Multi-Person Pose Tracking Using Dynamically Gated Similarities

Verfolgung von mehreren Personen unter Verwendung dynamisch gesteuerter Ähnlichkeiten
Master thesis in the field of study “Computational Engineering” by Martin Steinborn
Date of submission: December 6, 2024

1. Review: Jan Peters
2. Review: Suman Pal
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Martin Steinborn, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 6. Dezember 2024

M. Steinborn

Abstract

Human pose tracking is a crucial task for many applications, such as human-robot interaction or video surveillance. Most current methods use a tracking-by-detection approach, which has shown to be very successful. These methods either use the location and movement of the detected humans or their visual appearance to track them. In recent years, some models started using heuristic approaches to combine both the pose and the appearance information. Even these newer models are still using fixed weights for these combinations. Therefore, the weights are dataset-dependent and need to be tuned for each dataset.

In this thesis, we propose a new model that uses dynamically learned weights to combine the pose, appearance, and or other information. The so-called Dynamically-Gated Similarities (DGS) model is highly modular and customizable. Weights can be used and trained mostly dataset independent. Additionally, the weights are computed individually for every detection, allowing the model to adapt more freely to the current situation. The DGS model is evaluated on the PoseTrack21 and the DanceTrack datasets. The proposed algorithm does not outperform current state-of-the-art models due to the simplicity of the used parts, but it is shown that dynamically learned weights can increase the tracking performance of the basic modules slightly.

The implementation and results will be made publicly available at: `https://github.com/bmmtstb/dynamically-gated-similarities`.
The modules are structured in a way that they can be easily replaced by more complex models, which should increase the overall performance of the dynamically gated similarities model. Thus, this thesis only provides a proof of concept and baseline models for the dynamically gated similarities algorithm.

Zusammenfassung

Das Verfolgen von Menschen und deren Körperhaltung ist eine wichtige Aufgabe für viele Anwendungen. Beispiele dafür sind die Mensch-Roboter-Interaktion oder die Videoüberwachung. Viele aktuellen Methoden verwenden den Ansatz erst, in einem separaten Schritt, alle zu verfolgenden Objekte zu erkennen. Dieser Ansatz hat sich als sehr erfolgreich erwiesen. Diese Methoden verwenden zur Verfolgung entweder die Körperhaltung inklusive der Bewegung von erkannten Personen oder ihr visuelles Erscheinungsbild. In den letzten Jahren haben einige Modelle damit begonnen, heuristische Ansätze zu verwenden, um die Informationen über die Körperhaltung mit weiteren Informationen über das Aussehen in einem Modell zu kombinieren. Diese Algorithmen verwendeten bisher meist feste Gewichte für die Kombination. Diese Gewichte sind datensatzabhängig und müssen deshalb manuell angepasst werden.

In dieser Arbeit schlagen wir ein neues Modell vor, das dynamisch gelernte Gewichte verwendet, um die Körperhaltung, das Aussehen und andere Informationen zu kombinieren. Das sogenannte Modell mit dynamisch gesteuerten Ähnlichkeiten (DGS) ist hochgradig modular konzipiert und kann vom Nutzer fast beliebig angepasst werden. Diese Gewichte können weitgehend datensatzunabhängig trainiert und verwendet werden. Da die Gewichte individuell für jede Erkennung berechnet werden, kann sich das Modell dynamisch an die aktuelle Situation anzupassen. Das DGS-Modell wird auf den Datensätzen PoseTrack21 und DanceTrack evaluiert. Der vorgeschlagene Algorithmus übertrifft die aktuell besten Modelle, aufgrund der Einfachheit der verwendeten Teile, nicht. Es kann aber gezeigt werden, dass dynamisch gelernte Gewichte die Gesamtleistung der Basismodule leicht erhöhen können.

Die Implementierung und die Ergebnisse werden unter folgendem Link veröffentlicht: `https://github.com/bmmtstb/dynamically-gated-similarities`.
Die Module sind so aufgebaut, dass sie leicht durch komplexere Modelle ersetzt werden können, was die Gesamtleistung der Modelle erhöhen dürfte. Diese Arbeit liefert daher nur einen Konzeptnachweis und Basismodelle für das DGS-Modell.

Acknowledgments

First, I want to thank the team of IAS at TU-Darmstadt for the opportunity to write my thesis in this great environment. Namely, Jan Peters and Suman Pal for their support and leadership on this project. Additionally, huge thanks to Arjun vir Datta and Felix Kaiser for helping me find the right topic.

I would also like to thank my proofreaders, without whom this thesis would have been much more challenging to read.

Lastly, I want to thank my family and friends for their support and understanding during the last months.

Contents

List of Abbreviations

Notation	Description
AssA	association accuracy
bbox	bounding-box
cMOTA	custom MOTA
CNN	convolutional neural network
DanceTrack	DanceTrack dataset
DetA	detection accuracy
DGS	Dynamically-Gated Similarities
HOTA	higher order tracking accuracy
IoU	intersection over union
key point	key-point coordinate
`KeypointRCNN`	`keypointrcnn_resnet50_fpn`
MOT	multi-object tracking
MOTA	multiple object tracking accuracy
MPPT	multi-person pose tracking
NN	neural network
OKS	object-keypoint similarity
PT21	PoseTrack21 dataset
re-ID	re-identification
TbD	tracking by detection
TbR	tracking by regression

List of Shapes

Notation	Description
$\mathbb{B}$	Batch-size
$\mathbb{D}$	Number of detections in a single frame or in a batch of frames
$\mathbb{E}$	Size of the embedding. Typically 512 for the visual embeddings returned by the `torchreid` models
$\mathbb{H}$	Height of the original image
h	Height of a modified or cropped image
$\mathbb{J}$	Number of joints or key-points
$\mathbb{T}$	Number of currently tracked Tracks
$\mathbb{W}$	Width of the original image
w	Width of a modified or cropped image

List of Figures

List of Tables

1. Introduction to Human Pose-Tracking

The field of detecting and tracking humans in videos is an active field of research [38, 47]. The potential use-cases of these results are relevant for countless real-world applications that might already influence our day-to-day life. Such applications range from detecting humans for autonomous driving [6, 25], to being able to tell whether athletes perform exercises correctly [16, 24, 32], or for different kinds of (autonomous) security [9, 22, 40]. The success of these downstream tasks is contingent upon the accuracy of the detection and tracking of humans and their limbs.

For this reason, a number of challenges and corresponding datasets have been developed with the aim of evaluating the performance of different algorithms. All of these challenges share the need to first detect the humans in the given images or videos. The detection of humans in images or videos can be done by importing pre-trained, state-of-the-art models. Those models are often highly optimized, while still being general enough to work well on a wide range of datasets. This leaves the main challenge of actually tracking the identity and position of all visible humans over time. And this is no small task due to the high number of occlusions, long-term data-relations, camera movement, and overall noise in the video data.

Most of the early models use the positional information and movement of people to track them over time. Later models take advantage of advances in convolutional neural networks (CNNs) and have started to use the visual appearance to distinguish between people. More recently, a few models have used a combination of both the pose and the visual appearance information for tracking. These models use fixed weights that are chosen to be constant for each of the datasets. This means that the weights cannot differ between the frames, and the models therefore lack generalizability.

Intuitively, when another person partially or completely occludes the appearance of a person, the model should not rely on the visual appearance to track the occluded person. The object in front of an occluded person has considerable influence on their visual appearance. Consequently, a model based on the visual appearance would encounter

greater difficulty in distinguishing between the two individuals. In this scenario, a pose-based model may contain more relevant information from the part of the person that is still visible. However, visual similarity is indispensable for re-identifying a person based on their appearance if a person has been occluded for an extended period of time and the pose-based model has lost track of that person. The reason for this is that the visual appearance of a person does not change as much as the pose-based information over shorter time periods of a few seconds. We expect that the performance of these models could be improved by dynamically combining the information gained through visual and pose-based analysis in a frame-by-frame manner. This would allow the model to adapt to the current situation within the current frame, deciding on which of the aspects to focus.

Therefore, this thesis proposes a new algorithm for multi-object tracking (MOT) and multi-person pose tracking (MPPT) called Dynamically-Gated Similarities (DGS). The main goal of DGS is the reduction of the number of hand-crafted hyperparameters that must be found for every dataset, while providing a highly modular and customizable pipeline. This is done by introducing a per-frame method of computing a weighted similarity between detections and tracks. These so-called alpha-weights are trained using a regression-based approach. Those weights should allow for more modularity and dataset independency.

We will have a more thorough look at the tracking task including its respective metrics, datasets, and some baseline algorithms in chapter 2. Then, in chapter 3, the proposed DGS algorithm is presented, followed by the experiments (chapter 4) and the analysis (chapter 5).

2. Foundations

This chapter introduces the core concepts of human pose-tracking that are needed to understand the proposed algorithm. A definition and some history of tracking in computer vision will be provided in section 2.1. The metrics used to evaluate the performance of the tracking algorithms are described in section 2.2. Finally, the used datasets are presented in section 2.3.

2.1. Tracking

As mentioned in the introduction, there are multiple tasks related to tracking humans in images and videos. The overall goal of each of the tracking algorithms is to re-identify and re-locate one or multiple objects over the course of a video or multiple images. In this thesis, only algorithms for tracking humans through images and videos will be considered, even though the tracking of objects is closely related and could follow a similar path.

For the person re-identification (re-ID) task, the images of multiple humans need to be compared against each other to find the same person in different videos. In order to simplify the task, the human detections are given. That is in contrast to the person search task, where the algorithm first needs to find all humans in the images and videos, before it can compare them against each other. Therefore, it is mandatory to detect all the humans in a given frame. Most models solve this task of detection by finding the smallest box containing the person. This so-called bbox can then be used to extract the part containing this person from within the full-sized image. This part of the image is called the image-crop of that detection. To achieve the end goal of re-identifying the search- or query-person in the target video frames or images, either the bboxes or the visual appearance of the person is used. Those target videos can contain multiple different instances and can even be made using other cameras or a diverse perspective. Models that perform well are currently used in surveillance systems to find a target person in a

large dataset of videos. While the overall goal of the person search and re-ID tasks is to find the same person in different images and videos, a single person does not need to be tracked over time.

Now let us switch to a different task by exploring the vision system of a self-driving car. The car needs to track the position and speed of all the other cars and pedestrians around it. This is a typical multi-object tracking (MOT) task with mixed objects. For MOT the identity and location of a person or other objects is tracked over time. Tracking means that the overall goal is to correctly follow instances across the whole duration of the video. This is done by detecting the bbox of every instance and comparing either it or their respective content against previously seen data. After that, the respective instance- or track-ID can be updated for each of the detections after every frame. New or otherwise unseen detections of subsequent frames are then matched to the previously detected instances or their respective track-IDs. MOT generally considers only instances in a single video and not across videos. Considering the toy example, this means that the next time the car is driving, it might see the same person or car, but their old location or other information is not relevant anymore. The car needs to detect and track the person or car again, as if it was the first time it saw them.

The multi-person pose tracking (MPPT) challenge is based on the MOT challenge, except it is only tracking humans. Additionally, instead of detecting only the bboxes for every human, the full body pose and the respective kps need to be localized. The kps are basically the locations of the body joints in image coordinates. The key point dimensionality, the number of kps and their respective location is dependent on the used dataset and detector. As an example for this task, one can imagine a health app that records a person exercising, while the app automatically tracks what exercises the person does, and can further tell what the person did wrong in that exercise [2, 19]. The algorithm can choose to use the additional information of the kps to improve the tracking performance. But it does not have to, as we will further discuss in section 2.1.2.

2.1.1. Tracking by Detection or by Regression

A tracking algorithm that first detects all humans in a given frame and does the tracking later, is following the tracking by detection (TbD) paradigm. This approach is commonly used due to its simplicity. A pre-trained state-of-the-art detection model can be used to find all humans in the given frame in a first step. This detection model is separate from the tracking model, which means that the detection model can be trained using other tasks and thus, other datasets. Due to the high demand for detection models, there are multiple

public state-of-the-art models available, that yield near perfect results. This means that for tracking, which is the second step, a much smaller and independent tracking model can be used. One small downside of this approach is that the detection model cannot obtain more information from the tracking model.

But there is a second approach for the overall tracking algorithm called tracking by regression (TbR) using a totally different structure. For this approach, a single but much larger regression model is used, which trains to predict the location and the movement of one or multiple objects in consecutive video frames while also doing the tracking part. Thus, instead of using Kalman filters to predict the object's movement, the model has to intrinsically learn the parameters to predict the updated bboxes and kps. In the end, this approach does detection and tracking in a single end-to-end model. But due to the fact that this larger model has to be re-trained every time, this approach is only used sporadically [3, 48].

All things considered, the TbD approach is chosen for the proposed algorithm due to its speed, simplicity, and the availability of pre-trained detection models. To speed up the overall processes even more, the detections and respective image crops can be precomputed for the train-, validation-, and test-set.

2.1.2. Appearance- and Motion-Based Models

For a long time, motion-based models dominated TbD because of their simplicity [21]. Motion-based models use time-based positional data to track objects over time. Most models use some form of Kalman filter [21, 27, 33] and its internal velocities to warp the previous detections to the current time-frame. As a result, the previous and current positional states are closer, making matching easier. The detections and the warped results can then be compared using similarities. For example, the IoU and OKS similarities described in section 2.2.

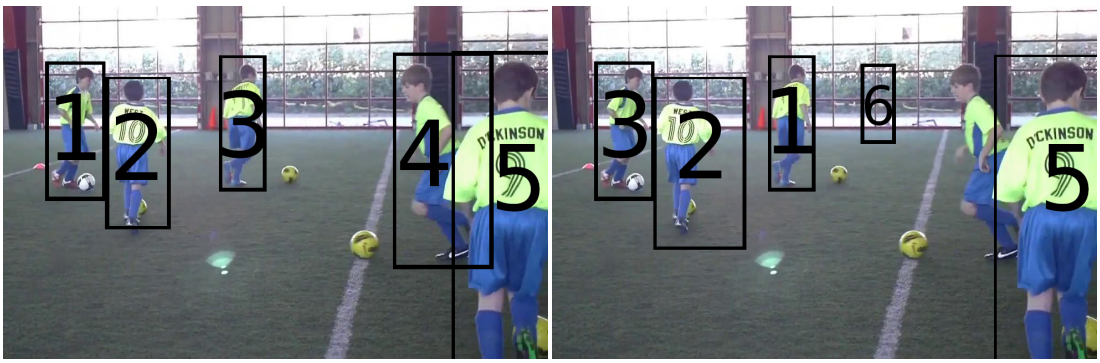
Many TbD-based algorithms use those similarities to obtain some kind of cost matrix. Then the Hungarian algorithm [23] or similar methods are used to obtain the solution with maximum similarity by solving the cost matrix as an assignment problem [5, 7, 14, 30]. Some models remove uncertain detections before the assignment to increase the overall tracking performance. After solving the assignment problem, every certain detection is matched to an existing or a new track. These steps are repeated for every frame of the video.

