remote sensing

MDPI

Article

Multiple Pedestrians and Vehicles Tracking in Aerial Imagery Using a Convolutional Neural Network

Seyed Majid Azimi [1,2,*], Maximilian Kraus [3], Reza Bahmanyar [1] and Peter Reinartz [1]

[1] Remote Sensing Technology Institute (IMF), German Aerospace Center (DLR), 82234 Wessling, Germany; reza.bahmanyar@dlr.de (R.B.); peter.reinartz@dlr.de (P.R.)
[2] Department of Aerospace and Geodesy, Technical University of Munich, 80333 Munich, Germany
[3] Department of Informatics, Technical University of Munich, 85748 Garching, Germany; maximilian.kraus@tum.de
* Correspondence: seyedmajid.azimi@dlr.de

Abstract: In this paper, we address various challenges in multi-pedestrian and vehicle tracking in high-resolution aerial imagery by intensive evaluation of a number of traditional and Deep Learning based Single- and Multi-Object Tracking methods. We also describe our proposed Deep Learning based Multi-Object Tracking method AerialMPTNet that fuses appearance, temporal, and graphical information using a Siamese Neural Network, a Long Short-Term Memory, and a Graph Convolutional Neural Network module for more accurate and stable tracking. Moreover, we investigate the influence of the Squeeze-and-Excitation layers and Online Hard Example Mining on the performance of AerialMPTNet. To the best of our knowledge, we are the first to use these two for regression-based Multi-Object Tracking. Additionally, we studied and compared the $L1$ and Huber loss functions. In our experiments, we extensively evaluate AerialMPTNet on three aerial Multi-Object Tracking datasets, namely AerialMPT and KIT AIS pedestrian and vehicle datasets. Qualitative and quantitative results show that AerialMPTNet outperforms all previous methods for the pedestrian datasets and achieves competitive results for the vehicle dataset. In addition, Long Short-Term Memory and Graph Convolutional Neural Network modules enhance the tracking performance. Moreover, using Squeeze-and-Excitation and Online Hard Example Mining significantly helps for some cases while degrades the results for other cases. In addition, according to the results, $L1$ yields better results with respect to Huber loss for most of the scenarios. The presented results provide a deep insight into challenges and opportunities of the aerial Multi-Object Tracking domain, paving the way for future research.

Keywords: aerial imagery; deep neural networks; GraphCNN; recurrent neural networks; multi-object tracking

Citation: Azimi, S.; Kraus, M.; Bahmanyar, R.; Reinartz, P. Multiple Pedestrians and Vehicles Tracking in Aerial Imagery Using a Convolutional Neural Network. *Remote Sens.* **2021**, *13*, 1953. https://doi.org/10.3390/rs13101953

Academic Editor: Melanie Vanderhoof

Received: 6 April 2021
Accepted: 4 May 2021
Published: 17 May 2021

1. Introduction

Visual Object Tracking (VOT), that is, locating objects in video frames over time, is a dynamic field of research with a wide variety of practical applications such as in autonomous driving, robot aided surgery, security, and safety. The recent advances in machine and deep learning techniques have drastically boosted the performance of VOT methods by solving long-standing issues such as modeling appearance feature changes and relocating the lost objects [1–3]. Nevertheless, the performance of the existing VOT methods is not always satisfactory due to hindrances such as heavy occlusions, difference in scales, background clutter or high-density in the crowded scenes. Thus, developing more sophisticated VOT methods overcoming these challenges is highly demanded.

The VOT methods can be categorized into Single-Object Tracking (SOT) and Multi-Object Tracking (MOT) methods, which track single and multiple objects throughout subsequent video frames, respectively. The MOT scenarios are often more complex than the SOT because the trackers must handle a larger number of objects in a reasonable time

(e.g., ideally real-time). Most previous VOT works using traditional approaches such as Kalman and particle filters [4,5], Discriminative Correlation Filter (DCF) [6], or silhouette tracking [7], simplify the tracking procedure by constraining the tracking scenarios with, for example, stationary cameras, limited number of objects, limited occlusions, or absence of sudden background or object appearance changes. These methods usually use handcrafted feature representations (e.g., Histogram of Gradients (HOG) [8], color, position) and their target modeling is not dynamic [9]. In real-world scenarios, however, such constraints are often not applicable and VOT methods based on these traditional approaches perform poorly.

The rise of Deep Learning (DL) offered several advantages in object detection, segmentation, and classification [10–12]. Approaches based on DL have also been successfully applied to VOT problems, and significantly enhancing the performance, especially in unconstrained scenarios. Examples include the Convolutional Neural Network (CNN) [13,14], Recurrent Neural Network (RNN) [15], Siamese Neural Network (SNN) [16,17], Generative Adversarial Network (GAN) [18] and several customized architectures [19].

Despite the many progress made for VOT in ground imagery, in the remote sensing domain, VOT has not been fully exploited, due to the limited available volume of images with high enough resolution and level of details. In recent years, the development of more advanced camera systems and the availability of very high-resolution aerial images have opened new opportunities for research and applications in the aerial VOT domain ranging from the analysis of ecological systems to aerial surveillance [20,21].

Aerial imagery allows collecting very high-resolution data from wide open areas in a cost- and time-efficient manner. Performing MOT based on such images (e.g., with Ground Sampling Distance (GSD) $<$ 20 cm/pixel) allows us to track and monitor the movement behaviours of multiple small objects such as pedestrians and vehicles for numerous applications such as disaster management and predictive traffic and event monitoring. However, few works have addressed aerial MOT [22–24], and the aerial MOT datasets are rare. The large number and the small sizes of moving objects compared to the ground imagery scenarios together with large image sizes, moving cameras, multiple image scale, low frame rates as well as various visibility levels and weather conditions makes MOT in aerial imagery especially complicate. Existing drone or ground surveillance datasets frequently used as MOT benchmarks, such as MOT16 and MOT17 [25], are very different from aerial MOT scenarios with respect to their image and object characteristics. For example, the objects are bigger and the scenes are less crowded, with the objects appearance features usually being discriminative enough to distinguish the objects. Moreover, the videos have higher frame rates and better qualities and contrasts.

In this paper, we aim at investigating various existing challenges in the tracking of multiple pedestrian and vehicles in aerial imagery through intensive experiments with a number of traditional and DL-based SOT and MOT methods. This paper extends our recent work [26], in which we introduced a new MOT dataset, the so-called Aerial Multi-Pedestrian Tracking (AerialMPT), as well as a novel DL-based MOT method, the so-called AerialMPTNet, that fuses appearance, temporal, and graphical information for a more accurate MOT. In this paper, we also extensively evaluate the effectiveness of different parts of AerialMPTNet and compare it to traditional and state-of-the-art DL-based MOT methods. Additionally, we propose a MOT method inspired by the SORT method [27], the so-called Euclidean Online Tracking (EOT), which employs GSD adapted Euclidean distance for object association in consecutive frames.

We conduct our experiments on three aerial MOT datasets, namely AerialMPT and KIT AIS (https://www.ipf.kit.edu/code.php, accessed on 10 May 2021) pedestrian and vehicle datasets. All image sequences were captured by an airborne platform during different flight campaigns of the German Aerospace Center (DLR) (https://www.dlr.de, accessed on 10 May 2021) and vary significantly in object density, movement patterns, and image size and quality. Figure 1 shows sample images from the AerialMPT dataset with the tracking results of our AerialMPTNet. The images were captured at different flight

altitudes and their GSD (reflecting the spatial size of a pixel) varies between 8 cm and 13 cm. The total number of objects per sequence ranges up to 609. Pedestrians in these datasets appear as small points, hardly exceeding an area of 4 × 4 pixels. Even for human experts, distinguishing multiple pedestrians based on their appearance is laborious and challenging. Vehicles appear as bigger objects and are easier to distinguish based on their appearance features. However, different vehicle sizes, fast movements together with the low frame rates (e.g., 2 fps) and occlusions by bridges, trees, or other vehicles presents challenges to the vehicle tracking algorithm, illustrated in Figure 2.

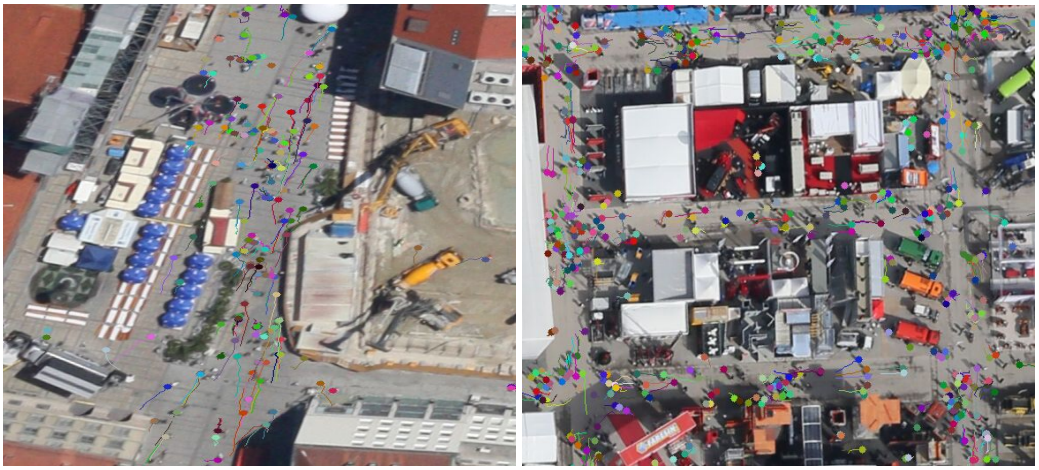
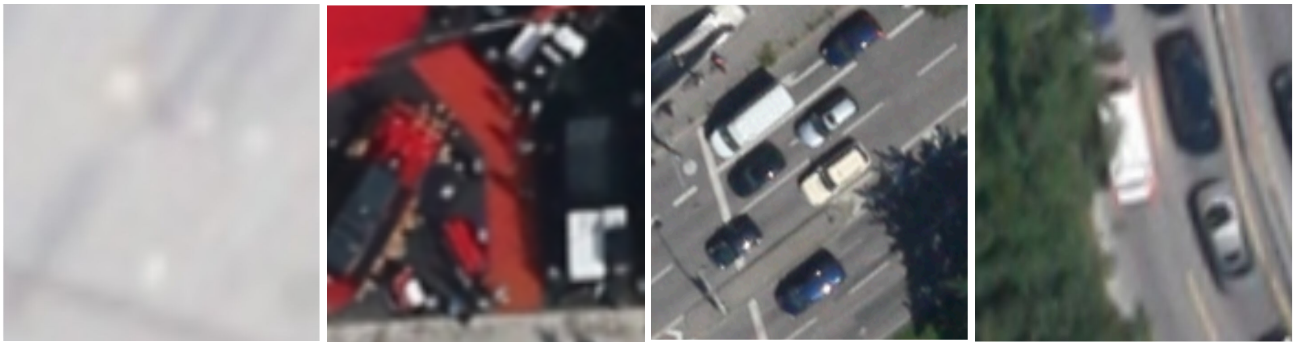


Figure 1. Multi-Pedestrian tracking results of AerialMPTNet on the frame 18 of the "Munich02" (**left**) and frame 10 of the "Bauma3" (**right**) sequences of the AerialMPT dataset. Different pedestrians are depicted in different colors with the corresponding trajectories.



(**a**) (**b**) (**c**) (**d**)

Figure 2. Illustrations of some challenges in aerial MOT datasets. The examples are from the KIT AIS pedestrian (**a**), AerialMPT (**b**), and KIT AIS vehicle datasets (**c**,**d**). Multiple pedestrians which are hard to distinguish due to their similar appearance features and low image contrast (**a**). Multiple pedestrians at a trade fair walking closely together with occlusions, shadows, and strong background colors (**b**). Multiple vehicles at a stop light where the shadow on the right hand side can be problematic (**c**). Multiple vehicles with some of them occluded by trees (**d**).

AerialMPTNet is an end-to-end trainable regression-based neural network comprising a SNN module which takes two image patches as inputs, a target and a search patch, cropped from a previous and a current frame, respectively. The object location is known in the target patch and should be predicted for the search patch. In order to overcome the tracking challenges of the aerial MOT such as the objects with similar appearance features and densely moving together, AerialMPTNet incorporates temporal and graphical information in addition to the appearance information provided by the SNN module. Our AerialMPTNet employs a Long Short-Term Memory (LSTM) for temporal information extraction and movement prediction, and a Graph Convolutional Neural Network (GCNN)

for modeling the spatial and temporal relationships between adjacent objects (graphical information). AerialMPTNet outputs four values indicating the coordinates of the top-left and bottom-right corners of each object's bounding box in the search patch. In this paper, we also investigate the influence of Squeeze-and-Excitation (SE) and Online Hard Example Mining (OHEM) [28] on the tracking performance of AerialMPTNet. To the best of our knowledge, we are the first work applying adaptive weighting of convolutional channels by SE and employ OHEM for the training of a DL-based tracking-by-regression method.

According to the results, our AerialMPTNet outperforms all previous methods for the pedestrian datasets and achieves competitive results for the vehicle dataset. Furthermore, LSTM and GCNN modules adds value to the tracking performance. Moreover, while using SE and OHEM can significantly help in some scenarios, in other cases they may degrade the tracking results. In summary, the contributions of this paper over our previous work [26] are:

- We apply OHEM and SE to a MOT task for the first time.
- We propose EOT which outperforms tracking methods with Intersection over Union (IoU)-based association strategy.
- We conduct an ablation study to evaluate the role of all different parts of AerialMPTNet.
- We evaluate the role of loss functions in the tracking performance by comparing $L1$ and Huber loss functions.
- We evaluated and compared various MOT methods for pedestrian tracking in aerial imagery.
- We conduct intensive qualitative and quantitative evaluations of AerialMPTNet on two aerial pedestrian and one aerial vehicle tracking datasets.

We believe that our paper can promote research on aerial MOT (esp. for pedestrians and vehicles) by providing a deep insight into its challenges and opportunities.

The rest of the paper is organized as follows: Section 2 presents an overviews on related works; Section 3 introduces the datasets used in our experiments; Section 4 represents the metrics used for our quantitative evaluations; Section 5 provides a comprehensive study on previous traditional and DL-based tracking methods on the aerial MOT datasets, with Section 8.4 explaining our AerialMPTNet with all its configurations; Section 7 represents our experimental setups; Section 8 provides an extensive evaluation of our AerialMPTNet and compares it to the other methods; and Section 10 concludes our paper and gives ideas for future works.

2. Related Works

Visual object tracking is defined as locating one or multiple objects in videos or image sequences over time. The traditional tracking process comprises four phases including initialization, appearance modeling, motion modeling, and object finding. During initialization, the targets are detected manually or by an object detector. In the appearance modeling step, visual features of the region of interest are extracted by various learning-based methods for detecting the target objects. The variety of scales, rotations, shifts, and occlusions makes this step challenging. Image features play a key role in the tracking algorithms. They can be mainly categorized into handcrafted and deep features. In recent years, research studies and applications have focused on developing and using deep features based on DNNs which have shown to be able to incorporate multi-level information and more robustness against appearance variations [29]. Nevertheless, DNNs require large enough training datasets, which are not always available. Thus, for many applications, the handcrafted features are still preferable. The motion modeling step aims at predicting the object movement in time and estimate the object locations in the next frames. This procedure effectively reduces the search space and consequently the computation cost. Widely used methods for motion modeling include Kalman filter [30], Sequential Monte Carlo methods [31] and RNNs. In the last step, object locations are found as the ones close to the estimated locations by the motion model.

2.1. Various Categorizations of VOT

Visual object tracking methods can be divided into SOT [32,33] and MOT [22,34] methods. While SOTs only track a single predetermined object throughout a video, even if there are multiple objects, MOTs can track multiple objects at the same time. Thus, MOTs can face exponential complexity and runtime increase based on the number of objects as compared to SOTs.

Object tracking methods also can be categorized into detection-based [35] and detection-free methods [36]. While the detection-based methods utilize object detectors to detect objects in each frame, the detection-free methods only need the initial object detection. Therefore, detection-free methods are usually faster than the detection-based ones; however, they are not able to detect new objects entering the scene and require manual initialization.

Object tracking methods can be further divided based on their training strategies using either online or offline learning strategy. The methods with an online learning strategy can learn about the tracked objects during runtime. Thus, they can track generic objects [37]. The methods with offline learning strategy are trained beforehand and are therefore faster during runtime [38].

Tracking methods can be categorized into online and offline. Offline trackers take advantage of past and futures frames, while online ones can only infer from past frames. Although having all frames by offline tracking methods can increase the performance, in real-world scenarios future frames are not available.

Most existing tracking approaches are based on a two-stage tracking-by-detection paradigm [39,40]. In the first stage, a set of target samples is generated around the previously estimated position using region proposal, random sampling, or similar methods. In the second stage, each target sample is either classified as background or as the target object. In one-stage-tracking, however, the model receives a search sample together with a target sample as two inputs and directly predicts a response map or object coordinates by a previously trained regressor [17,22].

Object tracking methods can be categorized into the Traditional and DL-Based ones. Traditional tracking methods mostly rely on the Kalman and particle filters to estimate object locations. They use velocity and location information to perform tracking [4,5,41]. Tracking methods only relying on such approaches have shown poor performance in unconstrained environments. Nevertheless, such filters can be advantageous in limiting the search space (decreasing the complexity and computational cost) by predicting and propagating object movements to the following frames. A number of traditional tracking methods follow a tracking-by-detection paradigm based on template matching [42]. A given target patch models the appearance of the region of interest in the first frame. Matched regions are then found in the next frame using correlation, normalized cross-correlation, or the sum of squared distances methods [43,44]. Scale, illumination, and rotation changes can cause difficulties with these methods. More advanced tracking-by-detection-based methods rely on discriminative modeling, separating targets from their backgrounds within a specific search space. Various methods have been proposed for discriminative modeling, such as boosting methods and Support Vector Machines (SVMs) [45,46]. A series of traditional tracking algorithms, such as MOSSE and KCF [6,47], utilizes correlation filters, which model the target's appearance by a set of filters trained on the images. In these methods, the target object is initially selected by cropping a small patch from the first frame centered at the object. For the tracking, the filters are convolved with a search window in the next frame. The output response map assumes to have a peak at the target's next location. As the correlation can be computed in the Fourier domain, such trackers achieve high frame rates.

Recently, many research works and applications have focused on using DL-based tracking methods. The great advantage of DL-based features over handcrafted ones such as HOG, raw pixels values or grey-scale templates have been presented previously for a variety of computer vision applications. These features are robust against appearance changes, occlusions, and dynamic environments. Examples of DL-based tracking methods

include re-identification with appearance modeling and deep features [34], position regression mainly based on SNNs [16,17], path prediction based on RNN-like networks [48], and object detection with DNNs such as YOLO [49].

2.2. SOTs and MOTs

Among various categorizations, in this section, we consider the SOT and MOT one for reviewing the existing object tracking methods. We believe that this is the fundamental categorization of the tracking methods which significantly affects the method design. In the following, we briefly introduce a few recent methods from both categories and experimentally discuss their strengths and limitations on aerial imagery in Section 5.

2.2.1. SOT Methods

Kalal et al. proposed Median Flow [50], which utilizes point and optical flow tracking. The inputs to the tracker are two consecutive images together with the initial bounding box of the target object. The tracker calculates a set of points from a rectangular grid within the bounding box. Each of these points is tracked by a Lucas-Kanade tracker generating a sparse motion flow. Afterwards, the framework evaluates the quality of the predictions and filters out the worst 50%. The remaining point predictions are used to calculate the new bounding box positions considering the displacement.

MOSSE [6], KFC [47] and CSRT [51] are based upon DCFs. Bolme et al. [6] proposed MOSSE which uses a new type of correlation filter called Minimum Output Sum of Squared Errors (MOSSE), which aims at producing stable filters when initialized using only one frame and grey-scale templates. MOSSE is trained with a set of training images f_i and training outputs g_i, where g_i is generated from the ground truth as a 2D Gaussian centered on the target. This method can achieve state-of-the-art performances while running with high frame rates. Henriques et al. [47] replaced the grey-scale templates with HOG features and proposed the idea of Kernelized Correlation Filter (KCF). KCF works with multiple channel-like correlation filters. Additionally, the authors proposed using non-linear regression functions which are stronger than linear functions and provide non-linear filters that can be trained and evaluated as efficiently as linear correlation filters. Similar to KCF, dual correlation filters use multiple channels. However, they are based on linear kernels to reduce the computational complexity while maintaining almost the same performance as the non-linear kernels. Recently, Lukezic et al. [51] proposed to use channel and reliability concepts to improve tracking based on DCFs. In this method, the channel-wise reliability scores weight the influence of the learned filters based on their quality to improve the localization performance. Furthermore, a spatial reliability map concentrates the filters to the relevant part of the object for tacking. This makes it possible to widen the search space and improves the tracking performance for non-rectangular objects.

As we stated before, the choice of appearance features plays a crucial role in object tracking. Most previous DCF-based works utilize handcrafted features such as HOG, grey-scale features, raw pixels, and color names or the deep features trained independently for other tasks. Wang et al. [32] proposed an end-to-end trainable network architecture able to learn convolutional features and perform the correlation-based tracking simultaneously. The authors encode a DCF as a correlation filter layer into the network, making it possible to backpropagate the weights through it. Since the calculations remain in the Fourier domain, the runtime complexity of the filter is not increased. The convolutional layers in front of the DCF encode the prior tracking knowledge learned during an offline training process. The DCF defines the network output as the probability heatmaps of object locations.

In the case of generic object tracking, the learning strategy is typically entirely online. However, online training of neural networks is slow due to backpropagation leading to a high run time complexity. However, Held et al. [17] developed a regression-based tracking method, called GOTURN, based on a SNN, which uses an offline training approach helping the network to learn the relationship between appearance and motion. This makes the tracking process significantly faster. This method utilizes the knowledge gained during the

offline training to track new unknown objects online. The authors showed that without online backpropagation, GOTURN can track generic objects at 100 fps. The inputs to the network are two image patches cropped from the previous and current frames, centered at the known object position in the previous frame. The size of the patches depends on the object bounding box sizes and can be controlled by a hyperparameter. This determines the amount of contextual information given to the network. The network output is the coordinates of the object in the current image patch, which is then transformed to the image coordinates. GOTURN achieves state-of-the-art performance on common SOT benchmarks such as VOT 2014 (https://www.votchallenge.net/vot2014/, accessed on 10 May 2021).

2.2.2. MOT Methods

Bewley et al. [27] proposed a simple multi-object tracking approach, called SORT, for online tracking applications. Bounding box position and size are the only values used for motion estimation and assigning the objects to their new positions in the next frame. In the first step, objects are detected using Faster R-CNN [12]. Subsequently, a linear constant velocity model approximates the movements of each object individually in consecutive frames. Afterwards, the algorithm compares the detected bounding boxes to the predicted ones based on IoU, resulting in a distance matrix. The Hungarian algorithm [52] then assigns each detected bounding box to a predicted (target) bounding box. Finally, the states of the assigned targets are updated using a Kalman filter. SORT runs with more than 250 Frames per Second (fps) with almost state-of-the-art accuracy. Nevertheless, occlusion scenarios and re-identification issues are not considered for this method, which makes it inappropriate for long-term tracking.

Wojke et al. [34] extended SORT to DeepSORT and tackled the occlusion and re-identification challenges, keeping the track handling and Kalman filtering modules almost unaltered. The main improvement takes place into the assignment process, in which two additional metrics are used: (1) motion information provided based on the Mahalanobis distance between the detected and predicted bounding boxes, (2) appearance information by calculating the cosine distance between the appearance features of a detected object and the already tracked object. The appearance features are computed by a deep neural network trained on a large person re-identification dataset [53]. A cascade strategy then determines object-to-track assignments. This strategy effectively encodes the probability spread in the association likelihood. DeepSORT performs poorly if the cascade strategy cannot match the detected and predicted bounding boxes.

Recently, Bergmann et al. [1] introduced Tracktor++ which is based on the Faster R-CNN object detection method. Faster R-CNN classifies region proposals to target and background and fits the selected bounding boxes to object contours by a regression head. The authors trained Faster R-CNN on the MOT17Det pedestrian dataset [25]. The first step is an object detection by Faster R-CNN. The detected objects in the first frame are then initialized as tracks. Afterwards, the tracks are tracked in the next frame by regressing their bounding boxes using the regression head. In this method, the lost or deactivated tracks can be re-identified in the following frames using a SNN and a constant velocity motion model.

2.3. Tracking in Satellite and Aerial Imagery

The reviewed object tracking methods in the previous sections have been mainly developed for computer vision datasets and challenges. In this section, we focus on the proposed methods for satellite and aerial imagery. Visual object tracking for targets such as pedestrians and vehicles in satellite and aerial imagery is a challenging task that has been addressed by only few works, compared to the huge number addressing pedestrian and vehicle tracking in ground imagery [13,54].Tracking in satellite and aerial imagery is much more complex. This is due to the moving cameras, large image sizes, different scales, large number of moving objects, tiny size of the objects (e.g., 4×4 pixels for pedestrians,

30 × 15 for vehicles), low frame rates, different visibility levels, and different atmospheric and weather conditions [25,55].

2.3.1. Tracking by Moving Object Detection

Most of the previous works in satellite and aerial object tracking are based on moving object detection [23,24,56]. Reilly et al. [23] proposed one of the earliest aerial object tracking approaches focusing on vehicle tracking mainly in highways. They compensate camera motion by a correction method based on point correspondence. A median background image is then modeled from ten frames and subtracted from the original frame for motion detection, resulting in the moving object positions. All images are split into overlapping grids, with each one defining an independent tracking problem. Objects are tracked using bipartite graph, matching a set of label nodes and a set of target nodes. The Hungarian algorithm solves the cost matrix afterwards to determine the assignments. The usage of the grids allows tracking large number of objects with the $O(n^3)$ runtime complexity for the Hungarian algorithm.

Meng et al. [24] followed the same direction. They addressed the tracking of ships and grounded aircrafts. Their method detects moving objects by calculating an Accumulative Difference Image (ADI) from frame to frame. Pixels with high values in the ADI are likely to be moving objects. Each target is afterwards modeled by extracting its spectral and spatial features, where spectral features refer to the target probability density functions and the spatial features to the target geometric areas. Given the target model, matching candidates are found in the following frames via regional feature matching using a sliding window paradigm.

Tracking methods based on moving object detection are not applicable for our pedestrian and vehicle tracking scenarios. For instance, Reilly et al. [23] use a road orientation estimate to constrain the assignment problem. Such estimations which may work for vehicles moving along predetermined paths (e.g., highways and streets), do not work for pedestrian tracking with much more diverse and complex movement behaviors (e.g., crowded situations and multiple crossings). In general, such methods perform poorly in unconstrained environments, are sensitive to illumination change and atmospheric conditions (e.g., clouds, shadows, or fog), suffer from the parallax effect, and cannot handle small or static objects. Additionally, since finding the moving objects requires considering multiple frames, these methods cannot be used for the real-time object tracking.

2.3.2. Tracking by Appearance Features

The methods based on appearance-like features overcome the issues of the tracking by moving object detection approaches [22,57–60], making it possible to detect small and static objects on single images. Butenuth et al. [57] deal with pedestrian tracking in aerial image sequences. They employ an iterative Bayesian tracking approach to track numerous pedestrians, where each pedestrian is described by its position, appearance features, and direction. A linear dynamic model then predicts futures states. Each link between a prediction and a detection is weighted by evaluating the state similarity and associated with the direct link method described in [35]. Schmidt et al. [58] developed a tracking-by-detection framework based on Haar-like features. They use a Gentle AdaBoost classifier for object detection and an iterative Bayesian tracking approach, similar to [57]. Additionally, they calculate the optical flow between consecutive frames to extract motion information. However, due to the difficulties of detecting small objects in aerial imagery, the performance of the method is degraded by a large number of false positives and negatives.

Bahmanyar et al. [22] proposed Stack of Multiple Single Object Tracking CNNs (SMSOT-CNN) and extended the GOTURN method, a SOT method developed by Held et al. [17], by stacking the architecture of GOTURN to track multiple pedestrians and vehicles in aerial image sequences. SMSOT-CNN is the only previous DL-based work dealing with MOT. SMSOT-CNN expands the GOTURN network by three additional convolutional layers to improve the tracker's performance in locating the object in the search area. In their

architecture, each SOT-CNN is responsible for tracking one object individually leading to a linear increase in the tracking complexity by the number of objects. They evaluate their approach on the vehicle and pedestrian sets of the KIT AIS aerial image sequence dataset. Experimental results show that SMSOT-CNN significantly outperforms GOTURN. Nevertheless, SMSOT-CNN performs poorly in crowded situations and when objects share similar appearance features.

In Section 5, we experimentally investigate a set of the reviewed visual object tracking methods on three aerial object tracking datasets.

3. Datasets

In this section, we introduce the datasets used in our experiments, namely the KIT AIS (pedestrian and vehicle sets), the Aerial Multi-Pedestrian Tracking (AerialMPT) [26], and DLR's Aerial Crowd Dataset (DLR-ACD) [61]. All these datasets are the first of their kind and aim at promoting pedestrian and vehicle detection and tracking based on aerial imagery. The images of all these datasetes have been acquired by the German Aerospace Center (DLR) using the 3K camera system, comprising a nadir-looking and two side-looking DSLR cameras, mounted on an airborne platform flying at different altitudes. The different flight altitudes and camera configurations allow capturing images with multiple spatial resolutions (ground sampling distances-GSDs) and viewing angles.

For the tracking datasets, since the camera is continuously moving, in a post-processing step, all images were orthorectified with a digital elevation model, co-registered, and geo-referenced with a GPS/IMU system. Afterwards, images taken at the same time were fused into a single image and cropped to the region of interest. This process caused small errors visible in the frame alignments. Moreover, the frame rate of all sequences is 2 Hz. The image sequences were captured during different flight campaigns and differ significantly in object density, movement patterns, qualities, image sizes, viewing angles, and terrains. Furthermore, different sequences are composed by a varying number of frames ranging from 4 to 47. The number of frames per sequence depends on the image overlap in flight direction and the camera configuration.

3.1. KIT AIS

The KIT AIS dataset is generated for two tasks, vehicle and pedestrian tracking. The data have been annotated manually by human experts and suffer from a few human errors. Vehicles are annotated by the smallest enclosing rectangle (i.e., bounding box) oriented in the direction of their travel, while individual pedestrians are marked by point annotations on their heads. In our experiments, we used bounding boxes of sizes 4×4 and 5×5 pixels for the pedestrians according to the GSDs of the images, ranging from 12 to 17 cm. As objects may leave the scene or be occluded by other objects, the tracks are not labeled continuously for all cases. For the vehicle set cars, trucks, and buses are annotated if they lie entirely within the image region with more than $\frac{2}{3}$ of their bodies visible. In the pedestrian set only pedestrians are labeled. Due to crowded scenarios or adverse atmospheric conditions in some frames, pedestrians can be hardly visible. In these cases, the tracks have been estimated by the annotators as precisely as possible. Tables 1 and 2 represent the statistics of the pedestrian and vehicle sets of the KIT AIS dataset, respectively.

The KIT AIS pedestrian is composed of 13 sequences with 2649 pedestrians (Pedest.), annotated by 32,760 annotation points (Anno.) throughout the frames Table 1. The dataset is split into 7 training and 6 testing sequences with 104 and 85 frames (Fr.), respectively. The sequences are characterized by different lengths ranging from 4 to 31 frames. The image sequences come from different flight campaigns over Allianz Arena (Munich, Germany), Rock am Ring concert (Nuremberg, Germany), and Karlsplatz (Munich, Germany).

Table 1. Statistics of the KIT AIS pedestrian dataset.

Train						
Seq.	**Image Size**	**#Fr.**	**#Pedest.**	**#Anno.**	**#Anno./Fr.**	**GSD**
AA_Crossing_01	309 × 487	18	164	2618	145.4	15.0
AA_Easy_01	161 × 168	14	8	112	8.0	15.0
AA_Easy_02	338 × 507	12	16	185	15.4	15.0
AA_Easy_Entrance	165 × 125	19	83	1105	58.3	15.0
AA_Walking_01	227 × 297	13	40	445	34.2	15.0
Munich01	509 × 579	24	100	1308	54.5	12.0
RaR_Snack_Zone_01	443 × 535	4	237	930	232.5	15.0
Total		104	633	6703	64.4	
Test						
AA_Crossing_02	322 × 537	13	94	1135	87.3	15.0
AA_Entrance_01	835 × 798	16	973	14,031	876.9	15.0
AA_Walking_02	516 × 445	17	188	2671	157.1	15.0
Munich02	702 × 790	31	230	6125	197.6	12.0
RaR_Snack_Zone_02	509 × 474	4	220	865	216.2	15.0
RaR_Snack_Zone_04	669 × 542	4	311	1230	307.5	15.0
Total		85	2016	26,057	306.5	

The KIT AIS vehicle comprises nine sequences with 464 vehicles annotated by 10,817 bounding boxes throughout 239 frames. It has no pre-defined train/test split. For our experiments, we split the dataset into five training and four testing sequences with 131 and 108 frames, respectively, similarly to [22]. According to Table 2, the lengths of the sequences vary between 14 and 47 frames. The image sequences have been acquired from a few highways, crossroads, and streets in Munich and Stuttgart, Germany. The dataset presents several tracking challenges such as lane change, overtaking, and turning maneuvers as well as partial and total occlusions by big objects (e.g., bridges). Figure 3 demonstrates sample images from the KIT AIS vehicle dataset.

Table 2. Statistics of the KIT AIS vehicle dataset.

Train						
Seq.	**Image Size**	**#Fr.**	**#Vehic.**	**#Anno.**	**#Anno./Fr.**	**GSD**
MunichAutobahn1	633 × 988	16	16	161	10.1	15.0
MunichCrossroad1	684 × 547	20	30	509	25.5	12.0
MunichStreet1	1764 × 430	25	57	1338	53.5	12.0
MunichStreet3	1771 × 422	47	88	3071	65.3	12.0
StuttgartAutobahn1	767 × 669	23	43	764	33.2	17.0
Total		131	234	5843	44.6	
Test						
MunichCrossroad2	895 × 1036	45	66	2155	47.9	12.0
MunichStreet2	1284 × 377	20	47	746	37.3	12.0
MunichStreet4	1284 × 388	29	68	1519	52.4	12.0
StuttgartCrossroad1	724 × 708	14	49	554	39.6	17.0
Total		108	230	4974	46.1	

Figure 3. Sample images from the KIT AIS vehicle dataset acquired at different locations in Munich and Stuttgart, Germany.

3.2. AerialMPT

The Aerial Multi-Pedestrian Tracking (AerialMPT) dataset [26] is newly introduced to the community, and deals with the shortcomings of the KIT AIS dataset such as the poor image quality and limited diversity. AerialMPT consists of 14 sequences with 2528 pedestrians annotated by 44,740 annotation points throughout 307 frames Table 3. Since the images have been acquired by a newer version of the DLR's 3K camera system, their quality and contrast are much better than the images of KIT AIS dataset. Figure 4 compares a few sample images from the AerialMPT and KIT AIS datasets.

Table 3. Statistics of the AerialMPT dataset.

Train						
Seq.	**Image Size**	**#Fr.**	**#Pedest.**	**#Anno.**	**#Anno./Fr.**	**GSD**
Bauma1	462 × 306	19	270	4448	234.1	11.5
Bauma2	310 × 249	29	148	3627	125.1	11.5
Bauma4	281 × 243	22	127	2399	109.1	11.5
Bauma5	281 × 243	17	94	1410	82.9	11.5
Marienplatz	316 × 355	30	215	5158	171.9	10.5
Pasing1L	614 × 366	28	100	2327	83.1	10.5
Pasing1R	667 × 220	16	86	1196	74.7	10.5
OAC	186 × 163	18	92	1287	71.5	8.0
Total		179	1132	21,852	122.1	
Test						
Bauma3	611 × 552	16	609	8788	549.2	11.5
Bauma6	310 × 249	26	270	5314	204.4	11.5
Karlsplatz	283 × 275	27	146	3374	125.0	10.0
Pasing7	667 × 220	24	103	2064	86.0	10.5
Pasing8	614 × 366	27	83	1932	71.6	10.5
Witt	353 × 1202	8	185	1416	177.0	13.0
Total		128	1396	22,888	178.8	

AerialMPT is split into 8 training and 6 testing sequences with 179 and 128 frames, respectively. The lengths of the sequences vary between 8 and 30 frames. The image sequences were selected from different crowd scenarios, for example, from moving pedestrians on mass events and fairs to sparser crowds in the city centers. Figure 1 demonstrates an image from the AerialMPT dataset with the overlaid annotations.

3.2.1. AerialMPT vs. KIT AIS

The AerialMPT has been generated in order to mitigate the limitations of the KIT AIS pedestrian dataset. In addition to the higher quality of the images, the numbers of minimum annotations per frame and the total annotations of AerialMPT are significantly larger than those of the KIT AIS dataset. All sequences in AerialMPT contain at least 50 pedestrians,

while more than 20% of the sequences of KIT AIS include less than ten pedestrians. Based on our visual inspection, not only the pedestrian movements in AerialMPT are more complex and realistic, but also the diversity of the crowd densities are greater than those of KIT AIS. The sequences in AerialMPT differ in weather conditions and visibility, incorporating more diverse kinds of shadows as compared to KIT AIS. Furthermore, the sequences of AerialMPT are longer in average, with 60% longer than 20 frames (less than 20% in KIT AIS). Further details on these datasets can be found in [26].

Figure 4. Sample images from the AerialMPT and KIT AIS datasets. "Bauma3", "Witt", "Pasing1" are from AerialMPT. "Entrance_01", "Walking_02", and "Munich02" are from KIT AIS.

3.3. DLR-ACD

DLR-ACD is the first aerial crowd image dataset [61] comprises 33 large aerial RGB images with average size of 3619 × 5226 pixels from different mass events and urban scenes containing crowds such as sports events, city centers, open-air fairs, and festivals. The GSDs of the images vary between 4.5 and 15 cm/pixel. In DLR-ACD 226,291 pedestrians have been manually labeled by point annotations, with the number of pedestrians ranging from 285 to 24,368 per image. In addition to its unique viewing angle, the large number of pedestrians in most of the images (>2 K) makes DLR-ACD stand out among the existing crowd datasets. Moreover, the crowd density can vary significantly within each image due to the large field of view of the images. Figure 5 demonstrates example images from the DLR-ACD dataset. For further details on this dataset, the interested reader is remanded to [61].

(**a**) (**b**)

Figure 5. Example images of the DLR-ACD dataset. The images are from an open-air (**a**) festival (**b**) and music concert.

4. Evaluation Metrics

In this section, we introduce the most important metrics we use for our quantitative evaluations. We adopted widely-used metrics in the MOT domain based on [25] which are listed in Table 4. In this table, ↑ and ↓ denote higher or lower values being better, respectively. The objective of MOT is finding the spatial positions of p objects as bounding boxes throughout an image sequence (object trajectories). Each bounding box is defined by the x and y coordinates of its top-left and bottom-right corners in each frame. Tracking performances are evaluated based on true positives (TP), correctly predicting the object positions, false positives (FP), predicting the position of another object instead of the target object's position, and false negatives (FN), where an object position is totally missed. In our experiments, a prediction (tracklet) is considered as TP if the intersection over union (IoU) of the predicted and the corresponding ground truth bounding boxes is greater than 0.5. Moreover, an identity switch (IDS) occurs if an annotated object a is associated with a tracklet t, and the assignment in the previous frame was $a \neq t$. The fragmentation metric shows the total number of times a trajectory is interrupted during tracking.

Table 4. Description of the metrics used for quantitative evaluations.

Metric		Description
IDF1	↑	ID F1-Score
IDP	↑	ID Global Min-Cost Precision
IDR	↑	ID Global Min-Cost Recall
Rcll	↑	Recall
Prcn	↑	Precision
FAR	↓	False Acceptance Rate
MT	↑	Ratio of Mostly Tracked Objects
PT	↑	Ratio of Partially Tracked Objects
ML	↓	Ratio of Mostly Lost Objects
FP	↓	False Positives
FN	↓	False Negatives
IDS	↓	Number of Identity Switches
FM	↓	Number of Fragmented Tracks
MOTA	↑	Multiple Object Tracker Accuracy
MOTP	↑	Multiple Object Tracker Precision
MOTAL	↑	Multiple Object Tracker Accuracy Log

Among these metrics, the crucial ones are the Multiple-Object Tracker Accuracy (MOTA) and the Multiple-Object Tracker Precision (MOTP). *MOTA* represents the ability of trackers in following the trajectories throughout the frames t, independently from the precision of the predictions:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t}. \quad (1)$$

The Multiple-Object Tracker Accuracy Log (MOTAL) is similar to *MOTA*; however, ID switches are considered on a logarithmic scale.

$$MOTAL = 1 - \frac{\sum FN_T + FP_t + log_{10}(IDS_t + 1)}{\sum GT_t}. \quad (2)$$

MOTP measures the performance of the trackers in precisely estimating object locations:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}, \quad (3)$$

where $d_{t,i}$ is the distance between a matched object i and the ground truth annotation in frame t, and c is the total number of matched objects.

Each tracklet can be considered as mostly tracked (MT), partially tracked (PT), or mostly lost (ML), based on how successful an object is tracked during its whole lifetime. A tracklet is mostly lost if it is only tracked less than 20% of its lifetime and mostly tracked if it is tracked more than 80% of its lifetime. Partially tracked applies to all remaining tracklets. We report MT, PT, and ML as percentages of the total amount of tracks. The false acceptance rate (*FAR*) for an image sequence with f frames describes the average amount of FPs per frame:

$$FAR = \frac{\sum FP_t}{f}. \tag{4}$$

In addition, we use recall and precision measures, defined as follows:

$$Rcll = \frac{\sum TP_t}{\sum(TP_t + FN_t)}, \tag{5}$$

$$Prcn = \frac{\sum TP_t}{\sum(TP_t + FP_t)}. \tag{6}$$

Identification precision (IDP), identification recall (IDR), and IDF1 are similar to precision and recall; however, they take into account how long the tracker correctly identifies the targets. IDP and IDR are the ratios of computed and ground-truth detections that are correctly identified, respectively. IDF1 is calculated as the ratio of correctly identified detections over the average number of computed and ground-truth detections. IDF1 allows ranking different trackers based on a single scalar value. For any further information on these metrics, the interested reader is remanded to [62].

5. Preliminary Experiments

This section empirically shows the existing challenges in aerial pedestrian tracking. We study the performance of a number of existing tracking methods including KCF [47], MOSSE [6], CSRT [51], Median Flow [50], SORT, DeepSORT [34], Stacked-DCFNet [32], Tracktor++ [1], SMSOT-CNN [22], and Euclidean Online Tracking on aerial data, and show their strengths and limitations. Since in the early phase of our research, only the KIT AIS pedestrian dataset was available to us, the experiments of this section have been conducted on this dataset. However, our findings also hold for the AerialMPT dataset.

The tracking performance is usually correlated to the detection accuracy for both detection-free and detection-based methods. As our main focus is at tracking performance, in most of our experiments we assume perfect detection results and use the ground truth data. While for the object locations in the first frame are given to the detection-free methods, the detection-based methods are provided with the object locations in every frame. Therefore, for the detection-based methods, the most substantial measure is the number of ID switches, while for the other methods all metrics are considered in our evaluations.

5.1. From Single- to Multi-Object Tracking

Many tracking methods have been initially designed to track only single objects. However, according to [22], most of them can be extended to handle MOT. Tracking management is an essential function in MOT which stores and exploits multiple active tracks at the same time, in order to remove and initialize the tracks of objects leaving from and entering into the scenes. For our experiments we developed a tracking management module for extending the SOT methods to MOT. It unites memory management, including the assignment of unique track IDs and individual object position storage, with track initialization, aging and removing functionalities.

OpenCV provides several built-in object tracking algorithms. Among them, we investigate the KCF, MOSSE, CSRT, and Median Flow SOT methods. We extend them to the MOT scenarios within the OpenCV framework. We initialize the trackers by the ground truth bounding box positions.

DCFNet [32] is also an SOT on a DCF. However, the DCF is implemented as part of a DNN and uses the features extracted by a light-weight CNN. Therefore, DCFNet is a perfect choice to study whether deep features improve the tracking performance compared to the handcrafted ones. For our experiments, we took the PyTorch implementation (https://github.com/foolwood/DCFNet_pytorch, accessed on 10 May 2021) of DCFNet and modified its network structure to handle multi-object tracking, and we refer to it as "Stacked-DCFNet". From the KIT AIS pedestrian training set we crop a total of 20,666 image patches centered at every pedestrian. The patch size is the bounding box size multiplied by 10 in order to consider contextual information to some degree. Then we scale the patches to 125×125 pixels to match the network input size. Using the patches, we retrain the convolutional layers of the network for 50 epochs with ADAM [63] optimizer, MSE loss, initial learning rate of 0.01, and a batch size of 64. Moreover, we set the spatial bandwidth to 0.1 for both online tracking and offline training. Furthermore, in order to adapt it to MOT, we use our developed Python module. Multiple targets are given to the network within one batch. For each target object, the network receives two image patches, from previous and current frames, centered on the known previous position of the object. The network output is the probability heatmap in which the highest value represents the most likely object location in the image patch of the current frame (search patch). If this value is below a certain threshold, we consider the object as lost. Furthermore, we propose a simple linear motion model and set the center point of the search patch to the position estimate of this model instead of the position of the object in the previous frame patch (as in the original work). Based on the latest movement $v_t(x, y)$ of a target, we estimate its position as:

$$p_{est}(x, y) = p(x, y) + k \cdot v_t(x, y), \tag{7}$$

where k determines the influence of the last movement. For all of the methods, we remove the objects if they leave the scene and their track ages are greater than 3 frames.

Tables 5 and 6 show the overall and sequence-wise tracking results of these methods on the KIT AIS pedestrian dataset, respectively. The results of Table 5 indicate the poor performance of all of these methods with a total MOTA scores varying between -85.8 and -55.9. The results of KCF and MOSSE are very similar. However, the use of HOG features and non-linear kernels in KCF improves MOTA by 0.9 and MOTP by 0.5 points respectively, compared to MOSSE. Moreover, both methods mostly track about 1% of the pedestrians in average. However, they have the first and second best MOTP values among the compared methods in Table 5. This indicates that although they lose track of many objects (partially or totally), their tracking localization is relatively precise. Moreover, according to the results, Stacked-DCFNet significantly outperforms the method with handcrafted features by a MOTA score of -37.3 (18.6 points higher than that of the CSRT). The MT and ML rates are also improving with only losing 23.6% of all tracks while mostly tracking the 13.8% of the pedestrians.

CSRT (which is also DCF-based) outperforms both prior methods significantly, reaching a total MOTA and MOTP of -55.9 and 78.4. The smaller MOTP value of CSRT indicates its slightly worse tracklet localization precision as compared to KCF and MOSSE. Furthermore, it mostly tracks about 10% of the pedestrians in average and proves the effectiveness of the channel and reliability scores. According to the table, Median Flow achieves comparable results to CSRT with total MOTA and MOTP scores of -63.8 and 77.7, respectively. Comparing the results of different sequences in Table 6 indicates that all algorithms perform significantly better on the "RaR_Snack_Zone_02" and "RaR_Snack_Zone_04" sequences. Based on visual inspection, we argue that this is due to their short length resulting in fewer lost objects and ID switches. Comparing their performances on the longer sequences ("AA_Crossing_02", "AA_Walking_02" and "Munich02") demonstrates that Stacked-DCFNet performs much better than the other methods on these sequences, showing the ability of the method in tracking objects for a longer time.

Altogether, according to the results, we argue that the deep features outperform the handcrafted ones by a large margin.

Table 5. Results of KCF, MOSSE, CSRT, Median Flow, and Stacked-DCFNet on the KIT AIS pedestrian dataset. The **first** and **second** best values are highlighted.

Methods	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KCF	9.0	8.8	9.3	10.3	9.8	165.6	1.1	53.8	45.1	11,426	10,782	32	**116**	−84.9	**87.2**	−84.7
MOSSE	9.1	8.9	9.3	10.5	10.0	163.8	0.8	54.0	45.2	11,303	10,765	**31**	**133**	−85.8	**86.7**	−83.5
CSRT	16.0	16.9	15.2	17.5	**19.4**	**126.5**	**9.6**	51.0	39.4	**8732**	9924	91	254	**−55.9**	78.4	**−55.1**
Median Flow	**18.5**	**18.3**	**18.8**	**19.5**	19.0	144.7	7.7	**55.8**	**36.5**	9986	**9678**	**30**	161	−63.8	77.7	−63.5
Stacked-DCFNet	**30.0**	**30.2**	**30.9**	**33.1**	**32.3**	**120.5**	**13.8**	**62.6**	**23.6**	**8316**	**8051**	139	651	**−37.3**	71.6	**−36.1**

Table 6. Results of KCF, MOSSE, CSRT, Median Flow, and Stacked-DCFNet on different sequences of KIT AIS pedestrian dataset. The **first** and **second** best values of each method on the sequences are highlighted.

Sequences	# Imgs	GT	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KCF																		
AA_Crossing_02	13	94	8.1	8.1	8.0	9.1	9.2	**78.1**	1.1	6.4	92.5	1015	1032	**0**	**8**	−80.4	**97.3**	−80.4
AA_Walking_02	17	188	6.5	6.3	6.7	7.8	7.3	154.9	**1.6**	10.6	**87.8**	2633	2463	**3**	14	−90.9	96.9	−90.8
Munich02	31	230	4.3	4.1	4.4	5.6	5.2	201.7	0.9	3.9	95.2	6254	5781	29	75	−97.0	62.2	−96.5
RaR_Snack_Zone_02	4	220	**29.3**	**29.1**	**29.5**	**29.8**	**29.5**	**154.5**	**1.8**	**98.2**	**0.0**	**618**	**607**	**0**	**8**	**−41.6**	95.1	**−41.6**
RaR_Snack_Zone_04	4	311	**25.8**	**25.7**	**25.9**	**26.9**	**26.8**	226.5	0.3	**99.7**	**0.0**	**906**	**899**	**0**	**11**	**−46.7**	**97.9**	**−46.7**
MOSSE																		
AA_Crossing_02	13	94	8.0	8.1	7.9	9.1	9.2	**78.1**	**1.1**	5.3	93.6	1015	1032	**0**	**9**	−80.4	**96.9**	-80.4
AA_Walking_02	17	188	6.6	6.4	6.7	8.0	7.6	**151.8**	**1.6**	10.1	**88.3**	2580	2458	**2**	20	−88.7	95.7	−88.6
Munich02	31	230	4.3	4.2	4.5	5.7	5.4	199.7	0.9	4.3	94.8	6190	5775	29	78	−95.8	61.9	−95.4
RaR_Snack_Zone_02	4	220	**29.4**	**29.2**	**29.6**	**30.4**	**30.0**	153.2	0.5	**99.5**	**0.0**	**613**	**602**	**0**	14	**−40.5**	94.9	**−40.5**
RaR_Snack_Zone_04	4	311	**25.8**	**25.7**	**25.9**	**27.0**	**26.8**	226.2	0.3	**99.7**	**0.0**	**905**	**898**	**0**	**12**	**−46.6**	**97.5**	**−46.6**
CSRT																		
AA_Crossing_02	13	94	12.9	13.2	12.5	15.1	15.9	**69.5**	1.1	30.9	**68.0**	904	964	10	**29**	−65.5	84.6	−64.7
AA_Walking_02	17	188	9.2	10.0	8.5	11	12.9	**116.9**	2.7	15.4	81.9	187	2378	12	41	−63.9	**88.0**	−63.5
Munich02	31	230	9.2	9.9	8.7	10.9	12.5	151.4	1.8	14.3	83.9	4696	5455	66	137	−66.8	61.2	−65.8
RaR_Snack_Zone_02	4	220	**43.2**	**42.0**	**42.5**	**43.8**	**43.3**	124.2	**17.3**	82.7	**0.0**	**497**	**486**	**0**	**16**	**−13.6**	**87.9**	**−13.6**
RaR_Snack_Zone_04	4	311	**45.6**	**45.5**	**45.0**	**47.9**	**47.6**	162.0	**16.7**	83.3	**0.0**	**648**	**641**	**3**	31	**−5.0**	85.2	**−4.8**
Median Flow																		
AA_Crossing_02	13	94	27.3	27.3	27.4	28.5	28.3	**62.8**	1.1	68.1	**30.8**	817	812	**4**	49	−43.9	74.9	−43.6
AA_Walking_02	17	188	10.0	9.9	10.0	11.1	11.0	141.1	1.6	21.3	77.1	2398	2374	8	**16**	−79.0	**86.3**	−78.7
Munich02	31	230	9.2	9.0	9.4	9.9	9.5	186.4	1.3	8.7	90.0	5778	5517	10	53	−84.6	64.7	−84.4
RaR_Snack_Zone_02	4	220	**51.7**	**51.4**	**52.0**	**52.8**	**52.2**	**104.7**	**8.6**	**91.4**	**0.0**	**419**	**408**	**2**	**14**	**4.2**	**83.7**	**4.3**
RaR_Snack_Zone_04	4	311	**53.1**	**53.0**	**53.3**	**53.9**	**53.6**	143.5	**17.4**	**82.6**	**0.0**	**574**	**567**	6	29	**6.7**	83.0	**7.2**
Stacked-DCFNet																		
AA_Crossing_02	13	94	**41.9**	42.4	41.3	42.7	43.9	**47.8**	12.8	58.5	**28.7**	621	650	15	71	−13.3	74.7	-12.1
AA_Walking_02	17	188	31.4	31.6	31.2	32.3	32.7	104.3	5.9	45.7	48.4	1773	1809	23	184	−35.0	74.1	−34.2
Munich02	31	230	21.2	20.6	21.9	25.0	23.6	160.4	1.7	50.0	48.3	4974	4591	97	322	−57.7	60.5	−56.2
RaR_Snack_Zone_02	4	220	**51.8**	**52.3**	**51.3**	**52.4**	**53.4**	**99.0**	**22.3**	**74.5**	**3.2**	**396**	**412**	**4**	**35**	**6.1**	**84.0**	**6.5**
RaR_Snack_Zone_04	4	311	**51.8**	**52.6**	**51.0**	**52.1**	**53.7**	138.0	**21.9**	**74.9**	**3.2**	**552**	**589**	**0**	**39**	**7.2**	**83.6**	**7.2**

5.2. Multi-Object Trackers

In this section, we study a number of MOT methods including SORT, DeepSORT, and Tracktor++. Additionally, we propose a new tracking algorithm called Euclidean Online Tracking (EOT) which uses the Euclidean distance for object matching.

5.2.1. DeepSORT and SORT

DeepSORT [34] is a MOT method comprising deep features and an IoU-based tracking strategy. For our experiments, we use the PyTorch implementation (https://github.com/ZQPei/deep_sort_pytorch, accessed on 10 May 2021) of DeepSORT and adapt it for the KIT AIS dataset by changing the bounding box size and IoU threshold, as well as fine-tuning the network on the training set of the KIT AIS dataset. As mentioned, for the object locations we use the ground truth and do not use the DeepSORT's object detector. Tables 7 and 8 show the tracking results of our experiments in which Rcll, Prcn, FAR, MT, PT, ML, FN, FM, and MOTP are not important in our evaluations as the ground truth is used instead of the detection results. Therefore, the best values for these metrics are not highlighted for non of the methods in Table 7 and for DeepSORTs and SORTs in Table 8.

Table 7. Results of DeepSORT, SORT, Tracktor++, and SMSOT-CNN on the KIT AIS pedestrian dataset. The **first** and **second** best values are highlighted.

Methods	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
DeepSORT	10.0	9.8	10.2	100.0	95.8	7.6	100.0	0.0	0.0	523	0	8627	9	23.9	81.1	**98.6**
DeepSORT-BBX$_{2\times}$	38.4	36.9	39.9	100.0	92.6	13.9	100.0	0.0	0.0	958	0	5073	9	49.9	78.7	92.0
DeepSORT-IoU$_{99}$	43.3	40.8	44.0	98.3	91.1	16.7	99.8	0.2	0.0	1152	205	4009	189	55.4	73.7	88.7
DeepSORT-BBX$_{2\times}$-IoU$_{99}$	82.1	80.7	83.6	99.4	96.0	7.3	99.8	0.2	0.0	502	75	738	70	89.1	74.7	95.2
DeepSORT-BBX$_{2\times}$-IoU$_{99}$-FT	82.4	81.0	83.8	99.4	96.0	7.1	99.8	0.2	0.0	**493**	71	734	68	**89.2**	74.7	95.3
SORT-IoU$_{99}$	42.9	41.8	44.2	98.7	93.4	12.2	99.8	0.2	0.0	840	151	3805	141	60.1	73.6	91.7
SORT-BBX$_{2\times}$-IoU$_{99}$	**86.5**	**85.5**	**87.2**	99.6	98.1	3.3	99.8	0.2	0.0	**231**	46	438	48	**94.1**	74.7	**97.7**
Tracktor++	13.7	27.3	9.2	28.5	85.0	–	13.2	44.2	42.6	604	8593	2188	725	5.3	0.1	–
SMSOT-CNN	34.0	33.2	34.9	38.2	36.4	116.4	25.0	52.5	22.5	8028	7427	**157**	614	−29.8	71.0	−28.5
EOT-D$_{17}$	**85.2**	**84.9**	**85.5**	86.5	86.0	24.5	80.2	19.6	0.2	1692	1619	37	1074	72.2	69.3	72.5

Table 8. Results of DeepSORT, SORT, Tracktor++, and SMSOT-CNN on the KIT AIS pedestrian dataset. The **first** and **second** best values of each method on the sequences are highlighted.

Sequences	# Imgs	GT	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
DeepSORT																		
AA_Crossing_02	13	94	3.1	3.1	3.1	100.0	100.0	0.0	100.0	0.0	0.0	**0**	0	940	1	17.2	99.7	**99.7**
AA_Walking_02	17	188	7.7	7.7	7.8	100.0	98.9	1.7	100.0	0.0	0.0	29	0	2145	5	18.6	99.0	98.8
Munich02	31	230	9.1	8.8	9.4	100.0	92.8	15.4	100.0	0.0	0.0	478	0	4681	1	15.8	64.0	92.1
RaR_Snack_Zone_02	4	220	**21.0**	**20.9**	**21.2**	100.0	98.7	2.7	100.0	0.0	0.0	11	0	**351**	2	**58.2**	98.1	98.4
RaR_Snack_Zone_04	4	311	**17.9**	**17.9**	**18.0**	100.0	99.6	1.2	100.0	0.0	0.0	**5**	0	**510**	0	**58.1**	98.6	**99.4**
DeepSORT-BBX$_{2\times}$																		
AA_Crossing_02	13	94	34.8	34.5	35.1	100.0	98.4	1.4	100.0	0.0	0.0	**18**	0	566	1	48.5	94.3	98.2
AA_Walking_02	17	188	46.6	46.0	47.1	100.0	98.8	3.6	100.0	0.0	0.0	61	0	1073	5	57.5	93.1	97.6
Munich02	31	230	29.5	27.6	31.5	100.0	87.7	27.7	100.0	0.0	0.0	859	0	2989	1	37.2	63.9	85.9
RaR_Snack_Zone_02	4	220	**52.2**	**51.9**	**52.5**	100.0	98.9	2.5	100.0	0.0	0.0	**10**	0	**203**	2	**75.4**	95.7	**98.6**
RaR_Snack_Zone_04	4	311	**61.2**	**61.0**	**61.5**	100.0	99.2	2.5	100.0	0.0	0.0	**10**	0	**242**	0	**79.5**	94.4	**99.0**
DeepSORT-IoU$_{99}$																		
AA_Crossing_02	13	94	55.0	54.4	55.6	99.0	96.9	2.8	100.0	0.0	0.0	36	11	347	10	65.3	83.6	95.6
AA_Walking_02	17	188	63.4	62.5	64.3	99.1	96.3	6.1	100.0	0.0	0.0	103	23	557	25	74.4	82.0	95.2
Munich02	31	230	24.2	22.8	25.8	97.2	85.8	31.8	99.6	0.4	0.0	985	170	2737	151	36.5	62.9	81.1
RaR_Snack_Zone_02	4	220	**57.7**	**57.3**	**58.2**	100.0	98.5	3.2	100.0	0.0	0.0	**13**	0	**177**	2	**78.0**	90.4	**98.2**
RaR_Snack_Zone_04	4	311	**69.1**	**68.7**	**69.5**	99.9	98.8	3.7	99.7	0.3	0.0	**15**	1	**191**	1	**83.2**	87.2	**98.5**
DeepSORT-BBX$_{2\times}$-IoU$_{99}$																		
AA_Crossing_02	13	94	**93.8**	**92.5**	**95.2**	99.8	96.9	2.8	100.0	0.0	0.0	36	2	45	2	93.8	85.0	96.5
AA_Walking_02	17	188	88.7	84.4	93.4	99.7	90.0	17.3	100.0	0.0	0.0	295	8	**42**	12	87.0	86.4	88.6
Munich02	31	230	73.1	70.9	75.3	98.9	93.2	14.2	100.0	0.0	0.0	441	67	565	56	82.5	62.9	91.7
RaR_Snack_Zone_02	4	220	90.1	89.9	**90.4**	99.8	99.2	1.7	99.1	0.9	0.0	**7**	2	**37**	4	**94.7**	87.9	**98.8**
RaR_Snack_Zone_04	4	311	**90.2**	**90.1**	90.3	100.0	99.8	0.7	100.0	0.0	0.0	**3**	0	49	0	**95.8**	88.4	**99.6**
DeepSORT-BBX$_{2\times}$-IoU$_{99}$-FT																		
AA_Crossing_02	13	94	**93.1**	**92.7**	**93.4**	100.0	99.3	0.6	100.0	0.0	0.0	8	0	43	1	95.5	85.1	**99.2**
AA_Walking_02	17	188	**93.1**	**92.4**	**93.7**	99.8	98.4	2.5	100.0	0.0	0.0	43	6	**42**	9	**96.6**	86.5	98.1
Munich02	31	230	73.3	71.2	75.5	99.0	93.3	13.9	100.0	0.0	0.0	432	63	563	54	82.7	62.9	91.9
RaR_Snack_Zone_02	4	220	90.1	89.9	90.4	99.8	99.2	1.7	99.1	0.9	0.0	**7**	2	**37**	4	94.7	87.9	98.8
RaR_Snack_Zone_04	4	311	**90.2**	90.1	90.3	100.0	99.8	0.7	100.0	0.0	0.0	**3**	0	49	0	**95.8**	88.4	**99.6**
SORT-IoU$_{99}$																		
AA_Crossing_02	13	94	55.9	55.4	56.5	99.1	97.2	5.5	100.0	0.0	0.0	33	10	343	9	66.0	83.5	96.0
AA_Walking_02	17	188	**64.0**	**63.2**	**64.9**	99.3	96.7	5.3	100.0	0.0	0.0	90	19	550	21	75.3	82.0	95.8
Munich02	31	230	24.6	23.6	25.8	98.0	89.7	22.2	99.6	0.4	0.0	689	122	2544	108	45.2	62.8	86.7
RaR_Snack_Zone_02	4	220	57.7	57.3	58.2	100.0	98.5	3.2	100.0	0.0	0.0	**13**	0	**177**	2	**78.0**	90.4	**98.2**
RaR_Snack_Zone_04	4	311	**69.1**	**68.7**	**69.5**	99.9	98.8	3.7	99.7	0.3	0.0	**15**	1	**191**	1	**83.2**	87.2	**98.5**
SORT-BBX$_{2\times}$-IoU$_{99}$																		
AA_Crossing_02	13	94	**93.1**	**92.7**	**93.4**	100.0	99.3	0.6	100.0	0.0	0.0	8	0	45	1	95.3	85.0	**99.1**
AA_Walking_02	17	188	**94.5**	**93.9**	**95.1**	99.3	98.6	2.2	100.0	0.0	0.0	37	2	**30**	6	**97.4**	86.5	98.5
Munich02	31	230	80.4	79.6	81.3	99.3	97.2	5.7	100.0	0.0	0.0	176	42	284	37	91.8	63.0	96.4
RaR_Snack_Zone_02	4	220	90.5	90.2	90.8	99.8	99.2	1.7	99.1	0.9	0.0	**7**	2	**34**	4	95.0	87.9	98.8
RaR_Snack_Zone_04	4	311	90.5	90.4	90.7	100.0	99.8	0.7	100.0	0.0	0.0	**3**	0	45	0	**96.1**	88.4	**99.6**
Tracktor++																		
AA_Crossing_02	13	94	12.7	19.6	9.4	**48.2**	**100.0**	–	**20.1**	**51.1**	28.8	**0**	**588**	432	107	10.1	**0.13**	–
AA_Walking_02	17	188	10.7	27.5	6.7	23.2	**95.8**	–	3.2	43.1	**53.7**	**27**	2050	426	154	6.3	**0.13**	–
Munich02	31	230	7.8	16.7	5.1	22.7	74.5	–	2.2	41.3	**56.6**	746	4736	965	412	−0.8	0.08	–
RaR_Snack_Zone_02	4	220	**33.8**	**54.5**	**24.5**	40.2	89.5	–	17.7	**45.5**	36.8	41	**517**	**134**	**27**	**20.0**	**0.09**	–
RaR_Snack_Zone_04	4	311	**32.5**	**50.2**	**24.0**	**42.9**	89.8	–	**22.2**	44.1	33.7	60	702	**231**	**25**	**19.3**	0.06	–
SMSOT-CNN																		
AA_Crossing_02	13	94	49.9	49.7	50.1	52.1	51.6	**42.6**	24.5	52.1	23.4	554	544	11	71	2.3	68.8	3.2
AA_Walking_02	17	188	30.7	30.2	31.3	33.8	32.7	109.6	15.5	38.9	45.6	1864	1767	34	140	−32.7	68.0	−36.0
Munich02	31	230	23.6	22.7	24.5	28.8	26.7	156.3	8.6	38.3	53.1	4846	4363	105	316	−52.1	68.4	−50.4
RaR_Snack_Zone_02	4	220	**61.6**	**61.4**	**61.8**	**64.4**	**63.9**	**78.5**	**37.3**	**62.3**	**0.4**	**314**	**308**	**2**	**39**	**27.9**	**77.9**	**28.0**
RaR_Snack_Zone_04	4	311	**61.2**	**61.1**	**61.3**	**63.8**	**63.6**	112.5	**34.4**	**64.6**	**1.0**	**450**	**445**	**5**	**48**	**26.8**	**76.7**	**27.2**
EOT-D$_{17}$																		
AA_Crossing_02	13	94	94.4	94.4	94.4	95.3	95.2	**4.1**	**91.5**	8.5	0.0	54	53	**4**	34	90.2	73.8	90.5
AA_Walking_02	17	188	94.6	94.0	**95.1**	**96.9**	**95.8**	**6.7**	**96.8**	2.7	0.5	114	82	10	63	**92.3**	76.6	92.6
Munich02	31	230	76.0	75.8	76.2	77.0	76.5	46.6	44.3	**54.8**	0.9	1446	1409	15	930	53.1	60.4	53.4
RaR_Snack_Zone_02	4	220	**95.0**	**94.9**	**95.1**	**96.5**	**96.3**	8.0	87.7	12.3	0.0	**32**	**30**	**3**	**16**	**92.5**	**77.6**	**92.8**
RaR_Snack_Zone_04	4	311	**95.2**	**95.1**	**95.2**	96.3	**96.3**	11.5	76.2	**23.8**	0.0	**46**	**45**	5	**31**	92.2	**78.6**	**92.5**

In the first experiment, we employ DeepSORT with its original parameter settings. As the results show, this configuration is not suitable for tracking small objects (pedestrians) in aerial imagery. DeepSORT utilizes deep appearance features to associate objects to tracklets; however, for the first few frames, it relies on IoU metric until enough appearance features are available. The original IoU threshold is 0.5. The standard DeepSORT uses a Kalman filter for each object to estimate its position in the next frame. However, due to small IoU overlaps between most predictions and detections, many tracks can not be associated with any detection, making it impossible to use the deep features afterwards. The main cause of minor overlaps is the small size of the bounding boxes. For example, if the Kalman filter estimates the object position only 2 pixels off the detection's position, for a bounding box of 4×4 pixels, the overlap would be below the threshold and, consequently, the tracklet and the object cannot be matched. These mismatches result in a large number of falsely initiated new tracks, leading to a total amount of 8627 ID switches, an average amount of 8.27 ID switches per person, and an average amount of 0.71 ID switches per detection.

We tackle this problem by enlarging the bounding boxes by a factor of two in order to increase the IoU overlaps, increase the number of matched tracklets and detections, and enable the use of appearance features. According to Table 7, this configuration (DeepSORT-BBX$_{2\times}$) results in a 41.19% decrease in the total number of ID switches (from 8627 to 5073), a 56.38% decrease in the average number of ID switches per person (from 8.62 to 4.86), and a 59.15% decrease in the average number of ID switches per detection (from 0.71 to 0.42). We further analyze the impact of using different IoU thresholds on the tracking performance. Figure 6 illustrates the number of ID switches with different IoU thresholds. It can be observed that by increasing the threshold (minimizing the required

overlap for object matching) the number of ID switches reduces. The least number of ID switches (738 switches) is achieved by the IoU threshold of 0.99, as can be seen in Table 7 for DeepSORT-IoU$_{99}$. Based on the results, enlarging the bounding boxes and changing the IoU threshold significantly improves the tracking results of DeepSORT-BBX$_{2\times}$-IoU$_{99}$ as compared to the original settings of DeepSORT (ID switches by 91.44% and MOTA by 3.7 times). This confirms that the missing IoU overlap is the main issue with the standard DeepSORT.

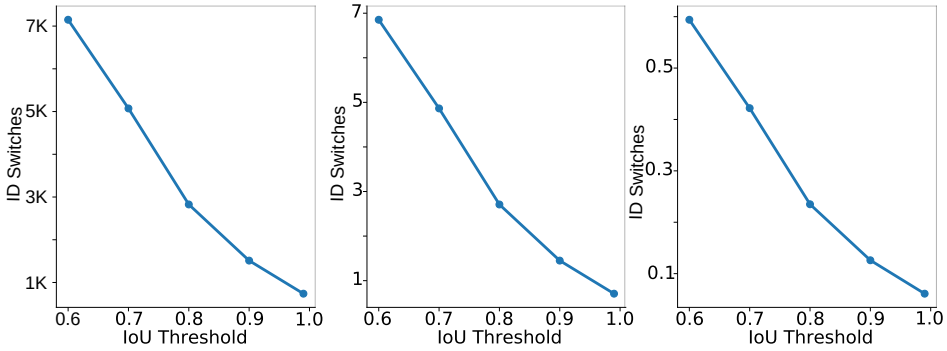


Figure 6. ID Switches versus IoU thresholds in DeepSORT. From left to right: total, average per person, and average per detection ID Switches.

After adapting the IoU object matching, the deep appearance features play a prominent role in the object tracking after the first few frames. Thus, a fine-tuning of the DeepSORT's neural network on the training set of the KIT AIS pedestrian dataset can further improve the results (DeepSORT-BBX$_{2\times}$-IoU$_{99}$-FT). Originally, the network has been trained on a large person re-identification dataset, which is very different from our scenario, especially in the looking angle and the object sizes, as the bounding boxes in aerial images are much smaller than in the person re-identification dataset (4×4 vs. 128×64 pixels). Scaling the bounding boxes of our aerial dataset to fit the network input size leads to relevant interpolation errors. For our experiments we initialize the last re-identification layers from scratch, and the rest of the network using the pre-trained weights and biases. We also changed the number of classes to 610, representing the number of different pedestrians after cropping the images into the patches with the size of the bounding boxes, and ignoring the patches located at the image border. Instead of scaling the patches to 128×64 pixels, we only scale them to 50×50. We trained the classifier for 20 epochs with SGD optimizer, Cross-Entropy loss function, batch size of 128, and an initial learning rate of 0.01. Moreover, we doubled the bounding box sizes for our experiment. The results in Table 7 show that the total number of ID switches only decreases from 738 to 734. This indicates that the deep appearance features of DeepSORT are not useful for our problem. While for a large object a small deviation of the bounding box position is tolerable (as the bounding box still mostly contains object-relevant areas), for our very small objects this can cause significant changes in object relevance. The extracted features mostly contain background information. Consequently, in the appearance matching step, the object features from its previous and currently estimated positions can differ significantly. Additionally, the appearance features of different pedestrians in aerial images are often not discriminative enough to distinguish them.

In order to better demonstrate this effect, we evaluate DeepSORT without any appearance feature, also known as SORT. Table 7 shows the tracking results with original and doubled bounding box sizes and an IoU threshold of 0.99. According to the results, SORT outperforms the fine-tuned DeepSORT with 438 ID switches. Nevertheless, the number of ID switches is still high, given that we use the ground truth object positions. This could be due to the low frame rate of the dataset and the small sizes of the the objects. Although enlarging the bounding boxes improved the performance significantly (60% and 56% better MOTA for DeepSORT and SORT, respectively), it leads to a poor localization accuracy.

5.2.2. Tracktor++

Tracktor++ [1] is an MOT method based on deep features. It employs a Faster-RCNN to perform object detection and tracking through regression. We use its PyTorch implementation (https://github.com/phil-bergmann/tracking_wo_bnw, accessed on 10 May 2021) and adapt it to our aerial dataset. We tested Tracktor++ with the ground truth object positions instead of using its detection module; however, it totally failed the tracking task with these settings. Faster-RCNN has been trained on the datasets which are very different to our aerial dataset, for example in looking angle, number and size of the objects. Therefore, we fine-tune Faster-RCNN on the KIT AIS dataset. To this end, we had to adjust the training procedure to the specification of our dataset.

We use Faster-RCNN with a ResNet50 backbone, pre-trained on the ImageNet dataset. We change the anchor sizes to {2, 3, 4, 5, 6} and the aspect ratios to {0.7, 1.0, 1.3}, enabling it to detect small objects. Additionally, we increase the maximum detections per image to 300, set the minimum size of an image to be rescaled to 400 pixels, the region proposal non-maximum suppression (NMS) threshold to 0.3, and the box predictor NMS threshold to 0.1. The NMS thresholds influence the amount of overlap for region proposals and box predictions. Instead of SGD, we use an ADAM optimizer with an initial learning rate of 0.0001 and a weight decay of 0.0005. Moreover, we decrease the learning rate every 40 epochs by a factor of 10 and set the number of classes to 2, corresponding to background and pedestrians. We also apply substantial online data augmentation including random flipping of every second image horizontally and vertically, color jitter, and random scaling in a range of 10%.

The tracking results of Tracktor++ with the fine-tuned Faster-RCNN are presented in Table 7. The detection precision and recall of Faster-RCNN are 25% and 31%, respectively, with this poor detection performance potentially propagated to the tracking part. According to the table, Tracktor++ only achieves an overall MOTA of 5.3 and 2188 ID switches even when we use ground truth object positions. We conclude by assuming that Tracktor++ has difficulties with the low frame rate of the dataset and the small object sizes.

5.2.3. SMSOT-CNN

SMSOT-CNN [22] is the first DL-based method for multi-object tracking in aerial imagery. It is an extension to GOTURN [17], an SOT regression-based method using CNNs to track generic objects at high speed. SMSOT-CNN adapts GOTURN for MOT scenarios by three additional convolution layers and a tacking management module. The network receives two image patches from the previous and current frames, where both are centered at the object position in the previous frame. The size of the image patches (the amount of contextual information) is adjusted by a hyperparameter. The network regresses the object position in the coordinates of the current frame's image patch. SMSOT-CNN has been evaluated on the KIT AIS pedestrian dataset in [22], where the objects' first positions are given based on the ground truth data. The tracking results can be seen in Table 7. Due to the use of a deep network and the local search for the next position of the objects, the number of ID switches by SMSOT-CNN is 157, which is small, relative to the other methods. Moreover, this algorithm achieves an overall MOTA and MOTP of -29.8 and 71.0, respectively. Based on our visual inspections, SMSOT-CNN has some difficulties in densely crowded situations where the objects share similar appearance features. In these cases, multiple similarly looking objects can be present in an image patch, resulting in ID switches and losing track of the target objects. Furthermore, the small sizes of the pedestrians make them similar to many non-pedestrian objects in the feature space causing a large number of FPs and FNs.

5.2.4. Euclidean Online Tracking

Inspired by the tracking results of SORT besides its simplicity, we propose EOT based on the architecture of SORT for pedestrian tracking in aerial imagery. EOT uses a Kalman filter similarly to SORT. Then it calculates the euclidean distance between all predictions

(x_i, y_i) and detections (x_j, y_j), and normalizes them w.r.t. the GSD of the frame to construct a cost matrix as follows:

$$D_{i,j} = GSD \cdot \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (8)$$

After that, as in SORT, we use the Hungarian algorithm to look for global minima. However, if objects enter or leave the scene, the Hungarian algorithm can propagate an error to the whole prediction-detection matching procedure: therefore, we constrain the cost matrix so that all distances greater than a certain threshold are ignored and set to an infinity cost. We empirically set the threshold to $17.GSD$ pixels. Furthermore, only objects successfully tracked in the previous frame are considered for the matching process. According to Table 7, while the total MOTA score is competitive with the previously studied methods, EOT achieves the least ID switches (only 37). Compared to SORT, as EOT keeps better track of the objects, the deviations in the Kalman filter predictions are smaller. Therefore, Euclidean distance is a better option as compared to IoU for our aerial image sequences.

5.3. Conclusion of the Experiments

In this section, we conclude our preliminary study. According to the results, our EOT is the best performing tracking method. Figure 7 illustrates a major case of success by our EOT method. We can observe that almost all pedestrians are tracked successfully, even though the sequence is crowded and people walk in different directions. Furthermore, the significant cases of false positives and negatives are caused by the limitation of the evaluation approach. In other words, while EOT tracks most of the objects, since the evaluation approach is constrained to the minimum 50% overlap (4 pixels), the correctly tracked objects with smaller overlaps are not considered.

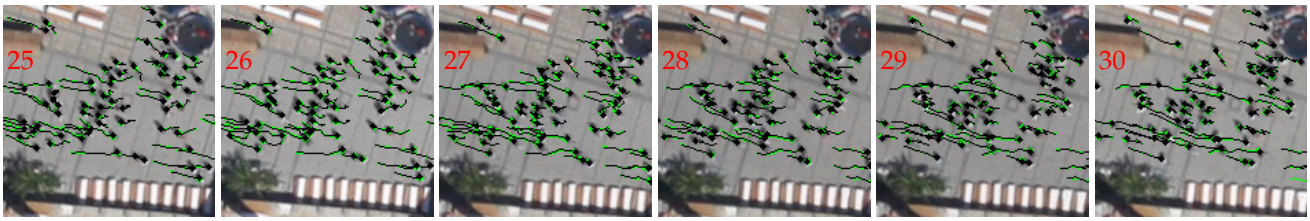


Figure 7. A success case processed by Stacked-DCFNet on the sequence "Munich02". The tracking results and ground truth are depicted in green and black, respectively.

Figure 8 shows a typical failure case of the Stacked-DCFNet method. In the first two frames, most of the objects are tracked correctly; however, after that, the diagonal line in the patch center is confused with the people walking across it. We assume that the line shares similar appearance features with the crossing people. Figure 9 demonstrates a successful tracking case by Stacked-DCFNet. People are not walking closely together and the background is more distinguishable from the people. Figure 10 illustrates another typical failure case of DCFNet. The image includes several people walking closely in different directions, introducing confusion into the tracking method due to the people's similar appearance features. We closely investigate these failure cases in Figure 11. In this figure, we visualize the activation map of the last convolution layer of the network. Although the convolutional layers of Stacked-DCFNet are supposed to be trained only for people, the line and the people (considering their shadows) appear indistinguishable. Moreover, based on the features, different people cannot be discriminated. We also evaluated SMSOT-CNN and found that it shares similar failure and success cases with Stacked-DCFNet, as both take advantage of convolutional layers for extracting appearance features.

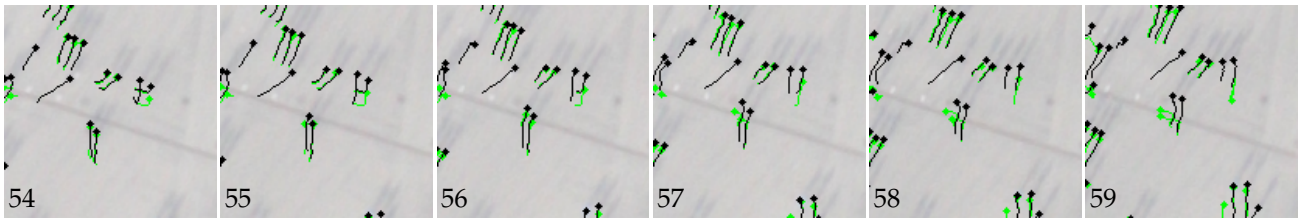


Figure 8. A failure case by Stacked-DCFNet on the sequence "AA_Walking_02". The tracking results and ground truth are depicted in green and black, respectively.

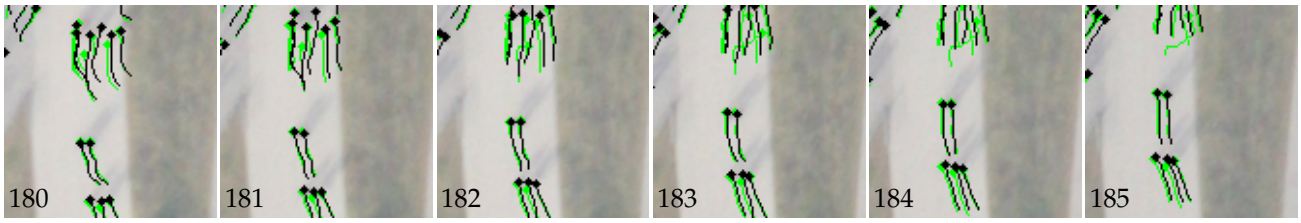


Figure 9. A success case by Stacked-DCFNet on the sequence "AA_Crossing_02". The tracking results and ground truth are depicted in green and black, respectively.

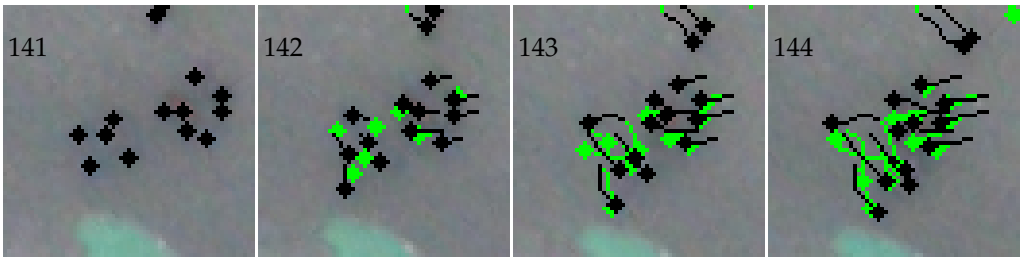


Figure 10. A failure case by Stacked-DCFNet on the test sequence "RaR_Snack_Zone_04". The tracking results and the ground truth are depicted in green and black, respectively.

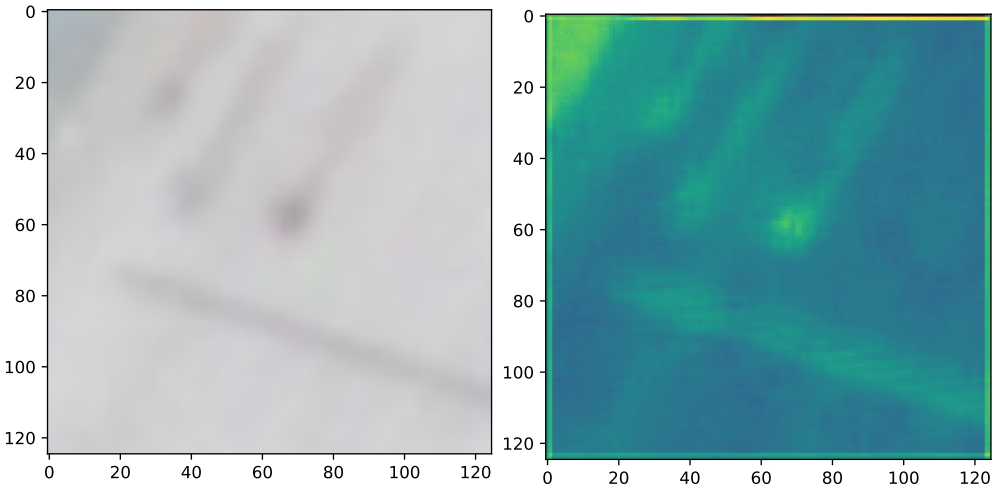


Figure 11. (**a**) An input image patch to the last convolutional layer of Stacked-DCFNetand and (**b**) its corresponding activation map.

Altogether, the Euclidean distance paired with trajectory information in EOT works better than IoU for tracking in aerial imagery. However, detection-based trackers such as EOT require object detection in every frame. As shown for Tracktor++, the detection accuracy of the object detectors is very poor for pedestrians in aerial images. Thus, detection-based methods are not appropriate for our scenarios. Moreover, the approaches

which employ deep appearance features for re-identification share similar problems with object detectors, features with poor discrimination abilities in the presence of similarly looking objects, leading to ID switches and loosing track of objects. The tracking methods based on regression and correlation (e.g., Stacked-DCFNet and SMSOT-CNN) show, in general, better performances than the methods based on re-identification because they track objects by local image patches that errors to be propagated to the whole image. Furthermore, according to our investigations, the path taken by every pedestrian is influenced by three factors: (1) the pedestrian's path history, (2) the positions and movements of the surrounding people, (3) the arrangement of the scene.

We conclude that both regression- and correlation-based tracking methods are good choices for our scenario. They can be improved by considering trajectory information and the pedestrians movement relationships.

6. AerialMPTNet

In this section we explain our proposed AerialMPTNet tracking algorithm with its different configurations. Part of its architecture and configurations has been presented in [26].

As stated in Section 5, a pedestrian's movement trajectory is influenced by its movement history, its motion relationships to its neighbours, and scene arrangements. The same holds for the vehicles in traffic scenarios. For the vehicles, there are other constraints such as moving along predetermined paths (e.g., streets, highways, railways) in most of the time. Different objects have different motion characteristics such as speed and acceleration. For example, several studies have shown that walking speed of pedestrians are strongly influenced by their age, gender, temporal variations as well as distractions (e.g., cell phone usage), whether the individual is moving in a group or not, and even the size of the city where the event takes place [64,65]. Regarding road traffic, similar factors could influence driving behaviors and movement characteristics (e.g., cell phone usage, age, stress level, and fatigue) [66,67]. Furthermore, similar to the pedestrians, maneuvers of a vehicle can directly affect the movements of other neighbouring vehicles: for example, if the vehicle brakes, all the following vehicles must brake, too.

The understanding of individual motion patterns is crucial for tracking algorithms, especially when only limited visual information about target objects is available. However, current regression-based tracking methods such as GOTURN and SMSOT-CNN do not incorporate movement histories or relationships between adjacent objects. These networks locate the next position of objects by monitoring a search area in their immediate proximity. Thus, the contextual information provided to the network is limited. Additionally, during the training phase, the networks do not learn how to differentiate the targets from similarly looking objects within the search area. Thus, as discussed in Section 5, ID switches and losing of object tracks happen often for these networks in crowded situations or by object intersections.

In order to tackle the limitations of previous works we propose to fuse visual features, track history, and the movement relationships of adjacent objects in an end-to-end fashion within a regression-based DNN, which we refer to as AerialMPTNet. Figure 12 shows an overview of the network architecture. AerialMPTNet takes advantage of a Siamese Neural Network (SNN) for visual features, a Long Short-Term Memory (LSTM) module for movement histories, and a GraphCNN for movement relationships. The network takes two local image patches cropped from two consecutive images (previous and current), called target and search patch in which the object location is known and has to be predicted, respectively. Both patches are centered at the object coordinates known from the previous frame. Their size (the degree of contextual information) is correlated with the size of the objects, and it is set to 227×227 pixels to be compatible to the network's input. Both patches are then given to the SNN module (retained from [22]) composed of two branches of five 2D convolutional, two local response normalization, and three max-pooling layers with shared weights. Afterwards, the two output features Out_{SNN} are concatenated and

given to three 2D convolutional layers and, finally, four fully connected layers regressing the object position in the search patch coordinates. We use ReLU activations for all these convolutional layers.

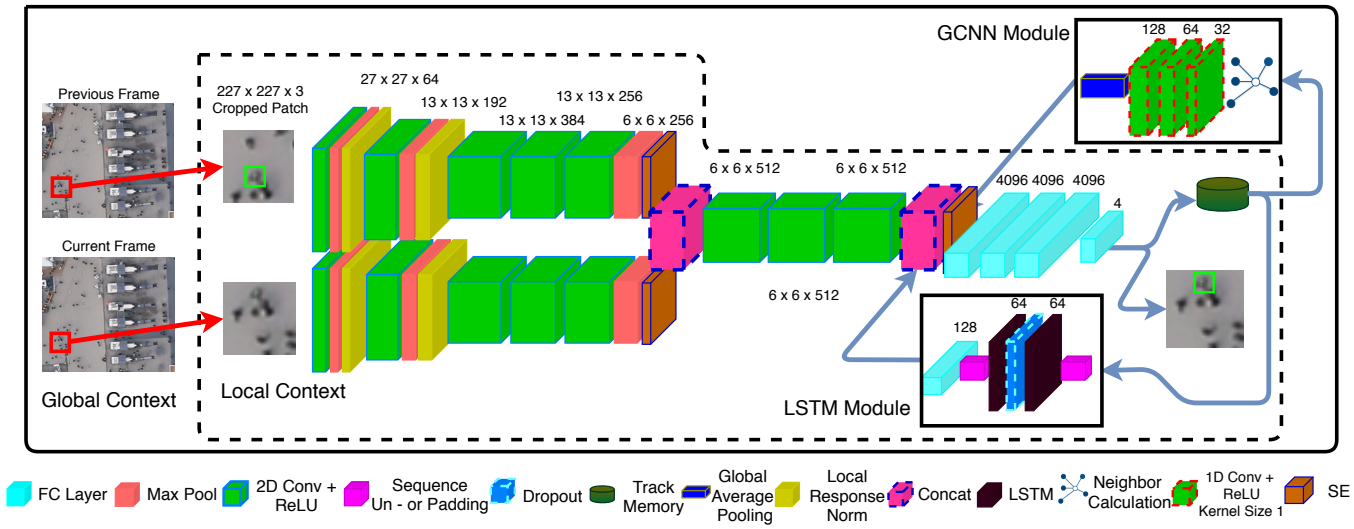


Figure 12. Overview of the network's architecture composing a SNN, a LSTM and a GraphCNN module. The inputs are two consecutive images cropped and centered to a target object, while the output is the object location in search crop coordinates.

The network output is a vector of four values indicating the x and y coordinates of the top-left and bottom-right corners of the objects' bounding boxes. These coordinates are then transformed into image coordinates. In our network, the LSTM module and the GraphCNN module use the object coordinates in the search patch and image domain, respectively.

6.1. Long Short-Term Memory Module

In order to encode movement histories and predict object trajectories, recent works mainly relied on LSTM- and RNN-based structures [68–70]. While these structures have been mostly used for individual objects, due to the large number of objects, we cannot apply these structures directly to our scenarios. Thus, we propose using a structure which treats all object by only one model and predicts the movements (movement vectors) instead of positions.

In order to test our idea, we built an LSTM comprising two bidirectional LSTM layers with 64 dimensions, a dropout layer with $p = 0.5$ in between, and a linear layer which generates two-dimensional outputs, representing the x and y values of the movement vector. The input of the LSTM module are two-dimensional movement vectors with dynamic lengths up to five steps of the objects' movement histories. We applied this module to our pedestrian tracking datasets. The results of this experiment show that our LSTM module can predict the next movement vector of multiple pedestrians with about 3.6 pixels (0.43 m) precision, which is acceptable for our scenarios. Therefore, training a single LSTM on multiple objects would be enough for predicting the objects' movement vectors.

We embed a similar LSTM module into our network as shown in Figure 12. For the training of the module, the network first generates a sequence of object movement vectors based on the object location predictions. In our experiments, each track has a dynamic history of up to five last predictions. As tracks are not assumed to start at the same time, the length of each track history can be different. Thus, we use zero-padding to make the lengths of track histories similar, allowing to process them together as a batch. These sequences are fed into the first LSTM layer with a hidden size of 64. A dropout with $p = 0.5$ is then applied to the hidden state of the first LSTM layer, and passes the results to the second LSTM layer. The output features of the second LSTM layer are fed into a linear layer of size 128. The 128-dimensional output of the LSTM module Out_{LSTM} is then concatenated with Out_{SNN} and Out_{Graph}, the output of the GCNN module. The concatenation allows

the network to predict object locations more precisely based on a fusion of appearance and movement features.

6.2. GraphCNN Module

The GraphCNN module consists of three 1D convolution layers with 1×1 kernels and respectively 32, 64, and 128 channels. We generate each object's adjacency graph based on the location prediction of all objects. To this end, the eight closest neighbors in a radius of 7.5 m from the object are considered and modeled as a directed graph by a set of vectors v_i from the neighbouring objects to the target object's position (x, y). The resulting graph is represented as $[x, y, x_{v_1}, y_{v_1}, \ldots, x_{v_8}, y_{v_8}]$. If less than eight neighbors are existing, we zero-pad the rest of the vectors.

The GraphCNN module also uses historical information by considering five previous graph configurations. Similarly to the LSTM module, we use zero-padding if less than five previous configurations are available. The resulting graph sequences are described by a 18×5 matrix which is fed into the first convolution layer. In our setup, graph sequences of multiple objects are given to the network as a batch of matrices. The output of the last convolutional layer is gone through a global average pooling in order to generate the final 128-dimensional output of the module Out_{Graph}, which is concatenated to Out_{SNN} and Out_{LSTM}. The features of the GraphCNN module enable the network to better understand group movements.

6.3. Squeeze-and-Excitation Layers

During our preliminary experiments in Section 5, we experienced a high deviation in the quality of activation maps produced by the convolution layers in DCFNet and SMSOT-CNN. This deviation shows the direct impact of single channels and their importance for the final result of the network. In order to consider this factor in our approach, we model the dominance of the single channels by Squeeze-And-Excitation (SE) layers [71].

CNNs extract image information by sliding spatial filters across the inputs to different layers. While the lower layers extract detailed features such as edges and corners, the higher layers can extract more abstract structures such as object parts. In this process, each filter at each layer has a different relevance to the network output. However, all filters (channels) are usually weighted equally. Adding the SE layers to a network helps weighting each channel adaptively based on their relevance. In the SE layers, each channel is squeezed to a single value by using global average pooling [72], resulting in a vector with k entries. This vector is given to a fully connected layer reducing the size of the output vector by a certain ratio, followed by a ReLu activation function. The result is fed into a second fully connected layer scaling the vector back to its original size and applying a sigmoid activation afterwards. In the final step, each channel of the convolution block is multiplied by the results of the SE layer. This channel weighting step adds less than 1% to the overall computational cost. As can bee seen in Figure 12, we add one SE layer after each branch of the SNN module, and one SE layer after the fusion of Out_{SNN}, Out_{LSTM}, and Out_{Graph}.

6.4. Online Hard Example Mining

In the object detection domain, datasets usually contain a large number of easy cases with respect to cases which are challenging for the algorithms. Several strategies have been developed in order to account for this, such as sample-aware loss functions (e.g., Focal Loss [73]), where the easy and hard samples are weighted based on their frequencies, and online hard example mining (OHEM) [28], which gives hard examples to the network if they are previously failed to be correctly predicted. The selection and focusing on such hard examples can make the training more effective. OHEM have been explored in the object detection task [74,75], however, its usage has not been investigated for the object tracking task. In the multi-object tracking domain, such strategies have been rarely used although the tracking datasets suffer from the sample problem as the detection datasets.

To the best of our knowledge, none of the previous works in the regression-based tracking used OHEM during their training process.

Thus, in order to deal with the sample imbalance problem of our datasets, we propose adapting and employing OHEM for our training process. To this end, if the tracker loses an object during training, we reset the object to its original starting position and the starting frame, and feed it to the network in the next iteration again. If the tracker fails again, we ignore the sample by removing it from the batch.

7. Experimental Setup

For all of our experiments, we used PyTorch and one Nvidia Titan XP GPU. We trained all networks with an SGD optimizer and an initial learning rate of 10^{-6}. For all training setups, unless indicated otherwise, we use the $L1$ loss, $L(x, \hat{x}) = |x - \hat{x}|$, where x and $\hat{x}$ represent the output of the network and ground truth, respectively. The batch size of all our experiments is 150; however, during offline feedback training, the batch size can differ due to unsuccessful tracking cases and subsequent removal of the object from the batch.

In our experiments, we consider SMSOT-CNN as baseline network and compare different parts of our approach to it. The original SMSOT-CNN is described in Caffe. In order to make it completely comparable to our approach, we re-implement it in PyTorch. For the training of SMSOT-CNN, we assign different fractions of the initial learning rate to each layer, as in the original Caffe implementation, inspired by the GOTURN's implementation. In more detail, we assign the initial learning rate to each convolutional layer, and assign a learning rate 10 times larger to the fully connected layers. Weights are initialized by Gaussians with different standard deviations, while biases are initialized by constant values (zero or one), as in the Caffe version. The training process of SMSOT-CNN is based on a so-called Example Generator. Provided with one target image with known object coordinates, this creates multiple examples by creating and shifting the search crop to create different kinds of movements. It is also possible to give the true target and search images. A hyperparameter set to 10 controls the number of examples generated for each image. For the pedestrian tracking, we use DLR-ACD to increase the number of available training samples. SMSOT-CNN is trained completely offline and learns to regress the object location based on only the previous location of the object.

For AerialMPTNet, we train the SNN module and the fully connected layers as in SMSOT-CNN. After that, the layers are initialized with the learnt weights, and the remaining layers are initialized with the standard PyTorch initialization. Moreover, we decay the learning rate by a factor of 0.1 for every twenty thousand iterations and train AerialMPTNet in an end-to-end fashion by using feedback loops to integrate previous movement and relationship information between adjacent objects. In contrast to the training process of SMSOT-CNN, which is based on artificial movements created by the example generator, we train our networks based on real tracks.

In the training process, a batch of 150 random tracks (i.e., objects from random sequences of the training set) is first selected starting at a random time step between 0 and the track end $t_{end} - 1$. We give the network the target and search patches for these objects. The network's goal is to regress each object position in the search patches consecutively until either the object is lost or the track ends. The target and search patches are generated based on the network predictions in consecutive frames. The object will remain in the batch as long as the network tracks it successfully. If the ground truth object position lies outside of the predicted search area or the track reaches its end frame, we remove the object from the batch and replace it with a new randomly selected object.

For each track and each time step, the network's prediction is stored and used from the LSTM and GraphCNN module. For each object in the batch, the LSTM module is given the objects' movement vectors from the latest time steps up to a maximum number of five, as explained in Section 6. This process provides the network with an understanding of each object's movement characteristics by a prediction of the next movement. As a result, our network uses its predictions as feedback to improve its performance. Furthermore, we

perform gradient clipping for the LSTM during training to prevent exploding gradients. The neighbor calculation of the GraphCNN module is also based on the network's prediction of each object's position, as mentioned in Section 6. Based on the network's prediction of the object position, we search for the nearest neighbors in the ground truth annotation of that frame. However, during the testing phase, we search nearest neighbors based on the network's prediction of the object positions.

For the pedestrian dataset, we set the context factor to 4, with each object with a bounding box size of 4×4 pixel resulting in an image patch of 16×16 pixels. For vehicle tracking, however, due to the larger sizes of their bounding boxes, we reduce the context factor to 3. This helps avoiding multiple vehicles in a single image patch which could cause track confusion.

8. Evaluation and Discussion

In this section, we evaluate different parts of our proposed AerialMPTNet on the KIT AIS and AerialMPT datasets through a set of ablation studies. Furthermore, we compare our results to the tracking methods discussed in Section 5. Table 9 reports the different network configurations for our ablation studies.

Table 9. Different network configurations.

Name	SNN	LSTM	GCNN	SE Layers	OHEM
SMSOT-CNN	✓	×	×	×	×
AerialMPTNet$_{LSTM}$	✓	✓	×	×	×
AerialMPTNet$_{GCNN}$	✓	×	✓	×	×
AerialMPTNet	✓	✓	✓	×	×
AerialMPTNet$_{SE}$	✓	✓	✓	✓	×
AerialMPTNet$_{OHEM}$	✓	✓	✓	×	✓

8.1. SMSOT-CNN (PyTorch)

The tracking results of our PyTorch SMOST-CNN on the ArialMPT and KIT AIS pedestrian and vehicle datasets are presented in Table 10. Therein, SMSOT-CNN achieves MOTA and MOTP scores of -35.0 and 70.0 for the KIT AIS pedestrian, and 37.1 and 75.8 for the KIT AIS vehicle dataset, respectively. It achieves, respectively, a MOTA and MOTP of -37.2 and 68.0 on the AerialMPT dataset. It can be seen that IDF is highest for the RaR_Snach_Zone and Pasing7 for the AerialMPT and KIT AIS dataset by achieving about 63.1 and 57.7 respectively. This is due to the less persons on those sequences, lowering the possibility of falsely tracking an ID. This shows its affect on other parameters such as IDP, IDR, FAR, MT, PT, ML as well. Regarding FP, FN and ID switch, Munich02 and Bauma3 have the highest wrong detections and id switches, however, the performance of algorithm on Bauma3 is comparable with other sequences to the less noise in the dataset. A comparison of the results to [22] shows that our PyTorch implementation works rather similarly to the original Caffe version, with only 5.2 and 4.0 points smaller MOTA for the KIT AIS pedestrian and vehicle, respectively. For the rest of our experiments, we consider the results of this implementation of SMOST-CNN as the baseline for our evaluations.

Table 10. SMSOT-CNN on the KIT AIS and AerialMPT datasets.

Sequences	# Imgs	GT	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS Pedestrian Dataset																		
AA_Crossing_02	13	94	49.4	49.2	49.6	51.7	51.3	42.92	22.4	60.6	17.0	558	548	15	88	1.2	66.8	2.4
AA_Walking_02	17	188	29.6	29.0	30.2	31.9	30.6	113.76	9.1	45.7	45.2	1934	1820	25	139	−41.5	65.7	−40.6
Munich02	31	20.7	230	19.9	21.5	24.5	22.6	165.45	3.5	44.3	52.2	5129	4625	91	271	−60.7	67.1	−59.3
RaR_Snack_Zone_02	4	220	63.1	62.9	63.4	64.2	63.7	79.0	35.0	63.6	1.4	316	310	1	39	27.5	78.2	27.6
RaR_Snack_Zone_04	4	311	63.5	63.3	63.7	65.3	64.9	108.5	35.0	64.0	1.0	434	427	3	48	29.8	76.7	30.0
Overall	69	1043	32.5	31.7	33.4	35.7	33.9	121.32	22.2	56.0	21.8	8371	7730	135	585	−35.0	70.0	−33.9
AerialMPT Dataset																		
Bauma3	16	609	29.3	28.6	30.0	34.6	33.0	385.69	9.9	47.1	43.0	6171	5748	200	458	−37.9	69.1	−35.7
Bauma6	26	270	30.8	28.6	33.3	37.7	32.3	161.23	12.2	57.4	30.4	4192	3311	115	302	−43.4	67.7	−41.2
Karlsplatz	27	146	30.7	29.4	32.2	33.8	30.8	94.93	6.9	58.2	34.9	2563	2233	26	95	−42.9	67.9	−42.2
Pasing7	24	103	57.7	54.5	61.3	61.9	55.1	43.42	35.9	54.4	9.7	1042	786	7	136	11.1	67.6	11.4
Pasing8	27	83	33.5	32.6	34.4	35.1	33.3	50.30	8.4	54.2	37.4	1358	1253	10	82	−35.7	67.0	−35.2
Witt	8	185	15.8	15.7	15.9	16.4	16.2	150.38	1.1	20.5	78.4	1203	1184	1	9	−68.6	61.5	−68.6
Overall	128	1396	32.0	30.7	33.4	36.6	33.6	129.13	10.7	47.7	41.6	16,529	14,515	359	1082	−37.2	68.0	−35.6
KIT AIS Vehicle Dataset																		
MunichStreet02	20	47	87.4	85.0	90.1	90.5	85.3	5.80	87.2	8.5	4.3	116	71	1	7	74.8	80.6	74.9
StuttgartCrossroad01	14	49	67.3	63.6	71.5	74.9	66.6	14.86	57.1	30.6	12.3	208	139	3	17	36.8	75.3	37.3
MunichCrossroad02	45	66	50.6	49.5	51.7	53.5	51.3	24.38	45.5	27.3	27.2	1097	1001	17	41	1.9	69.4	2.6
MunichStreet04	29	68	83.5	82.4	84.7	85.8	83.6	8.83	76.5	14.7	8.8	256	215	6	15	68.6	79.7	68.9
Overall	108	230	68.0	66.4	69.7	71.3	67.9	15.53	65.7	20.4	13.9	1677	1426	27	80	37.1	75.8	37.6

8.2. AerialMPTNet (LSTM Only)

In this step, we evaluate the influence of the LSTM module on the tracking performance of our AerialMPTNet. Table 11 reports the tracking result of AerialMPTNet$_{LSTM}$ on our experimental datasets. We use the pre-trained weights of SMSOT-CNN to initialize the convolutional weights and biases. For the KIT AIS pedestrian dataset, we evaluate the effects of freezing the weights during the training of LSTM. The tracking results with frozen and trainable convolutional weights in Table 11 show that the latter improves MOTA and MOTP values by 8.2 and 0.5, respectively. Moreover, the network trained with trainable weights tracks 6.9% more objects mostly during their lifetimes (MT). We can observe that this increase in performance holds for all sequences with different number of frames and objects with regard to IDF, IDPR, IDR, MT, ML, FP and FN. Having said that by not freeźing the initial weights, the number of ID switches (IDs) from 231 increases to 270, which we contemplate this is due to the small size of dataset and high number of trainable weights. However, after further investigation we notice that after visual inspections that although the network with the trainable weights can track objects for a longer time; however, when the objects get into crowded scenarios, it loses their track by switching their IDs. Based on these comparisons, we can argue that the computed features in SNN need fine tuning to some degree in order to work jointly with the LSTM module. That could be the reason why the training with the trainable weights outperforms the setting employing frozen weights. Thus, for the rest of our experiments, we use trainable weights. Consequently, Table 11 shows only the results with trainable weights for the AerialMPT and KIT AIS vehicle datasets.

Table 11. AerialMPTNet$_{LSTM}$ on the KIT AIS and AerialMPT datasets. The **best** overall values of the two configurations on the KIT AIS pedestrian dataset are highlighted.

Sequences	# Imgs	GT	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS Pedestrian Dataset—Frozen Weights																		
AA_Crossing_02	13	94	42.0	41.8	42.2	44.8	44.5	48.92	13.8	59.6	26.6	636	626	13	99	−12.3	68.4	−11.3
AA_Walking_02	17	188	34.7	34.0	35.4	37.2	35.8	104.94	8.0	55.3	36.7	1784	1678	22	227	−30.4	67.4	−29.7
Munich02	31	230	26.0	25.1	26.9	33.1	30.8	146.81	6.1	57.8	36.1	4551	4098	191	463	−44.3	67.8	−41.2
RaR_Snack_Zone_02	4	220	57.1	56.9	57.3	59.0	58.6	90.25	29.1	69.5	1.4	361	355	1	42	17.1	72.9	17.2
RaR_Snack_Zone_04	4	311	64.7	64.4	64.9	66.3	65.9	105.25	39.6	58.8	1.6	421	415	4	52	31.7	73.8	32.0
Overall	69	1043	35.5	34.6	36.3	40.4	38.5	112.36	22.0	**60.3**	17.7	7753	7172	**231**	**883**	−26.0	**69.3**	−24.1
KIT AIS Pedestrian Dataset—Trainable Weights																		
AA_Crossing_02	13	94	47.1	49.9	47.3	49.6	49.2	44.77	23.4	48.9	27.7	582	572	11	91	−2.6	68.2	−1.8
AA_Walking_02	17	188	39.8	39.2	40.5	41.9	40.5	96.47	18.6	46.8	34.6	1640	1553	31	215	−20.7	67.2	−19.6
Munich02	31	230	29.6	28.6	30.8	37.1	34.5	139.10	8.3	59.6	32.1	4312	3852	221	506	−36.9	67.1	−33.3
RaR_Snack_Zone_02	4	220	63.0	62.8	63.2	64.9	64.4	77.50	37.3	60.0	2.7	310	304	4	31	28.6	72.2	28.9
RaR_Snack_Zone_04	4	311	67.6	67.5	67.8	69.1	68.8	96.50	46.0	50.8	3.2	386	380	3	43	37.5	73.3	37.7
Overall	69	1043	**39.7**	**38.8**	**40.6**	**44.6**	**42.6**	**104.78**	**28.9**	53.8	**17.3**	**7230**	**6661**	270	886	**−17.8**	68.8	**−15.5**
AerialMPT Dataset																		
Bauma3	16	609	28.3	27.7	29.0	34.6	33.0	386.00	8.4	51.2	40.4	6176	5745	246	608	−38.5	71.0	−35.7
Bauma6	26	270	33.2	31.2	35.5	39.3	34.5	152.35	13.0	58.5	28.5	3961	3225	135	387	−37.8	70.1	−35.3
Karlsplatz	27	146	48.4	47.0	50.0	51.4	48.2	68.89	24.7	55.5	19.8	1860	1641	16	140	−4.2	69.7	−3.8
Pasing7	24	103	61.0	58.5	63.6	64.3	59.2	38.08	35.9	56.3	7.8	914	737	5	127	19.8	70.5	20.0
Pasing8	27	83	41.3	40.6	42.1	42.7	41.4	43.78	18.1	50.6	31.3	1182	1108	4	90	−18.7	69.4	−18.6
Witt	8	185	15.6	15.5	15.7	17.3	17.1	148.75	2.7	23.8	73.5	1190	1171	3	24	−66.9	61.1	−66.8
Overall	128	1396	35.7	34.5	37.0	40.5	37.7	119.40	12.8	49.8	37.4	15,283	13,627	409	1376	−28.1	70.1	−26.3
KIT AIS Vehicle Dataset																		
MunichStreet02	20	47	81.9	79.9	84.0	84.9	80.6	7.60	74.5	10.6	14.9	152	113	4	3	63.9	79.6	64.4
StuttgartCrossroad01	14	49	65.9	62.4	69.9	72.7	65.0	15.50	59.2	26.5	14.3	217	151	2	11	33.2	76.2	33.5
MunichCrossroad02	45	66	57.7	56.0	59.5	60.6	56.9	21.93	48.5	33.3	18.2	987	850	22	43	13.7	69.4	14.7
MunichStreet04	29	68	88.7	88.3	89.1	89.9	89.0	5.79	86.8	7.4	5.8	168	153	2	3	78.7	79.8	78.8
Overall	108	230	71.6	69.8	73.4	74.5	70.9	14.11	67.4	19.6	13.0	1524	1267	30	60	43.3	75.7	43.9

Table 12 represents the overall performances of different tracking methods on the KIT AIS and AerialMPT datasets. According to the table, AerialMPTNet$_{LSTM}$ outperforms

SMSOT-CNN with significant larger MOTA on all experimental datasets. In particular, based on Tables 10 and 11, the main improvements happen for complex sequences such as the "AA_Walking_02" and "Munich02" sequences of the KIT AIS pedestrian dataset, with a 20.8 and 23.8 points larger MOTA, respectively.

On the AerialMPT dataset, the most complex sequences are "Bauma3" and "Bauma6" presenting overcrowded scenarios with many pedestrians intersecting. According to the results, using the LSTM module does not help the performance relevantly. In such complex sequences, the trajectory information of the LSTM module is not enough for distinguishing pedestrians and tracking them within the crowds. Furthermore, the increase in the number of mostly and partially tracked objects (MT and PT) and the decrease in the number of mostly lost ones (ML) indicate that the LSTM module helps AerialMPTNet in the tracking of the objects for a longer time. This, however, causes a larger number of ID switches as discussed before. On the KIT AIS vehicle dataset, although the results show a significant improvement of AerialMPTNet$_{LSTM}$ over SMSOT-CNN, the performance improvements are minor compared to the pedestrian datasets. This could be due the more distinguishable appearance features of the vehicles, leading to a good performance even when relying solely on the SNN module.

Table 12. Overall Performances of Different Tracking Methods on the KIT AIS and AerialMPT Datasets. The **first** and **second** best values on each dataset are highlighted.

Methods	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS Pedestrian Dataset																
KCF	9.0	8.8	9.3	10.3	9.8	165.6	1.1	53.8	45.1	11,426	10,782	32	**116**	−84.9	**87.2**	−84.7
Median Flow	18.5	18.3	18.8	19.5	19.0	144.7	7.7	55.8	36.5	9986	9678	**30**	161	−63.8	77.7	−63.5
CSRT	16.0	16.9	15.2	17.5	19.4	126.5	9.6	51.0	39.4	8732	9924	91	254	−55.9	78.4	−55.1
MOSSE	9.1	8.9	9.3	10.5	10.0	163.8	0.8	54.0	45.2	11,303	10,765	**31**	**133**	−85.8	**86.7**	−83.5
Tracktor++	6.6	9.0	5.2	10.8	18.7	**81.7**	1.1	28.4	70.5	**5648**	10,723	648	367	−41.5	40.5	–
Stacked-DCFNet	30.0	30.2	30.9	33.1	32.3	120.5	13.8	**62.6**	23.6	8316	8051	139	651	−37.3	71.6	−36.1
SMSOT-CNN	32.5	31.7	33.4	35.7	33.9	121.3	22.2	**56.0**	21.8	8371	7730	135	585	−35.0	70.0	−33.9
AerialMPTNet$_{LSTM}$ (Ours)	**39.7**	**38.8**	**40.6**	**44.6**	**42.6**	104.8	**28.9**	53.8	17.3	7230	**6661**	270	886	**−17.8**	68.8	**−15.5**
AerialMPTNet$_{GCNN}$ (Ours)	37.5	36.7	38.4	42.0	40.0	109.5	25.3	55.3	19.4	7555	6980	259	814	−23.0	69.6	−20.9
AerialMPTNet (Ours)	**40.6**	**39.7**	**41.5**	**45.1**	**43.2**	**103.4**	**28.1**	55.3	**16.6**	**7138**	**6597**	236	897	**−16.2**	69.6	**−14.2**
AerialMPTNet$_{SE}$ (Ours)	38.3	37.5	39.1	42.8	41.1	107.2	27.4	54.5	**18.1**	7395	6876	250	818	−20.7	69.9	−18.7
AerialMPTNet$_{OHEM}$ (Ours)	38.6	37.7	39.4	42.7	40.9	107.7	26.1	55.8	**18.1**	7435	6889	254	854	−21.2	69.5	−19.1
AerialMPT Dataset																
KCF	11.9	11.5	12.3	13.4	12.5	167.2	3.7	17.0	79.3	21,407	19,820	86	212	−80.5	77.2	−80.1
Median Flow	12.2	12.0	12.4	13.1	12.7	162.0	1.7	20.2	78.1	20,732	19,883	**46**	**144**	−77.7	**77.8**	−77.5
CSRT	16.9	16.6	17.1	20.3	19.7	148.5	2.9	37.8	59.3	19,011	18,235	426	668	−64.6	74.6	−62.7
MOSSE	12.1	11.7	12.4	13.7	12.9	165.7	3.8	17.9	78.3	21,204	19,749	**85**	**194**	−79.3	**80.0**	−78.9
Tracktor++	4.0	8.8	3.1	5.0	8.7	**93.0**	0.1	7.6	92.3	**11,907**	21,752	399	345	−48.8	40.3	–
Stacked-DCFNet	28.0	27.6	28.5	31.4	30.4	128.3	9.4	44.2	46.4	16,422	15,712	322	944	−41.8	72.3	−40.4
SMSOT-CNN	32.0	30.7	33.4	36.6	33.6	129.1	10.7	47.7	41.6	16,529	14,515	359	1082	−37.2	68.0	−35.6
AerialMPTNet$_{LSTM}$ (Ours)	35.7	34.5	37.0	40.5	37.7	119.4	12.8	**49.8**	37.4	15,283	13,627	409	1376	−28.1	70.1	−26.3
AerialMPTNet$_{GCNN}$(Ours)	37.0	35.7	38.3	42.0	39.1	117.0	15.6	46.0	38.4	14,983	13,279	433	1229	−25.4	69.7	−23.5
AerialMPTNet (Ours)	**37.8**	**36.5**	**39.3**	**43.1**	**40.0**	115.5	15.3	**49.9**	**34.8**	14,782	**13,022**	436	1269	**−23.4**	69.7	**−21.5**
AerialMPTNet$_{SE}$ (Ours)	**38.9**	**37.5**	**40.4**	**44.1**	**40.9**	**113.8**	**17.0**	48.1	**34.9**	**14,568**	**12,799**	430	1212	**−21.4**	69.8	**−19.6**
AerialMPTNet$_{OHEM}$ (Ours)	37.2	35.8	38.7	42.4	39.3	117.3	**16.0**	46.8	37.2	15,016	13,181	430	1284	−25.1	69.8	−23.2
KIT AIS Vehicle Dataset																
KCF	41.3	39.0	43.9	45.6	40.4	30.9	27.0	33.5	39.5	3339	2708	53	96	−22.6	72.3	−21.6
Median Flow	42.0	39.5	44.9	46.3	40.8	31.0	32.2	**40.0**	27.8	3348	2669	23	47	−21.4	**82.0**	−21.0
CSRT	**76.7**	**72.1**	**81.9**	**83.1**	**73.1**	14.1	**72.6**	21.7	**5.7**	1520	**841**	**21**	**46**	**52.1**	**80.7**	**52.5**
MOSSE	29.0	27.4	30.8	32.4	28.8	36.8	19.6	30.0	50.4	3977	3364	56	81	−48.7	75.0	−47.6
Tracktor++	55.3	66.6	47.2	57.3	**80.7**	**6.3**	30.0	**47.4**	22.6	**681**	2125	323	204	37.1	77.4	–
Stacked-DCFNet	**73.8**	**71.2**	**76.6**	**77.2**	71.8	14.0	69.1	15.2	15.7	1512	**1133**	**9**	**39**	**46.6**	**82.0**	**46.8**
SMSOT-CNN	68.0	66.4	69.7	71.3	67.9	15.5	65.7	20.4	13.9	1677	1426	27	80	37.1	75.8	37.6
AerialMPTNet$_{LSTM}$ (Ours)	71.6	69.8	73.4	74.5	70.9	14.1	**67.4**	19.6	13.0	1524	1267	30	60	43.3	75.7	43.9
AerialMPTNet$_{GCNN}$ (Ours)	71.1	69.4	72.9	74.1	70.6	14.2	67.0	18.7	14.3	1536	1289	22	58	42.8	75.9	43.2
AerialMPTNet (Ours)	70.0	68.3	71.8	73.9	70.3	14.4	66.5	20.9	12.6	1556	1299	29	67	42.0	76.3	42.6
AerialMPTNet$_{SE}$ (Ours)	70.0	68.4	71.7	73.2	69.8	14.6	63.5	24.8	**11.7**	1574	1334	23	84	41.1	75.6	41.5
AerialMPTNet$_{OHEM}$ (Ours)	71.7	70.0	73.4	74.6	71.2	**13.9**	67.0	19.6	13.4	**1505**	1262	27	66	43.8	75.5	44.3

8.3. AerialMPTNet (GCNN Only)

In this step, we focus on the modeling of the movement relationships between adjacent objects by AerialMPTNet$_{GCNN}$. As described in Table 9, we only consider the SNN and GCNN modules, and train the network on our experimental datasets. The tracking results on the test sequences of the datasets are shown in Table 13, and the comparisons to the other methods are provided in Table 12. By adding GCNN the AerialMPTNet performance increases compared to the SMSOT-CNN significantly. MOTA is improved by 11.8, 12.0, and 5.7 points on the AerialMPT and KIT AIS pedestrian and vehicle datasets, respectively. MT, PT, and ML values also improve for the pedestrian datasets. However, MT is only enhanced on the vehicle dataset. IDF, IDP and IDR is improved on three datasets indicating GCNN can improve the performance when objects are close to each other and keeping the track of each object as a graph node is effective. Altogether, these results indicate that the relational information is more important for the pedestrians than the vehicles. Moreover, according to Table 13, as in LSTM results, the use of GCNN helps more

for complex sequences. For example, MOTA on the "AA_Walking_02" and "Munich02" sequences increase by 13.9 and 20.5, respectively; however, it decreases respectively by 12.1 and 14.8 on "AA_Crossing_02" and "RaR_Snack_Zone_02". This could be due to the negative impact of the large number of zero paddings in the less crowded sequences with smaller number of adjacent objects. Compared to AerialMPTNet$_{LSTM}$, for the AerialMPT, AerialMPTNet$_{GCNN}$ performs slightly better while on the other two datasets it performs worse with a narrow margin. We assume that, due to the higher crowd densities in the AerialMPT dataset, the relationships between adjacent objects are more critical with respect to their movement histories.

Table 13. AerialMPTNet$_{GCNN}$ on the KIT AIS and AerialMPT datasets.

Sequences	# Imgs	GT	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS Pedestrian Dataset																		
AA_Crossing_02	13	94	43.5	43.3	43.7	45.5	45.1	48.4	18.1	51.1	30.8	629	619	11	90	−10.9	68.5	−10.1
AA_Walking_02	17	188	35.8	35.3	36.2	38.2	37.2	101.3	14.9	47.9	37.2	1723	1650	35	204	−27.6	68.1	−26.3
Munich02	31	230	29.1	28	30.2	35.5	32.9	142.9	8.3	53.9	37.8	4431	3951	204	434	−40.2	68.1	−36.9
RaR_Snack_Zone_02	4	220	55.2	55.0	55.4	56.9	56.5	94.7	28.2	69.5	2.3	379	373	3	41	12.7	73.3	13.0
RaR_Snack_Zone_04	4	311	67.2	67	67.3	68.5	68.2	98.2	44.4	52.1	3.5	393	387	6	45	36.1	73.9	36.5
Overall	69	1043	37.5	36.7	38.4	42.0	40.0	109.5	25.3	55.3	19.4	7555	6980	259	814	−23.0	69.6	−20.9
AerialMPT Dataset																		
Bauma3	16	609	29.6	28.9	30.4	36.5	34.7	376.7	11.3	48.3	40.4	6028	5581	276	550	−35.2	70.0	−32.1
Bauma6	26	270	36.7	34.4	39.3	43.7	38.2	144.2	20.4	50.4	29.2	3750	2994	126	329	−29.3	70.6	−26.9
Karlsplatz	27	146	43.7	72.3	45.2	46.4	43.4	75.6	15.8	63.0	21.2	2042	1809	25	145	−14.9	68.5	−14.2
Pasing7	24	103	68.6	66.0	71.4	71.6	66.1	31.5	51.5	39.8	8.7	756	857	4	96	34.7	71.0	34.9
Pasing8	27	83	41.2	40.4	42.1	42.7	41.0	44.0	18.1	51.8	30.1	1188	1108	2	94	−18.9	68.2	−18.9
Witt	8	185	14.1	14.0	14.2	15.3	15.1	152.4	1.6	19.5	78.9	1219	1200	0	15	−70.8	60.8	−70.8
Overall	128	1396	37.0	35.7	38.3	42.0	39.1	117.1	15.6	46.0	38.4	14,983	13,279	433	1229	−25.4	69.7	−23.5
KIT AIS Vehicle Dataset																		
MunichStreet02	20	47	82.6	80.5	84.7	85.4	81.1	7.4	76.6	6.4	17.0	148	109	4	3	65.0	79.5	65.5
StuttgartCrossroad01	14	49	70.0	66.5	73.8	76.7	69.1	13.6	65.3	22.4	12.3	190	129	2	11	42.1	75.7	42.3
MunichCrossroad02	45	66	56.3	54.7	58.0	59.4	56.0	22.3	44.0	34.8	21.2	1005	876	14	41	12.1	70.0	12.7
MunichStreet04	29	68	87.3	86.8	87.8	88.5	87.4	6.7	83.8	8.8	7.4	193	175	2	3	75.6	79.7	75.7
Overall	108	230	71.1	69.4	72.9	74.1	70.6	14.2	67.0	18.7	14.3	1536	1289	22	58	42.8	75.9	43.2

8.4. AerialMPTNet

In this step, we evaluate the complete AerialMPTNet by fusing the SNN, LSTM, and GCNN modules. Table 14 represents the tracking results of AerialMPTNet on the test sets of our experimental datasets, and Table 12 compares its overall performance to the other tracking methods.

According to the results, the AerialMPTNet outperforms AerialMPTnet$_{LSTM}$ and AerialMPTNet$_{GCNN}$ for both pedestrian datasets. However, this is not the case for the vehicle dataset. This is due to the main idea behind the development of the network. Since AerialMPTNet is initially designed for pedestrian tracking, it needs to be further adapted to domain specific challenges posed by vehicle tracking. For example, the distance threshold for the modeling if the adjacent object relationships (in GCNN) which considers objects within a distance of 50 pixels from the target object might miss many neighbouring vehicles, as usually the distances between vehicles are larger than those between pedestrians. Finally, AerialMPTNet achieves better tracking results than SMSOT-CNN on all three datasets.

Table 14. AerialMPTNet on the KIT AIS and AerialMPT datasets.

Sequences	# Imgs	GT	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS Pedestrian Dataset																		
AA_Crossing_02	13	94	46.7	45.6	46.9	49.3	48.8	45.1	23.4	51.1	25.5	586	576	12	92	−3.4	69.7	−2.5
AA_Walking_02	17	188	41.4	40.8	42.1	43.7	42.3	93.6	17.0	51.6	31.4	1591	1504	25	231	−16.8	68.5	−15.9
Munich02	31	230	31.2	30.2	32.3	37.8	35.3	136.8	10.4	55.7	33.9	4240	3808	192	498	−34.5	67.6	−31.4
RaR_Snack_Zone_02	4	220	59.0	58.8	59.2	60.9	60.5	86.0	33.2	65.0	1.8	344	3338	4	34	20.7	73.4	21.1
RaR_Snack_Zone_04	4	311	68.5	68.3	68.6	69.8	69.5	94.2	45.7	51.8	2.5	377	371	3	42	38.9	74.2	39.1
Overall	69	1043	40.6	39.7	41.5	45.1	43.2	103.4	28.1	55.3	16.6	7138	6597	236	897	−16.2	69.6	−14.2
AerialMPT Dataset																		
Bauma3	16	606	31.2	30.4	32.0	38.2	36.3	368.1	11.6	51.7	36.7	5890	5435	277	582	−32.0	70.8	−28.9
Bauma6	26	270	37.2	34.8	39.9	44.2	38.6	143.7	17.0	58.1	24.9	3736	2964	123	333	−28.4	70.2	−26.1
Karlsplatz	27	146	45.6	44.2	47.1	48.6	45.6	72.4	19.9	61.6	18.5	1954	1733	25	153	−10.0	67.4	−9.3
Pasing7	24	103	67.6	64.8	70.7	71.3	65.3	32.6	49.5	43.7	6.8	782	593	5	93	33.1	70.7	33.3
Pasing8	27	83	39.7	38.7	40.8	41.3	39.2	45.8	15.7	55.4	28.9	1238	1134	2	83	−22.9	68.9	−22.8
Witt	8	185	16.0	15.9	16.1	17.9	17.6	147.7	2.7	24.3	73.0	1182	1163	4	25	−65.9	60.1	−65.7
Overall	128	1396	37.8	36.5	39.3	43.1	40.0	115.5	15.3	49.9	34.8	14,782	13,022	436	1269	−23.4	69.7	−21.5
KIT AIS Vehicle Dataset																		
MunichStreet02	20	47	83.2	81.1	85.4	86.3	82.0	07.1	76.6	10.6	12.7	141	102	4	3	66.9	80.1	67.3
StuttgartCrossroad01	14	49	68.4	65.0	72.2	75.3	67.8	14.14	61.2	26.5	12.3	198	137	1	16	39.4	76.3	39.5
MunichCrossroad02	45	66	54.5	52.9	56.3	58.5	54.9	22.9	43.9	37.9	18.2	1033	895	20	45	9.6	70.1	10.5
MunichStreet04	29	68	86.5	86.0	87.0	89.1	88.0	6.3	85.3	7.4	7.3	184	165	4	3	76.8	80.2	77.0
Overall	108	230	70.0	68.3	71.8	73.9	70.3	14.4	66.5	20.9	12.6	1556	1299	29	67	42.0	76.3	42.6

8.4.1. Pedestrian Tracking

In more detail, AerialMPTNet yields the best MOTA among the studied methods on the "AA_Walking_ 02", "Munich02", and "RaR_Snack_Zone_02" sequences of the KIT AIS pedestrian dataset (−16.8, −34.5, and 38.9, respectively.) These sequences are the most

complex ones in this dataset with respect to the length and number of objects, thing which could significantly influence the MOTA value. Longer sequences and a higher number of objects usually cause the MOTA value to decrease, as it is more probable that the tracking methods lose track of the objects or confuse their IDs in these cases. Figure 13 illustrates the tracking results on two frames of the "AA_Walking_ 02" sequence of the KIT AIS pedestrian dataset by AerialMPTNet and SMSOT-CNN. Comparing the predictions and ground truth points demonstrates that SMSOT-CNN loses track of a considerably higher number of pedestrians between these two frames. While SMSOT-CNN's predictions are stuck at the diagonal background lines due to their similar appearance features to the pedestrians, AerialMPTNet can easily handle this situation due to the LSTM module.

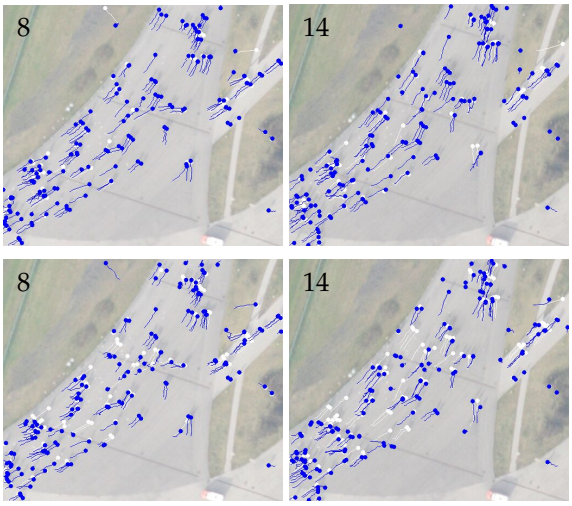


Figure 13. Tracking results by AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 8 and 14 of the "AA_Walking_ 02" sequence of the KIT AIS pedestrian dataset. The predictions and ground truth are depicted in blue and white, respectively.

We also visualized a cropped part of four frames from the "AA_Crossing_02" sequence of the KIT AIS pedestrian dataset in Figure 14. As in the previous example, AerialMPTNet clearly outperforms SMSOT-CNN on the tracking of the pedestrians crossing the background lines.

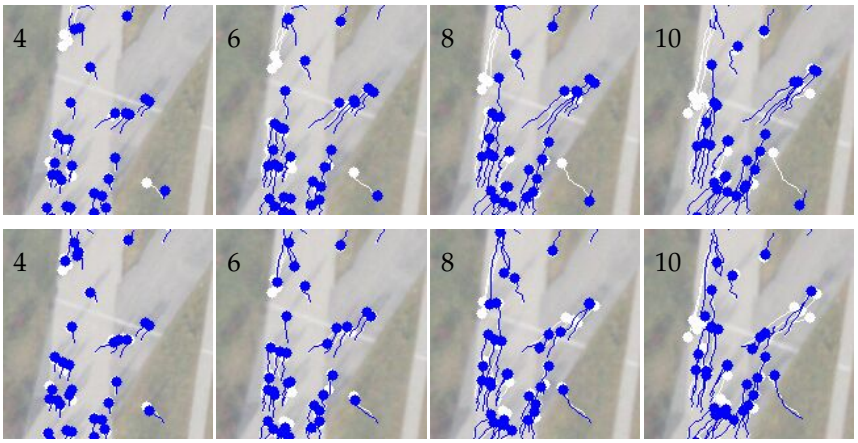


Figure 14. Tracking results by AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 4, 6, 8, and 10 of the "AA_Crossing_02" sequence of the KIT AIS pedestrian dataset. The predictions and ground truth are depicted in blue and white, respectively.

On the AerialMPT dataset, AerialMPTNet achieves the best MOTA scores among all studied methods in this paper on the "Bauma3", "Bauma6", and "Witt" sequences (-32.0, -28.4, -65.9), which contain the most complex scenarios regarding crowd density,

pedestrian movements, variety of the GSDs, and complexity of the terrain. However, in contrast to the KIT AIS pedestrian dataset, the MOTA scores are not correlated with the sequence lengths, indicating the impact of other complexities on the tracking results and the better distribution of complexities over the sequences of the AerialMPT dataset as compared to the KIT AIS pedestrian dataset.

Figure 15 exemplifies the role of the LSTM module in enhancing the tracking performance in AerialMPTNet. This figure shows an intersection of two pedestrians in the cropped patches from four frames of the "Pasing8" sequence of the AerialMPT dataset. According to the results, SMOT-CNN (bottom row) loses one of the pedestrians after their intersection leading to an ID switch. However, AerialMPTNet (top row) can track both pedestrians correctly, mainly relying on the pedestrians' movement histories (their movement directions) provided by the LSTM module.

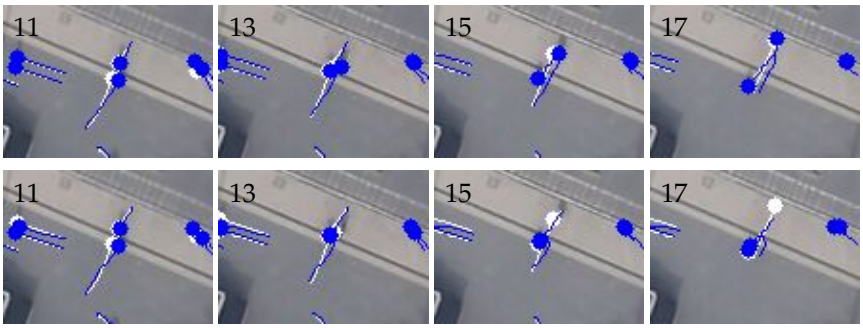


Figure 15. Tracking results by the AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 11, 13, 15, and 17 of the "Pasing8" sequence of the AerialMPT dataset. The predictions and ground truth are depicted in blue and white, respectively.

Figure 16 illustrates a case in which the advantage of the GCNN module can be clearly observed. The images are cropped from four frames of the "Karlsplatz" sequence of the AerialMPT dataset. It can be seen that SMSOT-CNN has difficulties in tracking the pedestrians in such crowded scenarios, where the pedestrians move in various directions. However, AerialMPTNet can handle this scenario mainly based on the pedestrian relationship models provided by the GCNN module.

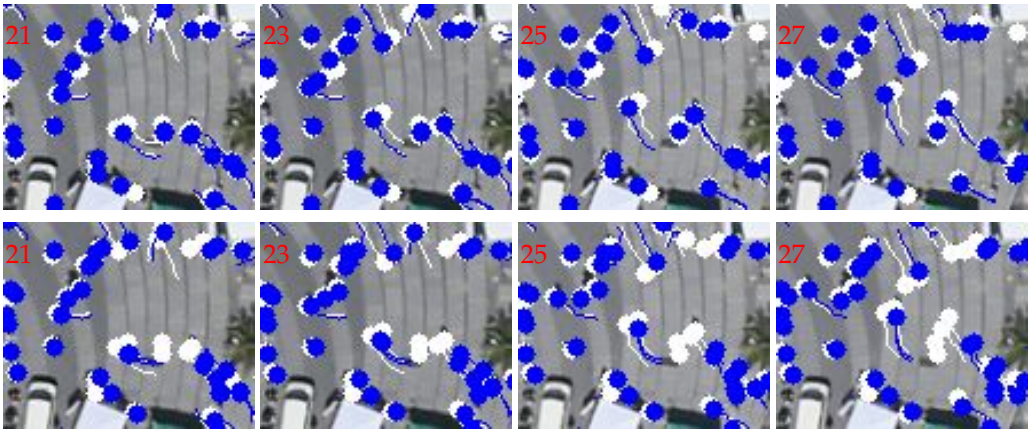


Figure 16. Tracking results by the AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 21, 23, 25, and 27 of the "Karlsplatz" sequence of the AerialMPT dataset. The predictions and ground truth are depicted in blue and white, respectively.

In addition, there are sequences where both methods reach their limits and perform poorly. Figure 17 illustrates the tracking results of AerialMPTNet (top row) and of SMSOT-CNN (bottom row) on two frames of the "Witt" sequence of the AerialMPT dataset. Comparing the predictions and ground truth object tracks indicates the large

number of lost objects by both methods. According to Tables 10 and 14, despite the small number of frames in the "Witt" sequence, the MOTA scores are low for both methods (−68.6 and −65.9). Further investigations show that these poor performances are caused by the non-adaptive search window size. In the "Witt" sequence, pedestrians move out of the search window and are lost by the tracker as a consequence. In order to solve this issue, the GSD of the frames as well as the pedestrian velocities should be considered in determining the search window size.

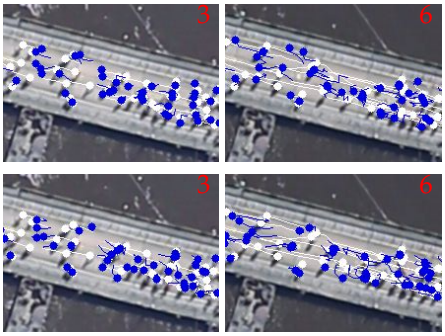


Figure 17. Tracking results by AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 3 and 6 of the "Witt" sequence of the AerialMPT dataset. The predictions and ground truth are depicted in blue and white, respectively.

In order to show the complexity of the pedestrian tracking task in the AerialMPT dataset, we report the tracking results of AerialMPTNet on the frames 18 and 10 of the "Munich02" and "Bauma3" sequences, respectively, in Figure 1.

8.4.2. Vehicle Tracking

According to Table 12, AerialMPTNet outperforms SMSOT-CNN also on the KIT AIS vehicle dataset, although the increase in performance is lower compared to the pedestrian tracking results. Results on different sequences in Tables 10 and 14 show that both methods perform poorly on the "MunichCrossroad02" sequence. Figure 18 visualizes the challenges that the tracking methods face in this sequence. For the visualization, we selected an early and a late frame to demonstrate the strong camera movements and changes in the viewing angle, which affect scene arrangements and object appearances. In addition, vehicles are partly or completely occluded by shadows and other objects such as trees. Finally, in this crossroad the movement patterns of the vehicles are complex.

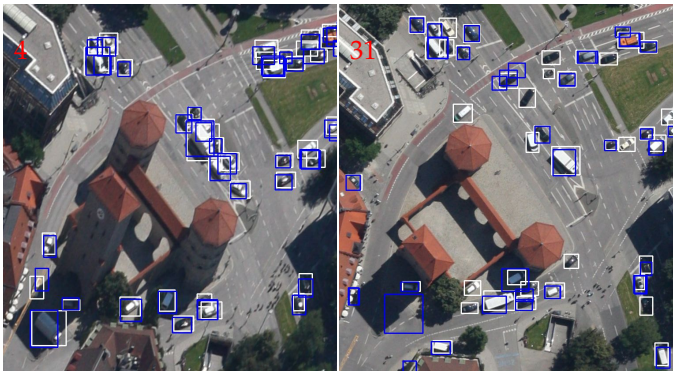


Figure 18. Tracking results by AerialMPTNet on the frames 4 and 31 of the "MunichCrossroad02" sequence of the KIT AIS vehicle dataset. The predictions and ground truth bounding boxes are depicted in blue and white, respectively. Several hindrances such as changing viewing angle, shadows, and occlusions (e.g., by trees) are visible.

In Figure 19, we compare the performances of AerialMPTNet and SMSOT-CNN on the "MunichCrossroad02" sequence. Both methods track AerialMPTNet tracks a few

vehicles better than SMSOT-CNN such as the ones located densely at the traffic lights. AerialMPTNet loses track of a few vehicles which are tracked correctly by SMSOT-CNN. These failures could be solved by a parameter adjustment in our AerialMPTNet.

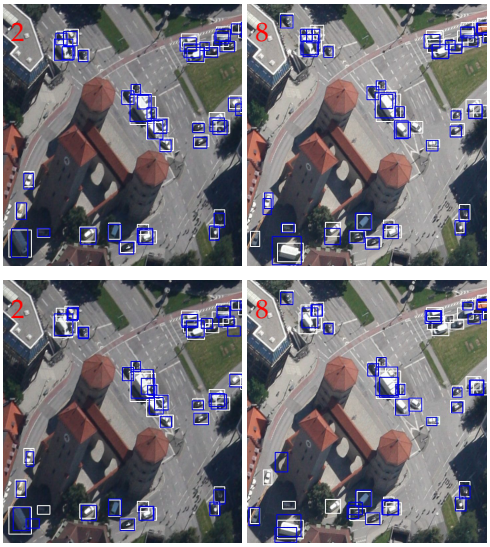


Figure 19. Tracking results by AerialMPTNet (top row) and SMSOT-CNN (bottom row) on the frames 2 and 8 of the "MunichCrossroad02" sequence of the KIT AIS vehicle dataset. The predictions and ground truth bounding boxes are depicted in blue and white, respectively.

In Figure 20 we compare performances on the "MunichStreet04" sequence. In this example, AerialMPTNet tracks the long vehicle much better than SMSOT-CNN.

Based on Tables 10 and 14, SMSOT-CNN outperforms our AerialMPTNet on the "MunichStreet02" sequence. In Figure 21, we exemplify the existing problems with our AerialMPTNet in this sequence. A background object (in the middle of the scene) has been recognized as a vehicle in frame 7, while the vehicle of interest is lost. A similar failure happens at the intersection. This is due to the parameter configurations of AerialMPTNet. As mentioned before, our method was initially proposed for pedestrian tracking, taking into account the characteristics and challenges of this task. Thus, we believe that by further investigations and parameter tuning, such issues should be solved.

Figure 20. Tracking results by AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 20 and 29 of the "MunichStreet04" sequence of the KIT AIS vehicle dataset. The predictions and ground truth bounding boxes are depicted in blue and white, respectively.

8.4.3. Localization Preciseness

In order to evaluate the preciseness of the object locations predicted by AerialMPTNet with respect to SMSOT-CNN, we vary the overlap criterion (IoU threshold) of the evaluation metrics for the Prcn, MOTA, MT, and ML metrics in Figure 22.

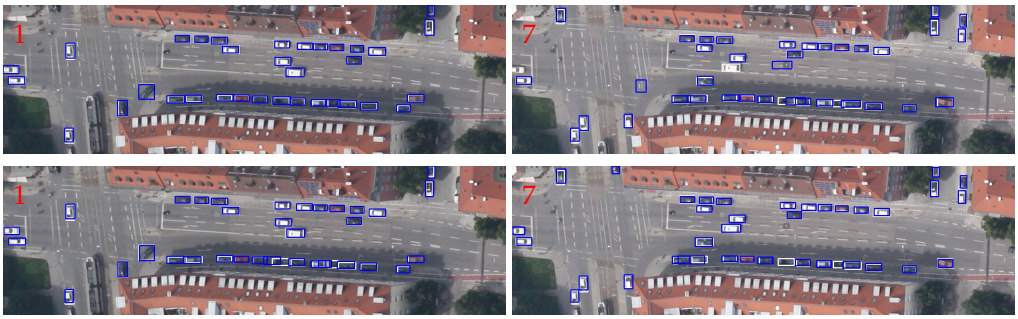


Figure 21. Tracking results by AerialMPTNet (**top row**) and SMSOT-CNN (**bottom row**) on the frames 1 and 7 of the "MunichStreet02" sequence of the KIT AIS vehicle dataset. The predictions and ground truth bounding boxes are depicted in blue and white, respectively.

According to the plots, the performance of both methods decreases by increasing the IoU threshold, requiring more overlap between the predicted and ground truth bonding boxes (more precise localization.) For all presented metrics, the preciseness of our ArialMPTNet surpasses that of the SMSOT-CNN. However, for the vehicle dataset the performance increase by our AerialMPTNet over SMSOT-CNN is lower than for the case of the pedestrian datasets.

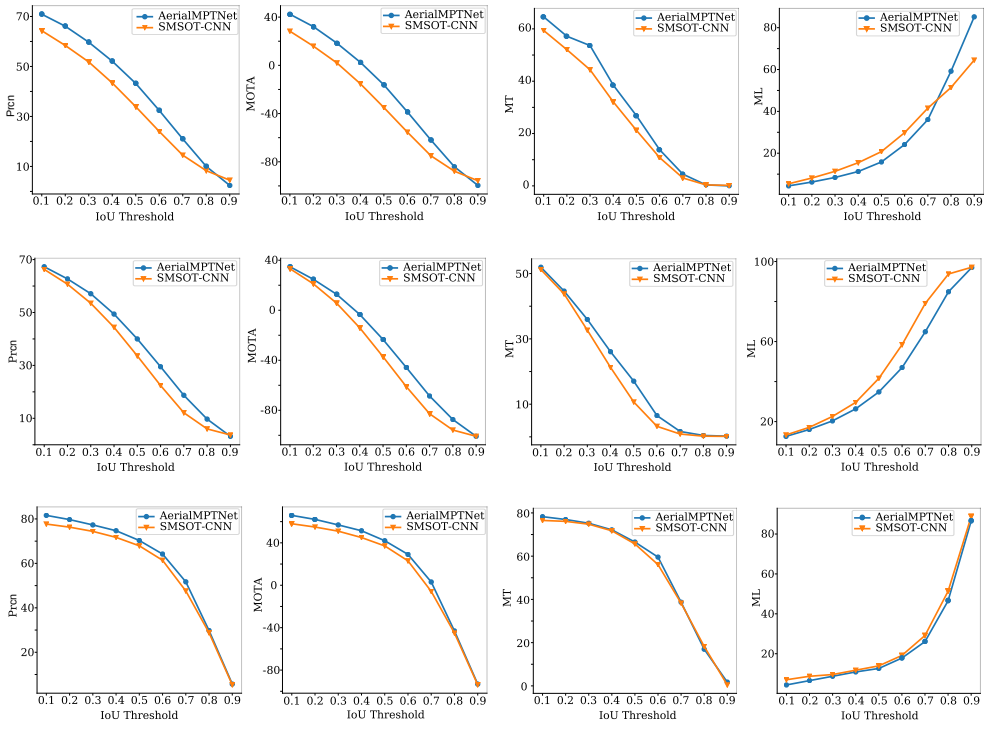


Figure 22. Comparing the Prcn, MOTA, MT, and ML of the AerialMPTNet and SMSOT-CNN on the KIT AIS pedestrian (**first row**), AerialMPT (**second row**), and KIT AIS vehicle (**third row**) datasets by changing the IoU thresholds of the evaluation metrics.

8.5. AerialMPTNet (with Squeeze-and-Excitation Layers)

In this step, we evaluate the improvement achieved by adding SE layers to our AerialMPTNet, as described in Section 6.3. We train the network on our three experimental datasets and report the tracking results in Table 12. Using the SE layers in AerialMPTNet$_{SE}$ degrades the results marginally for most of the metrics on the KIT AIS pedestrian and vehicle datasets as compared to AerialMPTNet. For the vehicle dataset, the SE layers improves the number of the mostly lost (ML) and partially tracked (PT) vehicles by 0.9% and 3.9%, respectively. On the AerialMPT dataset, however, the network behaviour is

totally different. AerialMPTNet$_{SE}$ outperforms AerialMPTNet for most of the metrics. SE layers improve MOTA and MOTP by 2 and 0.1 points, respectively. Moreover, the number of mostly tracked (MT) pedestrians increases by 1.7%. These inconstant behaviours could be due to the different image quality and contrast of the datasets. Since the images of the AerialMPT dataset are characterized by a higher quality, the adaptive channel weighting would be more meaningful.

8.6. Training with OHEM

We evaluate the influence of Online Hard Example Mining (OHEM) on the training of our AerialMPTNet as described in Section 6.4. The results are compared to those of the AerialMPTNet with its standard training procedure in Table 12. The use of OHEM in the training procedure reduces the performance marginally on both pedestrian datasets. For example, MOTA decreases by 5 and 1.7 points for the KIT AIS pedestrian and AerialMPT datasets, respectively. For the KIT AIS vehicle dataset, however, results show small improvements in the tracking results. For instance, MOTA rises by 1.8 points and the number of mostly tracked objects increases by 1.4%. We argue that pedestrian movement is highly complex and therefore, providing in input a similar situation multiple times to the tracker based on OHEM does not help the performance. For the vehicles, however, since they mostly moves in straight paths, OHEM can improve the training by retrying the failure cases. This is the first experiment on the benefits of OHEM in regression-based tracking. Further experiments have to be conducted in order to better understand the underlying reasons.

