.

☛ Function Prototype:

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

☛ Input Value:

FeatureType	in	The features of sort. Please refer to **SELECTFEATURE**.
SortType	in	The kinds of sort. 1: Ascending order. 2: Descending order. 3: The initial order.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example code:

```
//After DoBlobAnalysis
Blob. SortBlobUsingFeature (1, 1); // Ascending order by the area value.
```

6.2.6 GetBlobParameter

【Meaning】: Get the blob analysis parameter after **DoBlobAnalysis**.

☛ Function Prototype:

```
PBA_ROI    GetBlobParameter ();
```

☛ Return Value:

PBA_ROI. The information of Blob analysis. Please refer to **BLOBANALYSIS TAG**.

6.2.7 CalculateAdvancedFeature

【Meaning】: Calculate the advanced features of blob after **DoBlobAnalysis**.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	The reference image of calculate. Please input the image the same as DoBlobAnalysis.
FeatureType	in	The advanced feature that will be calculated. Please refer to **SELECTFEATURE** for the advanced feature.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.8 GetBlobBasicFeature

【Meaning】: Get the **basic features** of blob after **DoBlobAnalysis**. The definition of **basic features** are in the DataFile.h.

☛ Function Prototype:

```
PBLOB_BASICFEATURE    GetBlobBasicFeature ();
```

☛ Return Value:

PBLOB_BASICFEATURE. Record the basic feature of blob. Please refer to **_BLOBBASICFEATURE_TAG**.

☛ Example code:

```
//After DoBlobAnalysis.
BLOB_BASICFEATURE *sB; // Define the structure
```

```
array that save the blob's basic features.
sB = Blob. GetBlobBasicFeature ();
int Blob_Area = sB[0].Area; // Get the no.0 blob's
measure of area.
```

6.2.9 GetBlobAdvancedFeature

【Meaning】: Get the **advanced features** of blob after **DoBlobAnalysis** and **CalculateAdvancedFeature**. The definition of **advanced features** are in the DataFile.h.

☛ Function Prototype:

```
PBLOB_ADVANCEDFEATURE    GetBlobAdvancedFeature ();
```

☛ Return Value:

PBLOB_ADVANCEDFEATURE. Record the basic feature of blob. Please refer to **_BLOBADVANCEDFEATURE_TAG**.

☛ Example code:

```
//After DoBlobAnalysis.
BLOB_ADVANCEDFEATURE *sA; //Define the
structure array that save the blob's advanced features.
sA = Blob. GetBlobAdvancedFeature ();
FLOAT32 Blob_SigmaX = sB[0]. SigmaX; // Get No.0
blob's SigmaX value.
```

6.2.10 GetBlobConvexHull

【Meaning】: Get all blob's **Convex hull** information after **DoBlobAnalysis**. The definition of **Convex hull** is in the DataFile.h.

☛ Function Prototype:

```
PConvex_RESULT    GetBlobConvexHull ();
```

☛ Return Value:

PConvex_RESULT. Save the blob's **Convex hull** information. Please refer to **CONVEXHULL _TAG**.

6.2.11 GetSelectBlobParameter

【Meaning】: Get the information of **select blob** after **DoBlobAnalysis** and **SelectBlobUsingFeature**.

☛ Function Prototype:

```
PBA_ROI    GetSelectBlobParameter ();
```

☛ Return Value:

PBA_ROI. Get the select blob's information. Please refer to **_BLOBANALYSIS_TAG**.

6.2.12 CalculateSelectBlobAdvancedFeature

【Meaning】: Calculate the advanced information of select blob after **DoBlobAnalysis** and **SelectBlobUsingFeature**.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	The reference image of calculate. Please input the image the same as DoBlobAnalysis.
FeatureType	in	The advanced feature that will be calculated. Please refer to **SELECTFEATURE**.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.13 GetSelectBlobBasicFeature

【Meaning】: Get the all **basic features** of **select blob** after **DoBlobAnalysis** and **SelectBlobUsingFeature**.

☛ Function Prototype:

```
PBLOB_BASICFEATURE  GetSelectBlobBasicFeature();
```

☛ Return Value:

PBLOB_BASICFEATURE. The basic feature of Blob please refer to **_BLOBBASICFEATURE_TAG**.

☛ Example code:

```
//After DoBlobAnalysis and SelectBlobUsingFeature.
BLOB_BASICFEATURE *sB;// Announce the structure array that save the blob's basic feature.
sB = Blob. GetSelectBlobBasicFeature ();
int Blob_Area = sB[0].Area; // Get the no.0 blob's measure of area.
```

6.2.14 GetSelectBlobAdvancedFeature

【Meaning】: Get the all advanced features of blob after **DoBlobAnalysis**, **SelectBlobUsingFeature** and **CalculateAdvancedFeature**. The definition of advanced features are in the DataFile.h.

☛ Function Prototype:

```
PBLOB_ADVANCEDFEATURE GetSelectBlobAdvancedFeature();
```

☛ Return Value:

PBLOB_ADVANCEDFEATURE. The advanced features of Blob please refer to **_BLOBADVANCEDFEATURE_TAG**.

☛ Example code:

```
//After DoBlobAnalysis, SelectBlobUsingFeature, and
```

```
CalculateSelectBlobAdvancedFeature.
BLOB_BASICFEATURE *sB;// Announce the structure array that save the blob's basic feature.
sB = Blob. GetSelectBlobBasicFeature ();
int Blob_Area = sB[0].Area; // Get the no.0 select blob's SigmaX.
```

6.2.15 GetSelectBlobConvexHull

【Meaning】: Get the information about **Convex hull** of **SelectBlob** after **DoBlobAnalysis** and **SelectBlobFeature**. The definition of **Convex hull** is in the DataFile.h.

☛ Function Prototype:

```
PConvex_RESULT    GetSelectBlobConvexHull();
```

☛ Return Value:

PConvex_RESULT. The Convex hull information of Blob please refer to **_CONVEXHULL_TAG**.

6.2.16 SaveBlobImage/ SaveSelectBlobImage

【Meaning】: Save the image file that was marked **Blob** or **Select Blob** by different color in each area.

☛ Function Prototype:

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);
```

☛ Input Value:

FileName	in	The path and name of file.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.17 SaveSingleBlob/ SaveSingleSelectBlob

【Meaning】: Save the image file that has single Blob/Select Blob. There are two ways to assign the blob, coordinates and number, the axis is absolute coordinate, and it has two kinds of outputs to choose, Threshold and Gray-scale image. If save succeed, return TRUE; otherwise, return FALSE.

☛ Function Prototype:

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);
```

☛ Input Value:

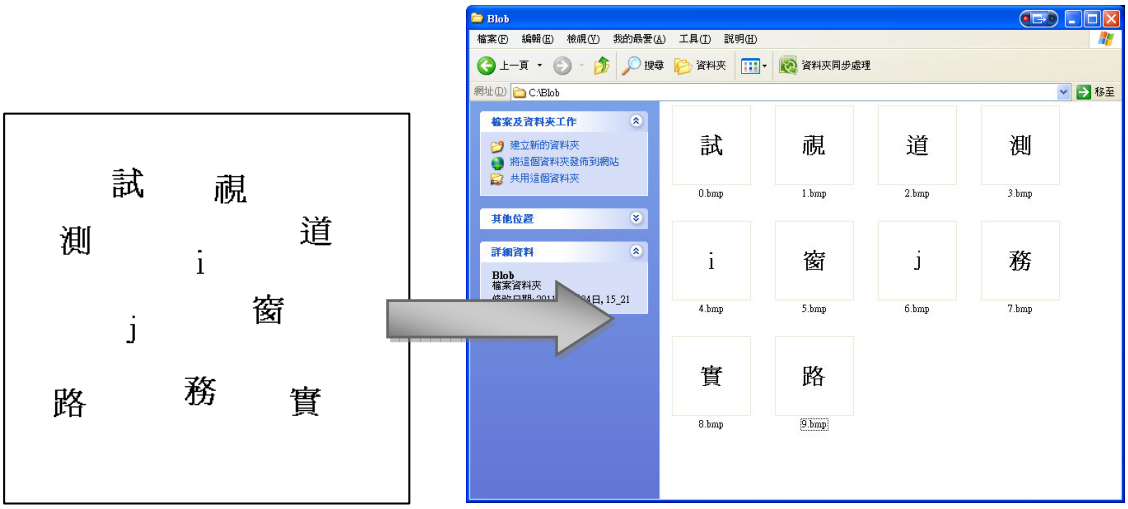
SrcImage	in	Source Image. Please input the image the same as blob analysis or the image before Threshold.
X_axis, Y_axis	in	Blob's absolute coordinates. Please input the coordinates within the range of blob.
NstBlob	in	Blob number.
SaveName	in	The path and name of file.
ProcMode	in	The output image is black-and-white or gray-scale. Please refer to **PROCESSMODE**.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.18 SaveAllBlob/SaveAllSelectBlob

【Meaning】: Save the image files of every single blob/select blob individually. Users can choose Threshold or Gray-scale image to output and save. The filename is supplied as their number. If save succeed, return TRUE; otherwise, return FALSE. As shown below:



☛ Function Prototype:

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);
```

☛ Input Value:

SrcImage	in	Source Image. Please input the image the same as blob analysis or not Threshold yet.
SavePath	in	Save path.
ImageType	in	Vice filename. Suggest format: ".bmp".
ProcMode	in	The output image is black-and-white or gray-scale. Please refer to **PROCESSMODE**.

Tips: SaveBlobImage function can help users to take blob's image directly. It will be useful while users need to transfer the blob images to OCR database.

6.2.19 AutoMerge

【Meaning】: Merge nearby blobs automatically. If operate succeed, return TRUE; otherwise, return FALSE.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image. Please input the image the same as blob analysis.
SetWidth	in	Merge blob width.

SetHeight	in	Merge blob height.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.20 Merge

【Meaning】: Merge the blobs that were inputted from Merge List.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image. Please input the image the same as blob analysis.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.21 AddMergeList

【Meaning】: Users can use blob's number or coordinates to add the blobs into Merge List for merging them together. If operate succeed, return TRUE; otherwise, return FALSE. The blobs in the Merge List will be merged by eCBlob::Merge form.

☛ Function Prototype:

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

☛ Input Value:

NstBlob	in	Blob's number.
X_axis, Y_axis	in	Blob's coordinates.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.22 CleanMergeList

【Meaning】: Clean the contents of Merge List

☛ Function Prototype:

```
BOOL    CleanMergeList ();
```

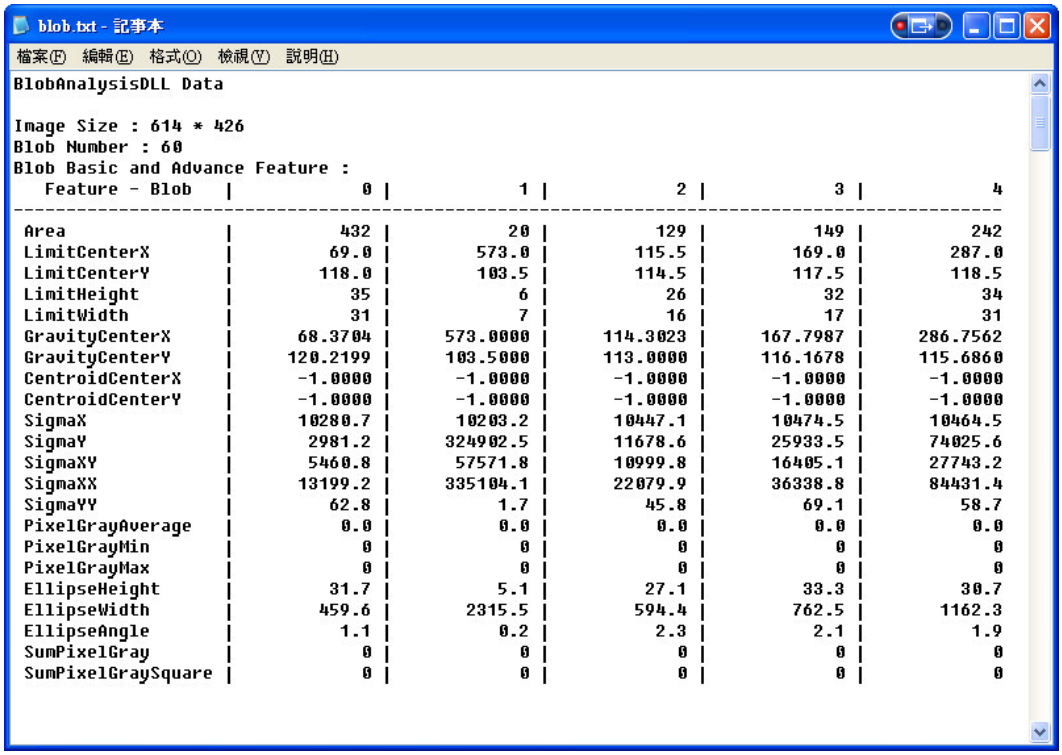
☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

Tips: Through relational functions such as GetSelectBlobBasicFeature and SaveSelectBlobImage are also can get the merge output of Merge function.

6.2.23 SaveTxt

【Meaning】: Save blob information to text file. Users can type in filename by themselves or the system save the BlobAnalysisDLL.txt in current folder automatically. As shown below:



blob.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
BlobAnalysisDLL Data

Image Size : 614 * 426
Blob Number : 60
Blob Basic and Advance Feature :
   Feature - Blob   |           0 |           1 |           2 |           3 |           4
-------------------------------------------------------------------------------------------
 Area               |         432 |          20 |         129 |         149 |         242
 LimitCenterX       |        69.0 |       573.0 |       115.5 |       169.0 |       287.0
 LimitCenterY       |       118.0 |       103.5 |       114.5 |       117.5 |       118.5
 LimitHeight        |          35 |           6 |          26 |          32 |          34
 LimitWidth         |          31 |           7 |          16 |          17 |          31
 GravityCenterX     |     68.3704 |    573.0000 |    114.3023 |    167.7987 |    286.7562
 GravityCenterY     |    120.2199 |    103.5000 |    113.0000 |    116.1678 |    115.6860
 CentroidCenterX    |     -1.0000 |     -1.0000 |     -1.0000 |     -1.0000 |     -1.0000
 CentroidCenterY    |     -1.0000 |     -1.0000 |     -1.0000 |     -1.0000 |     -1.0000
 SigmaX             |     10280.7 |     10203.2 |     10447.1 |     10474.5 |     10464.5
 SigmaY             |      2981.2 |    324902.5 |     11678.6 |     25933.5 |     74025.6
 SigmaXY            |      5460.8 |     57571.8 |     10999.8 |     16405.1 |     27743.2
 SigmaXX            |     13199.2 |    335104.1 |     22079.9 |     36338.8 |     84431.4
 SigmaYY            |        62.8 |         1.7 |        45.8 |        69.1 |        58.7
 PixelGrayAverage   |         0.0 |         0.0 |         0.0 |         0.0 |         0.0
 PixelGrayMin       |           0 |           0 |           0 |           0 |           0
 PixelGrayMax       |           0 |           0 |           0 |           0 |           0
 EllipseHeight      |        31.7 |         5.1 |        27.1 |        33.3 |        30.7
 EllipseWidth       |       459.6 |      2315.5 |       594.4 |       762.5 |      1162.3
 EllipseAngle       |         1.1 |         0.2 |         2.3 |         2.1 |         1.9
 SumPixelGray       |           0 |           0 |           0 |           0 |           0
 SumPixelGraySquare |           0 |           0 |           0 |           0 |           0
```

☛ Function Prototype:

Borland C++ Builder version:

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

Visual C++ version:

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

☛ Input Value:

TxtName	in	The path name of text file.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.24 CleanBuffer

【Meaning】: Clean all information about blob analysis. If operate succeed, return TRUE; otherwise, return FALSE.

☛ Function Prototype:

```
BOOL    CleanBuffer();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.25 CleanSelectBuffer

【Meaning】: Reset select blob data to original situation.

☛ Function Prototype:

```
BOOL    CleanSelectBuffer();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.26 SingleBlobFree

【Meaning】: Users can eliminate the specific blob by blob's number or coordinates.

☛ Function Prototype:

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

☛ Input Value:

NstBlob	in	Blob's number.
X_axis, Y_axis	in	Blob's coordinates.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

6.2.27 GetErrorCode

【Meaning】: Get the error code from eCBlob. Please refer to **ErrorCode List**.

☛ Function Prototype:

```
INT32    GetErrorCode ();
```

☛ Return Value:

The error code returned from eCBlob processing. Please refer to **ErrorCode List**.

6.3 eCBlob Data Structures

The data structures were used in eCBlob are shown as below. The relational data were announced in DataFile.h.

6.3.1 struct _BLOBANALYSIS_TAG

【Meaning】: Blob analysis information.

