OptimiData

... for optimized data handling

JPEG2000 Software Development Kit for C/C++

Reference Manual

Version 1.6

from 2004-07-29

(Windows and Linux Versions)

www.optimidata.com

Contents

1 Introduction

The JPEG2000 C-SDK allows simple compression and decompression of baseline JPEG2000 codestreams. Based on the International Image Compression Standard ITU-T.800, ISO/IEC 15444-1 the JPEG2000 C-SDK offers the following functions:

- Lossless and lossy compression of continuous tone still images using a single algorithm
- Improved compression performance with respect to most of other lossless or lossy image compression schemes
- Full control about the generated file size of a lossy compressed image
- Support of progressive decompression in several image viewers
- Baseline JPEG2000 codestreams can be decoded by any other JPEG2000 image decoding software

For further information about the JPEG2000 image compression algorithm please refer to the appropriate standardization document. The document can be found at

http://www.itu.org

For further information about this C-SDK please refer the website:

http://www.optimidata.com

2 Installation

The JPEG2000 C-SDK is delivered as a:

- Zip archive **OptimiData_JP2_SDK_E.zip** for Windows platforms
- Zipped tar archive **OptimiData_JP2_SDK_E.tgz** for Linux platforms

To extract the archive use an unzip utility as Winzip (to be found at http://www.winzip.com) for Windows platforms or the tar utility of the Unix or Linux operating system.

The files of the JPEG2000 C-SDK are extracted into the following files and folders:

Windows (C-SDK files):

```
.\opd_jp2sdk\dynamic_library\opd_jp2.dll
.\opd_jp2sdk\dynamic_library\opd_jp2.lib
.\opd_jp2sdk\static_library\opd_jp2_static.lib
.\opd_jp2sdk\include\opd_jp2.h
```

This documentation

```
.\opd_jp2skd\documentation\Manual_OptimiData_JP2_SDK_E.pdf
```

Windows (Demo program files):

```
.\opd_jp2sdk\sample\source\getopt.c
.\opd_jp2sdk\sample\source\main.c
.\opd_jp2sdk\sample\source\bmp.c
.\opd_jp2sdk\sample\source\jpeg.c
.\opd_jp2sdk\sample\include\getopt.h
.\opd_jp2sdk\sample\include\main.h
.\opd_jp2sdk\sample\include\jpeglib\jconfig.h
.\opd_jp2sdk\sample\include\jpeglib\jerror.h
.\opd_jp2sdk\sample\include\jpeglib\jmorecfg.h
.\opd_jp2sdk\sample\include\jpeglib\jpeglib.h
```

```
.\opd_jp2sdk\sample\include\opd_jp2.h
.\opd_jp2sdk\sample\library\opd_jp2_static.lib
.\opd_jp2sdk\sample\library\jpeglib.lib
.\opd_jp2sdk\sample\out\opd_jp2.exe
.\opd_jp2sdk\sample\images\image1.bmp
.\opd_jp2sdk\sample\images\image1.jpg
.\opd_jp2sdk\sample\images\image1.jp2
.\opd_jp2sdk\sample\opd_jp2.dsw
.\opd_jp2sdk\sample\opd_jp2.dsp
.\opd_jp2sdk\viewer\JP2view.exe
.\opd_jp2sdk\viewer\opd_jp2.dll
```

Linux (C-SDK files):

```
./opd_jp2sdk_linux/shared_library/libopd_jp2.so
./opd_jp2sdk_linux/static_library/libopd_jp2.a
./opd_jp2sdk_linux/include/opd_jp2.h
```

This documentation

```
./opd_jp2skd_linux/documentation/ Manual_OptimiData_JP2_SDK_E.pdf
```

Linux (Demo program files):

```
./opd_jp2sdk_linux/sample/source/getopt.c
./opd_jp2sdk_linux/sample/source/main.c
./opd_jp2sdk_linux/sample/source/bmp.c
./opd_jp2sdk_linux/sample/source/jpeg.c
./opd_jp2sdk_linux/sample/include/getopt.h
./opd_jp2sdk_linux/sample/include/main.h
./opd_jp2sdk_linux/sample/include/opd_jp2.h
./opd_jp2sdk_linux/sample/library/libopd_jp2.a
./opd_jp2sdk_linux/sample/out/opd_jp2
./opd_jp2sdk_linux/sample/images/image1.bmp
```

```
./opd_jp2sdk_linux/sample/images/image1.jpg
./opd_jp2sdk_linux/sample/images/image1.jp2
./opd_jp2sdk_linux/sample/makefile
```

2.1 Release History

2.1.1 Differences between Version 1.1 and Version 1.0

The Version 1.1 of this SDK allows the compression and decompression of a JPEG2000 File Format in addition to the JPEG2000 codestream. Please refer the appropriate compression and decompression chapters in this manual as well as the JPEG2000 international standard for further information about the JPEG2000 file format and the JPEG2000 codestream.

