Tracking On Video Using Single Object Trackers

1st Louis GUO
Student from EPITA
exchange semester at NTNU, Gjøvik, Norway
Paris, France
louis.guo@epita.fr
louisgu@ntnu.no

2nd Mohib Ullah
Department of Computer Science
NTNU
Gjøvik, Norway
mohib.ullah@ntnu.no

3rd Faouzi Alaya Cheikh
Department of Computer Science
NTNU
Gjøvik, Norway
faouzi.cheikh@ntnu.no

***Abstract*—Object tracking is a very challenging task for a computer although it is a very easy thing to do for humans. A lot of progress has been made these last decades about object detection and localisation and today's machine can almost be on par with a human observer. However for object tracking, it is still far from reaching that point. Because object tracking faces a lot of challenges such as the fact that the target disappears at some frames and reappear a dozen frames after, or the target can be partially hidden behind an obstacle. Changes in the environment, like the lighting of the capture video can also impact the results. All these issues make object tracking very challenging for computers. Current state of the art single object tracker struggle to differentiate between objects that look very similar to the actual target, it would often miss track a distractor object as the object itself. Distractors are objects that are similar to the object that is currently being tracked. In order to overcome this problem, Christoph Mayer, Martin Danelljan, Danda Pani Paudel, Luc Van Gool from the Computer vision lab, D-ITET, ETH Zurich, Switzerland decided to write the paper : *Learning Target Candidate Association to Keep Track of What Not To Track*. The OpenCV library has integrated seven different trackers that are available. We will compare Keeptrack and the seven trackers from OpenCV along with the Dimp tracker that is available in the pytracking git repository.**

***Index Terms*—Single Object Tracking, Video Tracking, KeepTrack, Dimp, OpenCV trackers**

I. Introduction

Object Tracking aims at tracking a target, the object could be a person, a car or something else; given a sequence of video, we will track the same object and know its localisation using bounding boxes throughout the whole video.

Object tracking is used in fields such as human surveillance, autonomous driving, medical imaging and traffic surveillance.

A naive question would be : why not using simply object detector to solve this tracking problem? For each frame of the video, we could apply object detection in order to locate and detect the object. Because sometimes there are occlusion (the object is hidden by an obstacle), change of illumination or change of pose it is not possible to detect those objects on the images. For these reasons, we use tracking to overcome these kinds of problems. Also, using object detection, we will not be able to distinguish between several instances of the same object throughout the video.

There are two types of object tracking : single object tracking [1] and multiple objects tracking [2].

Simple object tracking (SOT) tracks only one single object throughout a video. Although it's only one object that is being tracked, it does not make the task easier because the object could be anything not only part of widely known classes such as cars, human, animals etc. Single object trackers take as input a bounding box of the target we want to track on the first frame.

In the case of multiple object tracking [3] (MOT), the most approved approach is the **tracking by detection** [4] paradigm. It basically applies a detection algorithm on each frame of the source video and in order to link all the different objects in the video, it performs matching techniques between frames to keep track of the different targets. Nowadays, the multi object tracking problem became a matching problem [5].

II. Methodology

A. OpenCV Trackers

1) Boosting: The boosting [6] tracker relies on AdaBoost [7] algorithm, created 10 years ago. It uses the previous frame's bounding box in order to predict the current frame's bounding box by searching in the neighbourhood of the previous frame's position. Using a score map, it will choose the location that reaches the highest score as the current frame's location. It is going to be trained as there are more frames getting processed thus more positive examples.

2) MIL: The MIL [8] tracker (Multiple Instance Learning tracker) is similar to the boosting tracker because is also uses the neighbourhood and the location of the previous bounding box but takes into account small areas of the neighbourhood in order to train with the positive examples.

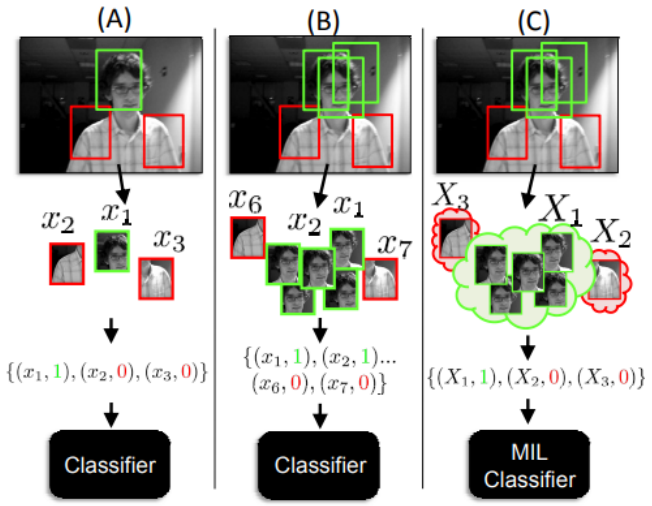

Fig. 1. Mil : (A) only one positive image is used (B) several positive images are used (C) several positive images paired with MIL

3) KCF: KCF [9] stands for Kernelized Correlation Filters, it is based on the MIL tracker, but optimised by mathematical properties in order to handle overlapping areas.

4) TLD: The TLD [10] tracker (Tracking, Learning, Detection) is divided into 3 different parts. In the tracking part, it will track the target throughout the frames. In the detection part, The detector learns the appearance of the target so it can perform detection. In the learning part, it will be trained on the detection part so that is does not make the same mistakes.

5) Median Flow: Median FLow [11] is a tracker that uses forward and backward position of the object in order to predict the future trajectories that the target is potentially taking Fig. 1. It assumes that the target's temporal order through the frames does not matter.

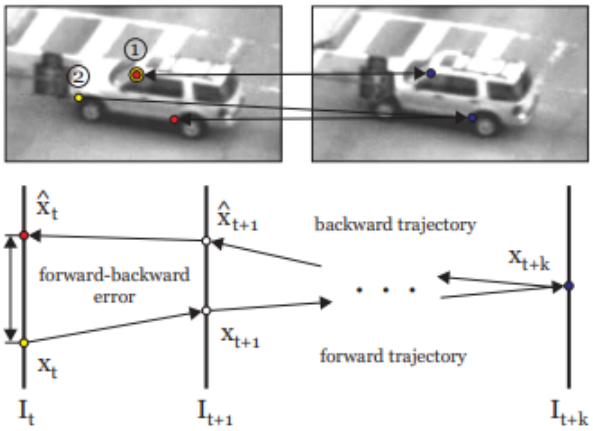

Fig. 2. Median Flow Tracker

6) Mosse: Mosse [12] stands for Minimum Output Sum of Squared Error. It uses adaptive correlation filters in order to track the target. This is the equation that describe the filter :

$$H = \frac{\sum_i F_i \odot G_i^*}{\sum_i F_i \odot F_i^*}$$

Where F is the Fourier Transform, G the groundtruth, $\odot$ for element wise multiplication and * for the complex conjugate.

7) CSRT: CSRT [13], also known as Discriminative Correlation Filter with Channel and Spatial Reliability (DCF-CSR), is a tracker that relies on training with correlation filters paired with Hog [14] and Colornames features. Then, it will search in the neighbourhood of the previous position of the target to find the new location of the target.

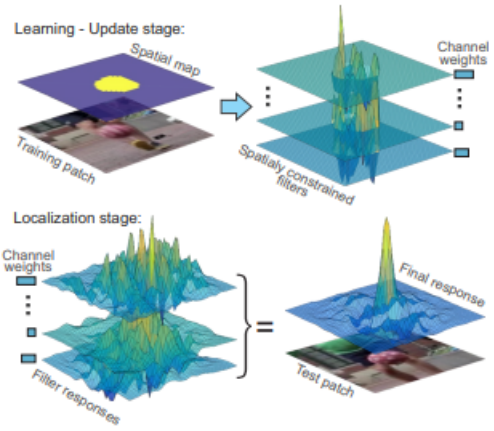

Fig. 3. CSRT : an overview of the method

B. KeepTrack

Keeptrack [15] was introduced by Christoph Mayer, Martin Danelljan, Danda Pani Paudel and Luc Van Gool from the Computer vision lab in Switzerland published in ICCV 2021.

The architecture of KeepTrack is divided into several parts. First, there is the Base tracker, in the paper they decided to use SuperDIMP as a base tracker which allow them to get the candidates that look similar to the target.

By extracting their positions, their resemblance to the target and also an appearance cue, they build a vector of feature that is going to describe each candidate.

After that, they use the previous frames' feature vectors and the current one to provide into the embedded network that will do matching between the last and current frame with SuperGlue [16]. This will give probabilities for the matching candidates in order to determine which object is the target on the current frame.

C. Dimp

The [17] Dimp tracker was published in 2019. Two of the four authors that are also mentioned in the KeepTrack tracker. In order to predict the target's position, it would use appearance information of the background as well as appearance information of the target.

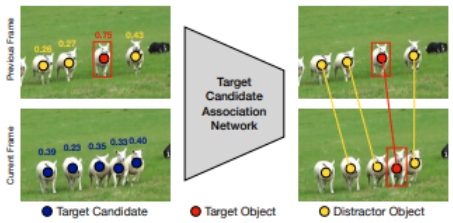

Fig. 4. Explication of the KeepTrack tracker : In blue are the target candidates. In red is the target. In yellow are the distractors.

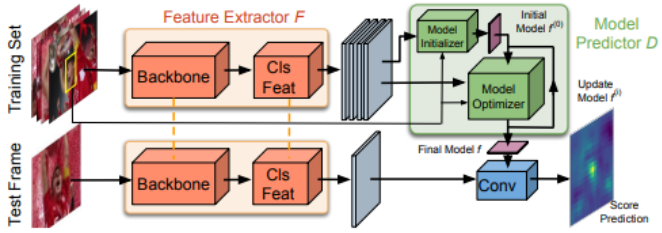

Fig. 5. Architecture of the Dimp tracker [18]

III. Experiments

A. Datasets

For the experiments, videos from the dataset GOT-10k [19] and the dataset Visual Tracker Benchmark [20] were used. The dataset GOT-10k possess more than 10 000 videos which bounding boxes are manually labelled. The Visual Tracker Benchmark has data with associated groundtruth and also benchmark results of different trackers along with the code.

B. Performance metrics

Object tracking is a very competitive topic, where researchers propose many methods in order to reach the state of the art in the domain. However, in order to measure the performance of a tracker, we need proper metrics. It is much more complex to rate the performance of an object tracker than a detector because there are many factors to take into account. The ability of the tracker to recover from occlusion or an error, how many times did the tracker lost the target, the size of the bounding box, how far the tracker drifted from the target.. These factors are hard to measure and to combine. In the paper *Visual Tracking: An Experimental Survey* [18], several metrics are mentioned.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

The metrics used in this paper are the IOU (Intersection Over Union) and the distance between the centre of the predicted bounding box and the centre of the groundtruth.

C. Quantitative Evaluation

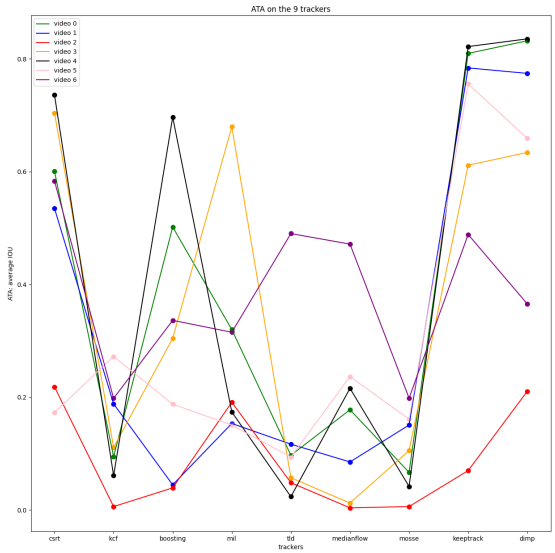

Fig. 7. Graph of ATA of the videos over the trackers

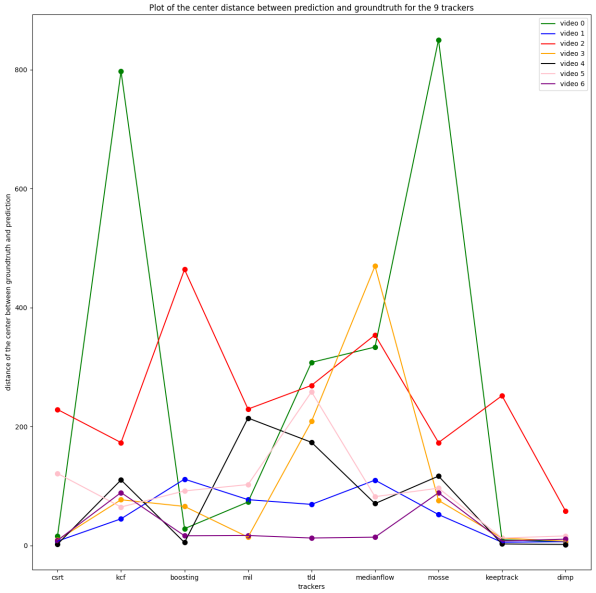

Fig. 8. Graph of centre distance of the videos over the trackers

As we can see on the Table I and Fig. 7, according to the metric of ATA (the mean average intersection over union), KeepTrack and Dimp perform better than the 7 trackers from

TABLE I
TABLE OF ATA OF THE TRACKERS OVER THE VIDEOS

video name	csrt	kcf	boosting	mil	tld	medianflow	mosse	keeptrack	dimp
0	0.600265	0.094722	0.501775	0.319953	0.097056	0.177751	0.066805	0.809470	0.831791
1	0.534560	0.188205	0.044699	0.153463	0.116721	0.085258	0.150708	0.783742	0.774023
2	0.218310	0.006161	0.039284	0.191123	0.048480	0.003888	0.006161	0.069920	0.210532
3	0.703570	0.111167	0.304375	0.679352	0.057243	0.012453	0.105341	0.611151	0.633959
4	0.735539	0.060980	0.696411	0.173931	0.023703	0.215694	0.041791	0.821551	0.835253
5	0.172848	0.272213	0.187599	0.150574	0.093857	0.236606	0.161179	0.755518	0.658759
6	0.583207	0.198029	0.336263	0.315155	0.490070	0.471421	0.198382	0.488448	0.365435

TABLE II
TABLE OF CENTRE DISTANCE OF THE TRACKERS OVER THE VIDEOS

video name	csrt	kcf	boosting	mil	tld	medianflow	mosse	keeptrack	dimp
0	16.285958	797.125860	28.432169	73.265004	307.867577	333.618888	850.095781	9.797371	6.738104
1	7.503575	44.963880	111.681498	77.173378	69.239519	110.007802	52.134744	5.603934	6.619004
2	228.721241	173.128560	464.056492	229.421996	269.278597	354.056408	173.128560	251.904415	58.276572
3	9.526874	77.035422	65.825294	14.231234	208.783326	469.869399	75.951135	12.613284	7.480112
4	2.388560	110.344163	5.155264	214.341271	173.520020	70.577834	116.789989	2.784909	1.900711
5	121.232537	64.851135	91.894698	102.495144	258.432496	82.226858	96.514973	12.712666	16.208998
6	8.723201	89.118729	16.467070	16.972837	12.729742	14.166967	88.727235	7.779259	10.843150

OpenCV because the higher the ATA is, the better performance the tracker has.

As on the Table II and Fig. 8, the less distance there is between the center of predicted bounding box and the center of the groundtruth bounding box, the better performing the tracker is. As a result, Keeptrack and Dimp from the official implementation [21] perform better than the other trackers from OpenCV.

Fig. 10. Inference on VisualTrackingBenchmark [20] dataset with a human walking

D. Qualitative Evaluation

Fig. 9. Inference on Got-10k dataset with a chicken [19]

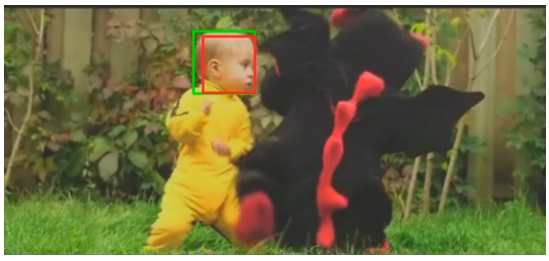

Fig. 11. Inference on VisualTrackingBenchmark [20] dataset with a fighting child

In green is the predicted bounding box of the Keeptrack tracker and in red is the groundtruth provided by the dataset. We can see that the predictions are very close to the groundtruth.

Fig. 12. Inference on VisualTrackingBenchmark [20] video with birds

However, sometimes the tracker Keeptrack still misses the target as a distractor as shown in Fig. 12.

IV. Conclusion

We have done a benchmark on OpenCV trackers with the Keeptrack tracker and Dimp. Using quantitative (ATA and Centroid distance) and qualitative measurements, we have seen that KeepTrack and Dimp achieve better results than the OpenCV trackers. However, when the tracker lose track of the target, the default behaviour is that the previous bounding box coordinate is duplicated on the current frame is case the tracker lose the target. It is because otherwise, the number of bounding box of the predicted tracker and the groundtruth won't match. An improvement would be to count the number of times each tracker loses track of the target and add it as a metric to measure the performance of the tracker.

