bbox-utils Documentation

Release unknown

Eric Wiener

Mar 01, 2021

CONTENTS

bbox-utils allows you to easily convert between different bounding box formats (YOLO, XYWH, XYXY, etc.).

It's as simple to use as:

```
from bbox_utils import BoundingBox

xy1 = np.array([100, 50])
xy2 = np.array([200, 75])
bbox = BoundingBox.from_xyxy(xy1, xy2)

# Get XYWH
xy, w, h = bbox.to_xywh()

# Get XYXY
xy1, xy2 = bbox.to_xyxy()

# Get YOLO
image_dim = 640, 420
yolo_bbox = bbox.to_yolo(image_dim)
```

You can install bbox-utils with PyPI: `pip install bbox-utils`

CHAPTER

ONE

CONVERSIONS

1.1 2D Bounding Box Conversions:

- List of points [top left, top right, bottom right, bottom left]
- XYWH: top left, width, height
- XYXY: top left, bottom right
- YOLO
- 3D Bounding Box Conversions You can create a 3D bounding box with either:

1.2 3D Bounding Box Conversions:

You can create a 3D bounding box with either:

- A center point, width, height, depth, and rotation
- The back-bottom-left point, width, height, depth, and rotation

You can convert between the two forms and also get a triangular polygon to use for plotting triangular meshes.

The majority of the 3D Bounding Box implementation comes from the bbox PyPI package.

CHAPTER

TWO

CONTENTS

2.1 bbox-utils

Utilities to easily convert between different bounding box formats (YOLO, XYWH, XYXY, etc.).

2.1.1 Description

You can find documentation for the project at here.

2D Bounding Box Conversions

- List of points [top left, top right, bottom right, bottom left]
- XYWH: top left, width, height
- XYXY: top left, bottom right
- YOLO

3D Bounding Box Conversions

You can create a 3D bounding box with either:

- A center point, width, height, depth, and rotation
- The back-bottom-left point, width, height, depth, and rotation

You can convert between the two forms and also get a triangular polygon to use for plotting triangular meshes.

The majority of the 3D Bounding Box implementation comes from the bbox PyPI package.

Visualizations You can use *bbox-utils* to visualize annotations within point clouds or images.

To use point clouds, you will need to install open3d and plotly with either:

```
pip3 install open3d plotly==4.14.3
pip install
# or
conda install -c open3d-admin open3d
conda install -c plotly plotly=4.14.3
```

At the time of writing this, *open3d* requires Python < 3.9

To use images, you will need to install OpenCV:

```
pip3 install opencv-python
# or
conda install opencv -c conda-forge
```

2.1.2 Making Changes & Contributing

This project uses pre-commit, please make sure to install it before making any changes:

```
pip install pre-commit
cd bbox-utils
pre-commit install
```

It is a good idea to update the hooks to the latest version:

```
pre-commit autoupdate
```

2.1.3 Note

This project has been set up using PyScaffold 4.0rc1. For details and usage information on PyScaffold see https://pyscaffold.org/.

2.2 License

2.3 Contributors

- Eric Wiener <ericwiener3@gmail.com>

2.4 Changelog

2.4.1 Version 0.1

- Feature A added
- FIX: nasty bug #1729 fixed
- add your changes here!

2.5 bbox_utils

2.5.1 bbox_utils package

Submodules

bbox_utils.bbox_2d module

class bbox_utils.bbox_2d.**BoundingBox**(*points*, *ordered=False*)
Bases: object

property center
Returns the center point of the bounding box as (x, y)

Returns center of bounding box in (x, y) form. NOT (row, column) form.

Return type np.array

static from_xywh(*top_left*, *width*, *height*)
Create a bounding box object from the top-left point, width, and height.

Parameters

- **top_left** (*np.array*) – array of form [x, y]
- **width** (*float*) – width of the bounding box
- **height** (*float*) – height of the bounding box

Returns a new bounding box instance

Return type *BoundingBox*

static from_xyxy(*top_left*, *bottom_right*)
Create a bounding box object from the top-left point and bottom-right point

Parameters

- **top_left** (*np.array*) – array of form [x, y]
- **bottom_right** (*(np.array*) – array of form [x, y]

Returns a new bounding box instance

Return type *BoundingBox*

static from_yolo(*center*, *width*, *height*, *image_dimension*)
Create a bounding box object from YOLO formatted data

Parameters

- **center** (*np.array*) – the center coordinate of the box (x, y). Scaled [0, 1]
- **width** (*float*) – the width of the bounding box [0, 1]
- **height** (*float*) – the height of the bounding box [0, 1]
- **image_dimension** (*np.array*) – the dimensions of the image (row, cols)

Returns a new bounding box instance

Return type *BoundingBox*

property height

property points

to_xywh()
Returns the top-left point, width, and height

Returns tuple of form (np.array, float, float)

Return type tuple

to_xyxy()
Returns the top-left and bottom-right coordinate

Returns tuple of form (np.array, np.array)

Return type tuple

to_yolo(*image_dimension*)
Generates a YOLO formatted np.array with center_x, center_y, width, height

Parameters

- **image_dimension** (*np.array*) – array with image dimensions of form
- **(rows** (*rows, cols, depth*) –
- **or** (*cols)*) –

Returns array with YOLO formatted bounding box

Return type np.array

validate_points(*image_dimension*)
Make sure all the bounding box points are within an image's dimensions

Parameters

- **image_dimension** (*np.array*) – array with the image's dimensions
- **form** (*in*) –

Returns whether the points are valid

Return type bool

property width

bbox_utils.bbox_3d module

3D bounding box module. This file is modified from https://github.com/varunagrawal/bbox/blob/master/bbox/bbox3d.py.

class bbox_utils.bbox_3d.**BoundingBox3D**(*x*, *y*, *z*, *length=1*, *width=1*, *height=1*, *rw=1*, *rx=0*, *ry=0*, *rz=0*, *q=None*, *euler_angles=None*, *is_center=True*)

Bases: object

Class for 3D Bounding Boxes (3-orthotope). It takes either the center of the 3D bounding box or the back-bottom-left corner, the width, height and length of the box, and quaternion values to indicate the rotation. :param x: X axis coordinate of 3D bounding box. Can be either center of bounding box or back-bottom-left corner. :type x: float :param y: Y axis coordinate of 3D bounding box. Can be either center of bounding box or back-bottom-left corner. :type y: float :param z: Z axis coordinate of 3D bounding box. Can be either center of bounding box or back-bottom-left corner. :type z: float :param length: The length of the box (default is 1).

This corresponds to the dimension along the x-axis.

Parameters

- **width** (float, optional) – The width of the box (default is 1). This corresponds to the dimension along the y-axis.
- **height** (float, optional) – The height of the box (default is 1). This corresponds to the dimension along the z-axis.
- **rw** (float, optional) – The real part of the rotation quaternion (default is 1).
- **rx** (int, optional) – The first element of the quaternion vector (default is 0).
- **ry** (int, optional) – The second element of the quaternion vector (default is 0).
- **rz** (int, optional) – The third element of the quaternion vector (default is 0).
- **euler_angles** (list or ndarray of float, optional) – Sequence of euler angles in *[x, y, z]* rotation order (the default is None).
- **is_center** (*bool*, optional) – Flag to indicate if the provided coordinate is the center of the box (the default is True).

property center

Attribute to access center coordinates of box in (x, y, z) format. Can be set to list or ndarray of float. :returns: 3-dimensional vector representing (x, y, z) coordinates of the box. :rtype: ndarray of float

Raises **ValueError** – If *c* is not a vector/list of length 3.

copy()

property cx

X coordinate of center.

Type float

property cy

Y coordinate of center.

Type float

property cz

Z coordinate of center.

Type float

property edges
Get the edge connections for the cube

Returns list of edges of the cube

Return type np.array

classmethod from_center_dimension_euler(*center*, *dimension*, *euler_angles=None*)
Factory function to create BoundingBox3D from center, dimension, and euler arrays. Can pass in either np.arrays or Python lists.

Parameters

- **center** (*list*) – list of length 3
- **dimension** (*list*) – list of length 3
- **euler_angles** (*list, optional*) – list of length 3. Defaults to None.

Returns a new 3D bounding box object

Return type *BoundingBox3D*

classmethod from_xyzxyz(*xyz1*, *xyz2*)

property height
The height of the box.

Type float

property length
Length of the box.

Type float

property p
Attribute to access ndarray of all corners of box in order. :returns: All corners of the bounding box in order.

The order goes bottom->top and clockwise starting from the bottom-left point.

Return type ndarray of float

property p1
Back-left-bottom point.

Type float

property p2
Back-right-bottom point.

Type float

property p3
Front-right-bottom point.

Type float

property p4
Front-left-bottom point.

Type float

property p5
Back-left-top point.

Type float

property p6
Back-right-top point.

Type float

property p7
Front-right-top point.

Type float

property p8
Front-left-top point.

Type float

property q
Syntactic sugar for the rotation quaternion of the box. Returns

ndarray of float: Quaternion values in (w, x, y, z) form.

property quaternion
The rotation quaternion. :returns: Quaternion values in (w, x, y, z) form. :rtype: ndarray of float

property triangle_vertices
Get triangle vertices to use when plotting a triangular mesh

Returns Triangular vertices of the cube

Return type np.array

property width
The width of the box.

Type float

bbox_utils.image module

class bbox_utils.image.**Image**(*image*, **args*, ***kwargs*)
Bases: object

display(*image=None*, *title=None*, *library='matplotlib'*)
Display an image using a library of your choice.

Parameters

- **image** (*np.ndarray, optional*) – the image to display. If none specified, uses self.image.
- **title** (*str, optional*) – the title to use. Defaults to None.
- **library** (*str, optional*) – the library to use to display the image. Defaults to "matplotlib". Can also choose "opencv".

Raises **ValueError** – an error if an invalid library argument is passed.

display_bbox(*bbox*, *color=(0, 0, 255)*, **args*, ***kwargs*)
Display a single bounding box

Parameters

- **bbox** (BoundingBox) – a single bounding box

- **color** (*tuple, optional*) – color of the bounding box in BGR. Defaults to (0, 0, 255).

display_bboxes(*bboxes*, *colors*, **args*, ***kwargs*)
Display a list of bounding boxes

Parameters

- **bboxes** (*list(BoundingBox)*) – a list of bounding boxes
- **color** (*str or list(str)*) – a list of colors for each bounding box. Color should be specified in BGR.

classmethod display_cv2(*image*, *title=None*)
Display an image using OpenCV.

Parameters

- **image** (*np.ndarray*) – a numpy ndarray in BGR format.
- **title** (*str, optional*) – the title to give the plot. Defaults to None.

classmethod display_matplotlib(*image*, *title=None*)
Display an image using matplotlib.

Parameters

- **image** (*np.ndarray*) – a numpy ndarray in BGR format.
- **title** (*str, optional*) – the title to give the plot. Defaults to None.

classmethod load_from_file(*file_path*, **args*, ***kwargs*)
Loads an image from a file

Parameters **file_path** (*str*) – the path to the file

classmethod validate_image(*image*)
Validate an image object

Parameters **image** (*obj*) – image to validate

Returns whether the image is valid.

Return type bool

bbox_utils.point_cloud module

class bbox_utils.point_cloud.**PointCloud**(*point_cloud*, **args*, ***kwargs*)
Bases: object

crop(*bbox*)
Extract a point cloud from a 3D bounding box.

Source: https://stackoverflow.com/a/65350251/6942666

Parameters **bbox** (BoundingBox3D) – a 3D bounding box

Returns

a new point cloud with just the points within the bounding box.

Return type *PointCloud*

display(*size=2*)

display_bbox(*bbox*, *color='#ff0000'*, *size=2*, **args*, ***kwargs*)
Display a single bounding box

Parameters

- **bbox** (BoundingBox) – a single bounding box
- **color** (*string, optional*) – a valid Plotly color. Defaults to '#ff0000'

Returns a Plotly figure

Return type Figure

display_bboxes(*bboxes*, *colors='#ff0000'*, *size=2*, **args*, ***kwargs*)
Display a list of bounding boxes

Parameters

- **bboxes** (*list(BoundingBox)*) – a list of bounding boxes
- **color** (*str or list(str)*) – a valid Plotly color: The 'color' property is a color and may be specified as:
  - A hex string (e.g. '#ff0000')
  - An rgb/rgba string (e.g. 'rgb(255,0,0)')
  - An hsl/hsla string (e.g. 'hsl(0,100%,50%)')
  - An hsv/hsva string (e.g. 'hsv(0,100%,100%)')
  - **A named CSS color:** aliceblue, antiquewhite, aqua, aquamarine, azure, beige, bisque, black, blanchedalmond, blue, blueviolet, brown, burlywood, cadetblue, chartreuse, chocolate, coral, cornflowerblue, cornsilk, crimson, cyan, darkblue, darkcyan, darkgoldenrod, darkgray, darkgrey, darkgreen, darkkhaki, darkmagenta, darkolivegreen, darkorange, darkorchid, darkred, darksalmon, darkseagreen, darkslateblue, darkslategray, darkslategrey, darkturquoise, darkviolet, deeppink, deepskyblue, dimgray, dimgrey, dodgerblue, firebrick, floralwhite, forestgreen, fuchsia, gainsboro, ghostwhite, gold, goldenrod, gray, grey, green, greenyellow, honeydew, hotpink, indianred, indigo, ivory, khaki, lavender, lavenderblush, lawngreen, lemonchiffon, lightblue, lightcoral, lightcyan, lightgoldenrodyellow, lightgray, lightgrey, lightgreen, lightpink, lightsalmon, lightseagreen, lightskyblue, lightslategray, lightslategrey, lightsteelblue, lightyellow, lime, limegreen, linen, magenta, maroon, mediumaquamarine, mediumblue, mediumorchid, mediumpurple, mediumseagreen, mediumslateblue, mediumspringgreen, mediumturquoise, mediumvioletred, midnightblue, mintcream, mistyrose, moccasin, navajowhite, navy, oldlace, olive, olivedrab, orange, orangered, orchid, palegoldenrod, palegreen, paleturquoise, palevioletred, papayawhip, peachpuff, peru, pink, plum, powderblue, purple, red, rosybrown, royalblue, rebeccapurple, saddlebrown, salmon, sandybrown, seagreen, seashell, sienna, silver, skyblue, slateblue, slategray, slategrey, snow, springgreen, steelblue, tan, teal, thistle, tomato, turquoise, violet, wheat, white, whitesmoke, yellow, yellowgreen
  - A number that will be interpreted as a color

  according to mesh3d.colorscale

classmethod load_from_file(*file_path*, **args*, ***kwargs*)
Loads a point cloud from a file

Parameters **file_path** (*str*) – the path to the file

property number_of_points
The number of points within a point cloud.

Returns the number of points within a point cloud.

Return type int

property points
Get a np.ndarray representation of the point cloud.

Returns the point cloud's points

Return type np.ndarray

classmethod validate_point_cloud(*point_cloud*)
Validates the point cloud

Parameters **image** (*obj*) – image to validate

Returns whether the image is valid.

Return type bool

bbox_utils.utils module

bbox_utils.utils.**in_google_colab**()
Checks to see whether currently running in Google Colab.

Returns True or False

Return type bool

bbox_utils.utils.**order_points**(*pts*)
Orders points in form [top left, top right, bottom right, bottom left]. Source: https://www.pyimagesearch.com/2016/03/21/ordering-coordinates-clockwise-with-python-and-opencv/

Parameters **pts** (*np.ndarray*) – list of points of form [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]

Returns [description]

Return type [type]

bbox_utils.utils.**point_within_dimensions**(*point*, *image_dimensions*)
Checks to see if a point falls inside an image's dimension. Works for any number of dimensions. Acceptable range is [0, dim)

Parameters

- **point** (*np.array*) – array with the point's coordinates
- **image_dimensions** (*np.array*) – array with the image dimensions

Returns whether the point lies within the dimensions

Return type bool

bbox_utils.utils.**pointwise_distance**(*pts1*, *pts2*)
Calculates the distance between pairs of points

Parameters

- **pts1** (*np.ndarray*) – array of form [[x1, y1], [x2, y2], . . .]
- **pts2** (*np.ndarray*) – array of form [[x1, y1], [x2, y2], . . .]

Returns distances between corresponding points

Return type np.array

bbox_utils.utils.**round_np**(*np_arr*)
Rounds values in a numpy array to the nearest integer

Parameters **np_arr** (*np.array*) – numpy array to round

Returns numpy array of type np.int32

Return type np.array(int32)

bbox_utils.utils.**round_scalar**(*scalar*)
Rounds a scalar to the nearest integer

Parameters **scalar** (*float*) – scalar to round

Returns input rounded to the nearest int

Return type int

Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

N

O

P

Q

R

T

V

W