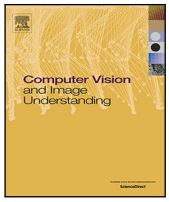
Computer Vision and Image Understanding 225 (2022) 103569

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



A multi-object tracker using dynamic Bayesian networks and a residual neural network based similarity estimator

Mohamad Saada *, Christos Kouppas, Baihua Li, Qinggang Meng

Department of Computer Science, Loughborough University, Loughborough, United Kingdom

ARTICLE INFO

Communicated by Nikos Paragios

Keywords:
Multi-object tracking
Dynamic Bayesian networks
Residual neural networks
YOLO V5
MOTChallenge

ABSTRACT

In this paper we introduce a novel multi-object tracker based on the tracking-by-detection paradigm. This tracker utilises a Dynamic Bayesian Network for predicting objects' positions through filtering and updating in real-time. The algorithm is trained and then tested using the MOTChallenge[1] challenge benchmark of video sequences. After initial testing, a state-of-the-art residual neural network for extracting feature descriptors is used. This ResNet feature extractor is integrated into the tracking algorithm for object similarity estimation to further enhance tracker performance. Finally, we demonstrate the effects of object detection on tracker performance using a custom trained state of the art You Only Look Once (YOLO) V5 object detector. Results are analysed and evaluated using the MOTChallenge Evaluation Kit, followed by a comparison to state-of-the-art methods.

1. Introduction

Multiple Object Tracking (MOT) problem, also known as Multiple Target Tracking (MTT), is a well-known problem within the computer vision field, that has had a large amount of interest in the past couple of decades within the research community, and with recent advancements in deep learning research, this interest has significantly increased. The main objective of multi-object tracking is to establish a correlation between objects locations in a video sequence or in a sequence of consecutive images. In other terms its main job is to recognise objects individually and keep track of their location over time as in Fig. 1.

MOT plays an important role in many real world applications, such as automated security surveillance systems for monitoring people on large scale CCTV networks, video based sports analysis where athlete performance monitoring depends on successful identification and tracking of multiple athletes in real time, also human–robot interaction, where interaction is based in dynamic environments with multiple moving targets, and also other applications with non-human objects, such as tracking cars from UAV video feeds, and tracking different animal species from video sequences, these are but a few of many real world applications that demonstrate the importance of this computer vision problem.

Over the years the MOT problem has been approached from many different angles, but one of the most prominent approaches to solve it is based on the Tracking-by-Detection paradigm. In this paradigm the problem is divided into two sub problems, at first the target objects are detected within the current frame, then using an association criterion utilising object detections from the current and previous frames, the tracker decides which objects are new and which are tracked from previous frames. In this paradigm the tracking can be performed on individual frame basis or using several consecutive frames also known as batch processing. Batch processing is deemed more accurate as it uses the information from several frames before deciding, but this comes at the expense of running offline.

1.1. Common challenges

The MOT problem faces many of the familiar challenges that are encountered by many other vision-based algorithms, some of the most prevalent challenges facing the MOT problem are:

- Partial occlusion: is when the tracked object is partially occluded by other moving objects in a crowded scene, or by stationary obstacles blocking the view between the camera and the object.
- Full occlusion: or also known as disappearance is when the object is physically occluded by other objects or stationary obstacles. It is also worth noting that in the tracking-by-detection paradigm this could simply occur when the object detector does not detect the object (false negative).
- Illumination changes: is when the image illumination changes drastically between frames, this could be the result of changing light in the environment (e.g. sunlight and overcast), or due to the object's new location (from light to shadow or vice versa), or due to camera exposure changes, etc...

* Corresponding author.
E-mail address: M.Saada@lboro.ac.uk (M. Saada).

[1] https://motchallenge.net/.

https://doi.org/10.1016/j.cviu.2022.103569
Received 27 July 2021; Received in revised form 11 August 2022; Accepted 26 September 2022
Available online 30 September 2022
1077-3142/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Fig. 1. Object detection vs Multiple object tracking performed on testing images from Ferryman and Shahrokni (2009) and Leal-Taixé et al. (2015).

- Colour changes/Colour response: This could be as a side effect of illumination change, or as a result of different colour response between different image sensors, and this is mainly in scenarios where objects are tracked over multiple camera systems with different colour response (Security surveillance network, sports event analysis)
- Pose and shape variation: this could be a major factor in tracking systems that use human shape analysis as a way for associating objects over multiple frames, as any change in human posture or a variation of heading in respect to the camera can affect the tracking
- Similar appearance: this is critical particularly in highly crowded scenes with multiple objects having very similar appearance interacting with one another, as it can easily cause switching for tracker relying on appearance models
- Non-linear motion: this could affect trackers that are built around specific linear motion models, which might not be the case for many real life application scenarios
- Real-time vs offline batching: There is a subset of tracker algorithms that analyse the video/image sequences in groups(batches) at a single time to make a more accurate association and thus better tracking of objects, check Fig. 2, this could be ideal for a some type of methods and application scenarios, but many systems require real time execution, and this has a limiting effect on the types of methods used. Those are often chosen according to their computational cost and running times.

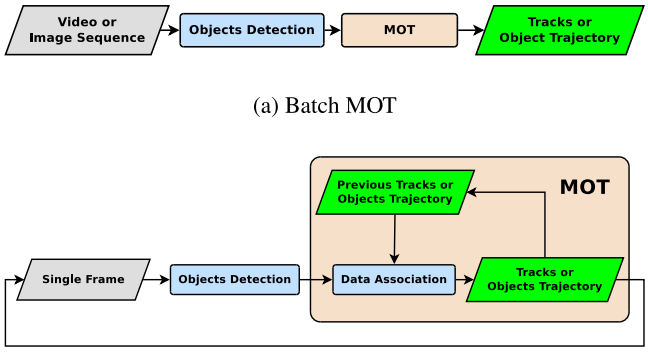


Fig. 2. MOT Online vs. Offline (Batch processing).

1.2. Related work

Most object tracking algorithms constructed using the tracking-by-detection paradigm focus mainly on the second stage of that process. Where one of many different data association methods are used to establish an association between detected objects and existing tracks across frames. Most of these methods can be to one of the following approaches:

Probabilistic methods

In the probabilistic methods approach, object states are treated as variables of a probability distribution, which the tracker through utilising one of many different inference techniques seeks to estimate the conditional probability distribution over the target states given a set of observations (e.g. detections and affinity information extracted from current frame). This type of approach is suitable for online tracking, as it only requires data from current and previous frames.

Some of the most notable implementations of probabilistic methods is the use of Kalman Filters. With a Kalman filter the objects are modelled as hidden variables of a linear dynamic system, where a Kalman filter is used to estimate their values given previous estimation and current observations. Such methods can be found in Bewley et al. (2016) and Wojke et al. (2017). Similar to Kalman filters, Extended Kalman filter are used to model object as states in a nonlinear system, examples of extended Kalman filter implementation for MOT are Kamen (1990) and Santosh and Mohan (2014).

Particle filters on the other hand, give up any predefined assumption about the probability distribution and instead represents the probability distribution as a number of weighted particles, it is particularly effective in modelling and estimating the posterior probability distribution of nonlinear stochastic noisy systems. There have been many particle filter implementation for MOT, here are some note worthy ones (Yang et al., 2005; Jin and Mokhtarian, 2007; Breitenstein et al., 2009; Hess and Fern, 2009).

There are other examples of Bayesian based methods, where models are used for representing different objects as states in a dynamic system.

These approaches try to maximise the posterior probability distribution of the object states at any given frame, given observations from current and previous states. This type of approach is largely similar to previous approaches and is mostly suitable for online tracking applications, examples of these can be found in Kratz and Nishino (2012), Yoon et al. (2015) and Pavlovic et al. (1999).

Optimisation methods

The other most notable approach for handling data associations in MOT is optimisation-based methods. Where the tracking problem is formulated as an optimisation problem, which is represented by a cost function. Here the problem is solved by choosing parameters that minimise or maximise the cost function depending on the formulation. One of the main benefits of optimisation techniques is that they are more computationally efficient and can be globally convergent. It is worth noting that optimisation methods are generally more suitable for offline tracking because they require observations from at least a few frames to work. In certain cases, this has the added benefit of running globally on all frames, which guarantees achieving best global association of all observations of a single object to a single track over the period of tracking. Generally, optimisation techniques can be split into two classes, depending on the number of frames they require.

Global optimisation: where entire sequences of frames (batches) are used for global object trajectory optimisation. Examples of these can be viewed in Butt and Collins (2013), Zhang et al. (2008), Lenz et al. (2015), Le et al. (2016), Chen et al. (2016), Yang et al. (2011), Yang and Nevatia (2012), Chen et al. (2014), Milan et al. (2015), Park et al. (2015) and Zhang et al. (2021).

Local optimisation: where short sequences of frames (frames window) are used for local object trajectory optimisation. In Zhong et al. (2014) researchers introduced a bipartite graph matching method to solve the MOT problem in dynamic environments. The nodes of the introduced bipartite graph represent objects in two neighbouring frames, the edges between these nodes represent the degree of similarity between the objects in these frames. The problem is then treated as a maximal matching problem that is solved with a dynamic Hungarian algorithm. Candidate occlusion regions are proposed based on overlapping between bounding boxes of the interacting objects, this help in dealing with occlusion problems.

Similarity measures

The object tracking process in its simplest forms is split into detection and data association. The data association on its own can never yield the best results, particularly in high uncertainty scenarios, such as multiple object occlusions and long periods of interactions between tracked objects.

This made it a necessity for tracking algorithms to have the ability to recognise objects after full/partial occlusions, or long periods of interaction. This ability comes in the form of different affinity measures, which are designed to calculate the similarity between objects in different frames using specific criteria, below are some of most widely used affinity measures in MOT algorithms.

Appearance models

Appearance models are an affinity computation method that extracts a visual appearance descriptor from detected objects and uses these descriptors to measure similarity to tracked objects appearance models to match them across different frames. There are many different types of appearance models, and most of these types focus solely on the MOT problem for tracking human objects. This also plays a critical role in person re-identification algorithms, where appearance models are used to re-identify human objects that were occluded or left the frame for a period. Some of the appearance models tailored specifically for the MOT problem use local features (Farenzena et al., 2010), global features (Liu et al., 2012), body based descriptors (Khan et al., 2010; Zheng et al., 2011), and many other types of appearance models (Riahi and Bilodeau, 2016; Huang et al., 2016; Wu et al., 2021) .

Motion models

Motion modelling encapsulates the object's dynamic behaviour over time and is considered one of the earliest core techniques used with object tracking. This has transpired in motion modelling becoming and remaining one of the most comprehensively researched techniques of affinity measures for object tracking. Motion modelling works under the assumption that moving objects tend to move in consistent patterns, rather than erratic random patterns. In simple terms motion models consider an object's motion patterns and attempt to predict its next location depending on previous motion observations. This helps drastically in reducing the search space. Most motion models used for MOT fall into linear motion models that assume an object's speed is linear over short time periods, therefore the tracked object velocity can be calculated using first order linear regression. Or non-linear motion models that do not assume that motion is linear, which is more agreeable with many application scenarios, where non-linear motion models tend to predict motion more accurately, example of motion models are Gwak (2017), Yin et al. (2011). Some MOT research focus on the type of environment, such as tracking in traffic environments (Jiménez-Bravo et al., 2022), or tracking in high density crowds (Stadler and Beyerer, 2022).

2. Multi-object tracker using dynamic Bayesian networks

The MOT problem is normally presented as a data association problem, between multiple object observations over a sequence of consecutive frames. In this section a new tracking framework for solving the MOT problem is introduced. This framework is designed in line with the tracking-by-detection paradigm, with a focus on online tracking rather than offline batch processing. Where in an online tracker setting, the main task of a tracker is to compute an association between observations of detected objects in the current frame, and tracked objects from previous frame/s. At the heart of this proposed tracking framework is an online object state predictor designed using a Dynamic Bayesian Networks (DBN) model. The newly introduced model's parameters are then estimated using the Expectation Maximisation (EM) algorithm.

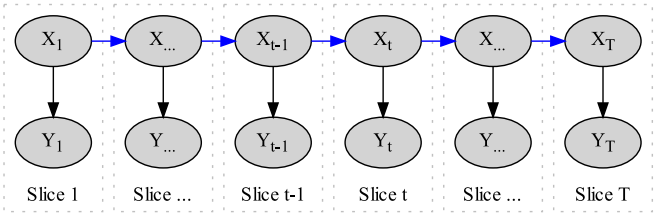
2.1. BNs and DBNs - an overview

Bayesian networks are a special type of directed graphical model, this type is called a direct acyclic graph or for short a DAG. DAGs are directed graphical models that do not have a cycle in the graph's structure. Dynamic Bayesian Networks (DBNs) are an extension of Bayesian networks that model domains within a temporal perspective. In essence they are Bayesian networks extended over time.

Both BNs and DBNs can be used to model different domains for the purpose of reasoning over uncertainty. They have been used extensively to problematically model systems in a variety of domains. Examples of such applications can be seen in fields of medical diagnosis, human activity recognition, industrial process modelling, fault prognosis, anomaly detection, reliability prognosis, plan recognition, and more recently a newer form of Bayesian Neural Networks has been used to combine Bayesian inference with neural networks to achieve a new form of powerful modelling framework. Example of these implementation can be found in Saada and Meng (2012), Cai et al. (2022), Saada et al. (2014), Cai et al. (2021b), Marini et al. (2015) and Cai et al. (2021a).

2.2. DBN variables

In the tracking-by-detection paradigm, object detections are presented as bounding boxes (bbox) in the image coordinate space. These bounding boxes along with the images themselves, are the only data used by the trackers to compute the association between the detected objects over frame sequences. The goal behind the proposed DBN model



MOT DBN model representation

Fig. 3. MOT DBN model representation.

Table 1
Multiple object tracker DBN model variables.

Variable	Description
$x_{c_{bb}}$	x coordinate of bbox centroid
$y_{c_{bb}}$	y coordinate of bbox centroid
A	Area of bbox
R	Aspect ratio of bbox height to width
$x'_{c_{bb}}$	Rate of change of x coordinate of bbox centroid
$y'_{c_{bb}}$	Rate of change of y coordinate of bbox centroid
A'	Rate of change for bbox area
R'	Rate of change for aspect ratio of bbox height to width

(Fig. 3), is to provide a Probabilistic Graphical Model (PGM) base that will be used along with available data to estimate the model's parameters. They will be then deployed to predict object state over consecutive time frames. Bounding box data is typically denoted by a variety of different variables, which all convey the same information, an example of this is: $\boldsymbol{bbox} = \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{width}, \boldsymbol{height}$ Where: $\boldsymbol{x}, \boldsymbol{y}$ are the $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates of the top left vertex of the bounding box and $\boldsymbol{width}, \boldsymbol{height}$ are the width and height of the bounding box Some representations include the coordinates of the top left and the bottom right vertices. Another important variable for tracking bounding boxes is the centroid of the box, which is the point at the centre of the box. This is represented by

$$\boldsymbol{C}_{bb} = (\boldsymbol{x} + \frac{\boldsymbol{width}}{2}, \boldsymbol{y} + \frac{\boldsymbol{height}}{2}) \tag{1}$$

The proposed DBN model is built using a set of 8 different variables, these can be found in Table 1. Those variables are chosen as they are good at representing the main attributes of the bounding boxes, which in turn represent the position of detected objects. It is worth noting that these only convey the basic location attributes of a detection, and thus do not convey behavioural attributes such as motion, or visual attributes, such as colour and shape. Those will be addressed later in the paper.

2.3. Structure and parameter estimation

For the tracked objects state-space in the MOT problem to be modelled as a DBN, it is assumed that objects states are part of a dynamical system that is of a linear nature over small periods of time. This allows the state space of the tracked objects to be treated as a linear Gaussian state-space model, which enables it to be represented as a DBN. In most instances linear Gaussian state space models can be represented by the following equations.

$$\boldsymbol{X}_{t+1} = \boldsymbol{A}\boldsymbol{X}_t + \boldsymbol{w}_t \tag{2}$$

$$\boldsymbol{Y}_t = \boldsymbol{C}\boldsymbol{X}_t + \boldsymbol{v}_t \tag{3}$$

Where $\boldsymbol{A}$ and $\boldsymbol{C}$ are general form matrices. $\boldsymbol{X}_t$ denotes the latent (hidden) variables in the model, they represent states of the system, and here are represented by:

$\boldsymbol{X}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{x}'_t, \boldsymbol{y}'_t, \boldsymbol{a}'_t, \boldsymbol{r}'_t)$ which represents location, area, ration and each of their velocities respectively.

In Eq. (3) $\boldsymbol{Y}_t$ represents the observable states in the system, and is assumed to be a linear Gaussian function of $\boldsymbol{X}_t$. The transition function of the hidden states of the system between different time steps is also considered to be a linear Gaussian function. $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ represent Gaussian noise that affects both hidden and observable states of the system respectively. They are normally distributed random variables with covariance matrices Q and R respectively. In the model it is assumed that only observable states of the system $\boldsymbol{Y}_t$ are directly measurable, and that the hidden states $\boldsymbol{X}_t$ and the noise variables $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ are not.

In the MOT problem objects are normally moving in the image coordinate space $\in \mathbb{R}^2$ if the tracking is in 2D space and not 3D space. Now the problem is dealt with as an online filtering problem (prediction and update), which is solved by recursively predicting the belief state of the dynamical system by applying Bayes rule.

$$\begin{gathered} \boldsymbol{P}(\boldsymbol{y}_t|\boldsymbol{X}t) = \frac{\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{y}_{1:t})}{\left[\sum_{\boldsymbol{x}_{t-1}} \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{x}_{t-1})\boldsymbol{P}(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})\right]} \\ \Downarrow \\ \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{y}_{1:t}) = \boldsymbol{P}(\boldsymbol{y}_t|\boldsymbol{X}t)\left[\sum_{\boldsymbol{x}_{t-1}} \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{x}_{t-1})\boldsymbol{P}(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})\right] \end{gathered} \tag{4}$$

Note that in Eq. (4) $\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{y}_{1:t-1})$ can be calculated from $\boldsymbol{P}(\boldsymbol{X}_{t-1}|\boldsymbol{y}_{1:t-1})$ because the latent states of the DBN comply to Markov restriction. This is also applies to the observable states of the system where $\boldsymbol{P}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{y}_{1:t-1})$ is equal to $\boldsymbol{P}(\boldsymbol{y}_t|\boldsymbol{X}_t)$.

In this approach the latent states of the system $\boldsymbol{X}_t$ are predicted for each time step $\boldsymbol{t}$. Those predictions are used along with other methods to compute the data association between the current observation and previously initiated tracks. Once the data association is complete, the new states values are used to update the belief state of the system, this new belief state is then used along with observations from next time step to predict the belief state of the system in that time step, applying the same Bayes rule in Eq. (4), this is done continuously in a predict and update loop. It is worth noting that normally velocity values cannot be measured directly from the system but can be inferred periodically through object position readings from consecutive time slices.

After the structure of the model has been set, the next step is to estimate the parameters of the system. Parameter estimation or density estimation for variables of the system involves finding the Probability Density Function (PDF) that best represents the joint probability distribution of the system's data. In essence it is a function that represents the joint probability distribution that best explains the system's data. A popular method to accomplish this in statistics is using Maximum Likelihood Estimation (MLE). MLE is a method of estimating the variables of a probability distribution through maximising a likelihood function. Given the defined model, this likelihood function ensures that the observed data is the most likely (highest probability) to occur.

The main problem with MLE is that it needs the training data to be complete, and that all variables of the system are observable, otherwise MLE becomes intractable. Unfortunately, this is not the case with the proposed DBN model since it contains hidden/latent states. Luckily there is an algorithm called the EM algorithm, which enables performing MLE on systems with latent states.

The EM algorithm that enables MLE in models with latent variables. The EM algorithm works in repetitive order, firstly it estimates the latent variables of the system in a step called estimation or expectation. Then it optimises the parameters of the model to maximise the likelihood estimate found in previous step, so that the probability density function found along with the model best explains the training data.

- **E-Step:** Expectation or estimation, where the latent parameters of the model are estimated, through computing the expected log likelihood.
- **M-Step:** Maximisation, where it optimises the parameters of the probability distribution in order to maximise the log likelihood computed in the E-Step

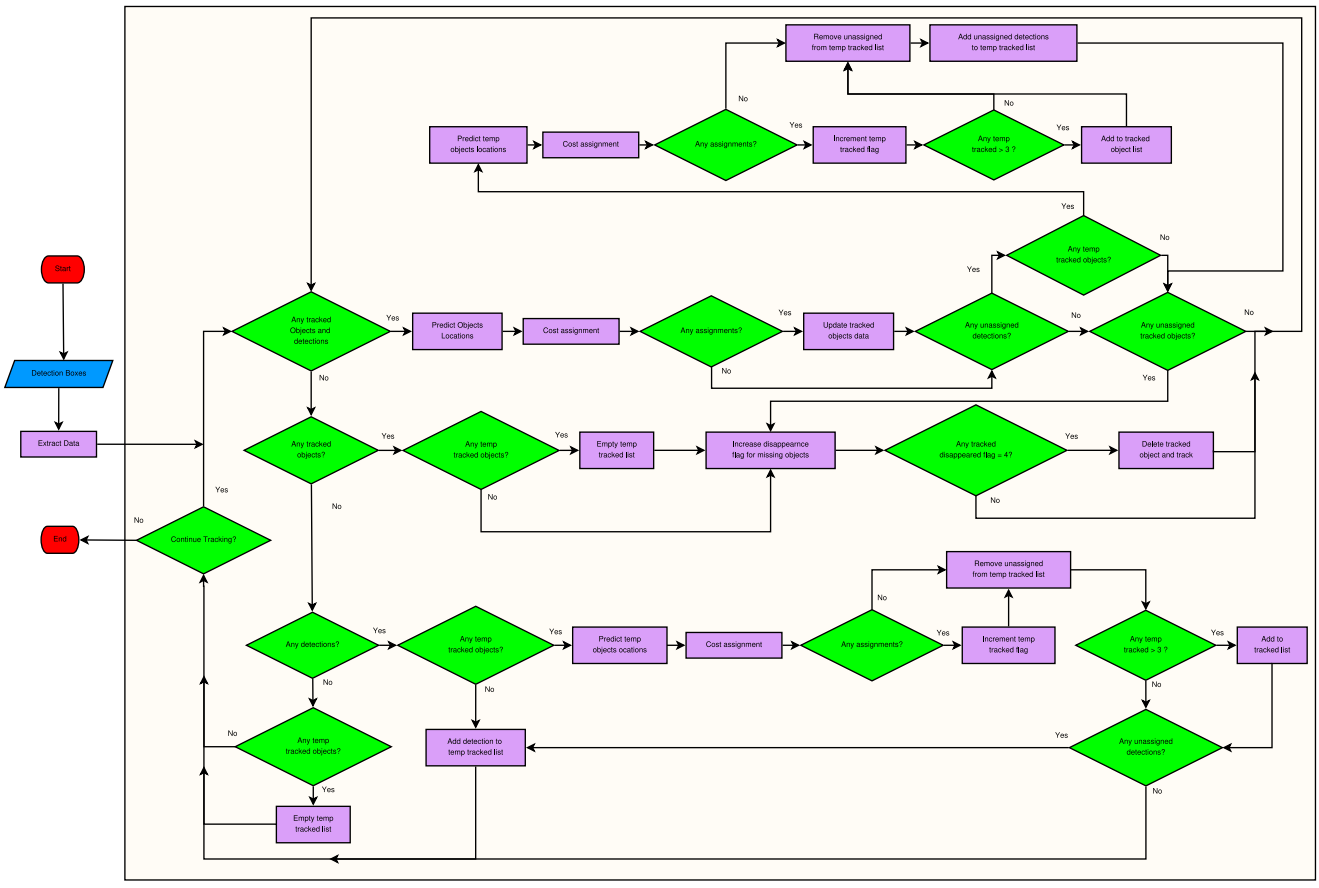


Fig. 4. DBN MOT tracker flowchart.

The EM algorithm repetitively goes through the two aforementioned steps (E and M), until convergence. For a further in depth overview of parameter estimation for a linear dynamical model, using the EM algorithm, refer to Ghahramani and Hinton (1996) for the mathematical proof.

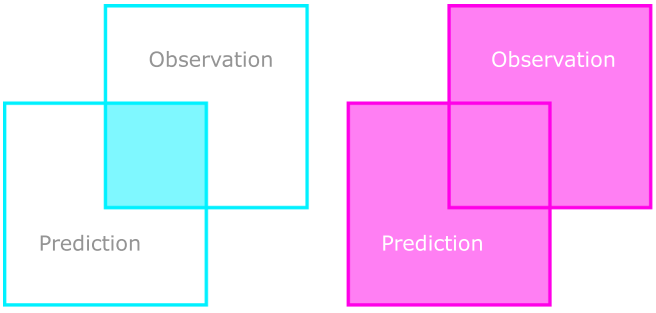
2.4. State prediction and data association

Now that the parameters of the model are estimated, the model and its parameters can be used to predict the positions of currently tracked objects in the current frame. In Fig. 4 a flowchart of the entire algorithm can be seen. At the start of tracking, there are no previous tracks or tracked objects, as a result all detect objects in the first frame are issued new object ids and assigned a track. Now data from those objects are used to update the belief state of the system, which allows using inference to predict the location of these object in the next time step.

As mentioned earlier in this paper, inferencing in linear Gaussian state-space models is achieved by filtering/smoothing. This is mainly done by utilising an inferencing algorithm, which computes the marginal probability distributions of all latent (hidden) states of the system, given current and past observation sequences. Where the aim is to calculate $\mathcal{N}(X_t|y_{1:t-1})$ for all latent states of the system $X_t \in \{X_1, \ldots, X_T\}$. This can be calculated from $\mathcal{N}(X_{t-1}|y_{1:t-1})$ because the latent states of the DBN comply to Markov's restriction.

$$\mathcal{N}(X_{t=i}|y_{1:t}) = \alpha_t(i) \propto \mathcal{N}(y_t|X_{t=i}) \sum_j \mathcal{N}(X_{t=i}|X_{t-1} = j)\alpha_{t-1}(j) \tag{5}$$

Now that all states in each time slice t have been predicted using the belief state of the system in time slice $t-1$ and observations in slice t, those predicted states need to be matched to existing tracks currently held by the tracker or initiated as new objects with new id if no association can be made.



(a) Intersection (b) Union

Fig. 5. Intersection over Union (IoU).

For data association an Intersection over Union (IoU) metric is used (also known as Jaccard index) see Fig. 5, this metric is normally utilised in many popular computer vision problems, such as segmentation, object detection and tracking.

This measure is mainly used here for detecting how accurate the predicted bounding boxes in matching any existing observations bounding box. Typically this is done by dividing the intersecting area between two bounding boxes by the their union area as in Eq. (6).

$$IoU_{distance} = \frac{bbox_{prd} \cap bbox_{obs}}{bbox_{prd} \cup bbox_{obs}} = \frac{Intersection}{Union} \tag{6}$$

Table 2
Proposed DBN based MOT performance evaluation on MOT15/MOT16.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
OVERALL MOT15	**25.97**	**72.50**	**30.97**	**40.52**	**25.06**	**44.89**	**72.57**	**17 915**	**6770**	**21 990**	**12.80**	**39.80**	**47.40**	**64**	**199**	**237**	**1.23**	**1174**	**26.15**	**780**	**17.4**
OVERALL MOT16	**24.91**	**77.88**	**29.72**	**45.01**	**22.18**	**37.59**	**76.27**	**41 503**	**12 916**	**68 904**	**7.93**	**39.65**	**52.42**	**41**	**205**	**271**	**2.43**	**1493**	**39.72**	**1090**	**29**

The IoU distance used to determine the assignment are counter intuitive, as for a better assignment the IoU distance must be larger and closer to **1**. If the bounding box of the predicted object aligns completely (100%) with the bounding box of one of the detections, the IoU distance is **1**, as intersection is equal to the union in this case. If the two bounding boxes do not intersect the IoU distance is **0**.

After calculating all the IoU distances for all predicted objects with all tracked objects, an IoU distance matrix of size $\boldsymbol{P} * \boldsymbol{O}$ is formed, where P is the number of predicted objects and O is the number of detected (observed) objects. It is worth noting that this is a rectangular matrix, in case both $\boldsymbol{P} \neq \boldsymbol{0}$ & $\boldsymbol{O} \neq \boldsymbol{0}$. After forming this cost matrix, a combinatorial optimisation algorithm is used to solve the assignment problem. The algorithm is called the Hungarian algorithm (Kuhn, 1955) (also known as the Munkres assignment algorithm and the Kuhn–Munkres algorithm), the formulation for this algorithm can be seen below.

- $\{c_{po}\}_{NxN}$ IoU cost matrix, where $\boldsymbol{C}_{po}$ is the IoU distance between predicted object $\boldsymbol{p}$ and observed detection $\boldsymbol{o}$.
- $\{\boldsymbol{m}_{po}\}_{NxN}$ One shot output binary matrix, where $\boldsymbol{m}_{po} = \boldsymbol{1}$ iff detection $\boldsymbol{o}$ is assigned to prediction $\boldsymbol{p}$
- $\sum_{p=1}^{N} \boldsymbol{m}_{po} = \boldsymbol{1}, \forall \boldsymbol{o} \in \{\boldsymbol{1}, \ldots, \boldsymbol{N}\}$ Only assign one detection to one prediction
- $\sum_{o=1}^{N} \boldsymbol{m}_{po} = \boldsymbol{1}, \forall \boldsymbol{p} \in \{\boldsymbol{1}, \ldots, \boldsymbol{N}\}$ Only assign one prediction to one detection
- $\sum_{p=1}^{N} \sum_{o=1}^{N} \boldsymbol{c}_{po} \boldsymbol{m}_{po} \rightarrow \boldsymbol{min}$ Minimise total cost function

Since the IoU measure is considered better when bigger, the Hungarian algorithm will be used to maximise the cost assignment rather than minimising it.

After assigning detected objects to predicted objects, the belief state of the system is updated using the values from the assignment. This belief state of the system is then used again for predicting the latent states of the system for the next time step, this process continues repeatedly in a predict and update fashion if the tracker is running. As for the objects that did not match in the assignment matrix, if the predicted states failed to match a detected observation, it could indicate that the object was occluded by another object or obstacle, or the object has not been detected by the object detector, or it could simply mean that the object has left the scene. The object's *not_detected* flag is incremented by **1** and state information remain the same as they were when the object was last tracked. If the object continues not to be assigned, a *max_disappearance* flag is set to a maximum number of **4** frames, after which the object and its track are deleted. If the object is assigned before the *max_disappearance* flag is reached, the *not_detected* flag is reset to **0**. If the object appears after it has been deleted (e.g., after long periods of occlusion), it will be treated as a new object and issued a new object id and track. As for the detections (observations) that did not match any predicted states, they are treated as possible new objects and added to the *temporary_object* list and their *temp_object* is increased by **1**, once the object has been predicted and detect **3** times, it is treated as a tracked object and issued a new object id and track. This is done to minimise detecting already tracked objects as new objects if the assignment fails for a frame or two.

2.5. Testing

For testing the proposed DBN based MOT, the MOT15 (Leal-Taixé et al., 2015) and MOT16 (Milan et al., 2016) benchmark data sets are used. The benchmark datasets are an effective way to compare the performance of new algorithms against state of the are techniques. The proposed algorithm takes as input a CSV text-file containing detection data, each line representing a single detection instance in a specific frame in the following format:

<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>

each frame can have multiple lines in the case of multiple objects being detected, or no lines in case no objects were detect. It is worth noting that the provided detection files do not contain *id* values for detected objects.

The output is in the same format as the detection files, the only difference is that there are *id* values generated by the algorithm for all tracked objects.

After running the algorithm on the provided benchmark data sets, the results are then put through the MOTChallenge benchmark portal. The MOTChallenge then returns the final evaluation results based on its comparison of the provided algorithm results to the ground truth data according to evaluation metrics set in their website (Leal-Taixé et al., 2015). The algorithm has been run on all data sets within the MOT15 (Leal-Taixé et al., 2015) and MOT16 (Milan et al., 2016) benchmarks, the results are then analysed through the dev kit and the final performance evaluation results are shown in Table 2.

3. Object similarity estimation using residual neural networks

As can be seen from the evaluation results of testing in the previous section in Table 2, ID switch (IDSW) and false negative (FN) numbers were considerably high and recall was considerably low compared to precision. The main reason behind this is that the DBN based proposed algorithm favours high certainty scenarios, In other words, it performs poorly whenever there is high uncertainty in tracking multiple targets, this is particularly true in scenarios when the tracked object (person) is either interacting with other objects in the frame for long periods of time, or is occluded by other objects for long periods of time. This is mainly because the proposed tracker relies completely and only on location and size information extracted from the bounding boxes of the detections in the frame sequences. To deal with this issue an extra affinity measure is added to the algorithm to enhance its ability to perform better tracking in high uncertainty scenarios. This is done using similarity detection, to recognise the objects by means of visual features, as well as by information from bounding box detections.

This enables the algorithm to decide with higher certainty in scenarios where previous DBN tracker failed, as it can use a similarity measure to decide whether the objects are the same or not. To compute unique features from the image, a feature extractor is created utilising a residual neural network structure known as (ResNet18). The feature extractor encodes object appearance as a low dimensional feature descriptor, which is resilient against an object's appearance variations between consecutive frames. This feature extractor takes in the object's image and returns the descriptor as a feature vector of real numbers. This feature vector encapsulates all the interesting information that makes the object identifiable, which is then used as a unique numerical fingerprint of the object's image and compared to other extracted feature descriptors to decide if they belong to the same object or different ones.

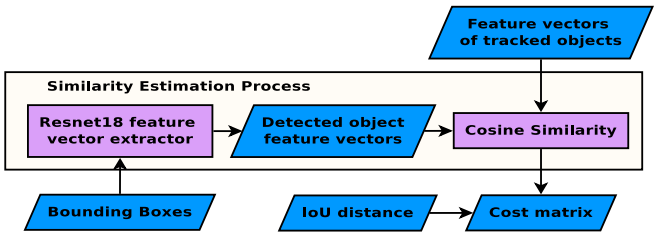


Fig. 6. Similarity estimation process.

3.1. Structure

In He et al. (2016) researchers introduce a new novel residual learning framework that makes it easier to train deeper neural networks, these residual neural networks are easier to optimise and can achieve more accuracy with considerably increased depth. Therefore, this type of networks is chosen to make the bases for the feature extractor used in the similarity comparison by the DBN tracker. The ResNet-18 is originally used as a classifier with a pre-trained variant trained on the ImageNet dataset using more than a million images, with 1000 different predefined classes. For the purposes of feature extraction, after the network is trained on the MARS dataset (Zheng et al., 2016), the fully connected layer is removed and **512** long output features are combined in a vector and used as the feature descriptor utilised by the tracker to measure the similarity between the tracked objects.

After extracting the feature descriptor using the residual neural network, this value is saved for all tracked objects, and is updated continuously. After calculating the IoU distance matrix for all predicted and detected objects. The similarity between all pairs of objects (predicted and detected) is calculated using the cosine similarity measure. The cosine similarity measure is a method to determine how similar two vectors are. From a mathematical perspective a cosine similarity function computes the cosine of the angle between two vectors in a multi-dimensional space, The similarity is at its most when the angle is closer to zero. In Eq. (7) the mathematical formula for calculating the cosine similarity is written down. This measure works for all vectors $\boldsymbol{A} = \{a_1, a_2, \ldots, a_n\}$ and $\boldsymbol{B} = \{b_1, b_2, \ldots, b_m\}$, with values $\in \mathbb{R}$, as long as they are of the same size $\boldsymbol{n = m}$.

$$\text{cosine\_similarity} = \cos(\theta) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\| \|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{7}$$

As with the IoU distance, the cosine similarity is a value between **0** and **1**, the closer the values are to **1** the higher the similarity. In order to combine the two measures in a single cost assignment, we use a weighted sum approach where the wights $\boldsymbol{w_1, w_2} \in \boldsymbol{[0, 1]}$ and $\boldsymbol{w_1 + w_2 = 1}$.

The new cost distance is written in Eq. (8).

$$\boldsymbol{Cost_{distance} = w_1 * Similarity_{distance} + w_2 * IoU_{distance}} \tag{8}$$

Where: $\boldsymbol{Cost_{distance} \in [0, 1]}$

The new $\boldsymbol{Cost_{distance}}$ is the weighted sum of both similarity and IoU distances, and the $\boldsymbol{Cost_{distance}}$ is closer to **1** when the prediction and observation are more likely to be the same object and is closer to 0 when they are not. Since $\boldsymbol{w_1 + w_2 = 1}$ it indicates that the weights give more importance to one measure over the other if they are not equal. In the case of $\boldsymbol{w_1 = w_2 = 0.5}$ both distances have the same importance. In practice the values $\boldsymbol{w_1 = 0.4}$ and $\boldsymbol{w_2 = 0.6}$ were found after experimentation to yield the best result in most cases. The similarity estimation process flowchart can be seen in Fig. 6.

3.2. Training

For training the feature extractor residual neural network, the ResNet-18 structure is used for its high performance and light weight.

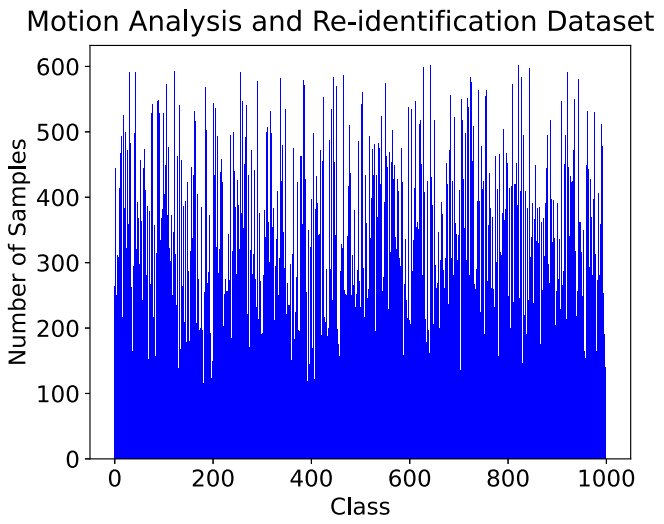


Fig. 7. MARS training data.

In many implementations the ResNet-18 is used as a classifier that is trained using the ImageNet dataset. For the specialised purpose-built feature extractor, the ResNet-18 structure is instead trained on the MARS dataset for person re-identification (Zheng et al., 2016). The MARS dataset is an extension of the Market1501 (Zheng et al., 2015) dataset and is provided for the main purpose of person re-identification implementations, it has 1261 different IDs and 20,000 tracklets, and it is one of the largest available datasets for person re-identification.

The MARS dataset was recorded using footage from 6 different cameras, 5 $\boldsymbol{x}$ HD (1920 × 1080) and 1 $\boldsymbol{x}$ SD (640 × 480) cameras. The data represents images of 1261 different pedestrians that were recorded by at least 2 of the 6 cameras see Fig. 8. The data set has **1191003** images in total, it also contains **3248** distractors to make the data more realistic. For the purposes of training the residual neural network, only half of these images are taken **583370**, which represent exactly **1000** different pedestrians each having between **100** and **600** images, see Fig. 7.

Before images are used for training they are scaled to a range of [0, 1] and then normalised using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. Images used in training are resized and cropped to become (224*224) width and height. The dataset was split **70%/30%** respectively for training and validation, a batch size of **256** was used, the training ran for **90** epochs. See Fig. 9 for training results of the model on the MARS dataset.

After training the residual neural network, the trained model is used to extract the feature vector, this is done by extracting the features after *avg pool*. A sample of two feature vectors can be seen in Table 3. The cosine similarity measure is calculated for those two feature vectors and is equal to 0.9649, which indicates that they are similar and probably originating from either two very similar looking objects, or the same object.

3.3. Testing

The ResNet18 feature extractor is incorporated into the proposed tracking algorithm, along with cosine similarity comparison functionality. The algorithm is then run again on all the data sets in the MOT15 (Leal-Taixé et al., 2015) and MOT16 (Milan et al., 2016) benchmarks. As before the algorithm takes as input the CSV text-file of detection instances and outputs the data for the tracked objects. The results are then inputted into the MOT Challenge Dev Kit where performance evaluation results are generated. It is worth noting, albeit the detections are manually fed in bulk to the proposed tracker. The feature extractor, along with the proposed tracking algorithm are run in parallel in real time. The performance evaluation results are displayed in Table 4.

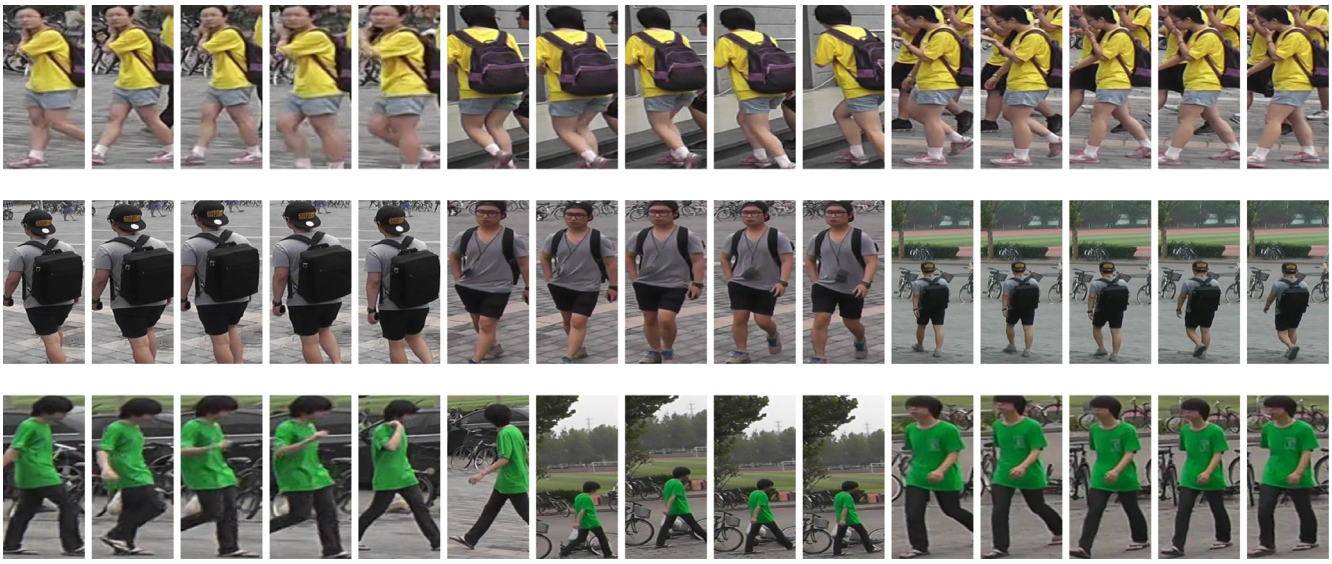


Fig. 8. MARS Sample of 3 different pedestrians (Zheng et al., 2016).

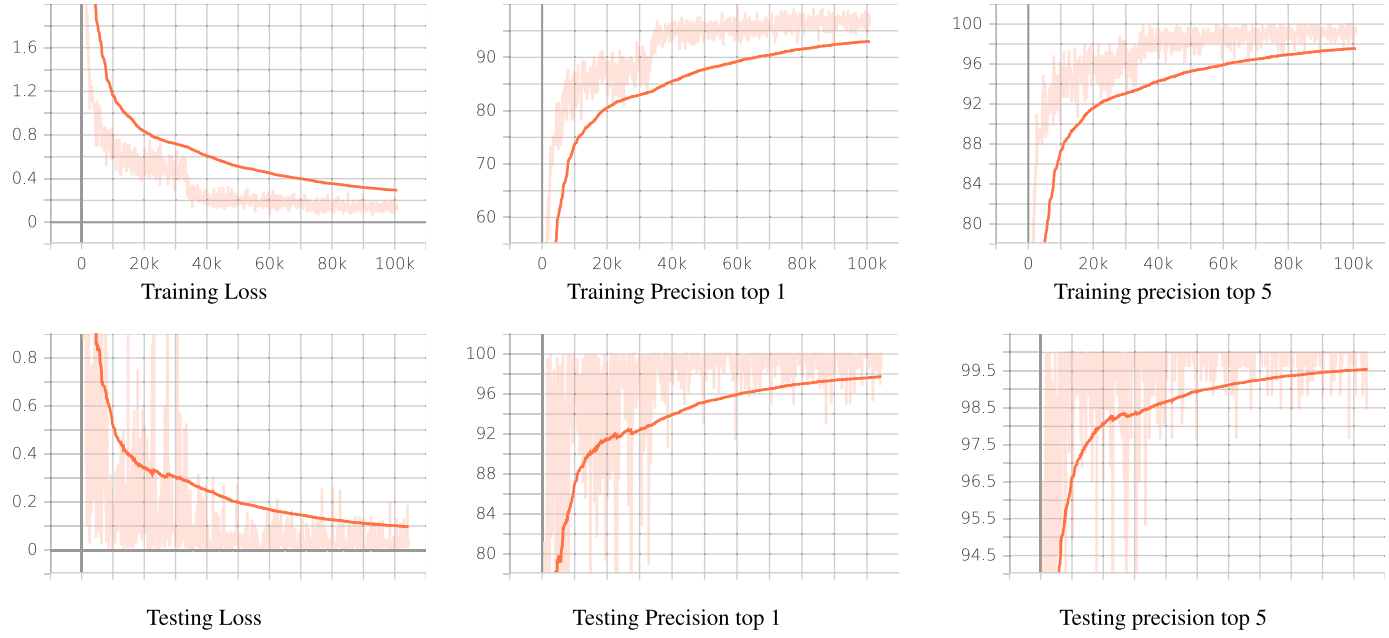


Fig. 9. Training Loss.

Table 3
Feature vectors for two similar pedestrian images from MARS.

Vector	1	2	3	...	510	511	512
Vector1	1.3798e−01	1.7302e+00	3.2078e−01	...	3.9611e−01	1.8910e+00	1.1030e−01
Vector2	2.5101e−01	3.3838e+00	5.8297e−01	...	2.7222e−01	1.4076e+00	2.6565e−01

Table 4
DBN MOT performance evaluation on MOT15/MOT16 using similarity estimation.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
OVERALL MOT15	**30.02**	**72.94**	**41.42**	**66.14**	**30.15**	**38.07**	**83.53**	**15193**	**2996**	**24712**	**11.40**	**29.80**	**58.80**	**57**	**149**	**294**	**0.54**	**833**	**21.88**	**218**	**5.7**
OVERALL MOT16	**29.11**	**78.54**	**36.91**	**73.82**	**24.60**	**31.33**	**93.99**	**34585**	**2212**	**75822**	**5.61**	**35.01**	**59.38**	**29**	**181**	**307**	**0.42**	**1187**	**37.89**	**239**	**7.6**

4. Object detection and MOT performance

It is all in the name, of the paradigm that is. It is called the tracking-by-detection paradigm because the entire tracking algorithm is based on the detections provided, hence the performance of any tracker built around this paradigm is largely reliant on the performance of the object detector that provided these detections in the first place.

One of the main strong points of the proposed DBN tracker in this work, is its ability of real-time performance, in this section a custom trained state of the art object detector (You Only Look Once

Table 5
A simplified view of a YOLO V5 structure (Jocher et al., 2021).

	layer	input_size	output_size
(1)	Focus	3 x 320 x 640	64 x 160 x 320
(2)	Conv	64 x 160 x 320	128 x 80 x 160
(3)	C3	128 x 80 x 160	128 x 80 x 160
(4)	Conv	128 x 80 x 160	256 x 40 x 80
(5)	C3	256 x 40 x 80	256 x 40 x 80
(6)	Conv	256 x 40 x 80	512 x 20 x 40
(7)	C3	512 x 20 x 40	512 x 20 x 40
(8)	Conv	512 x 20 x 40	1024 x 10 x 20
(9)	SPP	1024 x 10 x 20	1024 x 10 x 20
(10)	C3	1024 x 10 x 20	1024 x 10 x 20
(11)	Conv	1024 x 10 x 20	512 x 10 x 20
(12)	Upsample	512 x 10 x 20	512 x 20 x 40
(13)	Concat	512 x 20 x 40	1024 x 20 x 40
(14)	C3	1024 x 20 x 40	512 x 20 x 40
(15)	Conv	512 x 20 x 40	256 x 20 x 40
(16)	Upsample	256 x 20 x 40	256 x 40 x 80
(17)	Concat	256 x 40 x 80	512 x 40 x 80
(18)	C3	512 x 40 x 80	256 x 40 x 80
(19)	Conv	256 x 40 x 80	256 x 20 x 40
(20)	Concat	256 x 20 x 40	512 x 20 x 40
(21)	C3	512 x 20 x 40	512 x 20 x 40
(22)	Conv	512 x 20 x 40	512 x 10 x 20
(23)	Concat	512 x 10 x 20	1024 x 10 x 20
(24)	C3	1024 x 10 x 20	1024 x 10 x 20
	Conv2d	256 x 40 x 80	18 x 40 x 80
(25)	Conv2d	512 x 20 x 40	18 x 20 x 40
	Conv2d	1024 x 10 x 20	18 x 10 x 20

(YOLO) V5 Jocher et al., 2021) is used to provide the detections for the tracker. The sole purpose of this is to display the effects of the detector's performance on the tracking results, and to also provide a real-time detector that enables the tracker of running in real-time, tracking objects in live video feeds.

4.1. Structure

In contrast to traditional object detectors which reuse existing classifiers for the task of object detection, the YOLO object detector treats the object detection as regression problem, where a single neural network is used for detecting bounding boxes and predicting class probabilities. This is all done in a single run on a full image (Redmon et al., 2016). This gives YOLO the advantage of being directly optimisable based on performance results, as well as being much faster to detect and predict than most other detectors with a comparable level of accuracy. YOLO V5 is a much-refined version of the YOLO detector, with an extra emphasis on achieving very high detection speeds while retaining a high level of accuracy. To address all needs, four different structures are introduced *S, M, L, X* ranging from the fastest and least accurate to the slowest and most accurate respectively.

The YOLO V5 model structure is made of three parts, the backbone, neck, and head. It uses Cross Stage Partial Network (CSPNet) (Wang et al., 2019) as its backbone to extract features from the input images while maintaining a low computational cost, those features play a vital role in next stages. As for the model's neck it uses Path Aggregation Network (PANet) (Liu et al., 2018) for instance segmentation instead of Feature Pyramid Network (FPN) (Lin et al., 2017), which was used in previous YOLO models. The model neck aids in making the model invariant to scaling as well as performing accurately on new data. Lastly the model head or detection head, YOLO V5 uses the same detection head as previous YOLO versions, as with FPN YOLO V5 uses 3 detection heads that process the image features at different spatial compression, and each has separate anchor scales. These heads are responsible for producing objectness scores, class probabilities and bounding boxes. For a simplified view of the YOLO V5 structure see Table 5.

The main activation functions used with the YOLO V5 model are leaky Rectified Linear Unit (ReLU) and a Sigmoid activation function. As for optimisation function the model can be trained using Stochastic Gradient Descent (SGD) (Ruder, 2017) or Adam stochastic optimisation (Kingma and Ba, 2017). As for loss, the loss is calculated from compounded loss calculation utilising the Binary Cross-Entropy with Logits Loss function.

4.2. Training

The YOLO V5 object detector is originally trained using the Microsoft Common Objects in Context (COCO) dataset (Lin et al., 2015). The COCO dataset has in total a little over **117**K images covering objects from 80 different classes, one of these is the *person* class. As the proposed tracker is evaluated against the MOT challenge benchmarks which are mainly targeting human subjects, a different dataset is used for retraining the YOLO model with a focus on detecting only a *person* class.

Google's Open Images dataset is used for this task, the reason for this is that the open images dataset is much larger with around 1.9 million annotated images covering objects from **600** different classes, with the *person* class having around **250**K annotated images. An extra **25**K background images are added to the dataset to reduce false positives. The images are then split **80%/20%** for training and validation respectively, with a batch size of **16** and ran for **90** epochs. The training results for YOLO *L* structure are displayed in Fig. 10 respectively.

4.3. Performance

The newly trained YOLO V5 Structures *S*&*L* performed remarkably well as can be seen by the training results in Fig. 10. The *L* structure has an $\boldsymbol{mAP_{0.5:0.95}}$ of 48.2 for both validation and testing data, it performs inference in **8 ms** on an *NVIDIA® GeForce® RTX 2080 Ti* graphics card, while the S structure has an $\boldsymbol{mAP_{0.5:0.95}}$ of 36.7 for both validation and testing data, it performs inference in **5 ms** on the same hardware setup.

Both structures perform inference in very short times, which is more than adequate to run detection on live streams of up to **100** fps for the *L* structure and **200** fps for the *S* structure. Since both speeds are more than adequate to handle most live streaming applications, the *L* structure is chosen for integration into the proposed tracker, due to its higher detection accuracy, yet retaining a very operational high-performance speed.

The YOLO V5 object detector has been trained on a very large dataset instead of data from the MOT challenge benchmarks (Leal-Taixé et al., 2015; Milan et al., 2016), in order to generalise performance and reduce over fitting to training data. Training on the relatively small MOT Challenge benchmarks, could possibly yield much higher performance numbers, while sacrificing generality, this is due to model overfitting to the training data, which normally leads to poorer performance when applied to new data.

4.4. Testing

To evaluate the performance of the newly trained model using the MOT Challenge benchmarks, the model is tested on a particular benchmark the MOT17Det which uses the same datasets from the MOT16 benchmark (Milan et al., 2016), coupled with more accurate ground truth data, provided solely for the purpose of object detector performance evaluation. The object detector is run on the given datasets, and the results are run through the MOT Challenge DET DEV Kit, which then produces the final evaluation results. It is worth noting that the performance metrics for object detection is different from the metrics used for MOT. These can be found in Milan et al. (2016). The performance evaluation results are shown in Table 6.

5. Testing and results

As with previous tests, the complete algorithm is run on all datasets included in the MOT challenge benchmarks MOT15, MOT16, and

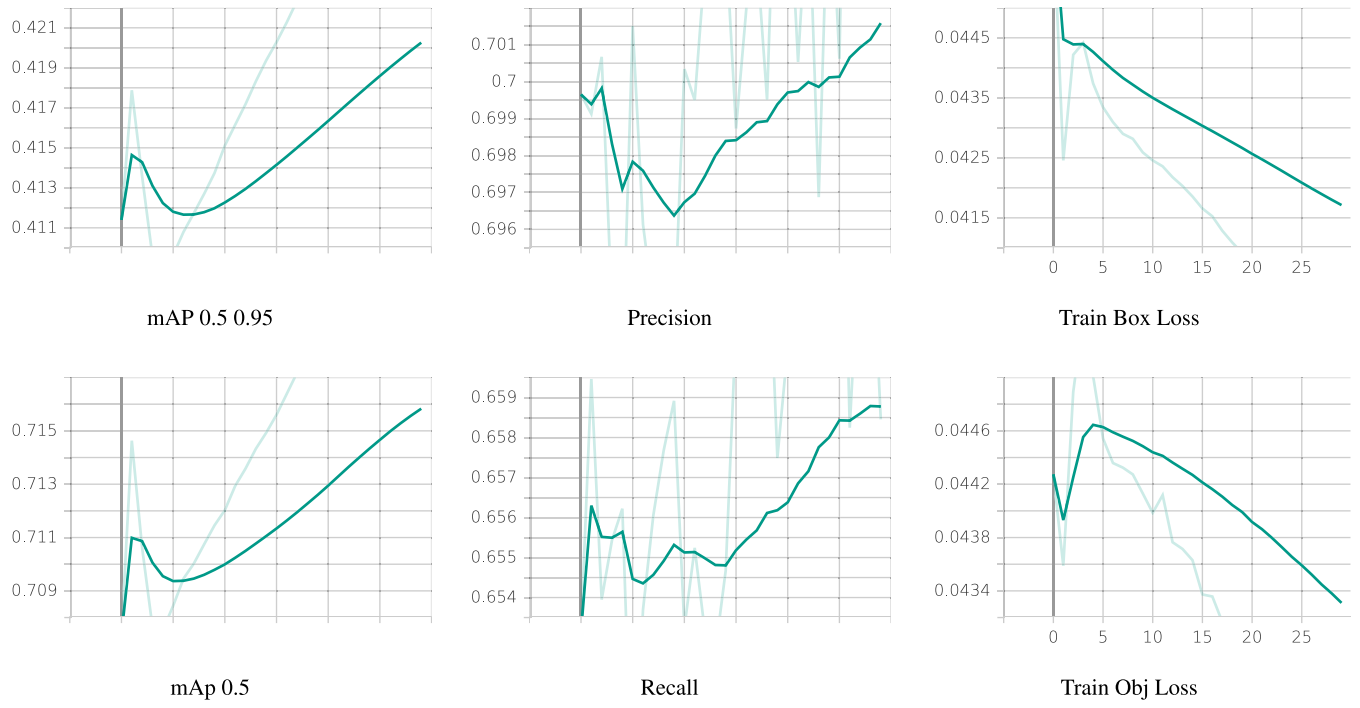


Fig. 10. YOLO V5 L structure training on Google's Open Images Dataset.

Table 6
YOLO V5 object detector evaluation on MOT17Det.

	MODA	MODP	AP	Rcll	Prcn	TP	FP	FN	F1	FAR	GT
OVERALL	55.52	82.38	0.47	68.15	84.36	45 249	8386	21 144	75.40	1.58	756

Table 7
DBN MOT performance evaluation on MOT15/MOT16 using YOLO V5-L detections.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
OVERALL MOT15	**29.56**	**76.56**	**56.87**	**50.21**	**65.57**	**80.37**	**61.54**	**32 073**	**20 042**	**7832**	**62.2**	**28.4**	**9.4**	**311**	**142**	**47**	**3.64**	**657**	**8.17**	**236**	**2.9**
OVERALL MOT16	**29.85**	**82.13**	**41.30**	**78.85**	**27.98**	**32.75**	**92.29**	**36 160**	**3020**	**74 247**	**17.79**	**34.24**	**47.97**	**92**	**177**	**248**	**0.57**	**738**	**22.53**	**188**	**5.7**

Table 8
DBN MOT performance evaluation comparison on MOT15.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
DBN only	**25.97**	**72.50**	**30.97**	**40.52**	**25.06**	**44.89**	**72.57**	**17 915**	**6770**	**21 990**	**12.80**	**39.80**	**47.40**	**64**	**199**	**237**	**1.23**	**1174**	**26.15**	**780**	**17.4**
Similarity est.	**30.02**	**72.94**	**41.42**	**66.14**	**30.15**	**38.07**	**83.53**	**15 193**	**2996**	**24 712**	**11.40**	**29.80**	**58.80**	**57**	**149**	**294**	**0.54**	**833**	**21.88**	**218**	**5.7**
YOLO V5-L	**29.56**	**76.56**	**56.87**	**50.21**	**65.57**	**80.37**	**61.54**	**32 073**	**20 042**	**7832**	**62.2**	**28.4**	**9.4**	**311**	**142**	**47**	**3.64**	**657**	**8.17**	**236**	**2.9**
All combined	**36.58**	**76.72**	**55.55**	**51.01**	**60.97**	**78.47**	**65.65**	**31 314**	**16 383**	**8591**	**55.00**	**34.20**	**10.80**	**275**	**171**	**54**	**2.98**	**647**	**8.25**	**333**	**4.2**

Table 9
DBN MOT performance evaluation comparison on MOT16.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
DBN only	**24.91**	**77.88**	**29.72**	**45.01**	**22.18**	**37.59**	**76.27**	**41 503**	**12 916**	**68 904**	**7.93**	**39.65**	**52.42**	**41**	**205**	**271**	**2.43**	**1493**	**39.72**	**1090**	**29**
Similarity est.	**29.11**	**78.54**	**36.91**	**73.82**	**24.60**	**31.33**	**93.99**	**34 585**	**2212**	**75 822**	**5.61**	**35.01**	**59.38**	**29**	**181**	**307**	**0.42**	**1187**	**37.89**	**239**	**7.6**
YOLO V5-L	**29.85**	**82.13**	**41.30**	**78.85**	**27.98**	**32.75**	**92.29**	**36 160**	**3020**	**74 247**	**17.79**	**34.24**	**47.97**	**92**	**177**	**248**	**0.57**	**738**	**22.53**	**188**	**5.7**
All combined	**37.43**	**80.91**	**48.48**	**71.56**	**36.66**	**44.48**	**86.84**	**49 113**	**7444**	**61 294**	**24.18**	**41.39**	**34.43**	**125**	**214**	**178**	**1.40**	**1103**	**24.80**	**341**	**7.7**

MOT17 (Leal-Taixé et al., 2015; Milan et al., 2016). To show the individual impact of similarity estimation and the YOLO V5 object detector on the proposed DBN MOT. The algorithm implementation is divided to four different algorithms, for the purpose of testing.