References

[1] Luka Cehovin, Ales Leonardis, and Matej Kristan. "Visual object tracking performance measures revisited". In: *CoRR* abs/1502.05803 (2015). arXiv: 1502.05803. URL: http://arxiv.org/abs/1502.05803.

[2] Anton Milan et al. "MOT16: A Benchmark for Multi-Object Tracking". In: *CoRR* abs/1603.00831 (2016). arXiv: 1603.00831. URL: http://arxiv.org/abs/1603.00831.

[3] Gioele Ciaparrone et al. "Deep Learning in Video Multi-Object Tracking: A Survey". In: *CoRR* abs/1907.12740 (2019). arXiv: 1907.12740. URL: http://arxiv.org/abs/1907.12740.

[4] Niels Ole Salscheider. "Object Tracking by Detection with Visual and Motion Cues". In: *CoRR* abs/2101.07549 (2021). arXiv: 2101.07549. URL: https://arxiv.org/abs/2101.07549.

[5] Wenhan Luo et al. "Multiple Object Tracking: A Literature Review". In: *Artificial Intelligence* 293 (May 2017). DOI: 10.1016/j.artint.2020.103448.

[6] Helmut Grabner, Michael Grabner, and Horst Bischof. "Real-Time Tracking via On-line Boosting". In: vol. 1. Jan. 2006, pp. 47–56. DOI: 10.5244/C.20.6.

[7] Tu Chengsheng, Liu Huacheng, and Xu Bing. "AdaBoost typical Algorithm and its application research". In: *MATEC Web of Conferences* 139 (Jan. 2017), p. 00222. DOI: 10.1051/matecconf/201713900222.

[8] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. "Visual tracking with online Multiple Instance Learning". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 983–990. DOI: 10.1109/CVPR.2009.5206737.

[9] João F. Henriques et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *CoRR* abs/1404.7584 (2014). arXiv: 1404.7584. URL: http://arxiv.org/abs/1404.7584.

[10] Z. Kalal, J. Matas, and K. Mikolajczyk. "Tracking-Learning-Detection". In: *IEEE Transactions on Pattern Analysis Machine Intelligence* 34.07 (July 2012), pp. 1409–1422. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2011.239.

[11] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Forward-Backward Error: Automatic Detection of Tracking Failures". In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 2756–2759. DOI: 10.1109/ICPR.2010.675.

[12] David S. Bolme et al. "Visual object tracking using adaptive correlation filters". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2544–2550. DOI: 10.1109/CVPR.2010.5539960.

[13] Khurshedjon Farkhodov, Suk-Hwan Lee, and Ki-Ryong Kwon. "Object Tracking using CSRT Tracker and RCNN". In: Jan. 2020, pp. 209–212. DOI: 10.5220/0009183802090212.

[14] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[15] Christoph Mayer et al. "Learning Target Candidate Association to Keep Track of What Not to Track". In: *CoRR* abs/2103.16556 (2021). arXiv: 2103.16556. URL: https://arxiv.org/abs/2103.16556.

[16] Paul-Edouard Sarlin et al. "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: *CoRR* abs/1911.11763 (2019). arXiv: 1911.11763. URL: http://arxiv.org/abs/1911.11763.

[17] Goutam Bhat et al. "Learning Discriminative Model Prediction for Tracking". In: *CoRR* abs/1904.07220 (2019). arXiv: 1904.07220. URL: http://arxiv.org/abs/1904.07220.

[18] Arnold W. M. Smeulders et al. "Visual Tracking: An Experimental Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1442–1468. DOI: 10.1109/TPAMI.2013.230.

[19] Lianghua Huang, Xin Zhao, and Kaiqi Huang. "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild". In: *CoRR* abs/1810.11981

(2018). arXiv: 1810.11981. URL: http://arxiv.org/abs/1810.11981.

[20] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Online Object Tracking: A Benchmark". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013. URL: http : / / cvlab . hanyang . ac . kr / tracker_benchmark/index.html.

[21] Christoph Mayer Martin Danelljan Goutam Bhat. *pytracking*. https://github.com/visionml/pytracking. 2021.