camera.py Documentation

Release 0.1

Matej Smid

Aug 30, 2017

Contents

CHAPTER 1

Features

- camera intrinsic and extrinsic parameters handling
- various lens distortion models
- model persistence
- projection of camera coordinates to an image
- conversion of image coordinates on a plane to camera coordinates
- visibility handling

CHAPTER 2

Installation

```
pip install git+https://github.com/smidm/camera.py.git
```

CHAPTER 3

Usage

Setup camera and project point [0, 0, 0]:

```
import camera
import numpy as np
import math
c = camera.Camera()
c.set_K_elements(1225.0, math.pi / 2, 1, 480, 384)
R = np.array(
    [[-0.9316877145365, -0.3608289515885, 0.002545329627547],
     [-0.1725273110187, 0.4247524018287, -0.8888909933995],
     [0.3296724908378, -0.8263880720441, -0.4579894432589]])
c.set_R(R)
c.set_t(np.array([[-1.365061486465], [3.431608806127], [17.74182159488]]))

print c.world_to_image(np.array([[0., 0., 0.]]).T)
```

Load camera parameters from a file:

```
import camera
import numpy as np
c = camera.Camera()
c.load('camera_01.yaml')
c.world_to_image(np.array([[0., 0., 0.]]).T)
```

CHAPTER 4

Reference

camera

class camera.**Camera**(*id=None*)

Projective camera model

- camera intrinsic and extrinsic parameters handling
- various lens distortion models
- model persistence
- projection of camera coordinates to an image
- conversion of image coordinates on a plane to camera coordinates
- visibility handling

distort(*undistorted_image_coords*, *Kundistortion=None*)

Apply distortion to ideal image coordinates.

Parameters

- **undistorted_image_coords** (*numpy.ndarray, shape=(2, n)*) – ideal image coordinates
- **Kundistortion** (*array-like, shape=(3, 3)*) – camera matrix for undistorted coordinates, None for self.K

Returns distorted image coordinates

Return type numpy.ndarray, shape=(2, n)

get_A(*K=None*)

Return part of K matrix that applies center, skew and aspect ratio to ideal image coordinates.

Return type np.ndarray, shape=(3, 3)

get_K_0()
Return ideal calibration matrix (only focal length present).

Returns ideal calibration matrix

Return type np.ndarray, shape=(3, 3)

get_camera_center()
Returns camera center in the world coordinates.

Returns camera center in projective coordinates

Return type np.ndarray, shape=(4, 1)

get_focal_length()
Get camera focal length.

Returns focal length

Return type double

get_principal_point_px()
Get camera principal point.

Returns x and y pixel coordinates

Return type numpy.ndarray, shape=(1, 2)

get_view_matrix(*alpha*)
Returns camera matrix for handling image and coordinates distortion and undistortion. Based on alpha, up to all pixels of the distorted image can be visible in the undistorted image.

Parameters **alpha** (*float or None*) – Free scaling parameter between 0 (when all the pixels in the undistorted image are valid) and 1 (when all the source image pixels are retained in the undistorted image). For convenience for -1 returns custom camera matrix self.Kundistortion and None returns self.K.

Returns camera matrix for a view defined by alpha

Return type array, shape=(3, 3)

get_z0_homography(*K=None*)
Return homography from world plane at z = 0 to image plane.

Returns 2d plane homography

Return type np.ndarray, shape=(3, 3)

image_to_world(*image_px*, *z*)
Project image points with defined world z to world coordinates.

Parameters

- **image_px** (*numpy.ndarray, shape=(2 or 3, n)*) – image points
- **z** (*float*) – world z coordinate of the projected image points

Returns n projective world coordinates

Return type numpy.ndarray, shape=(3, n)

is_visible(*xy_px*)
Check visibility of image points.

Parameters **xy_px** (*np.ndarray, shape=(2, n)*) – image point(s)

Returns visibility of image points

Return type numpy.ndarray, shape=(1, n), dtype=bool

is_visible_world(*world*)
Check visibility of world points.

Parameters **world** (*numpy.ndarray, shape=(3, n)*) – world points

Returns visibility of world points

Return type numpy.ndarray, shape=(1, n), dtype=bool

load(*filename*)
Load camera model from a YAML file.

Example:

```
calibration_type: standard
K:
- [1225.2, -7.502186291576686e-14, 480.0]
- [0.0, 1225.2, 384.0]
- [0.0, 0.0, 1.0]
R:
- [-0.9316877145365, -0.3608289515885, 0.002545329627547]
- [-0.1725273110187, 0.4247524018287, -0.8888909933995]
- [0.3296724908378, -0.8263880720441, -0.4579894432589]
id: 0
kappa: [0.0, 0.0]
size_px: [960, 768]
t:
- [-1.365061486465]
- [3.431608806127]
- [17.74182159488]
```

plot_world_points(*points*, *plot_style*, *label=None*, *solve_visibility=True*)
Plot world points to a matplotlib figure.

Parameters

- **points** (*numpy.ndarray, shape=(3 or 4, n) or list of lists*) – world points (projective or euclidean)
- **plot_style** (*str*) – matplotlib point and line style code, e.g. 'ro'
- **label** (*str*) – label plotted under points mean
- **solve_visibility** (*bool*) – if true then plot only if all points are visible

save(*filename*)
Save camera model to a YAML file.

set_K(*K*)
Set K and update P.

Parameters **K** (*numpy.ndarray, shape=(3, 3)*) – intrinsic camera parameters

set_K_elements(*u0_px*, *v0_px*, *f=1*, *theta_rad=1.5707963267948966*, *a=1*)
Update pinhole camera intrinsic parameters and updates P matrix.

Parameters

- **u0_px** (*double*) – principal point x position (pixels)
- **v0_px** (*double*) – principal point y position (pixels)
- **f** (*double*) – focal length

- **theta_rad** (*double*) – digitization raster skew (radians)
- **a** (*double*) – pixel aspect ratio

set_R(*R*)
Set camera extrinsic parameters and updates P.

Parameters **R** (*numpy.ndarray, shape=(3, 3)*) – camera extrinsic parameters matrix

set_R_euler_angles(*angles*)
Set rotation matrix according to euler angles and updates P.

Parameters **angles** (*double sequence, len=3*) – 3 euler angles in radians,

set_t(*t*)
Set camera translation and updates P.

Parameters **t** (*numpy.ndarray, shape=(3, 1)*) – camera translation vector

undistort(*distorted_image_coords*, *Kundistortion=None*)
Remove distortion from image coordinates.

Parameters

- **distorted_image_coords** (*numpy.ndarray, shape=(2, n)*) – real image coordinates
- **Kundistortion** (*array-like, shape=(3, 3)*) – camera matrix for undistorted view, None for self.K

Returns linear image coordinates

Return type numpy.ndarray, shape=(2, n)

undistort_image(*img*, *Kundistortion=None*)
Transform grayscale image such that radial distortion is removed.

Parameters

- **img** (*np.ndarray, shape=(n, m) or (n, m, 3)*) – input image
- **Kundistortion** (*array-like, shape=(3, 3)*) – camera matrix for undistorted view, None for self.K

Returns transformed image

Return type np.ndarray, shape=(n, m) or (n, m, 3)

update_P()
Update camera P matrix from K, R and t.

world_to_image(*world*)
Project world coordinates to image coordinates.

Parameters **world** (*numpy.ndarray, shape=(3 or 4, n)*) – world points in 3d projective or euclidean coordinates

Returns projective image coordinates

Return type numpy.ndarray, shape=(3, n)

camera.**calibrate_division_model**(*line_coordinates*, *y0*, *z_n*, *focal_length=1*)
Calibrate division model by making lines straight.

Parameters

- **line_coordinates** (*np.ndarray, shape=(nlines, npoints_per_line, 2)*) – coordinates of points on lines
- **y0** (*array-like, len=2*) – radial distortion center xy coordinates
- **z_n** (*float*) – distance to boundary (pincushion: image width / 2, barrel: image diagonal / 2)
- **focal_length** (*float*) – focal length of the camera (optional)

Returns Camera object with calibrated division model parameter lambda

Return type *Camera*

camera.**column**(*vector*)
Return column vector.

Parameters **vector** – np.ndarray

Returns column vector

Return type np.ndarray, shape=(n, 1)

camera.**e2p**(*euclidean*)
Convert 2d or 3d euclidean to projective coordinates.

Parameters **euclidean** (*numpy.ndarray, shape=(2 or 3, n)*) – projective coordinate(s)

Returns projective coordinate(s)

Return type numpy.ndarray, shape=(3 or 4, n)

camera.**fit_line**(*xy*)
Fit line to points.

Parameters **xy** (*np.ndarray, shape=(2, n)*) – point coordinates

Returns line parameters [m, c]

Rtype mc array like

camera.**line_point_distance**(*xy*, *mc*)
Distance from point(s) to line.

Parameters

- **xy** (*np.ndarray, shape=(2, n)*) – point coordinates
- **mc** (*array like*) – line parameters [m, c]

Returns distance(s)

Return type np.ndarray, shape=(n,)

camera.**nearest_point_on_line**(*xy*, *mc*)
Nearest point(s) to line.

Parameters

- **xy** (*np.ndarray, shape=(2, n)*) – point coordinates
- **mc** (*array like*) – line parameters [m, c]

Returns point(s) on line

Return type np.ndarray, shape=(2, n)

camera.**nview_linear_triangulation**(*cameras*, *correspondences*)
Computes ONE world coordinate from image correspondences in n views.

Parameters

- **cameras** (*sequence of Camera objects*) – pinhole models of cameras corresponding to views
- **correspondences** (*numpy.ndarray, shape=(2, n)*) – image coordinates correspondences in n views

Returns world coordinate

Return type numpy.ndarray, shape=(3, 1)

camera.**nview_linear_triangulations**(*cameras*, *image_points*)
Computes world coordinates from image correspondences in n views.

Parameters

- **cameras** (*sequence of Camera objects*) – pinhole models of cameras corresponding to views
- **image_points** (*sequence of m numpy.ndarray, shape=(2, n)*) – image coordinates of m correspondences in n views

Returns m world coordinates

Return type numpy.ndarray, shape=(3, m)

camera.**p2e**(*projective*)
Convert 2d or 3d projective to euclidean coordinates.

Parameters **projective** (*numpy.ndarray, shape=(3 or 4, n)*) – projective coordinate(s)

Returns euclidean coordinate(s)

Return type numpy.ndarray, shape=(2 or 3, n)

CHAPTER 5

Development

To test the package run:

```
$ ./run_tests.sh
```

There are some of the module missing. I appreciate pull requests that fill the gaps.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index