- The contents of announcement:

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

- Contents introduction:

StartX	The beginning of x-axis.
StartY	The beginning of y-axis.
Width	Blob analysis width.
Height	Blob analysis height.
Count	The total number of blob analysis.

6.3.2 struct _BLOBBASICFEATURE_TAG

【Meaning】: Blob's basic feature.

☛ The contents of announcement:

```
struct _BLOBBASICFEATURE_TAG
{
    UINT32    Area;
    FLOAT32    LimitCenterX;
    FLOAT32    LimitCenterY;
    UINT16    LimitHeight;
    UINT16    LimitWidth;
    FLOAT32    GravityCenterX;
    FLOAT32    GravityCenterY;
    FLOAT32    CentroidCenterX;
    FLOAT32    CentroidCenterY;
    FLOAT32    RectangleXY[4][2];
};
```

☛ Contents introduction:

Area	The square measure of blob.
LimitCenterX	The x-axis of blob's limit center.
LimitCenterY	The y-axis of blob's limit center.
LimitHeight	The height of limit center.
LimitWidth	The width of limit center.
GravityCenterX	The x-axis of blob's gravity center.
GravityCenterY	The y-axis of blob's gravity center.
CentroidCenterX	The x-axis of blob's centroid center.
CentroidCenterY	The y-axis of blob's centroid center.
RectangleXY[4][2]	The apex of rectangle around the blob.

Tips:
Limit Center: The 0, 22, 45, and 68 degree frame range.
Gravity Center: The center of the smallest circumscribed circle.
Centroid Center: The average of pixel axes.

6.3.3 struct _BLOBADVANCEDFEATURE_TAG

【Meaning】: Blob's advanced feature.

☛ The contents of announcement:

```
struct _BLOBADVANCEDFEATURE_TAG
{
    FLOAT32   SigmaX;
    FLOAT32   SigmaY;
    FLOAT32   SigmaXY;
    FLOAT32   SigmaXX;
    FLOAT32   SigmaYY;
    FLOAT32   PixelGrayAverage;
    UINT8      PixelGrayMin;
    UINT8      PixelGrayMax;
    FLOAT32   EllipseHeight;
    FLOAT32   EllipseWidth;
    FLOAT32   EllipseAngle;
    UINT32     SumPixelGray;
    UINT32     SumPixelGraySquare;
};
```

☛ Contents introduction:

SigmaX	Ellipse of Inertia. (parameter D)
SigmaY	Ellipse of Inertia. (parameter E)
SigmaXY	Ellipse of Inertia. (parameter 2B)
SigmaXX	Ellipse of Inertia. (parameter A)
SigmaYY	Ellipse of Inertia. (parameter C)
PixelGrayAverage	The average of grayscale pixel values.
PixelGrayMin	The minimum grayscale value.
PixelGrayMax	The maximum grayscale value.
EllipseHeight	Inertia Ellipse Height.
EllipseWidth	Inertia Ellipse Width.
EllipseAngle	The angle of Inertia Ellipse.
SumPixelGray	The sum of grayscale pixel values.

SumPixelGraySquare	The sum of square of grayscale pixel values.

Tips:
The circumscribed circle of inertia ellipse looks upon the limit center as its center.
Equation: $Ax^2 + 2Bxy + Cy^2 + Dx + Ey = 0$

6.3.4 struct _CONVEXHULL_TAG

【Meaning】: The Convex hull information of Blob.

☛ The contents of announcement:

```
struct _CONVEXHULL_TAG
{
    UINT16 **X;
     UINT16 **Y;
    UINT8 *ConvexCount;
};
```

☛ Contents introduction:

X	The blob's x-axis after Convex hull.
Y	The blob's y-axis after Convex hull.
ConvexCount	The numbers of blob after Convex hull.

Tips: The Convex Hull is a polygon that around the object and has smallest square measure.

6.4 eCBlob Enumeration Types

The enumeration types were used in eCBlob are shown as below.

6.4.1 enum PROCESSMODE

【Meaning】: The process mode of Blob.

☛ The contents of announcement:

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

☛ Contents introduction:

Binary_Mode	Set as black-and-white mode.
Gray_Mode	Set as grayscale mode.

6.4.2 enum SELECTFEATURE

【Meaning】: The features were used to select blobs.

☛ The contents of announcement:

```
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,
```

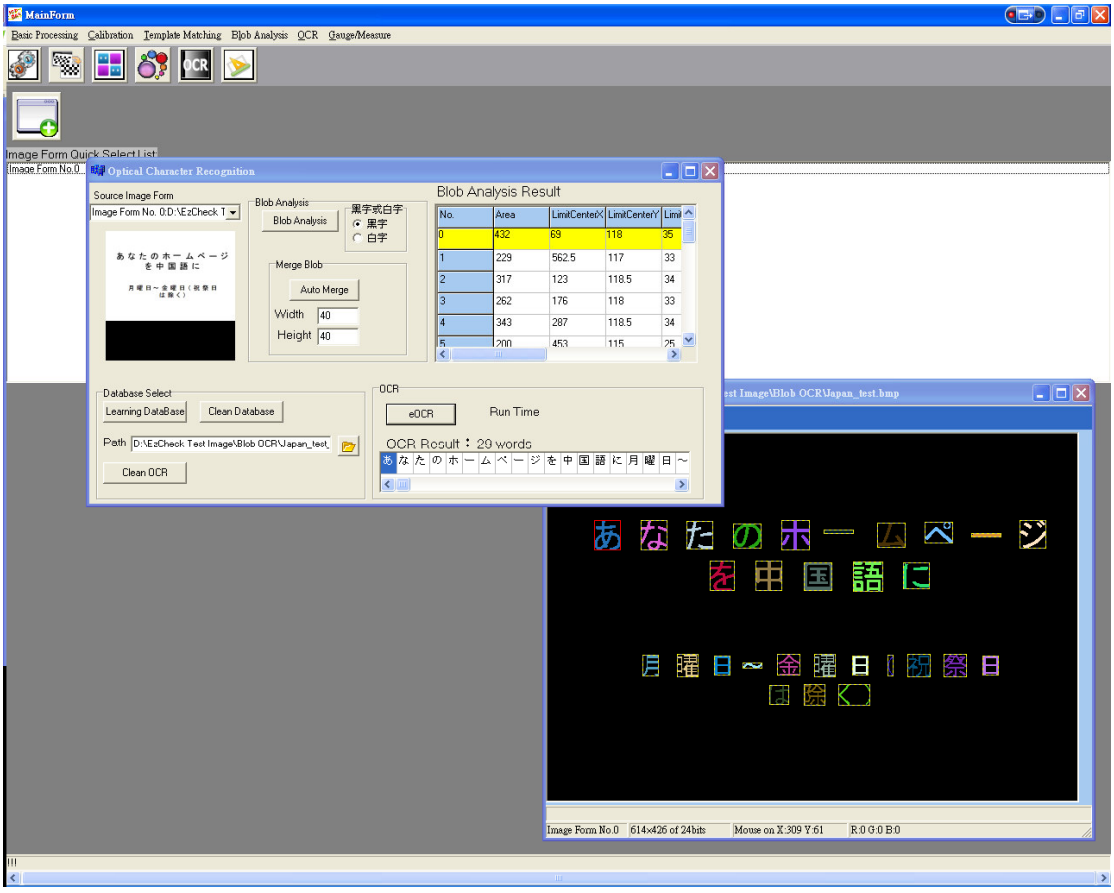
```
    SelectFeature_SigmaXY,
    SelectFeature_SigmaXX,
    SelectFeature_SigmaYY,
    SelectFeature_PixelGrayAverage,
    SelectFeature_PixelGrayMin,
    SelectFeature_PixelGrayMax,
    SelectFeature_SumPixelGray,
    SelectFeature_SumPixelGraySquare,
    SelectFeature_EllipseHeight,
    SelectFeature_EllipseWidth,
    SelectFeature_EllipseAngle,
    SelectFeature_Convex_Hull,
    SelectFeature_All,
};
```

☛ Contents introduction:

The announced definitions are the same as blob's basic features and advanced features. Please refer to **_BLOBBASICFEATURE_TAG** and **_BLOBADVANCEDFEATURE_TAG**.

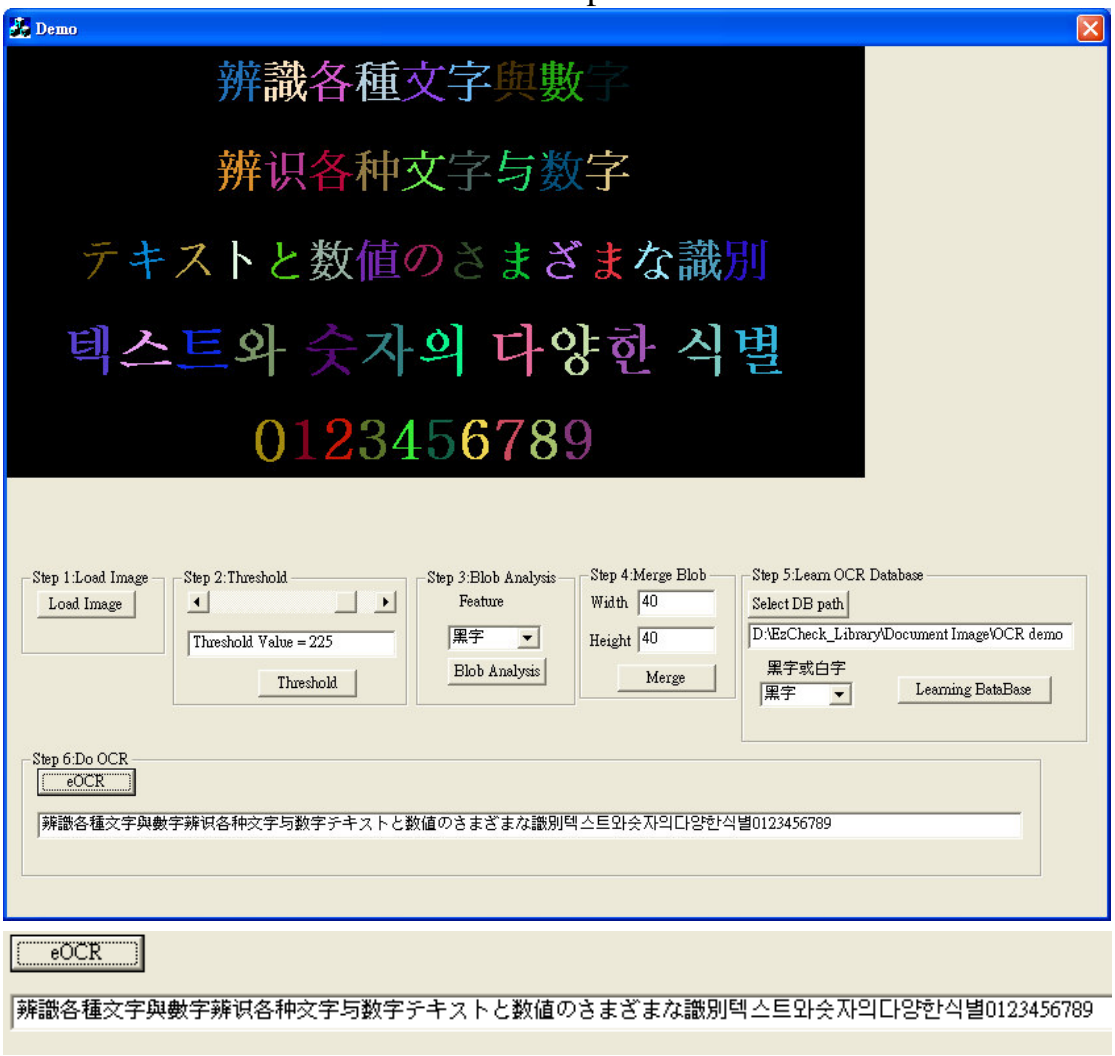
7. Optical Character Recognition —eCOCR

The eCOCR is an Optical Character Recognition library. With the trained database, users can recognize characters from several languages such as Chinese (Simplified and Traditional), Korean, Japanese, English or even numbers on the image.



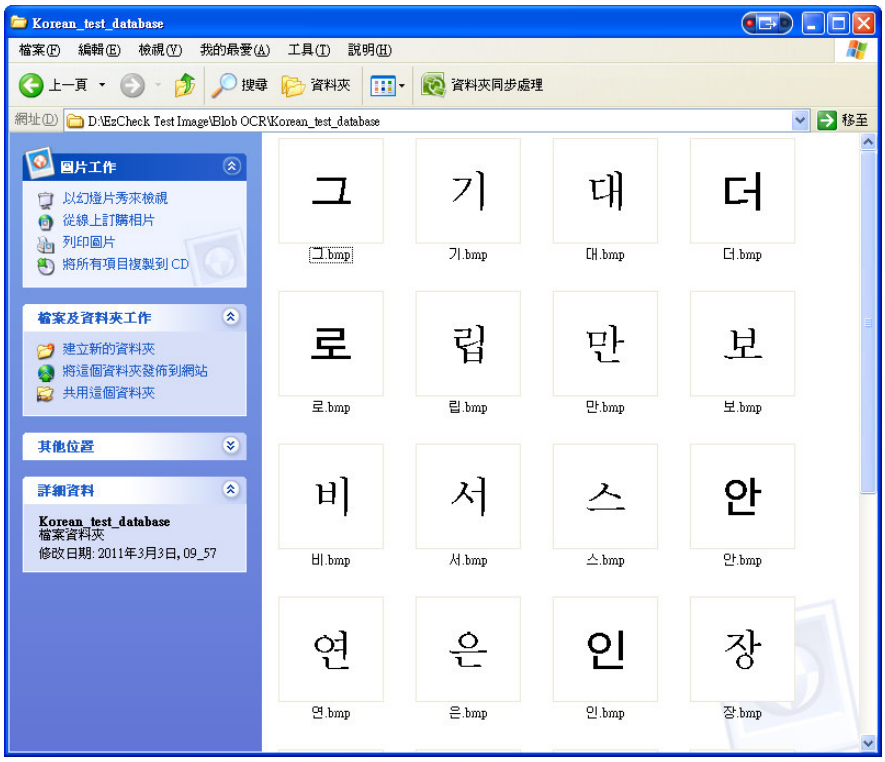
7.1 Main Features of eCOCR

- **Support numbers and characters recognition from several languages**
  Support numbers and characters recognition from several languages. As long as the OS can display that language, users can recognize it on the image by the eCOCR, no matter it is Simplified Chinese, Traditional Chinese, English, Japanese or even Korean.
- **Recognition database by folder management**
  With the classified character databases for different purpose which are saved in different folders, the recognition accuracy can be increased.
- **Support the output and sorting results**
  Users can not only obtain the OCR results such as coordinates and words but also output them as text files.



7.2 Main Functions of eCOCR

The eCOCR recognize characters in accordance with the results of eCBlob and source images. The eCOCR will begin to recognize characters with blob's information and the extra established database after Threshold and finish eCBlob's select and merge. As long as the OS can display that language, users can recognize it on the image by the eCOCR.
For establishing database, users can use SaveSingleBlob or SaveAllBlob of eCBlob to get every blob file and give them filename the same as character.



7.2.1 eCOCR

【Meaning】: Constructor.

☛ Function Prototype:

```
eCOCR ();
```

7.2.2 SetDataBaseCharacterType

【Meaning】: Set the database character type black or white. If operate succeed, return TRUE; otherwise, return FALSE.

☛ Function Prototype:

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

☛ Input Value:

Type	in	Recognize character type. Please refer to **CharacterType**.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

7.2.3 DistanceTransformDataBase

【Meaning】: Read database. If operate succeed, return TRUE; otherwise, return FALSE.

☛ Function Prototype:

Borland C++ Builder version:

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

Visual C++ version:

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

☛ Input Value:

DBFolder	in	The path of database.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

7.2.4 DistanceTransformRecognition /

7.2.5 SelectDistanceTransformRecognition

【Meaning】: These are the main function of the eCOCR.

Need to abtain the information from the eCBlob and the source image. Users can also input **select blob**.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image. Please input the image the same as blob analysis.
Blob	in	The eCBlob that provides recognize characters.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

7.2.6 GetSortResult

【Meaning】: Get the sorted result of recognition.

☛ Function Prototype:

```
PPRecognition    GetSortResult();
```

☛ Return Value:

PRecognition. It's the structure array that record the result of recognition. Please refer to **PRecognition**.

7.2.7 GetRecognitionCount

【Meaning】: Get the count of recognition result.

☛ Function Prototype:

```
UINT16    GetRecognitionCount();
```

☛ Return Value:

Return the count of recognition.

☛ 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(BlackCharacterWhit
eBackground);
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

【Meaning】: Get the operation time of OCR.

☛ Function Prototype:

```
FLOAT64    GetOCRTime();
```

☛ Return Value:

The unit of measurement is millisecond. Exclude sort processing time.

7.2.9 CleanBuffer

【Meaning】: Clean the temporary results of eCOCR.

☛ Function Prototype:

```
BOOL    CleanBuffer();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

7.2.10 CleanDataBaseBuffer

【Meaning】: Clean the database buffer in eCOCR.

☛ Function Prototype:

```
BOOL    CleanDataBaseBuffer();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

7.2.11 SaveTxt

【Meaning】: Save the OCR results to text file. Users can input the path and filename by themselves or produce it by system automatically.

☛ Function Prototype:

Borland C++ Builder version:

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

Visual C++ version:

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

☛ Input Value:

TxtName	in	The filename inputted by users.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

7.2.12 GetErrorCode

【Meaning】: Get the error code returned from eCOCR processing.

☛ Function Prototype:

```
INT32    GetErrorCode();
```

☛ Return Value:

The error code that were eCOCR processing returned. Please refer to ErrorCode List.

7.3 eCOCR Data Structures

7.3.1 PPRecognition

【Meaning】: Record the data structures of eCOCR results.

☛ The contents of announcement:

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

☛ Contents introduction:

x,y	The coordinates of the character center.
str	The result of eCOCR.

7.4 eCOCR Enumeration Types

7.4.1 CharacterType

【Meaning】: The reference of eCOCR.

☛ The contents of announcement:

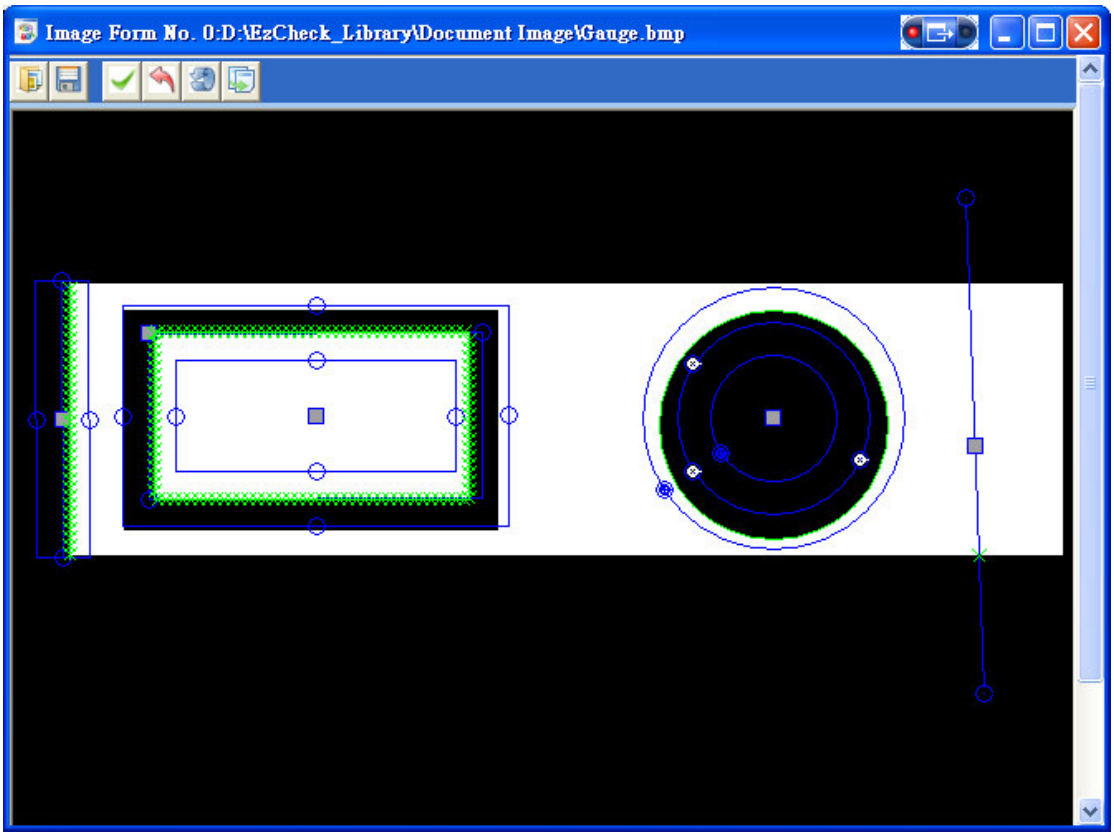
```
enum CharacterType
{
    BlackCharacterWhiteBackground = 1, //Black character with
    white background.
    WhiteCharacterBlackBackground = 2    //White character with
    black background.
};
```

☛ Contents introduction:

BlackCharacterWhiteBackground	Black character with white background
WhiteCharacterBlackBackground	White character with black background.

8. Measure and Gauge

The eGauge is an object-oriented measurement tool library which can measure several shapes and features by edge information. It is very easy to finish your measurement works by the object-oriented designed measure tools.



8.1 Main Features of eGauge

- **Object-Oriented design for Measurement Tool.**
  Every measurement tool in the eGauge is designed of object-oriented way, Users can fully use it without being an expert in programming.
- **Provide measurement tools for multiple shapes.**
  The eGauge includes four gauge tools to fit different measurement applications such as point, line, circle, and rectangle.
- **Provide flexible parameter settings.**
  Users can measure the images with different properties by tuning proper settings such as the direction and location of the gauge tools, the definition of edge, or the way that gauge result should be.
- **Coexistence of different gauge tools.**
  Users may have the demand of several different gauges applied on one image. The eGauge provides several shapes of gauge tools that can work on the same time.
- **Provide built-in image interface.**
  The eGauge contains built-in image interface for users to observe the effects with respect to different parameter settings.
- **Provide APIs for user interface development.**
  With the APIs integrated in the eGauge, Users can easily develop an interactive image interface.
- **Support sub-pixel measurement output.**
  The eGauge can perform sub-pixel gauge and output sub-pixel result to raise the accuracy of the measurement.

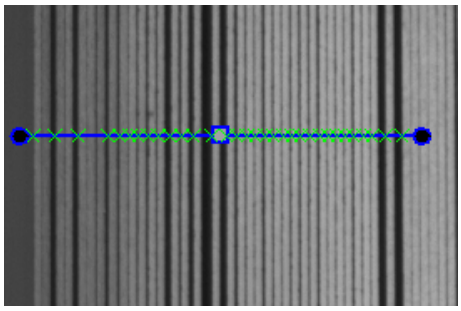
8.2 Measuring Tools

❖ **ePointGauge :**

Measure the number of feature points that the edges intersect a line segment. The location of points can also be obtained.

Gauge results:

- Number of feature points.
- The coordinates of the feature points.

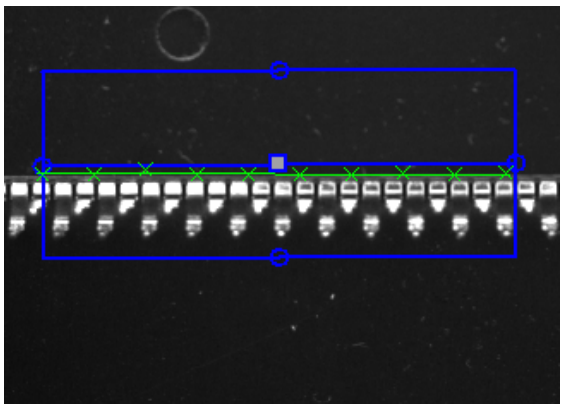


❖ **eLineGauge :**

Measure the lines formed by the edge points within a square area. The location and angle of the lines can be obtained.

Gauge results:

- The coordinates of the center of the line segment.
- The angle of the line.

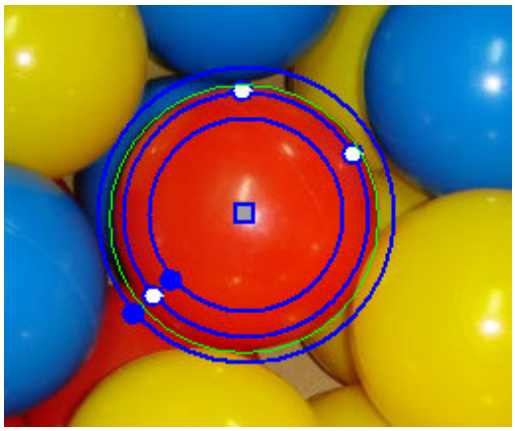


❖ eCircleGauge

Measure circles within the circular ring area of the eCircleGauge. The center of the circle and its diameter can be obtained.

Gauge results:

- The coordinates of the center of the circle.
- The diameter of the circle.

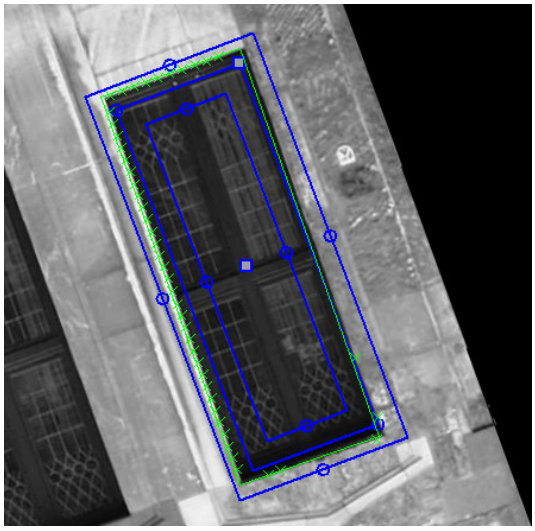


❖ eRectangleGauge

Measure the rectangles within the rectangular path area of the eRectangleGauge. The center of the rectangle and its size can be obtained.

Gauge results:

- The coordinates of the center of the rectangle.
- The width and height of the rectangle.
- The angle of the rectangle.



8.3 Classification of eTransition

The **eTransition** is the based class of ePointGauge, eLineGauge, eCircleGauge, and eRectangleGauge. All sub-class inherit its parameters and functions. The main function of **eTransistion** is that users can set their parameters by themselves. It can find out the best edge of the object. Count whether it is a shape such as Point, Line, Circle, or Rectangle after linking the edges.
All of the functions and parameters of **eTransition** are described here only, but not the chapters of sub-classes.

8.3.1 SetTolerance

【Meaning】:

1. ePointGauge: Set the half length of the object.
2. eLineGauge: Set the vertical width of the object.
3. eCircleGauge: Set the vertical width of the object.
4. eRectangleGauge: Set the vertical width of the object.

☛ Function Prototype:

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

☛ Input Value:

f32Tolerance	in	Length/Width
Angle	in	Just ePointGauge needs this parameter.

8.3.2 GetTolerance

【Meaning】:

1. ePointGauge: Get the half length of the object.
2. eLineGauge: Get the vertical width of the object.
3. eCircleGauge: Get the vertical width of the object.
4. eRectangleGauge: Get the vertical width of the object.

☛ Function Prototype:

```
FLOAT32    GetTolerance();
```

☛ Return Value:

1. ePointGauge: The half length of the object.
2. eLineGauge: The vertical width of the object.
3. eCircleGauge: The vertical width of the object.
4. eRectangleGauge: The vertical width of the object.

8.3.3 GetToleranceAngle

【Meaning】: Get the rotation angle of the ePointGauge object. Only ePointGauge support this operation.

☛ Function Prototype:

```
FLOAT32    GetToleranceAngle();
```

☛ Return Value:

The angle of the ePointGauge object.

8.3.4 SetThickness

【Meaning】: Add additional gray-scale information to aid the detection of edge. Appropriate **Thickness** parameters can increase the accuracy of edge placement and exclude noise interference. On the contrary, oversized **Thickness** parameters may make the placement distorted.

☛ Function Prototype:

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

☛ Input Value:

Un32Thickness	in	ePointGauge: The reference range for the ePointGauge object's gray-scale information. eLineGauge: The reference range width for the eLineGauge object's sampling. eCircleGauge: The reference range width for the eCircleGauge object's sampling. eRectangleGauge: The reference range width for the eRectangleGauge object's sampling.

8.3.5 GetThickness

【Meaning】: Add additional gray-scale information to aid the detection of edge. Appropriate Thickness parameters can increase the accuracy of edge placement and exclude noise interference. On the contrary, oversized Thickness parameters may make the placement distorted.

☛ Function Prototype:

```
UINT32    GetThickness();
```

☛ Return Value:

The reference range width for eGauge object's sampling. Please refer to the input value of **SetThickness**.

8.3.6 SetTransitionType

【Meaning】: Set the type of graylevel **Transition**.

☛ Function Prototype:

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

☛ Input Value:

eTransitionType	in	The **Transition** type of the measurement object. Please refer to **enum GGE_TRANSITION_TYPE**.

8.3.7 GetTransitionType

【Meaning】: Get the type of graylevel **Transition**.

☛ Function Prototype:

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

☛ Return Value:

The **Transition** type of the measurement object. Please refer to **enum GGE_TRANSITION_TYPE**.

8.3.8 SetTransitionChoice

【Meaning】: Set the type of graylevel **Choice**.

☛ Function Prototype:

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

☛ Input Value:

eTransitionChoice	in	**The way to decide the edge of the measurement objects**. Please refer to **enum GGE_TRANSITION_CHOICE**.

8.3.9 GetTransitionChoice

【Meaning】: Get the type of graylevel **Choice**.

☛ Function Prototype:

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

☛ Return Value:

The way to decide the edge of the measurement objects.
Please refer to enum GGE_TRANSITION_CHOICE.

8.3.10 SetTransitionIndex

【Meaning】: Set the index in some direction when the

enum GGE_TRANSITION_CHOICE mode is
GGE_NTH_FROM_BEGIN or GGE_NTH_FROM_END.

☛ Function Prototype:

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

☛ Input Value:

un32TransitionIndex	in	Assign the edge point from 0.

8.3.11 GetTransitionIndex

【Meaning】: Get the index in some direction when the

enum GGE_TRANSITION_CHOICE mode is
GGE_NTH_FROM_BEGIN or GGE_NTH_FROM_END.

☛ Function Prototype:

```
UINT32    GetTransitionIndex();
```

☛ Return Value:

The edge point index from 0.

8.3.12 SetThreshold

【Meaning】: Set **Threshold** value to decide which gray

level variation can look upon as an edge.

☛ Function Prototype:

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

☛ Input Value:

un32Threshold	in	Smaller the threshold is, the more edge points you can get; Larger the threshold is, the less edge points you can get. Users can adjust the settings by different situations to increase the credibility of edge measurement.

8.3.13 GetThreshold

【Meaning】: Get the **Threshold** value which decides the gray level variation looks upon as an edge.

☛ Function Prototype:

```
UINT32    GetThreshold();
```

☛ Return Value:

Please refer to the input of **SetThreshold**.

8.3.14 SetMinAmplitude

【Meaning】: Set the **minimum Amplitude** value.

☛ Function Prototype:

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

☛ Input Value:

un32MinAmplitude	in	The **minimum Amplitude** value.

8.3.15 GetMinAmplitude

【Meaning】: Get the **minimum Amplitude** value.

☛ Function Prototype:

```
UINT32    GetMinAmplitude();
```

☛ Return Value:

The **minimum Amplitude** value.

8.3.16 SetMinArea

【Meaning】: Set the minimum **measure of area**.

☛ Function Prototype:

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

☛ Input Value:

un32MinArea	in	The minimum **measure of area**.

8.3.17 SetMinArea/GetMinArea

【Meaning】: Get the setting of minimum **measure of area**.

☛ Function Prototype:

```
UINT32    GetMinArea();
```

☛ Return Value:

The setting of the **minimum** measure of area.

8.3.18 GetNumMeasuredPoints

【Meaning】: Get the numbers of **Points** after measuring.

The library is for the **ePointGauge**'s exclusive use.

☛ Function Prototype:

```
UINT32    GetNumMeasuredPoints();
```

☛ Return Value:

Get the numbers of Point after measuring.

8.3.19 GetMeasuredPoint

【Meaning】: Get the **Point**'s **coordinates** information after measuring. The **GetMeasurePoint** can apply to ePointGauge directly. But the eLineGauge, eCircleGauge, and eRectangleGauge must be with MeasureSample(index) to use it.

☛ Function Prototype:

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

☛ Input Value:

un32Index	in	Assign the number of Point (from 0). This parameter is for the ePointGauge exclusive use.

☛ Return Value:

Return the type of the ePoint which contains the information about the recorded points. Please refer to the definition in the **eCGeametry.h**. For example, users can get the coordinates of the point from GetX and GetY.

☛ Example Code:

```
CircleGauge.MeasureSample(3); // Assign the third sample path on CircleGauge.
If(CircleGauge.GetValid )    //We assume that there is edge point on the third sample path.
{
    Float x= CircleGauge.GetMeasuredPoint().GetX();
    Float y= CircleGauge.GetMeasuredPoint().GetY();
}
```

8.3.20 GetMeasuredPeak

【Meaning】: Get the **vector data** of Gauge.