2.1.2 Differences between Version 1.2 and Version 1.1

The Version 1.2 of this SDK includes a windows executable allowing the compression and decompression of JPEG2000 files. This windows programm offers a graphical user interface to control the compression and decompression functions as well as the registration of the SDK. The graphical user interface also allows the viewing of the original as well as the compressed images.

2.1.3 Differences between Version 1.5 and Version 1.2

The Version 1.5 of this SDK was subject of major internal restructuring of the code to achieve performance improvements up to 40% compared with version 1.2. Therefore also the user interface was modified and simplified. Please refer to Chapter 3 for the description of the corresponding interfaces.

2.1.4 Differences between Version 1.6 and Version 1.5

The Version 1.6 of this SDK introduces the following functionalities:

- Decompression of images up to 16bits / image channel
- Downscaled decompression
- Decompression of a given quality
- Decompression of tiled JPEG2000 images
- Performance improvements

For further about downscaled decompression please refer to section 3.2 of this manual.

Important Note:

The calling convention of the new decompression functions have been changed with this version. Please refer to the appropriate section 3.2 of this manual.

3 Library provided functions

The JPEG2000 library provides functions for:

- Compression of a raw pixel matrix into a JPEG2000 data stream
- Decompression of a JPEG2000 data stream to a raw pixel matrix
- Registration of the library (Windows only)

This chapter describes the functions and the data structures of the library at a higher detailed level.

3.1 Compression functions

There are two different compression function provided by the library depending whether standard licensing or explicit licensing during compilation time is used. Standard licensing is done via the windows registry and registry keys are written and / or read during compression and decompression. Explicit licensing is performed during compilation time and is bypassing the windows registry. No registry keys are read or written with explicit licensing.

The following function is defined in the header file "opd_jp2.h" and is used to compress an image given as a raw pixel matrix in memory into a compressed JPEG2000 stream. This function is using standard licensing via the windows registry.

Name:	`int opd_jp2_compress(…)`

Parameters:	`opd_image_t      *image`
	`unsigned char    *dest,`
	`int              dest_size`
	`opd_jp2_param    *param`

Return value:	`<  0 : Error code`
	`>= 0 : Number of bytes created`

The following function is defined in the header file “opd_jp2.h” and is used to compress an image given as a raw pixel matrix in memory into a compressed JPEG2000 stream. This function requires a valid license string as an additional parameter and is used for explicit licensing during compilation time. This function bypasses the license check via the windows registry. The linux version of the library ignores this parameter.

Name:	`int opd_jp2_compress_licensed(…)`	
Parameters:	`opd_image_t`	`*image`
	`unsigned char`	`*dest,`
	`int`	`dest_size`
	`opd_jp2_param`	`*param`
	`unsigned char`	`*licenseString`
Return value:	`<  0 : Error code`	
	`>= 0 : Number of bytes created`	

3.1.1 Parameter Description

The following chapter describes the parameters of the encoding function.

opd_image_t *image - Parameter

The input parameter `*image` of the encoding function contains the raw pixel data of the image to be compressed. The raw pixel data is stored in memory. The type of the `*image` parameter structure is defined in the header file “opd_jp2.h” and described below:

```
typedef struct {
    /* Simple structure for an interleaved image
     * introduced with version 1.5 of the library */
    unsigned char     *pucData;
    unsigned int      uiDataOffset;
    unsigned int      uiWidth;
    unsigned int      uiHeight;
    unsigned int      uiChannels;
    int               iDx;
    int               iDy;
    int               iDc;
    int               iPrecision;
} opd_image_t;
```

The parameter ***pucData** points to the raw pixel data and the rest of the parameters of the structure describe the interpretation of the raw pixel data.

The unsigned integer parameter **uiDataOffset** indices the position of the first component (red component for coloured images) of the first pixel positioned at the the topmost left corner of the image (e.g. windows bitmaps are organized bottom-up. The topmost left corner is therefore not at the beginning of the data array)

The unsigned integer parameter **uiWidth** describes the width (number of columns) of the raw input image and the unsigned integer parameter uiHeight describes the height (number of rows) of the image.

The unsigned integer parameter **uiChannels** contains the number of individual colour components of the raw image. For coloured images this parameter is most likely 3 for red, green and blue colour components respectively.

The integer parameter **iDx** describes the number of bytes to add to a certain pixel position within the raw image array to address the rightmost neighbored pixel of the same color component. This parameter can be negative.

The integer parameter **iDy** describes the number of bytes to add to a certain pixel position within the raw image array to address the bottommost neighbored pixel of the same color component. This parameter can be negative (e.g. windows bitmap are organized bottom-up and can be addressed using negative values for **iDy**)

The integer parameter **iDc** describes the number of bytes to add to a certain pixel of a given color component to address the next color component of the same pixel. This parameter can be negative. (e.g. windows bitmaps are organized in BGR. Here the Blue-component can be addressed by adding –1 to the position of the red component)

The integer parameter **iPrecision** describes the number of valid bits of a single pixel and color component within the raw data image. Negative values of iPrecision indicates signed pixel values while positive values indicate unsigned values. This parameter is most likely 8.

void *dest - Parameter

The ***dest** parameter points to a pre-allocated memory buffer the encoding function writes the compressed data into. After successfully compression of the pixel data the *dest memory block contains the result of the compression process.

int dest_size – Parameter

The Integer parameter dest_size contains the size of the output memory buffer. The dest_size parameter must not exceed the size of the preallocated *dest memory buffer to avoid buffer overflow.

opd_jp2_param *param – Parameter

The structured param parameter contains the parameters to control the compression process. The structure of the parameter is defined in the file “opd_jp2.h” and described below:

```
typedef struct opd_jp2_param_t {
    int filesize;
    int lossless;
    int resolution_levels;
    int quality_layers;
    int bCodestream;
    int iLibraryVersion;
} opd_jp2_param;
```

The integer parameter **`filesize`** describes the expected size of the compressed image data. If the filesize parameter is larger than the size of the output buffer, only as many bytes as can be placed in the buffer are written.

The integer parameter **`lossless`** determines whether lossless or lossy compression is expected. Use 1 (TRUE) for lossless and 0 (FALSE) for lossy image compression. Please note the size of the compressed image data for lossless compression cannot be estimated before compression. For lossless compression please use a destination buffer large enough to keep at least the uncompressed image and choose a large value even for the filesize parameter.

The **`resolution_levels`** parameter determines the number of different image resolutions the JPEG2000 file should allow access to. A resolution_level of 1 means that only the access to the full resolution is possible. A resolution_level parameter of 2 identifies that also access to the half-resolution (quarter size) image is allowed. Typical values are between 4 and 6.

The **`quality_layer`** parameter determines the number of quality chunks the image should contain. Typically JPEG2000 images only contain a single quality layer. Please refer to the JPEG2000 International Standard for more detailed information about this parameter.

The **`bCodestream`** parameter determines whether a JPEG2000 codestream rather than the JPEG2000 file format containing additional metadata has to be compressed. A value of 0 (false) determines that the file format will be generated. All other values (true) determine the generation of a JPEG2000 codestream. Please refer to the JPEG2000 International Standard for more detailed information about the file format and the codestream.

The integer parameter **`iLibraryVersion`** is used for run-time checking of the correct linkage of the library versions. This parameter has to be set to the pre-defined value `OPD_JP2_LIBRARY_VERSION` defined in the header file.

3.2 Decompression functions

There are two different decompression functions provided by the library depending whether standard licensing or explicit licensing during compilation time is used.

The following function is declared in “opd_jp2.h” and allows the decompression of JPEG2000 images with standard licensing via the windows registry.

Name:	`int opd_jp2_decompress(…)`	
Parameter:	`unsigned char`	`*src`
	`int`	`src_size`
	`opd_image_t`	`**image`
Return Value:	`< 0:`	`Error code`
	`OPD_NO_ERROR:`	`success`

The following function is declared in “opd_jp2.h” and allows the decompression of JPEG2000 images with explicit licensing at compilation time.

Name:	`int opd_jp2_decompress_licensed(…)`	
Parameter:	`unsigned char`	`*src`
	`int`	`src_size`
	`j2k_image_t`	`**image`
	`unsigned char`	`*licenseString`
Return Value:	`< 0:`	`Error code`
	`OPD_NO_ERROR:`	`success`

The following function is declared in “opd_jp2.h” and allows the decompression of a downscaled version of the JPEG2000 image and up to a given quality using standard licensing via the windows registry.

