# The BCIF lossless image compression program

User manual - version 1.0 beta

Stefano Brocchi

December 8, 2010

## Contents



## The BCIF algorithm

The BCIF algorithm is a image compression method that allows to encode an image without any loss of quality (lossless). It is designed to be a fast algorithm, in order to allow practical usage; its compression ratio results better than the one of other competing standards such as Jpeg2000, Jpeg-LS and PNG. Actually, the BCIF algorithm can only handle true color images with a 24 bit color depth (8 bit per primary color); it can read or write BMP files, but its GUI can integrate with Imagemagick's convert tool to handle also other formats. The implementing program has been released as open source under the GPL license. The website of BCIF, containing the results of the benchmarks and last version of the program, can be found at

www.researchandtechnology.net/bcif/

The current release of both the BCIF program and of this documentation is a **beta version**, so the program may contain some small issues or bugs. Feel free to contact the author to signal them; an email can be found on the BCIF website.

### Readable formats

The BCIF program basically handles true color images in the BMP format. The input files to be compressed must be of 8 bits per color (24 bits per pixel), be uncompressed and contain no alpha channel. To convert images in this format I suggests the convert free utility from Imagemagick, the recommended syntax is (example to convert the PNG image "lena.png" to "lena.bmp"):

```
convert lena.png -alpha off -type truecolor BMP3:lena.bmp
```

This command requires at least imagemagick version 6.3.6-9; previous version do the job without the `-alpha off` option, so in this case the user should assure that the input file has no alpha channel, otherwise the BCIF program could be unable to read the resulting BMP file. The BCIF GUI attempts to call this command automatically if the input file is not in the BMP format.

**Warning:** when compressing to a BCIF file only the data actually representing the image is stored, while other auxiliary information contained in the original file is not saved (for example, some file formats can contain a transparency channel or metadata about time and place where the picture was taken, model of the used camera, etc...).

**Issue about black and white images:** since black and white images are represented with 8 bit per pixel bitmaps, the BCIF program is actually not able to read them. There is a simple workaround: convert these images to true color with the above convert command or with any image manipulation program. The drawback is that when the image is decompressed, the resulting BMP file will still be in true color, resulting to be three times bigger than the original BMP.

## The Java implementation

The BCIF algorithm has been implemented in Java to allow its cross platform execution, to realize easily a Graphical User Interface and to allow the possibility to insert BCIF files in a website thanks to an applet that allows client side decompression and visualization. The drawback is that the program is much slower and hence may be unsuited for the encoding and decoding of large images. Another issue about large images is that, unless differently specified, the default maximum memory allocated for the program may be insufficient, requiring to set a greater maximum memory as explained in the next section.

The Java distribution comes in two executable JAR files, relative to the BCIF viewer and command line utility, and to the BCIF gui, named `bcif.jar` and `bcifGUI.jar`[1]. To launch the BCIF GUI, on most systems it is sufficient to click on the JAR file to load the Java virtual machine and execute the file. To launch it

[1]Technical note: the only difference between the two jar files is the manifest, so anyone of the two can be used in place of the other. For example, the GUI can be launched from the bcif.jar file with the command `java -classpath bcif.jar bcif/bcifGUI`

from a command console, or to launch the BCIF viewer and command line utility, the required syntax is

```
java -jar bcifGUI.jar
java -jar bcif.jar [options]
```

The BCIF program requires Java 6, and the source files can be compiled with Java 5 or superior. The latest Java release can be obtained on the Java website.

### Encoding and decoding high resolution images: memory issues

By default, the Java Virtual Machine allows a quite limited use of memory. Since the actual implementation has yet no memory optimization for encoding and visualization, this could be a problem if attempting to compress large images (say, greater than 3200x2400). To allow a greater memory use during execution, Java can be launched with the -Xmx*size* flag, as in the following examples:

```
java -Xmx1024M -jar bcifGUI.jar
java -Xmx1024M -jar bcif.jar -c large_image.bmp large_image.bcif
```

The compression procedure requires, as a rule of thumb, about twice the space required for the uncompressed image plus 7-8 MB for the GUI (all this space is not intrinsically necessary for the algorithm, and in future versions these limits may be greatly reduced). As a quick solution, imposing large limits (as the 1 GB in the example) will solve the problem. The decompression process instead, as long as the streaming option is specified, is almost costless and should not give any problems.

### The Java GUI

The Java GUI is a comfortable graphical interface to compress and decompress images with the BCIF algorithm. It is possible to select input and output files with a graphical file chooser, and to visualize directly BCIF images. If the imagemagick's convert utility is detected (i.e. the convert program can be called in the current working directory) then it is used to convert files also from other lossless formats, as PNG. In this case, the original file is converted to a temporary BMP file that is then compressed to a BCIF file.