☛ Function Prototype:

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

☛ Input Value:

un32Index	in	The index of m_Peaks. (from 0)

☛ Return Value:

Return **ePeak** data. Please refer to **m_Peaks**.

8.3.21 GetValid

【Meaning】: The component of Gauge is valid or not.

☛ Function Prototype:

```
BOOL    GetValid();
```

☛ Return Value:

1. ePointGauge: If the ePointGauge measured the point, return TRUE; otherwise, return FALSE.
2. eLingGauge: If the eLineGauge measured the point, return TRUE; otherwise, return FALSE. Subject to the last data of MeasureSample(index).
3. eCircleGauge: The same as eLineGauge.
4. eRectangleGauge: The same as eLineGauge.

8.3.22 SetTransitionRectangularSamplingArea

【Meaning】: Set the exterior of Gauge as rectangle or parallelogram.

☛ Function Prototype:

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

☛ Input Value:

bRectangularSamplingArea	in	The TRUE stands for rectangle. The FALSE stands for parallelogram.

8.3.23 GetTransitionRectangularSamplingArea

【Meaning】: Get the Gauge object that is rectangle or parallelogram.

☛ Function Prototype:

```
BOOL    GetTransitionRectangularSamplingArea ( )
```

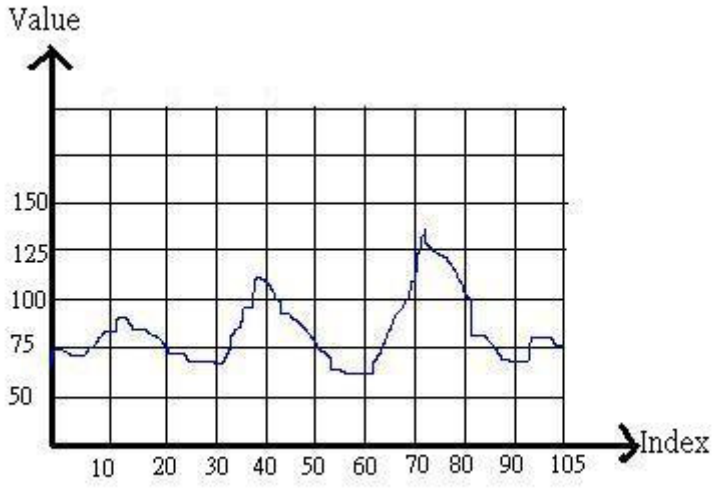
☛ Return Value:

Return the exterior of Gauge object is rectangle or parallelogram.

8.3.24 m_Profile

The parameter is the member of **eProfileVector** type in class **eTransition**. The grayscale value about the continuous path in an image was recorded as one-dimensional array **eProfileVector**. The diagrammatic curve as shown below:

1. Index: The number of Pixel in the continuous path.
2. Value: The grayscale value of Pixel.



8.3.25 m_Derivative

The parameter is the member of **eDrivativeVector** type in class **eTransition**. Count the first-order derivative value of grayscale value in the continuous path and record it as one-dimensional array **eDerivativeVector**. The diagrammatic curve is shown above.

1. Index: The number of Pixel in the continuous path.
2. Value: The first-order derivative value of Pixel.

8.3.26 m_Peaks

The parameter is the member of **ePeaksVector** type in eTransition category. The definition of Peak is **a string value that higher or lower than the first-order derivative value of Threshold**. Record it as one-dimensionas array **ePeaksVector**.

8.4 Data Structures and Enumeration Types

8.4.1 enum GGE_TRANSITION_TYPE

【Meaning】: The Transition type of grayscale.

☛ The contents of announcement:

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

☛ Contents introduction:

GGE_BW	The grayscale change is from black to white.
GGE_WB	The grayscale change is from white to black.
GGE_BW_OR_WB	The grayscale change is from black to white or from white to black.
GGE_BWB	The grayscale change is from black to white to black.
GGE_WBW	The grayscale change is from white to black to white.
GGE_UNKNOWN_TRANSITION_TYPE	= 0xFFFFFFFF

8.4.2 enum GGE_TRANSITION_CHOICE

【Meaning】: The ways to decide the edge.

☛ The contents of announcement:

```
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
};
```

☛ Contents introduction:

GGE_NTH_FROM_BEGIN	The direction of check begins at the end handle of Gauge. Just return the first edge point in this direction.
GGE_NTH_FROM_END	Just return the edge point of the biggest amplitude in the path.
GGE_ LARGEST_AMPLITUDE	Just return the edge point of the biggest area in the path to take.
GGE_ LARGEST_AREA	Just retrun the edge point that is the closest to the center handle.
GGE_ CLOSEST	Return all edge points.
GGE_ ALL	The direction of check begins at the end handle of Gauge. Just retrun to the first edge point in this direction.
GGE_UNKNOWN_TRANSITION_CHOICE	= 0xFFFFFFFF

8.4.3 struct ePeak

【Meaning】: Record the data structure of value fluctuations.

☛ The contents of announcement:

```
struct ePeak
{
   UINT32 m_un32Start, m_un32Length;
   FLOAT32 m_f32Center;
   INT32 m_n32Amplitude, m_n32Area;
};
```

☛ Contents introduction:

m_un32Start	The starting position of Peak.
m_un32Length	The length of Peak.
m_f32Center	The center position of Peak.
m_n32Amplitude	The amplitude of Peak.
m_n32Area	The measure of area of Peak.

8.5 eGauge

The eGauge contains four classes that are ePointGauge, eLineGauge, eCircleGauge and eRectangleGauge. Most of the opration of basic functions and use are similar.

8.5.1 ePointGauge

【Meaning】: The ePointGauge Constructor.

☛ Function Prototype:

```
ePointGauge    (FLOAT32 f32CenterX,
FLOAT32 f32CenterY) ;
ePointGauge    (const ePointGauge& otherInstance);
 ePointGauge&    operator= (const ePointGauge& otherInstance)
```

☛ Input Value:

f32CenterX, f32CenterY	in	The center axis of Gauge.
otherInstance	in	Another ePointGauge object.

8.5.2 eLineGauge

【Meaning】: The eLineGauge Constructor.

☛ Function Prototype:

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

☛ Input Value:

otherInstance	in	Another ePointGauge object.

8.5.3 eCircleGauge

【Meaning】: The eCircleGauge Constructor.

☛ Function Prototype:

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

☛ Input Value:

otherInstance	in	Another ePointGauge object.

8.5.4 eRectangleGauge

【Meaning】: The eRectangleGauge Constructor.

☛ Function Prototype:

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

☛ Input Value:

otherInstance	in	Another ePointGauge object.

8.5.5 SetCenter

【Meaning】: Set the center **coordinates** of Gauge.

☛ Function Prototype:

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

☛ Input Value:

f32CenterX, f32CenterY	in	The center **coordinates** of Gauge.
Point	in	The class of point. Users can use SetX and SetY

		to set the **coordinates**.

8.5.6 Rescale

【Meaning】: Narrow or enlarge the size of Gauge.

☛ Function Prototype:

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

☛ Input Value:

f32Factor	in	Percentage of narrow or enlarge.

8.5.7 SetActive

【Meaning】: Assign the Gauge can be operated or not.

☛ Function Prototype:

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

☛ Input Value:

bActive	in	TRUE: Can be operated. FALSE: Can't be operated (change the status).

8.5.8 SetZoom

【Meaning】: Assign the appearance ratio. Just the ePointGauge support this function in current version.

☛ Function Prototype:

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

☛ Input Value:

f32ZoomX	in	The percentage of the x-direction zoom.
f32ZoomY	in	The percentage of the y-direction zoom.

8.5.9 SetSelected

【Meaning】: Decide that the Gauge have been selected or not. Selected Gauge will be drawn in the more obvious way.

☛ Function Prototype:

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

☛ Input Value:

bSelected	in	The object be selected or not.

8.5.10 Measure

【Meaning】: The core function of the **ePointGauge**, measure the image to get information.

☛ Function Prototype:

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

☛ Input Value:

pSrc	in	The source image of the measured image.

8.5.11 GetMeasuredPoint

【Meaning】: Return the specific Point that had assigned the number. For **ePointGauge**, this function can get the last output. For others eGauge objects, it can get the reference point to decide the shape in processing.

☛ Function Prototype:

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

☛ Input Value:

un32Index	in	All selected Points will sort by the Gauge's detect direction. Users can get the Point coordinates by

		assign the number.

☛ Return Value:

The ePoint, please refer to input value.

☛ Example Code:

```
ePointGauge* PointGauge = new ePointGauge();//
Announce a new ePoingGauge component.
int Point_Number = 0;
PointGauge->SetCenter(100, 100);// Put the Point Gauge
component in the (100, 100).
PointGauge->Measure(SrcImage);// Measure the SrcImage
Point_Number =
PointGauge->GetNumMeasuredPoints();// Get the
numbers of measured points.
PointGauge->GetMeasuredPoint(0).GetX();// Get the
x-axis of the no.0 point.
PointGauge->GetMeasuredPoint(0).GetY();// Get the
y-axis of the no.0 point.
```

8.5.12 GetMeasuredLine

【Meaning】: Return the information about line measured by eLineGauge.

☛ Function Prototype:

```
eLine    GetMeasuredPoint();
```

☛ Return Value:

The **eLine** is a class that records the information of line. Users can use **CetCenterX**, **GetCenterY** and **GetAngle** to get the information about the line.

☛ Example Code:

```
eLineGauge* LineGauge = new eLineGauge();// Establish
a new LineGauge component.
LineGauge->Measure(SrcImage);// Measure the SrcImage.
```

```
LineGauge ->GetMeasuredLine().GetCenterX();// Get the x-axis of the line center.
LineGauge -> GetMeasuredLine().GetCenterY();// Get the y-axis of the line center.
LineGauge->GetMeasuredLine().GetAngle()/(PI/180);// Get the angle of line, and the PI stands for the ratio of the circumference of a circle to its diameter
```

8.5.13 GetMeasuredCircle

【Meaning】: Return the information about the circle measured by eLingGauge.

☛ Function Prototype:

```
eCircle    GetMeasuredCircle();
```

☛ Return Value:

The **eCircle** is a class records the information of the circle. Users can use **GetCenterX**, **GetCenterY**, and **GetDiameter** to get the information about the circle.

☛ Example Code:

```
eCircleGauge * CircleGauge = new eCircleGauge;// Establish a new eCircleGauge component.
CircleGauge ->Measure(SrcImage);// Measure the SrcImage.
CircleGauge->GetMeasuredCircle().GetCenterX();// Get the x-axis of the circle.
CircleGauge->GetMeasuredCircle().GetCenterY();// Get the y-axis of the circle.
CircleGauge->GetMeasuredCircle().GetDiameter();// Get the diameter of the circle.
```

8.5.14 GetMeasuredRectangle

【Meaning】: Return the information about the Rectangle measured by eRectangleGauge.

☛ Function Prototype:

```
eRectangle      GetMeasuredRectangle ();
```

☛ Return Value:

The **eRectangle** is a class that records the information of rectangle. Users can use **GetCenter**, **GetCenterY**, and **GetSizeX** to get the information about the Rectangle.

☛ Example Code:

```
eRectangleGauge * RectangleGauge = new
eRectangleGauge;// Establish a new eRectangleGauge
component.
CircleGauge ->Measure(SrcImage);// Measure the
SrcImage.
RectangleGauge->GetMeasuredRectangle().GetCenterX();
// Get the x-axis of the center of the Rectangle.
RectangleGauge->GetMeasuredRectangle().GetCenterY();
// Get the y-axis of the center of the Rectangle.
RectangleGauge->GetMeasuredRectangle().GetSizeX();//
Get the Rectangle width.
RectangleGauge->GetMeasuredRectangle().GetSizeY();//
Get the Rectangle height.
RectangleGauge->GetMeasuredRectangle().GetAngle()/(P
I/180)); // Get the tilt angle of the Rectangle.
```

8.5.15 GetType

【Meaning】: Return the Shape's type of the Gauge component.

☛ Function Prototype:

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

☛ Return Value:

The enum **INS_SHAPE_TYPES**, please refer to **enum INS_SHAPE_TYPES**.

8.5.16 Draw

【Meaning】: Draw the appearance of Gauge.

☛ Function Prototype:

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

☛ Input Value:

hDC	in	The index value of the output target.
eDrawingMode	in	The draw setting of the Gauge component. Please refer to **enum INS_DRAWING_MODES**.

8.5.17 HitTest

【Meaning】: Check whether the mouse hit in the Handle's range of eGauge.

☛ Function Prototype:

```
BOOL    HitTest ();
```

☛ Return Value:

If the mpuse hit in the Handle's range, retrun TRUE; otherwise, return FALSE.

8.5.18 Drag

【Meaning】: Use the mouse to drag and drop the eGauge component to the new axis.

☛ Function Prototype:

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

☛ Input Value:

n32CursorX, n32CursorY	in	Assign the new axis.

8.6 eGauge Enumeration Types

8.6.1 enum INS_SHAPE_TYPES

【Meaning】: The enumerated type is used to describe the shap.

☛ The contents of announcement:

```
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

【Meaning】: The way the **Gauge** compenent change

when the mouse press and drag the **Handle** of the **Gauge** component. For example:

1. eLineGauge:
   In the **INS_DRAG_STANDARD** mode, when users pull the **INS_HANDLE_ORG**, the axis of **INS_HANDLE_END** will be changed at the same time, and the direction of the change is contrary to **INS_HANDLE_ORG**.
   But in the **INS_DRAG _TO_EDGES** mode, when users pull the **INS_HANDLE_ORG**, the **INS_HANDLE_END** won't be changed together.
2. eCircleGauge:
   In the **INS_DRAG_STANDARD** mode, just leave the **INS_HANDLE_ORG** in the control point **INS_HANDLE_ORG/MID/END** to operate. The Circle size will be narrowed or enlarged in the center-based way when users pull the **INS_HANDLE_ORG**.
   In the **INS_DRAG _TO_EDGES** mode, users can pull the **INS_HANDLE_ORG/MID/END** in the single side, and the Circle will become to a circle according to this three Handle.

☛ The contents of definition:

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

☛ Contents introduction:

INS_DRAG_STANDARD	The change of the Gauge size is synchronous.
INS_DRAG_TO_EDGES	The change of the Gauge size is unilateral.
INS_DRAG_UNKNOWN	0xFFFFFFFF

8.6.3 enum INS_HANDLES

【Meaning】: Users can operate the Handle of Gauge component through the graph interface.

☛ The contents of announcement:

```
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,
```

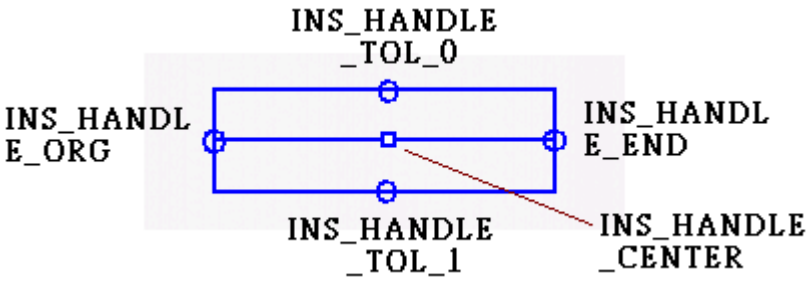
```
    INS_EDGE_Y                      = INS_HANDLE_EDGE_Y,

    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
};
```

☛ Contents introduction:

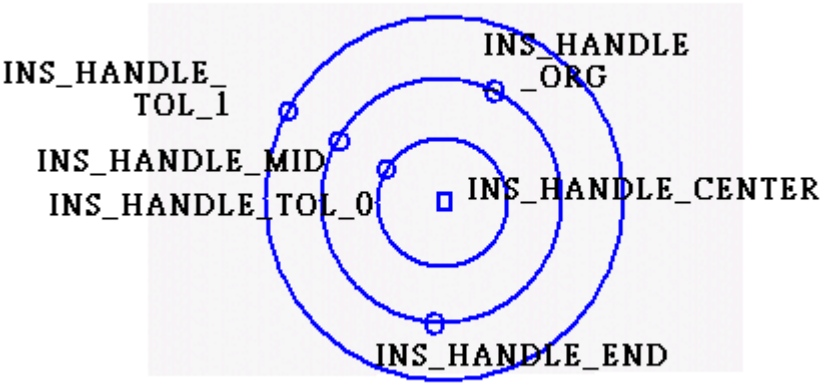
Please refer to the schematic diagram as shown below:

1. The schematic diagram of the ePointGauge Handle

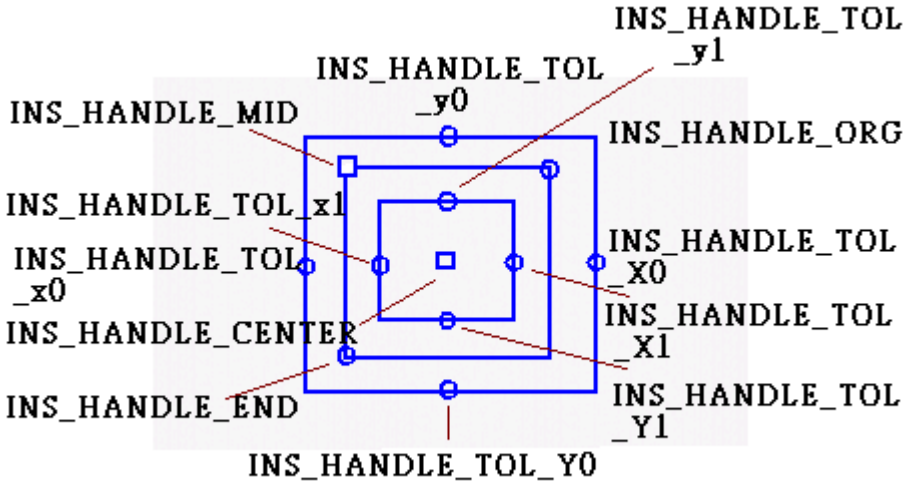
2. The schematic diagram of the eLineGauge Handle



3. The schematic diagram of the eCircleGauge Handle



4. The schematic diagram of the eRectangleGauge Handle



8.6.4 enum INS_DRAWING_MODES

【Meaning】: The draw setting of Gauge component.

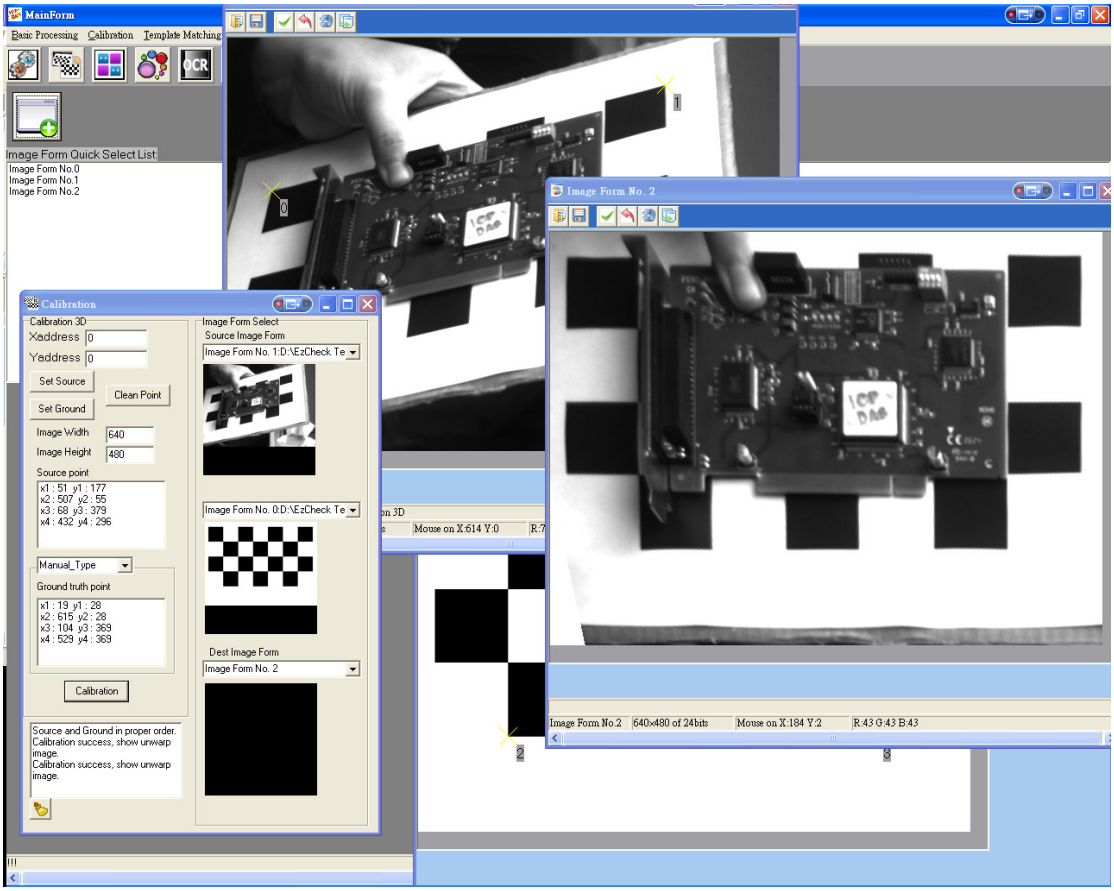
☛ The contents of announcement:

```
enum INS_DRAWING_MODES
{
   INS_DRAW_NOMINAL = 0x01,    // Draw the Gauge and its control point.
   INS_DRAW_ACTUAL, // Draw the measure result of Gauge (edge, Line,
      Circle, and Rectangle)
   INS_DRAW_TOLERANCE,          // Draw the range of Tolerance
   INS_DRAW_SAMPLED_PATHS,    // Draw the all sampling paths
   INS_DRAW_SAMPLED_PATH,     // Draw the assigned sampling path
   INS_DRAW_POINTS_IN_SKIP_RANGE,
   INS_DRAW_SAMPLED_POINTS, // Draw the all edge points
   INS_DRAW_SAMPLED_POINT,    // Draw the assigned edge point
   INS_DRAW_INVALID_SAMPLED_POINTS,
   INS_DRAWING_MODE_UNKNOWN    = 0xFFFFFFFF
};
```

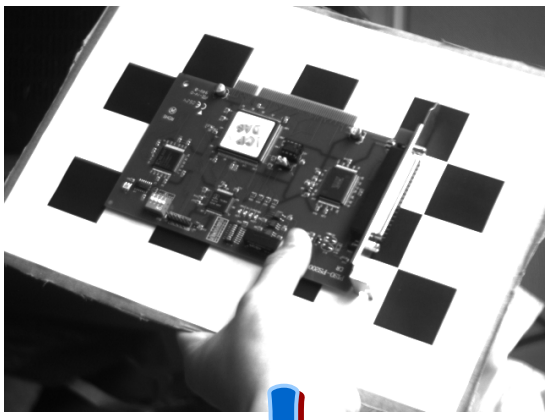
9. 3D Image Calibration – eCCalib3D

The eCCalib3D is an image calibration library. It can calibrate or unwarp the images which are skew or warped that caused by the difficulties in photography. The unwarped image may be very useful or even necessary to the subsequent measurement or recognition.

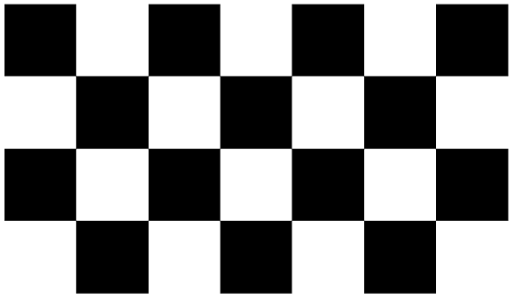
9.1 Main Features of eCCalib3D



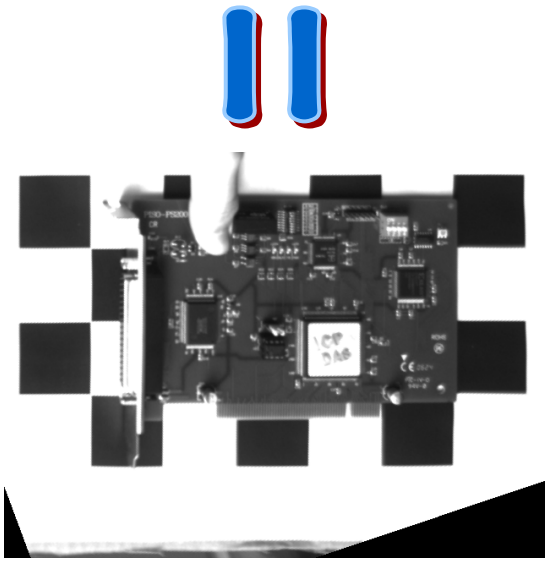
- **The image calibration is not limited by angle or dimensions.**
  Regardless of the rotation and warped of the image, the calibration can be achieved only by four correspondence points on both source image and pattern image.
- **Support semi-auto process.**
  The eCCalib3D can calibrate the image with assigned parameters and auto sorted feature points.



Source Image



calibration pattern



calibration result

9.2 Main Functions of eCCalib3D

The eCCalib3D calculate the skew and warp of the image in accordance with the four correspondence points. Users should input the source image (the image needs to be unwarped), the pattern image (otherwise known as ground truth) and the four correspondence points. Currently, the four correspondence points of the source image are inputed by manual way, and the four correspondence points of the pattern image are gotten by manual, semi-auto, or auto way.

9.2.1 eCCalib3D

【Meaning】: Constructor.

☛ Function Prototype:

```
eCCalib3D        ();
```

9.2.2 ~eCCalib3D

【Meaning】: Destructor. The eCCalib3D contends the temporary images and matrix cause that users should free it when they without to use to avoid the memory stack.

☛ Function Prototype:

```
~eCCalib3D ();
```

9.2.3 SrcPT

【Meaning】: It is the public member of the **eCCalib3D** class, records the four correspondence points of the source image. Users can access it directly.

☛ Function Prototype:

```
eCPoint    SrcPT[4];
```

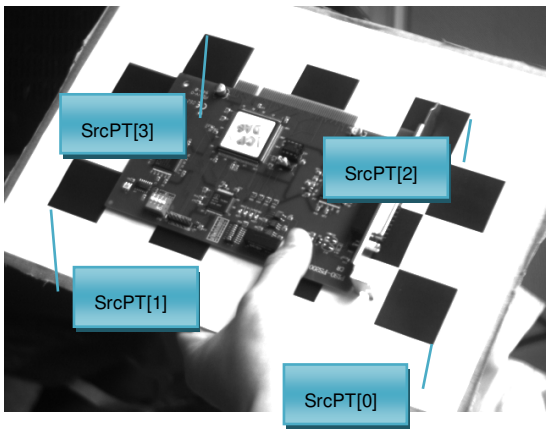
```
struct eCPoint
{
        int x;
        int y;
};
```

9.2.4 GroundPT

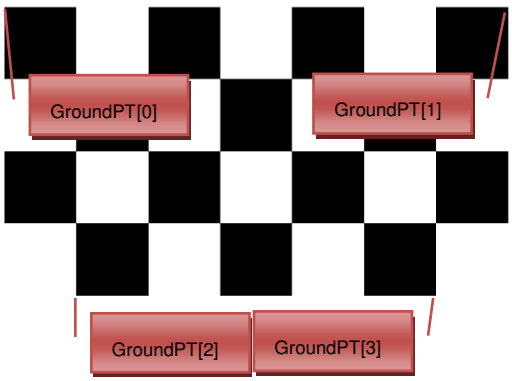
【Meaning】: It's the public member of the **eCCalib3D** class, record the four correspondence points of the pattern image. Users can access it directly.

Tips: The points of the SrcPT and GroundPT should have correspondence relationship in order to ensure the correctness of the subsequent calibration.

Example：



Source Image



Pattern Image

9.2.5 SeFourPoint

【Meaning】: Set the four correspondence points by automatic or manual.

☛ Function Prototype:

```
BOOL   SetFourPoint(UINT16 X_axis,
                    UINT16 Y_axis,
                    UINT8 WhichSet);
BOOL   SetFourPoint (UINT16 X_axis,
                    UINT16 Y_axis,
                    UINT8 WhichSet ,
                    UINT8 PointNumber);
```

☛ Input Value:

X_axis	in	The x-axis of the points.
Y_axis	in	The y-axis of the points.
WhichSet	in	Input the group of the points. 0: Input to SrcPT. 1: Input to GroundPT.
PointNumber	in	The number of the points.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

☛ Example Code:

```
eCCalib3D cali3D;//Announce the eCCalib3D component.
cali3D.SetFourPoint(100, 100, 0);// Write (100, 100) to SrcPT [0].
Automatically jump number.
cali3D.SetFourPoint(200, 100, 0); //.Write (200, 100) to SrcPT[1].
Automatically jump number.
cali3D.SetFourPoint(100, 200, 0, 2); // Write (100, 200) to SrcPT[2].
Specify the number.
cali3D.SetFourPoint(200, 200, 0, 3); // Write (200, 200) to SrcPT[3].
Specify the number.
```

Tips: Users can input the coordinates to the specific point. Moreover, because of the built-in counter, it can input the number of the point automatically. The fifth point will replace the first point if the points entered by users are more than four points; and the sixth point will replace the second point, and so on.

9.2.6 CleanPoint

【Meaning】: Clean the data of SrcPT and GroundPT.

☛ Function Prototype:

```
BOOL    CleanPoint();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

9.2.7 CleanSourcePoint

【Meaning】: Clean the data of SrcPT.

☛ Function Prototype:

```
BOOL    CleanSourcePoint();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

9.2.8 CleanGroundPoint

【Meaning】: Clean the data of GroundPT.

☛ Function Prototype:

```
BOOL    CleanSourcePoint();
```

☛ Return Value:

If operate succeed, return TRUE; otherwise, return

FALSE.

9.2.9 SetSampleSize

【Meaning】: Set the wanted size.

☛ Function Prototype:

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

☛ Input Value:

Width	in	Sample Width.
Height	in	Sample Height.
Sample size	in	Only needs to be set when using auto calibration. It will be the same as the source image's size if users didn't set.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

9.2.10 GetSampleWidth

【Meaning】: Get the current Sample Width setting.

☛ Function Prototype:

```
INT32    GetSampleWidth();
```

☛ Return Value:

Sample Width.

9.2.11 GetSampleHeight

【Meaning】: Get the current Sample Height setting.

☛ Function Prototype:

```
INT32    GetSampleHeight();
```

☛ Return Value:

Sample Height.

9.2.12 SetImageSize

【Meaning】: Set the output image size after calibration.

☛ Function Prototype:

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

☛ Input Value:

Width	in	The output image width.
Height	in	The output image height.
If users never input the assigned size of output image then the size of output image is the same as source image.		

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

9.2.13 GetImageWidth

【Meaning】: Get the output image width setting.

☛ Function Prototype:

```
INT32    GetImageWidth();
```

☛ Return Value:

The output image width setting.

9.2.14 GetImageHeight

【Meaning】: Get the output image height setting.

☛ Function Prototype:

```
INT32    GetImageHeight();
```

☛ Return Value:

The output image height setting.

9.2.15 Calibration

【Meaning】: It's the core function of the **eCCalib3D**. It unwarp the source image in accordance with the correspoindence points. If the calculation is successful, return TRUE; if the opreate processing without taking data, return FALSE and setted error code.
Users can begin to use it after setting the SrcPT and GroundPT. Users can set the auto mode then system will decide the GroundPT automatically, but users must set the sample size first in this mode. The auto mode will correct the "Sample" to the center of "Image", calculate the four points of GroundPT in accordance with Image size and Sample size, and sort the SrcPT points to fit the sequence of GroundP with simultaneous correction. The auto mode just suit to the skew angle is +/-90 degrees.

☛ Function Prototype:

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

☛ Input Value:

SrcImage	in	Source Image.
GD	in	Whether users set the GroundPT by themselves. True: Correct the image in accordance with SrcPT and GroundPT's four correspoindence points. False: The GroundPT is produced by system automatically in accordance with sample size and image size.

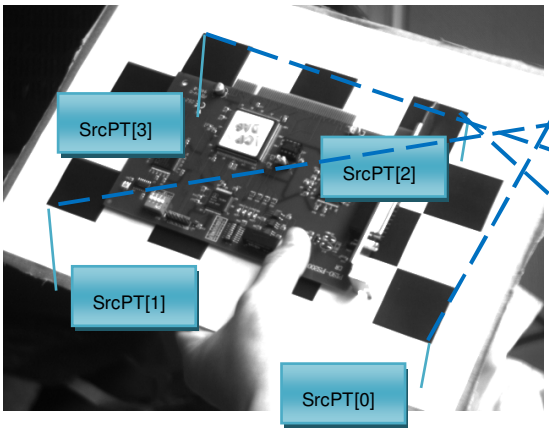
☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

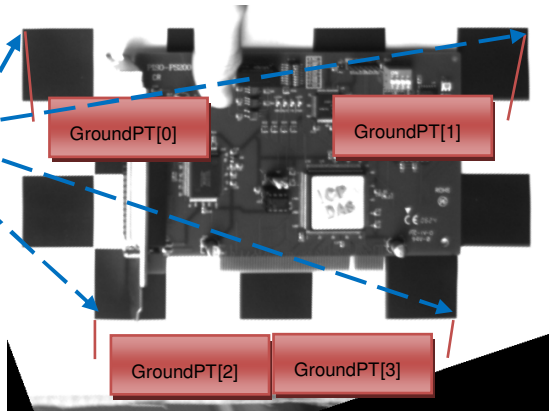
☛ Example Code 1 (manual correction) :

```
eCCalib3D cali3D;// Announce the eCCalib3D object
// Set the SrcPT and GroundPT, the number must be consistent with the order.
cali3D.SetImageSize(640, 480); // Set the image size to 640*480
```

cali3D.Calibration(simage, true); //Set the four correspoindence to correct by manual correction.



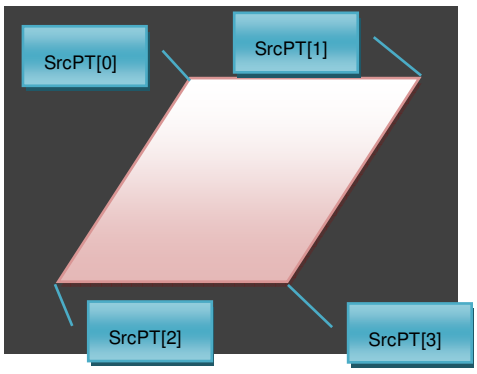
Source Image



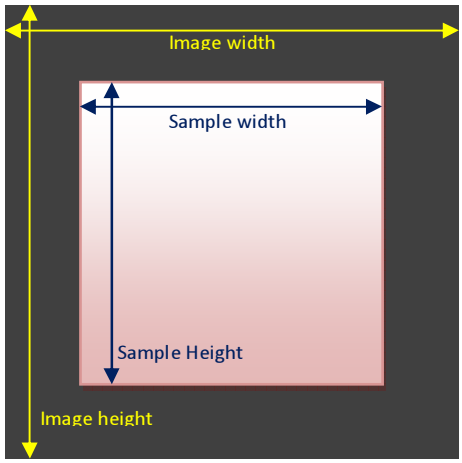
Pattern Image

☛ Example Code 2 (auto correction) :

eCCalib3D cali3D; // Announce the eCCalib3D object.
// Set the SrcPT, never mind the order.
cali3D. SetSampleSize(100, 100); // The object's size of ouput is 400*400.
cali3D. SetImageSize(300, 300); // The image size of output is 600*600.
cali3D.Calibration(simage, false); // Automatic correction.



The SrcPTof source image and auto sort



Correction result

Tips: Current computing will be stored in memory temporarily after Calibration. If users never use the SetFourPoint or SetSampleSize to change the operation, the computing will be applied the transformation matrix

directly to speed computing in a large number of image correction.

9.2.16 GetUnwarpImage

【Meaning】: Get the calibrate output image.

☛ Function Prototype:

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

☛ Input Value:

DstImage	out	The image that has corrected.

☛ Return Value:

If operate succeed, return TRUE; otherwise, return FALSE.

9.2.17 GetErrorCode

【Meaning】: Get the error code returned in the eCCalib3D processing.

☛ Function Prototype:

```
INT32    GetErrorCode();
```

☛ Input Value:

DstImage	out	The image that has corrected.

☛ Return Value:

Get the error code returned in the eCCalib3D processing. The error codes were defined in ErrorCode.h. Please refer to the **ErrorCode List**.

10. ErrorCode

10.1 ErrorCode Introduction

The ErrorCode contains all the problems or exceptions that will maybe happen when the EzCheck Vision Library operates. Users will can through it to find out the problems when the library returns FALSE or loses the expected results.

10.2 ErrorCode List

10.2.1 enum GENERAL_ERRORS

☛ The contents of this announcement:

```
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,
};
```

☛ Contents introduction:

ErrorCode	**No.**	**Meaning**
ERROR_OK	0	Normal operation.
General Error		
ERROR_INVALID_FILE_NAME	100	File path error, or file does not exist. Note that if there is a blank in the path name.
ERROR_INVALID_EXTENSION_NAME	101	This file has no Extension Name.
ERROR_INVALID_IMAGE_FORMAT	102	Non-image file format.
ERROR_INVALID_BPP	103	This current version of the EzCheck just support 24-bit or 8-bit grayscale image. Please note that the file must be supported.
ERROR_INVALID_DATA_STRUCTURE	104	Image data does not exist in this category. Please check to see if the data structure of this category has data.
ERROR_INVALID_IMPORT_PARAMETER	105	Please check the import parameters are correct.

ERROR_INVALID_CLEAN	106	Can't clear data because the data don’t exist.
ERROR_INVALID_FOLDER	107	The folder doesn’t exit.
ERROR_RESULT_ZERO	108	The result is zero. Please check the import data and setting conditions are correct.
eCImage Error		
ERROR_ECIMAGE_SOURCE_ERROR	109	The source image that had imported to eCImage there were something wrong with it.
ERROR_ECIMAGE_CREATE_ERROR	110	There is something wrong with importing eCImage.
ERROR_ECIMAGE_DIDNOT_NEW_ERROR	111	The eCImage hasn’t been initialized.
ERROR_ECIMAGE_IS_NULL	112	The eCImage is empty.
ERROR_ECIMAGE_LOAD_ERROR	113	There is something wrong with loading files.
ERROR_ECIMAGE_COPY_ERROR	114	There is something wrong with GetCopy.
ERROR_ECIMAGE_SETTING_ERROR	115	There is something wrong when setting eCImage.
ERROR_ECIMAGE_NOFILENAME_ERROR	116	Filename is invalid.
ERROR_ECIMAGE_16BITIMAGE_ERROR	117	The 16-bit images can not be displayed.
ERROR_ECIMAGE_ROIRANGE_ERROR	118	The range of ROI and Source Image are the same.
ERROR_ECIMAGE_CAMERA_SIZE	119	Incompatibility between the eCImage and camera buffer.
eCBlob Error		
ERROR_INVALID_BLOBANALYSIS_MODE	120	Invalid analysis mode. Please refer to the description in the datafile.h.
ERROR_BLOB_LIMIT	121	The blobs are overload. Just support 65535 blobs and 16-bits.

ERROR_INVALID_BLOBANALYSIS	122	The result of analysis is zero. Please check the import parameters and data are correct.
ERROR_INVALID_BASIC_FEATURE	123	The basic feature doesn't exist. Please check theBlob Analysis is done and its operating is correct.
ERROR_INVALID_IMAGE_DATA	124	Current image size is different from Blob Analysis image size.
ERROR_INVALID_BLOB_NUMBER	125	The blob numbers don't exist after blob analysis.
ERROR_INVALID_PROCESS_MODE	126	Invalid processing mode, please refer to the description of PROCESSMODE enum.
ERROR_INVALID_FEATURE_TYPE	127	Invalid processing mode, please refer to the description of SELECTFEATURE enum.
ERROR_INVALID_OPTION_TYPE	128	Invalid processing mode, please refer to the description of SELECTOPTIONS enum.
ERROR_INVALID_SORT_TYPE	129	Invalid processing mode, please refer to the description of SORTOPTIONS enum.
ERROR_INVALID_ADVANCED_FEATURE	130	The advanced feature doesn't exist. Please check to see if you have done the calculation of advanced features.
ERROR_INVALID_ADD_MERGE_LIST	131	The blob number has existed in the mergelist.
ERROR_INVALID_MANUAL_MERGE	132	The presence of blob numbers must have more than two in the mergelist.
ERROR_INVALID_THRESHOLD_VALUE	133	There is something wrong with Threshold imported by Select library. Please check to see if the

		minimum and maximum Threshold met the definition.
eCOCR Error		
ERROR_INVALID_DATABASE_TYPE	134	There is something wrong with the color setting of database. Please refer to the description of Character Type enum.
ERROR_INVALID_OCR_LEARNING	135	There is no quantity of database, please check to see if the database have image files.
ERROR_YET_LEARNING_DATABASE	136	Not yet learning database.
ERROR_INVALID_OCR_COUNT	137	The number of identified is zero.
ERROR_INVALID_OCR_TYPE	138	The identified color is wrong. Please refer to the description of ANALYSISCLASS enum.
ERROR_YET_OCR	139	OCR has not yet identified.
ERROR_INVALID_ROW_LIMIT	140	The sort is overload. Exceeds the limit of 255 rows
ERROR_INVALID_CLEAN_DATABASE	141	The database clean failed. Please check to see if done the learning function.
ERROR_INVALID_RECOGNITION_DATA	142	The size of Source and Blob are different.
eCImg Shared Error		
ERROR_EZCHECK_FUNC_ERROR	143	Other exceptions. Return the complete steps of processing to developers.
ERROR_SOURCE_IMAGE_ERROR	144	The Source Image is wrong.
ERROR_DEST_IMAGE_ERROR	145	The Destination Image is wrong.
ERROR_MASK_INFO_ERROR	146	The cause of opration failed is that the imported shape is wrong.
ERROR_COLOR_UNCHANGED	147	The cause of channel conversion

		failed is that the image is grayscale.
eCCalib3D Error		
ERROR_CALIB3D_POINT_ERROR	148	The axes of four correspondence points are out of range.
ERROR_CALIB3D_UNWARP_IMAGE_ERROR	149	The axes of four correspondence points are out of range.
ERROR_CALIB3D_UNWARP_NOT_READY	150	Users need to correct before reading the calibrate image.
eCTM Error		
ERROR_TM_NODATA_ERROR	151	Users need to do template matching before accessing data.
ERROR_TM_SORT_INPUT_ERROR	152	Users need to do template matching before accessing data.
USB Hardware Key Error		
ERROR_CHECK_USBHK	153	Didn’t find the correspondence USB hardkey.

11. EzCheck Utility

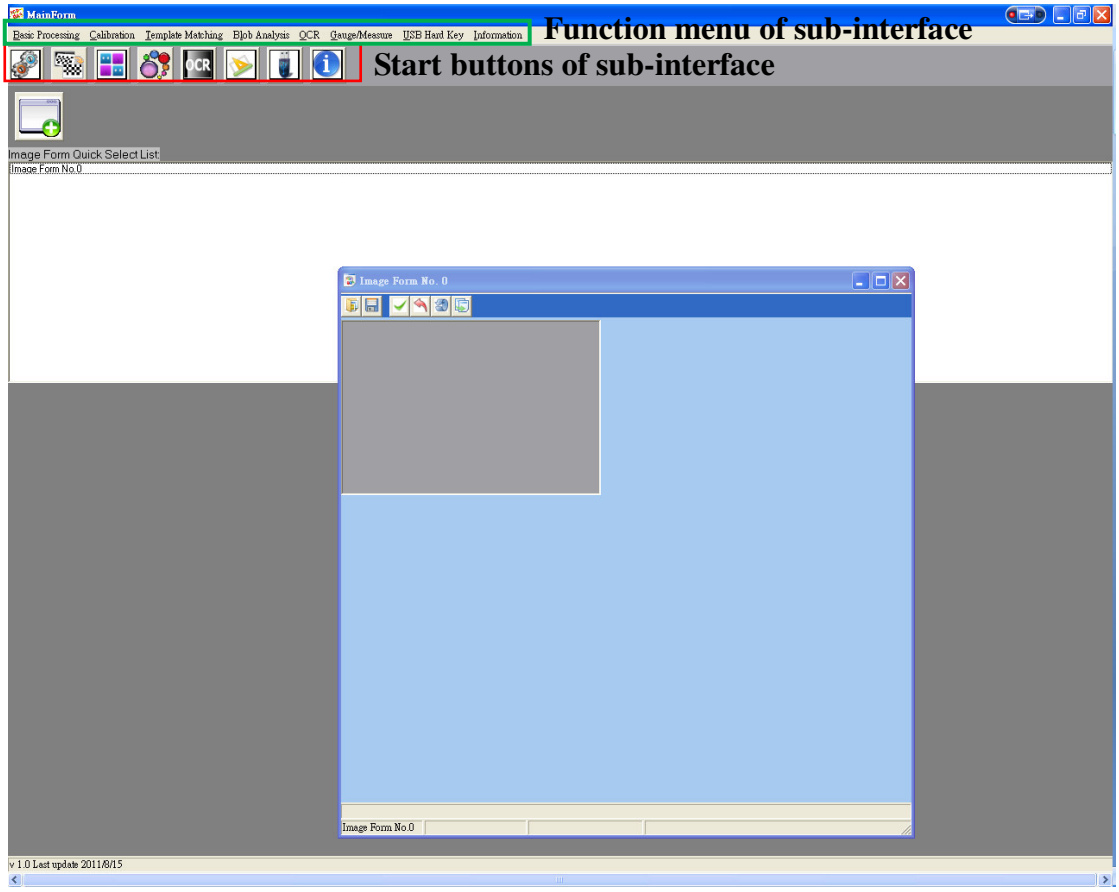
ICP DAS provide the **EzCheck Utility** developed with EzCheck Vision Library for users who would like to experience the usage and convenience of the EzCheck Vision Library.

11.1 Main Feature

- **Achieved with EzCheck Vision Library**
  Every capability including image accessing, processing and analysis is achieved with EzCheck Vision Library.
- **A full demonstration of EzCheck Vision Library**
  The Utility contains all of the main functions of EzCheck Vision Library which is combined with useful interactive user interface. Users can quickly understand EzCheck Vision Library.
- **Useful Interface**
  Utility demonstrates the easy and practical user interfaces for the main functions of the EzCheck Vision Library. Users can test the functions of the library and evaluate the images by the interfaces. It's a very convenient and helpful tool along with the EzCheck Vision Library.

11.2 Main Interface

The main interface contains start buttons, menu, and function buttons of several sub-interfaces. When users click one of start buttons or menu can open the correspondence sub-interface, and the system will close the other sub-interfaces (Except the USB Hard-key interface and the information interface).



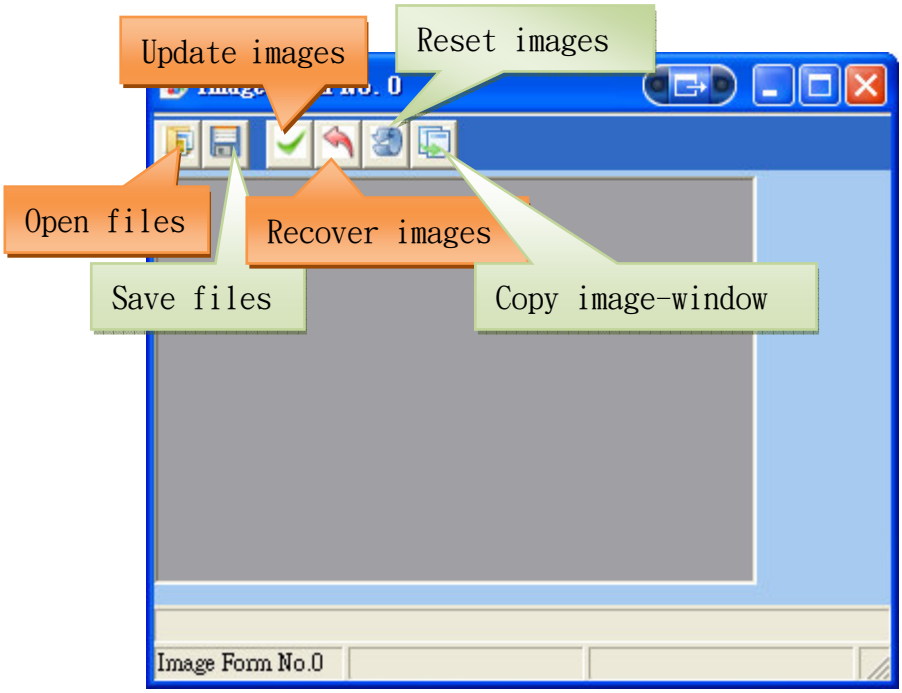
11.3 Image Window

The Image-window takes responsibility of loading, saving, and displaying image. Besides, it contains some interactive user interfaces of each main function. Users can create several Image-forms to do some complex works such as template matching and calibration. Just click  on the main screen to open a new image-window.

11.3.1 File Management

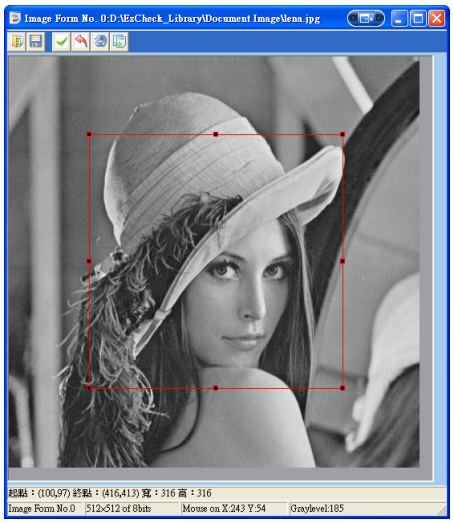
There are six buttons on the image-window, they are open files, save files, update files, recover files, reset files, and copy image-window.

(1) Open files: Open the file and display in the image-window. The first read image will be a fixed image buffer as an initial imgae when reset.
(2) Save files: Save the last image.Please update the image before saving.
(3) Update files: There are many functions in the EzCheck Utility output the result as temporary image. Therefore the shown images sometimes are temporary images, and it won't be saved after operating. We suggest users update to save the just completed file after every step.
(4) Recover files: Undo your last action.
(5) Reset files: Reset the image. The usual situation for reset files is restore the image buffer to the original images.
(6) Copy image-window: Create a new image-window and copy all information to it.

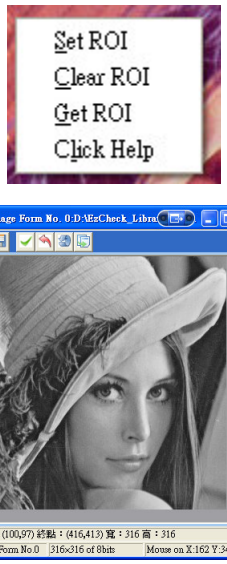


11.3.2 ROI Management

Users can draw the ROI frame by dragging, and set the ROI after establishing. Internal part of the frame and the eight control points of frame can be adjust the size and the position by dragging. Click the external part of the frame can reset the ROI.



Point right within the image-window then you can open the right menu. Include Set ROI, Clear ROI, Get ROI and Click Help, and the Get ROI function can capture the range of ROI.



Tips: Remember to updata if you want to retain the captured ROI range.

11.3.3 Click Help

Some of EzCheck Utility functions need precise picture detail when clicking image such as calibration. Click Help provides the similar functionality of a magnifier to help users do the fine-tuning of the cursor. Press right button on the image-window to open the right menu and choose the Click Help then can open the interface of Click Help. Click again to close it.

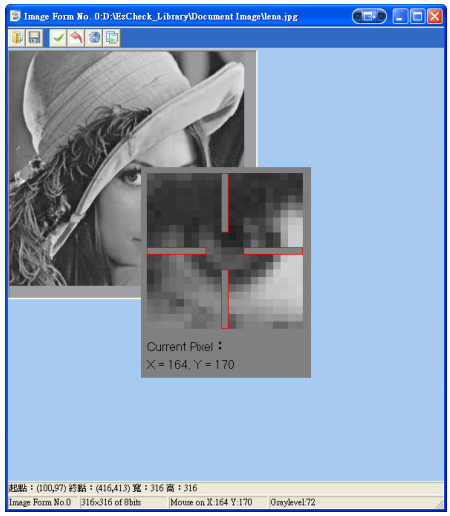
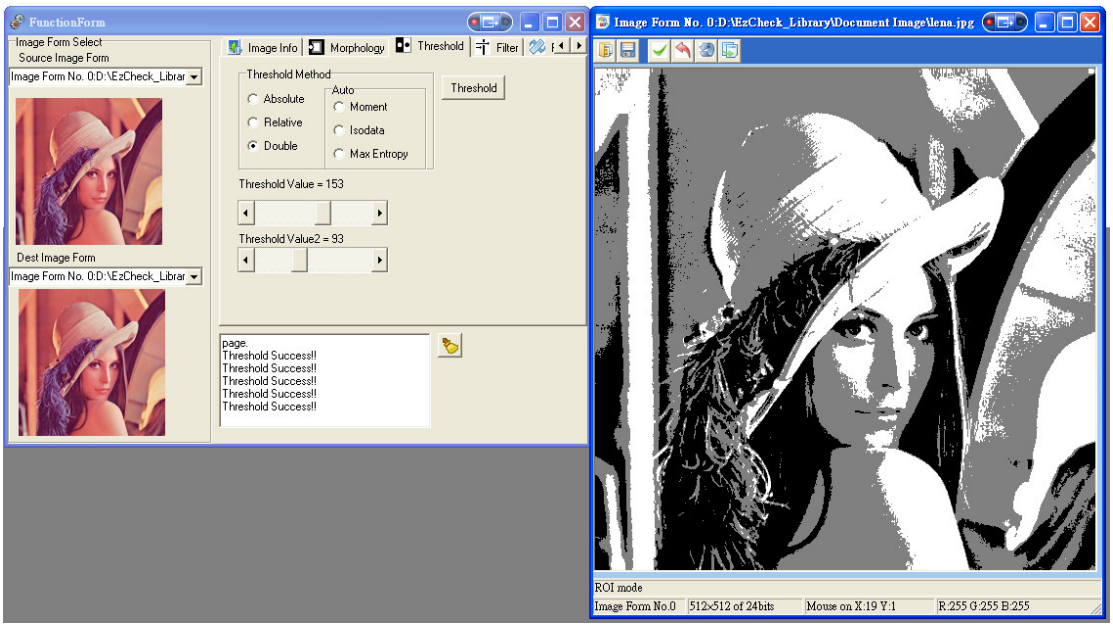


Image Form No. 0:D:\EzCheck_Library\Document Image\lena.jpg
Current Pixel :
X = 164, Y = 170
起點：(100,97) 終點：(416,413) 寬：316 高：316
Image Form No.0
316×316 of 8bits
Mouse on X:164 Y:170
Graylevel:72

11.4 Function Interface for Basic Image Processing

The Basic Image Processing interface provides some common functions of image processing such as threshold, color-channel processing, morphology operation…etc.
Users must select the **Source Image Form** and **Destination Image Form** first to open the function interface. All computing of the basic image process function output the result to the **Temporary Image** of the image-window to facilitate users to try and error. **Be sure to update the completed file before going to the next operation**.



11.5 Function Interface for Template Matching

The Template Matching interface provides the eCTM demo along with the interactive Image user interface. It includes the matching, selecting, and sorting functions. Beside, clicking on the analyzed outcome of the Image Form would be helpful for uses to understand the result of matching.

11.5.1 Choose Your Image Windows

In the template matching, users have to open two image-windows at least, one is Source Image Form to do template match; the other is Template Image Form to read the template that will be matched. Users have to assign the

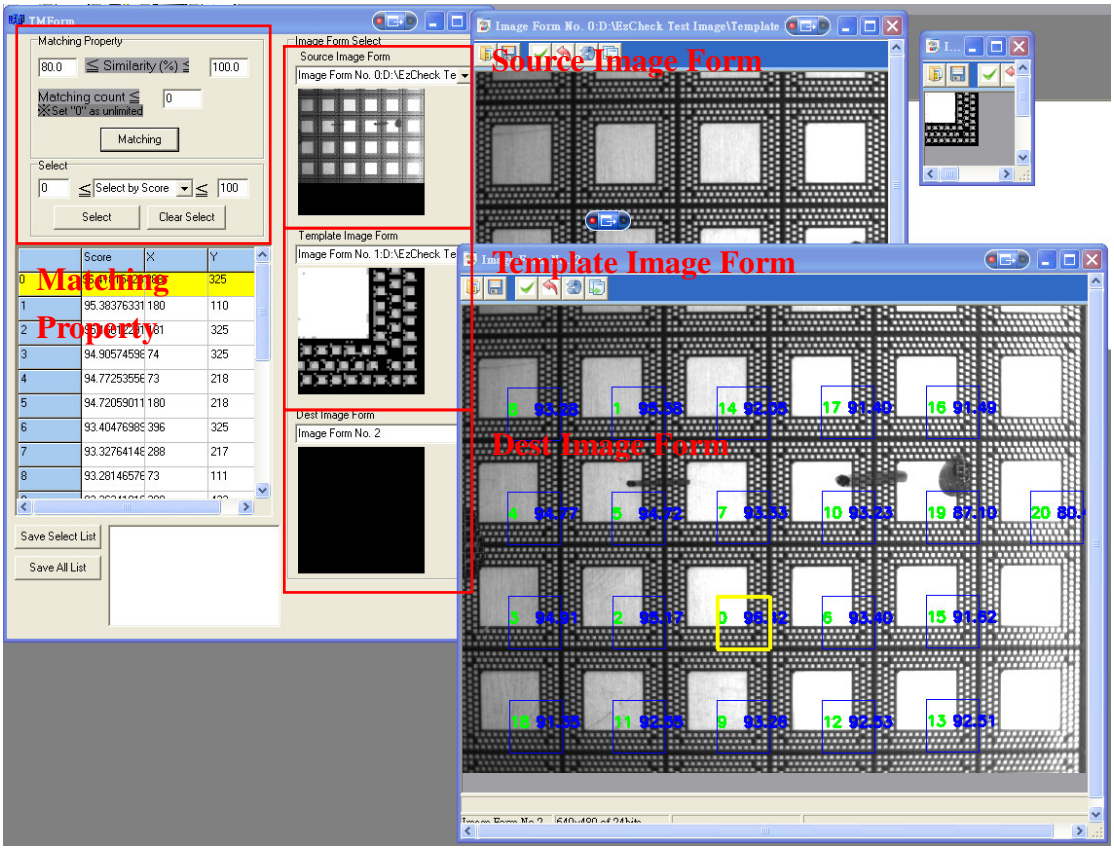
Dest Image Form to output the images, and this window can be the above two or a new one. Users can start to do the template matching after assing the three image-windows.

11.5.2 Template Matching and Similar Regions

After assigning windows, users can adjust the matching parameaters in the Matching Property and do the template matching. The similar regions will be shown in the form and screen after matching. Users can set the select conditions in this interface and display the result on the form and image. The system will sort the similar regions if users click the firt row of the form.

11.5.3 Image From Interaction

The system will mark the information of the similar area automatically if users click the output images on the Dest Image Form. The system will mark the position of the marked information if users click it.



11.6 Blob Analysis Interface

The Blob Analysis interface provides the eCBlob demo along with the interactive Image user interface. Users can test the blob types selecting, blob selecting, and see how the result image shows. Beside, clicking on the analyzed outcome of the Image Form would be helpful uses to understand the result of blob analysis.

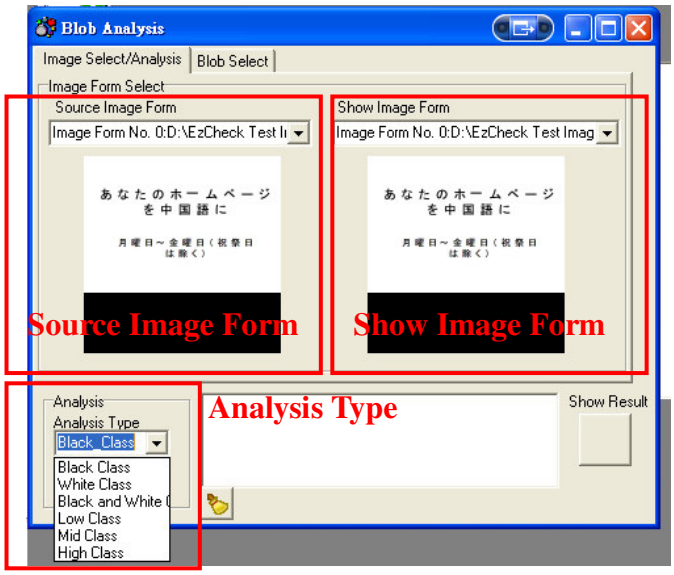
11.6.1 Choose Your Image Form Windows

Users must open one image-window at leat to look upon it as the Source Image Form in the Blob analysis interface. Users can assign anyone image-window to do the Show Image Form to output the blob analysis results.

Tips:
Our suggestion: Users should do image Threshold first to get the better results of blob analysis.

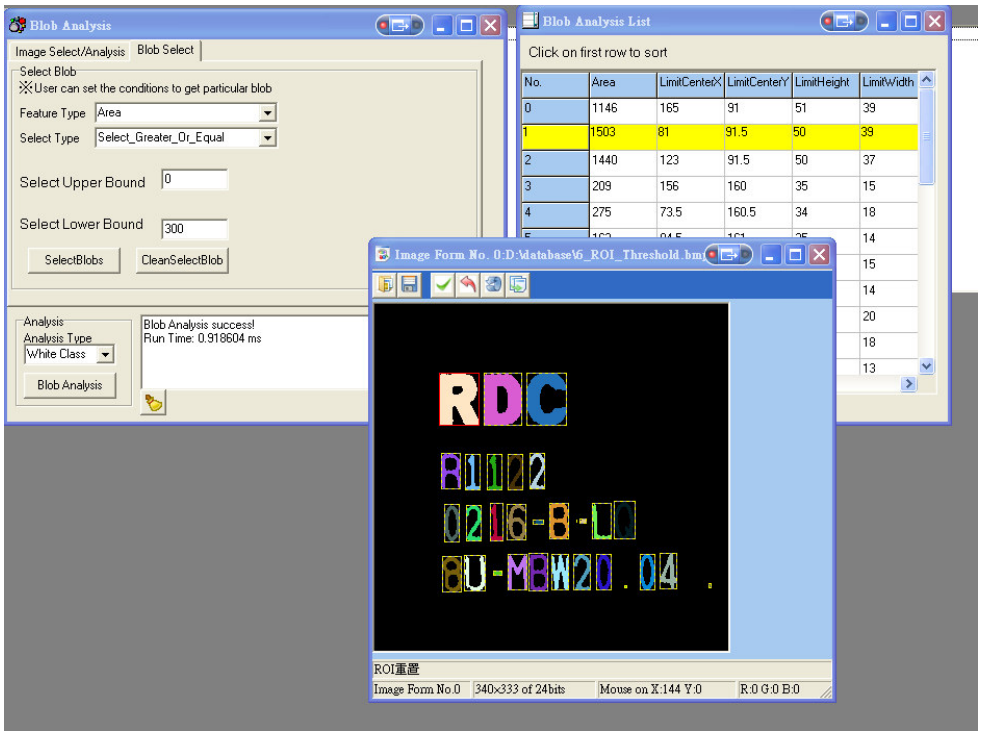
11.6.2 Blob Information

Assign the Analysis Type to do blob analysis for the image after assign the image-window, and display the results and information in the form after analyzing.
Users can choose different filter way in accordance with different characters of blobs in the Blob Select page. The results of selection will be shown in the form and image information real time.



11.6.3 Image Form Interaction

Click the output image of blob analysis in the Show Image Form will mark the similar area in the form automatically.



11.7 OCR Interface

The OCR interface provides the eCOCR demo. After some blob analysis steps such as selecting and merging, users can recognize the characters on the image by a pre-trained database.

11.7.1 Choose Your Image Form Windows

Users must open an image-window to look upon it as the Source Image Form in the OCR.

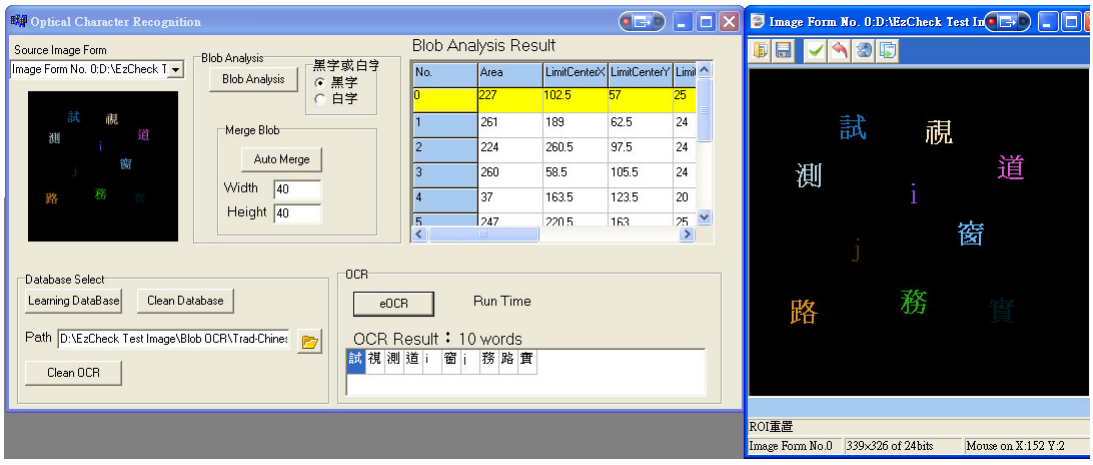
Tips:
Our suggestion: Users should do image Threshold first to get the better results of blob analysis.

11.7.2 Blob Information Management

Assign the black words or white words to do blob analysis. After assigning the system will mark all blobs and display the information in the form. You can use Select and Merge functions to stay the necessary blobs to get the better ruselts of OCR.

11.7.3 OCR and Learning DataBase

Users can choose the path via clicking the data path or type in by themselves. After entering, users can click the Learning DataBase to read the OCR data. Users can proceed with OCR after reading DataBase. As shown below:



Tips: Because of the limit of development environment the system doesn't support and display some fonts such as Korean. If the number of result is correct means the output

of the library is right.

11.8 Function Interface for Measure

The Gauge interface provides the eGauge demo along with the interactive image user interface. Users can create the eGauge tools on the specified image form and change their size and position by drag and drop. It demonstrates the convenience of the eGauge tool.

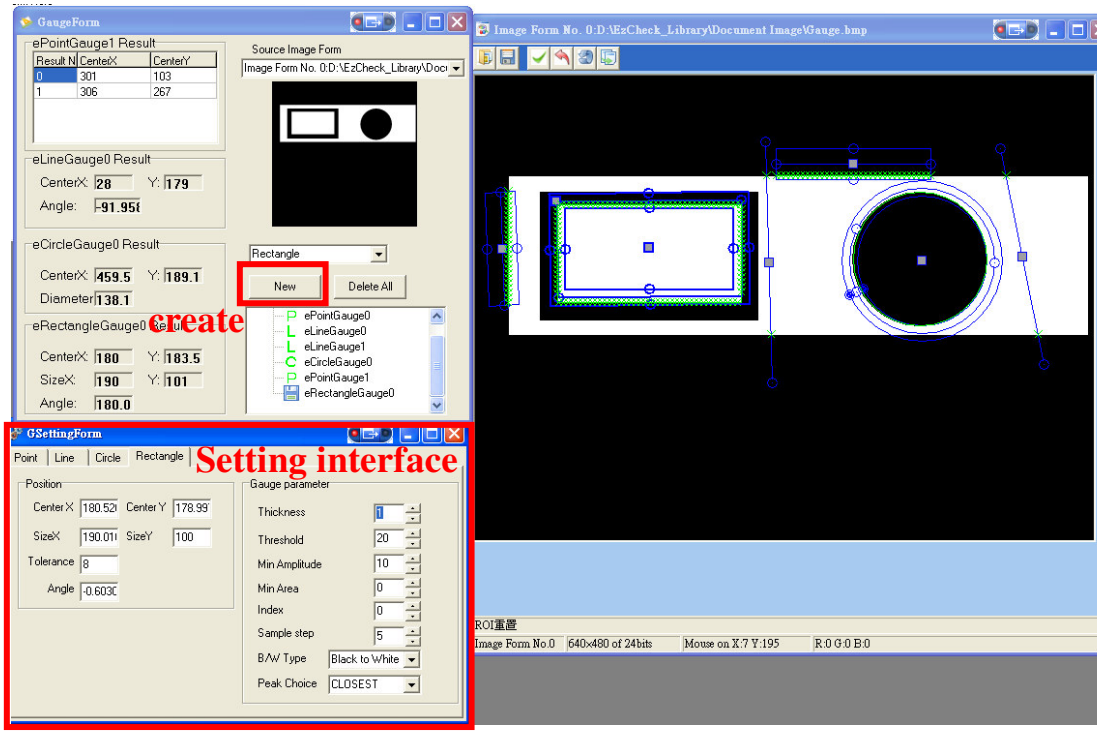
11.8.1 Choose Your Image Form Windows

Users must open an image-window to look upon it as the Source Image Form in the eGauge interface.

11.8.2 Create eGauge Tools And Set Elements

After assigning the image-window, users can choose the component they want such as Point, Line, Circle, and Rectangle. After choosing in the ComboBox, press New to create the component and show its parameter setting interface in the image-window. If the interface of parameter setting is closed, users can double-click the object of the below tree diagram that will be changed to open the setting interface.

Users can change the position, permissible value, and angle via dragging and dropping the characters in the image, also can input the value in the parameter interface. Both of their parameters are synchronous updates in real-time.


GaugeForm
ePointGauge1 Result
Result N
CenterX
CenterY
0
301
103
1
306
267
eLineGauge0 Result
CenterX: 28
Y: 179
Angle: -91.95
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
create
ePointGauge0
eLineGauge0
eLineGauge1
eCircleGauge0
ePointGauge1
eRectangleGauge0
GSettingForm
Point
Line
Circle
Rectangle
Setting interface
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
Image Form No. 0:D:\EzCheck_Library\Document Image\Gauge.bmp
ROI重置
Image Form No.0
640x480 of 24bits
Mouse on X:7 Y:195
R:0 G:0 B:0

11.9 Image Calibration Interface

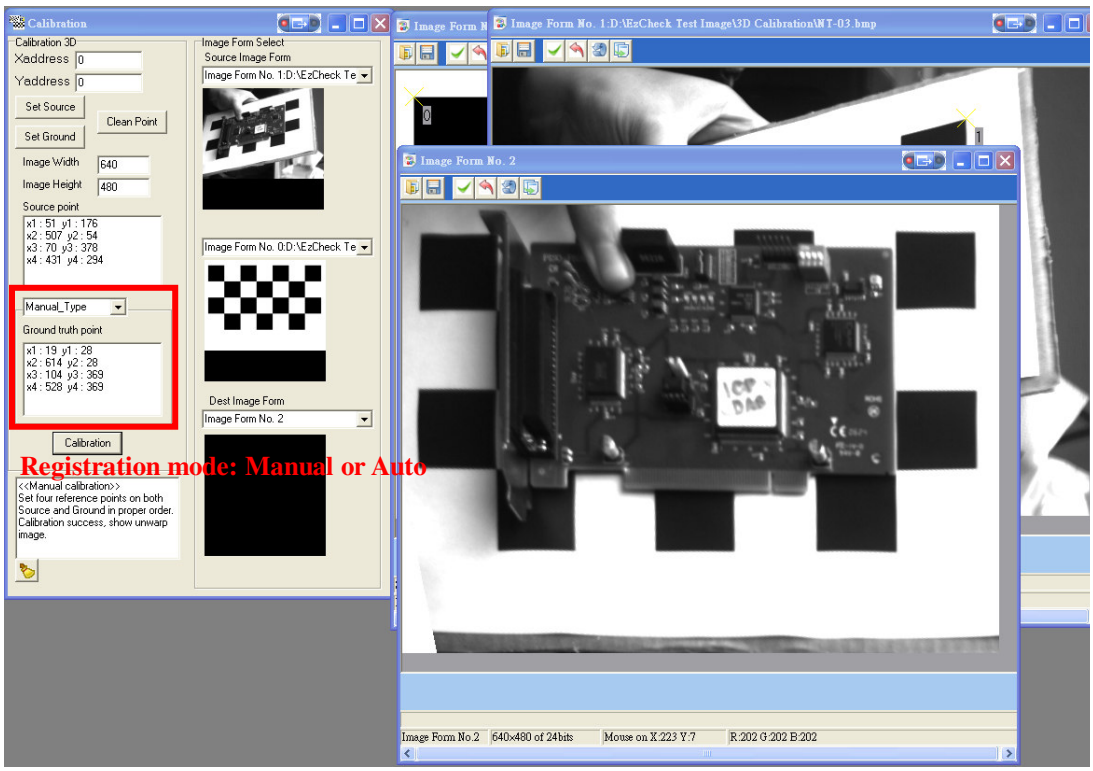
The Image Calibration interface provides the eCCalib3D demo along with the interactive image UI. Just click four relative points on both the source image and the pattern image, the source image will be unwarped without additional settings.

11.9.1 Choose Your Image Form Windows

In the interface of eCCalib3D function, users need at Least One or Two image-windows to look upon it as Source Image Form to read the the image that will be regulated. And need a Ground Truth Image Form to read the pattern image (It needn't do it if users choose the auto function). Users are also neet to assign an image-windoew to do the Dest Image Form.

11.9.2 Set the Reference Points

In Manual_Type Mode, users have to choose for calibration reference points on the Source Image Form and Ground Truth Image Form. Users can start to regulate after choosing the four reference points.
In Auto_Type Mode, users need to choose four reference points on the Source Image Form and input the value of Sample Width and Sample Height to regulate.
The "Click help" interface can help users more accurate to click on an image.


Calibration
Calibration 3D
Xaddress 0
Yaddress 0
Set Source
Clean Point
Set Ground
Image Width 640
Image Height 480
Source point
x1 : 51 y1 : 176
x2 : 507 y2 : 54
x3 : 70 y3 : 378
x4 : 431 y4 : 294
Manual_Type
Ground truth point
x1 : 19 y1 : 28
x2 : 614 y2 : 28
x3 : 104 y3 : 369
x4 : 528 y4 : 369
Calibration
<<Manual calibration>>
Set four reference points on both Source and Ground in proper order.
Calibration success, show unwarp image.
Image Form Select
Source Image Form
Image Form No. 1:D:\EzCheck Te
Image Form No. 0:D:\EzCheck Te
Dest Image Form
Image Form No. 2
Registration mode: Manual or Auto
Image Form No. 1:D:\EzCheck Test Image\3D Calibration\WT-03.bmp
Image Form No. 2
Image Form No.2 640x480 of 24bits Mouse on X:223 Y:7 R:202 G:202 B:202

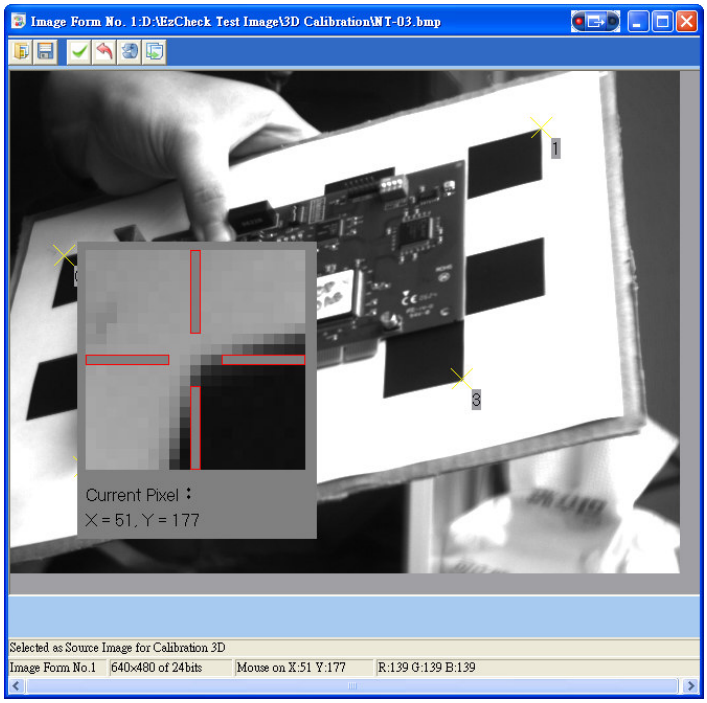


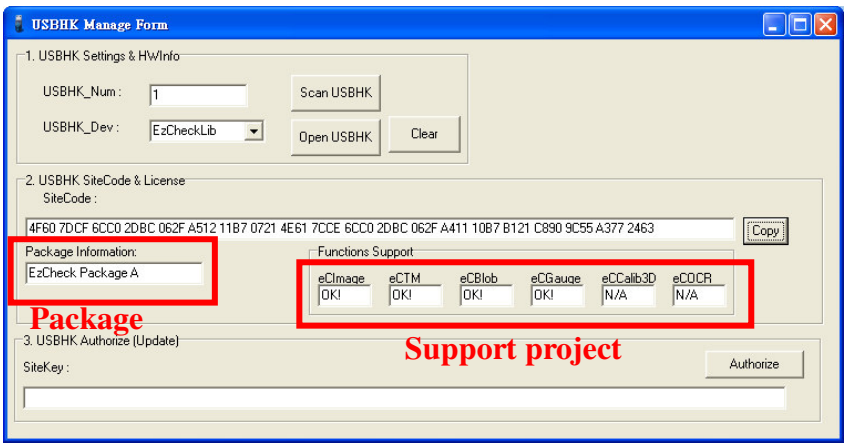
Image Form No. 1:D:\EzCheck Test Image\3D Calibration\WT-03.bmp
Current Pixel :
X = 51, Y = 177
Selected as Source Image for Calibration 3D
Image Form No.1 640x480 of 24bits Mouse on X:51 Y:177 R:139 G:139 B:139

12. USB Hardware Key and Package

12.1 Introduction

In order to defend the rights of users who buy EzCheck Vision Library, every EzCheck Vision Library will be protected by an USB Hardware Key from ICP DAS. ICP DAS also provides several packages of EzCheck Vision Library to fit different requirements of different users.

EzCheck Vision Library Grading Packages List						
Packages	**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 E	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ



▲ The management interface of the EzCheck Utility Hardware Key.

13. FAQs

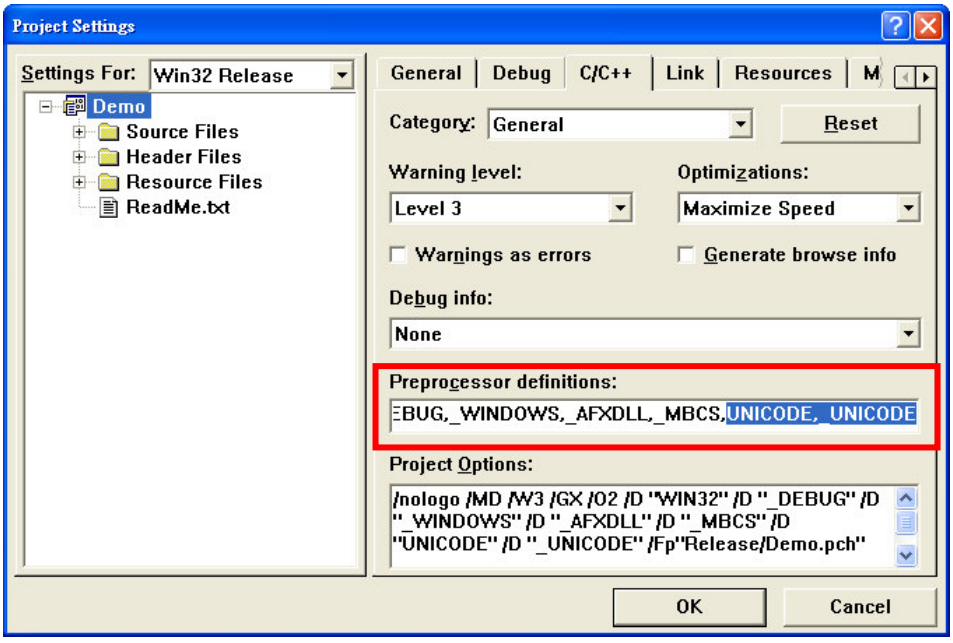
13.1 Borland C++ Builder

The eCOCR support numbers and characters recognition from several languages. As long as the OS can display that language, Users can recognize it on the image by the eCOCR. But the Borland C++ Builder (hereinafter referred to as BCB) don't support Unicode case that some words can't be displaied normally such as Japanese and Korean. But users can install the Unicode plug-in of BCB to save this problem.
Please refer to http://www.tmssoftware.com/site/tmsuni.asp TMS Unicode Component Pack or others Ucnicode plug-in of BCB.

13.2 VC

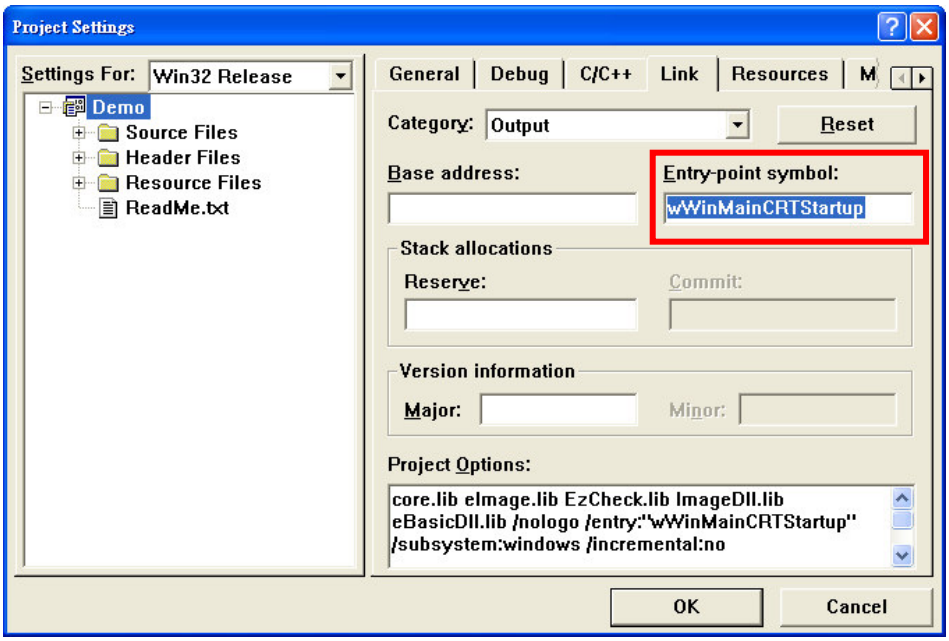
Visual C++ itself supports Unicode. Just have to set the related settings and program about Unicode. Set the Unicode settings by a case study of Visual C++ 6.0:

- Setting 1:
  Project->Setting->C/C++->Proprocessor definitions.
  Add in UNICODE and _UNICODE.



- Setting 2:

Project->Setting->Link->Category, choose the Output
Add wWinMainCRTStartup to the field of Entry-point symbol.



The string processing of the programming have to abide by the unicode way. Please refer to the Demo Program in C:\ICPDAS\EzCheck Vision Library\EzCheck for VC\Sample\Blob OCR.