Tandem Race

Alaa Abboud, Olav Førland, Julia Richter, Benjamin Rike, Keven Sahin
EPFL, Switzerland

***Abstract*—This report presents our work and findings from participating in the Tandem Race at EPFL. We use a combination of the object detector YOLOv4 and the tracking algorithm DeepSort to enable a Segway Loomo robot to follow a person during the race. The DeepSort algorithm is customized to initiate on the person in the center of the frame and tuned on hyperparameters to track efficiently. During the race, we achieved a time of 1 minute.**

I. Introduction

This report describes our contributions from participating in the Tandem Race at EPFL. The race consists of developing a computer vision method for tracking a person of interest across consecutive frames, implementing it on the Loomo robot from Segway, and racing against the other teams in the course. Real-time object tracking is a fast-developing field. Traditionally, tracking has been performed offline with methods such as Multiple Hypothesis Tracking (MHT) [1] and the Joint Probabilistic Data Association Filter (JPDAF) [2]. However, in 2016, Simple Online and Realtime Tracking (SORT) redefined the field by combining the Hungarian algorithm and Kalman Filters to produce an algorithm several orders of magnitudes faster than previous tracking algorithms with state-of-the-art accuracy [3]. Shortly after, DeepSort integrated a deep association metric with SORT, obtaining even better results [4]. DeepSort has since been the go-to baseline algorithm for most online multi-object tracker applications.

II. Models and Methods

The problem of person-tracking is two-fold. Firstly, an object detector needs to detect every person in each frame. Since the detector runs in real-time, it needs fast inference speed. Secondly, the people detected in the current frame need to be associated with already tracked people from previous frames. We will refer to previously detected people as *tracks*.

We use YOLOv4, first presented by Bochkovskiy et al. [5], to detect people. YOLOv4 is a state-of-the-art object detector developed to support the speed demands of real-time applications without sacrificing accuracy. Thereafter, we utilize the DeepSort algorithm to combine the incoming detections with existing tracks. It consists of 5 key components - detection, estimation, feature extraction, target association, and track creation/destruction. The method starts by detecting every person in the frame using an efficient object detector, in our case, YOLOv4. The model extracts people's features through Person Re-Identification (Re-ID), and Kalman filters then estimate their positions in the next frame. Subsequently, incoming detections are associated with existing tracks by combining the estimations from the Kalman filters with the *features* of the people and using the Hungarian algorithm to find an ideal matching between existing tracks and incoming detections. New objects are added when a new detection is not matched with any previous frames and has been detected for *n_init* number of frames. Tracked objects are discarded when they have not been detected for *max_age* number of frames. For the Re-ID, we use the convolutional residual network from the original DeepSort paper. The network is pre-trained on the Mars dataset [6]. The network is an encoder that extracts and embeds the features of a person into a representation space. Then, the tracker uses cosine similarity to compare a detection with previously saved features.

To create the YOLO and DeepSort algorithm, we started by using The AI Guy's GitHub repository [7] for a vanilla YOLO and DeepSort implementation. A vanilla implementation works great off the shelf for multi-object tracking on e.g., surveillance cameras and sports analysis. However, the settings of the Loomo race are quite different from such situations. Firstly, the model should only return the bounding boxes of a single person to the Loomo robot. Since DeepSort saves the IDs of every tracked person, we save the ID of our person of interest and only pass the corresponding bounding box to the robot. Secondly, the model needs a way to identify the person of interest. Having tried initializing on both a specific pose and the person closest to the center of the frame, we found the latter to be more robust.

We run the tracker on a NVIDIA Tesla V100 GPU. The Loomo robot sends a real-time camera feed to the V100 which, in turn, runs inference on the frame and returning the center of the object's bounding box and its confidence. As the communication between the robot and the GPU happens over WIFI, the images are resized to 160x120 to avoid bottlenecks.

We spent time with the robot to tune the hyperparameters of the model. For example, we tuned *n_init* and set it to 10 (frames). This ensured that initialization was fast, which could give us an advantage during the race. Moreover, an essential parameter is the maximum cosine distance. This parameter defines how different a person's vector embedding and the saved track can be and still be assumed to be the same. A high value would make it easier to track the wrong person, while a too low value would lead to it losing the person of interest too often. During our experiments with the robot, we discovered that the robot sometimes matched the wrong tracker and person. Therefore, we decreased the maximum

cosine similarity to make the model more robust.

III. Results

During the Loomo race, we achieved an acceptable lap performance of 1 minute. We used the following hyperparameters during the race: max_age = 24, n_init = 16, max_cosine_distance = 0.25.

IV. Discussion

Our implementation of DeepSort achieved the goal of tracking a person of interest. By initializing on the person closest to the center, we obtained a robust and controllable way to follow our person of interest, as our position relative to the Loomo can be controlled during the race. In addition, we had several ideas we pursued:

A. Initialization

We first tried obtaining a robust initialization by initiating the tracker on a pose by passing the bounding box obtained from a pose estimation [8] and classification model to DeepSort. The classification model utilized the captured human joint coordinates from the pose estimator to calculate the angles between certain joints. Once the angles for our desired pose are detected the pose is classified. We then tried to track the detected person with features most similar to the features within the bounding box returned by the pose estimator. However, as the bounding boxes returned from the pose estimation and YOLOv4 detection were substantially different in form, the DeepSort algorithm often disregarded the initial person and assigned a new id. This made the approach susceptible to randomness. Furthermore, we tried using pose estimation [9] on the bounding box outputs from the DeepSort algorithm. When the same pose was detected for 20 consecutive frames, the tracker initialized on that person. The method worked, but the overall performance of the model was subpar and we disregarded the initialization method. Instead, we went for a more straightforward method: initializing on the person closest to the center. This method is simple but performs well for us as you can always stand in the center of the frame when initializing. In the race, one was initializing while standing beside two other people. Using initialization on the center person instead of the first person detected ensured that the robot did not start tracking our competitors.

B. Assignment and single-object tracker

To make the assignment more efficient, we tried removing new tracks which didn't match the track of interest. Instead of using The Hungarian algorithm with time complexity O(N^3), we could do O(N) comparisons. However, this worked poorly, as new tracks were only compared to the track of interest. Due to the underlying assignment algorithm, new tracks were often assigned to our track of interest, resulting in the tracker switching between objects. This does not happen when not deleting tracks, as the tracks are more similar to the other saved tracks than the tracker of interest. Rewriting the source code would be time-consuming, and we stopped pursuing the idea when we realized that the inference speed of the assignment algorithm wasn't the bottleneck in our model.

We also tried using a single-object tracker, SiamMask, as we only are supposed to track one person. The rationale was that the inference speed would be much faster when not keeping track of many trackers simultaneously. However, we discovered that the tracker speed was not a problem. Additionally, the model performed severely weaker than DeepSort. We, therefore, rejected the method.

C. Proposed improvements

There are several elements that could have improved our model's performance and robustness. Firstly, we did not fine-tune the model with our own training data. Fine-tuning with images taken from Loomo in race-like lightning conditions would probably make the model more robust. Moreover, as the model does not need to generalize to all persons, we could specialize in tracking our group members. This way, we would mitigate the risk of the robot switching tracked person mid-race. Moreover, we could increase the complexity of our Re-ID encoder network. The network used is from 2017 and, due to computer vision moving at lightning speed, is subpar compared to today's SOTA networks.

When testing the Loomo in the race environment, we encountered some problems with the robot detecting objects in the ceiling which we believe was due to different lightning than in the training environment. In hindsight, we should have mitigated this by using data augmentations such that the model would be more invariant to changes in lighting. During the race, we managed to fix this, but experienced that the model performed worse than in the training environment. We believe this is due to us overfitting the model's hyperparameters in the training environment. We fine-tuned the model's performance before the race, but we only did this in one room with homogeneous lighting. Testing the model in different lighting conditions would probably give us more robust parameters. Another weakness is that we focused on creating a fast initialization and re-initalization algorithm. The rationale is getting a upper hand against teams that need ¿5 seconds to initialize while we initiate almost instantaneously. In the race, however, the time did not start before initialization was completed and we lost the edge in initialization performance.

V. Summary

We implemented the DeepSort tracking algorithm in combination with YOLOv4 as the object detector to participate in the Tandem Race at EPFL. We achieved decent performance thanks to a working tracker capable of accurately following a person. However, we neglected to test and train the model in differing environments, making the robot perform worse during the race than in the training phase. This highlights the importance of using data augmentation and testing in environments with varying conditions to achieve a robust tracker invariant to changes in lighting and other variables.

REFERENCES

[1] D. Reid, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[2] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE journal of Oceanic Engineering*, vol. 8, no. 3, pp. 173–184, 1983.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[4] N. Wojke, A. Bewley, and D. 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.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[6] *MARS: A Video Benchmark for Large-Scale Person Re-identification*. Springer, 2016.

[7] theAIGuysCode, "yolov4-deepsort," https://github.com/theAIGuysCode/yolov4-deepsort, 2020.

[8] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," *CoRR*, vol. abs/1906.08172, 2019. [Online]. Available: http://arxiv.org/abs/1906.08172

[9] Google, "Movenet: Ultra fast and accurate pose detection model," https://tfhub.dev/google/movenet/singlepose/lightning/4, 2021.