Name:	`int opd_jp2_decompress_scaled(…)`	
Parameter:	`unsigned char`	`*src`
	`int`	`src_size`
	`j2k_image_t`	`*image`
	`int`	`iDownscale`
	`int`	`iQuality`
Return Value:	`< 0:`	`Error code`
	`OPD_NO_ERROR:`	`success`

unsigned char* src – Parameter

This parameter points to a memory buffer containing the compressed image for the decompression.

int src_size – Parameter

This parameter contains the number of bytes of the compressed data in the input memory buffer. This parameter must be positive.

opd_image_t **image – Parameter

opd_image_t *image – Parameter

This parameter is used to return the uncompressed image data. It points to a **opd_image_t** structure (as described in chapter 3.1) which is filled by the decompression function. The memory to hold the uncompressed image data is allocated by the library using the standard **malloc()** functions and has to be freed by the calling application using the **free()** function.

NOTE: This parameter has been changed for the new decompression functions due to performance

improvements. If you are using the new enhanced decompression function, update your software appropriately.

int iDownscale – Parameter

A value greater than 0 for this parameter forces the decoder to create a downscaled version of the original image. Typical values are:

- 0 – decodes at the original size
- 1 – decodes half the width and height of the image
- 2 – decodes ¼ the width and height of the image
- 3 – respecive 2^i downscale factor

int iQuality – Parameter

A value between 0 and 100 forces the decoder not to decode the full quality of the original image. The decoding process is stopped before the complete image data is decoded and the resulting image appears smoothed (at a lower visual quality).

A value of 100 means full quality while a value of 0 means worst quality.

3.3 Memory management functions

To avoid memory management problems using the OptimiData JPEG2000 C-SDK in a multi-language environment e.g. with Borland Delphi, the C-SDK provides memory management functions for allocation and freeing memory shared with the library.

3.3.1 Memory allocation functions

There is only a single function to allocate memory provided by the JPEG2000 library:

Name:	`void *opd_jp2_alloc(size_t size)`
Parameter:	`size_t          size`
Return Value:	`pointer to the allocated memory block or NULL if there is an error`

size_t size – Parameter

This parameter specifies the amount of bytes the JPEG2000 library is requested to allocate.

3.3.2 Memory deallocation functions

There are two different memory deallocation (free()) functions, a simple and a specialized image deallocation function.

Name:	`void opd_jp2_free(void *ptr)`
Parameter:	`void            *ptr`

This function deallocates the memory addressed by the pointer given by `*ptr`. To avoid memory management problems the memory block given to the function should be allocated using the library-provided `opd_op2_alloc()` function.

void *ptr – Parameter

The pointer to the memory block for deallocation.

Name:	`void opd_jp2_freeImage( opd_image_t *image)`
Parameter:	`opd_image_t      *image`

This function is used to deallocate the image structure content created by the JPEG2000 library to hold the decompressed image data as a result of the `opd_jp2_decode(…)` function call.

opd_image_t *image – Parameter

This parameter points to the image structure content to be deallocated. If the `*image` structure is not generated by the decompression function `opd_jp2_decode(…)` please ensure that for the allocation of the structure only the JPEG2000 provided library allocation function `opd_op2_alloc()` have been used.

3.4 Registration functions (Windows only)

There are two functions allowing either the registration of the JPEG2000 C-SDK or the check of the current license. There are two different licenses available:

- 1 month evaluation license
- unlimited license

The evaluation license is available using the JPEG2000 C-SDK without any license code and will run for 1 month starting from the first call to any compression or decompression operation. The unlimited license can be registered using a license code.

3.4.1 Registering the JPEG2000 C-SDK

The following function is declared in the file “opd_jp2.h” and allows the registration of the JPEG2000 C-SDK.

Name:	`int opd_register(…)`	
Parameter:	`unsigned char *szLicense`	
Return Value:	`< 0:`	`Error code`
	`OPD_NO_ERROR:`	`success`

unsigned char *szLicense – Parameter

This parameter acts as an input Parameter to the registration function and contains a 0-terminated string carrying the license code. The license code is of the format “xxxx-xxxxxxxx” consisting of two character strings divided by a hyphen.

Using NULL for the **szLicense** parameter implies the registration of a 1 month demo license.

Possible return values are defined in “opd_jp2.h” and are listed in Table 1.

Table 1: Return values of opd_register(…)

Code	Remark
OPD_NO_ERROR	Success
OPD_ERROR_INVALID_LICENSE	The license code is invalid
OPD_ERROR_UNREGISTERED	Cannot write the registration information to the system registry. The library remains unregistered. Please

Code	Remark
	check your rights for installing software
OPD_ERROR_ALREADY_REGISTERED	The library is already registered. Licenses can only be “upgraded” that means an unlimited license cannot be overwritten with a demo license.

