ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

SEMESTER PROJECT - VITA (10ECTS)

SPRING SEMESTER 2022

Robust Navigation with Loomo

Théo HERMANN
SUPERVISION: Y. LIU, A. ALAHI

Robust Navigation with Loomo. Design and Implementation of a Modular Perception Pipeline. Benchmark of Single Person Trackers.

Théo Hermann, Y.Liu, A.Alahi

I. Introduction

The use of **mobile robots** and its integration in our society is increasing. It is a motivation to develop intelligent mobile robot aware of their environment and able to adapt to it in **real-time**. We need to replace the perception abilities from the human brain such as detection of a human target in a scene, tracking this person and being able to differentiate between people. In our work, we propose an implementation of a modular and robust **perception pipeline**. We design our structure to be generic and flexible to changes in modules combinations. Our goal is to robustly detect and track a person of interest in order to follow him, this task is an instance of Single-Object-Tracking (**SOT**) problem, with the specificity to only focus on human detection and tracking. Once we have implemented the different methods in our pipeline, we run quantitative analysis to benchmark the performances on relevant datasets for the different proposed methods. We implement our perception module on a **Loomo** Segway Robot. After testing the robustness and performance of our method in real situations, the objective will be to combine it with an existing autonomous driving pipeline combining our perception module in a framework including state estimation, prediction, path planning and control.

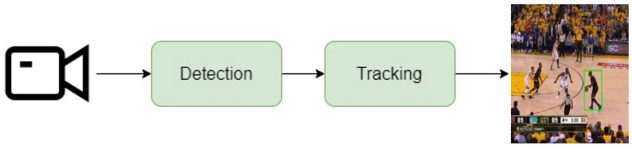

Fig. 1. Proposed Perception Pipeline. Taking image as input and output bounding box coordinates around the target to track.

II. Related Work

To expand our knowledge and have a solid basis for designing our module. We present the most useful bibliography and we classify it depending on the part it has contributed to our work. Our perception pipeline is composed of a detection part to get the bounding boxes around every person detected, then we need to track a unique person across time while being robust to disturbances and the subject going out of the frame. Our method is deployed on Loomo and we need a way to make our python code run on it. We also want to send data across the server running our algorithm and Loomo robot wirelessly. Finally, our perception module can be integrated inside a complete autonomous driving pipeline containing all the other modules to provide the autonomous driving capability to our robot.

A. Detection

We decide to use YOLOv5 (1) for the detection task. It is a family of object detection architectures and models pre-trained on the COCO dataset. Models can easily be loaded with pre-trained weights directly from the pytorch hub. Yolo allows for robust person detection, we restrict the output to the human class. To run in real-time we decided to choose a lightweight version of the Yolo model. We have to account for the trade-off between detection precision and runtime speed. In our case, we use **Yolov5m** as it has shown robust performance while still providing good runtime performance. Yolo is the acronym for "You Only Look Once." It's an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself.

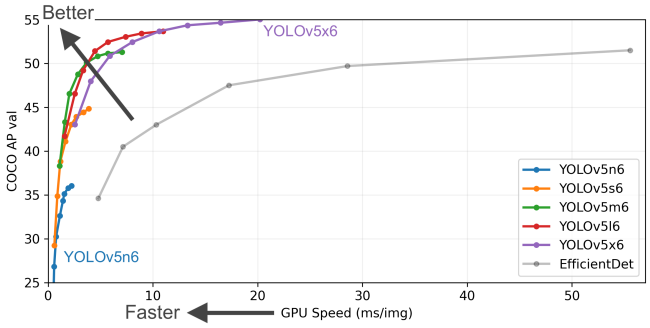

Fig. 2. Yolov5 architectures comparision Performance vs Runtime

We also looked into OpenPifpaf (2) for gesture recognition. It can be used to initialize the tracking based on a specific pose of the target. It can also be used to

get the position of the links of the person to track in real time.

B. Tracking

Our first approach was to only run a re-identification (**ReID**) algorithm on every new frame and compare the embedding generated for each detection to a reference embedding generated earlier. Then we used Deepsort as a tracker for our task. We can find a wide range of implementations available. We had to modify it as it is originally a Multi-Object Tracker (**MOT**) while in our case we are only interested in tracking a single person (SOT). After that, we tried state-of-the-art tracker available from MMTracking (3) model zoo which are SiameseRPN++ (4) and STARK (5). Siamese RPN ++ is a siamese deep neural network using cross-correlation between the convolution on the original image and the query. STARK is a Spatio-temporal Learning Visual transformer.

C. Re-Identification

Re-Identification algorithms can recognize the same person across a non-overlapping field of view and the learned representations allow them to be robust to disturbances such as illumination changes, occlusion, and blur due to camera motion... The first model we used has been trained using the method developed by G.Adaimi at EPFL VITA lab (6). It is based on a ResNet(50) backbone and uses the confidence loss penalty to model confidence in the representation learning framework. Concerning the State of the Art ReID model we used the FastReID repository (7), it provides code to train and evaluate ReID models on custom datasets but also a model zoo where some of the best performing ReID methods are exposed.

D. Connection with Loomo

First, to be able to implement our code on Loomo we need to build an Android App making a bridge between our python module and the implementation on Loomo. Indeed, the official Loomo SDK uses JAVA and is focusing on Android Developer but thanks to the developing kit (8) it enables the development of native C++ applications on Loomo. To build the **Follow** app, we needed to use Android Studio and update the SDK, NDK, and Gradle versions to resolve dependency conflicts. Replacing a few deprecated command calls to finally be able to build the app and then install it on Loomo. Now the latest version of the compiled APK is available on our Perception-Pipeline repository (9). When the **Follow** app is correctly installed on Loomo we can try to connect to the robot over WiFi using Android Debug Bridge (adb). Then using the socket protocol to transfer data we can receive the camera information in real-time on our workstation to process it and then we return the desired bounding box coordinates. The android **Follow** app already implements the control for the robot wheels and aims at reaching the bounding box coordinates. We now have a good framework to develop our python robust perception module and be able to test it directly on Loomo.

E. Pipeline for Mobile Robots

Designing a modular pipeline for any mobile robot can help generalize our work so that it could be deployed on a plurality of robots, while any specific module could be updated from the original pipeline without affecting the rest of the proposed framework. Indeed, modularity in the pipeline design is very helpful as methods for each different pillar of autonomous driving are evolving fast. We might want to replace or compare with previous methods. The most relevant work for this modular framework is the pipeline developed at EPFL VITA lab for a previous Master Project (10). This pipeline for autonomous driving robots inspired by Formula student teams, proposes a real-time, flexible and robust approach being easy to modify and implement on various types of robots. It relies on five different pillars: Perception, Estimation, Prediction, Path Planning and Control. Our goal is to provide a modular pipeline for the perception pillar so that it could be integrated into this more general autonomous driving pipeline.

F. Datasets

For our task, we want to evaluate the performance of our proposed perception methods on data containing a human target which we aim at robustly tracking in diverse environments. There are many datasets for Multi-Object Tracking publicly available (11) (12) (13) (14), problem is that it contains multiple targets for the tracking, meaning that the ground truth data need to be processed in our case as we only want to track a unique person and also it sometimes focus on objects other than persons (animals, cars...). That being said, in our case the more relevant datasets are the one focusing on Single Object Tracking (SOT) (15) (16) (17) (18). However, even in those datasets only a fraction of the data contain humans as most of the data focus on a different object. The most relevant Datasets for our work are LaSOT (16) and OTB100 (18).

a) LaSOT: It is a large-scale SOT dataset aiming to provide as much data as possible with high-quality labeling for long-term tracking. It provides more than 1500 sequences with a total above 3M frames, 85 categories of objects to track, and an average video length of about 2500 frames. In our case, we used the **person** category from the dataset, composed of twenty long length and high-quality sequences. The target of the tracking is a single person in complex and diverse environments: crowded scene, full-occlusion, body deformation, blur due to fast camera motion...

b) OTB100: This dataset provides more than 100 very diverse sequences. Each sequence provides annotated ground truth for the target to track, However it sometimes focuses on the face of the person or the object is another type (animals, cars...). This dataset proposes tagged attributes for each test sequence representing the challenging aspects of visual tracking (illumination variation, occlusion, body deformation, fast camera motion...). As said earlier the sequences provided in this dataset are very diverse: goes from basketball game videos to street cameras...)

G. Evaluation Metrics

To be able to assess the performance of our algorithm we must be able to get quantitative data about its performance. The most relevant evaluation metrics in our case use Intersection over Union (IoU). We define a correct detection by setting a threshold for the IoU score, based on this we can discriminate between true and false-positive detections. From those IoU scores we are able to compute two important metrics: **Precision** (P) and **Recall** (R). Precision measure the ability of the model to find the relevant target by computing the relevant detections (true positive) on all detections, while Recall measures the ability of the model to not mix target in detection and gives an indication of missed positive detections by computing the number of correct detections over the total ground truth available. Additionally, it is possible to plot the Precision vs Recall Curve by computing those values when increasing the IoU threshold. From there, other metrics such as: the Area Under Curve (AUC) or Average Precision (AP) over interpolated points for various recall values can be computed.

III. Method Implemented

A. Custom ReID Tracker

First we developed a custom tracking method based on Re-Identification. This approach uses a Re-Identification model (19) trained on market1501 dataset (20) using confidence penalty loss(6) instead of the standard cross-entropy to model confidence in the representation learning framework. This trained model is then loaded on a ResNet50 backbone and is able to generate embeddings from cropped detection image. To robustly track the person of interest, we then compare the generated embedding with the reference one which is kept, this method can be used with any other model by just replacing the embedding generation part. while trying to increase the robustness we tried to fully parametrize the method to allow for memorizing multiple reference embeddings and then averaging on them using increasing weights to put more influence on the most recent detections. Concerning the distance metric we used cosine similarity as it provided better results than L2 distance and was easier to tune. We explain in more details how the algorithm works.

1) Run Yolo Detection returns bounding box around every person
2) Crop the image bbox, preprocess image and embedding generation using ReID algorithm
3) Compare the distance between every embedding to the reference
4) Add the closest embedding to the reference and returns the correct bbox

We tried to parametrize as much as possible every aspect of this algorithm so that we could review the advantages brought by every combination of parameters

1) Yolo Model: Different size of yolo models results in the latency/precision tradeoff, we also have a parameter for the detection threshold (below this value detection is not considered valid)
2) Distance metric: We implemented the use of L2 distance as well as cosine similarity, we found that the second was easier to tune and more representative.
3) number of reference to keep: from 0 to 20
4) Averaging weights: constant, linear, and exponential.
5) Reference buffer access method: FIFO/smart replacement trying to keep high-diversity embedding.

This method was our first implementation and we did develop it from scratch. It allowed us to gain more knowledge about Tracking with re-identification and brought more insights into the challenges specific to this task. Advantages of this method are the fully-parametrized framework and the robustness to occlusions and other disturbances offered by the Re-ID. However, it should be noted that this method has not been optimized for runtime speed and it needs a proper tuning which might lead to overfitting, this results in not being able to compete with state-of-the-art methods.

B. Deepsort

For this, we tried to use a state-of-the-art Tracker publicly available with an easy-to-use implementation provided on Github. Deepsort (21) uses a Kalman filter in image space and solves the association problem using the Hungarian method with an association metric that measures the bounding box overlap. For our implementation, we started with a method combining Yolov5 for Detection and DeepSort for Tracking (22)(recently updated to STRONGSort) with an Omni-Scale Network as a Feature extractor. This approach has the advantage of combining both motion and appearance information. This method can be used to track every object that the Yolo detector has been trained on and as it is a Multi-Object Tracking (MOT) it can track multiple targets at the same time. Furthermore, this method is more efficient in terms of runtime than the previous one and results in better performance at high frame rates. Concerning the disadvantages of this method, we found during our test that it frequently resulted in ID-switches when the target was going out of the frame.

Meaning that the same original person was assigned a new ID, we also faced some problems when testing with occlusions.

C. Deepsort + ReID

Since we faced robustness problems in the observed behavior of the already existing DeepSort framework during our tests including occlusions, we developed a method trying to combine the robustness of our ReID method with the DeepSort framework which uses the motion information applying a Kalman Filter. To be able to include the ReID model in the existing framework, we fuse the outputs of both methods in the following way: We require a high similarity score for the matching algorithm of DeepSort to ensure that it only predicts if the association is certain. If not, we reinitialize with the ReID method as soon as DeepSort loses an ID.

D. Siamese RPN ++

Siamese network-based trackers formulate tracking as convolutional feature cross-correlation between a target template and a search region. This implementation (4) solves the previous problem of using siamese with deeper networks such as ResNet50, it shows that it comes from the lack of strict translation invariance. Moreover, this new model architecture performs layer-wise and depth-wise aggregations, which not only further improves the accuracy but also reduces the model size. For our implementation, we used the pre-trained model on LaSOT and the configuration file available in the ModelZOO (23). The results of this model on the LaSOT dataset are an Average Precision of 49.7 and an Inference time of 50 fps.[1]

E. Spatio-Temporal Transformer

This new tracking architecture named STARK (5) uses an encoder-decoder transformer as the key component. The encoder models the global Spatio-temporal feature dependencies between target objects and search regions, while the decoder learns a query embedding to predict the spatial positions of the target objects This proposed tracker achieves state-of-the-art performance on five challenging short-term and long-term benchmarks while running at real-time speed. For our implementation, we used the pre-trained model on LaSOT and the configuration file available in the ModelZOO (24). The results of this model on the LaSOT dataset are an Average Precision of 73 while the inference time is not communicated but from the paper, this method is supposed to run in real-time and produce increased inference speed.

[1]Results provided on the Github repository for the ModelZoo.

IV. Experiments

A. Dataset

To evaluate the performances of our implemented methods, we used Single Object Tracking Video Datasets publicly available, restricting to use of only sequences focusing on a person. As said previously, we used LaSOT (16), OTB100 (18) but also our custom Loomo Dataset.
To build this dataset we took recordings of experiments across the EPFL campus, the dataset is composed of 24 videos in total. This dataset shows the challenges we face for our perception algorithm: angle of the camera, different resolutions of the camera, real situations and crowded environments, and low framerate restricted by the communication of the image using the socket protocol.

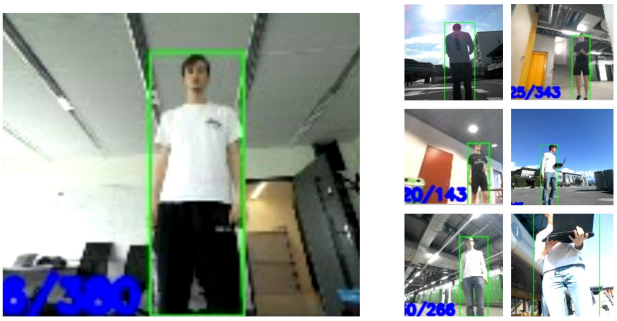

Fig. 3. Examples from Loomo Dataset with provided ground truth bounding boxes

B. Benchmarking

To interpret the results of the inference on the benchmark datasets, we need to have a clearly defined method to evaluate the performance of our methods. For this we first compute IoU scores for detection on each frame, after that, we can set the IoU threshold to interpret a detection as true positive, and from there we can finally compute precision and recall scores.

C. Results

This table resumes the results of the different methods implemented in our modular perception pipeline, first on the LaSOT dataset (person category) and then on our custom Loomo Dataset.

The configuration we used for those tests are:

1) Stark and Siamese: default configuration provided on mmtracking adding a threshold of 0.01 for the confidence score in the prediction.
2) Customreid: cosine similarity distance metric with a threshold of 0.88, keeping 15 references and using linear weights for the averaging.
3) Deepsort: max dist=0.1, max IoU distance=0.6, max age=30, n init=20 and nn budget=500.

Results on LaSOT Dataset			
Method	IoU > 0.5	IoU > 0.75	Time per frame [ms]
CustomReID	P=0.77/R=0.47	P=0.70/R=0.42	254.78
DeepSort	P=0.62/R=0.61	P=0.57/R=0.57	187.12
SiamRPN++	P=0.65/R=0.64	P=0.26/R=0.25	**36.2**
Stark	**P=0.94/R=0.93**	**P=0.81/R=0.81**	53.18

Results on the person category from LaSOT as the other are not relevant for our task

Results on Loomo Dataset			
Method	IoU > 0.5	IoU > 0.75	Time per frame [ms]
CustomReID	P=0.96/R=0.49	P=0.72/R=0.4	107.14
DeepSort	P=0.86/R=0.70	P=0.64/R=0.55	47.03
SiamRPN++	P=0.80/R=0.66	P=0.36/R=0.30	**30.04**
Stark	**P=0.96/R=0.77**	**P=0.73/R=0.60**	35.3

Results on our custom Loomo Dataset composed of more than 20 videos

Stark provides the best performances both in terms of precision and recalls while having a low inference time making it the more suitable method for our task. On the other hand, our custom ReID method shows good robustness by having a competing precision score on Loomo Dataset. However it should be noted that we did extensively tune this method on Loomo Videos which can have resulted in overfitting, also the inference speed makes it not suitable for real-time applications such as our task. SiamRPN++ provides the best inference time with good results. Deepsort shows decent performances and can be used as a good starting point. However, state-of-the-art SOT trackers from mmtracking have shown better performances while being as easy to implement.

V. Conclusion and Future Work

We have presented the design and implementation of a modular SOT Perception Pipeline. This pipeline allows easy module switching for each specific part. Concerning the future steps that could help to improve the performance:

- Improving Loomo Dataset: We started to record our Loomo Dataset but it would be useful to get more videos in different settings. Furthermore, we could use Video Augmentation techniques (25) to generate more Data by producing multiple slightly altered versions of the input.
- Fine-Tuning of the Tracker: to increase the performance of the Tracker we could leverage transfer learning techniques by fine-tuning a pre-trained model on a large dataset such as LaSOT on our Augmented Loomo Dataset which is smaller but much more representative of the task.
- Merging with ADP: Now that we tested and deployed our Perception-Pipeline we want to integrate it as the perception module of this autonomous driving pipeline (10)
- Application in Neuroscience research: When we manage to merge both pipelines and test their robustness, we can continue the research project to apply this technology in the field of neuroscience research.

We want to thank the Visual Intelligence for Transportation (VITA) Laboratory which supported the research project. We could acquire knowledge and material to implement our method, special acknowledgment to Y. Liu and A. Alahi for supervising this project.

REFERENCES

[1] G. Jocher, "ultralytics/yolov5," https://github.com/ultralytics/yolov5, 2021.

[2] S. Kreiss, L. Bertoni, and A. Alahi, "OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, March 2021.

[3] Open-MMLab, "Mmtracking," https://github.com/open-mmlab/mmtracking.

[4] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.

[5] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning spatio-temporal transformer for visual tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 448–10 457.

[6] G. Adaimi, S. Kreiss, and A. Alahi, "Deep visual re-identification with confidence," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103067, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21000929

[7] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, "Fastreid: A pytorch toolbox for general instance re-identification," *arXiv preprint arXiv:2006.02631*, 2020.

[8] Y. Liu and Segway, "Loomo-algodev: A native c++ development kit for loomo," https://github.com/segway-robotics/loomo-algodev, 2018.

[9] T. Hermann, "Perception-pipeline," https://github.com/theoh-io/Perception-Pipeline, 2022.

[10] C. Conejo and VITA-Lab, "Autonomous driving pipeline," https://github.com/cconejob/Autonomous_driving_pipeline/blob/main/paper.pdf, 2020.

[11] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," *arXiv:1504.01942 [cs]*, Apr. 2015, arXiv: 1504.01942. [Online]. Available: http://arxiv.org/abs/1504.01942

[12] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv:1603.00831 [cs]*, Mar. 2016, arXiv: 1603.00831. [Online]. Available: http://arxiv.org/abs/1603.00831

[13] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv:2003.09003[cs]*, Mar. 2020, arXiv: 2003.09003. [Online]. Available: http://arxiv.org/abs/1906.04567

[14] A. Dave, T. Khurana, P. Tokmakov, C. Schmid, and D. Ramanan, "Tao: A large-scale benchmark for tracking any object," in *European Conference on Computer Vision*, 2020. [Online]. Available: https://arxiv.org/abs/2005.10356

[15] K. H. Lianghua Huang, Xin Zhao, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," *arXiv preprint arXiv:1810.11981*, 2018.

[16] F. Y. P. C. G. D. S. Y. H. B. Y. X. C. L. H. L. Heng Fan, Liting Lin, "Lasot: A high-quality benchmark for large-scale single object tracking," *arXiv preprint arXiv:1809.07845*, 2018.

[17] M. Mueller, N. Smith, and B. Ghanem, *A Benchmark and Simulator for UAV Tracking*. Cham: Springer International Publishing, 2016, pp. 445–461. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_27

[18] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[19] K. Zhou and T. Xiang, "Torchreid: A library for deep learning person re-identification in pytorch," *arXiv preprint arXiv:1910.10093*, 2019.

[20] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Computer Vision, IEEE International Conference on*, 2015.

[21] 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.

[22] M. Broström, "Real-time multi-camera multi-object tracker using yolov5 and deepsort with osnet," https://github.com/mikel-brostrom/Yolov5_DeepSort_OSNet, 2022.

[23] MMTracking, "Siameserpn++ model zoo," https://github.com/open-mmlab/mmtracking/tree/master/configs/sot/siamese_rpn, 2019.

[24] ——, "Stark model zoo," https://github.com/open-mmlab/mmtracking/tree/master/configs/sot/stark, 2020.

[25] O. Köpüklü, "Video augmentation techniques," https://github.com/okankop/vidaug, 2021.