CIDAR v.1.0

Container ID Automatic Reader

Manual de Referencia de la DLL

Tabla de Contenidos

Inicialización / Finalización

cidarInit

Inicializa el **Container ID Automatic Reader** (**CIDAR**). Carga las Redes Neuronales Artificiales del OCR e inicializa parámetros. Esta función debe llamarse antes de usar cualquiera de las otras funciones de esta librería.

```
long cidarInit ( long lAverageCharacterHeight,
                 bool bDuplicateLines = false,
                 bool bTrace = false );
```

Parámetros

lAvCharacterHeight	Altura media aproximada de los caracteres de los códigos a leer (en *pixels*). Si pasamos un **-1** la librería calcula de manera automática la altura. Si se selecciona altura automática, el tiempo de proceso aumenta considerablemente.
bDuplicateLines	Para imágenes capturadas con la mitad de las líneas pasaremos el valor 1 (true). Para los demás casos 0 (false).
bTrace	Este parámetro debe valer false.

Valor de Retorno

0 → Error.
1 → Ok.

cidarEnd

Libera la memoria reservada para el **Container ID Automatic Reader** (**CIDAR**).

```
void cidarEnd ( void );
```

Lectura de Códigos de Contenedores

cidarRead

Lee el código de identificación de contenedor contenido en una imagen. Esta función recibe como entrada una imagen. **Analiza la imagen** en busca de un código de identificación de contenedor y si lo encuentra, **lo lee,** intenta **verificarlo** usando el dígito de verificación del código y retorna el **texto ASCII** del mismo.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixels* de la imagen en **256 niveles de gris** (1 *byte* por *pixel*).

```
long cidarRead ( long lWidth,
                 long lHeight,
                 unsigned char * pbImageData );
```

Parámetros

lWidth	Anchura de la imagen que será analizada por el CIDAR (en p*ixels*).
lHeight	Altura de la imagen que será analizada por el CIDAR (en p*ixels*).
pbImageData	*Buffer* con los datos (*pixels*) de la imagen en 256 niveles de grises (1 *byte* por *pixel*).

Valor de Retorno

0 → Error.
1 → Ok.

cidarReadRGB24

Lee el código de identificación de contenedor contenido en una imagen. Esta función recibe como entrada una imagen. **Analiza la imagen** en busca de un código de identificación de contenedor y si lo encuentra, **lo lee,** intenta **verificarlo** usando el dígito de verificación del código y retorna el **texto ASCII** del mismo.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixels* de la imagen usando **tres bytes por píxel**.

```
long cidarReadRGB24 ( long lWidth,
                      long lHeight,
                      unsigned char * pbImageData,
                      bool bFlip = false );
```

Parámetros

lWidth	Anchura de la imagen que será analizada por el CIDAR (en p*ixels*).
lHeight	Altura de la imagen que será analizada por el CIDAR (en p*ixels*).
pbImageData	*Buffer* con los datos (*pixels*) de la imagen usando 3 bytes por píxel (Red, Green, Blue).
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

0 → Error.
1 → Ok.

cidarReadRGB32

Lee el código de identificación de contenedor contenido en una imagen. Esta función recibe como entrada una imagen. **Analiza la imagen** en busca de un código de identificación de contenedor y si lo encuentra, **lo lee,** intenta **verificarlo** usando el dígito de verificación del código y retorna el **texto ASCII** del mismo.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixels* de la imagen usando **cuatro bytes por píxel**.

```
long cidarReadRGB32 ( long lWidth,
                      long lHeight,
                      unsigned char * pbImageData,
                      bool bFlip = false );
```

Parámetros

lWidth	Anchura de la imagen que será analizada por el CIDAR (en p*ixels*).
lHeight	Altura de la imagen que será analizada por el CIDAR (en p*ixels*).
pbImageData	*Buffer* con los datos (*pixels*) de la imagen usando 4 bytes por píxel (Red, Green, Blue, Alfa).
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

0 → Error.
1 → Ok.

cidarReadBMP

Lee el código de identificación de contenedor contenido en una imagen. Esta función recibe como entrada una imagen. **Analiza la imagen** en busca de un código de identificación de contenedor y si lo encuentra, **lo lee,** intenta **verificarlo** usando el dígito de verificación del código y retorna el **texto ASCII** del mismo.

