PyKV

Release 0.0.1

Anderson Tavares

Oct 31, 2019

CONTENTS:

CHAPTER

ONE

PYKV PACKAGE

1.1 Submodules

1.2 pykv.geo module

1.2.1 Geo

Functions for general geometric problems

Created on Wed Oct 30 22:00:43 2019

@author: andmo55

pykv.geo.**angleaxis_from_w**(*ws*)

pykv.geo.**exp_skew**(*so*, *out=None*)

Exponential of a list of skew-symmetric matrices so(3), i.e., the result is a list of rotations SO(3). It applies Rodrigues' formula.

$$e^{[\boldsymbol{\omega}]_\times} = \mathrm{I}_3 + \sin\theta[\boldsymbol{\omega}]_\times + (1-\cos\theta)[\boldsymbol{\omega}]_\times^2, \qquad \theta = \|\boldsymbol{\omega}\|$$

so: ndarray(...,3,3) The ndarray of skew-symmetric matrices

out: ndarray(...,3,3) The ndarray of rotation matrices (if `None` is supplied, a new array is created)

SO: ndarray(...,3,3) The ndarray of rotation matrices

```
>>> import pcdlib as pc
>>> import numpy as np
>>> aa = [1,2,3]
>>> so = pc.skew_from_angleaxis(aa)
>>> pc.exp_skew()
```

pykv.geo.**exp_skew_from_w**(*ws*, *out=None*, *so=None*, *theta=None*, *sintheta=None*, *costheta=None*, *dtype=<class 'numpy.float32'>*)

Exponential of skew-symmetric matrices directly from angle-axis vectors. Some intermediate steps (so, theta, sintheta, costheta) can be provided for efficiency (they are generated and returned if not provided).

ws: ndarray or list list of angle-axis vectors

out: ndarray list of 3x3 special orthogonal (SO) matrices (if `None`, then a new array is created)

so: ndarray

theta: ndarray

sintheta: ndarray costheta: ndarray

ndarray list of 3x3 special orthogonal (SO) matrices

ndarray list of skew-symmetric matrices

ndarray list of angles

ndarray list of sines of angles

ndarray list of cosines of angles

pykv.geo.**frustum**(*l*, *r*, *b*, *t*, *n*, *f*, *out=None*, *dtype=<class 'numpy.float32'>*)

Creates a perspective matrix from parameters of its frustrum Node: numpy uses row-major order (OpenGL uses column-major), so you see the transpose of common OpenGL frustrum matrices in the literature

dtype l: float

left

r: float right

b: float bottom

t: float top

n: float near

f: float far

out: ndarray (4x4) (default: None) perspective matrix (preallocated or fresh new)

ndarray(4x4) perspective matrix

pykv.geo.**log_SO**(*SO*, *out=None*)

Compute the logarithm of special orthogonal matrices (SO(3)), resulting in skew-symmetric matrices (so(3)) or angle-axis w

$$\log(R \in SO(3)) = \log e^{[\omega]_\times} = [\omega]_\times \in so(3)$$

SO: ndarray(...,3,3) The rotation matrices in SO(3)

angleaxis: bool (default: False) `True` to return just the angle-axis vectors instead of skew symmetric matrices.

out: ndarray(...,3,3) or ndarray(...,3) (default: None) Array of skew-symmetric matrices or angle-axis vectors (if `None`, a new array is generated and returned)

out: ndarray(...,3,3) Array of skew-symmetric matrices

what: ndarray(...,3) Array of directions

theta: ndarray(...,3) Array of angles

```
>>>
>>>
>>>
```

pykv.geo.**ortho**(*l*, *r*, *b*, *t*, *n*, *f*, *out=None*, *dtype=<class 'numpy.float32'>*)

Ortographic projection matrix

dtype l r b t n f out

pykv.geo.**perspective**(*fov=45*, *aspect=1.8*, *near=0.1*, *far=100*, *out=None*, *dtype=<class 'numpy.float32'>*)

Symmetric perspective projection matrix. It uses the same

fov: float define the vertical field of view (in degrees)

aspect: float relation between horizontal and vertical sizes

near: float positive (near > 0) number

far: float positive (far > near > 0) number

out: ndarray preallocated or new array

dtype type of new array (if `out` is None)

ndarray preallocated (if out != None) or fresh new array

pykv.geo.**rigid**(*pcd*, *R*, *out=None*)

pykv.geo.**rotate**(*M*, *n*, *theta*, *out=None*)

pykv.geo.**scale**(*M*, *s*, *out=None*)

pykv.geo.**skew_from_angleaxis**(*n=None*, *theta=None*, *ws=None*, *out=None*)

Convert angle-axis vectors to skew-symmetric matrices

pykv.geo.**skew_from_w**(*ws*, *out=None*)

Convert angle-axis vectors to skew-symmetric matrices. Opposite: `angleaxis_from_skew`.

$$\boldsymbol{\omega} \mapsto [\boldsymbol{\omega}]_\times, \qquad \boldsymbol{\omega} = \theta\hat{\boldsymbol{\omega}}$$

ws: ndarray(...,3) ND list/array of angle-axis vectors

out: ndarray(...,) Pre-allocated array for storing the skew-symmetric matrices (if `None`, a new fresh array is allocated and returned)

so: ndarray(...,3, 3) (N+1)D array of skew-symmetric matrices

```
>>> import pcdlib as pc
>>> import numpy as np
>>> pc.skew_from_angleaxis(np.random.rand(2,3))
array([[[ 0.        , -0.18267483,  0.66990937],
        [ 0.18267483,  0.        , -0.154273  ],
        [-0.66990937,  0.154273  ,  0.        ]],
       [[ 0.        , -0.63770203,  0.70667849],
        [ 0.63770203,  0.        , -0.36215796],
        [-0.70667849,  0.36215796,  0.        ]]])
```

pykv.geo.**translate**(*t*, *M=None*, *out=None*, *dtype=<class 'numpy.float32'>*)

Translate a NxN matrix M by a (N-1)-D vector. If M is `None`, then a new translation matrix is created

t: list or ndarray the translation vector of size N-1.

M: the source matrix of size NxN (identity if not provided)

out the destination matrix (`None` for new matrix, not `None` for existing one, or even M). Default: `None`.

out the destination matrix (`None` for new matrix, not `None` for existing one, or even M). Default: `None`.

pykv.geo.**w_from_skew**(*so*, *out=None*)
Extract angle-axis vectors from skew-symmetric matrices. Opposite: skew_from_angleaxis.

$$f : so = [\boldsymbol{\omega}]_\times \mapsto \boldsymbol{\omega}, \qquad \theta = \|\boldsymbol{\omega}\| \quad \hat{\boldsymbol{\omega}} = \frac{\boldsymbol{\omega}}{\theta}$$

so: ndarray(...,3,3) ND list/array of skew-symmetric matrices

out: ndarray(...,3) ND list of angle-axis vectors (if None is supplied, a new array is created).

out: ndarray(...,3) ND list of angle-axis vectors.

```
>>> import pcdlib as pc
>>> aa = [1,2,3]
>>> so = pc.skew_from_angleaxis(aa)
>>> aa2 = pc.angleaxis_from_skew(so)
>>> aa2
array([1., 2., 3.])
```

1.3 pykv.kitti module

1.4 pykv.ouster module

pykv.ouster.**apply_px_offset**(*images*, *px_offset*, *im=None*)
Apply offset Parameters ———- images px_offset

pykv.ouster.**get_px_offset**(*W: int*)
Get the offsets for each row of range images generated by Ouster lidar devices

W: width of the image

ndarray(64)

pykv.ouster.**make_xyz_lut**(*W*, *H*, *azimuth_angles*, *altitude_angles*)
Create the rays for the ouster lidar data, assuming constant device rotation speed.

W: width of the image (number of angles) H: height of the image (number of lasers) azimuth_angles (initial horizontal offset for each laser) altitude_angles (vertical angle for each laser)

ndarray(H, W, 3) the set of HW rays which represents

1.5 pykv.ouster_rosbag module

Created on Wed Oct 30 13:07:02 2019

@author: andmo55

pykv.ouster_rosbag.**bag2ndarray**(*bagfilename*, *H=64*, *W=1024*)
Convert a ROS bag to Numpy structured array in NPZ file format Parameters ———- bagfilename

pykv.ouster_rosbag.**get_nframes**(*bag*, *topics*)

1.6 Module contents

CHAPTER

TWO

INSTALLING

2.1 From source

```
git clone https://gitlab.com/anderflash/pykv.git
cd pykv
sh build.sh
```

If you are using Windows, you can replace *sh build.sh* by *python setup.py install*

To check whether it is installed: .. code:: bash

    python -c “import pykv”

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX