Review of current research into real-time multi-object detection, tracking and re-identification.

E.W.J. Bangma, B. van Beusekom, B.A.C. Dekker, R. Fidder,
M. van der Hart, T.A.W. van Kemenade, J.T.S. Kwa, M.D. van der Plaat,
G.J. van Schie, J. Teeuwissen, L. Xu

Software Project Computer Science,
Utrecht University
Department of Computer Science

Key words: computer vision; multi-object detection; multi-object tracking; object re-identification; real-time object detection; real-time object tracking; real-time object re-identification.

How to cite this article:
E.W.J. Bangma, B. van Beusekom, B.A.C. Dekker, R. Fidder, M. van der Hart,
T.A.W. van Kemenade, J.T.S. Kwa, M.D. van der Plaat, van Schie G.J., J. Teeuwissen, L. Xu
Review of current research into real-time multi-object detection, tracking and re-identification..
Department of Computer Science, Utrecht University
https://github.com/UU-tracktech/tracktech

Abstract

This paper contains a brief overview of open-source research repositories for real-time object detection, tracking and re-identification. The paper was written in relation to tracktech [1], an open-source repository aimed at creating a scalable system that can be connected to cameras and used for the real-time following of objects across multiple cameras. The team decided to implement YOLOv5 [2] and YOLOR [3] for real-time multi-object detection, SORT [4] and SortOH [5] for real-time multi-object tracking, and finally the Torchreid [6] and Fastreid [7] toolkits for real-time object re-identification.

Contents

1. Introduction

One of the most research-intensive and important parts of the project was the decision for detection, tracking and re-identification algorithms and research into whether suitable algorithms for the use-case even existed at all. This document aims to give an abbreviated overview of the research that went into these choices. The team decided to include this section because these algorithms were essential for the result. Furthermore, the research might be interesting for researchers and developers who wish to know the current state of research and algorithms in the field and possibly build upon the end product. The document will only be concerned with research connected to an open-source repository, as the algorithms had to be implemented in the final product.

The team will briefly define each of the stages, give an overview of the current state of research, especially the current challenges and shortcomings, and point out different benchmarks and sources for comparing different algorithms and argumentation for the chosen algorithms. Note that this choice of algorithm does not solely depend on performance (speed and accuracy) but also on ease of use, documentation, and ease of integration.

2. Detection

Detection is the first stage in the three-stage processing pipeline that the team implemented. In a way, this makes it the most important; the accuracy of the later stages depends significantly on the detection accuracy. When an object is not detected, it can not be tracked or re-identified either.

The team defines object detection as *the process of identifying the presence of an object of a certain class on a single image by defining a bounding box that completely but conservatively contains the object*. This definition splits into three parts. Firstly, detection deals with the identification of objects of a certain class. For example, these objects might be people, but other types of objects as well. Secondly, detection is a process that takes place over a single, still image. In our case, these images are the frames of the video. Therefore, detection is an independent process for each frame. Thirdly, detection defines bounding boxes around the identified objects. These boxes should contain the object in its entirety (*completely*) but not contain anything other than the object (*conservatively*).

Object detection also involves classification. The class of the detected objects should be known, such that the operator can filter on different objects. Detection of people is the most important in the current use case. This is also the class of object that is most present in different data sets. Vehicle detection might play a role, but the product can be trained on vehicle data sets if necessary.

The current use case also specifies multi-object detection. That is, the algorithm should detect any object that exceeds a certain threshold in the given image. This way, these objects can be tracked and re-identified later on.

2.1 Current Research

It was important to look at the current state of research into multi-object detection to make a well-thought-through decision. The most promising research into object detection uses deep learning and other artificial intelligence approaches to solve the object detection problem, specifically multi-object detection. The models trained on datasets of specific object types (single type or mixed) to achieve accurate results for unseen data. The paper describes types of object detection in the section.

It is important to distinguish between non-real-time and real-time algorithms when looking at the current state of research. In fact, they are listed as separate categories [8] [9] on the popular Machine Learning ranking platform Papers with Code [10]. Obviously, non-real-time object detection algorithm will have higher accuracy than real-time algorithms. For example, on the COCO dataset [11], The best non-real-time model, Swin-L [12], Achieves a box AP (for more information on accuracy measures and data sets, refer to the accuracy chapter) of 58.7% (June 9th, 2021).

However, real-time performance was an important constraint for the project. In principle, it might be possible to transform a non-real-time image algorithm into a video algorithm by processing each frame as an image. Still, the

performance would be too low for practical use cases. Therefore, the team focused the most research on real-time models. As a result, the real-time models are ranked not just on the accuracy but also on average frames per second (FPS).

Currently, the top real-time model on the list (as of June 9th, 2021) is the YOLOR-D6 [13] model. In the benchmark, this model achieved a MAP of 55.6% and an average FPS of 30. It is important to note that the speed of the algorithm obviously differs per system. Most models on the list are linked to the YOLO principle. These are all derived from the original YOLO-algorithm [14], which defined a certain model architecture. Notable improvements on the model are, for example, YOLOR-D6 [13] and YOLOv5 [2]. Another object detection algorithm that has been built upon and extended a lot is the R-CNN [15] algorithm, which defines another model structure. Notable improvements are, for example, Mask R-CNN [16] and Fast R-CNN [17]. This section will not describe the underlying principles of YOLO and RCNN, as these are too technical to explain here. Rather, consult the original papers for more information.

These two model types both aim for real-time object detection. Improved models often use the original idea for model structure but try to improve speed and/or the accuracy or generate more specific detection outputs (like the outline of a person rather than a bounding box). There are some independent models as well, but these two are the most prevalent.

2.2 Decisions

One might think that based on the benchmarks, it is easy to pick the top algorithm. However, the team had to think about more aspects for the specific use case.

First and foremost, the model should, apart from decent accuracy, achieve very high frames per second. The real-time performance might have been reached on a costly system but should also run with decent performance on systems with less available resources. On top of this, both a tracking and re-identification stage had to be built on top of the detection stage. This means that there are a lot more components that could improve performance. By focusing on detection FPS, the team could guarantee better speed for the end product. Secondly, the model needs to be documented well and should be easy to use. This way, future developers can extend upon the model as they wish, and the model can be implemented (and swapped out) easily for the team. Finally, it would also be preferable if the model supported some type of customization from the get-go. This includes the option to train weights on new data. This way, if better data sets for the use case are created, a developer or researcher can train the models themselves on this new data. It was also preferable if models came with multiple pre-trained weight sets that could be swapped out to be compared.

Keeping this in mind, the choice of the team landed on two algorithms: YOLOR-D6 and YOLOv5. YOLOv5 was mainly chosen for the solid repository documentation and very high detection speeds. It also came with several pre-trained weights of different sizes, where smaller weights had a lower accuracy but higher detection speed. YOLOR-D6 is slightly slower but does come with higher accuracy and is connected to a research paper that explains the model (YOLOv5 is one of the very few repositories without a paper backing it up). Some known issues with the YOLO-based algorithms are that they are relatively bad at detecting small objects (for example, those far away). The team recommends using YOLOv5 as the main detection algorithm since speed is essential for the use case.

The team decided not to use the RCNN-based models since these perform significantly slower than the YOLO-based models and are often too specific for the use case. The pipeline only needs bounding boxes and their classifications. Other information would make it too complex to combine the different stages.

3. Tracking

Tracking is the second stage in the three-stage processing pipeline. The team defines object tracking as *the process of following the movement of an object within a scene through associating a constant identifier for the object with the detected bounding boxes that contain the object*. This definition splits into three parts. First of all, the system should follow the movement of a specific object over the screen. Thus, if the object moves to another location, the algorithm

should know this new location. Secondly, tracking takes place within a single camera and is not shared over cameras. Thus, the pipeline tracks objects within the current scene and does not have to track them longer once they leave the frame. Finally, once an object moves to a new location and is detected there, the algorithm should associate this detection with the previous detection of the object and know that these are the same object. It outputs this knowledge by coupling both the bounding boxes to the same bounding box identifier. Thus, an equal bounding box identifier for two bounding boxes means that they contain the same object. Note, however, that if two bounding boxes do not share an identifier, this does not necessarily mean that they do not contain the same object: the algorithm may have lost track of the object and assigned a new bounding box identifier once the system could track it again.

The team decided that once again, the correct approach for the use case was multi-object tracking. All detected objects on the screen are tracked (object filtering occurs in the detection stage).

3.1 Current Research

The team chose a multi-object tracking approach since this approach is way more widely researched, especially in real-time environments. It is also easier to implement since any suspects that need to be followed are already being tracked by the tracking stage in theory. However, the team approached the problem using a multi-object tracking algorithm to track a tricky stage. Whereas it was possible to use learning models for object detection, it was not possible for tracking. This is since extracting features to track non-suspects is legally not allowed.

This fact immediately invalidated many of the current state-of-the-art algorithms since these were often multi-object tracking models that automatically tracked each object on the screen, with no possibility of tracking single objects, thus extracting illegal data for individuals. The team can find examples of such learning algorithms on the benchmarking platform Papers with Code [10], where multi-object tracking is a separate category [18]. High ranking learning algorithms (as of June 9th, 2021) include FairMOT [19], GSDT [20] and STGT [21], which all rank very highly on multiple benchmarked datasets. However, these algorithms are not optimized for speed, which is another reason not to use learning algorithms for the tracking stage.

Thus, for the current use case, the team searched for non-learning-based approaches to the problem of multi-object tracking. In practice, this means filter-based algorithms. These algorithms apply a filter over a frame that considers object velocity and direction and, with this information, predicts the location of new bounding boxes. These predicted bounding boxes are then compared to the actual location of new bounding boxes and are matched. The algorithm that stands out for the real-time use case is SORT [22] (simple online and real-time tracking). However, it has also been used as the basis for many other simple real-time tracking algorithms, such as SORT_OH, [23], which adds a simple occlusion-handling component and DeepSORT, [24], Adding a learning model for possible reidentification within a frame.

3.2 Decisions

Whereas the choice for a detection algorithm was quite broad, the options for tracking algorithms were more narrow down. In the end, the team decided to implement the SORT algorithm and its extension SORT_OH. SORT_OH is slightly slower and less well-documented than SORT, but it has a performance increase for occlusion.

Although the team chose them, both algorithms have quite some disadvantages. For example, objects are often not tracked for very long, and the ID of two different objects may get swapped during tracking. Furthermore, since there is no learning component, a subject cannot be re-identified once it has left the frame and comes back.

However, these disadvantages are outweighed by the benefits. Primarily, both SORT and SORT_OH are straightforward algorithms and impose very little overhead on the system. Their impact on the speed of the pipeline is negligible, and they provide a lot of reliable data for that speed. The algorithms are mainly used because tracking the object makes sure that the interface can highlight suspects as they move across the frame. Once the program loses track of an object, the re-identification stage can re-identify the object. The tracking stage can start from the newly

identified instance of the object. It is also important to note that SORT and its extensions are object-agnostic. Since the algorithm does not use a learning component but instead uses bounding box data, the algorithm can track any object type.

4. Re-identification

Re-identification is the third and final stage in the three-stage processing pipeline. The team defines object re-identification as *the process of determining whether a followed object appears in any scene based on their features through coupling a unique object identifier to any bounding box that contains the object*. This definition splits into three parts. First of all, the system should re-identify a followed subject on any camera. Therefore, the processor should share re-identification data between the camera processors. Secondly, this re-identification process works based on extracted data (features) about the objects. Finally, each followed object gets associated with a unique identifier. This identifier is then coupled to any boxes containing the object. The system can do this by coupling the object identifier to a list of bounding box identifiers: any bounding boxes with one of these bounding box identifiers contain the object. Note that, unlike with bounding box identifiers as mentioned in the tracking definition 3, when two boxes do not share an object identifier, it is certain that they do not contain the same object.

For the re-identification stage, only the followed objects are re-identified on the different cameras by sharing the object data over the different camera processors.

4.1 Current Research

Re-identification is the most complicated and most novel field of research of the three stages. Every research paper that the team found used some form of learning for the task, which is logical given the problem: some feature extraction must occur to allow for later object reidentification. Most current research focuses on comparing two images and determining whether they contain the same object, which is different from our use case.

Real-time and video re-identification research is especially sparse. Examples of video re-identification include STE-NVAN [25] and Jointly Attentive Spatial-Temporal Pooling Networks [26]. However, these papers did not perform in real-time and did not have repositories that the team could easily implement into the project.

Repositories, in general, posed a problem. Since the field is so novel, most repositories were mostly research-focused and not usable for integration into another project due to bad documentation, licenses or intensive system requirements. Therefore, the team focused on finding repositories with good documentation that backed by a well-performing model and research paper.

Another important thing to note is that almost all re-identification research has been trained using human data sets and that data sets for other object types are rare. Some models and weight sets, such as Vehiclenet [27] and A Strong Baseline For Vehicle Re-identification [28], Trained on vehicle re-identification data sets. However, research for human re-identification is definitely more prevalent. The person re-identification section on Papers with Code lists 258 papers and 32 data sets [29], Whereas the vehicle re-identification section listed 25 papers and 6 data sets [30].

In terms of person re-identification, some well-performing algorithms according to the Papers with Code ranking include (as of June 9th, 2021) the St-ReID [31], Adaptive L2-Regularization [32], the Bag of Tricks [33] model and many more.

4.2 Decisions

In conclusion, there are many re-identification models to choose from. However, many were unusable for the project. For example, their repositories were not well-implemented or well-documented, or they were unfit to work in any real-time environment. Some re-identification methods also required other inputs than the pipeline could provide.

The team felt that the decision to choose a re-identification model would be quite difficult. Furthermore, plug-and-play functionality would be even more difficult since it is hard to define an interface that would fit the in- and output of multiple re-identification models due to their different architectures. Finally, the research and product that the team should deliver is quite novel, and not many things have been done before. ML6 did something similar to a certain extent [34], But this project did not require the real-time component and did not constrain itself to non-learning tracking algorithms, thus having extracted features for each individual already.

The team did, however, eventually find a solution to numerous of these problems. FastReID [35] and Torchreid [36] are both open-source re-identification toolboxes implemented in Python. They implemented various re-identification models (such as the Bag of Tricks model mentioned before in Torchreid) that could easily be swapped out, inherently achieving the plug-and-play goal for the project. This also allowed the team to test out different re-id models and pre-trained weights for these models (these are also included in the repository) to find which configuration would lead to high accuracy and relatively high speeds. It also allows users to choose an implemented model in either of the two toolboxes they feel fits their use case well.

Another advantage was that these projects are well-documented and relatively easy to integrate into the pipeline. The team was able to write still an interface to which the re-identification methods should adhere, further enforcing the plug-and-play principle.

One disadvantage was that the models implemented in these libraries all work by comparing two images and determining whether they contained the same object, which was not perfectly in line with the use case. However, the team could still use this functionality by comparing the image of a followed subject to the contents of newly detected bounding boxes and determining whether their contents were the same. In this case, the bounding box identifier was coupled to the object identifier belonging to the followed object, as described before. This is still not ideal for the real-time constraint but does work very well for smaller settings.

In general, Torchreid seems to be slightly faster than FastReId, but less accurate, although this obviously depends largely on the used models. Eventually, the team landed on the choice for the Bag of Tricks model and its pre-trained weights in the Torchreid implementation. Also, it recommended this as the main configuration since it is quite fast and still accurate. For FastReId, the team chose SBS [37],, a stronger baseline model built on top of the Bag of Tricks model (it does not have a paper to support it).

5. References

[1] Tracktech. https://github.com/UU-tracktech/tracktech, 2021. Online; accessed 9-June-2021.

[2] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, wanghaoyang0106, Yann Defretin, Aditya Lohia, ml5ah, Ben Milanko, Benjamin Fineran, Daniel Khromov, Ding Yiwei, Doug, Durgesh, and Francisco Ingham. ultralytics/yolov5: v5.0, April 2021.

[3] YOLOR implementation - GitHub. https://github.com/WongKinYiu/yolor, 2021. Online; accessed 10-June-2021.

[4] SORT: A simple online and realtime tracking algorithm for 2D multiple object tracking in video sequences - GitHub. https://github.com/abewley/sort, 2020. Online; accessed 10-June-2021.

[5] SortOH - GitHub. https://github.com/mhnasseri/sort_oh, 2021. Online; accessed 9-June-2021.

[6] Torchreid: Deep learning person re-identification in PyTorch - GitHub. https://github.com/KaiyangZhou/deep-person-reid, 2021. Online; accessed 10-June-2021.

[7] fast-reid: SOTA Re-identification Methods and Toolbox - GitHub. https://github.com/JDAI-CV/fast-reid, 2021. Online; accessed 10-June-2021.

[8] Object Detection. https://paperswithcode.com/task/object-detection, 2021. Online; accessed 08-June-2021.

[9] Real-Time Object Detection. https://paperswithcode.com/task/real-time-object-detection, 2021. Online; accessed 08-June-2021.

[10] Papers with Code. https://paperswithcode.com, 2021. Online; accessed 08-June-2021.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, 2015.

[12] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *CoRR*, abs/2103.14030, 2021.

[13] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You Only Learn One Representation: Unified Network for Multiple Tasks. *CoRR*, abs/2105.04206, 2021.

[14] Ross Girshick Joseph Redmon, Santosh Divvala and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, 2016.

[15] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[17] Myung-Cheol Roh and Ju young Lee. Refining faster-RCNN for accurate object detection. *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, 91(4), 2017.

[18] Multi-Object Tracking. https://paperswithcode.com/task/multi-object-tracking, 2021. Online; accessed 08-June-2021.

[19] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A Simple Baseline for Multi-Object Tracking. *CoRR*, abs/2004.01888, 2020.

[20] Yongxin Wang, Xinshuo Weng, and Kris Kitani. Joint Detection and Multi-Object Tracking with Graph Neural Networks. *CoRR*, abs/2006.13164, 2020.

[21] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. TransMOT: Spatial-Temporal Graph Transformer for Multiple Object Tracking. *CoRR*, abs/2104.00194, 2021.

[22] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple Online and Realtime Tracking. *CoRR*, abs/1602.00763, 2016.

[23] Mohammad Hossein Nasseri, Hadi Moradi, Reshad Hosseini, and Mohammadreza Babaee. Simple online and real-time tracking with occlusion handling, 2021.

[24] DeepSORT. https://github.com/nwojke/deep_sort, 2028. Online; accessed 9-June-2021.

[25] Chih-Ting Liu, Chih-Wei Wu, Yu-Chiang Frank Wang, and Shao-Yi Chien. Spatially and Temporally Efficient Non-local Attention Network for Video-based Person Re-Identification. *CoRR*, abs/1908.01683, 2019.

[26] Shuangjie Xu, Yu Cheng, Kang Gu, Yang Yang, Shiyu Chang, and Pan Zhou. Jointly Attentive Spatial-Temporal Pooling Networks for Video-based Person Re-Identification. *CoRR*, abs/1708.02286, 2017.

[27] Zhedong Zheng, Tao Ruan, Yunchao Wei, Yi Yang, and Tao Mei. VehicleNet: Learning Robust Visual Representation for Vehicle Re-identification. *CoRR*, abs/2004.06305, 2020.

[28] Su V. Huynh, Nam H. Nguyen, Ngoc T. Nguyen, Vinh TQ. Nguyen, Chau Huynh, and Chuong Nguyen. A Strong Baseline for Vehicle Re-Identification. *CoRR*, abs/2104.10850, 2021.

[29] Person Re-identification. https://paperswithcode.com/task/person-re-identification, 2021. Online; accessed 08-June-2021.

[30] Vechile Re-identification. https://paperswithcode.com/task/vehicle-re-identification, 2021. Online; accessed 08-June-2021.

[31] Guangcong Wang, Jianhuang Lai, Peigen Huang, and Xiaohua Xie. Spatial-Temporal Person Re-identification, 2018.

[32] Xingyang Ni, Liang Fang, and Heikki Huttunen. AdaptiveReID: Adaptive L2 Regularization in Person Re-Identification. *CoRR*, abs/2007.07875, 2020.

[33] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of Tricks and A Strong Baseline for Deep Person Re-identification. *CoRR*, abs/1903.07071, 2019.

[34] Pedestrian tracking over multiple non-overlapping camera viewpoints. https://blog.ml6.eu/ml6-internship-pedestrian-tracking-over-multiple-non-overlapping-camera-viewpoints-5b405c6df7e0, 2028. Online; accessed 29-April-2021.

[35] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. FastReID: A Pytorch Toolbox for General Instance Re-identification. *CoRR*, abs/2006.02631, 2020.

[36] Kaiyang Zhou and Tao Xiang. Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch. *CoRR*, abs/1910.10093, 2019.

[37] SBS Model. https://github.com/JDAI-CV/fast-reid/blob/master/configs/Market1501/sbs_R101-ibn.yml. Online; accessed 11-June-2021.