The areas of the GUI contain the following components:

- Input file: the input file that must be compressed of decompressed; if the convert utility has not been detected, it must be a BMP or BCIF file, otherwise any format supported by convert is ok. The view button visualizes this file.
- Output file: the destination of the compression or decompression.
- Action buttons: the 'guess' option determines the action to do depending on the input/output file extensions. Otherwise, chose encode to compress images or decode to decompress them.

- Stream buttons: the default option 'stream' decompresses files using a minimal amount of memory, and is the recommended option. The 'do not stream' option requires a significative amount of memory, but in some cases it can be a bit faster.

- Encode button: executes the selected action on the two input and output files.

- Messages: a message about the execution status is displayed in the interface; below it there is a text box containing the output of the core module of the algorithm.

### The Java viewer and command line utility

The BCIF viewer and command line utility recognizes the following syntax:

```
java -jar bcif.jar inputfile [options] [outputfile]
```

The **inputfile** represents the BMP input file to compress of the BCIF file to decompress or view. The **outputfile** is resulting compressed or decompressed file to create; if not specified, an output filename is chosen by the program by changing the extension of the inputfile to BMP or BCIF, depending on the type of output. If this file exists, then BCIF overwrites it without prompting, so use it with care.

The options can be:

- *-v or -view* Visualizes the input file. Default option if the inputfile has extension BCIF.

- *-c or -compress* Compress the input file (BMP → BCIF). Default option if the inputfile has extension BMP.

- *-d or -decompress* Decompress the input file (BCIF → BMP).

- *-s or -stream* Stream output during decompression; requires a minimal amount of memory during the process. Default option.

- *-ns or -npstream* Do not stream output during decompression; requires much more memory, but the decoding process may result faster.

- *-hc or -hashcode* Prints an hash code for the image. Incompatible with -stream and -view

- *-h or -help* Prints help information.

Sometimes, an older JVM may cause Java to ignore the input parameters. In this case, it will be necessary to update to the newest version.

### Using the Java applet to embed BCIF images in a website

The Java implementation of the BCIF algorithm allows the embedding of BCIF images in websites. In this way, losslessly highly compressed images can be inserted in a website even with no client side compatibility, apart from Java installed. When the user loads the page containing the BCIF image, the Java applet will decompress it client side and show it in the web page. When the first image is found, the browser mat require some time to load Java, for the following images the visualization should be quite fast. You may need some basic notions about HTML and the use of Java applets to insert the images in a site.

Both the `bcifGUI.jar` and `bcif.jar` files contain the applet, implemented in the class `bcif.bcifapplet.class`. The required parameters are

- *Image:* the image name on the server, eventually relative to the path of the current page. Note that for the browser's security policy, it is impossible to load images externally to the website where the HTML page is hosted.
- *Width, height:* the width and the height of the image.

The following example visualizes the image lena.bcif (size 512x512 pixels) assuming that both the image and the bcifGUI.jar files are in the same folder of the webpage.

```
<applet codebase = "./"
        archive  = "bcifGUI.jar"
        code     = "bcif.bcifapplet.class"
        width    = 512
        height   = 512>

<param name="image" value="lena.bcif">
<param name="width" value="512">
<param name="height" value="512">

</applet>
```

Of all of the modules of BCIF, the applet is probably the least tested and the less responsive in case of error. Another issue: it is very unlikely that an user will be able to save the images on his local drive, since the link to the image is not in a standard `img` tag but it is specified as an applet parameter.

## The native executable command line utility

The native executable command line utility has been implemented in C++ in order to obtain the maximum speed allowed by the BCIF algorithm. Actually, only a native executable for windows is furnished, but soon the source code will be released allowing the compilation and execution of BCIF on any platform. The syntax is the following:

```
bcif inputfile [-c|-d] [outputfile] [-h]
```

The -c and -d options specify the action to execute (compress or decompress respectively). The input file must be a BMP file to be compressed or a BCIF file to be decompressed; if the output file is not specified, a default filename is created by changing the extension of the inputfile to an appropriate one for the output format. As in the Java version, output files are overwritten without prompting, so use with care. Finally, a -h option is available to print some help.

## About

The BCIF algorithm has been created and implemented by Stefano Brocchi as an evolution of the older PCIF algorithm. The first translation of the program in C++, comprehensive of a first optimization, has been done by Gabriele Nencini. Further details and a contact of the author can be found online at the BCIF website.