Luxand, Inc.
http://www.luxand.com

Luxand FaceCrop SDK 1.0

Face Detection Library

Developer's Guide

Table of Contents

Overview

Luxand FaceCrop is a cross-platform library intended for automatic cropping of faces from photos. It can be easily integrated into the customer's application. FaceCrop offers the API (Application Programming Interface) to automatically detect a face on photo, crop it into the desired resolution, and save to file. The library can just return the coordinates of a face (and not perform the cropping) if needed.

Luxand FaceCrop is available for all 32-bit and 64-bit versions of Windows and Linux, and 64-bit MacOS X. The SDK is supplied as a dynamic link library. FaceCrop SDK contains interface header files and sample applications for ASP.NET 2.0+, PHP 5.X, Microsoft Visual Basic .NET 2008, Microsoft C# 2008, C++, Microsoft Visual C++ 2008, Visual Basic 6.0 and Borland Delphi 6.0/7.0.

Requirements

The FaceCrop library supports the following platforms:

- Windows 2000/XP/2003/Vista/2008/Seven
- Linux (RHEL 5+, CentOS 5+ and other)
- Mac OS X 10.4+ x86_64

As the library is optimized for work with Intel processors, Intel processor is strongly recommended for better performance.

Minimum system requirements:

- 1.6 GHz processor
- 256 MB RAM
- 50 MB free disk space

Recommended system requirements:

- 2.0 GHz Intel Core 2 Duo processor
- 2 GB RAM

Technical Specifications

The FaceCrop library has the following technical specifications:

- Robust frontal face detection
- Head rotation support: –30..30 degrees of in-plane rotation and –30..30 degrees out-of-plane rotation
- Detection speed: from 0.01 to 0.7 sec*
    - Quick detection: 0.01 sec, webcam resolution, –15..15 degrees of in-plane head rotation
    - Reliable detection: 0.7 sec, digicam resolution, –30..30 degrees of in-plane head rotation
- Returned information: cropped image with detected face or coordinates of the face.
- Easy configuration of face detection parameters

* Measured on Intel 2.4 Ghz Processor

Installation

Windows

To install Luxand FaceCrop, run the installation file:

```
Luxand_FaceCropSDK_Setup.exe
```

and follow the instructions.

FaceCrop is installed to the `C:\Program Files\Luxand\FaceCrop` directory by default. FaceCrop is a copy-protected library, and must be activated with a license key before each use (see the Library Activation and Initialization chapter).

Linux/Mac OS X

Unpack the `Luxand_FaceCropSDK.tgz` archive into the desired directory.

Directory Structure

The FaceCrop directory contains the following directories and files:

bin\	FaceCrop binary files
bin\\linux_x86	FaceCrop Linux 32-bit binaries
bin\\linux_x86_64	FaceCrop Linux 64-bit binaries
bin\\osx_x86_64	FaceCrop Mac OS X 64-bit binaries
bin\\win32	FaceCrop Windows 32-bit binaries and stub library files
bin\\win64	FaceCrop Windows 64-bit binaries and stub library files
demo\	FaceCrop Demo application (win32)
include\	Header files
include\\.NET	.NET wrapper
include\\Delphi	Delphi header files
include\\C++	C/C++ header files
include\\php-extension	PHP extension
include\\VB	Visual Basic 6.0 header files
samples\	Sample applications
samples\\ASP.NET C#2008	ASP.NET sample application
samples\\C#2008	Microsoft C# 2008 sample application

samples\C++	C++ sample application
samples\Delphi	Delphi sample application
samples\MSVC2008	Microsoft Visual C++ 2008 sample application
samples\PHP	PHP sample application
samples\VB	Microsoft Visual Basic 6.0 sample application
samples\VB.NET2008	Microsoft VB.NET 2008 sample application

Sample Applications

FaceCrop is supplied with sample applications for ASP.NET 2.0+, PHP 5.X, Microsoft Visual Basic .NET 2008, Microsoft C# 2008, C++, Microsoft Visual C++ 2008, Visual Basic 6.0 and Borland Delphi 6.0/7.0. The sample applications can be found in the samples\ASP.NET C#2008, samples\PHP, samples\VB.NET2008, samples\C#2008, samples\C++, samples\MSVC2008, samples\VB and samples\Delphi directories.