Motion-based models work great when tracking many humans with slow and steady movements, like it is done for surveillance cameras [22, 40]. But the results of the Kalman filter-based tracking tend to fail for the dance data of DanceTrack with highly irregular motion (see section 2.3.2). That is why `MambaTrack+` [15] replaced the Kalman filter update with learned motion models based on neural networks (NNs) with some success. Still, the motion information is not robust enough over longer periods, resulting in failures to re-ID people when they are occluded for a longer time.

That is why, visual re-ID models have started to surpass purely motion-based models with the development of CNNs. These models work by computing a single vector describing the person's appearance in the detected image crop. This so-called visual embedding is then trained to be as distant as possible for each of the individual detections, while being as close as possible to detections of the same person. What exactly a model saves in the visual embedding is not known, but for interpretability you can think of it as the model's interpretation of the color of the T-shirt, the texture of the clothes, the person's skin tone, or any other visual markers that the model can find.

These visual embeddings are the reason why visual models like OSNet [46], OSNetAIN [45], DeepSORT [41] work well for datasets of diverse crowds, because the embeddings can be trained more easily with larger visual differences. They can even be used for long-term tracking, like in XMem [8], because the visual appearance of a human does not change if it was occluded for 30 or more seconds. Appearance-based models even work when tracking people over multiple cameras with different view-points. This is why this approach is regularly used for tracking CCTV footage with dozens of cameras [9, 40]. But all visual models that rely solely on appearance have a hard time tracking team sports or other videos where people are clothed highly similar.

On a side note, the size of the image crops varies between the different algorithms. Earlier models often used square sizes of 128 by 128 or later 256 by 256 pixels, but most classical datasets only contain upright or standing humans. For CCTV footage or similar, a resolution of 256 by 128 pixels is used, because the humans are often further away from the camera, and most of them are standing, resulting in an elongated bbox. OSNet [46] and other newer approaches used a middle-ground of 256 by 192 pixels, which does not have too much padded borders in case of a standing human, but can easily display a wide variety of motion. Those pose-variations include a human laying down, dancers with their arms out, or a human in a sitting position. All these image crops are wider than a standing person, but most cases are not wide enough to have a square bbox. Because the DGS model uses the OSNet model for visual embeddings, a crop size of 256 by 192 is used.



(a) Exemplary ground-truth detections.

(b) Exemplary detections by the backbone.

Figure 2.1.: Toy example using a single image from PT21 (test — 000004_mpiinew_test — 000010.jpg). The image on the left-hand side (fig. 2.1a) shows exemplary detections using the ground-truth data. Figure 2.1b shows the detections using the backbone model. The backbone model missed the person with ID 4 on the right side of the image, but detected a ghost with ID 6 in the middle of the image. Additionally, the bbox of the person with ID 2 is too large and the tracking algorithm switched the IDs of the persons with ID 1 and ID 3.

All in all, it seems that both visual and pose-based approaches have their advantages and disadvantages, and they can be used in different scenarios. Still, the human has to decide, which approach to use for the respective task. A model that does so on its own would be beneficial. Another option is an algorithm that decides on a frame-by-frame basis, which of the approaches to use. As promising as that sounds, how does the algorithm know or learn what to focus on? There are already models trying to solve that using heuristic approaches [30, 37]. But those heuristic models lack generalizability due to their intrinsically fixed parameters. Hu et al. [14] as well as Doering and Gall [10] use a more dynamic approach by merging visual and pose-based embeddings before tracking them. Still, both approaches combine the generated embeddings using a fixed ratio. GeneralTrack [37] started to allow for better generalizability and removed some of the necessity of balancing motion and appearance. They created a four-dimensional point-wise correlation tensor and extracted features and regions of interest using multiple CNN-based approaches. But the model still used only parts of the appearance information and focussed more on the motion information.

2.2. Metrics

To evaluate the performance of the different models, multiple performance metrics are used. To get a better understanding of the metrics, a toy example showing some of the typical problems in tracking is shown in fig. 2.1. The first cause for mistakes is the detection model. This can be seen in fig. 2.1b, where the backbone model missed the person with ID 4 on the right side of the image. A missed detection counts as a false negative detection. Additionally, the backbone model detected a ghost with ID 6 in the middle of the image, where no person is visible in the ground-truth data. This type of error is a false positive detection. And even if the person was found, the bbox can be in the wrong location or have the wrong shape, as can be seen for the player with ID 2 in the image.

The second cause for mistakes is the tracking algorithm. In the case of the toy example, the tracking algorithm mistakenly switched the IDs of the persons with ID 1 and ID 3. This is called an ID-switch. During tracking, there are even more types of association problems that can occur, which can be seen in fig. 2.2 [28]. For an in-depth explanation of the topic, read the paper or the online demo by Luiten et al. [28].

Now that we have seen some of the typical problems in tracking, we can look at the metrics used to evaluate the performance of the tracking algorithms. The first metric is the multiple object tracking accuracy (MOTA) proposed by Bernardin, Elbs, and Stiefelhagen [4]. It is defined in eq. (2.1). The MOTA is used for a broad overall tracking accuracy using the number of false detections and the association error.

$$\mathtt{MOTA} = 1 - \frac{|\mathrm{FN}| + |\mathrm{FP}| + |\mathrm{IDSW}|}{|\mathrm{TP}|} \qquad \begin{cases} \text{FN: nof false negatives} \\ \text{FP: nof false positives} \\ \text{IDSW: nof ID switches} \\ \text{TP: nof true positives} \end{cases} \tag{2.1}$$

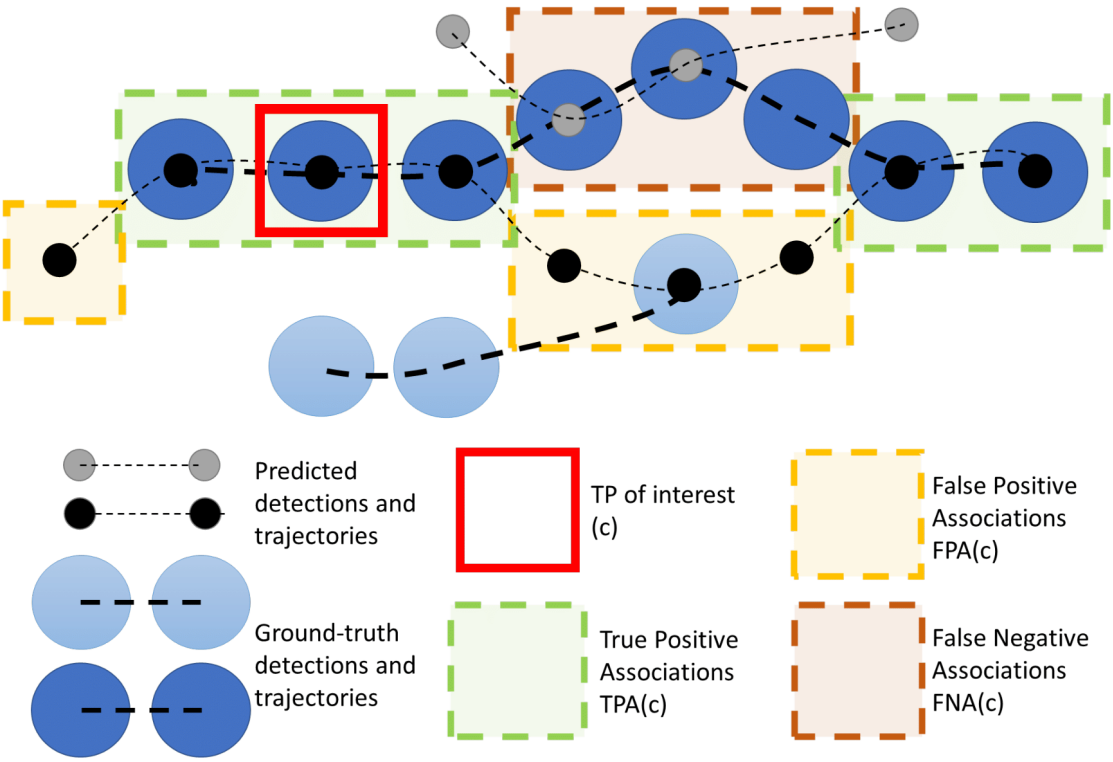


Figure 2.2.: Association errors during tracking. Credit to Luiten et al. [28].

The most important metric — the higher order tracking accuracy (HOTA) — is used as an accuracy for evaluating the performance of the tracking models. The HOTA was proposed by Luiten et al. [29] and is defined by eq. (2.2). While the MOTA focuses more on the localization errors of the detections, the HOTA is a more general tracking metric. It is implemented to balance between tracking related errors and the localization error.

$$\mathsf{HOTA} = \int_0^1 \mathsf{HOTA}_\alpha \, d\alpha \tag{2.2}$$

$$\mathsf{HOTA}_\alpha = \sqrt{\mathrm{DetA}_\alpha \cdot \mathrm{AssA}_\alpha} = \sqrt{\frac{\sum_{c \in \{\mathrm{TP}\}} \mathcal{A}(c)}{|\mathrm{TP}| + |\mathrm{FN}| + |\mathrm{FP}|}} \tag{2.3}$$

$$\mathcal{A}(c) = \frac{|\mathrm{TPA}(c)|}{|\mathrm{TPA}(c)| + |\mathrm{FNA}(c)| + |\mathrm{FPA}(c)|} \quad \begin{cases} \text{TPA: True Positive Associations} \\ \text{FNA: False Negative Associations} \\ \text{FPA: False Positive Associations} \end{cases} \tag{2.4}$$

The HOTA uses the number of correct associations between the ground truth and the predictions (true positive), the number of missed detections (false negative), and the number of false or additional detections (false positive). These values are first computed with a threshold α and then the total HOTA is integrated over all thresholds from 0 to 1. For both, MOTA and HOTA, the maximum value is 100, while in theory the minimum value is negative infinity, a value of 0 is considered inadequate.

The respective tools of the datasets are used for computing the HOTA and all the other metrics during evaluation. To obtain the final accuracies on the test data the results were uploaded to the respective scoring servers of the datasets.

Within the evaluation tools, the HOTA is often returned in tandem with multiple other values including the IDF1 score, the association accuracy (AssA), and detection accuracy (DetA). While the HOTA balances the importance of detection and tracking, the IDF1 score mainly focuses on the tracking part. The IDF1 score is computed as the harmonic mean of precision and recall. It focuses on how long a tracker correctly identifies the person or object. The DetA measures how well a given tracker localizes the bboxes in each frame, given IoU thresholds. The AssA describes how accurately the tracker tracks the person's identity over the course of the video. Both, DetA and AssA, are computed by matching the predictions of the current frame and the ground-truth detections for every frame of the video. Those metrics are used to compute the HOTA.

Additionally, the IoU (eq. (2.5)) is needed for multiple pre-steps of tracking, including the internal matching of the IDF1 and HOTA accuracies. The IoU is computed, as the name suggests, as the area of intersection divided by the area of union between two bboxes. The IoU lies in a range from zero to one, where zero means that there is no overlap between the two bboxes and one implies identical boxes.

In some modules within DGS, the OKS (eq. (2.6)) is used to compute the similarities between sets of kps. The OKS is computed as the average Euclidean distance (d_i) between a ground-truth key point and the prediction. Each distance is scaled by a per-key-point constant (k_i). The values of k_i have been defined by COCO [26] and remain unchanged. With s being the ground-truth scale of the object, s^2 becomes the object's segmented area. The Dirac-delta function ($\delta\left(v_i > 0\right)$) returns 1 if the key point i is labeled and visible and is 0 otherwise.

$$\mathtt{IoU} = \frac{\left|\mathbf{A} \cap \mathbf{B}\right|}{\left|\mathbf{A} \cup \mathbf{B}\right|} = \frac{\mathtt{intersection}}{\mathtt{union}} \tag{2.5}$$

$$\mathtt{OKS} = \frac{\sum_i \mathtt{KS}_i \cdot \delta\left(v_i > 0\right)}{\sum_i \delta\left(v_i > 0\right)} = \frac{\mathtt{key\ point\ similarity}}{\mathtt{nof\ labeled}} \tag{2.6}$$

$$\mathtt{KS}_i = \exp\left(\frac{-d_i^2}{2s^2k_i^2}\right) \tag{2.7}$$

2.3. Datasets

There are two datasets used for the experiments. Both have different properties and were developed with slightly different objectives in mind. The datasets are chosen to show the adaptability of the proposed algorithm in different scenarios regarding motion complexity, target density, and target size. In fig. 2.3 some example images of both datasets are shown.

(a) PT21 — train
004334_mpii_train
000040.jpg

(b) PT21 — train
000022_bonn_train
000000.jpg

(c) PT21 — train
000866_mpii_train
000063.jpg

(d) PT21 — val
024007_mpii_test
000006.jpg

(e) DanceTrack — train
dancetrack0002
00000016.jpg

(f) DanceTrack — train
dancetrack0080
00000393.jpg

(g) DanceTrack — val
dancetrack0007
00000248.jpg

(h) DanceTrack — val
dancetrack0004
00000306.jpg

Figure 2.3.: Example images of PT21 (figs. 2.3a to 2.3d) and DanceTrack (figs. 2.3e to 2.3h)

2.3.1. PoseTrack

The larger dataset, is the PoseTrack21 dataset (PT21) [11], which is based on PoseTrack18 [1]. It contains 593, 170, and 375 videos in the training, validation, and test datasets respectively. The training videos have a length of two to five seconds with a densely annotated sequence of 30 frames in the middle of each sequence. The validation and test data are densely annotated with a step size of four frames.