- The original DBN based MOT algorithm which has been tested in Section 2.5.
- The DBN MOT including similarity estimation algorithm which has been tested in Section 3.3.
- The DBN MOT including the YOLO V5-L object detector, which will be tested next.
- The complete DBN MOT utilising both similarity estimation and object detection, which will be tested next.

The DBN MOT is executed on all datasets in the three benchmarks using detections provided by the custom trained YOLO V5-L object detector, the results are then fed into the MOT Challenge DEV Kit and the performance evaluation results are produced. The results for running on MOT15 (Leal-Taixé et al., 2015) and MOT16 (Milan et al., 2016) are shown in Table 7.

In Tables 8 and 9 are the comparisons between the DBN MOT implementations on stock detections vs. implementations on the custom

trained YOLO V5-L object detector on the MOT15 (Leal-Taixé et al., 2015) and MOT16 (Milan et al., 2016) benchmark datasets respectively. As can be seen from Table 8, applying the DBN MOT using the detections from the custom trained YOLO V5-L object detector has improved the overall performance of the tracker, this can be seen by the increase in the Multi-Object Tracking Accuracy (MOTA) value from 25.97 to 29.56 when run on all the datasets in the MOT15 benchmark (Leal-Taixé et al., 2015). The number of Identity Switches (IDSW) has decreased considerably from **780** to **236**, and recall has increased considerably from 44.89 to 80.37.

As can be seen from Table 9, applying the DBN MOT using the detections from the custom trained YOLO V5-L object detector has again improved the overall performance of the tracker, this can be seen by the increase in the MOTA value from 24.91 to 29.85 when run on all the datasets in the MOT16 benchmark (Leal-Taixé et al., 2015). The number of IDSW has decreased considerably from **1090** to **188**, and precision has increased considerably from 76.27 to 92.29.

From the previous results it can be concluded that running the DBN MOT on detections from the custom trained YOLO V5-L object detector has significantly improved the performance results of the tracker. This lends to the same aforementioned conclusion at the beginning of the Section 4, any tracking-by-detection based tracker, is affected by the quality of the detections used for its operation. Another conclusion is that stock detections provided in both MOT15 and MOT16 benchmark datasets (Leal-Taixé et al., 2015; Milan et al., 2016) are of a lower quality than the detections produced by the custom trained YOLO V5-L object detector.

In Tables 8 and 9 are also comparisons between the DBN MOT implementations on stock detections vs. DBN MOT implementations on stock detections utilising similarity estimation and both comparisons are made on implementations run on the MOT15 (Leal-Taixé et al., 2015) and MOT16 (Milan et al., 2016) benchmark datasets respectively. As can be seen from Table 8, applying the DBN MOT using custom built ResNet-18 feature extractor with the similarity estimation has improved the overall performance of the tracker, this can be seen by the increase in the MOTA value from 25.97 to 30.02 when run on all the datasets in the MOT15 benchmark (Leal-Taixé et al., 2015). The number of IDSW has decreased considerably from **780** to **218**, and precision has increased from 72.57 to 83.53.

As can be seen from Table 9, applying the DBN MOT using the custom built ResNet-18 feature extractor with the similarity estimation has again improved the overall performance of the tracker, this can be seen by the increase in the MOTA value from 24.91 to 29.11 when run on all the datasets in the MOT16 benchmark (Leal-Taixé et al., 2015). The number of IDSW has decreased considerably from **1090** to **239**, and precision has increased considerably from 76.27 to 93.99.

From the previous results it can be concluded that running the DBN MOT using the custom built ResNet-18 feature extractor with the similarity estimation has significantly improved the performance results of the tracker. This led to the conclusion that similarity estimation aids the tracker, particularly in high uncertainty cases, where there are interactions or occlusions between objects in the frame.

After demonstrating that the DBN MOT performance was considerably enhanced by the addition of similarity estimation or by running on detections provided by the custom trained YOLO V5-L object detector. In this section the DBN MOT is run on detections provided by the YOLO V5-L object detector while utilising the custom built ResNet-18 feature extractor with the similarity estimation. This is run on all datasets in both MOT15 and MOT16 benchmarks (Leal-Taixé et al., 2015; Milan et al., 2016). The results are displayed in Table 10 respectively.

In order to make sense of the results in Table 10, and as with previous results, a comparison has been drawn, between all four implementations. The DBN MOT on its own, DBN MOT utilising similarity estimation, the DBN MOT run on custom YOLO V5-L object detector and finally the DBN MOT utilising similarity estimation and running on detections provided by the custom YOLO V5-L object detector. This comparison is displayed in Table 8 which is run on the MOT15 benchmark (Leal-Taixé et al., 2015), and in Table 9 which is run on the MOT16 benchmark (Milan et al., 2016).

In Tables 8 and 9 are the comparisons of all aforementioned implementations, as can be seen from Table 8, applying the DBN MOT using custom built ResNet-18 feature extractor with the similarity estimation running on detections provided by the custom trained YOLO V5-L object detector has improved the overall performance of the tracker the most, this can be seen by the increase in the MOTA value from 25.97 to 36.58 when run on all the datasets in the MOT15 benchmark (Leal-Taixé et al., 2015). The number of IDSW has decreased considerably from **780** to **333**, and recall has increased from 44.89 to 78.47.

As can be seen from Table 9, applying the DBN MOT using the custom built ResNet-18 feature extractor with the similarity estimation running on detections provided by the custom trained YOLO V5-L object detector has again improved the overall performance of the tracker the most, this can be seen by the increase in the MOTA value from 24.91 to 37.43 when run on all the datasets in the MOT16 benchmark (Leal-Taixé et al., 2015). The number of IDSW has decreased considerably from **1090** to **341**, precision has increased considerably from 76.27 to 86.84, and recall has increased from 37.59 to 44.48.

From the previous results it can now be concluded that running the DBN MOT using the custom built ResNet-18 feature extractor with the similarity estimation running on detections provided by the custom trained YOLO V5-L object detector has significantly improved the overall performance of the tracker.

Previously it has been shown that the detections quality has a crippling effect on the performance of a tracker running based on them. The proposed YOLO V5-L object detector is not the most accurate object detector that exists today, but it is one of the fastest ones. Therefore, it was chosen, as it creates a perfect balance between accuracy and speed. The MOT Challenge team (Leal-Taixé et al., 2015) have introduced an MOT17 benchmark, which uses the same datasets from the MOT16 benchmark (Milan et al., 2016), only this time these datasets are coupled with 3 different detections from 3 different object detectors, these are DPM, Faster-RCNN, and SDP object detectors.

The DBN MOT with and without the custom built ResNet-18 feature extractor with the similarity estimation is run on the MOT17 detections, and the results are displayed in Tables 11 and 12 respectively. The results demonstrate an immense improvement when compared to the same implementations run on the same dataset in the MOT16 benchmark (Milan et al., 2016), but without using the detections, even when compared to the DBN MOT running on detections produced by the custom trained YOLO V5-L object detector, the MOT17 performance out matches it. There are two main reasons for this, firstly the quality of the predictions from some of the object detectors (e.g., Faster-RCNN) are higher, and the quality of the ground truth provided with the datasets in the MOT17 benchmark is better. Albeit these object detectors provide superior detections, they lack the speed to make them executable in real-time, and as mentioned earlier, the reason the YOLO V5-L object detector was chosen is that it achieves a perfect balance between speed and accuracy, and this makes it the perfect candidate to run in real-time.

5.1. Comparison to other approaches

The proposed DBN based MOT has performed well, and this performance was further enhanced by the introduction of the custom built ResNet-18 feature extractor with the similarity estimation running on detections provided by the custom trained YOLO V5-L object detector. It was shown earlier that one of the main reasons to use publicly available benchmarks is to have the ability to objectively compare one's work to others in the field, and this was the primary motive behind using the MOT15, MOT16 and MOT17Det benchmark datasets with this work. In Table 13 are the comparison results of tracker performance for the DBN MOT against some of the algorithms that have been

Table 10
DBN MOT performance evaluation on MOT15/MOT16 using similarity estimation and YOLO V5-L detections.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
OVERALL MOT15	**36.58**	**76.72**	**55.55**	**51.01**	**60.97**	**78.47**	**65.65**	**31 314**	**16 383**	**8591**	**55.00**	**34.20**	**10.80**	**275**	**171**	**54**	**2.98**	**647**	**8.25**	**333**	**4.2**
OVERALL MOT16	**37.43**	**80.91**	**48.48**	**71.56**	**36.66**	**44.48**	**86.84**	**49 113**	**7444**	**61 294**	**24.18**	**41.39**	**34.43**	**125**	**214**	**178**	**1.40**	**1103**	**24.80**	**341**	**7.7**

Table 11
DBN MOT performance evaluation on MOT17.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
OVERALL	**45.20**	**85.39**	**45.89**	**70.57**	**33.99**	**47.02**	**97.62**	**158 417**	**3869**	**178 474**	**15.26**	**41.15**	**43.59**	**250**	**674**	**714**	**0.24**	**3180**	**67.63**	**2288**	**48.7**

Table 12
DBN MOT performance evaluation on MOT17 using similarity estimation.

	MOTA	MOTP	IDF1	IDP	IDR	Rcll	Prcn	TP	FP	FN	MTR	PTR	MLR	MT	PT	ML	FAR	FM	FMR	IDSW	IDSWR
OVERALL	**47.64**	**84.18**	**53.23**	**77.70**	**40.48**	**50.05**	**96.08**	**168 628**	**6877**	**168 263**	**19.41**	**42.55**	**38.03**	**318**	**697**	**623**	**0.43**	**3430**	**68.53**	**1255**	**25.1**

Table 13
DBN MOT utilising similarity estimation and YOLO V5-L object detector performance comparison with state of the art methods on MOT15.

Tracker	MOTA	IDF1	MOTP	MT	ML	FP	FN	Rcll	Prcn	ID Sw.	Hz
CRFTrack_ (Xiang et al., 2021)	40	49.6	71.9	166	28.6	10 295	25 917	57.8	77.5	658	3.2
KCF (Chu et al., 2019)	38.9	44.5	70.6	120	31.5	7321	29 501	52	81.4	720	0.3
AP_HWDPL_p (Chen et al., 2017)	38.5	47.1	72.6	63	37.4	4005	33 203	46	87.6	586	6.7
STRN (Xu et al., 2019)	38.1	46.6	72.1	83	33.4	5451	31 571	48.6	84.6	1033	13.8
DBN_MOT	36.58	55.55	76.72	275	54	16 383	8591	78.47	65.65	333	25
TPM (Peng et al., 2020)	36.2	43.6	71.5	111	42.6	5650	33 102	46.1	83.4	420	0.8
JointMC (Keuper et al., 2020)	35.6	45.1	71.9	167	39.3	10 580	28 508	53.6	75.7	457	0.6
TSN (Peng et al., 2018)	35.5	43	71.5	104	43.6	5682	33 515	45.5	83.1	454	0.8
RAR15pub (Fang et al., 2018)	35.1	45.4	70.9	94	42.3	6771	32 717	46.7	80.9	381	5.4
HybridDAT (Yang et al., 2017)	35	47.7	72.6	82	42.2	8455	31 140	49.3	78.2	358	4.6

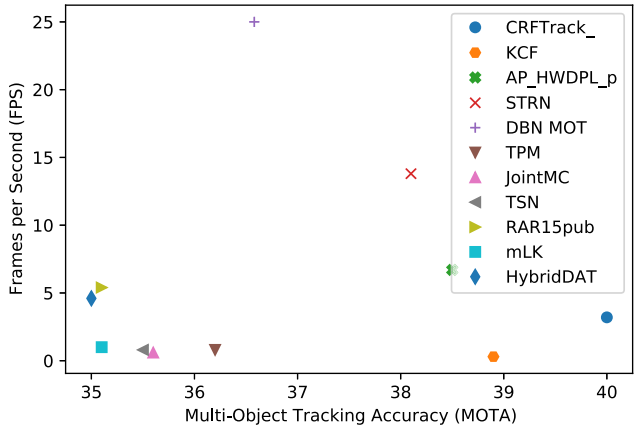


Fig. 11. Speed vs. Accuracy MOT comparison on MOT15.

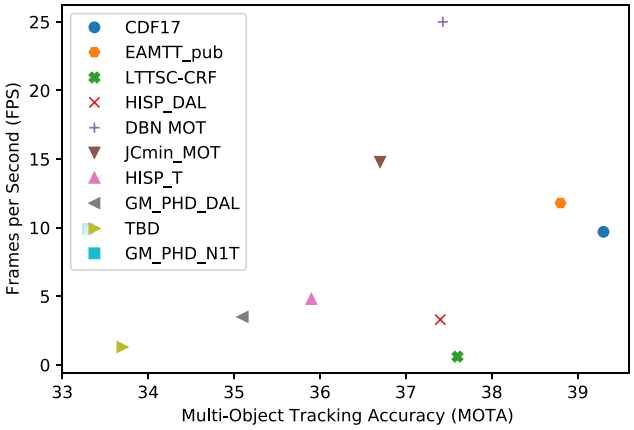


Fig. 12. Speed vs. Accuracy MOT comparison on MOT16.

implemented on the MOT15 benchmark, from the table it can be seen that the DBN MOT is not the tracker with the highest MOTA score, which is normally one of the main criteria for evaluating performance within the MOT community, but it is one of the fastest trackers, this can be seen clearly in Fig. 11.

These results are further confirmed, in Table 14 which contains the comparison results of tracker performance for the DBN MOT against some of the algorithms that have been implemented on the MOT16 benchmark (Milan et al., 2016). Where again the DBN MOT performance shows that it is not the tracker with the highest MOTA score, but that it is one of the fastest algorithms tested against both MOT15 and MOT16 benchmarks (Leal-Taixé et al., 2015; Milan et al., 2016). This goes to show that it has achieved a delicate balance between speed and accuracy. This can be seen clearly in Fig. 12.

Since the beginning of this work, it has been one of the primary goals to develop the tracker with the ability of achieving real-time performance tracking objects across multiple frames.

To a certain degree this has been accomplished, as the DBN tracker itself can run tracking loops for a single frame in less than **5 ms** on a 9th Generation Intel® Core™ i7 Processors with 8 cores. After the introduction of the custom built ResNet-18 feature extractor with the similarity estimation, the execution time has increased, where the custom feature extractor can achieve almost **750** $fps \approx$ **1.3 ms** when batch inferencing on an *NVIDIA® GeForce® RTX 2080 Ti* graphics card, and the cosine similarity estimation function can run **20**K comparisons per second on the same CPU, and as discussed in the previous section, the custom trained YOLO V5-L can run detections in **8 ms** on the *NVIDIA® GeForce® RTX 2080 Ti* graphics card, which indicates that the tracker can run a complete iteration, including object detection and similarity estimation in as little as **20 ms** ≈ **50** fps.

It should be noted that the tracker's various calculations are affected heavily by the number of tracked objects between consecutive frames, as this affects mainly the feature extraction running time, as this is done for individual bounding boxes for each newly detected object in the

Table 14
DBN MOT utilising similarity estimation and YOLO V5-L object detector performance comparison with state of the art methods on MOT16.

Tracker	MOTA	IDF1	MOTP	MT	ML	FP	FN	Rcll	Prcn	ID Sw.	Hz
CDF17 (Fu et al., 2018)	39.3	33.6	74.8	95	40.8	12 430	93 394	48.8	87.7	4934	9.7
EAMTT_pub (Sanchez-Matilla et al., 2016)	38.8	42.4	75.1	60	49.1	8114	102 452	43.8	90.8	965	11.8
LTTSC-CRF (Le et al., 2016)	37.6	42.1	75.9	73	55.2	11 969	101 343	44.4	87.1	481	0.6
DBN_MOT	37.43	48.48	80.91	125	178	7444	61 294	44.48	86.84	341	25
HISP_DAL (Baisa, 2020)	37.4	30.5	76.3	58	50.9	3222	108 865	40.3	95.8	2101	3.3
JCmin_MOT (Boragule and Jeon, 2017)	36.7	36.2	75.9	57	54.4	2936	111 890	38.6	96	667	14.8
HISP_T (Baisa., 2018)	35.9	28.9	76.1	59	50.1	6412	107 918	40.8	92.1	2594	4.8
GM_PHD_DAL (Baisa, 2019)	35.1	26.6	76.6	53	51.4	2350	111 886	38.6	96.8	4047	3.5
TBD (Geiger et al., 2014)	33.7	0	76.5	55	54.2	5804	112 587	38.2	92.3	2418	1.3
GM_PHD_N1T (Baisa and Wallace, 2019)	33.3	25.5	76.8	42	56	1750	116 452	36.1	97.4	3499	9.9

current frame, also in the rare case of tracking a very large number of objects, the cost assignment matrix operations can also be affected. In real world testing on both the MOT15 and MOT16 benchmark (Leal-Taixé et al., 2015; Milan et al., 2016) it was found that running time was at an average of **25** to **30** fps. Which is more than enough to track objects on live video streams (normally $\approx$ **30** fps) with a high level of accuracy.

6. Conclusion

In this paper a new and novel DBN based MOT has been introduced. This tracker is based on the track-by-detection tracking paradigm, where the DBN model serves as a predictor for possible locations of tracked objects based on provided detections. Then a cost assignment matrix is built on top of the IoU distances between predictions and observations, this matrix is then used for data association and tracks creation and deletions.

This tracker is further enhanced by the introduction of a custom-built feature extractor, that is built by re-purposing the structure of the ResNet-18 classifier, which is then trained on state-of-the-art pedestrian re-identification dataset (MARS (Zheng et al., 2016)). The features that are extracted are then used with cosine similarity measure to create a second distance that is compounded together with the IoU distance between predictions and observations to create a more accurate and certain measure for completing the cost assignment matrix and data association steps. This has resulted in a measurable increase in performance on all testing datasets.

A custom trained YOLO V5-L object detector is introduced as the generator for more accurate and fast online detections. This detector is trained on Google's extensive Open Images dataset (Kuznetsova et al., 2020). The detector is then used to emphasise the importance of detections accuracy and its effects on any MOT based on the tracking-by-detection paradigm. This is then tested in the MOT17Det benchmark. Results showed an unmistakably enhanced performance over provided detections while still being capable to run in real-time (**25 – 30** fps on average).

Finally, the tracker's performance is analysed and compared to state-of-the-art trackers, and it is demonstrated how the tracker achieves a delicate balance between accuracy and speed, which allows it to run in real-time yet achieve a high level of accuracy.

CRediT authorship contribution statement

Mohamad Saada: Conceptualization, Methodology, Software, Data curation, Formal analysis, Investigation, Visualization, Writing – original draft. **Christos Kouppas:** Validation, Writing – review & editing. **Baihua Li:** Supervision. **Qinggang Meng:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Baisa., N.L., 2018. Online multi-target visual tracking using a HISP filter. In: Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Vol. 5. VISAPP, SciTePress, INSTICC, pp. 429–438. http://dx.doi.org/10.5220/0006564504290438.

Baisa, N.L., 2019. Online multi-object visual tracking using a GM-PHD filter with deep appearance learning. In: 2019 22th International Conference on Information Fusion. FUSION, pp. 1–8.

Baisa, N.L., 2020. Robust online multi-target visual tracking using a HISP filter with discriminative deep appearance learning. arXiv:1908.03945.

Baisa, N.L., Wallace, A., 2019. Development of a N-type GM-PHD filter for multiple target, multiple type visual tracking. J. Vis. Commun. Image Represent. 59, 257–271. http://dx.doi.org/10.1016/j.jvcir.2019.01.026, URL https://www.sciencedirect.com/science/article/pii/S1047320319300343.

Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B., 2016. Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing. ICIP, IEEE, http://dx.doi.org/10.1109/icip.2016.7533003.

Boragule, A., Jeon, M., 2017. Joint cost minimization for multi-object tracking. In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance. AVSS, pp. 1–6. http://dx.doi.org/10.1109/AVSS.2017.8078481.

Breitenstein, M.D., Reichlin, F., Leibe, B., Koller-Meier, E., Van Gool, L., 2009. Robust tracking-by-detection using a detector confidence particle filter. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 1515–1522. http://dx.doi.org/10.1109/ICCV.2009.5459278.

Butt, A.A., Collins, R.T., 2013. Multi-target tracking by Lagrangian relaxation to min-cost network flow. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1846–1853. http://dx.doi.org/10.1109/CVPR.2013.241.

Cai, B., Hao, K., Wang, Z., Yang, C., Kong, X., Liu, Z., Ji, R., Liu, Y., 2021a. Data-driven early fault diagnostic methodology of permanent magnet synchronous motor. Expert Syst. Appl. 177, 115000. http://dx.doi.org/10.1016/j.eswa.2021.115000, URL https://www.sciencedirect.com/science/article/pii/S0957417421004413.

Cai, B., Sheng, C., Gao, C., Liu, Y., Shi, M., Liu, Z., Feng, Q., Liu, G., 2021b. Artificial intelligence enhanced reliability assessment methodology with small samples. IEEE Trans. Neural Netw. Learn. Syst. 1–13. http://dx.doi.org/10.1109/TNNLS.2021.3128514.

Cai, B., Wang, Z., Zhu, H., Liu, Y., Hao, K., Yang, Z., Ren, Y., Feng, Q., Liu, Z., 2022. Artificial intelligence enhanced two-stage hybrid fault prognosis methodology of PMSM. IEEE Trans. Ind. Inform. 18 (10), 7262–7273. http://dx.doi.org/10.1109/TII.2021.3128245.

Chen, L., Ai, H., Shang, C., Zhuang, Z., Bai, B., 2017. Online multi-object tracking with convolutional neural networks. In: 2017 IEEE International Conference on Image Processing. ICIP, pp. 645–649. http://dx.doi.org/10.1109/ICIP.2017.8296360.

Chen, S., Fern, A., Todorovic, S., 2014. Multi-object tracking via constrained sequential labeling. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1130–1137. http://dx.doi.org/10.1109/CVPR.2014.148.

Chen, J., Sheng, H., Li, C., Xiong, Z., 2016. PSTG-based multi-label optimization for multi-target tracking. Comput. Vis. Image Underst. 144, 217–227. http://dx.doi.org/10.1016/j.cviu.2015.06.002, Individual and Group Activities in Video Event Analysis URL https://www.sciencedirect.com/science/article/pii/S1077314215001307.

Chu, P., Fan, H., Tan, C.C., Ling, H., 2019. Online multi-object tracking with instance-aware tracker and dynamic model refreshment. In: 2019 IEEE Winter Conference on Applications of Computer Vision. WACV, pp. 161–170. http://dx.doi.org/10.1109/WACV.2019.00023.

Fang, K., Xiang, Y., Li, X., Savarese, S., 2018. Recurrent autoregressive networks for online multi-object tracking. In: 2018 IEEE Winter Conference on Applications of Computer Vision. WACV, pp. 466–475. http://dx.doi.org/10.1109/WACV.2018.00057.

Farenzena, M., Bazzani, L., Perina, A., Murino, V., Cristani, M., 2010. Person re-identification by symmetry-driven accumulation of local features. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 2360–2367. http://dx.doi.org/10.1109/CVPR.2010.5539926.

Ferryman, J., Shahrokni, A., 2009. PETS2009: Dataset and challenge. In: 2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance. pp. 1–6. http://dx.doi.org/10.1109/PETS-WINTER.2009.5399556.

Fu, Z., Naqvi, S.M., Chambers, J.A., 2018. Collaborative detector fusion of data-driven PHD filter for online multiple human tracking. In: 2018 21st International Conference on Information Fusion. FUSION, pp. 1976–1981. http://dx.doi.org/10.23919/ICIF.2018.8455432.

Geiger, A., Lauer, M., Wojek, C., Stiller, C., Urtasun, R., 2014. 3D traffic scene understanding from movable platforms. IEEE Trans. Pattern Anal. Mach. Intell. 36 (5), 1012–1025. http://dx.doi.org/10.1109/TPAMI.2013.185.

Ghahramani, Z., Hinton, G.E., 1996. Parameter estimation for linear dynamical systems. Technical Report, University of Toronto, Department of Computer Science, 6 King's College Road, Toronto, Canada M5S 1A4..

