Interview Task: Real-Time Object Tracking Simulation Documentation

A quick and minimal documentation of the interview task.

Known problems (with luxonis_task_video.mp4 as input only):

- The detection of objects is not pixel perfect.
- The script might not detect objects at 30 frames/s on lower end systems
- If a rectangle is overlapped, no calculation is done to keep track where, below the circle, the rectangle continues, meaning centroids get off-set.
- If a rectangle gets **completely** overlapped for 2 or more frames, the tracker will not keep up and assigns an object with a new ID (only 1 occurrence)

Use the object detector(s)

the detectors can be used as shown in the figure 1.

detect_circles(frame, frame number) method detects circles and takes a ***frame*** and ***frame number*** as arguments. It returns the detected circles and frame where the detected circles are deleted.

The detected circles have the following structure:

[ID, Frame number, 'Circle', x-coordinate, y-coordinate, radius, area, [B-channel, G-Channel, R-channel]]

detect_rectangles(frame, frame number) method detects rectangles and takes a ***frame*** and ***frame number*** as arguments. It returns the detected rectangles and any non-complete (overlapped) rectangles.

The detected rectangles have the following structure:

[ID, Frame number, 'Rectangle', x-coordinate, y-coordinate, [width, height], area, [B-channel, G-Channel, R-channel]]

The detected partial rectangles have the following structure:

[ID, Frame number, 'Partial_Rectangle', x-coordinate, y-coordinate, [[x-coordinate of point, y-coordinate of point]…], area, [B-channel, G-Channel, R-channel]]

```
circles, detect_frame = detect_circles(frame.copy(), i)
rectangles, partial_rectangles = detect_rectangles(detect_frame, i)
```

Figure 1. Example use of the detectors

Use the tracker

The tracker can be used as shown in the figure 2.

track(previous_objects, next_id, objects) method makes connections between objects across frames. It takes ***previous_objects, next_id*** and ***objects*** as arguments. It returns the new list with linked objects by IDs, and the next ID that can be used to assign an ID to the next new object.

```
combined, next_id = track(previous_objects, next_id, circles + rectangles + partial_rectangles)
```

Figure 2. Example use of the tracker

Run the app

As the app is only a simple python script, you can run the script as you would any other python scripts, from your favourite IDE or a command line. The name of the script is 'object_detection.py'. The script requires that the video (named 'luxonis_task_video.mp4') is in the same directory as the script, otherwise you need to update the path to the video in the script itself manually.

The following versions of python and libraries were used:

- Python version: 3.8.5
- cv2 (OpenCV) version: 4.6.0
- Numpy version: 1.19.2

However, the script should run just as fine with more up-to-date versions (as of 19.05.2024).

Once you run the script, a window with a real-time tracking opens up as shown in the figure 3.

After each frame is displayed, it gets saved to the output video, which is saved to the same directory as the script, named 'output_video.mp4'.

Additionally, a text file called 'data.txt' gets saved to the same directory as the script is in after each frame. The file contains information on all the detected objects across all the frames.

You can press the 'q' key to interrupt the script, but keep in mind that this also means only the information in the frames that you saw gets saved to the output video and the text file.

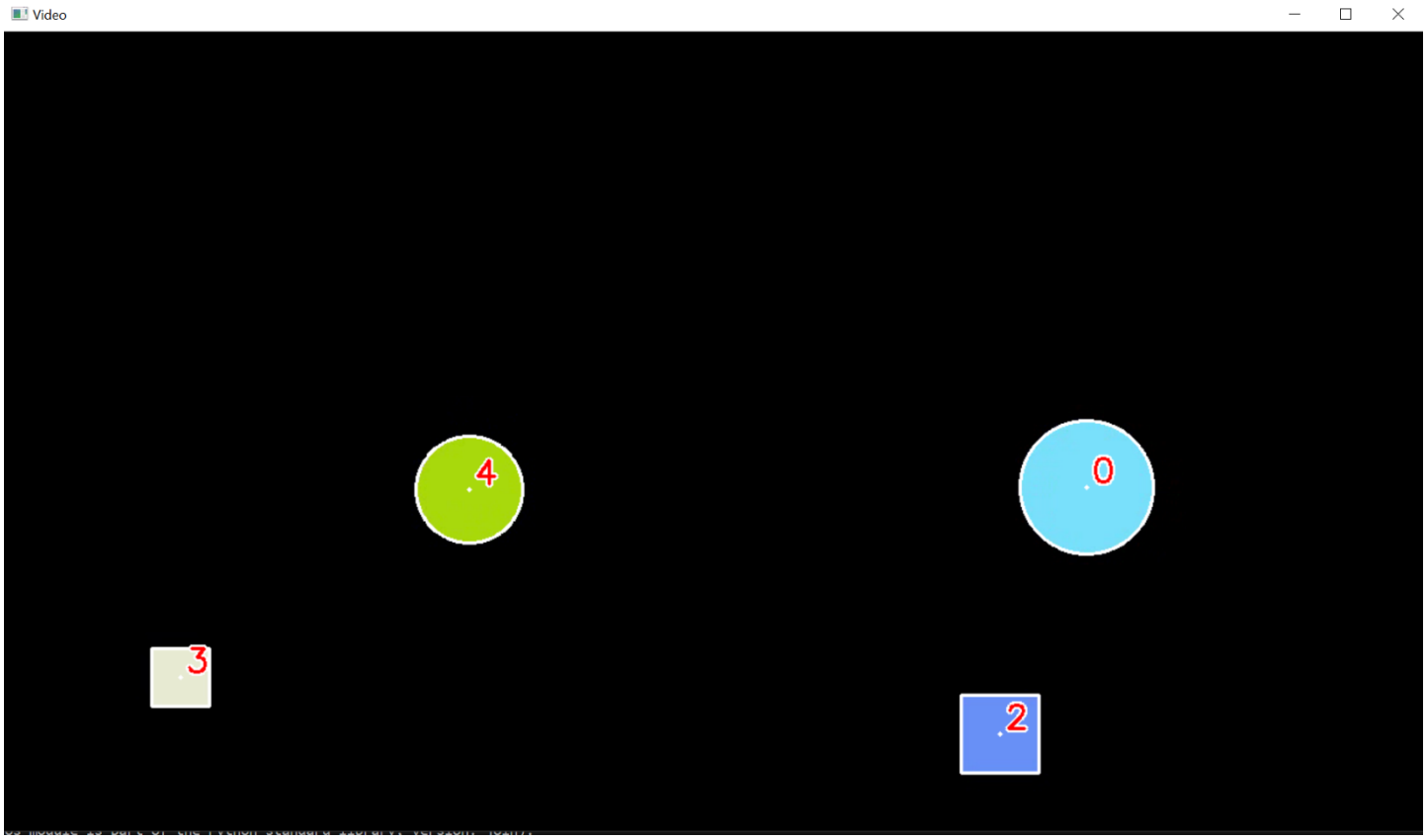

Figure 3. Window with a real-time tracking