VOLA Documentation

Release 1

Jonathan Byrne

Aug 28, 2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

VOLA is a compact data structure that unifies computer vision and 3D rendering and allows for the rapid calculation of connected components, per-voxel census/accounting, CNN inference, path planning and obstacle avoidance. Using a hierarchical bit array format allows it to run efficiently on embedded systems and maximize the level of data compression. The proposed format allows massive scale volumetric data to be used in embedded applications where it would be inconceivable to utilize point-clouds due to memory constraints. Furthermore, geographical and qualitative data is embedded in the file structure to allow it to be used in place of standard point cloud formats.

A paper detailing the format and its applications can be downloaded here

This API is developed to allow for comparison and analysis with existing formats. An overview of the api functions and their interactions are shown in the image below:

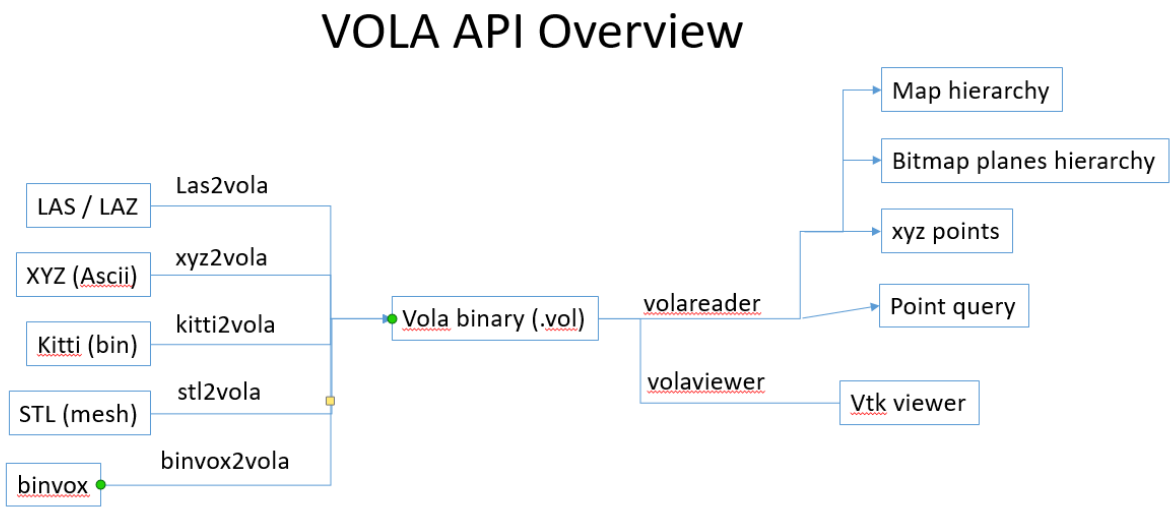

Fig. 1.1: layout of the VOLA api

The parsers (xxx2vola) will convert to the VOLA format, embedding information where information is available. The reader and the viewer will allow you to examine the data in VOLA format.

There is sample data for each of the formats in the samplefiles folder

An example workflow for a LIDAR file containing color information is as follows:

./las2vola samplefiles/cchurchdecimated 3 -n

this will parse the las file to a vola depth of 3 and the -n flag means the color information will be automatically added to VOLA file

./volareader samplefiles/cchurchdecimated.vol -c

print the coordinates of the voxels

./volaviewer samplefiles/cchurchdecimated.vol

view the vola file.

CHAPTER

TWO

PARSERS

2.1 las2vola module

```
usage: las2vola.py [-h] [--crs CRS] [-n] [-d] input depth

positional arguments:
  input        the name of the file / files / directory you want to open. You
               can used wildcards(*.las / *.laz) or the directory for multiple
               files
  depth        how many levels the vola tree will use

optional arguments:
  -h, --help   show this help message and exit
  --crs CRS    the coordinate system of the input, e.g., 29902 (irish grid
               epsg code)
  -n, --nbits  use 1+nbits per voxel. The parser works out what info to embed
  -d, --dense  output a dense point cloud
```

Las2vola: Converts Las files into VOLA format.

The ISPRS las format is the standard for LIDAR devices and stores information on the points obtained. This parser uses the las information for the nbit per voxel representation. The data stored is: color, height, number of returns, intensity and classification

@author: Jonathan Byrne

las2vola.**main**()
Read the file, build the tree. Write a Binary.

las2vola.**parse_las**(*filename*, *nbits*)
Read las format point data and return header and points.

2.2 xyz2vola module

```
usage: xyz2vola.py [-h] [--crs CRS] [-n] [-d] input depth

positional arguments:
  input        the name of the file / files / directory you want to open. You
               can used wildcards(*.asc / *.xyz) or the directory for multiple
               files
  depth        how many levels the vola tree will use

optional arguments:
```

```
  -h, --help   show this help message and exit
  --crs CRS    the coordinate system of the input, e.g., 29902 (irish grid
               epsg code)
  -n, --nbits  use 1+nbits per voxel. The parser works out what info to embed
  -d, --dense  output a dense point cloud
```

xyz2vola: Converts ascii point clouds into VOLA format.

This will automatically parse files with a structure x,y, z or x, y, z, r, g, b, intensity @author Jonathan Byrne

xyz2vola.**main**()
Read the file, build the tree. Write a Binary.

xyz2vola.**parse_xyz**(*filename*, *nbits*)
Read xyz format point data and return header, points and points data.

2.3 stl2vola module

```
usage: stl2vola.py [-h] [--crs CRS] [-n] [-d] input depth

positional arguments:
  input        the name of the file / files / directory you want to open. You
               can used wildcards(*.stl) or the directory for multiple files
  depth        how many levels the vola tree will use

optional arguments:
  -h, --help   show this help message and exit
  --crs CRS    the coordinate system of the input, e.g., 29902 (irish grid
               epsg code)
  -n, --nbits  use 1+nbits per voxel. The parser works out what info to embed
  -d, --dense  output a dense point cloud
```

Converts stl triangle meshes into VOLA format.

STL is an industry standard mesh format. There is no information other than triangles so the occupancy information is only available for this format.

TODO: Need to cleverly remove duplicate points and add subdivide function.

stl2vola.**main**()
Read the file, build the tree. Write a Binary.

stl2vola.**parse_stl**(*filename*)
Read las format point data and return header and points.

2.4 binvox2vola module

```
usage: binvox2vola.py [-h] [--crs CRS] [-n] [-d] input depth

positional arguments:
  input        the name of the file / files / directory you want to open. You
               can used wildcards(*.binvox) or the directory for multiple
               files
  depth        how many levels the vola tree will use
```

```
optional arguments:
  -h, --help   show this help message and exit
  --crs CRS    the coordinate system of the input, e.g., 29902 (irish grid
               epsg code)
  -n, --nbits  use 1+nbits per voxel. The parser works out what info to embed
  -d, --dense  output a dense point cloud
```

xyz2vola: Converts binvox files into VOLA format.

Binvox is a very popular volumetric representation that uses run length encoding to achieve significant compression. It is included as there are many datasets that are stored in binvox format. There is no information other than voxels so the occupancy information is only available for this format.

TODO: switch xyz and xzy encoding @author Jonathan Byrne

binvox2vola.**main**()
Read the file, build the tree. Write a Binary.

binvox2vola.**parse_binvox**(*filename*)
Read xyz format point data and return header, points and points data.

binvox2vola.**runlength_to_xyz**(*bytevals*, *header*)
Binvox uses a binary runlength encoding (valuebyte, countbyte).

2.5 kitti2vola module

```
usage: kitti2vola.py [-h] [--crs CRS] [-n] [-d] input depth

positional arguments:
  input        the name of the file / files / directory you want to open. You
               can used wildcards(*.bin) or the directory for multiple files
  depth        how many levels the vola tree will use

optional arguments:
  -h, --help   show this help message and exit
  --crs CRS    the coordinate system of the input, e.g., 29902 (irish grid
               epsg code)
  -n, --nbits  use 1+nbits per voxel. The parser works out what info to embed
  -d, --dense  output a dense point cloud
```

Converts bin files from the kitti dataset into VOLA format.

Kitti is a LIDAR dataset for automotive testing. The dataset stores an intensity value which is converted to a greyscale color for nbits VOLA.

@author: Ananya Gupta and Jonathan Byrne

kitti2vola.**main**()
Read the file, build the tree. Write a Binary.

kitti2vola.**parse_bin**(*filename*, *nbits*)
Read in float values and reshape to 2d numpy array.

CHAPTER

THREE

READERS

3.1 volareader module

```
usage: volareader.py [-h] [-c] [-g] [-b] [-i] [-m MAP] vol

positional arguments:
  vol                the name of the vola file to open

optional arguments:
  -h, --help         show this help message and exit
  -c, --coords       output the coordinates of the voxels in the vola file
  -g, --get          check if a voxel exists at a given location and return
                     the value if it does.
  -b, --bincoords    output the binary coordinates of the voxels in the vola
                     file
  -i, --images       output image planes for each depth
  -m MAP, --map MAP  output flattened map for a given height above the ground
```

VOLA Reader.

Processes sparse vola files (.vol). Reads in the header information and has a set of functions for extracting the voxel locations and data.

3.2 volaviewer module

```
usage: volaviewer.py [-h] fname

positional arguments:
  fname       the name of the file you want to open

optional arguments:
  -h, --help  show this help message and exit
```

VOLA viewer.

VTK and python 3 based viewer for showing the voxel data for individual tiles. Uses the VOLA reader to process the individual sparse vola tiles (.vol) @author Jonathan Byrne

volaviewer.**main**()
Draw the voxels for a given filename.

volaviewer.**view_voxels**(*positions*, *colors*)
VTK based viewer for sparse VOLA files (.vol).

Maps VOLA and draws opengl cubes for voxels and their color information.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX