École Polytechnique Fédérale de Lausanne

Deep learning for autonomous vehicles:

EPFL tandem race 2022.

Xinling LI
Nada GUERRAOUI
Thomas PEETERS
Kuan Lon VU

June 6, 2022

***Abstract* - Autonomous driving is no longer a distant future. It has become a more important area of research in recent years, especially thanks to deep learning. In this report, we discuss how to get a Loomo Robot to compete in the EPFL tandem race 2022 using deep learning based on state-of-the-art methods.**

1 Introduction

The project consists of three main steps, the first two (person detection and tracking) were trained and improved in Google Colab, then we tried to implement them to run on the V100 GPU server through the virtual environment, which acts as the middle-man between our model and the Loomo robot.

2 Milestone 1: Person detection

2.1 State of the art methods

Over the last years many detection algorithms using ML have been developed. Many are open source and can be adapted in a few lines of code. Yolov5 [1] and MediaPipe [2] are the two final ML models that we have selected for our detection.

YOLOv5 is a one staged, anchor based detector that can be used for real time detection when using a GPU. We use the weights that are pre-trained on large datasets.

We used the MediaPipe Pose solution that detects human pose in real time.

The goal was to use yolov5 to detect all the person on the frame and then to use MediaPipe Pose to detect a specific pose and assign this person as the person of interest.

2.2 Structure & Implementation

We started by implementing a custom Yolov5 for the detection of one person in the team. To do this, we built our own dataset using roboflow [3]. We made different preprocessing and data augmentation to make our detection more robust. This detection worked well, but it was too person-specific and non-generalisable, since we could not dynamically change the person of interest (POI).

Therefore, we decided to simply detect all persons with Yolov5 and then perform poses detection to make the detected person the POI. For person detection, we loaded a model coming from the Github repository[1]. We used the weight that were trained on the COCO dataset[4]. We only used here the person class detection. MediaPipe then was used to detect the pose of the person and if this pose corresponds to the pose that we have set (one hand above the shoulder and one hand below the shoulder where each hand should be aligned with the body) we set it as the person of interest. Since MediaPipe Pose detects only one pose on the image, if multiple person are detected we will crop each person and give them separately to the MediaPipe model so that all of them are detected correctly.

2.3 Result

Since those algorithms are trained on very large datasets, they worked well and gave us accurate results. Having an original pose is important so that any other group would not have the same pose detection. To make it more robust, we also tried to add gesture detection since MediaPipe Holistic allows hand detection. But when testing this method on a small image (160x120 pixels), the results were poor when we were far from the camera, so we decided to remove it and only kept the pose detection implementation.

3 Milestone 2: Person tracking

3.1 State of the art methods

We tried two types of tracking: non-detection-based and detection-based.

For the non-detection-based tracking, we used siamfc. Siamfc uses Siamese network[5] to compute the distance between two images. After initialisation, the image of the person of interest is stored, and we call it benchmark image. Then for each of the following frame, the algorithm tries to find the subregion in the new frame which has the closest distance with the benchmark image by Siamese network. This minimum distance subregion is the new position of the person of interest on this frame.

While siamfc is easy to understand and fast to run, it is not suitable for our task. Siamfc always outputs a subregion with minimum distance, so even the person of interest goes out of the frame, there is still a returned bounding box. Theoretically, the object within this bounding box can be anything as long as the distance between it and the benchmark image is small enough. This could result in the robot moving out of expectation. We set

a threshold of the distance between two images to be detected as the same one, this meaningless return value problem was solved, but a new problem appeared: sometimes it failed to locate the person of interest even though he or she was in frame. The threshold parameter highly depends on the environment and is difficult to tune.

The detection-based tracking we tried is Deepsort [6]. For each frame, it first calls the detector to get the position of all the objects (all persons in our case) in the frame; it then predicts the positions of the objects based on their positions from the last frame. Finally, kalman filter is used to combine these two positions and update the position for each object with the new position. Deepsort has two advantages over siamfc: (1) It keeps a unique ID for each object and its feature. When a new object appears, it will be first compared with the existing objects, if it is close enough (cosine distance under a threshold) with one existing object, it will be regarded as this object and will not be regarded as a new one. Therefore, if the POI goes out of the frame and comes back, deepsort is still able to re-detect and track them. (2) Deepsort does detection on every new frame, this means that we have more information to do the tracking compared with siamfc. This achieves more robust tracking.

After testing both algorithms, we found that deepsort indeed performed better than siamfc. So we selected deepsort as the tracking algorithm in implementation.

3.2 Structure & Implementation

We implemented yolov5 and deepsort based on a github repository from the AI Guy[7]. The original repository uses yolov4 and track all persons. We replaced it with yolov5 and added MediaPipe to do the initialisation. The workflow is as follows:

1. Use yolo5 to detect all the persons on the frame. For each person, use MediaPipe to check if he or she is doing the initialisation pose. If one person is doing the pose, keep its id and set this person as the POI. Initialisation phase ends.

2. Tracking phase starts. Use yolo5 and deepsort to track all the persons on frame, but only return the bounding box of the POI by the stored id.

3.3 Result

The algorithm functions mostly as expected. Sometimes problems occurs when the POI leaves the frame. After coming back, the POI is detected as a new person. This can be solved by increasing the distance threshold for two persons to be detected as the same one, but this brings another risk of mistaking another person as the POI. For the robot following task, we suppose that following a wrong person is more severe than not moving, so we keep the threshold a small value. If the POI is lost, initialisation is done again.

4 Milestone 3: Tandem Race

4.1 Implementation on the Loomo Robot

We created a virtual environment and installed the required packages on V100. Then, we implemented detector.py which interacts with client.py where the latter interacts with the Loomo robot. We also added some print message in client.py to get real-time tracking state.

4.2 Challenges

We found that our model can sometimes be sensitive to the lighting. We also observed that we cannot be too close or too far from the robot's camera to be detected. To reduce those uncertainties during the race, the algorithm will redo the initialisation (the detection of the POI) if the POI has not been detected and tracked in the previous 40 frames.

4.2.1 Finetuning parameters

max_age (number of frames the tracker retains):

- max_age is a deepsort parameter that has been set to 200 in milestone 2. If an ID tracked before has not been tracked for 200 frames, all its features will be forgotten - it will not be tracked anymore. For the redo-initialisation part, it is important to reduce this value so that there will never be two different tracking ID corresponding to the same person.
  Solution: As we redo the initialisation after 40 frames if the POI is lost, it makes more sense to also change the max_age to 40 frames so that the model no longer retains the previous tracking ID at the moment of re-initialisation.

References

[1] ultralytics. *yolov5*. https://github.com/ultralytics/yolov5. 2022.

[2] Camillo Lugaresi et al. "MediaPipe: A Framework for Building Perception Pipelines". In: *arXiv e-prints*, arXiv:1906.08172 (June 2019), arXiv:1906.08172. arXiv: 1906.08172 [cs.DC].

[3] Roboflow Inc. *Roboflow*. https://app.roboflow.com/nada-guerraoui/face-asp8m/13. 2022.

[4] COCO Consortium. *COCO*. https://cocodataset.org/#home.

[5] Luca Bertinetto et al. "Fully-convolutional siamese networks for object tracking". In: *European conference on computer vision*. Springer. 2016, pp. 850–865.

[6] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[7] The AI Guy. *yolo4-deepsort*. https://github.com/theAIGuysCode/yolov4-deepsort. 2020.