Big Image Group

Radomír Vávra, Jiří Filip,
Michal Havlíček, Libor Bakajsa

Institute of Information Theory and Automation of the CAS
Czech Republic

Version 3.0

Updated June 11, 2018

1 Overview

BIG is an open-source file format for storage of a group of high-dynamic-range floating-point high-quality images. This document defines the BIG format and explains concepts that are specific to this format.

Contents

2 Introduction

The BIG file format is indented to store many relevant images in one data file in order to speed up processing of the images, prevent loss of individual images, etc.

3 Definition

The data are stored in little-endian format. There is no compression of the data other than usage of up-to several types of arithmetic representations in order to allow fast random access to the data.

The file starts with the magic number 42 49 47 00 00 00 00 00 hexa.

Next, there are so-called chunks in arbitrary order.

The stored data have to be in the form of *nD* array, where *n* is at most 5 in basic version. The format is not intended to store images with different dimensions or to store individual numbers of tiles for the images. Only exception is possibility to use different data formats using index maps.

3.1 Chunk

Figure 1: Chunk.

Each chunk consists of three parts. Its length is always divisible by eight. Therefore starts of the individual chunks in a memory are 8-bytes aligned. Length of the first two parts of a chunk is 8 bytes each. Length of the third part depends on the type of the chunk.

The first part of a chunk is its identifier according to Table 1. This part is 8 bytes long as it is unsigned 64-bit number.

The second part of a chunk defines length of the third part in bytes and it is 8 bytes long. It might seems unnecessary to store length of the third part for chunks that store only single value (e.g., image width), however this concept enables to jump over redefined or unknown chunks when the BIG format was somehow extended in future releases.

The third part of a chunk represents the stored data or meta data. Length of this part depends on the type of the chunk, however it is always divisible by eight. Therefore, the data might be padded with zeros to fulfill this requirement.

Table 1: List of the data chunks.

ID	Name	Data Type	Default Value
1	number of images	unsigned 64-bit	0
2	number of tiles	unsigned 64-bit	1
3	image height	unsigned 64-bit	0
4	image width	unsigned 64-bit	0
5	number of image planes	unsigned 64-bit	0
6	data order	unsigned 64-bit [ ]	1 2 3 4 5
7	data type	unsigned 8-bit [ ]	2
8	data	any [ ]	
9	color space	char [ ]	0
10	illuminant	char [ ]	0
11	ppmm	double	0
12	name	char [ ]	
13	text	char [ ]	
14	XML	char [ ]	
15	coordinates of tiles	double [ ]	
16	list of directions	unsigned 64-bit, double [ ]	

3.2 Types of Chunks

3.2.1 Number of Images

The first chunk stores the number of images that are stored in the BIG data file.

3.2.2 Number of Tiles

Usually, a tile is meant to be a (possibly enhanced) sub-area of an image which has a periodic character. It means, that if we place several copies of a tile next to each other in both directions, we get bigger image without noticeable disturbances. From each image can be extracted arbitrary number of tiles. Only limitation is that all the tiles have to have the same dimensions. If this chunk is not present (or its value is set to default value explicitly), each image is assumed to be one (possibly not periodic) tile.

3.2.3 Image Height

This chunk represents height of the stored images (tiles).

3.2.4 Image Width

This chunk represents width of the stored images (tiles).

Table 2: Data types which can be stored in the data chunk.

ID	Name	No. of bits	Description
1	half	16-bit	floating point number according to IEEE 754
2	float	32-bit	floating point number according to IEEE 754
3	double	64-bit	floating point number according to IEEE 754
4	signed char	8-bit	signed integer
5	unsigned char	8-bit	unsigned integer
6	short	16-bit	signed integer
7	unsigned short	16-bit	unsigned integer
8	int	32-bit	signed integer
9	unsigned int	32-bit	unsigned integer
10	long long	64-bit	signed integer
11	unsigned long long	64-bit	unsigned integer
12	bool	1-bit	boolean value

3.2.5 Number of Image Planes

Through this chunk, the BIG file format enables to store gray-scale images, RGB color images, multi-spectral images, etc.

3.2.6 Data Order

Specifies order in which data are stored in the data chunk. Each possible order can be beneficial for some application. Therefore it is enabled to specify any data order by providing numbers of chunks according to how the data are stored. For example, in the default order *1 2 3 4 5*, data are sored as follows. First, the outer most loop is over individual images. Then, we iterate over tiles. Next loop goes through individual rows and the one after that through individual pixels. The inner most loop iterates over image planes, finally accessing stored values.

3.2.7 Data Type(s)

This chunk defines data type(s) used in the data chunk according to Table 2. If all data in the data chunk are of the same data type, we store only one number in this chunk. On the other hand, data can be of the different data types. Then, zero is stored in the first 8-bits of this chunk followed by data types of the outermost entities (e.g. images) of the format. Which entities are the outermost is given by the data order chunk. In default, the images are the outermost entities. Therefore, each image can have its own data type in default. If, e.g., image height is placed to the first position in the data order chunk, data type of each row can be specified by the data type(s) chunk.

Table 3: List of Directions.

ID	Description
1	ideal device angles
2	real device angles
3	ideal spherical angles
4	real spherical angles
5	ideal half-difference angles
6	real half-difference angles

3.2.8 Data

3.2.9 Color Space

This chunk (if applicable) together with illuminant defines color space of the data.

3.2.10 Illuminant

3.2.11 Pixels per millimeter (ppmm)

3.2.12 Name

3.2.13 Text

3.2.14 XML

3.2.15 Coordinates of Tiles

3.2.16 List of Directions

This chunk serves to store additional data for each image, particularly directions toward illumination and an observer in various parameterizations. This chunk is allowed to be defined several times in one file. First, there is stored unsigned 64-bit integer according to Table 3 indicating which list of directions is present in this chunk. Then, there are four double values for each image stored in the format.

4 Usage

5 Extensions

6 Credits

BIG file format was developed at Institute of Information Theory and Automation of the Czech Academy of Sciences, Czech Republic. This research has been supported by the Czech Science Foundation grant GA17-18407S.