The dataset contains annotations for three challenges. Namely, MPPT, MOT, and person search. PT21 provided the first dataset, where the different challenges could be worked on simultaneously, thus allowing models to be trained on different tasks in parallel. For this thesis, only the annotations of the MPPT challenge are used, because the other annotations do not contain the key point information. The bboxes, the head-bboxes, and the kps are annotated, including whether the respective key point is visible.

The videos in PT21 contain different scenarios from everyday life and outdoor activities to fast-paced Olympic sports. A few exemplary images can be seen in figs. 2.3a to 2.3d. Due to the large variety of videos, the dataset contains frames with zero detections, only a few detections, or up to a dozen humans in a single frame. The poses vary from walking, standing, and sitting to extreme poses while bungee-jumping, synchronyzed swimming, or floor exercises in gymnastics.

During analysis, the evaluation tools of the dataset [11] are used to compute the MOTA of PT21. The `trackeval` [20] package is used to compute the HOTA.

Hyper-Parameter Search

Most detection models return the detection score, that is the model's certainty of the detection being correct. To speed up training and inference, the bboxes and kps of the training and validation sets were pre-computed, extracted, and stored. First, the given image masks were used to blacken out unlabeled sections of the images. Then, a backbone model is used for detection, which is `keypointrcnn_resnet50_fpn` (`KeypointRCNN`) for all further experiments. `KeypointRCNN` is a model by `PyTorch` [34, 35] and is based on the Mask R-CNN [13] model.

There are notable duplicates and non-human detections without filtering the detections of the backbone model. Therefore, the decision was made to run a hyperparameter grid-search over the validation set. The detection score of the backbone model and the maximum IoU between two detections are the two hyperparameters that are analyzed dataset-wide. The full analysis of the used values can be found in appendix A.1 on page 51. The thresholds found in this grid-search are then used for all the experiments in chapter 4.

In addition to the two thresholds, all bboxes with a minimal side-length of less than 50 pixels are removed. The length is calculated in pixel coordinates of the original image. The detection-score threshold removed most of the small bboxes. Hence, only a few detections containing the head or other small body parts were removed by the minimum size threshold.

All this extra work is necessary because only every fourth image is labeled completely, and therefore there are still thousands of unlabeled humans. Additionally, the image masks of unlabeled areas are not given for every image of every video, even if there are additional humans visible. Like with fig. 2.3a these people are mainly in the background and in the outer sections of the image. There are even some cases where the backbone model detected humans in masked areas because their shadow, reflection, or just a single hand is visible beyond the blackened boundary. This is because the backbone detection modules are getting better and better.

Usage of Custom MOTA

Additionally, because `KeypointRCNN` is used as the backbone model to predict the bboxes and kps, there are kps returned that slighly differ from the original definition in PT21. Namely, `KeypointRCNN` returns the `left_eye` and `right_eye` instead of the `head_bottom` (or sometimes called `neck`) and `head_top`. This means that the results

for those two kps will always be shifted by a certain amount and thus can never be correct. Therefore, a slightly modified MOTA score has to be used for a fair comparison. It will be denoted cMOTA in the experiments. cMOTA is computed as the average MOTA score over all 15 kps that are present in both PT21 and `KeypointRCNN`. Nevertheless, there will be results published of the final model that include all the kps.

This could have been avoided by fine-tuning the existing backbone model on PT21 or by using a different model that returns the correct kps. This was not done because the backbone model is already trained on a large dataset and the resulting scores of DGS are not that good either way.

2.3.2. DanceTrack

The DanceTrack dataset (DanceTrack) [39] is a dataset containing 100 videos of dancers. The dataset is split into 40 videos for training, 25 for validation, and 35 for testing. The videos are longer than the ones of PT21, with an average length of over 50 seconds. The dataset contains annotations for the MOT challenge — thus meaning that the kps are not annotated at all. The bboxes, however, are densely annotated over all frames of the dataset.

Dance poses vary in appearance from regular standing, and the poses might not be upright at all. This means that the detection model has to be able to locate humans even in unnatural poses. The authors of DanceTrack [39] mentioned a lack of models that do not rely on visual appearance, thus having a hard time with the uniformly dressed group of dancers. Incidentally, the synchronized movements of the dancers render the kps in bbox-local coordinates almost completely useless. Lastly, during some of the dance sequences, up to five dancers move in a straight line behind each other (see fig. 2.3h). These sections increase the overall difficulty of the tracking tasks drastically.

A hyperparameter search was done on DanceTrack similar to the one of PT21 (see section 2.3.1). The results can be found in the appendix (see appendix A.2 on page 55).

The dataset is evaluated using the official evaluation functions from `trackeval` [20] to get the MOTA and HOTA metrics. A custom MOTA metric is not required.

3. Dynamically Gated Similarities

This thesis proposes the algorithm called Dynamically-Gated Similarities (DGS). The main idea of DGS is to create non-heuristic dynamic-weights for each of the similarity functions. Those are then used to compute a weighted sum of all the similarity-matrices. This weighted similarity matrix is then used as the cost matrix for matching and assignment.

3.1. The Idea Behind DGS

But let us start from the beginning. If you want to follow along, a simplified visual representation of the pipeline can be seen in fig. 3.1.

First, following the regular TbD approach, a detection model is used to detect all humans in the current frame, including their respective bboxes, kps and the resulting image crops. As mentioned, the detection model is `KeypointRCNN` for all experiments. Then, two or more similarity models can be used to compute the base similarity matrices. This similarity is computed for each of the pairs between the current detections and all existing previous tracks. These similarity modules can be nearly arbitrary in how complex they are, but it is easier and safer to re-use previously tested functions. Therefore, within DGS a mixture of IoU, OKS, and different embedding-based visual similarities are used to obtain the base similarity scores. We will have a longer look at the different similarity functions in section 3.1.1.

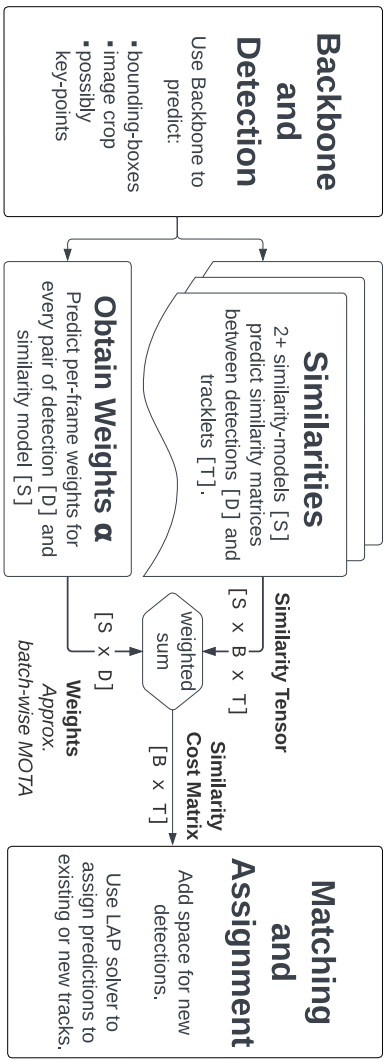


Figure 3.1.: The basic DGS pipeline. Given backbone image detections, obtain similarity matrices and weight them according to the alpha-weights. Finally, create a weighted sum of all similarities and use the result as a cost matrix.

However, the similarity scores only dictate how similar the two states are, not whether the models are certain that those scores are correct. That is the reason why the similarity confidence weight is computed next. This computation is done in the so called alpha-modules for each of the similarity functions and for every detection in the current frame. These alpha-weights represent whether the algorithm should be confident in the similarity score. This means intuitively, that the weight for the visual model should be lower for an occluded person. The plain visual similarity between this occluded person and to the person in front will be high because the occludor heavily influences the visual appearance of the occluded person. Therefore, this high similarity should be multiplied with a low confidence to obtain an overall lower weighted similarity.

In another situation, where a person has not been seen at all, the confidence and thus, the alpha-weight, of the pose-based similarity should decrease over time due to uncertain predictions in the pose-updates. And because these weights are computed independently for every detection in every frame, the model should be able to adapt to the current situation in the frame.

Using the predicted alpha-weights, a single weighted similarity matrix can be computed. This similarity matrix is then used — similar to a cost matrix in a linear-sum-assignment problem — to obtain the indices maximizing the total similarity score. Finally, with the indices, each of the current detections can be assigned to an already existing or a new track.

This process is repeated for the whole video in a frame-by-frame-based manner, updating the tracks with every frame. Old tracks are not deleted immediately after a person was not seen for a single frame. Allowing the detections to be re-assigned to inactive tracks, when a person has been occluded by another person for a few frames. To make sure that faulty tracks get deleted eventually, all tracks that have not been updated for 15 frames will be permanently deleted.

3.1.1. Similarity Functions

The goal of any similarity function is to compute a score describing how close the two given input-values are on a scale from 1.0 to 0.0. A score of 1.0 means the values are identical. The IoU and OKS have already been defined in section 2.2 by eqs. (2.5) and (2.6) and are used to compute the similarity between bboxes and kps respectively. This means that no additional data is needed and the similarity can be computed directly from the detections. Pose-based tracking algorithms like SORT [5] and OC-SORT [7] used those

metrics successfully in combination with the Kalman-filter-based pose-warping. Both metrics can be parallelized and are computed using basic vector math, which means they are incredibly fast to compute.

To be able to re-ID people even after longer periods of occlusion, it was opted to additionally use a visual similarity. For the visual similarity, the approach for obtaining the similarity scores is a little more complex. First, for every detection its respective visual embedding is computed from the image crop using a re-ID-model like OSNet [46]. Those models and weights were made publicly available by `torchreid` [44]. This also results directly in an embedding size of 512 for most of the models, but the size can vary. Secondly, the cosine similarity can be computed between the embeddings of the tracks and detections. To speed up training and evaluation times, the computed visual embeddings are stored in the states of the respective detections.

Looking back at the pictures in fig. 2.3, all the dancers in the DanceTrack images are wearing similar clothes, making it hard for a visual similarity to distinguish between them. But the same visual model will have an easier time differentiating between the two wrestlers (see fig. 2.3a) or the two people in the lawn mower image (see fig. 2.3d). Additionally, the visual similarity plays a crucial part in long-term tracking. Even though the positions of the people might change drastically, their appearance stays roughly the same.

The IoU similarity on the other hand will have a hard time distinguishing between highly overlapping people, like the stacked dancers on fig. 2.3h or when the two wrestlers start butting their heads a few frames later. The IoU and OKS both compute distances between the respective detections. Both have access to the bounding box, the OKS only indirectly through the outermost kps of the limbs. But intuitively, because the IoU similarity only has 4 values to work with, compared to the $2 \cdot 17 = 34$ values for the OKS, the IoU is expected to perform worse overall.

For this proof of concept, no time-warping model is used to make sure that the detections and the tracks are in the same frame.

3.1.2. Training Independent Dynamic Weights

To have the most modularity, the trained weights are only dependent on the currently used similarity module. This modularity is chosen to ensure that the model can be used across datasets. It has the advantage that the IoU model trained on PT21 can be used on DanceTrack because it does not depend on the kps of PT21 which are technically not

available in DanceTrack. Additionally, the trained models can be used independently of the used number of similarities or their respective configuration. This means that the similarity functions can be chosen dataset-dependent, but the models predicting the weights do not necessarily need to be retrained for new datasets. This should allow for a pipeline that is straightforward to use and extend. New modules can be added without the need to retrain the weight generation modules of already existing modules each time or for each new combination.

But how are the weights trained? The basic training pipeline can be seen in fig. 3.2. The training follows the approach of a supervised regression model, where the target weights are approximated by a small NN module. The target is chosen to be similar to the batch-wise MOTA score. This means that the alpha-weight module will be trained to predict the tracking accuracy of the batched detections. The following explanation will be for a visual or otherwise embedding-based similarity function.

At time-step t_1, compute the visual embeddings $\mathbf{E}_{t_1} \in [\mathbb{D}_{t_1}, \mathbb{E}]$ for all detections $d_{t_1} \in \mathbb{D}_{t_1}$. Make sure that the embeddings $\mathbf{E}_{t_0} \in [\mathbb{D}_{t_0}, \mathbb{E}]$ of the previous time-step (t_0) are computed or stored too. Both embeddings contain one value per detection and have a matching embedding-size of $\mathbb{E}$. The embedding size depends on one visual similarity module and is 512 for the models from `torchreid` [44].

For training the IoU and OKS modules, the embedding represents the normalized bboxes and kps respectively. The values have to be normalized to lie in the range of 0.0 to 1.0, to ensure correct training behavior. With unnormalized values, the input values for the NN would be too large and the training would become unstable.

The next step is to compute the unweighted similarity tensor as $\hat{\mathbf{S}}_{t_1}\left(\mathbf{E}_{t_0}, \mathbf{E}_{t_1}\right) \in \left[\mathbb{S}, \mathbb{D}_{t_1}, \mathbb{D}^{+}_{t_0}\right]$ using the embeddings. The similarity matrix has to be computed for each of the $\mathbb{S}$ similarity functions. Make sure to add empty columns for new detections, denoted here with $\square^{+}$, and to compute the softmax of the similarity matrix, here denoted $\hat{\square}$. Use the alpha-module to create a prediction of the alpha-weight $\boldsymbol{\alpha}_{t_1}$ using the current embeddings $\boldsymbol{\alpha}_{t_1}\left(\mathbf{E}_{t_1}\right) = \mathbf{W}\mathbf{E}_{t_1}$. Where $\mathbf{W}$ is the simplified weight matrix of the current alpha-module. In practice, the alpha-module is a small NN with a single output neuron with non-linear activation functions.

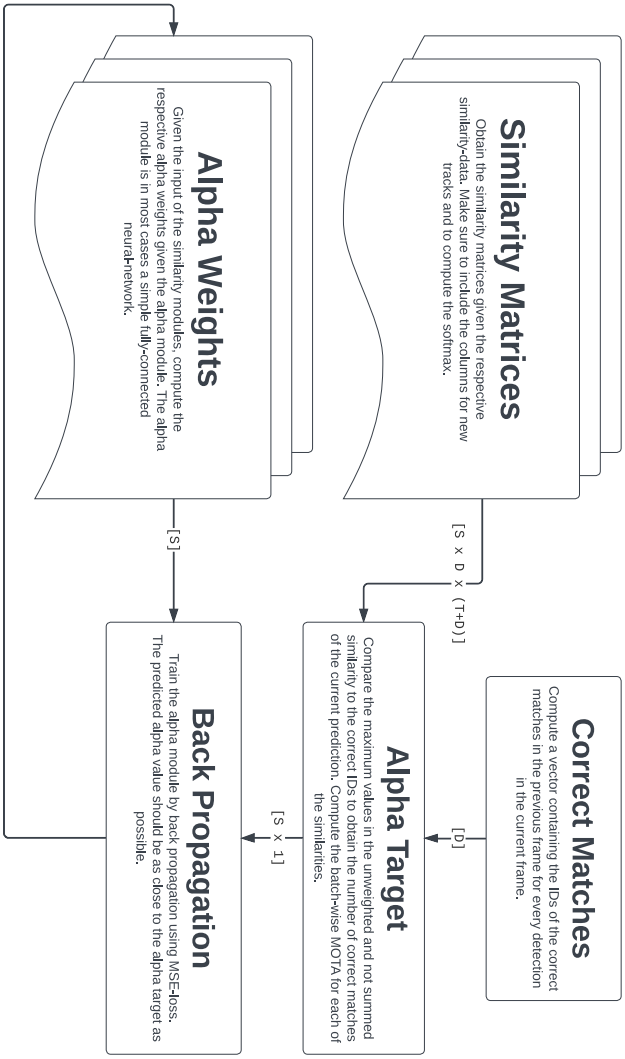


Figure 3.2.: Pipeline for training dynamic weights in the DGS module.

Continue by computing a vector containing the indices of the correct matches between the detections of the current frame and the previously detected tracks. This means that for each detection d_{t_1} in $\hat{\mathbf{S}}_{t_1}$ we have the index of the track t_{t_0} in the previous frame that is the correct match. This supervised learning approach is the reason why the training can only be done on labeled ground-truth training data. The resulting vector is denoted as $\mathbf{I}_{t_1} \in [\mathbb{D}_{t_1}]$. Using this vector, we can compute the number of correct matches the current similarity module has found, by counting where the indices of the correct matches are equal to the locations with maximum values in the similarity matrix. Then using 1 minus this value to obtain the incorrect matches. This is done to approximate the MOTA score of the current batch (see eq. (3.2)).

Due to the usage of the ground-truth labels, there are no false positives or false negatives in the detections. Thus, this batch-wise MOTA score should be a decent approximation of the similarity modules performance, excluding the performance of the detection model.

$$\boldsymbol{\alpha}_{t_1} \approx 1 - \frac{\texttt{nof incorrect matches}}{\texttt{total nof detections}} \tag{3.1}$$

$$\mathbf{WE}_{t_1} \approx 1 - \frac{\sum \left(\arg\max_{\texttt{dim}-1} \left(\hat{\mathbf{S}}_{t_1} \right) == \mathbf{I}_{t_1} \right)}{|\mathbb{D}_{t_1}|} \tag{3.2}$$

Finally, the loss can be computed between the predicted alpha-weights and the batch-wise MOTA. As the loss function, the cross-entropy loss is used. The loss is then back-propagated through the alpha-module. Hence, resulting in an alpha-module trained to predict whether the similarity function is useful given the current detections.

3.2. Experiments

To evaluate the performance of the proposed DGS pipeline, multiple experiments are run. The goal is to show, which parts contribute most under which circumstances. Most importantly, whether a dynamically trained weight can achieve a similar performance in comparison to the static versions or some of the state-of-the-art models.

3.2.1. On the Importance of the Initial Track Weights

Assume $\mathbb{D}$ detections and $\mathbb{T}$ previously seen tracks, then each of the similarity matrices is of shape $[\mathbb{D} \times \mathbb{T}]$. Every single detection d_i should be matched to one of the t_j tracks. Within the DGS pipeline it is not possible to match multiple detections to a single track, and every detection has to be matched to a track — detections cannot be skipped. During the matching phase, the cost matrix is used to assign the detections to the tracks using the combination of matches with the highest (weighted) similarity score. This means that if $\mathbb{D} > \mathbb{T}$ new tracks have to be inserted to be able to match every detection. In the code, this insertion is achieved by adding $\mathbb{D}$ additional columns to the similarity matrix. Using this approach, every detection can be matched to a new empty track. The initial weight of these new tracks has large influence over the effectiveness of the other modules. That is why the initial weight of new tracks is still one important parameter that remains fixed for DGS.

The first large batch of experiments therefore analyzes individual similarity models to find the best value for the initial weight. Using only a single similarity model means that there is no dynamic or static weight for the similarity matrices yet. Thus, the only variable will be the initial weight. The results will be discussed in section 4.1. For all future experiments, the initial weight parameter will be fixed according to these results.

3.2.2. Single, Pairwise, and Triplet Models

For the next big batch of experiments, the DGS module is evaluated using static weights and the fixed initial weight as discussed before. The previously done evaluation of the initial weights can also be used as a baseline of the capabilities for each of the individual similarity models, after the parameter-search in section 2.3.1 was done. This means that every similarity model is evaluated individually, resulting in a static weight of 100% for the current similarity model. Thereby, yielding a performance baseline for the importance of each similarity model for each of the datasets. The results of these experiments will be discussed in section 4.2.1.

Secondly, the models are evaluated in pairs. Every combination of the similarity models is run for each of the static weight combinations ranging in 10% increments from 10% to 90%. The best of these models will be the baseline to test the trained dynamic modules against. Additionally, it will be interesting to examine whether the similarity models that performed well individually do also perform better in pairs.

As the last batch of experiment with static weights, the triplet combinations are run. Again, all possible combinations are run, with the weights ranging from 10% to 80% with 10% increments. This evaluation should yield the best possible static weights for each of the two datasets. Furthermore, the results may indicate whether there exists a single best combination that is dataset independent. For both, pairwise- and triplet-evaluations, the results will be discussed in section 4.2.2.

3.2.3. Models with Trained Weights

Finally, the last batch of experiments will evaluate the modules using the trained dynamic weights. While the weight generation modules can be trained individually, they have to be evaluated and tested in combinations of at least two. This has to be done because the probability of choosing that model will always evaluate to 1.0 when running a single similarity module. This is due to the softmax function being used to get a probability distribution from the predicted weights. Therefore, the results of the static pairwise and triplet models will be compared against the results of the dynamic pairwise and triplet models in section 4.3.

4. Experiments and Results

First, the initial track weight is analyzed in section 4.1, followed by the baseline results of the constant models in section 4.2. After that, the results of the dynamic models are compared against the constant models in section 4.3.

4.1. Analysis of the Initial Track-Weight

Extensive experiments were run to obtain the initial weight of new tracks — a value that is used in all further experiments. The results for the initial track weights for the ground-truth validation data of PT21 can be seen in table 4.1. The results for the experiments on the validation data but using the `KeypointRCNN` backbone can be seen in table 4.2. In both tables, it is straightforward to see that for all the metrics and similarity functions, an initial track weight of 0.00 yields the best results. The same is true for the results on DanceTrack. The results of the evaluation on the ground-truth can be seen in table 4.3, while the results using the `KeypointRCNN` backbone can be seen in table 4.4.

Metric	Similarity Function	Initial Track Weight										
		0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
cMOTA↑	IoU	<u>98.23</u>	57.05	24.80	14.58	9.93	9.13	3.85	3.60	3.60	3.60	3.60
	OKS	<u>98.30</u>	56.83	24.62	14.64	11.00	9.60	4.18	3.60	3.60	3.60	3.60
	OSNet [44, 46]	<u>97.81</u>	55.20	22.92	16.43	11.97	8.74	7.93	3.60	3.60	3.60	3.60
HOTA↑	IoU	<u>93.13</u>	71.19	47.88	37.21	30.77	29.61	19.56	18.99	18.99	18.99	18.99
	OKS	<u>93.02</u>	70.96	47.84	37.37	32.45	30.42	19.87	18.99	18.99	18.99	18.99
	OSNet [44, 46]	<u>90.78</u>	69.22	45.70	39.34	33.58	28.93	27.33	18.99	18.99	18.99	18.99

Table 4.1.: Evaluation of the initial weight parameter for the ground-truth validation data of PT21. The best value per row is <u>underlined</u>.

Metric	Similarity Function	Initial Track Weight									
		Score Threshold: 0.00, IoU Threshold: 1.00					Score Threshold: 0.85, IoU Threshold: 0.40				
		0.00	0.10	0.20	0.30	0.40	0.00	0.10	0.20	0.30	0.40
cMOTA↑	IoU	-163	-229	-232	-233	-234	34.05	7.93	-18.98	-26.32	-29.51
	OKS	-164	-229	-233	-233	-234	34.07	7.76	-18.98	-26.36	-29.79
	OSNet [44, 46]	-164	-229	-233	-233	-234	33.81	7.38	-19.42	-25.15	-27.54
HOTA↑	IoU	18.88	8.46	7.42	6.89	6.85	34.77	28.07	17.89	13.89	11.73
	OKS	18.11	8.36	7.35	6.89	6.85	34.83	28.02	17.99	13.87	11.59
	OSNet [44, 46]	18.75	8.35	7.31	6.91	6.85	34.77	27.96	17.66	14.47	13.23

Table 4.2.: Evaluation of the initial weight parameter for the validation data of PT21 with detections of `KeypointRCNN`. The respective score- and IoU-thresholds are given per column. The best value per experiment is underlined.

Metric	Similarity Function	Initial Track Weight										
		0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
MOTA↑	IoU	99.67	52.37	7.87	0.14	0.12	0.12	0.12	0.12	0.12	0.12	0.12
	OSNet [44, 46]	98.52	51.74	6.32	0.22	0.13	0.12	0.12	0.12	0.12	0.12	0.12
HOTA↑	IoU	79.09	60.49	21.40	3.48	3.48	3.48	3.48	3.48	3.48	3.48	3.48
	OSNet [44, 46]	57.17	43.45	16.52	3.52	3.48	3.48	3.48	3.48	3.48	3.48	3.48

Table 4.3.: Evaluation of the initial weight parameter for the ground-truth validation data of DanceTrack. The best value per row is underlined.

Metric	Similarity Function	Initial Track Weight									
		Score Threshold: 0.00, IoU Threshold: 1.00					Score Threshold: 0.75, IoU Threshold: 0.35				
		0.00	0.10	0.20	0.30	0.40	0.00	0.10	0.20	0.30	0.40
MOTA↑	IoU	-44.80	-126	-126	-126	-126	66.15	30.35	-4.79	-6.76	-6.80
	OSNet [44, 46]	-47.66	-126	-126	-126	-126	64.86	29.95	-4.64	-6.61	-6.78
HOTA↑	IoU	25.18	2.35	2.32	2.32	2.32	44.36	32.24	7.11	3.00	2.99
	OSNet [44, 46]	20.36	2.32	2.32	2.32	2.32	35.81	27.35	7.32	3.13	2.99

Table 4.4.: Evaluation of the initial weight parameter for the validation data of DanceTrack with detections of `KeypointRCNN`. The respective score- and IoU-thresholds are given per column. The best value per experiment is underlined.

Note that the data of the KeypointRCNN experiments is only given to an initial-weight value of 0.40. This is because the results for the higher values are nearly identical to the value at 0.40 and the values for the ground-truth case yield the same information. Additionally, for both datasets the results of the KeypointRCNN model are given for the unfiltered dataset with a score threshold of 0.00 and an IoU threshold of 1.00 as well as for the filtered dataset with the values chosen by the hyperparameter search.

At first glance, it seems unintuitive that an initial track weight of 0.00 yields the best results overall. Looking at the ground-truth data first, there are in most cases a small fixed number of people in every video. Most of the people are visible after only a few frames. Thus after those first few frames, every ground-truth person has its own track. After that, there is simply no need for new tracks, thus increasing the weight of the new tracks only lowers the probability of matching with the already existing (correct) tracks — thereby lowering the matching scores.

But what about the experiments using the KeypointRCNN backbone? After the hyperparameter search, the detections should include only a few false positives, which in the end yields the same result as in the ground-truth case. There are nearly only valid detections, which will be matched with other valid detections, while most of the false positives are discarded before they get analyzed.

4.2. Constant Results

Before analyzing the results of the dynamic models, we first need to take a look at the results of the constant models as a baseline for all further experiments.

4.2.1. Single Models

Table 4.5 shows the results of the single models for DanceTrack and PT21 using the KeypointRCNN backbone. The best value per experiment is underlined. Even though the results on DanceTrack are really close for both the IoU and OKS, it is shown that for the MOTA metric the IoU similarity function yields the best results, while the OKS maximizes the HOTA. Due to the highly irregular motion of the dancers in DanceTrack, it was not expected that the HOTA metric is maximized by the OKS. But because the other values are close-by and the overall values are low in comparison to the results of other models, it just shows that a single model should not be used for the DanceTrack dataset.

		DanceTrack Score Threshold: 0.70 IoU Threshold: 0.35	DanceTrack Score Threshold: 0.75 IoU Threshold: 0.35			PT21 Score Threshold: 0.85 IoU Threshold: 0.40
MOTA↑	iou	<u>65.59</u>	<u>66.15</u>	cMOTA↑	iou	34.05
	oks	65.52	66.10		oks	<u>34.07</u>
	OSNet [44, 46]	64.28	64.89		OSNet [44, 46]	33.81
	Resnet50 [44]	63.52	64.10		Resnet50 [44]	33.44
	Resnet152 [44]	63.00	63.62		Resnet152 [44]	33.33
HOTA↑	iou	44.71	44.36	HOTA↑	iou	49.47
	oks	<u>46.19</u>	<u>47.11</u>		oks	<u>49.57</u>
	OSNet [44, 46]	35.74	35.92		OSNet [44, 46]	49.41
	Resnet50 [44]	33.94	33.24		Resnet50 [44]	48.59
	Resnet152 [44]	31.36	31.75		Resnet152 [44]	48.38

Table 4.5.: Performance of the individual (static) models for DanceTrack and PT21. The best value per experiment is <u>underlined</u>.

The results on PT21 show that the OKS similarity function yields the best results for both the MOTA and HOTA metric. This can be expected, especially because the values for all other individual metrics are close by, because PT21 does not have the same irregular motion as DanceTrack.

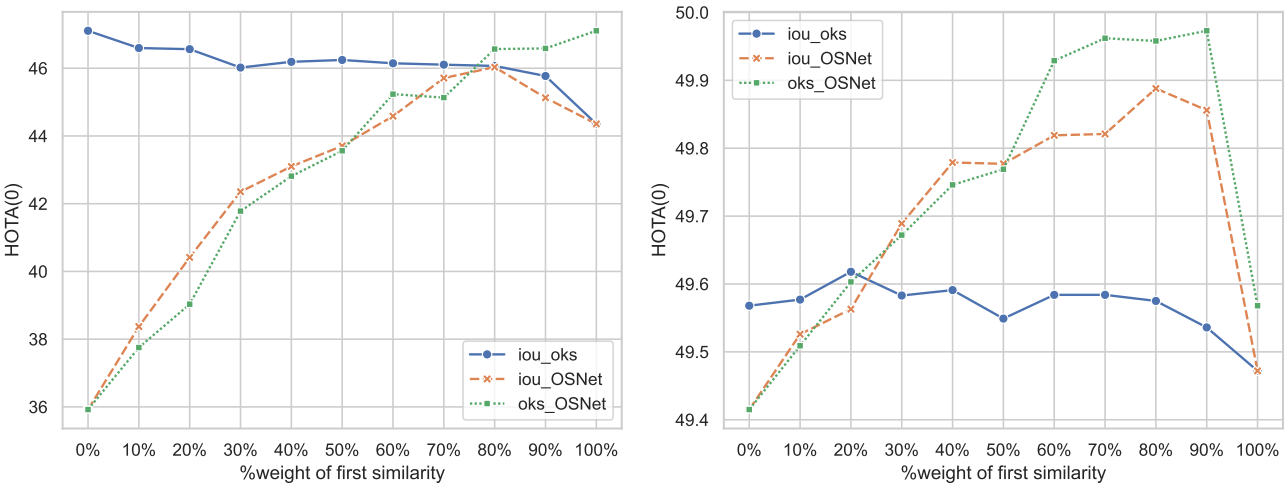
In table 4.6 the best results of the single models are compared against other models. The DGS module is marked in bold. On both datasets, the MOTA score of the single DGS modules is significantly worse than that of the external models. The HOTA score is a little closer, but still not as good as the external models. So let us see if the pairwise static modules can beat that performance.

For all following experiments with static weights, the visual model will be OSNet [46] because it reached the best results in the single model experiments. This helps to reduce the number of parameters for the pairwise and triplet experiments.

Additional results for the single models are provided in the hyperparameter search in the appendix, starting on page 51.

	DanceTrack		PT21	
MOTA↑	**DGS - iou** - $(0.75, 0.35)$	66.1	**DGS - oks*** - $(0.85, 0.40)$	34.1
	Mamba Track+ [15]	90.3	YOLOQ [18]	57.0
	GeneralTrack [37]	91.8	Tracktor++ [3]	63.3
	Deep-OC-SORT [30]	92.3	CorrTrack(offline) [11]	63.9
	FastTrack [27]	<u>92.8</u>	SKPFE [17]	<u>64.7</u>
HOTA↑	**DGS - oks** - $(0.75, 0.35)$	47.1	**DGS - oks** - $(0.75, 0.35)$	49.6
	Mamba Track+ [15]	56.1	GAT [10]	53.9
	GeneralTrack [37]	59.2	SKPFE [17]	<u>54.1</u>
	Deep-OC-SORT [30]	<u>61.3</u>		
	FastTrack [27]	57.4		

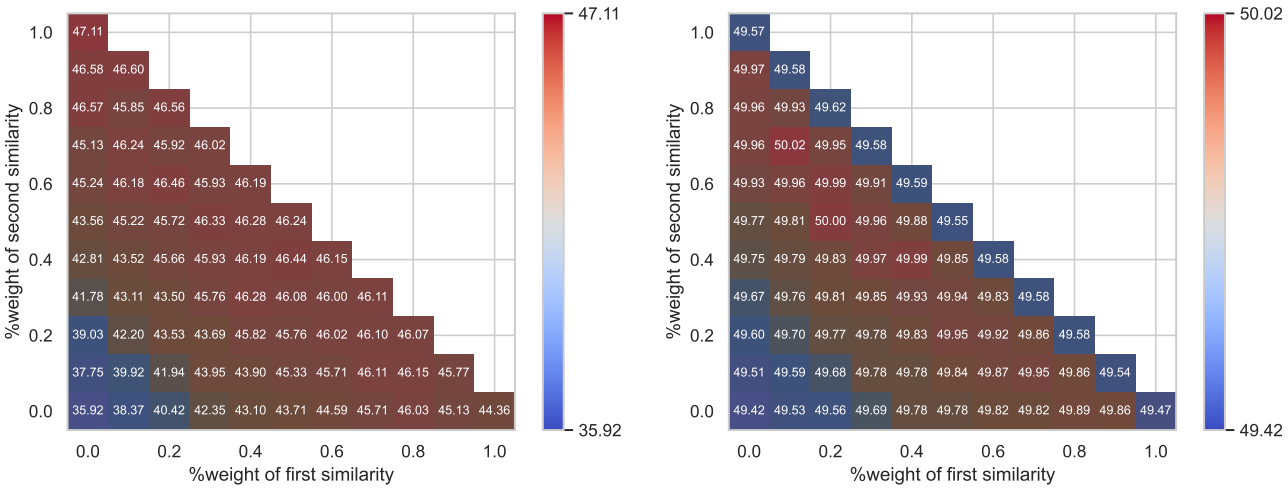
Table 4.6.: Comparison of the best static models against other publically available models on DanceTrack and PT21. The best value per experiment is <u>underlined</u>. The DGS model is marked in **bold**. Both sides of the table are sorted by the respective MOTA scores. Note that the OKS similarity function is marked with an asterisk (*) because it again uses the cMOTA score while all other models use the regular MOTA score. But because the value is still lower than that of the other baselines, it was decided, that comparing the cMOTA and MOTA scores is still valid.



(a) Results on DanceTrack.

(b) Results on PT21.

Figure 4.1.: Comparison of different static pairwise models on both datasets. On the x-axis, the percentage of the first similarity models is shown, $100 - x$ is the value of the second model. The values for 0% and 100% equal the values of the individual models. The y-axis shows the HOTA score. Note the heavily modified scale on the y-axis.



(a) Results on DanceTrack.

(b) Results on PT21.

Figure 4.2.: Comparison of the HOTA score of a single static triplet model between IoU, OKS, and OSNet [46] on both datasets. On the x-axis the percentage of the first similarity models is shown. On the y-axis the percentage of the second similarity models is shown. This results in $100 - x - y$ being the value of the third model for all positive values of x and y. The values for 0% and 100% equal the values of the individual models, the same is true for all combinations of the pairwise models, which lie on the edge of the plot. The color shows the HOTA score according to the colormap.

4.2.2. Pairwise and Triplet Models

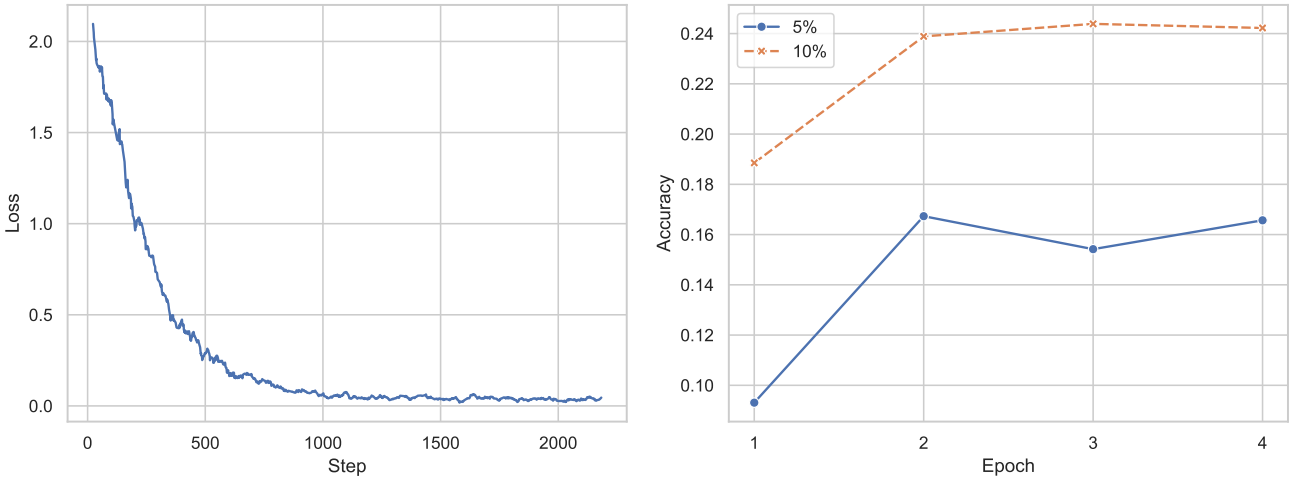
Figure 4.1 shows the results of the pairwise static models on both datasets. It is clear to see, that the graphs change for different combinations of the similarity functions. For DanceTrack, the best result is still achieved by the standalone OKS module. But it is also notable that in fig. 4.1a on the graph of ‘iou_OSNet‘ there is a maximum that is a combination of the two similarities.

On the plots for PT21 in fig. 4.1b all graphs show a slight parabolic shape. But keep in mind, that the y-axis of this plot only shows a difference in the HOTA score of 1.0. For PT21 it seems that a combination between OSNet and IoU or OKS yields the best results. The maximum peak is at 10% OSNet and 90% OKS. A second slightly lower peak is at 20% OSNet and 80% IoU.

The results of the triplet static models are shown in fig. 4.2. The plots show the weight percentage of the first and second model on the x- and y-axis respectively, while the color shows the HOTA score. This plot includes the results of the single models on the corners and the results of the pairwise models on the edge of the plot.

On DanceTrack the values span a larger range than the values on PT21, which makes it harder to spot the maximum on the plot. The best value on DanceTrack is still the standalone OKS module, even though it was expected that the combination of the three models would yield better results. This again proves that tracking on DanceTrack using visual modules is nearly impossible and yields far worse results than the IoU and OKS.

On PT21 the best results are achieved by a combination of the three models. Namely 10% IoU, 70% OKS, and 20% OSNet. But, yet again, there is no significant performance leap over the pairwise and single models.



(a) The loss during training. Lower values are better.

(b) The evaluated accuracy scores for a 5% and 10% window. Higher values are better.

Figure 4.3.: The learning curve and evaluation accuracies of one of the dynamic models. The visual OSNet model was used for this plot. The model has 5 fully connected layers with respective sizes of $[512, 256, 128, 64, 32, 1]$ and Sigmoid activation functions between each of the layers.

4.3. Comparing Constant with Dynamic Models

Let us now focus our attention on the dynamic models and whether they learned to predict the batch-wise MOTA scores successfully. The learning curves of a single dynamic model can be seen in fig. 4.3a. The learning curves of other models look similar. Some of the smaller models learned faster, while the larger models needed more epochs to learn. The models were only training for up to four epochs because they are so small. The general form of the learning curve suggests that the model learned to predict the weights, because the loss is steadily decreasing over time.

The MOTA and HOTA scores cannot be computed online, because they depend on the whole dataset. To obtaining the accuracy faster during evaluation, a custom batch-based accuracy function was used for the dynamic models. This custom accuracy function provides insight on whether the predicted weights of the evaluation dataset are close to the real weights without the need to compute the dataset wide MOTA and HOTA scores. This can be done because the goal of the alpha modules is to predict the alpha-weight as close as possible to the ground truth batch-wise MOTA score. This custom accuracy uses a window size to determine how many predictions lie within a certain percentage of the ground truth value. Thus, for the 5%-accuracy, all predictions that are within 5% of the ground truth percentage value are counted as correct.

During evaluation, the dataset wide accuracies were computed for windows with a size of 1%, 5%, 10%, and 25%. Because the values for 1% and 25% are near 0 and 1 respectively, they are not shown in the plots. In fig. 4.3b the evaluation scores for a prediction accuracy with a 5%- and 10%-window are shown. The curves for both show an upwards trend for the given four epochs. This suggests that the model learned to predict the weights, because the accuracy is increasing during training. We expected the accuracies to be higher, but due to the small window sizes, perfect predictions are hard to achieve. All in all it seems that the model learned to predict the weights in account of the decreasing loss, while all the prediction accuracies have an upwards trend.

Still, the main evaluation of the dynamic models is done using the MOTA and HOTA scores. For DanceTrack the IDF1 score is also shown - like commonly done for results on this dataset.

The results for the pairwise dynamic models on DanceTrack can be found in table 4.7. All the results in the first block of results are fairly close together, showing that no matter the model the results are fairly constant. The best results of the dynamic models, highlighted with bold font, is achieved by the alpha-model that uses the kps and was trained on PT21. But even though the MOTA of the cross-trained run is close, the score of the static weights are still better than those of the dynamic models.

For PT21 the results can be found in table 4.8. The results show a similar trend as on DanceTrack. Again, the best score of the dynamic models is achieved by a cross-trained variant. The overall best model on the PT21 dataset is still a static pairwise model.

First Model	Second Model	DanceTrack — **Validation** Score Threshold: 0.75, IoU Threshold: 0.35 MOTA↑	HOTA↑	IDF1↑
iou_fc1_Sigmoid_ep4	vis_osn_fc3_2ReLUSigmoid_ep4	65.98	43.99	37.32
iou_fc2_ReLUSigmoid_ep4	vis_osn_fc3_2ReLUSigmoid_ep4	65.98	43.99	37.32
iou_fc2_ReLUSigmoid_ep4	vis_osn_fc3_3Sigmoid_ep4	66.00	43.95	37.33
iou_fc1_Sigmoid_ep4_DanceTrack	pose_coco_fc1_Sigmoid_ep4_PT21	**66.21**	**45.59**	**38.76**
Best static pairwise model		<u>66.21</u>	<u>47.10</u>	<u>39.86</u>

Table 4.7.: The results of the dynamic models on the DanceTrack validation dataset. The best value of the dynamic experiments is **bold**, the overall best value is <u>underlined</u>. The `KeypointRCNN` backbone was used for the experiments. Two models were used for the evaluation, the names indicate the type of similarity model, the number of fully connected layers, the activation functions, and the number of epochs the model was trained. The middle part of the table shows the model evaluated on DanceTrack with the key-point model trained on PT21.

First Model	Second Model	PT21 — **Validation** Score Threshold: 0.85, IoU Threshold: 0.40 MOTA↑	HOTA↑
pose_coco_fc1_Sigmoid_ep4	vis_osn_fc5_4ReLUSigmoid_ep4	34.14	49.83
iou_fc1_Sigmoid_ep4	vis_osn_fc3_2ReLUSigmoid_ep4	34.09	49.80
iou_fc1_Sigmoid_ep4	pose_coco_fc1_Sigmoid_ep4	**34.17**	49.74
iou_fc1_Sigmoid_ep4_Dance	vis_osn_fc3_2ReLUSigmoid_ep4_Dance	34.12	**49.84**
Best static pairwise model		<u>34.23</u>	<u>49.97</u>

Table 4.8.: The results of the dynamic models on the PT21 validation dataset. The best value of the dynamic experiments is **bold**, the overall best value is <u>underlined</u>. The `KeypointRCNN` backbone was used for the experiments. Two models were used for the evaluation, the names indicate the type of similarity model, the number of fully connected layers, the activation functions, and the number of epochs the model was trained. The middle part of the table shows the model evaluated on PT21 with the IoU and visual model trained on DanceTrack.

These results of the pairwise dynamic models are not as good as expected. Seeing as the dynamic models should be able to adapt on a frame-wise level to the current situation, while the static models use fixed weights over the whole dataset. Additionally, even though we expected the cross-trained models to perform well overall, we did not expect them to beat all the models trained on the same dataset. This begs to reason, whether the models were not able to learn the weights correctly.

First Model	Second Model	Third Model	DanceTrack — **Validation** Score Threshold: 0.75, IoU Threshold: 0.35 MOTA↑	HOTA↑	IDF1↑
iou_fc2_2Sigmoid_ep2	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_4ReLUSigmoid_ep4	66.10	**46.47**	**40.14**
iou_fc2_2Sigmoid_ep4	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_5Sigmoid_ep4	**66.11**	46.17	40.03
iou_fc2_2Sigmoid_ep4	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc3_2ReLUSigmoid_ep4	66.10	46.16	39.98
iou_fc2_2Sigmoid_ep4	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_4ReLUSigmoid_ep4	66.10	46.07	39.89
iou_fc2_2Sigmoid_ep4_pt21	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_5Sigmoid_ep4_Dance	66.09	46.22	40.02
iou_fc2_2Sigmoid_ep4_pt21	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_5Sigmoid_ep4_pt21	66.08	46.28	40.11
Best static triplet model			66.21	47.11	39.94

Table 4.9.: The results of the dynamic models on the DanceTrack validation dataset. The best value of the dynamic experiments is **bold**, the overall best value is underlined. The `KeypointRCNN` backbone was used for the experiments. Three models were used for the evaluation, the names indicate the type of similarity model, the number of fully connected layers, the activation functions, and the number of epochs the model was trained. The middle part of the table shows the model evaluated on DanceTrack with the key-point model and at least one additional model trained on PT21.

But let us look at the dynamic triplet models before we jump to conclusions. The results of the dynamic triplet models can be found in table 4.9 for DanceTrack and in table 4.10 for PT21. In both datasets, the best model was a model trained on the current dataset, while the cross-trained results are slightly worse overall. For DanceTrack the best model in terms of MOTA and HOTA is still the static OKS model. One notable difference, is that most of the dynamic models beat the static OKS model in terms of the IDF1 score. Seeing as the IDF1 score weighs the tracking accuracy more than the detection accuracy, this could be a hint that the dynamic models are slightly better at tracking than the static models. But because the improvement is so small, it is hard to say whether that really is the case.

All in all it seems, that the dynamic models are not able to beat the static models. This might be due to the fact that the models were not able to learn the weights correctly — even though the learning curves in fig. 4.3 suggest otherwise. Another reason might be that the MOTA score predicted by the alpha-modules is trained on the ground-truth data, while the evaluation is on the data predicted by `KeypointRCNN`. This might lead to a discrepancy between the predicted and the real MOTA score, which in turn leads to worse results. Additionally, because the training is done on the ground-truth data, there is barely any noise in the training data, which might lead to the models overfitting the data. Especially, because the models are so small. Therefore, and due to the fact that the training data is so clean, the models might not be able to generalize well enough to perform satisfactorily on the evaluation data.

First Model	Second Model	Third Model	PT21 — **Validation** Score Threshold: 0.85, IoU Threshold: 0.40 MOTA↑	HOTA↑
iou_fc1_Sigmoid_ep4	pose_coco_fc1_Sigmoid_ep4	vis_osn_fc5_4ReLUSigmoid_ep4	**34.22**	**49.97**
iou_fc1_Sigmoid_ep2	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc3_2ReLUSigmoid_ep4	34.19	49.91
iou_fc2_2Sigmoid_ep2	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_5Sigmoid_ep4	34.20	49.91
iou_fc2_2Sigmoid_ep2	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_4ReLUSigmoid_ep4	34.19	49.91
iou_fc1_Sigmoid_ep4_Dance	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc3_3Sigmoid_ep4_Dance	34.21	49.93
iou_fc2_2Sigmoid_ep4_Dance	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_5Sigmoid_ep4_Dance	34.20	49.93
iou_fc2_2Sigmoid_ep4_Dance	pose_coco_fc2_2Sigmoid_ep4_pt21	vis_osn_fc5_5Sigmoid_ep4_pt21	34.20	49.91
Best static triplet model			34.25	50.02

Table 4.10.: The results of the dynamic models on the PT21 validation dataset. The best value of the dynamic experiments is **bold**, the overall best value is underlined. The `KeypointRCNN` backbone was used for the experiments. Three models were used for the evaluation, the names indicate the type of similarity model, the number of fully connected layers, the activation functions, and the number of epochs the model was trained. The middle part of the table shows the model evaluated on PT21 with at least one or both of the IoU or visual models trained on DanceTrack.

The results of DanceTrack on the test set can be found in table 4.11. The overall values of DGS are as expected and are significantly lower than those of the current state-of-the-art models. This is most likely due to the simplicity of the used similarity modules. It is expected that the overall performance increases drastically by including some methods of time-warping the poses and bboxes to the current time-frame. The best model on the test set in regard to the HOTA is the static model with the OKS similarity function. The optimal model in terms of MOTA and IDF1 is the dynamic model with two alpha-modules for IoU and OKS. The OKS model for the dynamic run was once again the model cross-trained on PT21.

A comparison of the values of the association- and the detection accuracy with those of the state-of-the-art models reveals that the DGS models are not able to compete. Furthermore, the DGS modules appears to exhibit lower AssA performance than other models, particularly Deep-OC-SORT [30]. When comparing the detection accuracy of FastTrack [27] (81.1) to that of the best DGS model (60.13), the question arises, whether the backbone model of FastTrack, which is YOLOX [12]-based, might be superior to the utilized `KeypointRCNN` backbone.

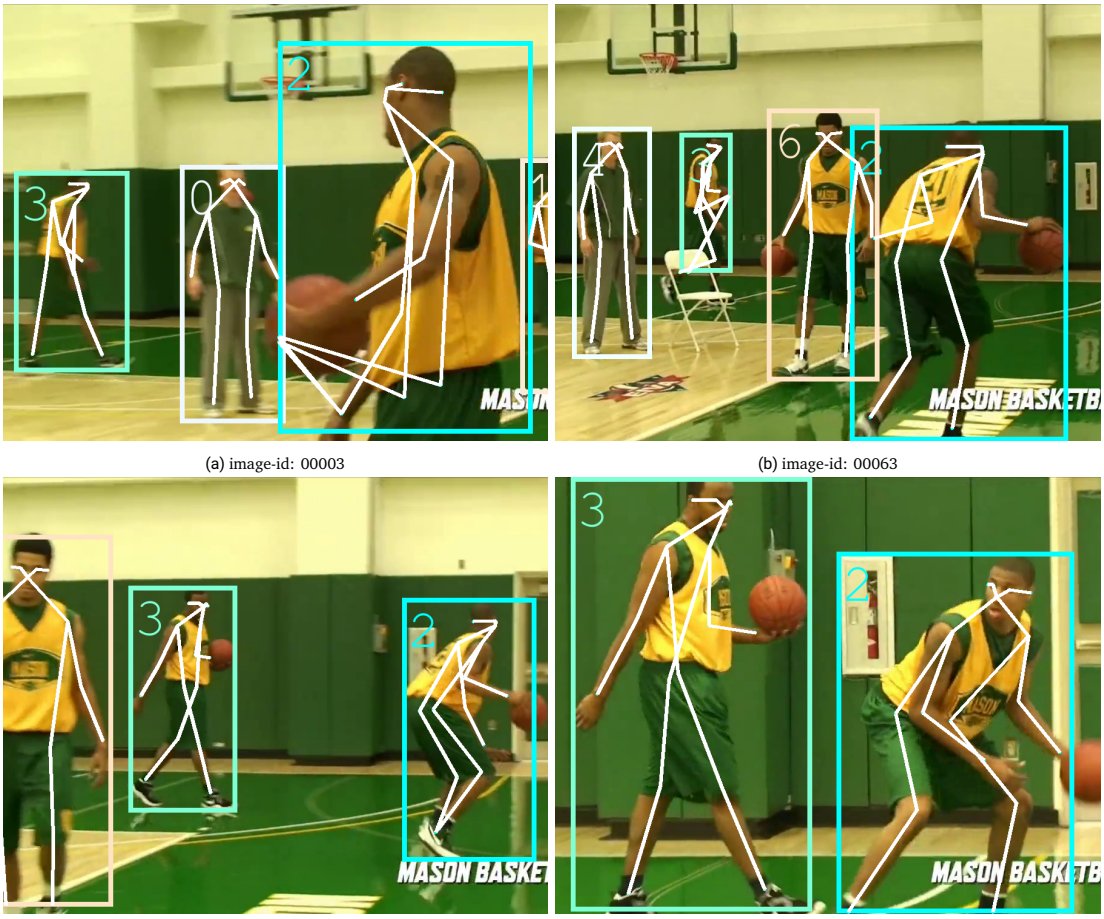
Because the evaluation server for the PT21 dataset has not been available for multiple years at the time of writing, the results for the DGS models on the PT21 test set are not available.

Model	DanceTrack — **Test** Score Threshold: 0.75, IoU Threshold: 0.35 MOTA↑	HOTA↑	IDF1↑	AssA↑	DetA↑
DGS — static — single — iou	70.06	36.20	38.29	21.62	61.03
DGS — dynamic — triplet — iou-oks-OSNet	70.04	37.30	39.26	23.03	60.88
DGS — dynamic — double — iou-oks	**70.15**	37.61	40.00	23.30	**61.13**
DGS — static — triplet — iou-OKS-OSNet [0.2, 0.6, 0.2]	70.10	38.00	40.00	23.83	61.06
DGS — static — single — OKS	70.06	**38.42**	**40.91**	**24.36**	61.05
Mamba Track [42]	90.10	55.50	53.90	38.30	80.80
Mamba Track+ [15]	90.30	56.10	54.90	39.00	80.80
FastTrack [27]	92.80	57.40	58.20	40.70	81.10
Dynamic Adaptive Parametric Multi-Target Tracking [14]	91.10	59.80	62.10	46.10	-
Deep-OC-SORT [30]	92.30	61.30	61.50	82.20	82.20

Table 4.11.: The results of the best static and dynamic models on the DanceTrack test dataset. The best value of the DGS experiments is **bold**, the overall best value is underlined. The `KeypointRCNN` backbone was used for the experiments. The lower part of the table shows the results of other publically available models. The table is sorted by the HOTA score.

4.4. Video-Analysis

Now the best model from the dynamic experiments will be used to analyze some of the video sequences of the validation and test datasets. For the first visual analysis, we will have a look at the images taken from PT21, visible in fig. 4.4. The scene shows a training sequence of basketball players and a coach. The main focus of this analysis is the player with ID 3 on the left side of fig. 4.4a in the far background. In the roughly two seconds between this and the next image (fig. 4.4b), the player crossed behind the coach (ID 0, later ID 4). During this time, the player with ID 3 was not visible for nine frames. Due to the camera movement and the player's movement the IoU and OKS similarities could not match that player successfully, but the visual similarity was able to track that player successfully. After passing the coach in fig. 4.4b, the player kept its ID and is therefore successfully tracked. The player continues to move, and walks behind the player with ID 6 in fig. 4.4c, again while bein tracked successfully. This time the player was not detected by the backbone model for 14 frames. In the last image (fig. 4.4d) the player is visible again and is tracked successfully by all three similarity functions. This image shows the precision of the key-point prediction by the backbone model. Because, even though some of the points are not even visible, they are annotated correctly. Examples might be the left elbow of the player with ID 3 or the left side of the hip of both players.



(a) image-id: 00003

(b) image-id: 00063

(c) image-id: 00110

(d) image-id: 00141

Figure 4.4.: The images are taken from the video “012968_mpii_test“ in the validation set of PT21. The respective image-id is shown under each of the images. All images are slighly cropped to enlarge the relevant detections. The video shows four seconds of a training sequence of Basketball players and a coach. Throughout the video, the camera is following the player with the number 20 on its back - the player with the ID 2. The bounding-boxes and the COCO key-points predicted by `KeypointRCNN` are shown along the predicted person ID using the best dynamic DGS-model.

(a) image-id: 00000

(b) image-id: 00164

(c) image-id: 00298

(d) image-id: 00512

Figure 4.5.: The images are taken from the video “dancetrack0017“ in the test set of DanceTrack. The respective image-id is shown under each of the images. All images are slighly cropped to enlarge the relevant detections. The full video shows roughly 30 seconds of a dance routine, where the dancers start and end as a close group, while they are spread apart in between. The camera is static throughout the whole video. The bboxes and the COCO kps predicted by `KeypointRCNN` are shown along the predicted person ID using the best dynamic DGS-model.

The second analysis is on the test data from DanceTrack and can be seen in fig. 4.5. The group of dancers starts out as a close group in the first image (fig. 4.5a). The detection model did not detect all the dancers in the first frame, as you can see by looking at the person in the striped shirt with red hair. As long as the dancers are this close together, the backbone detection model has a hard time detecting all the dancers. This inconsistency is also displayed in the tracking behaviour of the dynamic model. The model tracks and recognizes the dancers most of the time, but there are regular jumps in the tracking.

This is better during the second part of the video, where the dancers are spread apart further (see fig. 4.5b). The model is able to track the dancers more consistently, even though the dancers still switch places in pairs. Such a jump can be seen between figure 4.5b and figure 4.5c. During the middle part of the video, the tracker worked nearly perfect. Unfortunately, the last part of the video, where the dancers regroup, shows the same problems as the first part. This can be seen in the last image (fig. 4.5d), where the IDs of the dancers switched multiple times when the dancers got close again.

5. Conclusion

The models with constant weights outperformed expectations on the challenging DanceTrack dataset, and roughly as expected on the PT21 dataset. The DanceTrack dataset proved its claim that models relying solely on visual similarity achieve suboptimal performance. However, it was demonstrated that the constant pairwise model between IoU and the visual similarity had increased performance using parts of the visual information. The same is true on the PT21 dataset, where both similarities — IoU and OKS — had improved performance with some information of the visual similarity. The overall best constant model on DanceTrack is still the standalone model of the OKS. This is true for the single, pairwise, and triplet similarities. On the PT21 dataset, the best model is a combination of the triplet similarities, mainly focusing on the OKS similarity. In contrast to the DanceTrack dataset, the best model is a combination of models, not a standalone model. This shows that even the static weights can improve the performance of the base models.

While it seems that the dynamic weights were successfully trained, the overall performance of the dynamic weights was not superior to the models using constant weights. Some dynamic weights achieved a similar performance than the static models, but these scores were neither significant nor consistent across different models or datasets. The main reason appears to be the small model size and the absence of temporal information. Another contributing factor is the imbalance in the dataset between detections that are easy to match and those that fail to match prior to weight training. This imbalance results in weights being trained primarily on easy detections, neglecting the hard ones that are more critical. This imbalance could be fixed in multiple ways. Larger or deeper models theoretically could learn more complex patterns and thus may be better equipped to handle the imbalanced dataset. Another way would be to oversample the minority class, focussing on those detections where one or multiple similarity models fail, or employing an alternative weighting strategies to prioritize those detections in the dataset.

The goal of reducing the number of hand-crafted hyperparameters was only partially achieved. The dynamic weights can be trained and would reduce the number of hand-crafted hyperparameters, but because they are inferior to the static weights, the dynamic weights are not yet ready to replace the static ones. Additionally, the hyperparameter grid-search done to obtain filtered predictions by the backbone model is still mandatory, because there currently is no way of discarding uncertain detections. This means that the hyperparameter grid-search is still necessary to obtain the best possible results. Hence, the dynamic weights could be used to reduce the number of hyperparameters during matching, but additional dataset wide hyperparameters are still needed to obtain the best possible results.

Another goal of this thesis was to provide a highly modular and customizable pipeline. This goal was achieved, as all similarities and alpha-weight generation modules can be replaced by more complex models using values in a configuration file. The size and activation functions of each of the alpha-modules can be changed as well as the backbone model, which can be replaced by other models. It is also possible to choose between different loss functions, optimizers, and learning rate schedulers.

All in all this means that with some clever modifications of the sampling strategy, a better batch-wise MOTA score for training, and a module for time-warping the positions in tracks, the DGS algorithm could close the still large gap to other handcrafted methods. While DGS is intended as a proof of concept, and a baseline model for dynamically learned weights, it can be used in real-world applications. Those applications might be ones that do not require the best possible total performance but ones that currently struggle with a dataset where frame-wise decisions are necessary.

6. Outlook

There are still some open questions and possible improvements for the DGS algorithm. This chapter will discuss possible next steps to take with this project. Due to the modular nature of the DGS algorithm, most of these improvements should be fairly easy to implement and can even be done in different case studies.

6.1. Improving Training and Evaluation Accuracies

The main critique in chapter 5 is that the overall performance of the dynamic weights has to be improved. This can be done either by improving the training process or by increasing the things the model can learn.

6.1.1. Improving the Training Process

The biggest problem with the training process is the imbalance in the dataset between detections that are easy to match and those that fail to match prior to weight training. This can be solved as mentioned in the conclusion by employing alternative weighting strategies to prioritize hard-to-match detections in the dataset. Another way would be to oversample the minority class.

6.1.2. Changing the Training Accuracy

Another crucial part of DGS is the accuracy of the training process. Currently, the training accuracy is calculated as the percentage of correct predictions, but with the imbalanced dataset, this is most likely another point of failure. Thus, the training accuracy should be changed to a more robust metric. Maybe the accuracy can use weighted classes, where

the weights are the inverse of the class frequency. Another thing to keep in mind is that the accuracy should actually be mimicking the MOTA and HOTA metrics of the current batch. Custom batch-based HOTA accuracies could increase the model performance after training.

Another way could include adding noise to the training process to make the model more robust to noisy detections. Because right now the model is only trained on the ground truth detections. But adding some faulty detections or marking certain detections as missing would result in a greater variance in the batch-based MOTA score.

6.1.3. Increasing the Model Size

It is expected that larger or deeper models could learn more complex patterns and thus may be better equipped to handle the imbalanced dataset. But due to the small input sizes, there is no easy way to increase the model size. One option would be to input the bboxes, kps, and the visual embeddings into each of the models to simply increase the number of parameters in the models. These larger models have to be trained for longer to ensure convergence.

6.2. Temporal Information

As discussed in chapter 5 the temporal information could be a crucial next step for a better performance of the DGS algorithm.

6.2.1. Time-warping and better temporal matching

As mentioned before, it would be better to first time-warp the detections into the next frame before comparing them, like most other papers do. This should increase the overall performance of DGS drastically and be the next step to beating the current state-of-the-art algorithms. This step should be fairly straight-forward to implement, as the modularity of the DGS algorithm allows for easy integration of new modules.

6.2.2. Track-Memory and Re-Identifying Previously Seen People

Even though there currently is the possibility to use a track-memory, there is no module implemented that uses this information. From a history of tracks, a better time-warping module could be created using the information of the track-memory to predict better key point-positions for the next frame. A module like this could be trained in an end-to-end manner and in combination with the current alpha modules.

6.3. Improving Matching

The other open issue is the overall matching of detections to tracks. Having to match every detection to a track is not ideal because the algorithm cannot decide to ignore uncertain detections. This leads to some false negatives and or false positives in the tracking process. Hence, adding a possibility to remove uncertain detections from the tracking process would be a crucial step to improve the overall tracking performance. And all that without adding additional false positives from lowering the detection thresholds.

Ergo, having to fix the hyperparameters for the detection thresholds is not ideal and should be improved (section 6.3.1). Additionally, the initial weight is currently fixed and could possibly be learned during the training process (section 6.3.2).

6.3.1. Remove Uncertain Detections

Currently, there is no way to remove uncertain detections from the tracking process once they have been detected. This could be a crucial step to improve the overall tracking performance. This would also remove the necessity for the hyperparameter search for the detection thresholds. The easiest way would be to create a score or detection threshold and remove all detections below this threshold.

6.3.2. Dynamic Initial Weight

To remove one of the last fixed hyperparameters of the DGS module, the initial weight should not be fixed. Instead, the initial weight should be learned during the training process, similar to the training of the alpha modules. This would allow the algorithm

to adapt to the respective dataset and the current detections. For example, when the detections are noisy, the initial weight could be higher to prevent the algorithm from tracking false positives.

Bibliography

[1] M. Andriluka et al. “PoseTrack: A Benchmark for Human Pose Estimation and Tracking”. In: *CVPR*. 2018.

[2] Valentin Bazarevsky et al. *BlazePose: On-device Real-time Body Pose tracking*. 2020. arXiv: 2006.10204 [cs.CV]. URL: https://arxiv.org/abs/2006.10204.

[3] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. “Tracking Without Bells and Whistles”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00103. URL: http://dx.doi.org/10.1109/ICCV.2019.00103.

[4] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. “Multiple object tracking performance metrics and evaluation in a smart room environment”. In: *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*. Vol. 90. 91. Citeseer. 2006.

[5] Alex Bewley et al. “Simple online and realtime tracking”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2016. DOI: 10.1109/icip.2016.7533003. URL: http://dx.doi.org/10.1109/ICIP.2016.7533003.

[6] Jinkun Cao et al. “Instance-Aware Predictive Navigation in Multi-Agent Environments”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 5096–5102. DOI: 10.1109/ICRA48506.2021.9561235.

[7] Jinkun Cao et al. *Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking*. 2023. arXiv: 2203.14360 [cs.CV]. URL: https://arxiv.org/abs/2203.14360.

[8] Ho Kei Cheng and Alexander G. Schwing. “XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 640–658. ISBN: 978-3-031-19815-1.

[9] Arjun S. Dileep et al. “Suspicious Human Activity Recognition using 2D Pose Estimation and Convolutional Neural Network”. In: *2022 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*. 2022, pp. 19–23. DOI: 10.1109/WiSPNET54241.2022.9767152.

[10] Andreas Doering and Juergen Gall. *A Gated Attention Transformer for Multi-Person Pose Tracking*. 2023. arXiv: 2306.05807 [cs.CV]. URL: https://arxiv.org/abs/2306.05807.

[11] Andreas Doering et al. “PoseTrack21: A Dataset for Person Search, Multi-Object Tracking and Multi-Person Pose Tracking”. In: *CVPR*. 2022.

[12] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 [cs.CV]. URL: https://arxiv.org/abs/2107.08430.

[13] Kaiming He et al. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV]. URL: https://arxiv.org/abs/1703.06870.

[14] Jiale Hu et al. “Dynamic Adaptive Parametric Multi-Target Tracking Algorithm Based on Appearance and Pose Feature”. In: *2023 8th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*. 2023, pp. 107–111. DOI: 10.1109/IC-NIDC59918.2023.10390705.

[15] Hsiang-Wei Huang et al. *Exploring Learning-based Motion Models in Multi-Object Tracking*. 2024. arXiv: 2403.10826 [cs.CV]. URL: https://arxiv.org/abs/2403.10826.

[16] Christian Keilstrup Ingwersen et al. “SportsPose - A Dynamic 3D Sports Pose Dataset”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 5219–5228.

[17] Yalong Jiang et al. “Multi-Person Pose Tracking With Sparse Key-Point Flow Estimation and Hierarchical Graph Distance Minimization”. In: *IEEE Transactions on Image Processing* 33 (2024), pp. 3590–3605. DOI: 10.1109/TIP.2024.3405339.

[18] Sheng Jin et al. “You Only Learn One Query: Learning Unified Human Query for Single-Stage Multi-person Multi-task Human-Centric Perception”. In: *Computer Vision – ECCV 2024*. Ed. by Aleš Leonardis et al. Cham: Springer Nature Switzerland, 2025, pp. 126–146. ISBN: 978-3-031-72649-1.

[19] Hye-Young Jo et al. “Flowar: How different augmented reality visualizations of online fitness videos support flow for at-home yoga exercises”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–17.

[20] Arne Hoffhues Jonathon Luiten. *TrackEval*. 2020. URL: https://github.com/JonathonLuiten/TrackEval.

[21] Hyeonchul Jung et al. “ConfTrack: Kalman Filter-Based Multi-Person Tracking by Utilizing Confidence Score of Detection Box”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 6583–6592.

[22] Jeongho Kim et al. “Addressing the Occlusion Problem in Multi-Camera People Tracking With Human Pose Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 5463–5469.

[23] Harold W Kuhn. “The Hungarian method for the assignment problem”. In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[24] Van-Hung Le. “Deep learning-based for human segmentation and tracking, 3D human pose estimation and action recognition on monocular video of MADS dataset”. In: *Multimedia Tools and Applications* 82.14 (June 2023), pp. 20771–20818. ISSN: 1573-7721. DOI: 10.1007/s11042-022-13921-w. URL: https://doi.org/10.1007/s11042-022-13921-w.

[25] Florin Leon and Marius Gavrilescu. “A Review of Tracking and Trajectory Prediction Methods for Autonomous Driving”. In: *Mathematics* 9.6 (Mar. 2021), p. 660. ISSN: 2227-7390. DOI: 10.3390/math9060660. URL: http://dx.doi.org/10.3390/math9060660.

[26] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[27] Chongwei Liu, Haojie Li, and Zhihui Wang. “FastTrack: A Highly Efficient and Generic GPU-Based Multi-object Tracking Method with Parallel Kalman Filter”. In: *International Journal of Computer Vision* 132.5 (May 2024), pp. 1463–1483. ISSN: 1573-1405. DOI: 10.1007/s11263-023-01933-4. URL: https://doi.org/10.1007/s11263-023-01933-4.

[28] Jonathon Luiten et al. “HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking”. In: *International Journal of Computer Vision* (2020), pp. 1–31.

[29] Jonathon Luiten et al. “Hota: A higher order metric for evaluating multi-object tracking”. In: *International journal of computer vision* 129 (2021), pp. 548–578.

[30] Gerard Maggiolino et al. *Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification*. 2023. arXiv: 2302.11813 [cs.CV]. URL: https://arxiv.org/abs/2302.11813.

[31] Gyeongsik Moon et al. “InterHand2.6M: A Dataset and Baseline for 3D Interacting Hand Pose Estimation from a Single RGB Image”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[32] Amir Nadeem, Ahmad Jalal, and Kibum Kim. “Automatic human posture estimation for sport activity recognition with robust body parts detection and entropy markov model”. In: *Multimedia Tools and Applications* 80.14 (June 2021), pp. 21465–21498. ISSN: 1573-7721. DOI: 10.1007/s11042-021-10687-5. URL: https://doi.org/10.1007/s11042-021-10687-5.

[33] Van-Truong Nguyen et al. “An improvement of the camshift human tracking algorithm based on deep learning and the Kalman filter”. In: *Journal of Robotics* 2023.1 (2023), p. 5525744.

[34] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: (2017).

[35] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[36] Marei Peischl. *TUDaPub – LaTeX-Paper im Corporate Design der TU Darmstadt*. Oct. 12, 2021. URL: http://mirrors.ctan.org/macros/latex/contrib/tuda-ci/doc/DEMO-TUDaPub.pdf (visited on 10/12/2021).

[37] Zheng Qin et al. “Towards Generalizable Multi-Object Tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2024, pp. 18995–19004.

