Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche

PhD. Thesis XXXIII CICLO

ANNO ACCADEMICO 2019/2020

Local and Global Deep Features for Multi Object Tracking

Luca Fiori

Supervisor: Alessandro Mecocci

28/02/2021

LUCA FIORI

Signed 20/07/2021

ABSTRACT

Multiple Object Tracking (MOT) has been a central topic among various computer vision tasks, especially in recent years, with the advent of deep learning. The approaches to MOT, lately, have in fact shifted towards the ”tracking by detection” paradigm supported by a re-identification (ReID) architecture, since a robust model of the target appearance is fundamental to recover a target after long term occlusions.

Such approaches have achieved great performances on famous dataset, i.e. MOT challenge datasets, defining the state of the art, but still leaving a lot of space for improvements.

The majority of re-identification architectures consist in a deep convolutional neural network taking as input a region of interest (ROI), an image containing all the pixels belonging to the target person and as few pixel as possible belonging to what is considered background, and giving as output a ”descriptor” of the appearance of said ROI as a vector of values. A common approach in the tracking problem is to extract the ROIs using a person detector, which usually is based on another deep convolutional network such as YOLO or MaskRCNN.

A common problem arises in using ReID in the tracking pipeline, and it is evident when dealing with the partial occlusion of a target caused by another target: since the appearance of both targets is modeled from a ROI which contains inevitably foreground (target) pixels and background pixels, when two target are partially overlapping each other during the occlusion, the ROIs of the detections are intersected, which means that the obtained models will be describing a more or less great quantity of common pixels, resulting in similar descriptors for different targets.

In this work I will present an effective way to lessen this problem by flanking the ”global” descriptor from the ReID network with ”local” descriptors of parts of each target, perform the matching separately and then fusing the ”global” and ”local” match scores using a custom designed algorithm inspired by the theory of ensemble classifiers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Multiple Object Tracking

Today, especially in the more developed countries, video data is essential for many different crucial tasks, from security and surveillance to behavioural analysis for market research. The amount of data gathered by the already existing systems is huge, to the point that the processing and analysis must be automated. Many of these tasks require monitoring every single person within a crowded scene and track their movements.

The task of tracking multiple subjects in a scene on a visual system is called **Multi-Object Tracking** (**MOT**). Such field recently has become a central topic in computer vision research, due to the high demand both in the security and in the industrial sectors. One of the first attempt in visually tracking multiple targets in a scene has been done as early as 1988, by Z. Pylyshyn and R. Storm[34].

1.2 Tracking by Detection

Most modern MOT algorithms follow the *Tracking by detection* paradigm: a *detector* is applied to the whole image to find the target, and then a *tracker* performs the label association for each detected target.

Some of the most commonly used detectors are implemented as deep neural networks, such as YOLO[35] or Mask-RCNN[13] which, while being slower than YOLO, performs the segmentation (generally in form of masks of the targets) alongside the detection, thus leading to more precise target bounding boxes.

As for the tracker portion of the pipeline, many strategies have been used for tracking, from the simplest ones such as SORT[4], which relies upon Kalman filters (one for each target) without using any ego-motion information, to more complex ones that will be illustrated in Chapter 2.

SORT (Simple Online and Realtime Tracking) uses Faster Region CNN (FrRCNN)[36] for the detection part; for the tracking task the authors associate an independent Kalman Filter (KF)[18] to each target and use the prediction of the KF for estimating the position of each target (bounding box) in the next frame. The labels of the target at every frame

are associated by the *Hungarian Algorithm*[24] on a *scoretable* containing the values of the *intersection over union* (IOU) between the bounding boxes of each detected target in the current frame and the predicted bounding boxes of each target detected in the previous frames. The SORT algorithm is indeed very fast and works reasonably well in non-crowded scenarios, however, its tracking performances in presence of prolonged occlusion are very poor, thus making it unsuitable for real life application.

1.3 Re-Identification and Occlusion Handling in the Tracking Problem

The problem with algorithms such as SORT is that they rely only on the spatial information provided by the detection bounding boxes and on the predictions of the KF, which makes them unreliable in presence of long term occlusions.

During the entire period of the occlusion the target is not visible, thus the detector cannot find it, so the system must provide a prediction with missing states. Typically, the last prediction is used without updating the filter. This decreases the prediction accuracy for each contiguous frame in which the target is occluded. In practice the predicted bounding box start to drift away very quickly from the actual position of the target.

SORT idea of using IOU distance of the bounding boxes implicitly handles very short term occlusions. Tracks are terminated if they are not detected for T_{Lost} frames (which the authors set to 1). This means that SORT does not make any effort to handle long term occlusions, plus it does not perform well in case of partial occlusions that continue to recur for example when two targets walk side by side in the same direction.

To recover a target after a long term occlusion, since spatial information only is unreliable due to the cumulative error for every missing observation of the target, is necessary to have an appearance model of each target to try a re-match based on that model. Early efforts in this direction can be found in [46],[2] and [48]; the same Bewley et al. later followed these examples in DeepSORT[44], to extend the SORT paradigm by integrating the targets appearance encoded by a *convolutional neural network* (CNN).

In chapter 2 it is shown that the majority of modern tracking algorithms at the state of the art employ a model of the appearance of the target in their tracking strategies. Such a model is usually generated by some kind of deep network.

1.4 Contribution and Outline

As explained in the previous section, a robust appearance model for each target can help the system to recover the tracking of a target after a long term occlusion. However relying on such models can prove detrimental when a target is partially occluded by another target. This is due to the way the appearance information of the target is generated: the

ROI of the target, which is the area inside a rectangular bounding box (as extracted by the detector), is fed as input to a ReID neural network that outputs a descriptor of a given size. Thus, the information encoded in the descriptor comes from a rectangular image containing both foreground pixels (belonging to the target) and background pixels (surrounding the target). In the case of partial occlusion, as described above, the ROIs of two partially overlapping targets are intersected, thus they share a number of common pixels. The number of common pixels increases as the percentage of shared area increases. This will result in descriptors for each of the involved targets that will tend to become more similar with the increasing of the percentage of common pixels in the ROIs. Throughout this document I will refer to this type of target appearance description as the target **global appearance model**.

In this thesis I will present a method to increase the performances of the tracking by detection algorithms that employ this type of ReID network to generate the target global appearance model by flanking it with **local appearance models** that describe local parts of the target in a topologically coherent way. I will illustrate an execution pipeline that performs for each target a separate confrontation for the local and global models, and then fusing the results in a single *scoretables* using a custom designed algorithm inspired by the theory of ensemble classifiers, in order to tackle the label assignment problem for the targets by using the Hungarian algorithm.

The main contributions of this thesis are:

1. the introduction of a novel ”local” appearance model of the targets which performs better than the ”global” appearance description in the case that the target is partially occluded by another target

2. a novel alternative use (respect to the one in SORT) of the Kalman filter to keep the spatio-temporal coherence of the trajectories

3. a custom-designed score level fusion algorithm for resolving association conflicts between the local and global identity association scores

4. a novel selective and incremental update strategy for both the local and global appearance models of the targets, which takes advantage of the respective strengths

The next Chapter is an overview of related literature, chapter 3 presents the theory behind some of the algorithms used throughout the thesis.

Chapter 4 illustrates the operational principles of the entire elaboration pipeline, while chapter 5 goes deeper into the implementation details.

Chapter 6 presents the dataset used, especially for testing, and explains the performance metrics used in the evaluation.

Chapter 7 contains the results of the evaluation of the entire system and those of the individual parts that compose the system, which have also been tested separately.

Lastly, chapter 8 contains some thoughts and conclusion about this work, and draws the lines about possible improvements and future research directions.

CHAPTER 2

Literature Overview

Since almost every person tracking algorithm at the state of the art is based upon the tracking-by-detection paradigm, and this work focuses solely on said tracking methods, this Chapter will explore the various approaches, analysing their strengths and weaknesses. The recent literature emphasises the importance of generating a model for the target appearance in order to address long term occlusion scenarios; the next section describes some among the most performing algorithms evaluated in the most meaningful benchmarks about this topic: the *Multi Object Tracking* (*MOT*) challenge.

2.1 Tracking by Detection in Recent Literature

In [19] Karthik et al. propose SimpleReID, an algorithm based on an unsupervised re-identification network; SimpleReID generates tracking labels using SORT[4] and trains a ReID network to predict the generated labels using cross-entropy loss. In the end, given unlabeled videos and corresponding bounding boxes, SimpleReID first generates tracking labels and then learns a ReID network by predicting the generated label given a detection.

Generating labels: An object detector is run on a video sequence. A Kalman Filter is then run through the detections to obtain short contiguous tracks (tracklets). Then to cluster and associate detections, the Hungarian algorithm is used, obtaining a set of (noisy) track labels for each video, resulting in a pool of labeled video tracklets.

Training ReID models: The authors chose to use SORT to generate the labels, whereas for training the model they decided to assign a unique label to each tracklet and to train a network (with ResNet50 as a backbone) with cross-entropy loss to predict this label given any image from that tracklet. At inference time, for any tracker used, the authors utilize existing tracking algorithms, leaving all the hyperparameters unchanged from the original implementation and simply replacing their supervised ReID model with SimpleReID.

In [16], the authors propose a solution for the tracking problem based on the Tracktor[3] algorithm: they use parts of Tracktor pipeline exclusively to reject false positive detec-

tions and to correct misaligned ones, by sending each input detection through the regression and classification part of their detector, but disabling all tracking parts involved in Tracktor, which is used only to reshape or reject the detections, and not to assign labels to them. Input detections are rejected if Tracktor's detector outputs a confidence score $\sigma \leq 0.5$.

The authors then recover missing detections within the time range of a trajectory with a procedure that they call interpolation and extend a trajectory in forward and backward directions with a procedure which they denote as extrapolation. Afterwords, the authors apply Tracktor to recover eventual missing positions based on the target appearance at the last known position and finally they perform linear interpolation on the remaining gaps.

The method presented in [5], by operating directly on the graph domain, reasons globally over an entire set of detections and predicts final solutions, by applying learning to both feature extraction and to the data association step. The authors propose to combine both tasks of feature extraction and data association into a unified learning-based solver that can:

1. learn features for MOT, and
2. learn to provide a solution by reasoning over the entire graph.

To do so this method learns how to make direct predictions of the trajectories from the final partitions of the graph. This means that the authors perform learning directly in the natural MOT domain: the graph domain. They use a message passing network (MPN)[10] which learns to combine deep features into high-order information across the graph, rendering this method capable of accounting for global interactions among detections. The authors show that this framework yields substantial improvements with respect to state of the art, without requiring heavily engineered features and running up to one order of magnitude faster than some traditional graph partitioning methods.

It is important to note that this is not an online method, in that the system processes the input videos offline in batches of 15 frames, with 14 overlapping frames between batches to ensure that the maximum time distance between two connected nodes in the graph remains stable along the whole graph.

Another noteworthy algorithm is Tracktor[3]. In order to perform the tracking, this algorithm does not rely on anything else besides a detection network (Faster R-CNN with ResNet-101 and Feature Pyramid Networks (FPN)) to perform the tracking. It propagates the BBox position prediction at frame t to the the next frame and considers the IOU of the prediction with that of the regression at frame t+1: if the IOU is below a certain threshold a new label is generated, otherwise the label of the target at frame

t is assigned to the bbox at t+1. Furthermore the authors extend this algorithm with a motion model to compensate camera movements (camera motion compensation (CMC) by aligning frames via image registration using the Enhanced Correlation Coefficient (ECC) maximization) and to solve the problem of long term occlusion they perform ReID by a Siamese CNN for the ”dead” targets and use the model comparison only in the case a new label is about to be generated.

In [1], the proposed people tracking framework works by hierarchically clustering tracklets (which are partial trajectories) to overcome occlusions and minimize association errors, then employs a deep network to evaluate tracklet similarity via ReID by computing pairwise similarity scores between tracklets by jointly learning visual and spatio-temporal features. This network consists of a CNN that learns pairwise detection visual appearance, and two bidirectional RNNs that learn spatio-temporal features, and aggregate visual and spatio-temporal features, respectively. Hierarchical clustering is formulated as a series of constrained minimum cost multi-cut graph problems with vertices representing tracklets, and edges representing tracklet-similarities as computed by the network.

Another notable approach is FairMOT[49], which differs from the previous ones in the fact that it stretches the tracking-by-detection paradigm by performing the detection and the appearance description in parallel. The authors argue that by treating re-ID as a secondary task whose accuracy depends on the accuracy of primary detection task, the learning of the appearance features can be biased, resulting in an increased number of identity switches. Plus by using ROI-Align to extract re-ID features which is directly borrowed from object detection, a lot of ambiguity in characterizing the targets is introduced, since many of the sampling points selected may belong to the background, resulting in noisy informations being included in the final appearance descriptor.

In order to address such problems, the authors resort to a multi-task learning based approach, consisting of a single network with two homogeneous branches for the tasks of target detection and appearance description. This approach treats the two tasks equally, allowing FairMOT to obtain high levels of detection and tracking accuracy. An overview of FairMOT architecture is shown in figure 2.1.

The detection head splits its task into three sub-tasks: the heatmap representation estimates the location of target centers by using the same method which is the standard for the landmark point estimation task, while the bounding boxes sizes and offset are handled separately as two different sub-tasks.

The author demonstrate that this *anchorless* approach to the detection task, paired with the fact that the Re-ID is performed in parallel on an homogenous branch, decreases the negative effect of noise and background data on the final target appearance

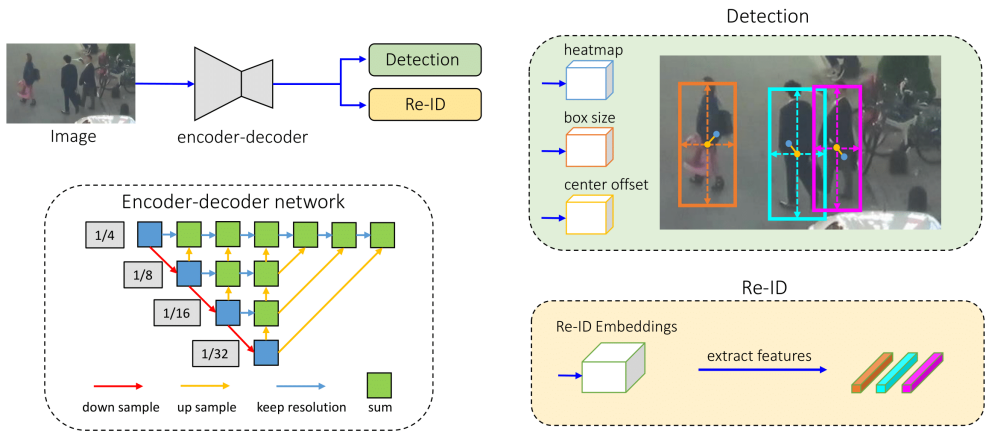

Fig. 2.1 Overview of FairMOT. The input image is fed to an encoder-decoder backbone based on ResNet-34 to extract high resolution feature maps (with stride=4). The maps are then propagated to two homogeneous branches for detecting objects and extracting re-ID features, respectively. The features at the predicted object centers are then used for tracking.

descriptors, thus leading to a reduced number of identity switches.

It is worth noting also that this approach, in its official implementation, has proven to perform respectively at $30.5Hz$ and $25.9Hz$ on the MOT15 and MOT16 datasets, with MOTA scores of 60.6 and 74.9, which is way faster than other algorithms of similar accuracy.

CHAPTER 3

Theoretical Background

This chapter presents a theoretical overview of algorithms that have been used throughout this thesis, which will be useful in order to understand the details of the approach and of the implementation presented in subsequent chapters.

3.1 Alpha Beta Filters

An alpha beta filter is a simple observer for estimation, which means that it provides an estimate of the internal state of a system by observing measurements of its input and output. Its application is usually in control theory and its main advantage against most of its more complex competitors such as the Kalman Filter lies in its low computational cost[39][38][43][17].

Alpha beta filters rely on the assumption that the system in question can be adequately approximated by a model having only two internal states. The first state is obtained by integrating the value of the second state over time, such as in simple mechanical systems, where position can be obtained as the time integral of velocity. There will be no in-depth explanation of the inner workings of the AB-filters, as its performance was lacking for the particular application in my proposed algorithm, as it introduced extreme drifting of the prediction after very few missing observations, and was in the end discarded in favor of the Kalman filter, which even though more computationally expensive, outperformed the AB-filter in precision and robustness.

3.2 Kalman Filters

The Kalman filter (KF) or linear quadratic estimation (LQE)[18], is an algorithm using a set of measurements from different sources (or sensors) taken over time, which are called observations; observations usually contain noise, coming both from the environment and the sensors used to take the measurements. From a series of observations the KF produces estimates of a set of unknown variables, constituting the state of the system. Such estimates are usually more accurate than estimates based on a single measurement alone, especially if there is a correlation between the different measurements

in the observation; this is due to the fact that the KF estimates the joint probability distribution over the measurements for each timeframe.

Kalman filters are often used in trajectory optimization in the field of robotics and motion planning. The algorithm is a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables and their uncertainties. Afterwards, in the second step, such estimates are updated with a weighted average. The higher weights are given to those estimates having an higher certainty. The algorithm is recursive and can run in online modality, since it only needs the present input measurements and the last state estimate alongside its uncertainty matrix.

For the purpose of this work, the capability of trajectory optimization and position estimation from uncertain, noisy measurements is the most interesting among all the possible applications of the Kalman Filter. As mentioned above, the KF is a mean squared error minimiser. KF can estimate the state of the system in the future from the past observations of the state $\vec{x}$. The Kalman filter assumes that all state variables are random and Gaussian distributed. Each of these state variables can be represented by a mean value μ and a variance σ^2, which can be viewed as the measurement uncertainty. If the state variable are correlated the KF is able to predict the next state reducing the uncertainty. This can be observed in the covariance matrix Σ. So the state at time k can be modeled by the vector of the means of the state variables $\hat{x}_k$ and their covariance matrix P_k. The goal of the KF is to predict the next state (at time k) by using the information in the observations of the previous states (up to the time $k-1$). This is done by constructing a prediction matrix F_k such that:

$$\hat{x}_k = F_k \hat{x}_{k-1}$$
$$P_k = F_k P_{k-1} {F_k}^T$$

It is to note that this simplified form of the KF formulation does not account for external influences, while the complete one does that by modelling them in the form of a control matrix B_k and a control vector $\vec{u}_k$, plus it is possible to treat the untracked influences as noise with covariance Q_k, so that the prediction problem can be described as:

$$\hat{x}_k = F_k \hat{x}_{k-1} + B_k \vec{u}_k$$
$$P_k = F_k P_{k-1} {F_k}^T + Q_k$$

This means that the KF predicts the new best estimate from both the previous best estimate and a correction for known external influences, while the new uncertainty is a

prediction obtained from the old uncertainty while taking into account some additional uncertainty coming from the environment.

In the real world, information about the state is taken by sensors; a sensor produces a reading which can be considered as a indirect observation and might not be the same as the units and scale of the state we considered in the model. The sensors can be modeled through a matrix H_k, and the distribution of the predicted measurement will be:

$$\mu_0 = H_k \hat{x}_k$$
$$\Sigma_0 = H_k P_k {H_k}^T$$

So the sensor readings are represented by a distribution, with covariance R_k which models the sensors' noise, and mean $\vec{z}_k$:

$$\mu_1 = \vec{z}_k$$
$$\Sigma_1 = R_k$$

The KF combines the information contained in the sensor readings distribution and the prediction in order to reduce the uncertainty. In order to do so it is necessary to define the Kalman Gain K:

$$K = \Sigma_0 (\Sigma_0 + \Sigma_1)^{-1}$$

which implies that:

$$\mu' = \mu_0 + K(\mu_1 - \mu_0)$$
$$\Sigma' = \Sigma_0 - K\Sigma_0$$

Thus obtaining the new best estimate mean and covariance for the next state:

$$\hat{x}'_k = \hat{x}_k + K'(\vec{z}_k - H_k \hat{x}_k)$$
$$P'_k = P_k - K' H_k P_k$$
$$K' = P_k {H_k}^T (H_k P_k {H_k}^T + R_k)^{-1}$$

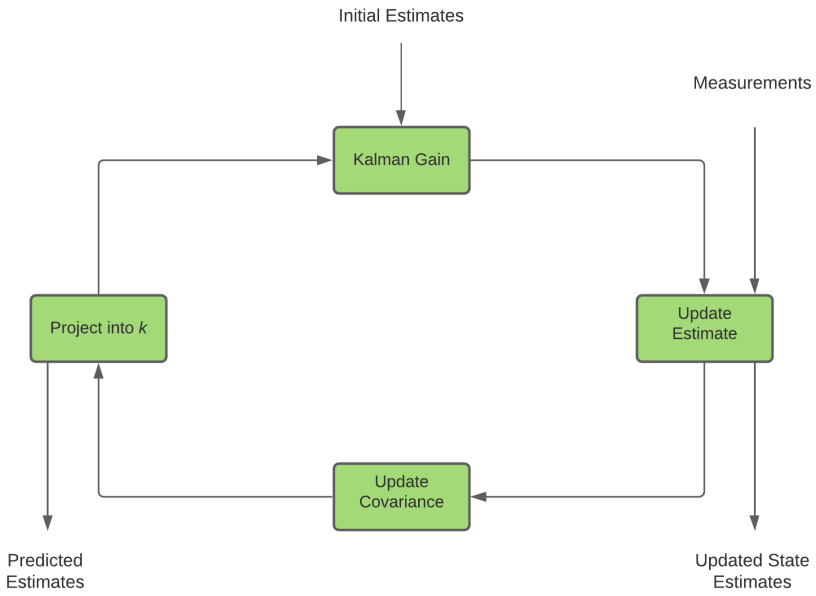

Fig. 3.1 An overview of the Kalman filter execution pipeline

3.3 Image Hashing

Image hashing techniques are a form of perceptual hashing[6] techniques, meaning that the algorithms produces a *fingerprint* of the target multimedia. An image hashing algorithm works by constructing a hash value that uniquely identifies an input image based on the contents of an image, thus the algorithm computes an image hash based on the input image's visual appearance, so that images having a perceptually similar appearance will result in having similar hashes.

Image hashing functions differ from regular hashing functions such as MD5[37] or sha-1[31] in nature because traditional hash functions are sensitive to small local variations in the input, and in the case of images it translates in the fact that altering the color of a single pixel will produce a completely different checksum with respect to the original. In order to judge the perceptual similarity in the content of two images in a robust manner, hash functions that produce hash collision in case of similar input content are required.

Many different algorithms have been devised to this purpose, some of which are capable of detecting similarity only in the texture content, while others capable of detecting similarity also in the color content. This means that it is possible to encode both texture and color information of a given image in a fixed length appearance descriptor[41].

The difference in similarity between two images are then computed by calculating the Hamming distance between the two respective hashes.

Again, since in the end the performances of perceptual hashing observed through

tests for my particular application resulted insufficient, I will not go in depth explaining the detail of each algorithm, but for the sake of completeness, here follows a brief summary of the algorithms that have been tested: among the perceptual hashing algorithms which do not take into account the color of the patches, the most interesting were aHash[21] (average hash), pHash[20] (perceptive hash) and dHash[22] (difference hash), which require gray-scale images as input. They were considered because they were known to be used in the context of image retrieval. From tests performed on my dataset of patches (see chapter 6), aHash resulted to be fast but it generates a huge number of false positives, phash is the best performing of the three in terms of accuracy, with fewer false positives and false negatives, but it is slower, while dHash generates few false positive and it is as fast as aHash. As previously mentioned, also some color-invariant perceptual hashing technniques exist, such as the one presented in[41], which is based on the invariant moments.

3.4 CIELab Color Space and Delta-E Metrics

Color difference as perceived from a human perspective it is not easily translated by most used color spaces such as RGB or HSV, as they are not perceptually uniform[33].

The CIELAB color space, defined by the International Commission on Illumination (abbreviated CIE) in 1976[32], expresses color as a triplet of values:

- L* representing perceptual lightness
- a* and b* representing the four unique colors of human vision: red, green, blue, and yellow.

CIELAB is a non-linear transformation of another color space, CIEXYZ, and it was intended as a perceptually uniform space, meaning that a given numerical change would produce a similar perceived change in color.

In order to convert from RGB color space to LAB, it is necessary to pass through the XYZ color space first:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where

$$
\begin{aligned}
M &= \begin{bmatrix} S_r X_r & S_g X_g & S_b X_b \\ S_r Y_r & S_g Y_g & S_b Y_b \\ S_r Z_r & S_g Z_g & S_b Z_b \end{bmatrix} \\
X_r &= \frac{x_r}{y_r} \\
Y_r &= 1 \\
Z_r &= \frac{1 - x_r - y_r}{y_r} \\
X_g &= \frac{x_g}{y_g} \\
Y_g &= 1 \\
Z_g &= \frac{1 - x_g - y_g}{y_g} \\
X_b &= \frac{x_b}{y_b} \\
Y_b &= 1 \\
Z_b &= \frac{1 - x_b - y_b}{y_b} \\
\begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} &= \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix}
\end{aligned}
$$

and

- (x_r, y_r), (x_g, y_g) and (x_b, y_b) are the color coordinates in the RGB system
- X_W, Y_W and Z_W are the RGB system reference white

Then, from XYZ we can convert to LAB by the following:

$$
\begin{aligned}
L &= 116 f_y 16 \\
a &= 500(f_x f_y) \\
b &= 200(f_y f_z)
\end{aligned}
$$

where

$$
\begin{aligned}
f_x &= \begin{cases} \sqrt[3]{x_r} & \text{if } x_r > \epsilon \\ \frac{Kx_r+16}{116} & \text{otherwise} \end{cases} \\
f_y &= \begin{cases} \sqrt[3]{y_r} & \text{if } y_r > \epsilon \\ \frac{Ky_r+16}{116} & \text{otherwise} \end{cases} \\
f_z &= \begin{cases} \sqrt[3]{z_r} & \text{if } z_r > \epsilon \\ \frac{Kz_r+16}{116} & \text{otherwise} \end{cases} \\
x_r &= \frac{X}{X_r} \\
y_r &= \frac{Y}{Y_r} \\
z_r &= \frac{Z}{Z_r} \\
\epsilon &= 0.008856 \\
K &= 903.3
\end{aligned}
$$

and

- (X_r, Y_r, Z_r) is the reference white
- (X, Y, Z) is the XYZ value

In practice the LAB space is not really perceptually uniform[32], but it is still used in industry for detecting small differences in color.

In order to quantify the perceived difference between colors, a purposefully designed metrics was introduced in 1976: the Delta-E (or ΔE^*_{00})[42]. In 1994 and then in 2000 the formula was udated because the CIELAB obtained space was far from being perceptually uniform as intended, especially in the saturated regions.

For completeness here follows the δE_{00} formula, from the 2000 update:

$$
\Delta E^*_{00} = \sqrt{\frac{\Delta L'}{k_L S_L}^2 + \frac{\Delta C'}{k_C S_C}^2 + \frac{\Delta H'}{k_H S_H}^2 + R_T \frac{\Delta C'}{k_C S_C} \frac{\Delta H'}{k_H S_H}}
$$

Where:

- R_T is the hue rotation term, needed to handle hue angles near $275°$
- S_L is the lightness compensation term
- S_C is the chroma compensation term

- S_H is the hue compensation term
- $\Delta L' = L_2^* - L_1^*$

3.5 Mixture Of Gaussians Approximation

Mixture of Gaussians (MOG) are often used in background estimation[40] for achieving the foreground-background segmentation. MOG assumes that we deal with a pool of images of the same size as input. Every pixel's intensity values in the pool of images is modeled using a Gaussian mixture model. This way a statistical model of the intensities (it can be easily extended for multiple channel data) of each pixel position is produced. The mean of each Gaussian (or of each pixel in this case) can be used to visualize the image approximation over the entire pool.

3.6 Convolutional Autoencoders

A convolutional autoencoder (or CAE)[8][30][47][12] is a neural network trained to reproduce its input image in the output layer. Autoencoders in general, and by extension convolutional autoencoders are composed by two major components: an encoder and a decoder. The input image is passed through the encoder, which is a convolutional network that produces a low-dimensional representation of the image. The decoder, which is another convolutional network, takes this compressed image representation and tries to reconstruct the original image.

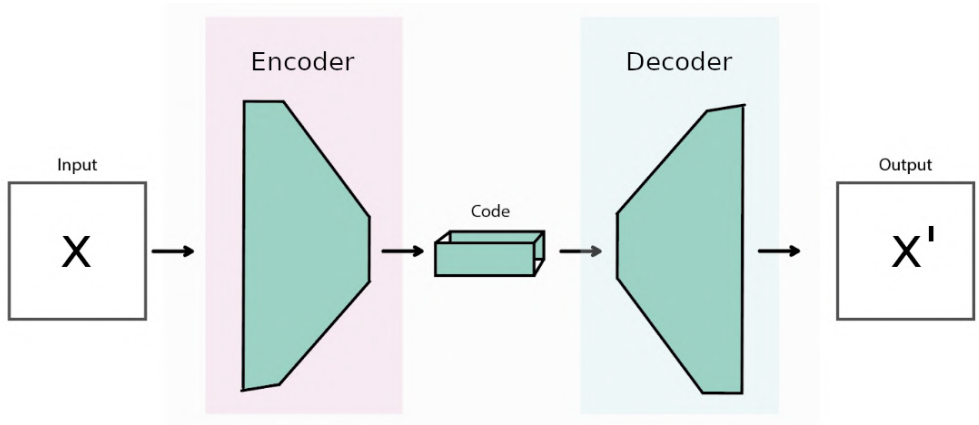

Fig. 3.2 The generic architecture of the autoencoder. The encoder part compresses the image X in a lossy manner, then, from the generated code (compressed representation) the decoder reconstructs X' which, if the architecture has been correctly designed and trained, is a noise-filtered representation of the original input

Often this type of autoencoders have been used for image compression[8], but this is not the only application. Since the encoder network is typically a convolutional pyra-

mid, each convolutional layer is followed by a max-pooling layer, reducing the dimensions of the layers. This leads to a lossy compression which, with a proper training, is a mean to filter out the noise, keeping only the significant features of the input. The decoder converts the input from a sparse representation to a fully reconstructed image, returning from the compressed lower dimensionality to the original one. Therefore another common application of autoencoders is data (image) denoising[30][47][12]. The application of CAEs that is interesting in the context of this thesis is yet another one: image confrontation. Training a CAE to compress and then reconstruct a certain type of images, it can be seen as learning to extract a compressed, denoised form of the image (in practice a descriptor of the meaningful features, filtering out the noise), and then reconstruct it. At inference time it becomes then possible to detach from the trained model the decoder part and just keep the encoder. By flattening the output of the encoder a descriptor of the original image can be obtained, such descriptor can then be confronted with the descriptors of other images by using a predetermined distance like the angular distance (i.e. cosine similarity). This is the principle used by some image retrieval algorithms.

3.7 Ensemble Classifiers and Score Fusion

Ensemble methods refer to those machine learning methods that use multiple learning algorithms for obtaining a better predictive performance than the one which could be obtained from any of the constituent learning algorithms as a stand-alone[52].

An ensemble is a supervised learning algorithm, deriving from the aggregation of multiple, independent algorithms. The trained ensemble generates a single hypothesis, which is not necessarily contained within the hypothesis space of any of the separate models that compose it.

It may seem at a first glance that this solution may present more chances of over-fitting than the individual algorithms of which it is composed, but in practice many ensemble techniques (and especially bagging) tend to reduce the chances of over-fitting on the training data. In fact ensembles tend to perform better when there is a significant diversity among the single models it aggregates. Ensemble methods train multiple learners to solve the same problem and then try to combine them. The set of learners contained by an ensemble are called base learners. When the different learners in the ensemble belong to the same type, the ensemble is defined as homogeneous, otherwise it is called heterogeneous.

One of the strengths of ensemble methods is that they are able to greatly boost the performances of a group of weak (base) learners, resulting often in better performances than those of a single strong learner designed to make very accurate predictions.

After selecting a set of base learners, rather than trying to find the best single learner,

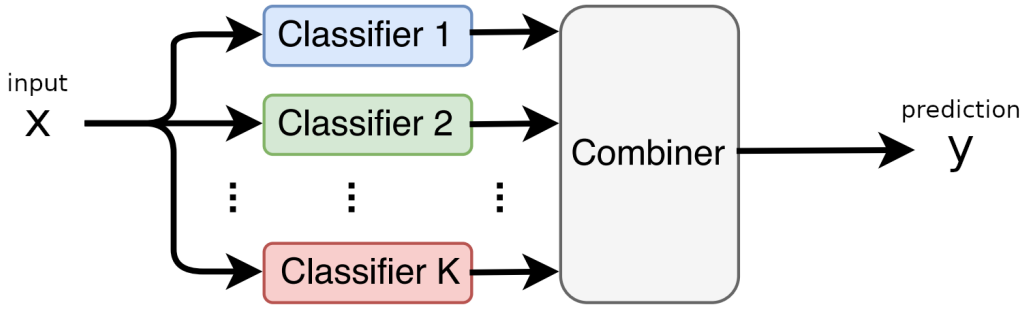

Fig. 3.3 The generic architecture of an ensemble classifier, where multiple, independent classifiers sharing the same task contributions are merged to obtain a single, unified prediction. Note that in this case they all share the same input, but they can also have independent input sources

ensemble methods resort to combination to achieve a strong generalization ability. The reason is tri-fold: -the hypothesis space is too large for the limited training data, combining different learners that cover different parts of the hypothesis space can reduce the chance of choosing the wrong hypothesis to base the decision on. -reducing the chance of being stuck in a local minimum by combining the hypothesis. -combining the hypothesis may expand the space of representable functions leading to a more accurate representation of the true unknown hypothesis.

There are many ways to perform the combination of the results of the different learners; one of the simplest is by averaging. Averaging is the most popular combination method for numeric outputs, and can be performed under many different variants: -Simple averaging, as implied by the name, combines the outputs by averaging the values from individual learners. The use of this method is based on the assumption that the errors of individual learners are uncorrelated (which typically is not true) and that each learner has similar individual performance. -Weighted averaging obtains the combined output by averaging the values of individual learners, each multiplied by a different weight, implying different importance (accuracy) of the learners. It can be noted that simple averaging as well as methods such as voting are different special cases of weighted averaging. Of course weighted averaging assumes that the weights are to be chosen in a meaningful manner. The way this is done is strictly dependent on the situation and is outside the scope of this discussion. -Voting methods are a family of methods similar in concept to weighted averaging, but are more suited to nominal outputs instead of numerical.

3.8 Data Association and Hungarian Algorithm

For many visual algorithms such as tracking and re-identification, one of the main challenges has always been the association problem. The Data association is the problem

of assigning labels (identities in the case of re-ID or tracking) to a set of measurements. Even if both the aforementioned cases involve the assignment of identities, the two are very different for the level of complexity: in fact, in the case of re-ID the problem consists of a one-to-many association, which means that every query is handled as independent from any other, resulting to the confrontation of a query targets' measurements with a set of all the measurements of the targets in the gallery; in the case of tracking instead, the problem is much more involved, as many target are simultaneously present in a single scene (frame) and duplicate identity association must be avoided, rendering necessary a many-to-many confrontation between the all query targets and all the gallery targets.

Since this thesis deals with the tracking problem, the many-to-many association must be the focus (plus for the one-to-many paradigm a simple greedy best match is already a solution). There are many algorithms that perform a many-to-many association, but there are some intrinsic problems with this type of algorithms: usually their scalability for handling large numbers of objects is very poor (i.e. their complexity is very high), plus, for the specific task of tracking, the presence of occlusions can break the assumptions that the tracker makes, rendering the association unreliable.

In literature many classic sequential data association approaches refer to the over-the-radar (OTR) algorithms, such as global nearest neighbour standard filter (GNNSF), joint probabilistic data association (JPDA) and multiple hypothesis tracking (MHT).

In the GNNSF algorithm, gating means defining a search region for a list of measurement-to-track candidates, eliminating unlikely measurement-to-track candidates that are located outside this region in order to reduce computation and noisy associations.

GNNSF considers all possible measurement-to-track assignments within appropriate gating regions and generates the most likely assignment hypothesis by solving a 2D binary assignment problem.

While GNNSF updates the tracks based on a single measurement from the best assignment solution, JPDA takes into account all the measurements inside the gate: the state estimate of a track is given by the weighted sum of contributions given by all the feasible measurements.

MHT takes into account all possible association over time, using future measurements to resolve ambiguities in the current frame. To do so, MHT propagates the current hypotheses, waiting for subsequent data in order to achieve a better estimation. The computational cost limits the scalability of MHT, but heuristic techniques have been adopted to enable real-time performance.

Summarizing GNNSF is a heuristic approach prone to produce non recoverable misassociations, JPDA is a sub-optimal method using Bayesian estimation of the object state by combining the information from all the measurements that are possibly

associated, but due to the size of the enumeration of all joint association events, only approximations of this method can be used in practice. Finally MHT addresses the association problem by maximizing the a posteriori probability of measurement-to-track associations given all measurements collected so far, but the complexity and computational cost of the algorithm limit its application in the real world, so approximations of the algorithm are used in practice.

Another approach is known as the tracklet linking method; a tracklet is a trajectory segment. Trajectory segmentation occurs because of occlusion or misassociation. Tracklet linking problem is an extension of the data association problem, involving association of both single observations and already established tracks. Tracks contains more information than single observations. Data association ambiguity can be solved by optimizing a cost function that considers the smoothness of object motion and appearance over several frames, making it possible to stitch together different tracklets to recover full trajectories, and this can be achieved by using track graphs.

In the tracking by detection paradigm instead, one of the most used strategies to perform the identity association task is the Hungarian algorithm (or Munkres algorithm), developed and published in 1955 by Harold Kuhn. The Hungarian algorithm is an optimization algorithm with polynomial complexity for solving the assignment problem. Its time complexity is $O(n^4)$, but there are variant and implementations that reduce the complexity to $O(n^3)$.

In its matrix formulation, the algorithm assumes a $n \times n$ matrix, but it can be extended to $n \times m$ matrices by the use of padding. It can be formulated both as a cost minimization problem or a similarity maximization problem, so in the case of assignment the matrix often contains the similarity scores for each couple of targets confronted by certain metrics.

3.9 Deep Learning Frameworks

This section introduces some deep learning architectures that have been used in this thesis in their original form, from the respective official implementation and with the official model and pre-trained weights, without any major modification.

3.9.1 Detectron2

Detectron2[45] is Facebook AI Research's software system that implements state-of-the-art object detection algorithms, originating from Mask-RCNN[14].

Detectron2 is essentially a framework containing many different network models, designed for different tasks; figure 3.4 illustrates one of such models, which includes

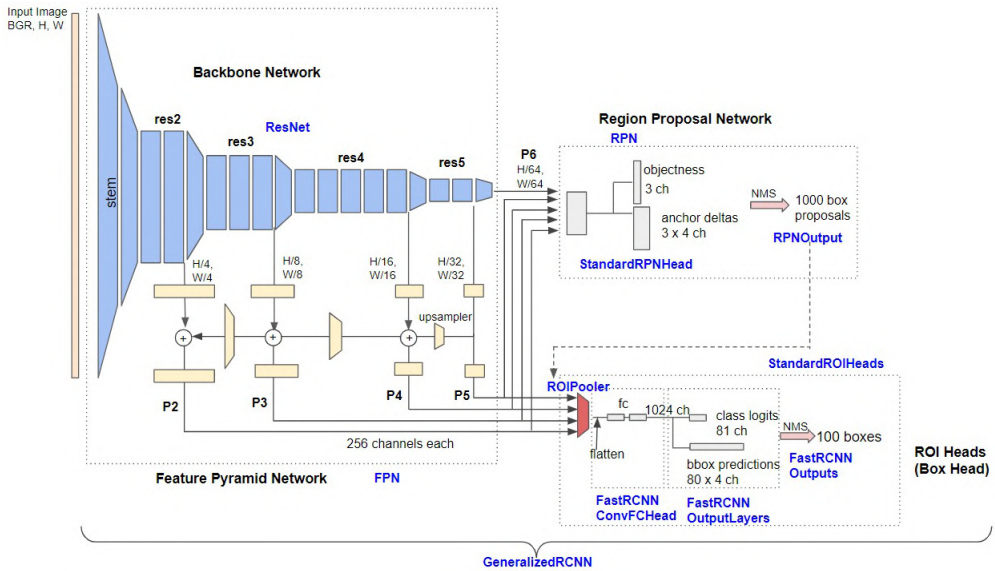

Fig. 3.4 This figure illustrates Detectron2 backbone (ResNet50+FPN), the Region Proposal Network (RPN) and the head tasked with predicting the coordinates of the bounding boxes for the targets.

a *backbone*, the *region proposal network* and the *bounding box head*. Other models include also other heads specialized for different tasks, such as *semantic segmentation*, *pan-optic segmentation* and *skeleton landmarks detection*.

The model used in this thesis was taken from the official Facebook Research GitHub repository, and the model with ID 137849621 was selected from their Model Zoo; the model is built upon a ResNet50+FPN[15][27] backbone, with standard convolutional and fully connected heads for skeleton landmarks and box prediction, respectively. It is designed to obtain the best speed/accuracy trade-off and the weights were trained on Microsoft COCO[28] dataset for both the tasks of object detection (boxes) and skeleton landmarks prediction. It achieves an average inference time of 0.066 seconds per image and reaches an average precision of 55.4 for the boxes and 65.5 for the landmarks.

3.9.2 DG-Net

DG-Net is a person Re-Identification framework presented in [51]. DG-Net consists of a learning framework that jointly couples discriminative and generative learning in a unified network. The generative module is composed by encoders which decompose each pedestrian image into two latent spaces: an *appearance space* encoding appearance and identity related semantics; and a *structure space* enclosing geometry and position related structural information. The appearance space encoder is also shared with the discriminative module, which serves as a re-id learning backbone. Summarizing, the generative module produces synthesized images that are taken to refine the appearance encoder online; the encoder, in turn, influences the generative module with improved

appearance encoding; and finally both modules are jointly optimized, given the shared appearance encoder. An overview of the architecture can be seen in figure 3.5.

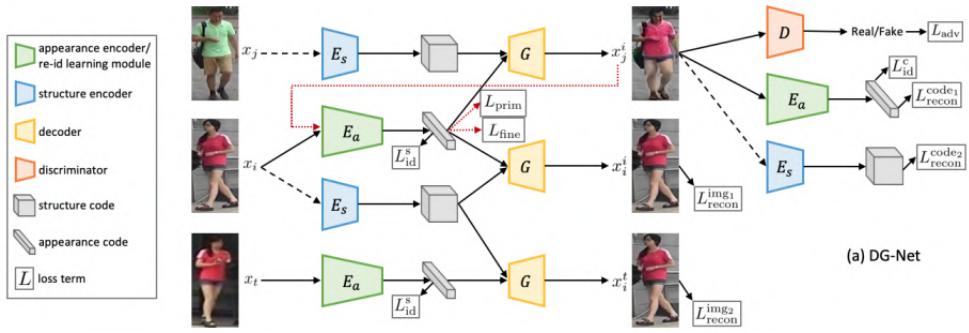

Fig. 3.5 A schematic overview of DG-Net. The discriminative re-id learning module is embedded in the generative module by sharing appearance encoder E_a. A dash black line denotes the input image to structure encoder E_s is converted to gray. The red line indicates the generated images are online fed back to E_a.

The implementation used in this thesis was not from the official repository as it was not yet present at the moment of implementation; the model and pre-trained weights were instead taken from an unofficial GitHub repository. The backbone is based upon ResNet50, and with the pretrained weights selected it is capable of achieving a $Rank@1$ score of 88.24 and a mean average precision of 71.59 on the Market1501[50] dataset, which is way less than the latest model on the official repository implementation, reaching a $Rank@1$ score of 94.8 and a mean average precision of 86.0 on the same dataset.

3.9.3 FD-Gan

FD-Gan[11] is another Re-Identification framework aiming to achieve the generation of person appearance descriptors which are completely pose-independent. In order to achieve this goal, the authors propose a Feature Distilling Generative Adversarial Network (FD-GAN), capable to learn identity-related and pose-unrelated representations. This framework is based on a Siamese structure with discriminators on human poses and identities. In addition to such discriminators, a novel same-pose loss is also integrated, requiring appearance of a same person's generated images to be similar. The architecture of FD-Gan can be seen in figure 3.6.

The notable thing is that at inference time, all the information on the pose of the target that were needed for training are unnecessary, as the appearance descriptor learned is pose independent, meaning that the network loaded at inference time is much less resources expensive than the one used in training.