Using FaceCrop with Programming Languages

To access the FaceCrop library functions, you need to use its binary file (`facecrop.dll` for Windows, `libfacecrop.so` for Linux, `libfacecrop.dylib` for Mac OS X) in your applications. Usually it is recommended to keep the binary file in the working directory of your application or in the directory specified in the PATH (Windows), LD_LIBRARY_PATH (Linux) or DYLD_LIBRARY_PATH (Mac OS X) environment variable. You need to include interface header files into your application project in order to use FaceCrop SDK.

Using with .NET (C#, VB and ASP)

For Microsoft .NET applications put `facecrop.dll` and the .NET wrapper (`include\.NET\FaceCrop.NET.dll`) into the working directory, add `FaceCrop.NET.dll` into the references (Select `Project->Add Reference->Browse` and choose library location) and use methods of class `fc` from `Luxand namespace`.

For instance, if you want to use the `fcFaceCrop` function, you need to call it in the following way:

```
Luxand.fc.FaceCrop
```

If you want the call to be simply `fc.FaceCrop` (without prefix `Luxand`, you need to add the following string to the beginning of the program:

```
using Luxand
```

Using with PHP

The FaceCrop library offers an extension for PHP called luxand_facecrop. Using this extension, one can call the FaceCrop functions from applications written in PHP. The method to connect this extension to PHP depends on the type of operating system in use.

Installing on UNIX

There are two ways to install FaceCrop PHP extension on UNIX-compatible OS (such as Linux, Mac OS X).

A. Installing as pluggable module

1. Copy `libfacecrop.so` (or `libfacecrop.dylib`) to `/usr/lib` and LuxandFaceCrop.h to `/usr/include` (or to `/usr/local/lib` and `/usr/local/include`)
2. Go to luxand_facecrop folder and type:

   ```
   phpize && ./configure && make && make install
   ```

3. Make sure you have `extension=luxand_facecrop.so` in your php.ini

B. Compiling luxand_facecrop into PHP

1. Copy `libfacecrop.so` (or `libfacecrop.dylib`) to `/usr/lib` and LuxandFaceCrop.h to `/usr/include` (or to `/usr/local/lib` and `/usr/local/include`)
2. Copy files from luxand_facecrop to $PHP_SOURCE_DIR/ext/luxand_facecrop
3. In PHP source root directory, run commands:

   ```
   rm configure && ./buildconf --force
   ```

4. Configure PHP with command:

   ```
   ./configure –with-luxand_facecrop
   ```

5. Run make && make install:

   ```
   make && make install
   ```

Installing on Windows

To install FaceCrop PHP extension on Windows with PHP 5.3:

1. Install Microsoft Visual Studio 2008.
2. Install Windows SDK 6.1
3. Get a PHP 5.3 current snapshot or 5.3 stable release archive (but do not extract yet)
4. Create the folder "`c:\php-sdk`"
5. Unpack the binary-tools.zip archive into this directory. There should be one sub-directory called "`bin`" and one called "`script`". The binary-tools.zip archive can be downloaded from http://pecl2.php.net/downloads/php-windows-builds/php-libs/
6. Open the "windows sdk 6.1 cmd shell" (it is available from the Start menu group) and execute the following commands inside the shell:

   ```
   setenv /x86 /xp /release
   cd c:\php-sdk\
   bin\phpsdk_setvars.bat
   bin\phpsdk_buildtree.bat php53dev
   ```

7. Extract the PHP snapshot from 3. to:

   ```
   c:\php-sdk\php53dev\vc9\x86
   ```

   so the following directory is created:

   ```
   c:\php-sdk\php53dev\vc9\x86\php-5.3XXXXXXX
   ```

8. Copy `facecrop.lib` to:

   ```
   c:\php-sdk\php53dev\vc9\x86\deps\lib\
   ```

9. Copy luxand_facecrop directory to:

   ```
   c:\php-sdk\php53dev\vc9\x86\php-5.3XXXXXXX\ext\
   ```

10. If you want to build PHP with Apache support, copy all from lib\ and include\ directories from the Apache directory to:

    ```
    c:\php-sdk\php53dev\vc9\x86\deps\lib\
    ```

    and:

    ```
    c:\php-sdk\php53dev\vc9\x86\deps\include\
    ```

11. Run in the windows-sdk-shell:

    ```
    cd C:\php-sdk\php53dev\vc9\x86\php-5.3XXXXXXX
    buildconf
    configure --help
    ```

12. Run configure with options you wish to include --with-luxand_facecrop, e.g.:

    ```
    configure --enable-apache2-2handler --enable-
    isapi --enable-cli --with-luxand_facecrop
    ```

    ```
    nmake
    ```

13. If you need the resulting PHP to be zipped, run after this also:

    ```
    nmake snap
    ```

14. The compiled PHP supporting FaceCrop is now under:

    ```
    c:\php-sdk\php53dev\vc9\x86\php-
    5.3XXXXXXX\Release_TS
    ```

    Copy it with facecrop.dll to the desired location (e.g. to c:\Windows\System32\)

Functions Available in PHP

In the PHP extension, there are the following functions available:

```
fcActivate,
fcGetHardwareID,
fcGetLicenseInfo,
fcFaceCrop,
fcGetFacePosition,
fcSetFaceScale,
fcSetFaceShift,
```

```
fcSetDetectionThreshold,
fcSetDetectionPerformance,
fcSetJpegQuality,
fcCreateContextID,
fcFreeContextID.
```

As you work with the extension, it is necessary to use the mechanism of contexts (see the Thread Safety and Contexts chapter). Therefore all specified functions, except:

```
fcActivate,
fcGetHardwareID,
fcGetLicenseInfo,
```

accept context ID as the final parameter.

You should work with the PHP extension as follows:

1. Activate the library using the fcActivate call
2. Create the context using the fcCreateContextID call
3. Call extension functions by transferring context ID as the final parameter of each used function.
4. When the program is over, free the context using fcFreeContextID.

The fcFinalize call in PHP applications is not required.

Example:

```
<?php
fcActivate("YOUR_LICENSE_KEY");
$c_id = fcCreateContextID();
fcFaceCrop($infile, $outfile, 128, 196, $c_id);
fcFreeContextID($c_id);
?>
```

Using with C/C++

For MSVC2008 applications, you need to include the header file `include\C++\LuxandFaceCrop.h` in your project, and the stub library file `facecrop.lib` in your project: put `facecrop.lib` into the project directory, and add string "facecrop.lib" to the `Project Properties->Linker->Input->Additional Dependencies`. Also put `facecrop.dll` into the application working directory.

Using with Delphi

For Delphi applications, put `facecrop.dll` into the working directory and use the `include\Delphi\LuxandFaceCrop.pas` unit in your project.

Data types

Luxand FaceCrop declares the PInteger type – a pointer to Integer, and PHBITMAP type - a pointer to HBITMAP.

Delphi Declaration:

```
PInteger = ^Integer;

PHBITMAP = ^HBITMAP;
```

Using with Visual Basic 6.0

For Visual Basic 6.0 applications put the Visual Basic wrapper (`include\VB\FaceCrop-vb.dll`) into the project directory, add `FaceCrop-vb.dll` into the references (Select `Project->References->Browse` and choose library location) and add LuxandFaceCrop.bas module to your project (Select `Project->Add module->Existing` and choose module location). Also put `facecrop.dll` into the application working directory.

Using in Command Line

FaceCrop is supplied with a command line utility `facecrop.exe`. It has the following syntax:

```
facecrop <in_file><out_file>
     --key <activation key>
     [--size <width>x<height>]
     [--scale <scale_value>]
     [--shift <shift_x><shift_y>]
     [--jpeg_quality <quality>]
     [--performance <performance_level>]
     [--threshold <threshold_value>]
     [--coords]
```

Parameters:
`<in_file><out_file>` – specifies the names of input and output files.

`--key <activation key>` – activates the library like the fcActivate function.

`--size <width>x<height>` – specifies the values of width and height to use in fcFaceCrop and fcGetFacePosition functions.

`--scale <scale_value>` – sets the scale like the fcSetFaceScale function.

`--shift <shift_x>x<shift_y>` – sets the face shift like the fcSetFaceShift function.

`--jpeg_quality <quality>` – sets the JPEG compression quality like the fcSetJpegQuality function.