8.7. Huber Loss Function

We assess the effects of loss function in the tracking performance by using the Huber loss [76] instead of the traditional $L1$ loss function. The Huber loss is a mixture of the $L1$ and $L2$ losses, both commonly used for regression problems, and combines their strengths. The $L1$ loss measures the Mean Absolute Error (MAE) between the output of the network x and the ground truth $\hat{x}$:

$$L1(x, \hat{x}) = \sum_i |x_i - \hat{x}_i|. \tag{9}$$

The $L2$ loss calculates the Mean Squared Error (MSE) between the network output and the ground truth value:

$$L2(x, \hat{x}) = \sum_i (x_i - \hat{x}_i)^2. \tag{10}$$

The $L1$ loss is less affected by outliers with respect to the $L2$ loss. The Huber loss acts as a MSE when the error is small, and as a MAE when the error is large:

$$L_H(x, \hat{x}) = \sum_i z_i, \tag{11}$$

$$z_i = \begin{cases} 0.5(x_i - \hat{x}_i)^2, & if \ \ |x_i - \hat{x}_i| < 1 \\ |x_i - \hat{x}_i| - 0.5, & otherwise. \end{cases}$$

The Huber loss is more robust to outliers with respect to $L2$ and improves the $L1$ loss for the missing minima at the end of the training.

Table 15 compares results obtained by $L1$ and Huber loss functions. The model trained with the $L1$ loss outperforms the one trained with the Huber loss in general on all three datasets. There are a few metrics for which the Huber loss shows an improvement over $L1$, such as MT in the vehicle dataset or IDS in the AerialMPT dataset; however, these are marginal. Altogether, we can conclude that the $L1$ loss is a better option for our method in these tracking scenarios.

Table 15. Comparison of AerialMPTNet trained with the $L1$ and Huber Losses.

Loss	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS Pedestrian Dataset																	
L1	**40.6**	**39.7**	**41.5**	**45.1**	**43.2**	**103.45**	1043	**28.1**	55.3	**16.6**	**7138**	**6597**	236	897	**−16.2**	**69.6**	**−14.2**
Huber	38.8	37.9	39.7	43.1	41.1	107.42	1043	25.0	**56.5**	18.5	7412	6845	**212**	**866**	−20.3	69.4	−18.6
AerialMPT Dataset																	
L1	37.8	36.5	39.3	**43.1**	**40.0**	**115.48**	1396	15.3	**49.9**	**34.8**	**14,782**	**13,022**	436	1269	**−23.4**	69.7	**−21.5**
Huber	**38.0**	**36.7**	**39.5**	43.0	39.9	115.70	1396	**15.6**	48.4	36.0	14,809	13,051	**415**	**1196**	−23.5	**69.9**	−21.7
KIT AIS Vehicle Dataset																	
L1	**70.0**	**68.3**	**71.8**	**73.9**	**70.3**	**14.41**	230	66.5	**20.9**	**12.6**	**1556**	**1299**	**29**	67	**42.0**	**76.3**	**42.6**
Huber	67.2	65.5	69.0	70.6	67.1	15.98	230	**67.0**	17.4	15.6	1726	1461	34	**65**	35.2	76.1	35.9

9. Comparing AerialMPTNet to Other Methods

In this section, we compare the results of our AerialMPTNet with a set of traditional methods including KCF, Median Flow, CSRT, and MOSSE as well as DL-based methods such as Tracktor++, Stacked-DCFNet, and SMSOT-CNN. Table 12 reports the results of different tracking methods on the KIT AIS and AerialMPT datasets. In general, the DL-based methods outperform the traditional ones, with MOTA scores varying between -16.2 and -48.8 rather then between -55.9 and -85.8, respectively. The percentages of mostly tracked and mostly lost objects vary between 0.8% and 9.6% for the DL-based methods, while they lie between 36.5% and 78.3% for the traditional ones.

9.1. Pedestrian Tracking

Among the traditional methods, CSRT is the best performing one on the AerialMPT and KIT AIS pedestrian datasets, with MOTA values of -55.9 and -64.6. CSRT mostly tracks 9.6% and 2.9%, and of the pedestrians while it mostly loses 39.4% and 59.3% of the objects in these datasets. The DL-based methods, apart from Tracktor++, track much more pedestrians mostly ($>$13.8%) and lose much less pedestrians ($<$23.6%) with respect to traditional methods. The poor performances of Tracktor++ is due to its limitations in working with small objects. AerialMPTNet outperforms all other methods according to most of the adopted figures of merit on the pedestrian datasets with significantly larger MOTA values (-16.2 and -23.4) and competitive MOTP (69.6 and 69.7) values. It mostly tracks 5.9% and 4.6% more pedestrians and loses 5.2% and 6.8% less pedestrians with respect to the best performing previous method, SMSOT-CNN on the KIT AIS and AerialMPT pedestrian datasets, respectively.

9.2. Vehicle Tracking

As Table 12 demonstrates, the DL-based methods and CSRT outperform KCF, Median Flow, and MOSSE significantly, with average MOTA value of 42.9 versus -30.9. The DL-based methods and CSRT are also better with respect to the number of mostly tracked and mostly lost vehicles, varying between 30.0% and 69.1% and between 22.6% and 12.6%, respectively. These values for KCF, MOSSE, and Median Flow are between 19.6% and 32.2% and between 50.4% and 27.8%. Among the DL-based methods, Stacked-DCFNet has the best performance in terms of MOTA and MOTP, outperforming AerialMPTNet by 4.6 and 5.7 points, respectively. While the number of mostly tracked vehicles by Stacked-DCFNet is 2.6% larger than in the case of AerialMPTNet, it mostly loses 3.1% more vehicles. The performance of Tracktor++ increases significantly compared to the pedestrian scenarios, due to the ability of its object detector in detecting vehicles. Tracktor++ achieves a competitive MOTA of 37.1 without any ground truth initialization. The best performing method in terms of MOTA, MT, and ML is CSRT. It outperforms all other methods with a MOTA of 51.1 and MOTP of 80.7.

We rank the studied tracking methods based on their MOTA and MOTP values in Figure 23, with the diagrams offering a clear overview on their performance. AerialMPTNet appears the best method in terms of MOTA for both pedestrian datasets, and achieves competitive MOTP values. Median Flow, for example, achieves a very high MOTP values; however, because of the low number of matched track-object pairs after the first frame, it is not able to track many objects. Hence, the MOTP value solely is not a good performance indicator. For the KIT AIS vehicle dataset, AerialMPTNet shows worse performance than

the other methods according to the MOTA and MOTP values. CSRT and Stacked-DCFNet, however, perform favorably for vehicle tracking.

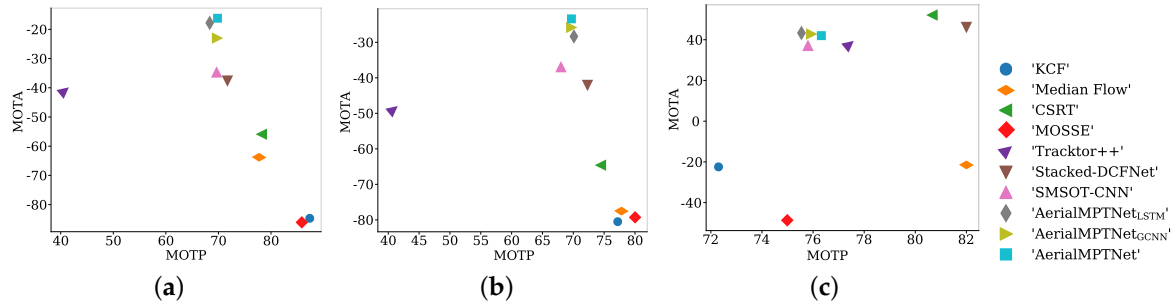


Figure 23. Ranking the tracking methods based on their MOTA and MOTP values on the (**a**) KIT AIS pedestrian, (**b**) AerialMPT, and (**c**) KIT AIS vehicle datasets.

10. Conclusions and Future Works

In this paper, we investigate the challenges posed by the tracking of pedestrians and vehicles in aerial imagery by applying a number of traditional and DL-based SOT and MOT methods on three aerial MOT datasets. We also describe our proposed DL-based aerial MOT method, the so-called AerialMPTNet. Our proposed network fuses appearance, temporal, and graphical information for a more accurate and stable tracking by employing a SNN, a LSTM, and a GCNN module. The influence of SE and OHEM on the performance of AerialMPTNet is investigated, as well as the impact of adopting an $L1$ rather than a Huber loss function. An extensive qualitative and quantitative evaluation shows that the proposed AerialMPTNet outperforms both traditional and state-of-the-art DL-based MOT methods for the pedestrian datasets, and achieves competitive results for the vehicle dataset. On the one hand, it is verified that LSTM and GCNN modules enhance the tracking performance; on the other hand, the use of SE and OHEM significantly helps only in some cases, while degrading the tracking results in other cases. The comparison of $L1$ and Huber loss shows that $L1$ is a better option for most of the scenarios in our experimental datasets.

We believe that the present paper can promote research on aerial MOT by providing a deep insight into its challenges and opportunities, and pave the path for future works in this domain. In the future, within the framework of AerialMPTNet, the search area size can be adapted to the image GSDs and object velocities and accelerations. Additionally, the SNN module can be modified in order to improve the appearance features extraction. The training process of most DL-based tracking methods relies on common loss functions, which do not correlate with tracking evaluation metrics such as MOTA and MOTP, as they are usually differentiable. Recently, differentiable proxies of MOTA and MOTP have been proposed [77], which can be also investigated for the aerial MOT scenarios.

Author Contributions: S.M.A., M.K. and R.B. designed the algorithm. S.M.A. and M.K. prepared the data. M.K. performed the experiments. S.M.A., M.K. and R.B. analyzed the results. R.B. wrote the manuscript. S.M.A., M.K., R.B. and P.R. revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bergmann, P.; Meinhardt, T.; Leal-Taixe, L. Tracking without bells and whistles. In Proceedings of the IEEE International Conference on Computer Vision (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 941–951.
2. Xiang, Y.; Alahi, A.; Savarese, S. Learning to track: Online multi-object tracking by decision making. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 4705–4713.

3. Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H. Fully-convolutional siamese networks for object tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherland, 8–16 October 2016; pp. 850–865.
4. Cuevas, E.V.; Zaldivar, D.; Rojas, R. *Kalman Filter for Vision Tracking*; Technical Report; Freie Universität Berlin: Berlin, Germany, 2005; doi:10.17169/refubium-22852. [CrossRef]
5. Cuevas, E.; Zaldivar, D.; Rojas, R. Particle filter in vision tracking. *e-Gnosis* **2007**, *5*, 1–11.
6. Bolme, D.S.; Beveridge, J.R.; Draper, B.A.; Lui, Y.M. Visual object tracking using adaptive correlation filters. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; pp. 2544–2550.
7. Boudoukh, G.; Leichter, I.; Rivlin, E. Visual tracking of object silhouettes. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 7–10 November 2009; pp. 3625–3628.
8. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–26 June 2005; pp. 886–893.
9. Marvasti-Zadeh, S.M.; Cheng, L.; Ghanei-Yakhdan, H.; Kasaei, S. Deep Learning for Visual Tracking: A Comprehensive Survey. *arXiv* **2019**, arXiv:1912.00535.
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
11. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826.
12. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497.
13. Wang, L.; Ouyang, W.; Wang, X.; Lu, H. Visual tracking with fully convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 3119–3127.
14. Zhang, K.; Liu, Q.; Wu, Y.; Yang, M.H. Robust visual tracking via convolutional networks without training. *IEEE Trans. Image Process.* **2016**, *25*, 1779–1792. [CrossRef] [PubMed]
15. Kim, H.I.; Park, R.H. Residual LSTM attention network for object tracking. *IEEE Signal Process. Lett.* **2018**, *25*, 1029–1033. [CrossRef]
16. Li, B.; Yan, J.; Wu, W.; Zhu, Z.; Hu, X. High performance visual tracking with siamese region proposal network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8971–8980.
17. Held, D.; Thrun, S.; Savarese, S. Learning to track at 100 fps with deep regression networks. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherland, 8–16 October 2016; pp. 749–765.
18. Song, Y.; Ma, C.; Wu, X.; Gong, L.; Bao, L.; Zuo, W.; Shen, C.; Lau, R.W.; Yang, M.H. Vital: Visual tracking via adversarial learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8990–8999.
19. Zhang, D.; Maei, H.; Wang, X.; Wang, Y.F. Deep reinforcement learning for visual object tracking in videos. *arXiv* **2017**, arXiv:1701.08936.
20. U.S. Government Printing Office. *Remote Sensing Data: Applications and Benefits*; Technical Report; Subcommittee on Space and Aeronautics, Committee on Science and Technology, Serial No. 110-91; 2008. Available online: https://www.govinfo.gov/content/pkg/CHRG-110hhrg41573/html/CHRG-110hhrg41573.html (accessed on 2 January 2020).
21. Everaerts, J. The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2008**, *37*, 1187–1192.
22. Bahmanyar, R.; Azimi, S.M.; Reinartz, P. Multiple vehicle and people tracking in aerial imagery using stack of micro single-object-tracking CNNs. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *42*, 163–170. [CrossRef]
23. Reilly, V.; Idrees, H.; Shah, M. Detection and tracking of large number of targets in wide area surveillance. In Proceedings of the European Conference on Computer Vision, Crete, Greece, 5–11 September 2010; pp. 186–199.
24. Meng, L.; Kerekes, J.P. Object tracking using high resolution satellite imagery. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2012**, *5*, 146–152. [CrossRef]
25. Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; Schindler, K. MOT16: A benchmark for multi-object tracking. *arXiv* **2016**, arXiv:1603.00831.
26. Kraus, M.; Azimi, S.M.; Ercelik, E.; Bahmanyar, R.; Reinartz, P.; Knoll, A. AerialMPTNet: Multi-Pedestrian Tracking in Aerial Imagery Using Temporal and Graphical Features. In Proceedings of the International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2020.
27. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468.
28. Shrivastava, A.; Gupta, A.; Girshick, R. Training Region-Based Object Detectors with Online Hard Example Mining. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 761–769.
29. Fiaz, M.; Mahmood, A.; Javed, S.; Jung, S.K. Handcrafted and deep trackers: Recent visual object tracking approaches and trends. *Acm Comput. Surv.* **2019**, *52*, 1–44. [CrossRef]

30. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng. Mar* **1960**, *82*, 35–45. doi:10.1115/1.3662552. [CrossRef]
31. Mihaylova, L.; Carmi, A.Y.; Septier, F.; Gning, A.; Pang, S.K.; Godsill, S. Overview of Bayesian sequential Monte Carlo methods for group and extended object tracking. *Digit. Signal Process.* **2014**, *25*, 1–16. [CrossRef]
32. Wang, Q.; Gao, J.; Xing, J.; Zhang, M.; Hu, W. DCFNet: Discriminant Correlation Filters Network for Visual Tracking. *arXiv* **2017**, arXiv:1704.04057.
33. Ma, C.; Huang, J.B.; Yang, X.; Yang, M.H. Hierarchical convolutional features for visual tracking. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 3074–3082.
34. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649, [CrossRef]
35. Huang, C.; Wu, B.; Nevatia, R. Robust object tracking by hierarchical association of detection responses. In Proceedings of the European Conference on Computer Vision, Marseille, France, 12–18 October 2008; pp. 788–801.
36. Lu, X.; Ma, C.; Ni, B.; Yang, X.; Reid, I.; Yang, M.H. Deep Regression Tracking with Shrinkage Loss. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; p. 17.
37. Wang, L.; Ouyang, W.; Wang, X.; Lu, H. Stct: Sequentially training convolutional networks for visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1373–1381.
38. Huang, C.; Lucey, S.; Ramanan, D. Learning policies for adaptive tracking with deep feature cascades. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 105–114.
39. Chahyati, D.; Fanany, M.I.; Arymurthy, A.M. Tracking people by detection using CNN features. *Procedia Comput. Sci.* **2017**, *124*, 167–172. [CrossRef]
40. Zhang, Y.; Wang, J.; Yang, X. Real-time vehicle detection and tracking in video based on faster R-CNN. *J. Physics Conf. Ser. IOP Publ.* **2017**, *887*, 012068. [CrossRef]
41. Okuma, K.; Taleghani, A.; De Freitas, N.; Little, J.J.; Lowe, D.G. A boosted particle filter: Multitarget detection and tracking. In Proceedings of the European Conference on Computer Vision, Prague, Czech Republic, 11–14 May 2004; pp. 28–39.
42. Brunelli, R. *Template Matching Techniques in Computer Vision: Theory and Practice*; John Wiley & Sons: Hobokon, NJ, USA, 2009.
43. Hager, G.D.; Belhumeur, P.N. Real-time tracking of image regions with changes in geometry and illumination. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 18–20 June 1996; pp. 403–410.
44. Briechle, K.; Hanebeck, U.D. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*; International Society for Optics and Photonics: Bellingham, WA, USA, 2001; Volume 4387, pp. 95–102.
45. Avidan, S. Ensemble tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 261–271. [CrossRef] [PubMed]
46. Hare, S.; Golodetz, S.; Saffari, A.; Vineet, V.; Cheng, M.M.; Hicks, S.L.; Torr, P.H. Struck: Structured output tracking with kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 2096–2109. [CrossRef]
47. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 583–596. [CrossRef]
48. Sadeghian, A.; Alahi, A.; Savarese, S. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 300–311.
49. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
50. Kalal, Z.; Mikolajczyk, K.; Matas, J. Forward-backward error: Automatic detection of tracking failures. In Proceedings of the 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 23–26 August 2010; pp. 2756–2759.
51. Lukezic, A.; Vojir, T.; Cehovin Zajc, L.; Matas, J.; Kristan, M. Discriminative correlation filter with channel and spatial reliability. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy, 22–29 October 2017; pp. 6309–6318.
52. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
53. Zheng, L.; Bie, Z.; Sun, Y.; Wang, J.; Su, C.; Wang, S.; Tian, Q. Mars: A video benchmark for large-scale person re-identification. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherland, 8–16 October 2016; pp. 868–884.
54. Yokoyama, M.; Poggio, T. A contour-based moving object detection and tracking. In Proceedings of the 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Breckenridge, CO, USA, 7 January 2005; pp. 271–276.
55. Jadhav, A.; Mukherjee, P.; Kaushik, V.; Lall, B. Aerial multi-object tracking by detection using deep association networks. *arXiv* **2019**, arXiv:1909.01547.
56. Benedek, C.; Szirányi, T.; Kato, Z.; Zerubia, J. Detection of object motion regions in aerial image pairs with a multilayer Markovian model. *IEEE Trans. Image Process.* **2009**, *18*, 2303–2315. [CrossRef] [PubMed]
57. Butenuth, M.; Burkert, F.; Schmidt, F.; Hinz, S.; Hartmann, D.; Kneidl, A.; Borrmann, A.; Sirmacek, B. Integrating pedestrian simulation, tracking and event detection for crowd analysis. In Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), Barcelona, Spain, 8–13 November 2011; pp. 150–157.
58. Schmidt, F.; Hinz, S. A Scheme for the Detection and Tracking of People Tuned for Aerial Image Sequences. In Proceedings of the ISPRS conference on Photogrammetric Image Analysis (PIA), Munich, Germany, 5–7 October 2011; Volume 6952, pp. 257–270.
59. Liu, K.; Mattyus, G. Fast multiclass vehicle detection on aerial images. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 1938–1942.

60. Qi, S.; Ma, J.; Lin, J.; Li, Y.; Tian, J. Unsupervised ship detection based on saliency and S-HOG descriptor from optical satellite images. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 1451–1455.
61. Bahmanyar, R.; Vig, E.; Reinartz, P. MRCNet: Crowd Counting and Density Map Estimation in Aerial and Ground Imagery. In Proceedings of the BMVC's Workshop on Object Detection and Recognition for Security Screenin (BMVC-ODRSS), Cardiff, UK, 9–12 September 2019.
62. Ristani, E.; Solera, F.; Zou, R.; Cucchiara, R.; Tomasi, C. Performance measures and a data set for multi-target, multi-camera tracking. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherland, 8–16 October 2016; pp. 17–35.
63. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
64. Rastogi, R.; Thaniarasu, I.; Chandra, S. Design implications of walking speed for pedestrian facilities. *J. Transp. Eng.* **2011**, *137*, 687–696. [CrossRef]
65. Finnis, K.; Walton, D. Field observations of factors influencing walking speeds. *Ergonomics* **2006**, *51*, 827–842. [CrossRef]
66. Strayer, D.L.; Drew, F.A. Profiles in driver distraction: Effects of cell phone conversations on younger and older drivers. *Hum. Factors* **2004**, *46*, 640–649. [CrossRef] [PubMed]
67. Rakha, H.; El-Shawarby, I.; Setti, J.R. Characterizing driver behavior on signalized intersection approaches at the onset of a yellow-phase trigger. *IEEE Trans. Intell. Transp. Syst.* **2007**, *8*, 630–640. [CrossRef]
68. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social LSTM: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 961–971.
69. Xue, H.; Huynh, D.Q.; Reynolds, M. SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Stateline, NV, USA, 12–14 March 2018; pp. 1186–1194.
70. Vemula, A.; Muelling, K.; Oh, J. Social attention: Modeling attention in human crowds. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbone, Australia, 21–25 May 2018; pp. 1–7.
71. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
72. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400.
73. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
74. Lin, C.T.; Chen, S.P.; Santoso, P.S.; Lin, H.J.; Lai, S.H. Real-Time Single-Stage Vehicle Detector Optimized by Multi-Stage Image-Based Online Hard Example Mining. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1505–1518. [CrossRef]
75. Koga, Y.; Miyazaki, H.; Shibasaki, R. A CNN-based method of vehicle detection from aerial images using hard example mining. *Remote Sens.* **2018**, *10*, 124. [CrossRef]
76. Huber, P.J. Robust estimation of a location parameter. In *Breakthroughs in Statistics*; Springer: New York, NY, USA, 1992; pp. 492–518.
77. Xu, Y.; Osep, A.; Ban, Y.; Horaud, R.; Leal-Taixé, L.; Alameda-Pineda, X. How To Train Your Deep Multi-Object Tracker. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Virtual, 14–19 June 2020.