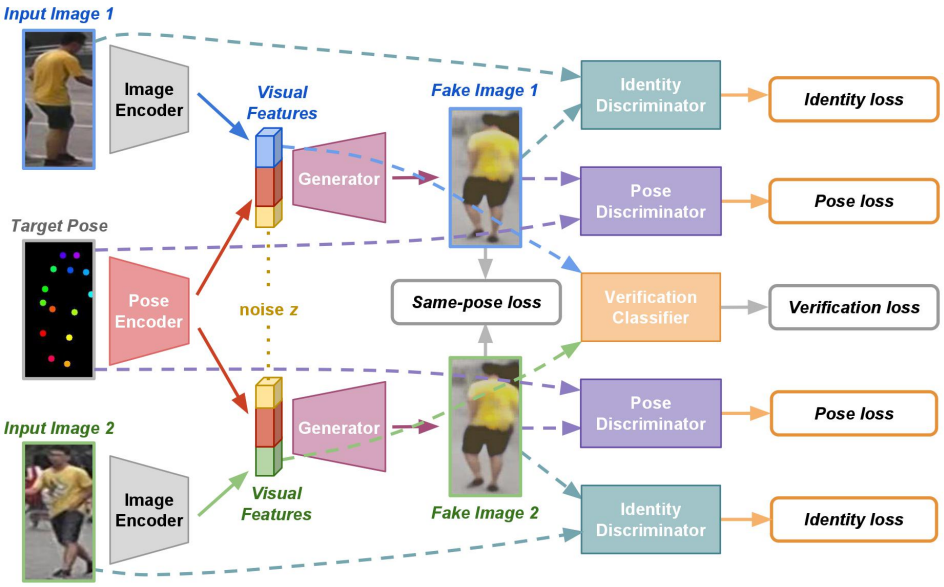

Fig. 3.6 The Siamese structure of the proposed FD-GAN. Robust identity-related and pose-unrelated features are learned by the image encoder E with a verification loss and the auxiliary task of generating fake images to fool identity and pose discriminators. A novel same-pose loss term is introduced to further encourage learning identity-related and pose-unrelated visual features.

CHAPTER 4

Problem Statement and Proposed Solution

This chapter will go deeper in detail about a common problem of tracking algorithms that rely on re-identification networks to model the appearance of the targets (as mentioned in the introduction), and then a family of possible solutions will be presented.

4.1 Tracking by Detection

All multi-object tracking algorithms based on the tracking by detection paradigm achieve their goal by subdividing it into two separate tasks:

1. Detecting the targets
2. Assigning the identities to the detected targets

The detection task is usually entrusted to a specifically designed deep neural network, such as YOLO[35] or Faster-RCNN[36]. Usually such detectors take as input the whole frame and return as output the coordinates of the bounding boxes containing the targets, as well as the class of each target, and optionally other information (some detector also returns masks that identify all the pixels belonging to the target[14],[45]).

After the coordinates denoting the *regions of interest* (*ROI*) of each target, the tracking algorithm can extract additional information on the targets in order to formulate the assignment problem as an optimization problem. The information that can be extracted from the ROIs can for instance be spatio-temporal, as it happens in SORT[4], and can be used in order to predict the position of the target in the next frame or to build a motion model of the target; the problem with this type of approach is that they lack robustness in case of long term occlusions.

In fact, since the target position used in motion models and Kalman filters is often the center of the ROI of the target itself, the accuracy of the measurement strictly depends on the quality of the detection. Of course Kalman filters partially compensate small measurement errors, but when a target suddenly becomes partially occluded,

it can experience extreme variations on the size and aspect ratio of the detected ROI, which in turn results in great measurement errors.

The real problem though, is when a target becomes completely occluded (or just non detected) for a number of consecutive frames: since as we said the accuracy of the prediction of Kalman filters is somewhat influenced by the accuracy of the measurements, when we encounter cases of multiple consecutive missing observations of a target, the prediction of the KF rapidly drifts from the actual position of the target, causing mislabelling.

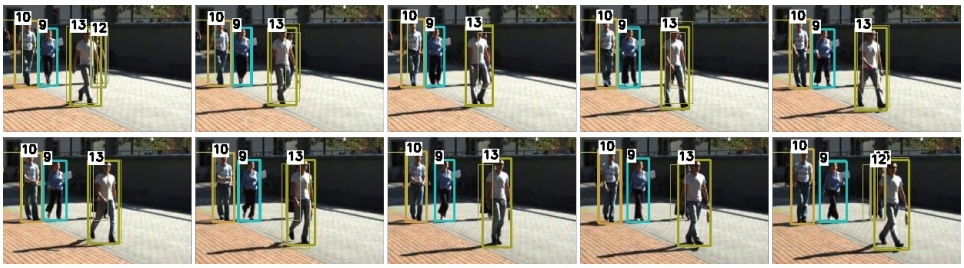

Fig. 4.1 Example of drifting prediction of the Kalman filter with respect to the actual target position after repeated missing observations. The Kalman filter box position prediction for each target is represented by the box of the same respective color as the target, but with thinner border thickness. In the sequence of pictures (from MOT15 dataset), target 12 is completely overlapped by target 13, leading to missing observations, leading to incorrect box position predictions.

In fact, in SORT, the authors explicitly chose not to address the problem of long term occlusion, by considering a target as *lost* after just a single missing observation. In the light of this problem, in their subsequent work, the authors of SORT integrated appearance information to improve the performance of the algorithm, creating deepSORT[44]. The introduction of an appearance model of the target can compensate for the lack of robustness in the case of prolonged missing observations of the targets, in that the model can be used to recover the track after a long term occlusion independently from the performances of the Kalman filter.

It is today common practice to incorporate an appearance model of the target in order to recover the tracking of the same after prolonged occlusions[1][3][5][16][19][49].

4.2 ReId Network Appearance Modelling

Re-Identification (*ReID*) is the task of recognizing the same target across different settings (usually through different cameras), which may vary by lighting conditions/time of the day, resolution of the images, position/angle and distance of the camera with respect to the target, target pose, etc.

The most common approach at the state of the art to perform ReId is to use a deep

neural network architecture in order to create a model of the target. The input of such networks is usually the already extracted rectangular ROI containing all the pixels of the target and as few as possible pixels belonging to the background; the output of the network is a numerical descriptor (usually a 512 Bytes vector) that represent the target appearance information.

The networks are trained in such a way that, independently of the conditions (illumination, perspective, resolution, background clutter, target pose, etc.), different instances of the same target will result in similar descriptors with respect to a defined metric, for instance the cosine similarity.

4.2.1 Limitations of the Re-Identification approach

The principle above mentioned works really well as an approach to the ReId task, and is now the most commonly adopted solution for modelling the targets appearance in the tracking task. The problem is that the ReId task is a much simpler problem, since it involves the association of the identity of a single query target from a pool of gallery targets (*one to many*), while the tracking task require the association of the identities a number of targets detected in the current frame from the pool of gallery targets seen in previous frames (*many to many*), with the possibility that the target has not been seen before (is not present in the pool). Usually the association step in the tracking task is tackled by using the Hungarian algorithm[23], which maximizes the sum of the similarity scores for all the targets in the frame.

This system would give perfect results if the scores could differentiate enough between the targets, but in reality many problems may arise, such as partial occlusions, incorrect ROI coordinates (the ROI is not centered on the target or has the wrong aspect ratio/dimension) or targets with very similar appearance may be present (same/similar clothing).