Gwak, J., 2017. Multi-object tracking through learning relational appearance features and motion patterns. Comput. Vis. Image Underst. 162, 103–115. http://dx.doi.org/10.1016/j.cviu.2017.05.010, URL https://www.sciencedirect.com/science/article/pii/S1077314217300966.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 770–778. http://dx.doi.org/10.1109/CVPR.2016.90.

Hess, R., Fern, A., 2009. Discriminatively trained particle filters for complex multi-object tracking. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 240–247. http://dx.doi.org/10.1109/CVPR.2009.5206801.

Huang, S., Jiang, S., Zhu, X., 2016. Multi-object tracking via discriminative appearance modeling. Comput. Vis. Image Underst. 153, 77–87. http://dx.doi.org/10.1016/j.cviu.2016.06.003, Special issue on Visual Tracking URL https://www.sciencedirect.com/science/article/pii/S1077314216300790.

Jiménez-Bravo, D.M., Lozano Murciego, Á., Sales Mendes, A., Sánchez San Blás, H., Bajo, J., 2022. Multi-object tracking in traffic environments: A systematic literature review. Neurocomputing 494, 43–55. http://dx.doi.org/10.1016/j.neucom.2022.04.087, URL https://www.sciencedirect.com/science/article/pii/S0925231222004672.

Jin, Y., Mokhtarian, F., 2007. Variational particle filter for multi-object tracking. In: 2007 IEEE 11th International Conference on Computer Vision. pp. 1–8. http://dx.doi.org/10.1109/ICCV.2007.4408952.

Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., Laughing, tkianai, yxNONG, Hogan, A., lorenzomammana, AlexWang1900, Chaurasia, A., Diaconu, L., Marc, wanghaoyang0106, ml5ah, Doug, Durgesh, Ingham, F., Frederik, Guilhen, Colmagro, A., Ye, H., Jacobsolawetz, Poznanski, J., Fang, J., Kim, J., Doan, K., Yu, L., 2021. ultralytics/yolov5: v4.0 - nn.SiLU activations, Weights & Biases logging, PyTorch Hub integration. Zenodo, http://dx.doi.org/10.5281/zenodo.4418161.

Kamen, E., 1990. Multiple target tracking using an extended Kalman filter. In: Drummond, O.E. (Ed.), Signal and Data Processing of Small Targets 1990, Vol. 1305. SPIE, International Society for Optics and Photonics, http://dx.doi.org/10.1117/12.2321782.

Keuper, M., Tang, S., Andres, B., Brox, T., Schiele, B., 2020. Motion segmentation multiple object tracking by correlation co-clustering. IEEE Trans. Pattern Anal. Mach. Intell. 42 (1), 140–153. http://dx.doi.org/10.1109/TPAMI.2018.2876253.

Khan, A., Zhang, J., Wang, Y., 2010. Appearance-based re-identification of people in video. In: 2010 International Conference on Digital Image Computing: Techniques and Applications. pp. 357–362. http://dx.doi.org/10.1109/DICTA.2010.67.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv:1412.6980.

Kratz, L., Nishino, K., 2012. Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. IEEE Trans. Pattern Anal. Mach. Intell. 34 (5), 987–1002. http://dx.doi.org/10.1109/TPAMI.2011.173.

Kuhn, H.W., 1955. The hungarian method for the assignment problem. Nav. Res. Logist. Q. 2 (1–2), 83–97. http://dx.doi.org/10.1002/nav.3800020109, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V., 2020. The open images dataset V4. Int. J. Comput. Vis. 128 (7), 1956–1981. http://dx.doi.org/10.1007/s11263-020-01316-z.

Le, N., Heili, A., Odobez, J.-M., 2016. Long-term time-sensitive costs for CRF-based tracking by detection. In: Hua, G., Jégou, H. (Eds.), Computer Vision – ECCV 2016 Workshops. Springer International Publishing, Cham, pp. 43–51.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K., 2015. MOTChallenge 2015: Towards a benchmark for multi-target tracking. arXiv:1504.01942.

Lenz, P., Geiger, A., Urtasun, R., 2015. Followme: Efficient online min-cost flow tracking with bounded memory and computation. In: 2015 IEEE International Conference on Computer Vision. ICCV, pp. 4364–4372. http://dx.doi.org/10.1109/ICCV.2015.496.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. arXiv:1612.03144.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P., 2015. Microsoft COCO: Common objects in context. arXiv:1405.0312.

Liu, C., Gong, S., Loy, C.C., Lin, X., 2012. Person re-identification: What features are important? In: Fusiello, A., Murino, V., Cucchiara, R. (Eds.), Computer Vision – ECCV 2012. Workshops and Demonstrations. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 391–401.

Liu, S., Qi, L., Qin, H., Shi, J., Jia, J., 2018. Path aggregation network for instance segmentation. arXiv:1803.01534.

Marini, S., Trifoglio, E., Barbarini, N., Sambo, F., Di Camillo, B., Malovini, A., Manfrini, M., Cobelli, C., Bellazzi, R., 2015. A dynamic Bayesian network model for long-term simulation of clinical complications in type 1 diabetes. J. Biomed. Inform. 57, 369–376. http://dx.doi.org/10.1016/j.jbi.2015.08.021, URL https://www.sciencedirect.com/science/article/pii/S1532046415001896.

Milan, A., Leal-Taixe, L., Reid, I., Roth, S., Schindler, K., 2016. MOT16: A benchmark for multi-object tracking. arXiv:1603.00831.

Milan, A., Leal-Taixé, L., Schindler, K., Reid, I., 2015. Joint tracking and segmentation of multiple targets. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 5397–5406. http://dx.doi.org/10.1109/CVPR.2015.7299178.

Park, C., Woehl, T.J., Evans, J.E., Browning, N.D., 2015. Minimum cost multi-way data association for optimizing multitarget tracking of interacting objects. IEEE Trans. Pattern Anal. Mach. Intell. 37 (3), 611–624. http://dx.doi.org/10.1109/TPAMI.2014.2346202.

Pavlovic, V., Rehg, J.M., Cham, T.-J., Murphy, K.P., 1999. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 1. pp. 94–101. http://dx.doi.org/10.1109/ICCV.1999.791203.

Peng, J., Qiu, F., See, J., Guo, Q., Huang, S., Duan, L.-Y., Lin, W., 2018. Tracklet siamese network with constrained clustering for multiple object tracking. In: 2018 IEEE Visual Communications and Image Processing. VCIP, pp. 1–4. http://dx.doi.org/10.1109/VCIP.2018.8698623.

Peng, J., Wang, T., Lin, W., Wang, J., See, J., Wen, S., Ding, E., 2020. TPM: Multiple object tracking with tracklet-plane matching. Pattern Recognit. 107, 107480. http://dx.doi.org/10.1016/j.patcog.2020.107480, URL https://www.sciencedirect.com/science/article/pii/S0031320320302831.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. arXiv:1506.02640.

Riahi, D., Bilodeau, G.-A., 2016. Online multi-object tracking by detection based on generative appearance models. Comput. Vis. Image Underst. 152, 88–102. http://dx.doi.org/10.1016/j.cviu.2016.07.012, URL https://www.sciencedirect.com/science/article/pii/S1077314216301023.

Ruder, S., 2017. An overview of gradient descent optimization algorithms. arXiv:1609.04747.

Saada, M., Meng, Q., 2012. An efficient algorithm for anomaly detection in a flight system using dynamic Bayesian networks. In: Proceedings of the 19th International Conference on Neural Information Processing - Volume Part III. In: ICONIP'12, Springer-Verlag, Berlin, Heidelberg, pp. 620–628. http://dx.doi.org/10.1007/978-3-642-34487-9_75, URL http://dx.doi.org/10.1007/978-3-642-34487-9_75.

Saada, M., Meng, Q., Huang, T., 2014. A novel approach for pilot error detection using dynamic Bayesian networks. Cogn. Neurodyn. 8 (3), 227–238. http://dx.doi.org/10.1007/s11571-013-9278-5.

Sanchez-Matilla, R., Poiesi, F., Cavallaro, A., 2016. Online multi-target tracking with strong and weak detections. In: Hua, G., Jégou, H. (Eds.), Computer Vision – ECCV 2016 Workshops. Springer International Publishing, Cham, pp. 84–99.

Santosh, D.H., Mohan, P.G.K., 2014. Multiple objects tracking using extended Kalman filter, GMM and mean shift algorithm - a comparative study. In: 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies. pp. 1484–1488. http://dx.doi.org/10.1109/ICACCCT.2014.7019350.

Stadler, D., Beyerer, J., 2022. Modelling ambiguous assignments for multi-person tracking in crowds. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops. WACVW, pp. 133–142. http://dx.doi.org/10.1109/WACVW54805.2022.00019.

Wang, C.-Y., Liao, H.-Y.M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., 2019. CSPNet: A new backbone that can enhance learning capability of CNN. arXiv:1911.11929.

Wojke, N., Bewley, A., Paulus, D., 2017. Simple online and realtime tracking with a deep association metric. arXiv:1703.07402.

Wu, J., Cao, J., Song, L., Wang, Y., Yang, M., Yuan, J., 2021. Track to detect and segment: An online multi-object tracker. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR.

Xiang, J., Xu, G., Ma, C., Hou, J., 2021. End-to-end learning deep CRF models for multi-object tracking deep CRF models. IEEE Trans. Circuits Syst. Video Technol. 31 (1), 275–288. http://dx.doi.org/10.1109/TCSVT.2020.2975842.

Xu, J., Cao, Y., Zhang, Z., Hu, H., 2019. Spatial-temporal relation networks for multi-object tracking. In: 2019 IEEE/CVF International Conference on Computer Vision. ICCV, pp. 3987–3997. http://dx.doi.org/10.1109/ICCV.2019.00409.

Yang, C., Duraiswami, R., Davis, L., 2005. Fast multiple object tracking via a hierarchical particle filter. In: Tenth IEEE International Conference on Computer Vision, Vol. 1. ICCV'05, pp. 212–219. http://dx.doi.org/10.1109/ICCV.2005.95.

Yang, B., Huang, C., Nevatia, R., 2011. Learning affinities and dependencies for multi-target tracking using a CRF model. In: CVPR 2011. pp. 1233–1240. http://dx.doi.org/10.1109/CVPR.2011.5995587.

Yang, B., Nevatia, R., 2012. An online learned CRF model for multi-target tracking. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2034–2041. http://dx.doi.org/10.1109/CVPR.2012.6247907.

Yang, M., Wu, Y., Jia, Y., 2017. A hybrid data association framework for robust online multi-object tracking. IEEE Trans. Image Process. 26 (12), 5667–5679. http://dx.doi.org/10.1109/tip.2017.2745103.

Yin, S., Na, J.H., Choi, J.Y., Oh, S., 2011. Hierarchical Kalman-particle filter with adaptation to motion changes for object tracking. Comput. Vis. Image Underst. 115 (6), 885–900. http://dx.doi.org/10.1016/j.cviu.2011.02.010, URL https://www.sciencedirect.com/science/article/pii/S1077314211000877.

Yoon, J.H., Yang, M., Lim, J., Yoon, K., 2015. Bayesian multi-object tracking using motion context from multiple objects. In: 2015 IEEE Winter Conference on Applications of Computer Vision. pp. 33–40. http://dx.doi.org/10.1109/WACV.2015.12.

Zhang, L., Li, Y., Nevatia, R., 2008. Global data association for multi-object tracking using network flows. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. http://dx.doi.org/10.1109/CVPR.2008.4587584.

Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W., 2021. FairMOT: On the fairness of detection and re-identification in multiple object tracking. Int. J. Comput. Vis. 129 (11), 3069–3087. http://dx.doi.org/10.1007/s11263-021-01513-4.

Zheng, L., Bie, Z., Sun, Y., Wang, J., Su, C., Wang, S., Tian, Q., 2016. MARS: A video benchmark for large-scale person re-identification. In: European Conference on Computer Vision. Springer.

Zheng, W., Gong, S., Xiang, T., 2011. Person re-identification by probabilistic relative distance comparison. In: CVPR 2011. pp. 649–656. http://dx.doi.org/10.1109/CVPR.2011.5995598.

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., Tian, Q., 2015. Scalable person re-identification: A benchmark. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV.

Zhong, J., Tan, J., Li, Y., Gu, L., Chen, G., 2014. Multi-targets tracking based on bipartite graph matching. Cybern. Inf. Technol. 14 (5), 78–87.