RADIATE SDK

Marcel Sheeny

Jun 03, 2020

CONTENTS:

CHAPTER

ONE

RADIATE_SDK

1.1 radiate module

class radiate.**Sequence**(*sequence_path*, *config_file='config/config.yaml'*)

Bases: object

This class loads the sequence of RADIATE dataset

Example:
>>> import radiate
>>> root_path = 'path/to/radiate/city_3_7/'
>>> seq = radiate.Sequence(root_path)
>>> output = seq.get_from_timestamp(seq.init_timestamp)
>>> seq.vis_all(output)

cfar(*x*, *num_train*, *num_guard*, *rate_fa*)

Detect peaks with CFAR algorithm.

Parameters

- **x** (*np.array*) – input 1d array
- **num_train** (*int*) – Number of training cells.
- **num_guard** (*int*) – Number of guard cells.
- **rate_fa** (*float*) – False alarm rate.

Returns detected points

Return type np.array

cfar2d(*x*, *num_train*, *num_guard*, *rate_fa*)

Detect peaks with 2D CFAR algorithm in each row.

Parameters

- **x** (*np.array*) – input 2d array
- **num_train** (*int*) – Number of training cells.
- **num_guard** (*int*) – Number of guard cells.
- **rate_fa** (*float*) – False alarm rate.

Returns detected points

Return type np.array

draw_boundingbox_rot(*im*, *bbox*, *angle*, *color*)

get_annotation_from_id(*annotation_id*)
get the annotation from an id

Parameters **annotation_id** (*int*) – frame id

Returns list of annotations for the id given as parameter

Return type list

get_from_timestamp(*t*, *get_sensors=True*, *get_annotations=True*)
method to get sensor and annotation information from some timestamp

Parameters

- **t** (*float*) – This is the timestamp which access the sensors/annotations
- **get_sensors** (*bool, optional*) – whether to retrieve sensor information, defaults to True
- **get_annotations** (*bool, optional*) – whether to retrieve annotation info, defaults to True

Returns returns a single variable as a dictionary with 'sensors' and 'annotations' as key

Return type dict

get_id(*t*, *all_timestamps*, *time_offset=0.0*)
get the closest id given the timestamp

Parameters

- **t** (*float*) – timestamp in seconds
- **all_timestamps** (*np.array*) – a list with all timestamps
- **time_offset** (*float, optional*) – offset in case there is some unsynchronoised sensor, defaults to 0.0

Returns the closest id

Return type int

get_lidar_annotations(*id_radar*, *interp=False*, *t_c=None*, *t_r1=None*, *t_r2=None*)
get the annotations in lidar image coordinate frame

Parameters

- **id_radar** (*int*) – the annotation radar id
- **interp** (*bool*) – whether to use interpolation or not
- **t** (*float*) – timestamp

Returns the annotations in lidar image coordinate frame

Return type dict

get_rectfied(*left_im*, *right_im*)
get the left and right image rectfied

Parameters

- **left_im** (*np.array*) – raw left image
- **right_im** (*np.array*) – raw right image

Returns tuple (left_rect, right_rect, disp_to_depth) WHERE np.array left_rect is the rectfied left image np.array right_rect is the rectfied right image np.array disp_to_depth is a matrix that converts the disparity values to distance in meters

Return type tuple

lidar_to_image (*lidar*)
Convert an lidar point cloud to an 2d bird's eye view image

Parameters **lidar** (*np.array*) – lidar point cloud Nx5 (x,y,z, intensity, ring)

Returns 2d bird's eye image with the lidar information

Return type np.array

load_timestamp (*timestamp_path*)
load all timestamps from a sensor

Parameters **timestamp_path** (*string*) – path to text file with all timestamps

Returns list of all timestamps

Return type dict

overlay_camera_lidar (*camera*, *lidar*)
Method that joins camera and projected lidar in one image for visualisation

Parameters

- **camera** (*np.array*) – camera image
- **lidar** (*np.array*) – lidar image with the same size as camera

Returns overlayed image

Return type np.array

project_bboxes_to_camera (*annotations*, *intrinsict*, *extrinsic*)
method to project the bounding boxes to the camera

Parameters

- **annotations** (*list*) – the annotations for the current frame
- **intrinsict** (*np.array*) – intrisic camera parameters
- **extrinsic** (*np.array*) – extrinsic parameters

Returns dictionary with the list of bbounding boxes with camera coordinate frames

Return type dict

project_lidar (*lidar*, *lidar_extrinsics*, *cam_intrinsic*, *color_mode='same'*)
Method to project the lidar into the camera

Parameters

- **lidar** (*np.array*) – lidar point cloud with shape Nx5 (x,y,z,intensity,ring)
- **lidar_extrinsics** (*np.array*) – 4x4 matrix with lidar extrinsic parameters (Rotation and translations)
- **cam_intrinsic** (*np.array*) – 3x3 matrix with camera intrinsic parameters in the form [[fx 0 cx], [0 fx cy], [0 0 1]]
- **color_mode** (*string*) – what type of information is going to be representend in the lidar image

options: ‘same’ always constant color. ‘pseudo_distance’: uses a color map to create a psedo color which refers to the distance. ‘distance’ creates an image with the actual distance as float

Return type np.array

Returns returns the projected lidar into the respective camera with the same size as the camera

read_lidar(*lidar_path*)
given a lidar raw path returns it lidar point cloud

Parameters **lidar_path** (*string*) – path to lidar raw point

Returns lidar point cloud Nx5 (x,y,z,intensity,ring)

Return type np.array

transform_annotations(*annotations*, *M*)
method to transform the annotations to annother coordinate

Parameters

- **annotations** (*list*) – the list of annotations
- **M** (*np.array*) – transformation matrix

Returns the list of annotations in another coodinate frame

Return type list

transform_point_cloud(*pc*, *M*)
transform a 3d point cloud to another coordinate frame

Parameters

- **pc** (*np.array*) – point cloud in the form Nx% (x,y,z,intensity, ring)
- **M** (*np.array*) – transformation matrix

Returns transformed point cloud

Return type np.array

vis(*sensor*, *objects*, *color=None*, *mode='rot'*)
visualise the sensor and its annotation

Parameters

- **sensor** (*the given sensor*) –
- **objects** (*list of objects*) – np.array

Returns image with the objects overlayed

Return type np.array

vis_3d_bbox_cam(*image*, *bboxes_3d*, *pc_size=0.7*)
diplay pseudo 3d bounding box from camera

Parameters

- **image** (*np.array*) – camera which the bounding box is going to be projected
- **bboxes_3d** (*dict*) – list of bounding box information with pseudo-3d image coordinate frame
- **pc_size** (*float*) – percentage of the size of the bounding box [0.0 1.0]

Returns camera image with the correspondent bounding boxes

Return type np.array

vis_all(*output*, *wait_time=1*)
method to diplay all the sensors/annotations

Parameters

- **output** (*dict*) – gets the output from self.get_from_timestamp(t)
- **wait_time** (*int, optional*) – how to long to wait until display next frame. 0 means it will wait for any key, defaults to 1

vis_bbox_cam(*image*, *bboxes_3d*, *pc_size=0.7*)
diplay pseudo 2d bounding box from camera

Parameters

- **image** (*np.array*) – camera which the bounding box is going to be projected
- **bboxes_3d** (*dict*) – list of bounding box information with pseudo-3d image coordinate frame
- **pc_size** (*float*) – percentage of the size of the bounding box [0.0 1.0]

Returns camera image with the correspondent bounding boxes

Return type np.array

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX