limhpone ML fef7e4e · 2 hours ago

312 lines (312 loc) · 8.05 KB

Preview Code Blame Raw

Tutorial: Image Annotation for Computer Vision

- Annotation is the process of labeling images so machines can learn to "see" like humans.
- It's the ground truth used to train AI models in tasks such as object detection and segmentation.

Types of annotation

- Bounding box
- Segmentation

Bounding Box

- Definition: The simplest form of annotation. Draw a rectangle around the object of interest.
- Use case: Object detection (e.g., detecting cats, cars, faces).
- How it's stored: Coordinates of top-left corner (x, y), plus width and height.
- Advantages: Simple, fast, small annotation files.
- Limitations: Doesn't capture the exact shape of the object (lots of background included).

Example:

- A box drawn around a cat in an image.
- Supported by formats: YOLO, COCO, Pascal VOC.

Segmentation

- Definition: More precise labeling, where each object is outlined by its exact shape.

Types:

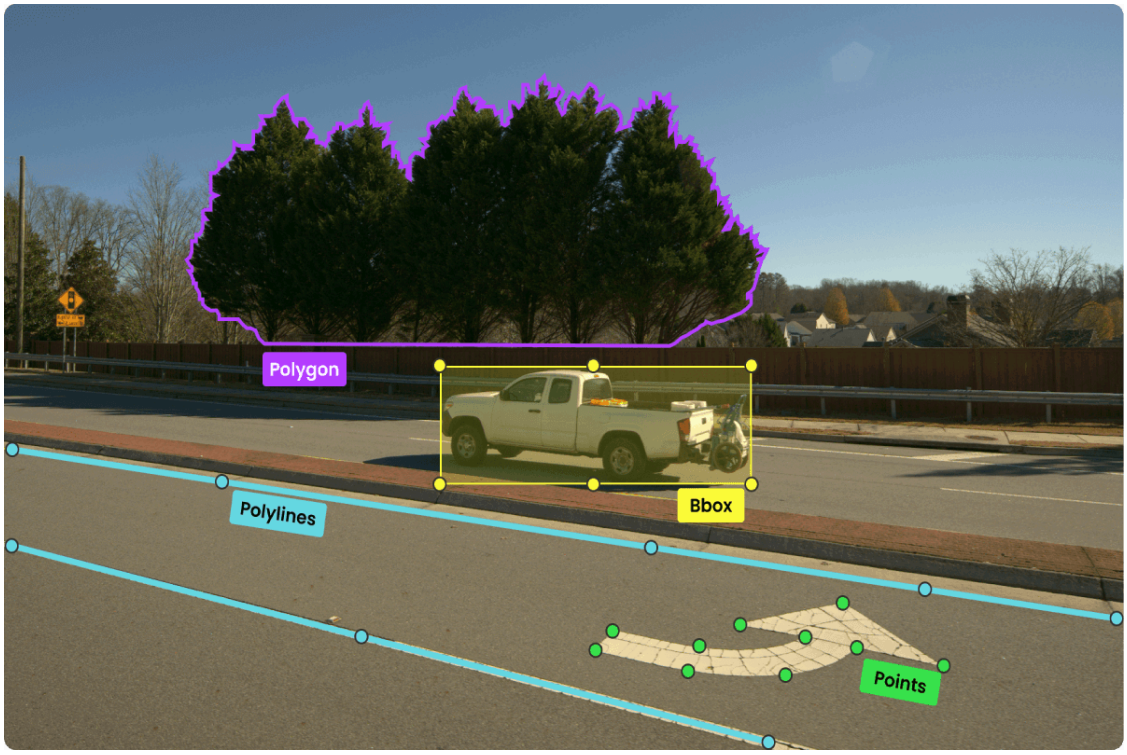
- Polygonal segmentation: draw polygons around objects.
- Pixel-wise segmentation (masking): each pixel is assigned a class label.
- Use case: Semantic segmentation (road/lane markings), instance segmentation

(detecting multiple objects of the same class).

- How it's stored: Polygons (list of points) or binary masks.
- Advantages: Very accurate, useful for detailed tasks.
- Limitations: Time-consuming, large annotation files.

Example:

- Carefully outlining the cat's ears, tail, and body instead of just using a rectangle.
- Supported by formats: COCO, Pascal VOC (mask), LabelMe JSON.



Annotation Tools

- **For bounding boxes**: labelImg, VGG Image Annotator (VIA) , CVAT.
- **For segmentation**: LabelMe, VGG Image Annotator (VIA), CVAT.

Annotation Formats

Different datasets and frameworks use different annotation formats. Let's look at the **three most common ones** with examples.

COCO Format (JSON)

- **Supports**: Bounding boxes, segmentation, keypoints

- **Bounding box format**: `[x_min, y_min, width, height]`
- **Widely used in**: Detectron2, MMDetection, COCO dataset

Example:

```
{
  "images": [
    {
      "id": 1,
      "file_name": "cat.jpg",
      "width": 800,
      "height": 600
    }
  ],
  "annotations": [
    {
      "id": 1,
      "image_id": 1,
      "category_id": 1,
      "bbox": [100, 150, 200, 300],
      "area": 60000,
      "iscrowd": 0
    }
  ],
  "categories": [
    {
      "id": 1,
      "name": "cat",
      "supercategory": "animal"
    }
  ]
}
```

Pascal VOC Format (XML)

- **Supports**: Bounding boxes, segmentation masks
- **Bounding box format**: `(xmin, ymin, xmax, ymax)`
- **Widely used in**: TensorFlow Object Detection API, Pascal VOC dataset

Example

```
<annotation>
    <folder>images</folder>
    <filename>cat.jpg</filename>
    <size>
        <width>800</width>
        <height>600</height>
        <depth>3</depth>
    </size>
    <object>
        <name>cat</name>
```

```
            <pose>Unspecified</pose>
            <truncated>0</truncated>
            <difficult>0</difficult>
            <bndbox>
                <xmin>100</xmin>
                <ymin>150</ymin>
                <xmax>300</xmax>
                <ymax>450</ymax>
            </bndbox>
        </object>
    </annotation>
```

YOLO Format (TXT)

Supports: Bounding boxes only

Bounding box format: `class_id x_center y_center width height`

(all values normalized between `0–1`)

Example `0 0.25 0.5 0.25 0.5`

Explanation (for an image of width 800 and height 600):

- 0 → class ID for "cat"
- x_center = (100 + 200/2) / 800 = 0.25
- y_center = (150 + 300/2) / 600 = 0.5
- width = 200 / 800 = 0.25
- height = 300 / 600 = 0.5

VGG IMAGE ANNOTATOR

In [ ]:
```
!git clone https://github.com/nearkyh/via-1.0.5.git
```

1. Open the folder via-1.0.5
2. Run via.html

Follow the demonstration in the lab for simple bbox and segmentation.

labelimg

If activation scripts are blocked, run the process-scope bypass first:

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Register the venv as Jupyter Kernel

```
python -m ipykernel install --user --name=cv-lab-venv --display-name
'Python (cv-lab-venv)'
```

Install pip

```
python -m ensurepip --upgrade
```

```
python -m pip install --upgrade pip setuptools wheel
```

Install package labelimg

```
python -m pip install labelimg
```

Run labelimg

```
labelimg
```

Check more on labelimg here: https://pypi.org/project/labelImg/