AL3D FILE FORMAT

FILE FORMAT SPECIFICATION

Version 1.0

Author: **Thomas Ascher**

Alicona Imaging GmbH • info@alicona.com • www.alicona.com
Parkring 2 • A-8074 Grambach
t +43 (316) 4000 700 • Fax +43 (316) 4000 711

CONTENT

INTRODUCTION

This document is a technical specification for the Alicona 3D file format (*.AL3D files).

Features:

- 32 bit depth data
- 8 bit (monochrome/RGB) or 16 bit texture data
- including calibration data
- supporting texture types
    - stereoscopic/trinocular images
    - texture image for the depth data
    - 4 photometric/backscatter electron microscope images
    - variable size image stack/series (batch of images)

Applications:

- Storing of 3D surface data with textures and calibration data
- Storing of stereoscopic images with calibration data
- Storing of light microscopy image stacks with calibration data
- Storing of photometric/backscatter electron microscope images

1 FORMAT OVERVIEW

The Alicona 3D format is designed to store 3D surface data (depth image), multiple surface related texture images (texture, left/right stereo image, …), a dataset icon and corresponding calibration data. All images within an AL3D file, except for the icon, share the same width, height and proportional or unproportional pixelsize.

An AL3D file is made of the following parts, in the given order:

Header	File format identification, variable file format + calibration data, user defined comment.
Data	Images of the dataset: icon, depth image, texture images

Header:

Type String	17 Byte	A special string to identify the file as an AL3D file. Identical in future versions.
Version Tag	52 Byte – Tag Size	Tag (see used datatypes) that specifies the file format version. Identical in future versions.
Counter Tag	52 Byte – Tag Size	Tag (see used datatypes) that specifies how many tags are stored in the header.
Format and Calibration Tags	52 Byte – Tag Size	Header part with file format information (size, offsets, ...) and meta information (pixelsize, …)
Comment	52 Byte – Tag Size	A user defined comment, to give a basic information about the data stored in the AL3D file.

Data:

Icon data (optional)	150x150 pixel RGB icon for the dataset
Depth image data (optional)	3D surface data field – a float point image/array (see used datatypes) which contains metric height information
Texture data plane 0 (optional) – part of texture data block	The texture data is stored in a series of equal size 8 bit planes/channels within a texture data block. A monochrome 8 bit image refers to 1 plane wheras an 8 bit RGB image refers to 3 planes. Such a reference is set by tags (decribed later in the document). 16 bit images use 2 tags for plane referencing.
Texture data plane 1 (optional)	All planes are part of one combined texture data block with a variable number of planes.
….	
Texture data plane n-1 (optional)	The last plane.

2 USED DATATYPES

This section gives an overview of the used datatypes used in the AL3D format.

2.1 TAGS

A tag is a key-value pair to store all kinds of meta data, like file format information and calibration data, with a given identifier (you can find a list of defined identifiers later in the document). Tags are only used in the header part of an AL3D file.

A tag can contain values in this formats:

- Zero terminated 8 bit ASCII C string
- Integer values as zero terminated C strings in the form: 12345
- Floating point values stored as zero terminated C strings in the form: -12345.5 or +1e-05. The "." is used as seperator.

A tag is a 42 byte long binary field split up in a 20 byte key C string that consists of a word (of max. 19 characters) like "InvalidPixelValue", a 30 byte value (string of max. 29 characters, integer, ...) and 2 byte line termination (only used because of the representation in text editors).

All non used parts of a tag (space after zero termination, non used value fields) should be initialized with the binary value 0. A key field is not allowed to be empty!

Tag (52 byte) =	**Tag key (20 byte)**	**Tag value (30 byte)**	**Windows line termination (2 byte)**
Relative data offset	0 byte	20 byte	50 byte
Example content	Word TEST with 0 termination	Word TEST with 0 termination	Hex values: D, A = CR, LF
Hex representation	54 45 53 54 00	54 45 53 54 00	0D 0A

2.2 TEXTURE IMAGE DATA

Textures images, within the texture data field, are stored as a series of unsigned 8 bit planes (channels). A monochrome image consists of one plane. An RGB image consists of 3 planes. 16 bit images are split over two 8 bit planes.

A plane consists of a series of scanlines, each aligned with 8 byte. The first byte in the first plane refers to the upper left corner of an image: x = 0, y = 0.

Planes are assigned to images by tags in the header field. For example, the TexturePtr = 0;1;2 says that the image texture is an RGB image consisting of the planes 0 as red color, 1 as blue color and 2 as green color channel. For more information, see the format and calibration tags section.

Scanline byte size = Cols with an alignment of 8 bytes// Cols = Pixel width of image, specified by Cols tag

Image byte size = Scanline byte size * Rows // Rows = Pixel height of image, specified by Rows tag

2.3 DEPTH IMAGE

The depth image in an AL3D file is an image consisting of 4 byte C floating point values (per pixel) as used on 32 bit mashines, stored in little endian byte order. For each pixel, the depth image contains height information. Pixels which don't contain valid height information have to be set to floating point value defined by the "InvalidPixelValue" tag.

The format of the depth image is similar to the texture image data. It also uses 8 byte alignment per scanline but has never more than 1 plane. Note that there is only one depth image per AL3D file. For this reason, there is only one tag called "DepthImageOffset" which specifies the byte offset of the depth image in bytes, from the begin of the AL3D file.

Scanline byte size = Cols * 4 byte with an alignment of 8 bytes

Depth image byte size = Scanline byte size * Rows

3 DETAILED OVERVIEW

Field	Description
Type String – 17 byte	A zero terminated c string consisting of the word "AliconaImaging" (without quotes in given case) with Windows line termination in the last 2 bytes (as used in tags).
Version Tag – 52 byte	Field that specifies the file format version. Stored as a tag with the key "Version" (without quotes) and an integer value as version number. Version 1 is reffered as the integer value 1.
Counter Tag – 52 byte	Field that specifies how many tags are stored in the header part. Stored as a tag with the key "TagCount" (without quotes)
Tags – 52 byte * Counter Tag value	Series of tags which describe the file internal and and calibration data. See the header field contents section for more information. The number of meta data values
Comment field – 256 byte	A user comment stored as zero terminated c string with Windows line termination in the last 2 bytes.
Icon data – 68400 byte (optional)	150 x 150 RGB icon stored as 3 plane image, as described in the texture image datatype section. Planes are stored in order. Only the icon offset is taged as "IconOffset".
Depth image field (optional, see calculation)	Floating point image with depth information, 8 byte alignment. See the depth image datatype section for more information.
Texture image data field (optional, see calculation)	Series of monochrome or RGB images, as described in the texture image datatype section.

4 FORMAT AND CALIBRATION TAG IN THE HEADER

The header is a list of tags with values (as decribed in the used datatypes section) in no given order. The number of the fields is specified before the actual tag section in the "Counter Tag" field. The permanent tags described below are required for a valid AL3D file. Please note that the tag keys have to be stored in the given case!

You are free to store your own tags if you want but you should use some sort of prefix to prevent name clash in the future. Unknown tags within the header can be ignored or preserved, but it is safer to ignore them.

4.1 PERMANENT TAGS

Key	Value/Contents/Description
DepthImageOffset	Offset of the depth image data in bytes, from the begin of the AL3D file, stored as a meta string. If this value is set to 0 or the tag is not present, the file has no depth image.
TextureImageOffset	Offset of the texture image data in bytes, from the begin of the AL3D file, stored as a meta string. If this value is set to 0 or the tag is not present, the file has no additional image data.
IconOffset	Offset of the RGB icon image data in bytes, from the begin of the AL3D file, stored as a meta string. If this value is set to 0 or the tag is not present, the file has no icon.
PixelSizeXMeter	Horizontal pixelsize of the contained images in meter. This value is stored as a meta floating point value.
PixelSizeYMeter	Verticale pixelsize of the contained images in meter. This value is stored as a meta floating point value.
Cols	Width of the contained images in pixel. All texture images and the depth image share the same width! This value is an integer value stored as meta string.
Rows	Height of the contained images in pixel. All texture images and the depth image share the same width! This value is an integer value stored as meta string.
NumberOfPlanes	Number of planes/byte images in the image data part. If this value is set to 0, the file has no images.
InvalidPixelValue	Invalid pixel value: Defines which pixels within the depth image are not valid height information. A pixel that is equal to this value is not valid. We set the value to the maximum C floating point value (4byte float on 32 bit mashines).
ImageCode	This value is reserved for internal use. Set this value to 0 or ignore this tag.
TexturePtr	Points to the texture of the depth image within the texture image block = describes a plane reference. The values in this field refer to a number of planes (plane number list, seperated with ";") within the texture data field.

<table>
<tr><td></td><td>This way the tag can either refer to a monochrome or RGB image.

Possible values for assignment (as meta string):

• Monochrome image: 7

• RGB image: 0;1;2

• No texture available: empty value field or no TexturePtr tag

An image/image channel (RGB) is assigned with a plane number. You can calculate the file offset in an easy way:

File offset = TextureImageOffset + (Image byte size * plane number)</td></tr>
<tr><td>TextureLoPtr

(LeftStereoLoPtr, RightStereoLoPtr, PhotometricLoPtrX, ImageStackLoPtrX)</td><td>This tag has the same function as the TexturePtr tag but it is used as an addition if you want to store 16 bit images.

To store a 16 bit image, you have to split up the 16 bit values into 8 bit fractions so that they will fit within the texture image field. Each fraction is stored in another plane. First, set the TexturePtr to planes which point to the higher 8 bit fraction. Then, set the TextureLoPtr to the lower 8 bit fraction of the 16 bit values.

Example binary value 1111111100000000:

<table><tr><td>TexturePtr = 7</td><td>TextureLoPtr = 12</td></tr><tr><td>11111111</td><td>000000</td></tr></table></td></tr>
<tr><td>LeftStereoPtr

RightStereoPtr</td><td>Points to a left/right stereo image, for the dataset within the texture image block. Read the TexturePtr description for more information. If a trinocular dataset is present the middle image should be stored in the texture image</td></tr>
<tr><td>PhotometricPtrX</td><td>X = value from 0 – 3. Points to an photometric image/bascatter electron microscope image.

0: image with upper left lightsource

1: image with upper right lightsource

2: image with lower left lightsource

3: image with lower right lightsource

Read the TexturePtr description for more information.</td></tr>
<tr><td>ImageStackPtrX</td><td>X = value from 0 – n. Points to a sepcific image within an image stack. The image stack tag with the lowest X refers to the image that was captured on the smallest Z position within the stack..</td></tr>
<tr><td>CreatingApplication</td><td>Name of the creating application. Here can you mark the name of the application which writes the AL3D file.</td></tr>
</table>

4.2 ADDITIONAL TAGS

WorkingDistance	The workingdistance (in meter) used to capture the dataset.
TiltAngle	Tilt angle (in degree) between the left and right stereo image.
StereoImageOffsetX	Horizontal offset between the left and right stereo image in pixel.
StereoImageOffsetY	Verticale offset between the left and right stereo image in pixel.
Magnification	The magnification factor used when the data in the AL3D file was captured.
Voltage	Voltag (in Volts) used when the data in the AL3D file was captured from a SEM.
RecordingDevice	Name of the device that was used to acquire the AL3D file content.

5 COPYRIGHTS