Como entrada a esta función debe proporcionarse una imagen en **formato BMP**.

```
long cidarReadBMP ( char * strFilename );
```

Parámetros

strFilename	Nombre del fichero BMP a procesar

Valor de Retorno

0 → Error.
1 → Ok.

cidarReadJPG

Lee el código de identificación de contenedor contenido en una imagen. Esta función recibe como entrada una imagen. **Analiza la imagen** en busca de un código de identificación de contenedor y si lo encuentra, **lo lee,** intenta **verificarlo** usando el dígito de verificación del código y retorna el **texto ASCII** del mismo.

Como entrada a esta función debe proporcionarse una imagen en **formato JPEG**.

```
long cidarReadJPG ( char * strFilename );
```

Parámetros

strFilename	Nombre del fichero JPEG a procesar

Valor de Retorno

0 → Error.
1 → Ok.

Consulta de Resultados

cidarCodeFound

Retorna si se ha encontrado y leído un código o no.

```
bool cidarCodeFound ( void );
```

Valor de Retorno

Retorna **true** si se ha encontrado y leído un código y **false** en caso contrario.

cidarCodeVerified

Retorna si el código de identificación de contenedor leído fue verificado con éxito usando el dígito de *checksum*.

```
bool cidarCodeVerified ( void );
```

Valor de Retorno

Retorna **true** si el dígito de *checksum* (el décimo-primero y último dígito) corresponde con el valor de *checksum* calculado a partir de los primeros diez caracteres del código leído. Retorna **false** en caso contrario.

cidarGetCode

Retorna el texto (**ASCII**) del código identificador reconocido en la última lectura.

```
long cidarGetCode ( char * strResult );
```

Parámetros

strResult	Variable donde se retorna la cadena de texto.

Valor de Retorno

0 → Error.
1 → Ok.

cidarGetNumberOfCharacters

Retorna el número de caracteres del código leído en la última imagen analizada por **Container ID Automatic Reader (CIDAR)**.

```
long cidarGetNumberOfCharacters ( );
```

Valor de Retorno

Retorna el número caracteres leídos.

cidarGetGlobalConfidence

Retorna un valor que expresa la fiabilidad de la lectura del código de identificación de la última imagen analizada por el **CIDAR.**

Este valor está expresado en tanto por ciento (0% – 100%).

```
float cidarGetGlobalConfidence ( );
```

Valor de Retorno

Retorna la fiabilidad de la lectura.

cidarGetAverageCharacterHeight

Retorna un valor que expresa la altura de carácter detectada automáticamente por el **CIDAR** en la última imagen analizada.

Este valor está expresado en *pixels*.

```
float cidarGetAverageCharacterHeight ( );
```

Valor de Retorno

Retorna la altura en *pixels* de los caracteres del último código de identificación leído.

cidarGetCharacterConfidence

Retorna un valor que expresa la fiabilidad de un carácter determinado en el último código de identificación de contenedor leído.

Este valor está expresado en tanto por ciento (0% – 100%).

```
float cidarGetCharacterConfidence ( long lIndex );
```

Parámetros

lIndex (0..n) carácter cuya fiabilidad deseamos consultar.

Valor de Retorno

Valor de Fiabilidad del carácter.

cidarGetRectangle

Retorna las coordenadas del rectángulo que engloba el código de identificación de contenedor leído en el último análisis.

```
void cidarGetRectangle ( long * plLeft,
                         long * plTop,
                         long * plRight,
                         long * plBottom );
```

Parámetros

plLeft	Coordenada X de la esquina Superior Izquierda del rectángulo
plTop	Coordenada Y de la esquina Superior Izquierda del rectángulo
plRight	Coordenada X de la esquina Inferior Derecha del rectángulo
plBottom	Coordenada Y de la esquina Inferior Derecha del rectángulo

Control del Tiempo

cidarGetProcessingTime

Retorna el tiempo de proceso en la última operación de lectura.

Este valor esta expresado en milisegundos.

```
long cidarGetProcessingTime ( );
```

Valor de Retorno

Tiempo de proceso en la última operación de lectura (en milisegundos).

cidarSetTimeOut

Indica al motor de reconocimiento un tiempo de proceso máximo tras el cual retornará con el resultado que haya obtenido hasta el momento.

Este valor está expresado en milisegundos.

```
void cidarSetTimeOut ( long lMilliseconds );
```

Parámetros

lMilliseconds	Tiempo máximo de proceso (en milisegundos).

Configuración Opcional

cidarSetCorrectionCoefficients

Esta función establece los coeficientes para el proceso de corrección de la distorsión que será aplicado a las imágenes antes de ser analizadas.

Los tipos de distorsiones que pueden corregirse son: tangencial (perspectiva horizontal y/o vertical) y rotación (inclinación).

Los coeficientes especificados en los parámetros serán aplicados a todas las imágenes analizadas hasta que esta función sea llamada con diferentes parámetros o se llame a la función ***cidarSetDistortionCorrectionOff***.

```
void cidarSetCorrectionCoefficients ( float fDistance,
                                      float fVerticalCoeff,
                                      float fHorizontalCoeff,
                                      float fAngle);
```

Parámetros

fDistance	Distancia aproximada entre la cámara y el objeto (en metros).
fVerticalCoeff	Coeficiente para corregir la Perspectiva Vertical.
fHorizontalCoeff	Coeficiente para corregir la Perspectiva Horizontal.
fAngle	Ángulo para corregir la Rotación (inclinación).

cidarSetDistortionCorrectionOff

Esta función desactiva el proceso de corrección de distorsión.

La función ***cidarSetCorrectionCoefficients*** debe usarse para volver a activar la corrección de distorsión.

```
void cidarSetDistortionCorrectionOff ();
```

cidarConfigureAutomaticCharacterHeight

Esta función configura el rango de la lectura en modo de *Altura de Caracteres Automática*.

Cuando ***cidarInit*** es llamado con el parámetro *lAvCharacterHeight* a -1, se selecciona el modo de *Altura de Caracteres Automática*. Por defecto, el rango de alturas escaneado va de 30 a 100 *pixels* de altura para los caracteres.

Usando esta función el usuario puede seleccionar un rango de alturas diferente.

```
long cidarConfigureAutomaticCharacterHeight ( long lMinCharHeight,
                                              long lMaxCharHeight );
```

Parámetros

lMinCharHeight	Mínima altura (en *pixels*) de los caracteres a reconocer.
lMaxCharHeight	Máxima altura (en *pixels*) de los caracteres a reconocer.

Valor de Retorno

0 → Error.
1 → Ok.

NOTA: Para recuperar la configuración por defecto, debe ejecutarse el siguiente código:

```
cidarConfigureAutomaticCharacterHeight ( 30, 100 );
```

cidarSetRectangle

Esta función asigna el rectángulo dentro de la imagen donde la inspección tendrá lugar. Sólo la región especificada será inspeccionada para buscar y reconocer el código de identificación del contenedor.

Esta función puede ser usada, por ejemplo, para acelerar el proceso cuando se conoce a priori la posición aproximada del código de identificación dentro de la imagen.

```
long cidarSetRectangle ( long lLeft,
                         long lTop,
                         long lWidth,
                         long lHeight );
```

Parámetros

lLeft, lTop	Coordenadas de la esquina superior izquierda del rectángulo (en *píxels*).
lWidth, lHeight	Dimensiones del rectángulo (en *píxels*).

Valor de Retorno

0 → Error.
1 → Ok.

NOTA:
El rectángulo de inspección especificado se aplicará a todas las inspecciones ejecutadas a partir de entonces, hasta que ***cidarSetRectangle*** sea llamado de nuevo con parámetros diferentes.
Si se desea volver inspeccionar la totalidad de la imagen, ***cidarSetRectangle*** debe ser llamado con todos sus parámetros a 0, es decir:

```
cidarSetRectangle (0, 0, 0, 0);
```

Por defecto, tras la inicialización de la librería con ***cidarInit***, el rectángulo de inspección corresponde a toda la imagen.

cidarConfigureCodeFormat

Proporciona información al CIDAR sobre los posibles formatos de códigos de identificación de contenedor a reconocer.

Los códigos de identificación de contenedor pueden ser horizontales o verticales. Si son horizontales, los caracteres pueden estar en una única línea o bien en dos líneas. En cualquiera de los casos anteriores, los caracteres del código de identificación pueden ser más oscuros que el fondo donde están impresos, o bien más claros que el fondo.

