Multi-Object Tracking with ByteTrack

Tomasz Stańczyk
tomasz.stanczyk@inria.fr

Inria

3iA Côte d'Azur
Interdisciplinary Institute for Artificial Intelligence

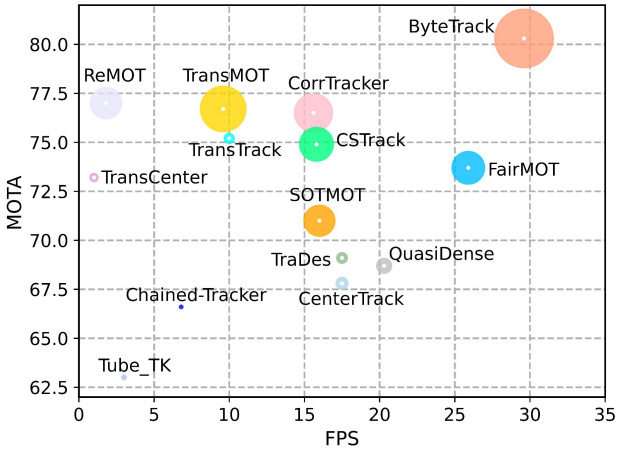
Overview

In this practical assignment you are going to use ByteTrack and slightly modify it to read your own detection outputs as the input

You can either:

- Work on Google Colab
- Work on your local machine (if you have a GPU)
- On the university cluster (if you have an access)

ByteTrack github repository:
https://github.com/ifzhang/ByteTrack



ByteTrack github

As in the previous assignment, go to the project github, **all the installation and usage instructions are there**.

Installation -> 1. Installing on the host machine
Do **not** perform: *2. Docker build*

As you are neither going to train the model nor evaluate it numerically, you can skip the *Data preparation* part

ByteTrack github

Based on the instructions in the Demo part, you will need to download the following pre-trained model: bytetrack_x_mot17.pth.tar (see *Model Zoo*)

However, you are also welcome to play with the other ones. Remember to adjust the experiment file (the -f argument) accordingly

Downloading model from the Google Drive into your local machine should go smoothly.

Loading it into your Google Colab space:

- Either make a copy to your Google Drive and then mount your drive to your Google Colab (you can find instructions how to do it online)
- Or use a specified command, e.g. *!curl -L link -o output_name*
  (search online for more commands on how to upload your Google Drive files)

Your tasks 1/2

At first, set up the environment and perform all the necessary steps to run the demo file of the ByteTrack. After running it successfully, find and see its output visual outputs

Browse the code a bit to understand where the actual tracking part is happening

See the paper https://arxiv.org/pdf/2110.06864.pdf, Algorithm 1, page 4
(also mentioned during the lecture, slide 20)

Your tasks 2/2

During the last week object detection assignment, your aim was to obtain the detection results on a video sequence. It required some modification/expansion of the YOLOX code.

Besides, you can see that the ByteTrack implementation is based on that of YOLOX.

Now, your task is to find the part in the ByteTrack code where detector outputs are passed to the tracker as inputs. Modify this code in such a manner that the detections from your previous week files can be read and passed as tracker inputs. Adjust the output text file format in your previous work if needed. You will need to read a few lines per frame
(= all detections from the given frame)

Again, it requires delving into the code and performing manual modifications yourself, thus more than just running instructions provided on github

Your tasks 2/2

Try to also do it with another detector, e.g. different version of Yolo, FasterRCNN or so

You will see that the tracking performance will vary based on (the quality of) the detections provided.

Obtain the visual outputs again and compare with those from the first part of this practical.

Notice that after changing the detection part from using the detector into reading the detections from a file, no GPU is needed.

Good luck!

And again do it earnestly, **you will learn a lot and it will become useful to you on the later stages of your studies and afterwards**.

It will help you understand how ByteTrack (and tracking in general) works and how to adjust it to your needs and inputs when you are already given some detection data.

When facing issues and errors, google it at first and see if someone has had such an issue before. That's how you learn in practice

Feel free to discuss the stuff with your classmates, but ensure it is you who does and understands the assignment

If you are seriously stuck and cannot proceed, you can contact me via e-mail

Happy learning!