Binary Point Cloud Version 1.0 Documentation

Version: 1.03 draft
Datum: 04.02.2009
Authors: Fredie Kern (i3mainz) kern@geoinform.fh-mainz.de
Michael Pospiš (Lupos3D) pospis@lupos3d.de

1 Introduction

This document describes the data format called **B**inary **P**oint **C**loud (BPC). The BPC is a manufacturer independently data exchange format for point clouds, who captured by terrestrial laser scanner.

A point cloud is defined as an amount of measurement points. Each measurement point consist as a three dimensional cartesian coordinate (x, y, z) which refers to a sensor integrated coordinate frame. The origin of this sensor integrated coordinate frame is the point of intersection of the rotation axes of the TLS scanner (Fig. 1). The sensor integrated coordinate frame is right-handed (for example: x-axis = easting, y-axis = northing, z-axis = up-direction). Each measurement point is captured by polar coordinates, two directions *H* and *V* in space plus a refectorless slope distance *D*, relative to the origin. The point cloud can be structured rather sorted in respect of this two directions (arrangement in a grid manner).

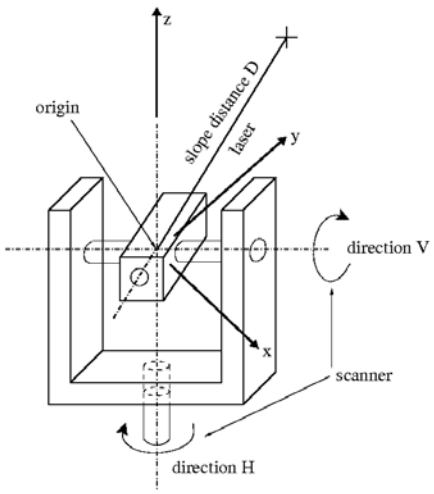

Fig. 1: Definition of the sensor integrated coordinate frame.

Additional measurement elements every measurment point could be:

1. one intensity value as a rate of the received backscatter of the laser light energy at TLS-sensor. The intensity value is the integer representation of the pulse return magnitude. This value is system specific.
2. one color value (typically RGB); captured by an additional camera sensor. The color value can be used for inking the point cloud by a photo-realistic visualization.

With the BPC-format only one particular point cloud including meta-information per file can stored. The data, which are stored, are typically original measurements. If any corrections or reductions are applied to the measurement, and in which way, are not documented by the BPC-format.

The conception of the BPC-format follows the especial conditions of the terrestrial laser-scanning (limited measuring volume, limited resolution). It covers the main application area without following each one special case. It is a compromise between efficiently use of memory and efficient data processing and should be easy to implement. Using the XML-technology the BPC-format is easy to expand and will satisfy future requirements

2 Structure

The BPC-format is composed of two parts. A BPC-file starts with the part including metainformation (ASCII). The second part includes the intrinsically measurement data using a binary coding.

2.1 part metainformation

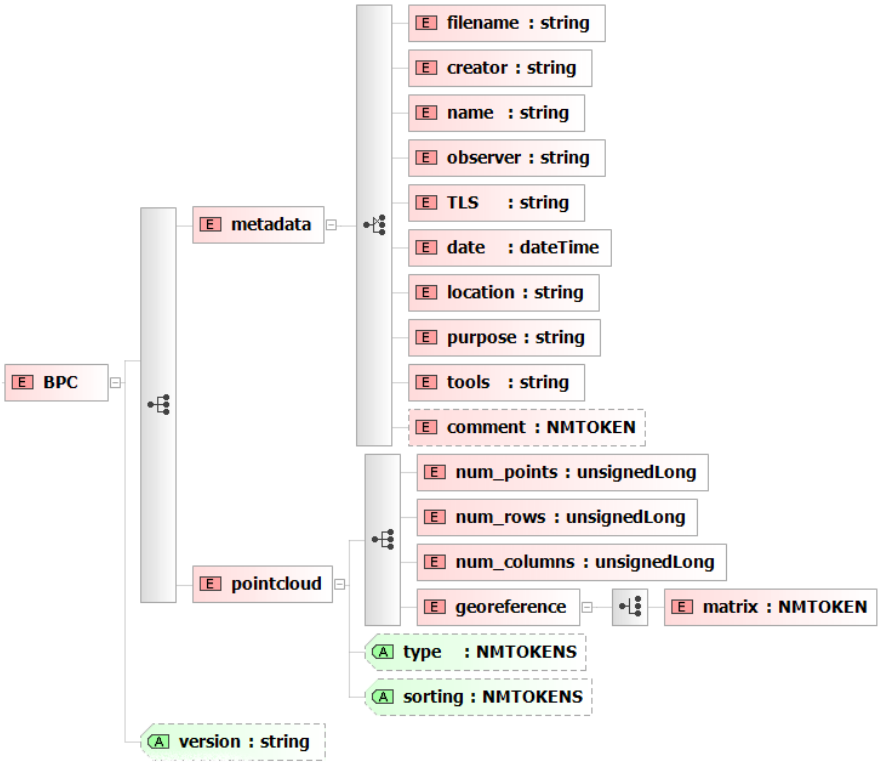

Fig. 2: Structure of the BPC-XML-Document (`bpc.xsd`).

The part of metainformation is stored and structured as a valid XML-document (Fig. 2). The XML-document is defined by a Document-Type-Definition (`bpc.dtd`) and XML-Schema (`bpc.xsd`).

The XML-document is grouped by the primary element name <BPC>. With the attribute `version` the version number of the BPC-file is specified. The monetary value is `"1.0"`.

There are two sections inside the XML-document. The first section is separated with the element name `<metadata>` and second with the element name `<pointcloud>`, includes technical information about data amount and arrangement in data part of file.

The important, but not strictly required, metadata declared with `<metadata>` are:

`<filename>`	Own file name.
`<creator>`	Information about the program, which created this file.
`<name>`	What is it? Possible information storing here are: ▪ point of observation ▪ point number ▪ point attribute
`<TLS>`	Whereby the data was captured or processed? Possible information storing here are: ▪ kind of sensor ▪ serial number ▪ sensor name ▪ firmware ▪ version
`<observer>`	Who did it? Possible information storing here are: ▪ name of observer ▪ name of operator ▪ name of software
`<date>`	When the data was captured or processed? Date and time of observation.
`<location>`	Where the data was captured or processed? Possible information storing here are: ▪ coordinates ▪ object name ▪ adress
`<purpose>`	Why the data was captured or processed? Possible information storing here are: ▪ function ▪ aim ▪ intention
`<tool>`	Which tools or algorithms are used? Possible information storing here are: ▪ support ▪ program
`<comment>`	Other unstructured and unsorted meta data.

For correct analyzing the second section the following attributes and elements of `<pointcloud>` are important and necessary (strictly required).

Attributes of pointcloud:

Kind of data type: `type="xyzIrgb" | "xyz" | "xyzI" | "xyzIRGB"`
Kind of data arrangement: `sorting="graticule" |` *`anything else`*

About the effects of these attributes see below.

Elements of pointcloud:

Number of points in pointcloud:	`<num_points>`
Number of rows and collumns in grid:	`<num_rows>, <num_columns>`
Parameter of referencing the local coordinates of pointcloud in a reference frame:	`<georeference>`

The 4x4-georeferencing-matrix T is organized in follow manner:

$$\underset{4\times 4}{\mathbf{T}} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_X \\ r_{2,1} & r_{2,2} & r_{2,3} & t_Y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_Z \\ 0 & 0 & 0 & s \end{bmatrix}$$

with:

$r_{i,j}$: elements of rotation matrix (including shearing and mirroring)
t_i : elements of translation vector
s : scale factor

To transform the pointcloud data from local sensor system into global coordinate frame use the following formula:

homogenous coordinates: $$\mathbf{X}^h_{global} = \begin{bmatrix} x_{global} \\ y_{global} \\ z_{global} \\ s \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_{lokal} \\ y_{lokal} \\ z_{lokal} \\ 1 \end{bmatrix}$$

cartesian coordinates: $$\mathbf{X}_{global} = \begin{bmatrix} X_{global} \\ Y_{global} \\ Z_{global} \end{bmatrix} = \begin{bmatrix} x_{global}/s \\ x_{global}/s \\ x_{global}/s \end{bmatrix} = \frac{1}{s}\begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_{local} \\ y_{local} \\ z_{local} \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix}$$

The length of the metainformation part is 2048 byte and the end of metainformation part is marked by `0x1A` (Ctrl-Z) at file position 2047.

2.2 Part pointcloud data

The arrangement of the data inside of the binary part pointcloud data is appointed by the attribute `sorting` of the element <`pointcloud`>. The choice of arrangement is structured (“`graticule`”) or not structured (anything else than “`graticule`”). The default sorting mode is gridded. Also if the attribute `sorting` is missing.

If the arrangement is structured the data starts at first row and than inside each row column by column (like the arrangement of an image).

Each point in the data part is defined by one of the following structure (C-style-syntax) depends on the attribute `type` of element <`pointcloud`>:

```
// if data type is “xyzIrgb”
struct xyzIrgb {
  float          x[3];   // cartesian 3D coordinates   4 byte  [-1000.0000,1000.0000]  12 byte
  unsigned short I;      // intensity                 16 bit   [0,65536]                2 byte
  unsigned char  rgb[3]; // 24bit-RGB-color            1 byte  [0,255]                  3 byte
};

// if data is type “xyz”
struct xyz {
  float          x[3];   // cartesian 3D coordinates   4 byte  [-1000.0000,1000.0000]  12 byte
};

// if data type is “xyzI”
struct xyzI {
  float          x[3];    // cartesian 3D coordinates  4 byte  [-1000.0000,1000.0000]  12 byte
  unsigned short I;       // intensity                 16 bit   [0,65536]                2 byte
}

// if data type is “xyzIRGB”
struct xyzIRGB {
  float           x[3]; // cartesian 3D coordinates    4 byte  [-1000.0000,1000.0000]  12 byte
  unsigned short  I;    // intensity                  16 bit   [0,65536]                2 byte
  unsigned short RGB[3];// 48bit-RGB-color             2 byte  [0,65535]                6 byte
};

// float : IEEE-Format 4Byte = 4*8 =32Bit
// short : 2-byte-integer
// char  : 1-byte-integer
```

The default structure is “`struct xyzIrgb`”. This is used, if the attribute `type` in <`pointcloud`>-element is set to“`xyzIrgb`” or if no type is specified.

In the future different kinds of point-data will be supported.

3 Limitations

- Only one point cloud per file is supported.
- If the point data are arranged in a grid manner with rows and columns, missing points have to be saved as point witch zero information (`x[]={0.0,0.0,0.0}, rgb[]={0,0,0}, I=0`). No gaps in the matrix are allowed.

4 Example

Look a the file 'mini.bpc'.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="bpc.css"?>
<!-- created by ptx_cls.exe Version 0.001 20.09.2008 (Sep 22 2008 00:29:37)
                 (c) 2008  i3mainz - Prof. Dr.-Ing. Fredie Kern -->
<!DOCTYPE TLS_IO SYSTEM "bpc.dtd">
<BPC version="1.0">
  <metadata>
    <filename>mini.ptx</filename>
    <creator>PBC-Processor v1.0 (c) Fast-Software Ltd.</creator>
    <name>Test</name>
    <TLS>(unknown) Serial-No: 900267-007</TLS>
    <observer>Kaluschke, Alfred</observer>
    <date>2008-09-22T12:13:47</date>
    <location>Mainz, Holzstraße</location>
    <purpose>testing TLS-IO</purpose>
    <tool>export from i3mainzscene, no changes</tool>
    <comment>ptx_bpc.exe</comment>
  </metadata>
  <pointcloud type="xyzIRGB" sorting="graticule">
    <num_points>3</num_points>
    <num_rows>1</num_rows>
    <num_columns>3</num_columns>
    <georeference>
      <matrix>
        1.000000 0.000000 0.000000 0.000000
        0.000000 1.000000 0.000000 0.000000
        0.000000 0.000000 1.000000 0.000000
        0.000000 0.000000 0.000000 1.000000
      </matrix>
    </georeference>
  </pointcloud>
</BPC>
…
← pointcloud data will following here at file position 2048
```

5 Future prospects

(empty)

6 Implementation

(empty)