3.4.2 Checking the current license

The following function allows the check of the currently installed license and is declared in the file “`opd_jp2.h`”.

Name: `int opd_checkRegistration()`

Parameter: none

Return Value: one of the status values listed in Table 2.

Table 2: Status values for the checkRegistration() function

Code	Remark
OPD_ERROR_UNREGISTERED	Cannot write the registration information to the system registry. The library remains unregistered. Please check your rights for installing software
OPD_ERROR_DEMO_LICENSE	A demo license is registered. The demo license allows full functionality witin a 1-month

Code	Remark
	time period
OPD_ERROR_DEMO_EXPIRED	The demo license is expired. Please check the website http://www.optimidata.com for licensing information

4 Sample program

4.1 Commandline tool (Windows and LINUX)

The deliveries of the JPEG2000 C-SDK contains a Microsoft Visual C project of a simple commandline tool illustrating the usage of the C-SDK. This commandline tool allows the conversion of Windows Bitmap files into baseline JPEG2000 files and vice-versa. The commandline-tool can also be used to register the JPEG2000 C-SDK on Windows machines.

The project has to be compiled using the Version 6.0 of Microsoft Visual C.

The commandline-tool can be started using the MS-DOS command prompt typing:

```
X:> opd_jp2 [options] –i input_file –o output_file
```

The input and the output image formats are determined by the file extension of the input and the output file name. The following input and output image file formats are supported:

- Windows Bitmap file (Extension *.bmp)
- JPEG file (Extension *.jpg)
- JPEG2000 file (Extension *.jp2)
- Baseline JPEG2000 file (JPEG2000 codestream, Extension *.j2k)

Additional options are allowed for the compression of an image file into JPEG2000. These options are:

-I : specifies the input file name

-o: specifies the output file name

-s <integer>[k/m] specifies the size of the output JPEG2000 file. If the given value ends with the character ‘k’ the given number specifies the size in kilobytes. If the value ends with a ‘m’ the expected filesize is given in Megabytes. Otherwise the size is specified in bytes.

-r: <float> Specifies the compression ratio of the compressed image file with respect to the original file size. A value of 10 forces the output file to become 10 times smaller than the input Image Bitmap file

If neither the option –s nor the option –r is specified, lossless JPEG2000 compression is enforced.

NOTE: IF THE INPUT IMAGE FILE IS COMPRESSED (E.G. JPEG) AND LOSSLESS IMAGE COMPRESSION IS ENFORCED, THE OUTPUT IMAGE FILE MAY BECOME LARGER THAN THE INPUT IMAGE FILE.

The following options are allowed for the compression of an image into the old JPEG format:

-q: <integer> This option specifies the resulting JPEG quality. The value is between 1 and 100.

-q: *lossless* If the string “lossless” is specified, lossless JPEG compression is enforced[1]

If the option –q is not specified, the default JPEG quality value 90 is used.

NOTE: IF THE INPUT IMAGE FILE IS COMPRESSED (E.G. JPEG2000) AND LOSSLESS IMAGE COMPRESSION IS ENFORCED, THE OUTPUT IMAGE FILE MAY BECOME LARGER THAN THE INPUT IMAGE FILE.

-l: <String> This option is used for registration of the JPEG2000 library. Please specify the license string for registration of the codec. The demo-license is installed automatically if the commandline-tool is first started. No further registration is necessary for a demo license.

[1] Lossless option not supported with library version 1.5

4.2 Windows Viewer program

The Windows viewer program is delivered as a compiled version and not available in source code. It allows the compression and decompression of images into JPEG2000 and other general purpose image formats as the old JPEG or Windows Bitmap. The viewer program can be found in the subdirectory

```
.\opd_jp2sdk\viewer\JP2view.exe
```

of the delivery. Just start the viewer program by double-clicking the symbol of the file **`opd_viewer.exe`**.

NOTE: THE EXECUTABLE NEEDS THE OPD_JP2.DLL DYNAMIC LINK LIBRARY FOR CORRECT EXECUTION. PLEASE MAKE SURE THAT THE DYNAMIC LINK LIBRARY IS EITHER IN THE SAME DIRECTORY AS THE EXECUTABLE OR IN THE WINDOWS SEARCH PATH OR WINDOWS SYSTEM FOLDER.

4.2.1 Features of the Viewer program

The viewer program `JP2view.exe` offers the following features:

- Compression and Decompression of various image file formats (Windows Bitmap *.bmp, Standard JPEG *.jpg) into JPEG2000 (file format *.jp2 and codestream *.j2k)
- Display of multiple image files in one viewer
- Drag and Drop of multiple image files into the viewer
- Zoom in, Zoom out in 10% steps with indication of the current zooming step
- Window display modes: cascade, tile horizontally, tile vertically
- License Registration
- Supported operating system (Windows XP / 2000 / NT / 95 / 98 / ME)

- Image information in windows title bar: filename, image size, bit depth, compression ratio and filesize
- Current image information in status bar: Mouse position, pixel value (RGB) and zoom factor
- Context menu (right mouse button click)

4.3 Usage of the Viewer program

Important note:

YOU SHOULD RUN THE PROGRAM WITH TRUE COLOR MODE. PLEASE CHOOSE THE APPROPRIATE COLOR DEPTH OF YOUR GRAPHICS ADAPTER

A splash-screen, as depicted in Figure 1, is shown and disappears after some seconds or after a mouse-click. After that, the main window of the viewer is immediately shown.

If the current settings of your graphics adapter do not support the display of true-color (either 24 or 32 bit) images, an information box as depicted in Figure 2 recommends you to choose other graphic adapter settings.

Figure 1: Splash screen of the viewer program

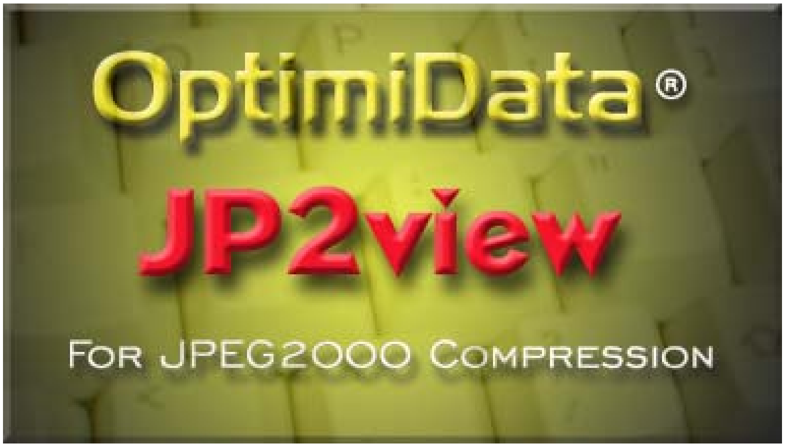
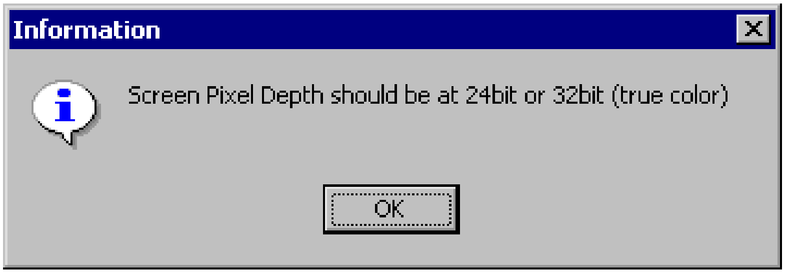
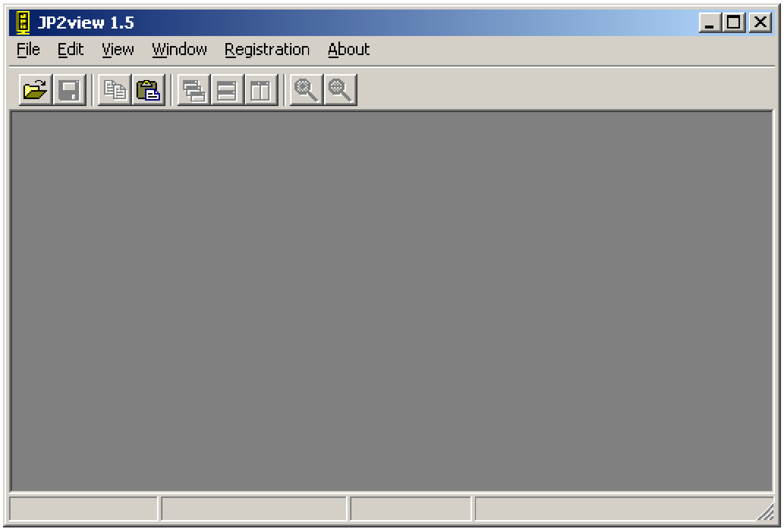


Figure 2: Information box recommending true-color display mode



Screen Pixel Depth should be at 24bit or 32bit (true color)

Figure 3: Main Window of the viewer program



The Main Window offers a menu bar as well as a toolbar to control the compression and decompression parameters as well as functions for the registration and image window arrangement.

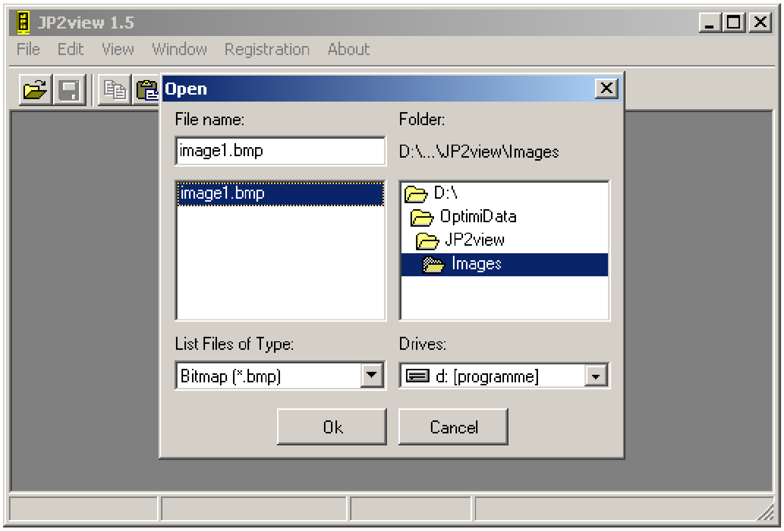
4.4 Open Images

The viewer allows to open images of the following formats:

- Windows Bitmap (*.bmp)
- Old JPEG (*.jpg)
- JPEG2000 File (*.jp2)
- JPEG2000 codestream (*.j2k)

This can be performed by clicking the symbol of the toolbar or by using the ***File - open*** menu option. The “File Open” dialog box appears as shown in Figure 4.

Figure 4: File Open dialog box



File name: image1.bmp

Folder: D:\...\JP2view\Images

image1.bmp

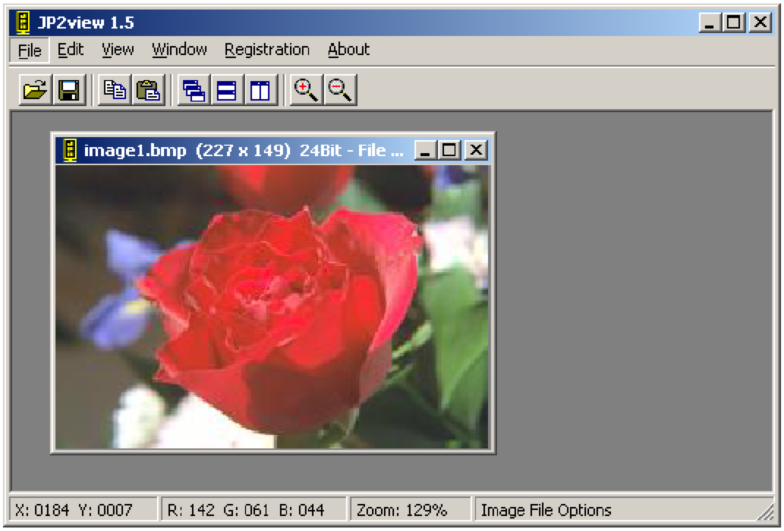
D:\
OptimiData
JP2view
Images

List Files of Type: Bitmap (*.bmp)

Drives: d: [programme]

You can navigate through the directory structure and select the image to open into the viewer. After selection click **OK** to open the image. The image file is loaded and shown as in Figure 5.

Figure 5: Image viewer with opened image



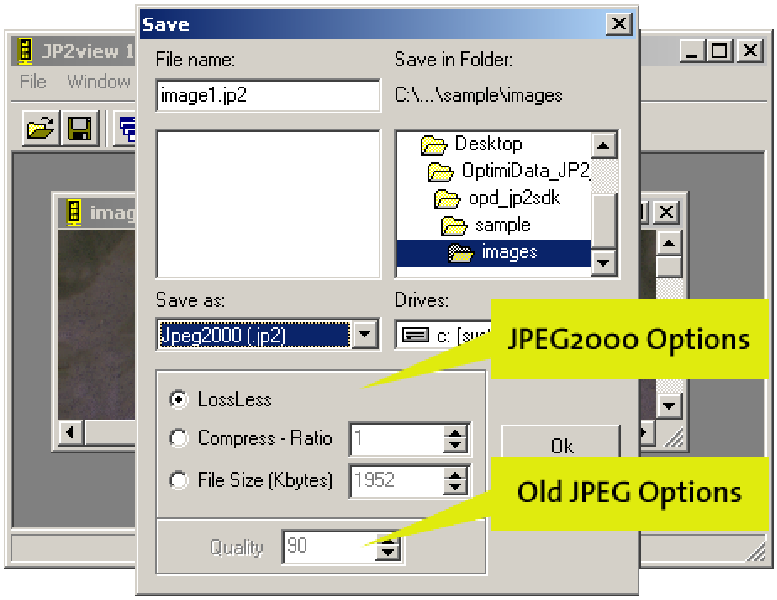
4.5 Save and compress images

The viewer can save the images using the following image formats:

- Windows Bitmap (*.bmp)
- Old JPEG (*.jpg)
- JPEG2000 File (*.jp2)
- JPEG2000 codestream (*.j2k)

This can performed by clicking the 💾 button of the toolbar or by using the ***File - save*** menu option. The **“Save File”** dialog box is shown in Figure 6. Depending on the choosen image format the **“Save File”** dialog box enables control elements to enter compression parameters. Saving images as windows bitmaps does not allow any compression. All additional control elements are disabled while saving as windows bitmap.

Figure 6: Save File Dialog box



Save

File name: image1.jp2

Save in Folder: C:\...\sample\images

Desktop
OptimiData_JP2
opd_jp2sdk
sample
images

Save as: Jpeg2000 (.jp2)

Drives: c: [sys

[✓ LossLess]
Compress - Ratio 1
File Size (Kbytes) 1952

Quality 90

Ok

Save as JPEG2000 (*.jp2 or *.j2k):

The upper part of the compression options are used to control the parameters. The compression can either be performed lossless by means of no image content is lost during compression or lossy at a specific compression ratio given as a file size or a compression ratio. The radio button control is used to choose this compression option. While the image file size of a lossless compressed image cannot be determined, no destination file size or compression ratio can be entered using the lossless option. The resulting file size is image dependent and will most likely result in a file which is between 30% (3:1 compression) and 20% (5:1 compression) of the original file size.

By entering a determined compression ratio, the JPEG2000 compression will be performed in lossy mode and will result in a destination file with a file size as given with the compression ratio. A compression ratio of 10 will reduce the file to 10% of the original file size. Typical compression ratios for lossy JPEG2000 compression are between 30 and 50.

By entering a destination file size, the JPEG2000 compression will be performed in lossy mode and will result in a file with a file size as given in the save dialog box.

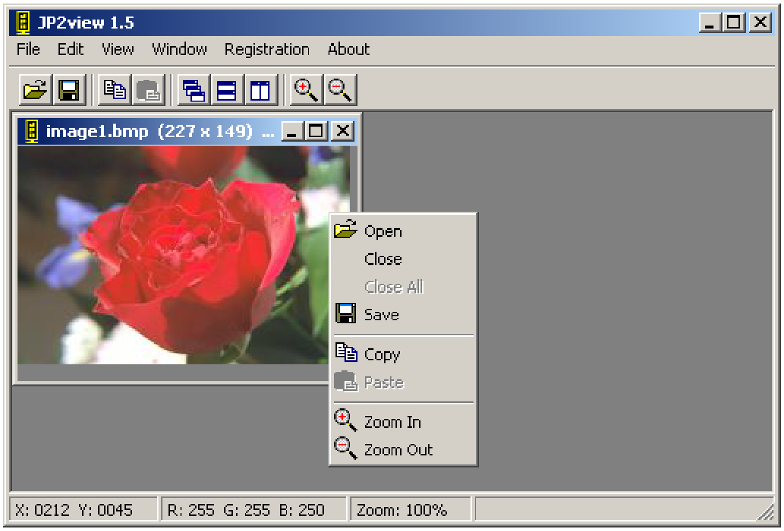
Save as Old JPEG (*.jpg):

The lower part of the compression options is used to control the parameters of Old JPEG compression. An abstract quality parameter ranging between 1 and 100 can be used to control the quality of the compressed image. While a quality of 100 will result in a moderate compressed image at a good image quality, a quality of 1 will result in a highly compressed image with clearly visible loss of image quality.

4.6 Context Menu

Context menu may be opened with right mouse button click. Menu functions are depicted in Figure 7.

Figure 7: Context Menu



5 Contact information

OptimiData

Corporation for optimized data handling

Postfach 33 21 34
14180 Berlin
Germany

Fon: +49 (0)30/827 064 66
Fax: +49 (0)30/827 064 67

Email: info@optimidata.com
URL: http://www.optimidata.com