Yolov8 object counting in real-time with webcam

* Create virtual environment: conda create -n yolov8 python=3.10
* Activate virtual environment: conda activate yolov8
* pip install ultralytics
* pip install supervision==0.3.0
* Using CLI 🡪 yolo detect predict model=yolov8l source=0 show=True
* Open vscode
* CTRL+SHIFT+P search for Python: Select Interpreter
* Select the virtual environment we created earlier
* Then, we want to test the main function:

A screen shot of a computer code

Description automatically generated

* Save the file and run in terminal.



* Now, let’s access the webcam
* First, import cv2
* In the main, we create a video capture with the index of camera 0
* In linux, we can check the camera index using ls -la /dev/ | grep video
* Access the webcam

A screen shot of a computer program

Description automatically generated

* Might get this error: OpenCV Error: Unspecified error (The function is not implemented. Rebuild the library with Windows, GTK+ 2.x or Carbon support. If you are on Ubuntu or Debian, install libgtk2.0-dev and pkg-config, then re-run cmake or configure script) in cvShowImage, file /io/opencv/modules/highgui/src/window.cpp, line 545
* pip uninstall opencv-python-headless
* pip uninstall opencv-python
* pip install opencv-python==4.7.0.68

CLI: yolo track model=yolov8n.pt source=0 show=True

Python script

A computer screen shot of a code

Description automatically generated

But we want to access every frame of the webcam to perform some manipulation. Therefore, we need to use another argument called *stream* in the model track. As a result, the program will produce a generator that enables us to process each frame separately.

A white background with black text

Description automatically generated

Now, we will use result.orig\_img to extract the frames. We use cv2.imshow to show the camera frames as to check whether this works or not.

A computer screen shot of a program code

Description automatically generated

Now we will add bounding box annotation using supervision.

A screen shot of a computer program

Description automatically generated

The problem now is that we only show the class ID instead of class name. We can add labels method in supervision to also show the class name.

A screen shot of a computer program

Description automatically generated

The tracker ID is stored in the boxes class for object detection models and inside mask class for object segmentation model. We also convert the pytorch tensor into a numpy array and cast the numpy array as an integer instead of float. Then, go back to the list comprehension of the labels and add tracker\_id. We also add if result.boxes.id is not None: so that the program does not crash whenever there is no detection in the frame.

If we want to exclude the person class from the tracking, we can do so by adding detections= detections[detections.class\_id !=0]

Now we can add the functionality of count on top of the tracker.

First we initialize the start and end of a line. Let say we take the centre of our frame as the line for counting. Then we can define out line\_zone and line\_zone annotator which is responsible for drawing the line and show the counter. In the for loop, we add line\_zone.trigger(detections) and line\_zone\_annotator.annotate(frame,line\_zone) to detect if the object has crossed the line and start annotating.

A screen shot of a computer program

Description automatically generated

