Humanising Autonomy Python Challenge

Introduction

Humanising Autonomny have developed a detection model which looks at videos and for each frame identifies the locations of objects and our confidence that we have identified the class of the object correctly. This is an example of the model output for a 3 frame video:


```
{
"1": {
       "bounding boxes": [
       [
              867,
              712,
              24,
              22
       ]
       ],
       "detection scores": [
       0.5579711198806763
       ],
       "detected classes": [
       "car"
       ]
},
"2": {
       "bounding boxes": [
       [
              865,
              712,
              24,
              21
       ],
       [
              745,
              692,
              112,
              95
       ],
       [
              973,
              717,
              16,
              32
       ],
       [
              1081,
              710,
              20,
              48
       ]
       ],
       "detection scores": [
       0.6426031589508057,
```

```
        0.6739102005958557,
        0.6911854147911072,
        0.7216816544532776
        ],
        "detected classes": [
        "car",
        "car",
        "person",
        "person"
        ]
},
"3": {
        "bounding boxes": [
        [
                868,
                707,
                25,
                19
        ],
        [
                843,
                708,
                28,
                33
        ],
        [
                865,
                711,
                25,
                21
        ],
        [
                744,
                688,
                110,
                101
        ],
        [
                1080,
                704,
                21,
                54
        ]
        ],
        "detection scores": [
        0.5245904922485352,
        0.5434021949768066,
        0.5852199792861938,
        0.7162638306617737,
        0.7476180791854858
        ],
        "detected classes": [
        "car",
        "car",
        "car",
        "car",
        "person"
```

```
        ]
}}
```

The model outputs are stored as a dictionary in a json file, with frame ids as key. Each frame contains a list of bounding boxes, a list of scores and a list of classes. You can match them all by taking the element at the same index in all the lists. The bounding boxes are stored in a list containing first their top left corner coordinates (x and y pixel coordinates) then their width and height. Ex: [150, 36, 60, 45] The top left corner coordinates on the frame are (150, 36), the bounding box width is 60 and the bounding box height is 45.

Files

You should have received a number of files from us:

instructions.txt

this file

display_video.py

an example OpenCV file which displays a video at half its native resolution

resources/video_{1,2,3}.mp3

3 example video files

resouces/video_{1,2,3}_detection.json

the output of our detection model in json format for each of the three videos.

requirements.pdf

the library requirements for the example code

In order for the display_video.py to run, you must be running python 3.10.5.

Warm up

We'll provide you with open-cv code to open and display a video, but we would like you to overlay the bounding boxes on the screen with different colors for different classes.

Tracking Test

A classic, hard CV problem is how to track individual objects from frame to frame. The best minds in the world are still struggling with this problem.

We'd like you to have a crack at it. We're not expecting you to solve this problem (obviously), but we want to see how you write code and we look forward to discussing with you, the challenges you faced.

For pedestrians, only, write a program that uses their bounding boxes to track them for as many frames as you can manage. Even if you only manage a few frames of tracking, we'll be impressed.

Here are some guidelines about an object tracking using Euclidean distance:

- Generate an id number per new object and display them on the screen, so that we can track individual objects.
- Use the previous frame as the referenced frame: save the object coordinates of the previous frame and compare them to the current frame.
- Use the Euclidian distance to match objects between the current frame and referenced frame.

Submission

Please upload your submission to a public github repository and let Julia have the URL.