bbox$_visualizer Documentation$

Release 0.1.0

Shoumik Sharar Chowdhury

Jul 08, 2023

Contents:

1 Installation **3**
- 1.1 Stable release . . . 3
- 1.2 From sources . . . 3

2 Usage **5**

3 Project Info **11**
- 3.1 Contributing . . . 11
- 3.2 History . . . 13
- 3.3 Credits . . . 13

4 Indices and tables **15**

Python Module Index **17**

Index **19**

bbox-visualizer helps you easily draw bounding boxes with their corresponding labels after detecting them using any object detection method. The bounding boxes are expected to be in the format `(xmin, ymin, xmax, ymax)`.

CHAPTER 1

Installation

1.1 Stable release

To install bbox-visualizer, run this command in your terminal:

```
$ pip install bbox-visualizer
```

This is the preferred method to install bbox-visualizer, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for bbox_visualizer can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/shoumikchow/bbox-visualizer
```

Or download the tarball:

```
$ curl -OJL https://github.com/shoumikchow/bbox-visualizer/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

or

```
$ pip install . --user
```

CHAPTER 2

Usage

```
import bbox_visualizer as bbv
```

bbox_visualizer.bbox_visualizer.**add_T_label**(*img*, *label*, *bbox*, *size=1*, *thickness=2*, *draw_bg=True*, *text_bg_color=(255, 255, 255)*, *text_color=(0, 0, 0)*)

adds a T label to the rectangle, originating from the top of the rectangle

Parameters

- **img** (*ndarray*) – the image on which the T label is to be written/drawn, preferably the image with the rectangular bounding box drawn
- **label** (*str*) – the text (label) to be written
- **bbox** (*list*) – a list containing x_min, y_min, x_max and y_max of the rectangle positions
- **size** (*int, optional*) – size of the label, by default 1
- **thickness** (*int, optional*) – thickness of the label, by default 2
- **draw_bg** (*bool, optional*) – if True, draws the background of the text, else just the text is written, by default True
- **text_bg_color** (*tuple, optional*) – the background color of the label that is filled, by default (255, 255, 255)
- **text_color** (*tuple, optional*) – color of the text (label) to be written, by default (0, 0, 0)

Returns the image with the T label drawn/written

Return type ndarray

bbox_visualizer.bbox_visualizer.**add_label**(*img*, *label*, *bbox*, *size=1*, *thickness=2*, *draw_bg=True*, *text_bg_color=(255, 255, 255)*, *text_color=(0, 0, 0)*, *top=True*)

adds label, inside or outside the rectangle

Parameters

- **img** (*ndarray*) – the image on which the label is to be written, preferably the image with the rectangular bounding box drawn
- **label** (*str*) – the text (label) to be written
- **bbox** (*list*) – a list containing x_min, y_min, x_max and y_max of the rectangle positions
- **size** (*int, optional*) – size of the label, by default 1
- **thickness** (*int, optional*) – thickness of the label, by default 2
- **draw_bg** (*bool, optional*) – if True, draws the background of the text, else just the text is written, by default True
- **text_bg_color** (*tuple, optional*) – the background color of the label that is filled, by default (255, 255, 255)
- **text_color** (*tuple, optional*) – color of the text (label) to be written, by default (0, 0, 0)
- **top** (*bool, optional*) – if True, writes the label on top of the bounding box, else inside, by default True

Returns the image with the label written

Return type ndarray

bbox_visualizer.bbox_visualizer.**add_multiple_T_labels**(*img, labels, bboxes, draw_bg=True, text_bg_color=(255, 255, 255), text_color=(0, 0, 0)*)

adds T labels to the rectangles, each originating from the top of the rectangle

Parameters

- **img** (*ndarray*) – the image on which the T labels are to be written/drawn, preferably the image with the rectangular bounding boxes drawn
- **labels** (*list*) – the texts (labels) to be written
- **bboxes** (*list*) – a list of lists, each inner list containing x_min, y_min, x_max and y_max of the rectangle positions
- **draw_bg** (*bool, optional*) – if True, draws the background of the texts, else just the texts are written, by default True
- **text_bg_color** (*tuple, optional*) – the background color of the labels that are filled, by default (255, 255, 255)
- **text_color** (*tuple, optional*) – color of the texts (labels) to be written, by default (0, 0, 0)

Returns the image with the T labels drawn/written

Return type ndarray

bbox_visualizer.bbox_visualizer.**add_multiple_labels**(*img, labels, bboxes, size=1, thickness=2, draw_bg=True, text_bg_color=(255, 255, 255), text_color=(0, 0, 0), top=True*)

add labels, inside or outside the rectangles

Parameters

- **img** (*ndarray*) – the image on which the labels are to be written, preferably the image with the rectangular bounding boxes drawn

- **labels** (*list*) – a list of string of the texts (labels) to be written
- **bboxes** (*list*) – a list of lists, each inner list containing x_min, y_min, x_max and y_max of the rectangle positions
- **draw_bg** (*bool, optional*) – if True, draws the background of the texts, else just the texts are written, by default True
- **text_bg_color** (*tuple, optional*) – the background color of the labels that are filled, by default (255, 255, 255)
- **text_color** (*tuple, optional*) – color of the texts (labels) to be written, by default (0, 0, 0)
- **top** (*bool, optional*) – if True, writes the labels on top of the bounding boxes, else inside, by default True

Returns the image with the labels written

Return type ndarray

bbox_visualizer.bbox_visualizer.**draw_flag_with_label**(*img, label, bbox, size=1, thickness=2, write_label=True, line_color=(255, 255, 255), text_bg_color=(255, 255, 255), text_color=(0, 0, 0)*)

draws a pole from the middle of the object that is to be labeled and adds the label to the flag

Parameters

- **img** (*ndarray*) – the image on which the flag is to be drawn
- **label** (*str*) – label that is written inside the flag
- **bbox** (*list*) – a list containing x_min, y_min, x_max and y_max of the rectangle positions
- **size** (*int, optional*) – size of the label, by default 1
- **thickness** (*int, optional*) – thickness of the label, by default 2
- **write_label** (*bool, optional*) – if True, writes the label, otherwise, it's just a vertical line, by default True
- **line_color** (*tuple, optional*) – the color of the pole of the flag, by default (255, 255, 255)
- **text_bg_color** (*tuple, optional*) – the background color of the label that is filled, by default (255, 255, 255)
- **text_color** (*tuple, optional*) – color of the text (label) to be written, by default (0, 0, 0)

Returns the image with flag drawn and the label written in the flag

Return type ndarray

bbox_visualizer.bbox_visualizer.**draw_multiple_flags_with_labels**(*img, labels, bboxes, write_label=True, line_color=(255, 255, 255), text_bg_color=(255, 255, 255), text_color=(0, 0, 0)*)

draws poles from the middle of the objects that are to be labeled and adds the labels to the flags

Parameters

- **img** (*ndarray*) – the image on which the flags are to be drawn
- **labels** (*list*) – labels that are written inside the flags
- **bbox** (*list*) – a list of lists, each inner list containing x_min, y_min, x_max and y_max of the rectangle positions
- **write_label** (*bool, optional*) – if True, writes the labels, otherwise, it's just a vertical line for each object, by default True
- **line_color** (*tuple, optional*) – the color of the pole of the flags, by default (255, 255, 255)
- **text_bg_color** (*tuple, optional*) – the background color of the labels that are filled, by default (255, 255, 255)
- **text_color** (*tuple, optional*) – color of the texts (labels) to be written, by default (0, 0, 0)

Returns the image with flags drawn and the labels written in the flags

Return type ndarray

`bbox_visualizer.bbox_visualizer.`**`draw_multiple_rectangles`**(*img, bboxes, bbox_color=(255, 255, 255), thickness=3, is_opaque=False, alpha=0.5*)

draws multiple rectangles

img [ndarray] the actual image

bboxes [list] a list of lists, each inner list containing x_min, y_min, x_max and y_max of the rectangle positions

bbox_color [tuple, optional] the color of the boxes, by default (255,255,255)

thickness [int, optional] thickness of the outline of the boxes, by default 3

is_opaque [bool, optional] if False, draws solid rectangular outlines for rectangles. Else, filled rectangles which are semi transparent, by default False

alpha [float, optional] strength of the opacity, by default 0.5

Returns the image with the bounding boxes drawn

Return type ndarray

`bbox_visualizer.bbox_visualizer.`**`draw_rectangle`**(*img, bbox, bbox_color=(255, 255, 255), thickness=3, is_opaque=False, alpha=0.5*)

Draws the rectangle around the object

Parameters

- **img** (*ndarray*) – the actual image
- **bbox** (*list*) – a list containing x_min, y_min, x_max and y_max of the rectangle positions
- **bbox_color** (*tuple, optional*) – the color of the box, by default (255,255,255)
- **thickness** (*int, optional*) – thickness of the outline of the box, by default 3

- **is_opaque** (*bool, optional*) – if False, draws a solid rectangular outline. Else, a filled rectangle which is semi transparent, by default False
- **alpha** (*float, optional*) – strength of the opacity, by default 0.5

Returns the image with the bounding box drawn

Return type ndarray

CHAPTER 3

Project Info

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/shoumikchow/bbox-visualizer/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

bbox-visualizer could always use more documentation, whether as part of the official bbox_visualizer docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/shoumikchow/bbox-visualizer/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.2 Get Started!

Ready to contribute? Here's how to set up `bbox_visualizer` for local development.

1. Fork the `bbox_visualizer` repo on GitHub.
2. Clone your fork locally::

```
git clone git@github.com:your_name_here/bbox_visualizer.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development::

```
mkvirtualenv bbox_visualizer
cd bbox_visualizer/
python setup.py develop
```

4. Create a branch for local development::

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8::

```
flake8 bbox_visualizer demo
```

   To get flake8, just pip install it into your virtualenv.

6. Commit your changes and push your branch to GitHub::

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
2. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy.

3.2 History

3.2.1 0.1.0 (2020-09-24)

- First release on PyPI.

3.3 Credits

3.3.1 Development Leads

- Shoumik Sharar Chowdhury (@shoumikchow)

3.3.2 Contributors

None yet. Why not be the first?

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

bbox_visualizer.bbox_visualizer, 5

Index

A

`add_label()` (*in module bbox_visualizer.bbox_visualizer*), 5
`add_multiple_labels()` (*in module bbox_visualizer.bbox_visualizer*), 6
`add_multiple_T_labels()` (*in module bbox_visualizer.bbox_visualizer*), 6
`add_T_label()` (*in module bbox_visualizer.bbox_visualizer*), 5

B

`bbox_visualizer.bbox_visualizer` (*module*), 5

D

`draw_flag_with_label()` (*in module bbox_visualizer.bbox_visualizer*), 7
`draw_multiple_flags_with_labels()` (*in module bbox_visualizer.bbox_visualizer*), 7
`draw_multiple_rectangles()` (*in module bbox_visualizer.bbox_visualizer*), 8
`draw_rectangle()` (*in module bbox_visualizer.bbox_visualizer*), 8