jakteristics

David Caron

Apr 20, 2023

CONTENTS

1 Installation **3**
- 1.1 Pypi . . . 3
- 1.2 Building . . . 3

2 Usage **5**
- 2.1 From python . . . 5
- 2.2 CLI . . . 6

3 Reference **9**
- 3.1 jakteristics . . . 9
- 3.2 jakteristics.las_utils . . . 10

Index **11**

Jakteristics is a python package to compute point cloud geometric features.

A **geometric feature** is a description of the geometric shape around a point based on its neighborhood. For example, a point located on a wall will have a high *planarity*.

The features used in this package are described in the paper Contour detection in unstructured 3D point clouds. They are based on the eigenvalues *1*, *2* and *3* and the eigenvectors *e1*, *e2* and *e3*.

- Eigenvalue sum : $1 + 2 + 3$
- Omnivariance: $(1 \cdot 2 \cdot 3)^{1/3}$
- Eigenentropy: $-_{i=1}^{3} i \cdot \ln(i)$
- Anisotropy: $(13)/1$
- Planarity: $(23)/1$
- Linearity: $(12)/1$
- PCA1: $1/(1 + 2 + 3)$
- PCA2: $2/(1 + 2 + 3)$
- Surface Variation: $3/(1 + 2 + 3)$
- Sphericity: $3/1$
- Verticality: $1 - |e3[2]|$
- Nx, Ny, Nz: The normal vector

CHAPTER

ONE

INSTALLATION

1.1 Pypi

To install from pypi, run:

```
pip install jakteristics
```

1.2 Building

To build the package from source, you need to have a compiler to build the cython extensions.

On Windows, you need Microsoft C++ Build Tools

Then, run:

```
python -m pip install .
```

CHAPTER

TWO

USAGE

2.1 From python

If your input is a las file, you can use the *jakteristics.las_utils.read_las_xyz()* function to read the xyz coordinates from your input file as a numpy array.

If your input is already a numpy array, you can use the *jakteristics.compute_features()* directly. By default, every geometry feature will be computed. In a future version, the **:py:arg:`feature_names`** argument will be required.

Once you have the features, you can use them like any numpy array, or write them to a las file using `jakteristics.write_with_extra_dims()`.

Example:

```
from jakteristics import las_utils, compute_features, FEATURE_NAMES

input_path = "/path/to/a/las_file.las"
xyz = las_utils.read_las_xyz(input_path)

features = compute_features(xyz, search_radius=0.15, feature_names=FEATURE_NAMES)

output_path = "/path/to/output_file.las"
las_utils.write_with_extra_dims(input_path, output_path, features, FEATURE_NAMES)

# or for a specific feature:
omnivariance = compute_features(xyz, search_radius=0.15, feature_names=["omnivariance"])
output_omnivariance = "/path/to/output_omnivariance.las"
las_utils.write_with_extra_dims(input_path, output_omnivariance, omnivariance, [
↪"omnivariance"])
```

If you want to to all of these steps (read a las file, compute certain features and write it back to disk), you can use the command line:

```
jakteristics input/las/file.las output/file.las --search-radius 0.15 --num-threads 4
```

2.2 CLI

2.2.1 jakteristics

```
jakteristics [OPTIONS] LAS_INPUT OUTPUT
```

Options

-s, --search-radius <search_radius>

Required The search radius to use to query neighbors.

-t, --num-threads <num_threads>

The number of threads to use for computation. Defaults to the number of cpu on the machine.

Default
`-1`

-f, --feature <feature_names>

The feature names to compute. Repeat this parameter to compute multiple features. Use –show-features to see the list of possible choices. Default: All features.

Default
`eigenvalue_sum, omnivariance, eigenentropy, anisotropy, planarity, linearity, PCA1, PCA2, surface_variation, sphericity, verticality, nx, ny, nz, number_of_neighbors, eigenvalue1, eigenvalue2, eigenvalue3, eigenvector1x, eigenvector1y, eigenvector1z, eigenvector2x, eigenvector2y, eigenvector2z, eigenvector3x, eigenvector3y, eigenvector3z`

-m, --manhattan-distance

How to compute the distance between 2 points. If provided, the sum-of-absolute-values is used ('Manhattan' distance).By default, the standard Euclidean distance is used.

Default
`False`

-e, --eps <eps>

Return approximate nearest neighbors; the k-th returned value is guaranteed to be no further than (1+eps) times the distance to the real k-th nearest neighbor.

Default
`0`

--show-features

Show a list of possible feature names and exit.

--install-completion <install_completion>

Install completion for the specified shell.

Options
bash | zsh | fish | powershell | pwsh

--show-completion <show_completion>

Show completion for the specified shell, to copy it or customize the installation.

Options
bash | zsh | fish | powershell | pwsh

Arguments

LAS_INPUT

Required argument

OUTPUT

Required argument

CHAPTER

THREE

REFERENCE

3.1 jakteristics

`jakteristics.compute_features(points, search_radius, *, kdtree=None, num_threads=-1, max_k_neighbors=50000, euclidean_distance=True, feature_names=None, eps=0.0)`

Compute features for a set of points.

Parameters

- **points** (ndarray) – A contiguous (n, 3) array of xyz coordinates to query.
- **search_radius** (float) – The radius to query neighbors at each point.
- **kdtree** (Optional[cKDTree]) – If None, the kdtree is computed from the list of points. Must be an instance of *jakteristics.cKDTree* (and not *scipy.spatial.cKDTree*).
- **num_threads** (int) – The number of threads (OpenMP) to use when doing the computation. Default: The number of cores on the machine.
- **max_k_neighbors** (int) – The maximum number of neighbors to query Larger number will use more memory, but the neighbor points are not all kept at the same time in memory. Note: if this number is smaller, the neighbor search will not be faster. The radius is used to do the query, and the neighbors are then removed according to this parameter.
- **euclidean_distance** (bool) – How to compute the distance between 2 points. If true, the Euclidean distance is used. If false, the sum-of-absolute-values is used ("Manhattan" distance).
- **feature_names** (Optional[List[str]]) – The feature names to compute (see *constants.FEATURE_NAMES* for possible values) Default: all features
- **eps** (float) – Return approximate nearest neighbors; the k-th returned value is guaranteed to be no further than (1+eps) times the distance to the real k-th nearest neighbor.

Return type

ndarray

Returns

The computed features, one row per query point, and one column per requested feature.

3.2 jakteristics.las_utils

`jakteristics.las_utils.read_las_xyz(filename, with_offset=False)`

Reads xyz coordinates of a las file, optionally as single precision floating point.

Parameters

- **filename** (Union[str, Path]) – The las file to read
- **with_offset** (bool) – If True, returns a tuple of a float32 array of coordinates, and the las header offset If False, returns only float64 coordinates Default: False

Return type

Union[array, Tuple[ndarray, List[float]]]

Returns

Depending on the *with_offset* parameter, either an (n x 3) array, or a tuple of an (n x 3) array and the file offset.

`jakteristics.las_utils.write_with_extra_dims(input_path, output_path, extra_dims, extra_dims_names)`

From an existing las file, create a new las file with extra dimensions

Parameters

- **input_path** (Path) – The input las file.
- **output_path** (Path) – The output las file.
- **extra_dims** (array) – The numpy array containing geometric features.
- **extra_dims_names** (List) – A list of names corresponding to each column of *extra_dims*.

INDEX

Symbols

--eps
    jakteristics command line option, 6
--feature
    jakteristics command line option, 6
--install-completion
    jakteristics command line option, 6
--manhattan-distance
    jakteristics command line option, 6
--num-threads
    jakteristics command line option, 6
--search-radius
    jakteristics command line option, 6
--show-completion
    jakteristics command line option, 6
--show-features
    jakteristics command line option, 6
-e
    jakteristics command line option, 6
-f
    jakteristics command line option, 6
-m
    jakteristics command line option, 6
-s
    jakteristics command line option, 6
-t
    jakteristics command line option, 6

C

compute_features() (*in module jakteristics*), 9

J

jakteristics command line option
    --eps, 6
    --feature, 6
    --install-completion, 6
    --manhattan-distance, 6
    --num-threads, 6
    --search-radius, 6
    --show-completion, 6
    --show-features, 6
    -e, 6
    -f, 6
    -m, 6
    -s, 6
    -t, 6
    LAS_INPUT, 7
    OUTPUT, 7

L

LAS_INPUT
    jakteristics command line option, 7

O

OUTPUT
    jakteristics command line option, 7

R

read_las_xyz() (*in module jakteristics.las_utils*), 10

W

write_with_extra_dims() (*in module jakteristics.las_utils*), 10