`--performance <performance_level>` – sets the performance level like the fcSetDetectionPerformance function.

`--threshold <threshold_value>` – sets the face detection threshold like the fcSetDetectionThreshold function.

`--coords` – if this key is specified, the command returns the coordinates of the face-containing rectangular like the fcGetFacePosition function, else it saves the cropped face into the `<out_file>` file like the fcFaceCrop function.

Redistributables

The following files should be redistributed with the end-user application:

Windows	bin\win32\facecrop.dll (for 32-bit systems) include\.NET\FaceCrop.NET.dll (for .NET applications) bin\win64\facecrop.dll (for 64-bit systems)
Linux	bin\linux_x86\libfcrop.so (for 32-bit systems) bin\linux_x86_64\libfcrop.so (for 64-bit systems)
Mac OS X	bin\osx_x86_64\libfcrop.dylib (for 64-bit systems)

Thread Safety and Contexts

This chapter describes the use of FaceCrop SDK in applications, processing multiple threads. If your program is executed in a single thread (by default it happens in almost all environments, except PHP), you can skip this chapter.

The FaceCrop SDK library can be used in multithread applications. All its functions are thread-safe, except the following:

```
fcActivate
fcFinalize
```

However if your application is multithreaded, you need to take into account that calling any function that sets face detection parameters (e.g. fcSetFaceScale) has impact on all threads. Therefore FaceCrop SDK has the mechanism of contexts. This mechanism allows setting parameters of detection for each thread separately.

As you work with contexts, you need to use special variants of functions with postfix "_C". Functions with this postfix as the final parameter accept context ID. These variants of functions are determined for all functions of FaceCrop SDK, except:

```
fcActivate
fcCreateContextID
fcFreeContextID
fcFinalize
```

Syntax of calling a function with postfix "_C" is distinguished only by presence of an additional parameter (context ID). For example, for the following function:

```
int fcFaceCrop(char * inFileName, char * outFileName, int
width, int height);
```

the corresponding function working with context will be:

```
int fcFaceCrop_C(char * inFileName, char * outFileName, int
width, int height, int ContextID);
```

If context ID is not specified (function without postfix “_C” is called), the function is considered to be working in special context “by default”.

Contexts are created with the fcCreateContextID call and freed with fcFreeContextID call.

To work with contexts, it is necessary to follow these steps:

1. Call fcActivate
2. Using the fcCreateContextID function, you need to get ContextID in each thread and then transfer it to the FaceCrop functions, which are used.
3. On finishing work with FaceCrop in the thread it is necessary to call the fcFreeContextID function in this thread.
4. Call fcFinalize

The context mechanism is used in all functions of the PHP extension because in many realizations of the PHP interpreter, end-user applications are executed in different threads of one interpreter. More detailed information can be found in the Using with PHP chapter of the Using FaceCrop with Programming Languages chapter.

fcCreateContextID Function

Creates a context ID.

C++ Syntax:

```
int fcCreateContextID(ref int ContextID);
```

Delphi Syntax:

```
function fcCreateContextID(ContextID: PInteger): integer;
```

C# Syntax:

```
int fc.CreateContextID(ref int ContextID);
```

PHP Syntax:

```
int fcCreateContextID(void);
```

VB Syntax:

```
Function fcVBCreateContextID(ByRef contextID As Long) As Long
```

Parameters:
ContextID – context ID for current thread.

Return Value:

Returns fcErrorOk if successful.

fcFreeContextID Function

Frees a context ID.

C++ Syntax:

```
int fcFreeContextID(int ContextID);
```

Delphi Syntax:

```
function fcFreeContextID(ContextID: integer): integer;
```

C# Syntax:

```
int fc.FreeContextID(int ContextID);
```

PHP Syntax:

```
int fcFreeContextID(int $ContextID);
```

VB Syntax:

```
Function fcVBFreeContextID(ByVal contextID As Long) As Long
```

Parameters:
`ContextID` – context ID for current thread.

Return Value:

Returns fcErrorOk if successful.

Using Luxand FaceCrop

The usage level of the library depends on the functionality required from Luxand FaceCrop SDK. The typical scenario is:

1. Activate and initialize FaceCrop by calling up the fcActivate function with the key sent by Luxand, Inc.
2. Set face detection parameters if needed (fcSetDetectionThreshold, fcSetDetectionPerformance, fcSetJpegQuality).
3. Set face cropping parameters if needed (fcSetFaceShift, fcSetFaceScale).
4. Use FaceCrop functions:
    - Detect and crop face from the image file and save it to the image file (fcFaceCrop).
    - Detect and crop face from the image file and save it to the HBITMAP handle (fcFaceCrop_FileToHBITMAP).
    - Detect and crop face from the HBITMAP handle and save it to the HBITMAP handle (fcFaceCrop_HBITMAPToHBITMAP).
    - Detect face from the image file and get its position (fcGetFacePosition)
    - Detect face from the HBITMAP handle and get its position (fcGetFacePosition_HBITMAP)
5. Finalize the FaceCrop library (fcFinalize function).

Library Activation and Initialization

FaceCrop is a copy-protected library and must be activated with a registration key before its use. You need to pass the license key received from Luxand, Inc. to the fcActivate function before using Luxand FaceCrop functions. Almost all FaceCrop SDK functions will return the fcErrorNotActivated error code in case the library is not activated. To retrieve your license information, call fcGetLicenseInfo. This function returns the name the library is licensed to. You may need to use the fcGetHardwareID function to obtain your hardware ID if your license is restricted to one machine only. Additionally, you can find out a hardware ID by running the hardwareid program (ShowHardwareID.exe for Windows), which is located in `/bin/` directory of the subdirectory that corresponds to the platform used.

fcGetHardwareID Function

Generates a Hardware ID code.

C++ Syntax:

```
int fcGetHardwareID(char* HardwareID);
```

Delphi Syntax:

```
function fcGetHardwareID(HardwareID: PChar): integer;
```

C# Syntax:

```
int fc.GetHardwareID(string HardwareID);
```

PHP Syntax:

```
string fcGetHardwareID(void);
```

VB Syntax:

```
Function fcVBGetHardwareID(ByRef HardwareID As Byte) As Long
```

Parameters:
`HardwareID` – address of the null-terminated string for receiving the Hardware ID code. You need to allocate at least 32 bytes of memory for this variable.

Return Value:

Returns fcErrorOk or hardware id in PHP, if successful.

fcActivate Function

Activates and initializes the FaceCrop library. It should be called before using any face cropping functions.

C++ Syntax:

```
int fcActivate(char* LicenseKey);
```

Delphi Syntax:

```
function fcActivate(LicenseKey: PChar): integer;
```

C# Syntax:

```
int fc.Activate(out string LicenseKey);
```

PHP Syntax:

```
int fcActivate(string $LicenseKey);
```

VB Syntax:

```
Function fcVBActivate(ByVal LicenseKey As String) As Long
```

Parameters:
`LicenseKey` – license key you received from Luxand, Inc.

Return Value:

Returns fcErrorOk if the registration key is valid and not expired.

fcGetLicenseInfo Function

Retrieves license information.

C++ Syntax:

```
int fcGetLicenseInfo(char* LicenseInfo);
```

Delphi Syntax:

```
function fcGetLicenseInfo(LicenseInfo: PChar): integer;
```

C# Syntax:

```
int fcGetLicenseInfo(out string LicenseInfo);
```

PHP Syntax:

```
string fcGetLicenseInfo(void);
```

VB Syntax:

```
Function fcVBGetLicenseInfo(ByRef LicenseInfo As Byte) As Long
```

Parameters:
`LicenseInfo` – address of the null-terminated string for receiving the license information. This variable should be allocated no less than 256 bytes of memory.

Return Value:

Returns fcErrorOk (or license info string in PHP), if successful.

fcFinalize Function

Finalizes the FaceCrop library. Should be called when the application is exited (note: it is not required to call this function in PHP).

C++ Syntax:

```
int fcFinalize();
```

Delphi Syntax:

```
function fcFinalize: integer;
```

C# Syntax:

```
int fc.Finalize();
```

VB Syntax:

```
Function fcVBFinalize() As Long
```

Return Value:

Returns fcErrorOk if successful.

Face Cropping Functions

You can use the fcFaceCrop to detect a frontal face in an image and crop it. Also you can use the fcGetFacePosition to just get the coordinates of the detected face in an image.

fcFaceCrop Function

Crops the detected face from an image file and saves it to another image file.

C++ Syntax:

```
int fcFaceCrop(char* inFileName, char* outFileName, int width,
int height);
```

Delphi Syntax:

```
function fcFaceCrop(inFileName: PChar; outFileName: PChar;
width: integer; height: integer): integer;
```

C# Syntax:

```
int fc.FaceCrop(string inFileName, string outFileName, int
width, int height);
```

PHP Syntax:

```
int fcFaceCrop(string $inFileName, string $outFileName, int
$width, int $height, int $ContextID);
```

VB Syntax:

```
Function fcVBFaceCrop(ByVal inFileName As String, ByVal
outFileName As String, ByVal width As Long, ByVal height As
Long) As Long
```

Parameters:
inFileName – path to the image file to crop the face from.
outFileName – path to the image file to save face to.
width – width of the cropped face image.
height – height of the cropped face image.

Return Value:

Returns fcErrorOk if successful.

fcFaceCrop_FileToHBITMAP Function

Crops the detected face from an image file and saves it to the HBITMAP handle. The function is available only on the Windows platform.

C++ Syntax:

```
int fcFaceCrop_FileToHBITMAP(char* inFileName, HBITMAP*
outHBITMAP, int width, int height);
```

Delphi Syntax:

```
function fcFaceCrop_FileToHBITMAP(inFileName: PChar;
outHBITMAP: PHBITMAP; width: integer; height: integer):
integer;
```

C# Syntax:

```
int fc.FaceCrop_FileToHBITMAP(string inFileName, ref IntPtr
outHBITMAP, int width, int height);
```

VB Syntax:

```
Function fcVBFaceCrop_FileToHBITMAP(ByVal inFileName As
String, ByRef outHBITMAP As Long, ByVal width As Long, ByVal
height As Long) As Long
```

Parameters:
inFileName – path to the image file to crop the face from.
outHBITMAP – a handle of the image to save cropped face to.
width – width of the cropped face image.
height – height of the cropped face image.

Return Value:

Returns fcErrorOk if successful.

fcFaceCrop_HBITMAPToHBITMAP Function

Crops the detected face from an HBITMAP and saves it to the HBITMAP. The function is available only on the Windows platform.

C++ Syntax:

```
int fcFaceCrop_HBITMAPToHBITMAP(HBITMAP* inHBITMAP, HBITMAP*
outHBITMAP, int width, int height);
```

Delphi Syntax:

```
function fcFaceCrop_HBITMAPToHBITMAP(inHBITMAP: PHBITMAP;
outHBITMAP: PHBITMAP; width: integer; height: integer):
integer;
```

C# Syntax:

```
int fc.FaceCrop_HBITMAPToHBITMAP(ref IntPtr inHBITMAP, ref
IntPtr outHBITMAP, int width, int height);
```

VB Syntax:

```
Function fcVBFaceCrop_HBITMAPToHBITMAP(ByRef inHBITMAP As
Long, ByRef outHBITMAP As Long, ByVal width As Long, ByVal
height As Long) As Long
```

Parameters:
`inHBITMAP` – a handle of the image to crop the face from.
`outHBITMAP` – a handle of the image to save cropped face to.
`width` – width of the cropped face image.
`height` – height of the cropped face image.

Return Value:

Returns fcErrorOk if successful.

fcGetFacePosition Function

Detects a frontal face in an image and stores information about the coordinates of the face.

C++ Syntax:

```
int fcGetFacePosition(char* inFileName, int width, int height,
int* x1, int* y1, int* x2, int* y2);
```

Delphi Syntax:

```
function fcGetFacePosition(inFileName: PChar; width: integer;
height: integer; x1: PInteger; y1: PInteger; x2: PInteger; y2:
PInteger): integer;
```

C# Syntax:

```
int fc.GetFacePosition(string inFileName, int width, int
height, ref int x1, ref int y1, ref int x2, ref int y2);
```

PHP Syntax:

```
int fcGetFacePosition(string $inFileName, int $width, int
$height, int $x1, int $y1, int $x2, int $y2, int $ContextID);
```

VB Syntax:

```
Function fcVBGetFacePosition(ByVal inFileName As String, ByVal
width As Long, ByVal height As Long, ByRef x1 As Long, ByRef
y1 As Long, ByRef x2 As Long, ByRef y2 As Long) As Long
```

Parameters:
`inFileName` – path to the image file to detect the face coordinates on.
`width` – width of the detected face.
`height` – height of the detected face.
`x1, y1` – coordinates of the left-top corner of the detected face.
`x2, y2` – coordinates of the right-bottom corner of the detected face.

Return Value:

Returns fcErrorOk if successful.

fcGetFacePosition_HBITMAP Function

Detects a frontal face in a HBITMAP handle and stores information about the coordinates of the containing face rectangle. The function is available only on the Windows platform.

C++ Syntax:

```
int fcGetFacePosition_HBITMAP(HBITMAP* inHBITMAP, int width,
int height, int* x1, int* y1, int* x2, int* y2);
```

Delphi Syntax:

```
function fcGetFacePosition_HBITMAP(inHBITMAP: PHBITMAP; width:
integer; height: integer; x1: PInteger; y1: PInteger; x2:
PInteger; y2: PInteger): integer;
```

C# Syntax:

```
int fc.GetFacePosition_HBITMAP(ref IntPtr inHBITMAP, int
width, int height, ref int x1, ref int y1, ref int x2, ref int
y2);
```

VB Syntax:

```
Function fcVBGetFacePosition_HBITMAP(ByRef inHBITMAP As Long,
ByVal width As Long, ByVal height As Long, ByRef x1 As Long,
ByRef y1 As Long, ByRef x2 As Long, ByRef y2 As Long) As Long
```

Parameters:

`inHBITMAP` – a handle of the image to detect the face coordinates on.
`width` – width of the detected face.
`height` – height of the detected face.
`x1, y1` – coordinates of the left-top corner of the detected face.
`x2, y2` – coordinates of the right-bottom corner of the detected face.

Return Value:

Returns fcErrorOk if successful.

Face Detection and Cropping Parameters

You can use the fcCropFace and fcGetFacePosition functions to detect and crop a frontal face. The performance and reliability of face detection is controlled by the fcSetDetectionPerfomance, fcSetDetectionThreshold. Parameters of the cropped face are controlled by the fcSetFaceScale and fcSetFaceShift functions.
Three levels of face detection performance are stored in the `FcDetectionPerformance` type.

C++ Declaration:

```
typedef enum {
     fcPerformanceRealtime = 0,
     fcPerformanceNormal,
     fcPerformancePrecise
} FcDetectionPerformance;
```

The `fcPerformanceRealtime` value detects only those faces that take a considerable part of the photo, whose rotation angle (in the picture plane) lies within the range from -15 degrees to +15 degrees. This value should be used if faces need to be detected in real time.

The `fcPerformanceNormal` value detects small faces, whose rotation angle (in the picture plane) lies within the range from -30 degrees to +30 degrees. This is the default value and suitable for the majority of applications.

The `fcPerformancePrecise` value detects even the smallest faces, whose rotation angle (in the picture plane) lies within the range from -30 degrees to +30 degrees. This value should be used if it is necessary to detect a face on a large photo and position the face in the returned rectangle as accurately as possible. Please note that it takes more time to detect a face when `fcPerformancePrecise` is used.

Face detection performance is determined by the fcSetDetectionPerformance function.

fcSetDetectionPerfomance Function

Sets the level of detection performance. The default value is fcPerformanceNormal.

C++ Syntax:

```
int fcSetDetectionPerformance(FcDetectionPerformance level);
```

Delphi Syntax:

```
function fcSetDetectionPerformance(level:
FcDetectionPerformance): integer;
```

C# Syntax:

```
int fc.SetDetectionPerformance(fc.FcDetectionPerformance
level);
```

PHP Syntax:

```
int fcSetDetectionPerformance(int $level, int $ContextID);
```

VB Syntax:

```
Function fcVBSetDetectionPerformance(ByVal level As
FcDetectionPerformance) As Long
```

Parameters:
`level` – level of face detection performance.

Return Value:

Returns fcErrorOk if successful.

fcSetDetectionThreshold Function

Sets a threshold value for face detection. The default value is 3.

The function allows adjusting the sensitivity of detection. If the threshold value is set to a higher value, the detector will only recognize faces with sharp, clearly defined details, thus reducing the number of false positive detections. Setting the threshold lower allows detecting more faces with less clearly defined features at the expense of increased number of false positives.

C++ Syntax:

```
int fcSetDetectionThreshold(int Threshold);
```

Delphi Syntax:

```
function fcSetDetectionThreshold(Threshold: integer): integer;
```

C# Syntax:

```
int fc.SetDetectionThreshold(int Threshold);
```

PHP Syntax:

```
int fcSetDetectionThreshold(int $Threshold, int $ContextID);
```

VB Syntax:

```
Function fcVBSetDetectionThreshold(ByVal threshold As Long) As
Long
```

Parameters:
`Threshold` – threshold value.

Return Value:

Returns fcErrorOk if successful.

fcSetFaceScale Function

Sets face scaling rate for cropping. The higher the value, the larger part of the resulting rectangle is taken by the face. The lower the value, the larger part of the resulting rectangle is taken by background. The default value is 1.

C++ Syntax:

```
int fcSetFaceScale(float scale);
```

Delphi Syntax:

```
function fcSetFaceScale(scale: single): integer;
```

C# Syntax:

```
int fc.SetFaceScale(float scale);
```

PHP Syntax:

```
int fcSetFaceScale(float $scale, int $ContextID);
```

VB Syntax:

```
Function fcVBSetFaceScale(ByVal faceScale As Single) As Long
```

Parameters:
`scale` – scale value. It must be higher than 0.

Return Value:

Returns fcErrorOk if successful.

fcSetFaceShift Function

Sets off-centering rate for the cropped face. Parameter values are set approximately within the face width (for example, if you select 0.5, the face will be shifted from the center approximately by half of its width). By default, both parameters are equal to 0.

C++ Syntax:

```
int fcSetFaceShift(float ShiftX, float ShiftY);
```

Delphi Syntax:

```
function fcSetFaceShift(ShiftX: single; ScaleY: single):
integer;
```

C# Syntax:

```
int fc.SetFaceShift(float ShiftX, float ShiftY);
```

PHP Syntax:

```
int fcSetFaceShift(float $ShiftX, float $ShiftY, int
$ContextID);
```

VB Syntax:

```
Function fcVBSetFaceShift(ByVal shiftX As Single, ByVal shiftY
As Single) As Long
```

Parameters:
`ShiftX` – horizontal shift value. It is set as fraction of the width of the cropped rectangle.
`ShiftY` – vertical shift value. It is set as fraction of the height of the cropped rectangle.

Return Value:

Returns fcErrorOk if successful.

Support functions

fcSetJpegQuality

Sets quality of JPEG compression to use in the fcFaceCrop function.

C++ Syntax:

```
int fcSetJpegQuality(int Quality);
```

Delphi Syntax:

```
function fcSetJpegQuality(Quality: integer): integer;
```

C# Syntax:

```
int fc.SetJpegQuality(int Quality);
```

PHP Syntax:

```
int fcSetJpegQuality(int $Quality, int $ContextID);
```

VB Syntax:

```
Function fcVBSetJpegQuality Lib(ByVal quality As Long) As Long
```

Parameters:
`Quality` – quality of JPEG compression. Varies from 0 to 100.

Return Value:

Returns fcErrorOk if successful.

Error Codes

The FaceCrop library defines the following error codes:

Error Name	Value
fcErrorOk	0
fcErrorFailed	–1
fcErrorNotActivated	–2
fcErrorOutOfMemory	–3
fcErrorInvalidArgument	–4
fcErrorIOError	–5
fcErrorImageTooSmall	–6
fcErrorFaceNotFound	–7
fcErrorInsufficientBufferSize	–8
fcUnsupportedImageExtension	–9
fcCannotOpenFile	–10
fcCannotCreateFile	–11
fcBadFileFormat	–12
fcFileNotFound	–13

Library Information

The FaceCrop library uses libjpeg © IJG; libpng © Glenn Randers-Pehrson; easybmp © The EasyBMP Project (http://easybmp.sourceforge.net); RSA Data Security, Inc. MD5 Message-Digest Algorithm.