Online and Real-Time Tracking in a Surveillance Scenario

Oliver Urbann[1,*], Oliver Bredtmann[2], Maximilian Otten[1], Jan-Philip Richter[1], Thilo Bauer[2], David Zibriczky[2]
[1]Fraunhofer IML, Dortmund, Germany [2]DB Schenker, Essen, Germany
[*]oliver.urbann@iml.fraunhofer.de

***Abstract*— This paper presents an approach for tracking in a surveillance scenario. Typical aspects for this scenario are a 24/7 operation with a static camera mounted above the height of a human with many objects or people. The Multiple Object Tracking Benchmark 20 (MOT20) reflects this scenario best. We can show that our approach is real-time capable on this benchmark and outperforms all other real-time capable approaches in HOTA, MOTA, and IDF1. We achieve this with two contributions. First, we apply a fast Siamese network reformulated for linear runtime (instead of quadratic) to generate fingerprints from detections. Second, we extend the walking path as predicted by the Kalman filter with an additional motion model that also takes into account unforeseen changes in the intention of the tracked person. Thus, it is possible to associate the detections to Kalman filters based on multiple tracking specific ratings: Cosine similarity of fingerprints and Intersection over Union (IoU).**

I. Introduction

Tracking is a broad research area with a long history and a wide area of application. This paper focuses on scenarios in a typical surveillance application: A 24/7 video stream where many objects or persons must be tracked at the same time. Here, cameras are usually mounted at a height that reduces occlusions and have fixed positions and angles. Due to the 24/7 operation, the tracking algorithm must run in real-time to avoid a growing buffer with unprocessed data. Typical applications are in warehouses optimizing material routing or fork lifter paths, passenger routing in airports to reduce queues, or crowd management in a sports stadium. The MOT20 dataset [4] reflects all these challenges best and is thus chosen for evaluation in this paper. It furthermore includes day and night scenes and provides a frame rate of 30 Hz giving an indicator for a real-time capable algorithm.

We intentionally do not consider datasets containing images captured by moving cameras (e.g. MOT17). This would require an additional time-consuming motion compensation that is not necessary in our targeted scenario.

A. Related Work

Evaluations of over 20 different approaches are available on MOT20. As depicted in Fig. 1, a significant gap divides two clusters of algorithms regarding the runtime given by the authors. These algorithms are evaluated on different systems and thus the definition of real-time can only be vague. Furthermore, the execution time also depends on the number of detections. Thus, within our development we focus on linear runtime with respect to the big O notation. For comparison with other approaches based on MOT20, we define real-time capability based on the gap in Fig. 1. One cluster can be seen below the gap with varying performance regarding High Order Tracking Accuracy (HOTA). We define the algorithms belonging to the other cluster above the gap as realtime capable, although not all are above 30 fps which is the frame rate of MOT20. Solutions belonging to this cluster rely on the detections given in MOT20. To remain fast, one cannot expand those algorithms by complex image processing. Faster RCNN [10] is used to provide detections in MOT20, but it reveals a weak performance in crowded test images. Thus, the performance of real-time capable approaches is rather low.

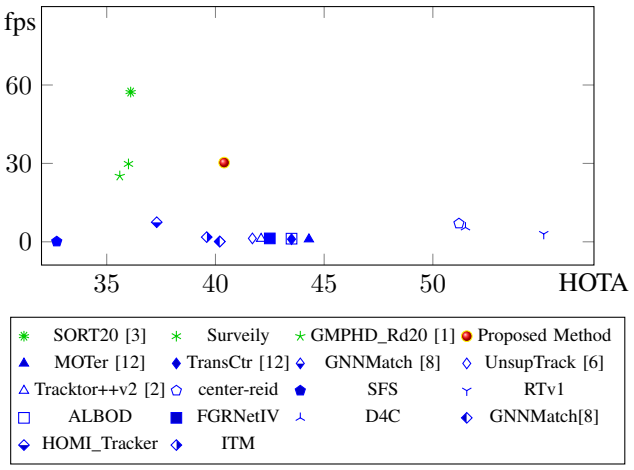

Fig. 1. Approaches solving the MOT20 benchmark with focus on runtime vs. Higher Order Tracking Accuracy (HOTA) [7]. A clear gap can be seen between real-time (green) and non-real-time approaches (blue). The red dot indicates the proposed approach.

This is even more obvious when the solutions are sorted by the MOTA metric. All real-time capable solutions perform below all non-real-time approaches, see Table I.