Por defecto, el CIDAR intenta reconocer todos los tipos posibles de código pero si se pueden restringir los formatos de códigos a reconocer, el tiempo de proceso disminuirá. Por ejemplo, si sabemos que para un sistema o instalación determinada la orientación de todos los códigos a reconocer serán verticales, podemos indicárselo a la librería CIDAR y no intentará reconocer códigos horizontales. Si estamos instalando un sistema donde sabemos que todos los contenedores serán oscuros y tendrán los códigos impresos en caracteres blancos o amarillos, podemos decir a CIDAR que los caracteres serán siempre más claros que el fondo, ahorrando así tiempo de proceso.

```
long cidarConfigureCodeFormat ( unsigned short usOrientation,
                                unsigned short usLines,
                                unsigned short usPolarity );
```

Parámetros

usOrientation — Indica la orientación de los códigos a reconocer. Puede tomar los siguientes valores:

- ***CODE_FORMAT_HORIZONTAL***
- **CODE_FORMAT_VERTICAL**
- **CODE_FORMAT_HORIZONTAL_OR_VERTICAL**

usLines — Indica el número de líneas que los códigos horizontales pueden tener:

- **CODE_FORMAT_ONE_LINE**
- **CODE_FORMAT_TWO_LINES**
- **CODE_FORMAT_ANY_LINES**

usPolarity — Indica la polaridad de los caracteres respecto al fondo donde están impresos:

- **CODE_FORMAT_BLACK_ON_WHITE**
- **CODE_FORMAT_WHITE_ON_BLACK**
- **CODE_FORMAT_ANY_ON_ANY**

Valor de Retorno

0 → Error.
1 → Ok.

Almacenamiento de datos de usuario en el HASP

cidarWriteHASP

Escribe datos en la memoria interna de la mochila de protección (*dongle*). Esta capacidad de almacenar datos en el HASP puede ser usada por el usuario para cualquier propósito. Los datos son encriptados automáticamente antes de escribirse en la memoria del HASP.

Pueden almacenarse un máximo de 24 bytes.

```
long cidarWriteHASP ( unsigned char * pData,
                      long lSize );
```

Parámetros

pData	Buffer con los datos a escribir en la memoria del HASP.
lSize	Tamaño (en bytes) de los datos a escribir (máximo 24).

Valor de Retorno

0 → Error.
1 → Ok.

cidarReadHASP

Lee los datos almacenados en la memoria interna de la mochila de protección (*dongle*). Esta capacidad de almacenar datos en el HASP puede ser usada por el usuario para cualquier propósito. Los datos son desencriptados automáticamente después de leerse de la memoria del HASP.

```
long cidarReadHASP ( unsigned char * pData,
                     long lSize );
```

Parámetros

pData	Buffer donde se almacenarán los datos leídos de la memoria del HASP.
lSize	Tamaño (en bytes) de los datos a leer.

Valor de Retorno

0 → Error.
1 → Ok.

Ejemplo de Uso

```
void func ()
{
   // La altura de los caracteres puede variar y no hay que duplicar
   // las líneas horizontales de la imagen.
   bool ok = (bool) cidarInit (30, false);
   if (ok)
       {
       // La altura de los caracteres puede variar entre 30 y 50 pixels.
       cidarConfigureAutomaticCharacterHeight (30, 50);

       unsigned char * buffer;     // Buffer de la imagen.
       bool verified;              // Código verificado o no.
       long numchars;              // Número de caracteres leídos.
       char text[32];              // String para almacenar el código leído.
       float Cf;                   // Fiabilidad de la lectura.
       float characterCf[32];      // Array para almacenar la fiabilidad
                                   // para cada carácter.

       // Reserva de memoria para el buffer.
       . . .
       // Obtener el buffer de la imagen en 256 niveles de gris (1 byte por
       // pixel) y almacenarlo en buffer.
       . . .
       // Las dimensiones de la imagen son 384x288 pixels.

       ok = (bool) cidarRead (384, 288, buffer);
       if( (ok) && ( cidarCodeFound ( ) )
           {
           verified = cidarCodeVerified ( );
           cidarGetCode ( text );
           cf = cidarGetGlobalConfidence ( );
           numchars = cidarGetNumberOfCharacters ( );
           for (long i = 0; i < numchars; i++)
               {
               characterCf[i] = cidarGetCharacterConfidence ( i );
               }
           }
       else
           {
           // Error al leer el código o código no encontrado.
           }
      cidarEnd ();
      }
     else
      {
      // Error al inicializar el CIDAR.
      }
 }
```