[38] Esraa Samkari et al. “Human Pose Estimation Using Deep Learning: A Systematic Literature Review”. In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1612–1659. ISSN: 2504-4990. DOI: 10.3390/make5040081. URL: https://www.mdpi.com/2504-4990/5/4/81.

[39] Peize Sun et al. “DanceTrack: Multi-Object Tracking in Uniform Appearance and Diverse Motion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[40] J. Usha Rani and P. Raviraj. “Real-Time Human Detection for Intelligent Video Surveillance: An Empirical Research and In-depth Review of its Applications”. In: *SN Computer Science* 4.3 (Mar. 2023), p. 258. ISSN: 2661-8907. DOI: 10.1007/s42979-022-01654-4. URL: https://doi.org/10.1007/s42979-022-01654-4.

[41] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[42] Changcheng Xiao et al. *MambaTrack: A Simple Baseline for Multiple Object Tracking with State Space Model*. 2024. arXiv: 2408.09178 [cs.CV]. URL: https://arxiv.org/abs/2408.09178.

[43] Yuang Zhang, Tiancai Wang, and Xiangyu Zhang. “MOTRv2: Bootstrapping End-to-End Multi-Object Tracking by Pretrained Object Detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 22056–22065.

[44] Kaiyang Zhou and Tao Xiang. “Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch”. In: *arXiv preprint arXiv:1910.10093* (2019).

[45] Kaiyang Zhou et al. “Learning Generalisable Omni-Scale Representations for Person Re-Identification”. In: *TPAMI* (2021).

[46] Kaiyang Zhou et al. “Omni-Scale Feature Learning for Person Re-Identification”. In: *ICCV*. 2019.

[47] Lijuan Zhou et al. *Human Pose-based Estimation, Tracking and Action Recognition with Deep Learning: A Survey*. 2023. arXiv: 2310.13039 [cs.CV]. URL: https://arxiv.org/abs/2310.13039.

[48] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. *Tracking Objects as Points*. 2020. arXiv: 2004.01177 [cs.CV]. URL: https://arxiv.org/abs/2004.01177.

A. Appendix

A.1. PoseTrack21 — Hyperparameter Gridsearch

Three standalone similarity models are run on the PT21 dataset. The results can be seen in table A.1 for the IoU, in table A.2 for the OKS, and in table A.3 for the OSNet [46] model. The tables show the cMOTA and HOTA for every pair of score- and IoU-thresholds. Better results are marked with a green background color and the best result of each table is underlined. In addition to the two analyzed thresholds, a fixed minimum side-length of 50.0 pixels is enforced for all detected bboxes.

Because there is no clear overall winner, the hyperparameters are chosen to roughly maximize all three similarity models for cMOTA as well as HOTA. The IoU-threshold will be set to 0.40 because for all score thresholds every maximum value of the cMOTA and HOTA is contained. The score threshold is will be to 0.85 because there should rather be more false positives than false negatives. This is because there was a plan to add a minimum weight to further filter the similarity scores before matching, but this was not implemented in time, as mentioned in section 6.3.1.

To find out whether a different image size does change the results, the same hyperparameter gridsearch is done with a smaller image size of 256 by 128 pixels. The results of the IoU and OKS are not shown because they are, as expected, nearly identical to the results of the 256 by 192 pixel image size. The results for the visual OSNet model in table A.4 show that the cMOTA and HOTA are slightly worse compared to the values from the larger images. The maximum value is roughly at the same score threshold of 0.90 and the same IoU threshold of 0.40.

IoU		IoU-Threshold							
Score-Threshold		0.20	0.30	0.35	0.40	0.45	0.50	0.60	0.70
cMOTA↑	0.75	30.76	31.27	30.81	29.96	27.88	23.91	23.91	23.91
	0.80	31.95	32.81	32.52	31.95	30.37	27.48	27.48	27.48
	0.85	33.21	34.48	34.37	34.05	33.05	31.15	31.15	31.15
	0.90	34.87	36.42	36.54	36.45	35.91	34.82	34.82	34.82
	0.95	36.84	38.60	38.96	39.08	38.92	38.61	38.61	38.61
	0.99	35.90	37.41	37.67	37.87	37.94	37.98	37.98	37.98
HOTA↑	0.75	33.06	34.35	34.53	34.54	34.09	33.35	33.35	33.35
	0.80	33.13	34.47	34.63	34.66	34.34	33.80	33.80	33.80
	0.85	33.12	34.51	34.73	34.77	34.60	34.15	34.15	34.15
	0.90	33.02	34.41	34.67	34.74	34.71	34.44	34.44	34.44
	0.95	31.65	32.96	33.22	33.29	33.35	33.36	33.36	33.36
	0.99	20.18	20.93	21.05	21.12	21.18	21.21	21.21	21.21

Table A.1.: Hyperparameter search on the PT21 dataset with a resolution of 256 by 192 for the image crop. The similarity module is: IoU. Note that the IoU increments are not in constant 0.1 increments. The best value for each metric is underlined.

OKS		IoU-Threshold							
Score-Threshold		0.20	0.30	0.35	0.40	0.45	0.50	0.60	0.70
cMOTA↑	0.75	30.77	31.27	30.83	29.94	27.84	23.94	23.94	23.94
	0.80	31.95	32.82	32.53	31.94	30.33	27.49	27.49	27.49
	0.85	33.21	34.49	34.38	34.07	33.00	31.16	31.16	31.16
	0.90	34.92	36.42	36.57	36.47	35.89	34.84	34.84	34.84
	0.95	36.83	38.58	38.92	39.04	38.88	38.58	38.58	38.58
	0.99	35.88	37.39	37.64	37.84	37.91	37.96	37.96	37.96
HOTA↑	0.75	33.23	34.42	34.62	34.55	34.05	33.41	33.41	33.41
	0.80	33.26	34.52	34.69	34.70	34.33	33.80	33.80	33.80
	0.85	33.25	34.57	34.77	34.83	34.53	34.17	34.17	34.17
	0.90	33.22	34.49	34.72	34.86	34.69	34.47	34.47	34.47
	0.95	31.79	32.94	33.21	33.29	33.35	33.32	33.32	33.32
	0.99	20.21	20.94	21.07	21.14	21.20	21.22	21.22	21.22

Table A.2.: Hyperparameter search on the PT21 dataset with a resolution of 256 by 192 for the image crop. The similarity module is: OKS. Note that the IoU increments are not in constant 0.1 increments. The best value for each metric is underlined.

	OSNet	IoU-Threshold							
	Score-Threshold	0.20	0.30	0.35	0.40	0.45	0.50	0.60	0.70
cMOTA↑	0.75	30.41	30.95	30.50	29.62	27.53	23.63	23.63	23.63
	0.80	31.61	32.50	32.24	31.61	30.05	27.24	27.24	27.24
	0.85	32.97	34.21	34.16	33.81	32.75	30.93	30.93	30.93
	0.90	34.72	36.23	36.39	36.26	35.71	34.61	34.61	34.61
	0.95	36.69	38.43	38.82	38.93	38.78	38.46	38.46	38.46
	0.99	35.70	37.28	37.57	37.77	37.84	37.88	37.88	37.88
HOTA↑	0.75	33.05	34.28	34.53	34.54	34.13	33.47	33.47	33.47
	0.80	33.12	34.41	34.64	34.64	34.34	33.92	33.92	33.92
	0.85	33.16	34.48	34.72	34.77	34.49	34.23	34.23	34.23
	0.90	33.10	34.33	34.70	34.78	34.66	34.46	34.46	34.46
	0.95	31.66	32.94	33.23	33.34	33.37	33.35	33.35	33.35
	0.99	20.12	20.90	21.06	21.13	21.20	21.24	21.24	21.24

Table A.3.: Hyperparameter search on the PT21 dataset with a resolution of 256 by 192 for the image crop. The similarity module is: OSNet [46]. Note that the IoU increments are not in constant 0.1 increments. The best value for each metric is underlined.

	OSNet	IoU-Threshold			
	Score-Threshold	0.30	0.40	0.50	0.60
cMOTA↑	0.80	32.44	31.53	27.06	27.06
	0.85	34.15	33.73	30.78	30.78
	0.90	36.13	36.12	34.43	34.43
	0.95	38.33	38.83	38.34	38.34
	0.99	37.24	37.70	37.81	37.81
HOTA↑	0.80	34.36	34.55	33.65	33.65
	0.85	34.33	34.63	34.02	34.02
	0.90	34.23	34.62	34.28	34.28
	0.95	32.79	33.27	33.30	33.30
	0.99	20.84	21.05	21.13	21.13

Table A.4.: Hyperparameter search on the PT21 dataset with a resolution of 256 by **128** for the image crop. The similarity module is: OSNet [46]. The best value for each metric is underlined.

IoU		IoU-Threshold							
	Score-Threshold	0.20	0.30	0.35	0.40	0.45	0.50	0.60	0.70
MOTA↑	0.70	60.41	64.69	65.59	66.03	65.87	64.70	64.70	64.70
	0.75	60.83	65.21	66.15	66.67	66.72	66.03	66.03	66.03
	0.80	61.22	65.61	66.59	67.21	67.42	67.12	67.12	67.12
	0.85	61.58	65.93	66.91	67.54	67.88	67.84	67.84	67.84
	0.90	61.66	65.93	66.85	67.46	67.83	68.01	68.01	68.01
	0.95	60.34	64.17	64.93	65.38	65.69	65.91	65.91	65.91
	0.99	50.23	52.23	52.54	52.67	52.75	52.80	52.80	52.80
HOTA↑	0.70	39.17	43.94	44.71	44.48	42.74	40.12	40.12	40.12
	0.75	38.64	43.74	44.36	44.50	43.33	41.34	41.34	41.34
	0.80	38.61	42.68	43.72	44.14	42.77	41.14	41.14	41.14
	0.85	38.79	43.03	43.05	43.74	43.76	43.17	43.17	43.17
	0.90	38.35	42.13	42.51	42.27	43.50	42.61	42.61	42.61
	0.95	37.24	40.56	41.29	41.87	41.79	41.42	41.42	41.42
	0.99	30.26	32.64	32.93	33.01	32.91	32.94	32.94	32.94

Table A.5.: Hyperparameter search on the DanceTrack dataset with a resolution of 256 by 192 for the image crop. The similarity module is: IoU. Note that the IoU increments are not in constant 0.1 increments. The best value for each metric is underlined.

OKS		IoU-Threshold							
	Score-Threshold	0.20	0.30	0.35	0.40	0.45	0.50	0.60	0.70
MOTA↑	0.70	60.41	64.65	65.52	65.96	65.79	64.48	64.48	64.48
	0.75	60.83	65.17	66.10	66.64	66.68	65.88	65.88	65.88
	0.80	61.21	65.59	66.55	67.15	67.37	67.01	67.01	67.01
	0.85	61.57	65.91	66.89	67.54	67.83	67.80	67.80	67.80
	0.90	61.67	65.95	66.86	67.48	67.85	68.03	68.03	68.03
	0.95	60.36	64.19	64.96	65.41	65.71	65.93	65.93	65.93
	0.99	50.28	52.27	52.57	52.72	52.81	52.85	52.85	52.85
HOTA↑	0.70	39.98	45.58	46.19	45.92	42.63	40.04	40.04	40.04
	0.75	39.70	46.25	47.11	46.90	44.73	41.52	41.52	41.52
	0.80	39.38	45.98	46.89	46.16	44.21	41.49	41.49	41.49
	0.85	39.65	46.34	46.10	45.68	44.72	43.39	43.39	43.39
	0.90	38.81	45.18	45.50	44.76	45.76	43.92	43.92	43.92
	0.95	37.13	43.42	43.76	44.59	43.92	43.30	43.30	43.30
	0.99	31.36	33.61	34.35	34.39	34.14	34.32	34.32	34.32

Table A.6.: Hyperparameter search on the DanceTrack dataset with a resolution of 256 by 192 for the image crop. The similarity module is: OKS. Note that the IoU increments are not in constant 0.1 increments. The best value for each metric is underlined.

	OSNet	IoU-Threshold							
	Score-Threshold	0.20	0.30	0.35	0.40	0.45	0.50	0.60	0.70
MOTA↑	0.70	59.11	63.44	64.28	64.74	64.55	63.16	63.16	63.16
	0.75	59.59	63.96	64.89	65.42	65.45	64.60	64.57	64.57
	0.80	59.98	64.40	65.36	65.97	66.19	65.81	65.77	65.77
	0.85	60.30	64.76	65.72	66.37	66.70	66.62	66.61	66.61
	0.90	60.41	64.80	65.73	66.32	66.73	66.86	66.86	66.86
	0.95	59.18	63.09	63.85	64.32	64.60	64.81	64.81	64.81
	0.99	49.27	51.33	51.64	51.79	51.87	51.91	51.91	51.91
HOTA↑	0.70	31.56	34.37	35.74	36.03	35.12	33.85	33.85	33.85
	0.75	32.20	34.47	35.92	35.70	35.25	34.89	34.13	34.13
	0.80	31.43	34.89	35.72	35.66	35.57	35.05	35.17	35.17
	0.85	31.59	35.20	36.33	35.84	36.39	35.77	35.71	35.71
	0.90	31.05	34.73	35.66	35.55	36.41	35.83	35.88	35.88
	0.95	30.96	33.37	34.90	34.93	35.24	36.41	36.41	36.41
	0.99	26.19	28.07	28.08	28.69	28.69	28.75	28.75	28.75

Table A.7.: Hyperparameter search on the DanceTrack dataset with a resolution of 256 by 192 for the image crop. The similarity module is: OSNet [46]. Note that the IoU increments are not in constant 0.1 increments. The best value for each metric is underlined.

A.2. DanceTrack — Hyperparameter Gridsearch

For DanceTrack the decision was a lot harder. Tables A.5 to A.7 show the results of the hyperparameter gridsearch on DanceTrack for the IoU, OKS, and OSNet similarity models. For all three similarity models, the maximum of the MOTA is reached at a score threshold of 0.90 and an IoU threshold of 0.50 and higher. On the other hand, does the HOTA has its maximum for the non-visual models around $(0.35, 0.70 - 0.75)$ respectively. Because the overall score of the OSNet model is lower than that of the IoU and OKS models, the decision was made to not maximize on the values of the OSNet model. Therefore, and due to the fact that the HOTA has a much more defined optima than the MOTA, the score threshold is set to 0.70 and the IoU threshold to 0.35.