SORT [3] is an example of a simple but fast approach. It applies a Kalman filter for tracking that is updated with detection bounding boxes. The assignment is done by applying the Intersection over Union (IoU) distance to build a cost matrix solved by the Hungarian algorithm. However, as this approach ignores appearance features, it is fast but tracking performance is rather low (see Fig. 1).

Baisa [1] proposes to improve tracking performance by applying an identification network (IdNet) that extracts features

The research was supported by the German Federal Ministry of Education and Research and the State of North Rhine-Westphalia under the Lamarr Institute for Machine Learning and Artificial Intelligence, grant number LAMARR22B.

from detections. A GM-PHD filter first uses detections to output estimates which are then used for an estimate to track association. Two disadvantages are worth mentioning here: 1) Different and inconsistent distance metrics are applied throughout the pipeline and 2) IdNet is trained on single images instead of the (dis)similarity of two patches.

Using a CNN for similarity estimation is a common approach. Ding et al. [5] propose to build triplets for training a CNN that extracts feature representations from image patches. Siamese networks are widely used in single object tracking [9] and person re-identification [11].

B. Contribution

In this section we introduce the contribution of this work with a short introduction before.

1) LTSiam: The base of the proposed tracker is similar to SORT. I.e. we apply Kalman filter, one for each track, and update them utilizing detections. For our targeted scenario, this solution is sufficiently fast but lacks accuracy due to erroneous detections. We thus improve this approach by applying an additional feature extraction from image method. Siamese networks could help to improve the association of possibly erroneous detections to tracks. However, Siamese networks applied for tracking usually have a $O(N \cdot M) \approx O(N^2)$ runtime, where N is the number of tracks and M the number of detections. This is especially problematic in a 24/7 surveillance scenario.

Our first contribution is LTSiam, a CNN that

- is based on well-evaluated and well-performing Siamese networks,
- is trained with the same similarity measure used for inference,
- is specifically trained for multi object tracking application,
- can be partially applied with linear instead of quadratic complexity and
- can be applied in an online and real-time capable algorithm.

2) Human Motion Model: Kalman filters designed for tracking can predict the person's position in the next frame based on current motion. However, by design, only the current velocity and acceleration are considered. In addition, the estimated uncertainty is based only on the measurements. Thus, Kalman filters do not take into account possible changes in intentions, walking directions, etc., which can lead to poor tracking performance.

Our second contribution is to incorporate a motion model that estimates the uncertainty due to unpredictable changes in speed and direction. This uncertainty is then applied as an additional cost to the mappings of detections to tracks, where the cost is higher, if the distance is higher relative to the expectation. A concrete prediction of a position is thus not provided, since this could only be done on the basis of actual information, for which the Kalman filter is provided here. The estimation is therefore a complement to the Kalman filter, not a replacement or correction.

In the evaluation, we can show that this approach outperforms other real-time approaches in HOTA, MOTA, and IDF1 on the MOT20 dataset while maintaining real time.

II. Approach

In this paper, we assume that detections are given from an external source like a CNN detector. We thus exclude this step from our timing analysis as a second system could be utilized for obtaining detections in parallel.

A. Track to Detection Assignment

For each person tracked we apply a Kalman filter. This allows us to continue tracking even if a person is not detected for some frames. Thus, detections must be associated with Kalman filters. We do this by creating an $N \times M$ cost matrix C where a single value $c_{n,m}$ expresses a cost for assigning detection m to track n. Afterwards, we utilize the Hungarian algorithm to minimize the overall cost and to output a set of selected associations $A = \{(m_1, n_1), \dots\}$.

This is a multi-criteria optimization consisting of the Intersection over Union c^{IoU}, human motion model c^h (see Sec. II-C) and the cost c^f of the fingerprint similarity (see Sec. II-B):

$$c_{n,m} = c^{IoU}_{n,m} + \alpha \cdot c^h_{n,m} + \beta \cdot c^f_{n,m}, \tag{1}$$

where α and β are weights heuristically determined.

1) Appearance of new untracked persons: Let us assume a person enters the observed area with detection j. Two cases can occur: 1) The detection is not assigned to any existing track which can and should happen if $N < M$, 2) detection j is assigned to an existing track i. The second case can occur if another person i left the observed area at the same time. To handle this, if

$$c_{i,j} > \Lambda_c \tag{2}$$

we assume that this assignment is wrong, where Λ_c is a heuristically determined threshold. In this case the assignment (i, j) is removed from A. Afterwards, for all detections not in A new tracks are created.

2) Disappearance of tracked persons: In case a track is not contained in A (i.e. no detection is assigned to this track in this frame) there are three possible reasons: 1) the person finally left the observed area, 2) it is temporarily hidden and 3) it is a false negative detection. Cases 2) and 3) cannot be distinguished and are thus handled equally by continuing the track (without sensor updates). To handle case 1) the track is deleted if it did not get any updates for T frames.

B. LTSiam

Fig. 2 depicts the proposed LTSiam network. As the setup of the cost matrix (see Eq. 1) has necessarily a quadratic complexity we limit the required calculations for this to a minimum. Therefore, the comparison of the fingerprints F_A and F_B is realized by a simple cosine similarity. The result is -1 for diametrically opposed vectors, 0 for vectors oriented at $90°$ relative to each other, and 1 for same

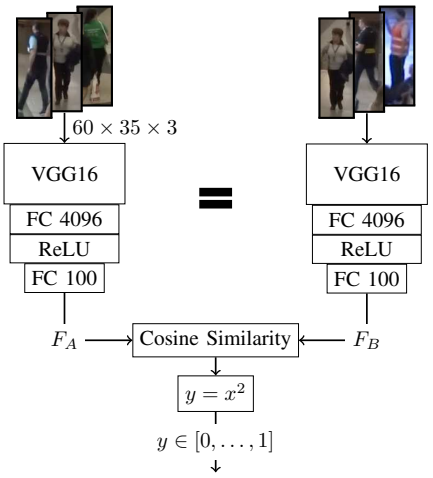

Fig. 2. Complete LTSiam model used for training. Input for a VGG backbone is a small image patch. A first fully connected layer gives a feature vector consisting of 4096 values. Another fully connected layer shortens this to 100 values to ensure a short runtime of the cosine similarity. The latter has a complexity of $O(n^2)$ during inference. After squaring the output 0 means "dissimilar" and 1 "similar".

orientation. However, interpreting -1 as dissimilar and 1 as similar patches (with 0 in between) does not lead to adequate training results. Thus, the similarity is squared, so both diametrically opposed and same oriented vectors are interpreted as similar patches. Given this network setup, the training leads to satisfying results and additionally opens up the possibility for inference with linear complexity.

To achieve this, we split off the backbone including fully connected layers. This can be utilized to infer the fingerprint at complexity $O(M)$. The resulting fingerprint is saved in the track to which the detection was assigned and can be reused in the next frame. The only remaining part with squared complexity is then the application of the fingerprints for determining the squared cosine similarity for Eq. 1:

$$c^f_{n,m} = 1 - \left[\frac{F_n \cdot F_m}{\|F_n\| \|F_m\|} \right]^2 \tag{3}$$

Note that the fingerprints are inferred for image patches derived from detections only. These patches thus do not depend on the tracking results. This is an important property, because GPUs are fast in processing large batches of images, but to run the inference an overhead in the calculation time compromises the real-time capability. We thus buffer detections for 1-2 s and run the inference then once. This hides the overhead due to initialization sufficiently. Although the tracking results are then delayed about this buffer length, it is still an online algorithm as results are continuously provided during runtime.

For training the network, we utilize the training scenes of the MOT20 and MOT17[1] datasets providing 3856 annotated tracks. From this, we extract 1437801 patches from detections with resolution 35×60. Each training batch consists of 50% pairs showing the same person and 50% showing different persons. We only use pairs from the same scene as otherwise the background from different scenes would obviously indicate different persons. Furthermore, in contrast to Siamese networks for reidentification, the temporal distance between image pairs is at most the timeout T, see Sec. II-A.2. Thus, we limit the temporal distance during training to 50 frames for a pair[2]. Due to the large number of possible pairs under these constraints (up to 10^{12}) we generate pairs randomly during training.

Training is performed with a batch size of 50 in 1000000 steps. The mean average error is minimized utilizing stochastic gradient descent.

C. Human Motion Model

As motivated in Sec. I-B, $c^h_{n,m}$ is a cost to model the human behavior in the cost function as a complement to the Kalman filter. A Kalman filter predicts the current path based on velocity and acceleration. However, the tracked person can change his or her mind at any time and e.g. turn back. The uncertainty estimation of the Kalman filter is not intended for the resulting error in the prediction. We thus model the uncertainty due to unpredictable changes in speed and direction by the following assumptions: A human walks generally at speed v_{max} and a change in direction is always and instantly possible. Assuming that t_d is the last time frame with a sensor update for track n, the maximum distance d^{max}_n walked since t_d is achieved by changing direction only at t_p and then walking at v_{max}. It can thus be defined by a linear function

$$d^{max}_n = (t - t_d) \cdot v_{max} + c_d, \tag{4}$$

where c_d is a constant that can be utilized to defined an initial uncertainty in the detected position. To determine $c^h_{n,m}$ for a track and detection pair (n, m), we use the euclidian distance $d(n, m)$ between them and scale this value by d_{max}:

$$c^h_{n,m} = \frac{d(n,m)}{d^{max}_n}. \tag{5}$$

As you can see, the cost is lower when the detection m is close to the track n and increases more slowly when we expect the track to be farther away due to missing detections.

III. Evaluation

As motivated in the introduction, we evaluate the effectiveness and real-time capability based on the MOT20 benchmark [4] using the usual metrics. The High Order Tracking Accuracy (HOTA) is the geometric mean of detection and association accuracy [7] and is considered as a better alignment with human subjective perception. It is considered as a furter development of the Multi Object Tracking Accuracy (MOTA), which combines false positives, missed targets and identity switches. In contrast, IDF1 is the F1 score for the ID and focuses on the association accuracy

[1]Note that moving cameras are only problematic for the Kalman filter. Training image patch similarities are not affected by this.

[2]We do not limit it to timeout T as this value may change after training.

Short	HOTA	MOTA	IDF1	MOTP	RT (s)
UnsupTrack	41.7	53.6	50.6	**80.1**	3467.3
TransCtr	43.5	61.0	49.8	79.5	4478.5
Tracktor++v2	42.1	52.6	52.7	79.9	3795.0
SFS	32.7	50.8	41.1	74.9	44 500.0
RTv1	**55.1**	60.6	**67.9**	78.8	1500.0
MOTer	44.3	**62.3**	50.3	79.9	4478.5
ITM	39.6	50.6	48.6	78.6	2500.0
HOMI_Tracker	37.3	51.2	43.0	79.6	**600.0**
GNNMatch	40.2	54.5	49.0	79.4	86 400.0
FGRNetIV	42.5	55.4	52.7	79.4	3500.0
D4C	51.5	54.8	64.4	77.7	819.6
ALBOD	43.5	56.5	51.1	79.4	3600.0
Surveily	36.0	44.6	42.5	76.1	150.5
SORT20	36.1	42.7	45.1	**78.5**	**78.2**
GMPHD_Rd20	35.6	44.7	43.5	77.5	177.9
LTSiam	**40.4**	**46.5**	**49.4**	77.1	148

TABLE I

RESULTS ON THE MOT20 BENCHMARK FOR ONLINE ALGORITHMS, DEVIDED INTO TWO PARTS FOR REAL-TIME SOLUTIONS (BOTTOM) AND NON-REAL-TIME (TOP). HERE, THE FIRST FOUR COLUMNS OF THE MOT20 BENCHMARK RESULTS ARE SHOWN. THE FULL LIST IS AVAILABLE AT MOTCHALLENGE.NET/RESULTS/MOT20. THE COLUMN RT SHOWS THE RUNTIME OF THE CORRESPONDING ALGORITHM FOR ALL 4479 FRAMES OF THE TEST SCENES.

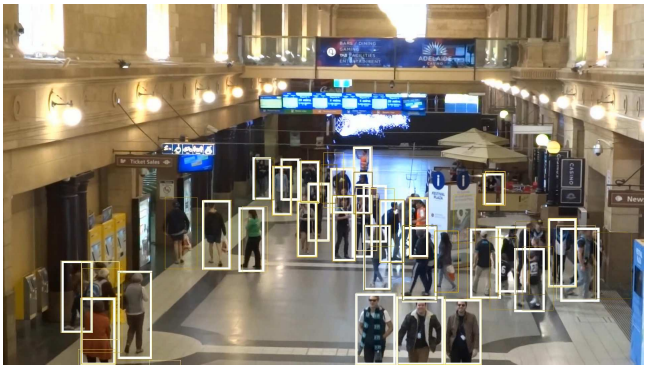

Fig. 3. Example shot from the first scene in MOT20 dataset. White boxes represent detections as given by the dataset and thin yellow lines tracks. Overall three person currently are not detected but still tracked well during motion.

rather than detection. The MOT Precision (MOTP) measures the overlap bewteen correct predictions and ground truth.

The tracking results of the test scenes must be submitted, ground truth data for own evaluation is not provided. Results are then automatically generated, listed in Table I. Fig. 3 and Fig. 4 depict qualitative results.

Note that in contrast to all other values the runtime is provided by the authors of the algorithms. Our evaluation system is equipped with an Intel Xeon Platinum 8180 Processor. We did not parallelize the algorithm, so only a single core is utilized except for the GPU parts. Running on the GPU is the inference of a fingerprint and the cosine similarity (in different steps). For this, we utilize an NVIDIA V100 GPU.

As can be seen in Table I and Fig. 1, among real-time capable approaches our proposed method performs best in

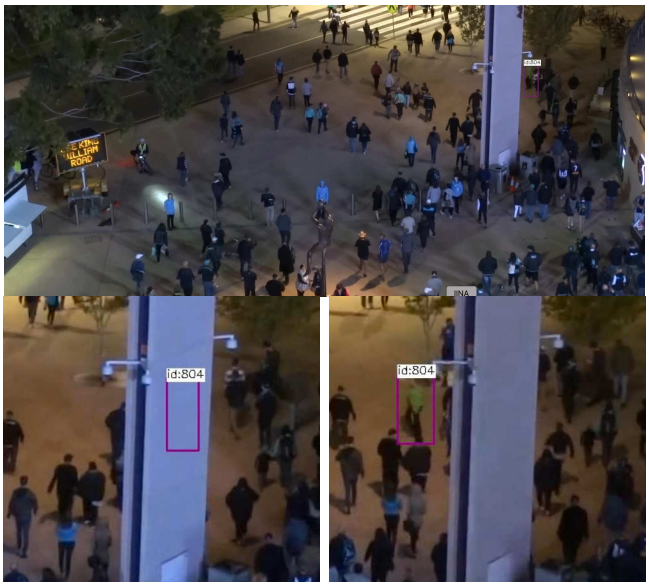

Fig. 4. Scene four in MOT20 dataset (overall image and 2 cutouts). From all tracks id 804 is marked before entering a hidden area, while walking behind and after that area. As can be seen, the location where the person is visible again is predicted well.

HOTA, MOTA and IDF1 and even outperforms non-real-time capable approaches.

As described in section I-A, SORT20 follows a similar approach ignoring appearance features. Thus, the proposed LTSiam and human motion model can be assumed as the main cause for the improved performance. In contrast, both utilize a Kalman filter to track positions. Thus, the overlap measured by MOTP is a matter of weighting reactivity and smooth tracking, with SORT20 having higher priority for precision here.

GMPHD_Rd20 applies a fast CNN called IdNet to include appearance features. However, caused by the design where training differs from inference, this leads to inferior results.

IV. CONCLUSION AND OUTLOOK

In this paper, we present a novel approach for real-time capable multi-object tracking in a surveillance scenario. It is based on the basic idea of associating given detections with tracks. For this, we use the Hungarian algorithm, which minimizes a cost matrix with fingerprints provided by LTSiam in linear time. In addition, a human motion model is added to improve the results. The evaluation shows that this outperforms other real-time capable approaches.

In future research, utilizing fingerprints could help to distinguish between different reasons for the disappearance of a person. To be precise, case 3 in Sec. II-A.2 could be identified by comparing the fingerprint of the patch at the current tracking position with the last patch where the person is known to be visible. However, as the current tracking position is required and vice versa, the fingerprint must be inferred in each frame. Further research is required to avoid the additional overhead.

REFERENCES

[1] Nathanael L. Baisa. Occlusion-robust online multi-object visual tracking using a gm-phd filter with cnn-based re-identification. 2020. arXiv: 1912.05949.

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles, 2019. arXiv: 1903.05625.

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[4] 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. Mar. 2020. arXiv: 2003.09003.

[5] Shengyong Ding, Liang Lin, Guangrun Wang, and Hongyang Chao. Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*, 48(10):2993–3003, 2015.

[6] Shyamgopal Karthik, Ameya Prabhu, and Vineet Gandhi. Simple unsupervised multi-object tracking, 2020. arXiv: 2006.02609.

[7] Jonathon Luiten, Aljossa Ossep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixe, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578, Oct 2020.

[8] Ioannis Papakis, Abhijit Sarkar, and Anuj Karpatne. Gcnnmatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization, 2021.

[9] Roman Pflugfelder. An in-depth analysis of visual tracking with siamese neural networks. *arXiv preprint arXiv:1707.00569*, 2017.

[10] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 91–99, 2015.

[11] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. Gated siamese convolutional neural network architecture for human re-identification. In *European conference on computer vision*, pages 791–808. Springer, 2016.

[12] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking, 2021. arXiv: 2103.15145.