All of this problems are still independent from the method used to extract the appearance features, but there is an intrinsic flaw in the use of a ReId network for modelling the targets appearance: as previously explained, such networks receive as input a rectangular ROI, which contains (ideally) all pixels belonging to the target and as few as possible pixels not belonging to the target. The *signal to noise ratio* (*SNR*) (target pixels/background pixels) inside of the extracted ROI is dependant by many factors, such as the quality of the detector, but also the pose of the target, which directly influences the aspect ratio of the ROI. A higher SNR implies a cleaner, more accurate description of the target appearance, less influenced by noise. In fact, the descriptors generated by such networks are still influenced by the background content inevitably present in the ROIs.

Figure 4.2 proposes a simple scheme of a tracking-by-detection algorithm, which

relies only on ReId features extracted from the detected ROIs for performing the identity association.

In real multi-target tracking applications this can become a huge disadvantage, especially in a particular (yet very common) case, which will be denominated **targets collision**. Targets collision (or simply collision) defines the case when two (or more) detected targets have partially overlapping ROIs; this translates in the fact that the ROI of each of the involved targets contain pixels belonging to some of the other involved targets, which results in the ReId network generating descriptors for different targets that tend to become more similar as the percentage of shared pixels between the ROIs involved increases (a visual demonstration of this phenomenon is depicted in figures 4.3 and 4.4).

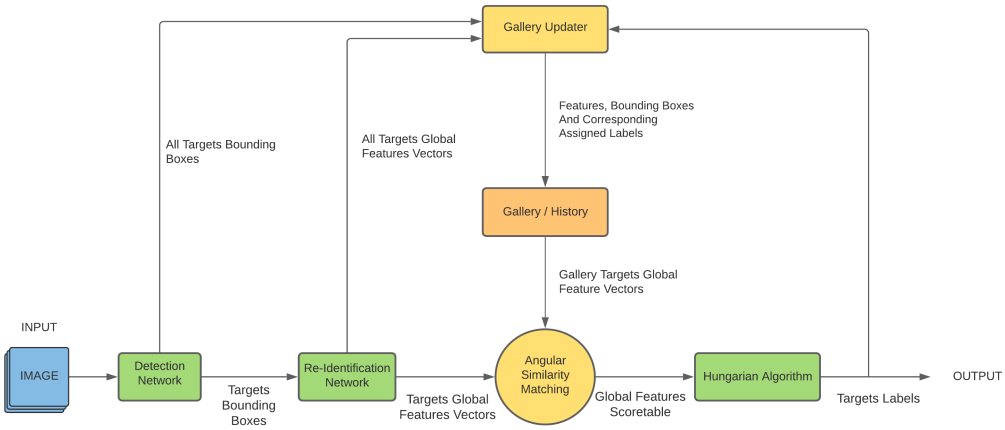

Fig. 4.2 Overview of the simple pipeline for multi-object tracking, relying only on the *global* appearance descriptor for generating the scores for the identity association

Similar descriptors for different targets will generate less distinct feature that may lead to completely wrong matching scores which may corrupt the work of the algorithm used for the identity association, resulting in mislabelling and noisy or interrupted tracks for the involved targets.

Even though this problem may be crippling for tracking in crowded environments, the benefits of such descriptors for recovering the tracking of a target after a long term occlusion is undeniable. In the next section will be presented a general idea for a family of possible solutions to flank the ReId network appearance modelling with the intent to make up for the shortcomings mentioned above.

4.3 Possible Solutions

The ideas presented in this section are not meant as a definitive approach, but as a baseline for future and more refined work. This work is in fact intended as a proof of

Fig. 4.3 A sequence of (cropped) images from the test set, with labels assigned by the system using only the global features (with DG-Net). Note that target 3 (bounding box in pink), during the partial occlusion suffers from mislabelling, caused by an identity switch.

concept, demonstrating the potentiality of this type of solutions.

4.3.1 The General Idea

The concept proposed is based on the idea of complementing the target descriptor generated by the ReId network (which from now on will be referred to as the target's **global descriptor**) with a set of features describing parts of the target (and will be referred to as the target's **local descriptor**).

In order for the idea to make sense, such local descriptors should have the following characteristics:

1. Must be extracted only from target pixels, so avoiding the risk to be affected by background pixels data
2. Must be invariant to changes in illumination, pose, blur, etc.
3. Must be topologically coherent (i.e. it must not be possible to confront a feature which describes a body part, for instance the left arm, with a feature describing a different body part, for example the right leg)

At this point each target will have both a global and a local descriptor; the idea for handling the identity association task is also very simple: designing an algorithm for fusing these separate information, in order to have a single similarity score for each

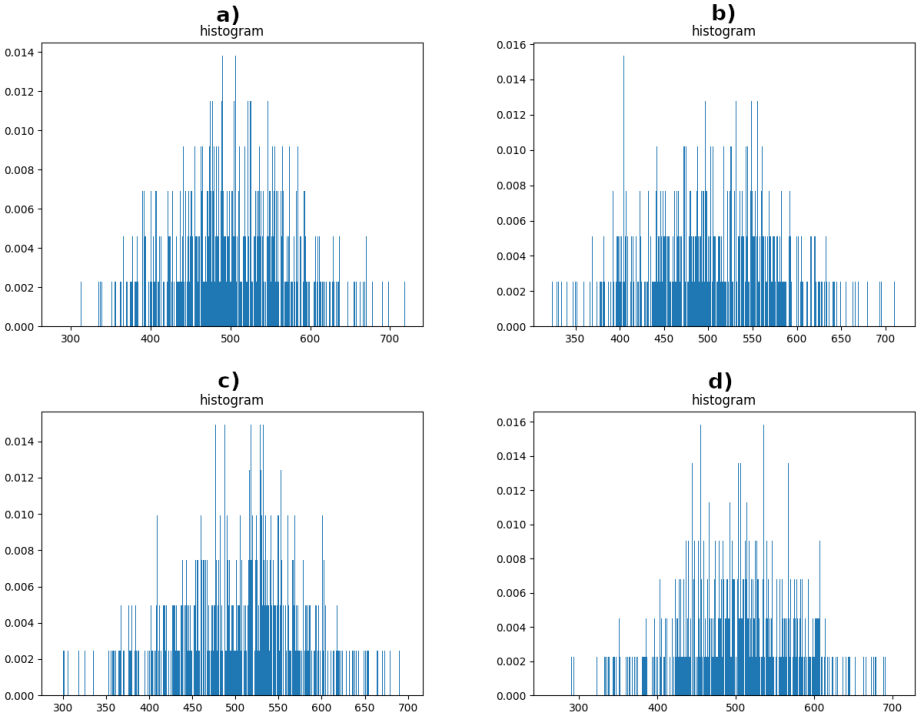

Fig. 4.4 Overlapping influence on the descriptors, referring to the scene in figure 4.3. a) target #3 when not occluded, b) target #3 during partial occlusion, when the ID-switch happens (mislabelled as #4), c) target #1 when not occluded, d) target #1 during the same partial occlusion described for b). The descriptors are extracted with DG-Net. The descriptors are depicted as histograms, where each bin in the horizontal axis represent a position in the tensor, and the values int the vertical axis have been scaled (multiplied by a constant) in order to make the visual interpretation easier

couple of query-gallery targets, which then opens the possibility of using the standard Hungarian algorithm for the association.

The next two subsections will go more in depth describing respectively the methods of generating local features and the algorithm for the fusion of the descriptor derived information.

4.3.2 Local Feature Description

In order to obtain a meaningful, fine grained, local description of the targets, many different approaches are possible. One of such approaches is based on the idea of generating a separate model for each of the body parts of the target (e.g. right arm, left arm, torso, right leg, left leg, etc.). To do so one needs first to locate and segment the body parts, then find a suitable way to describe them and finally perform the match operation against the corresponding descriptor of the already tracked target present in the history gallery (i.e. targets seen in previous frames).

Keeping a low computational cost for said set of operations is a very desirable fea-

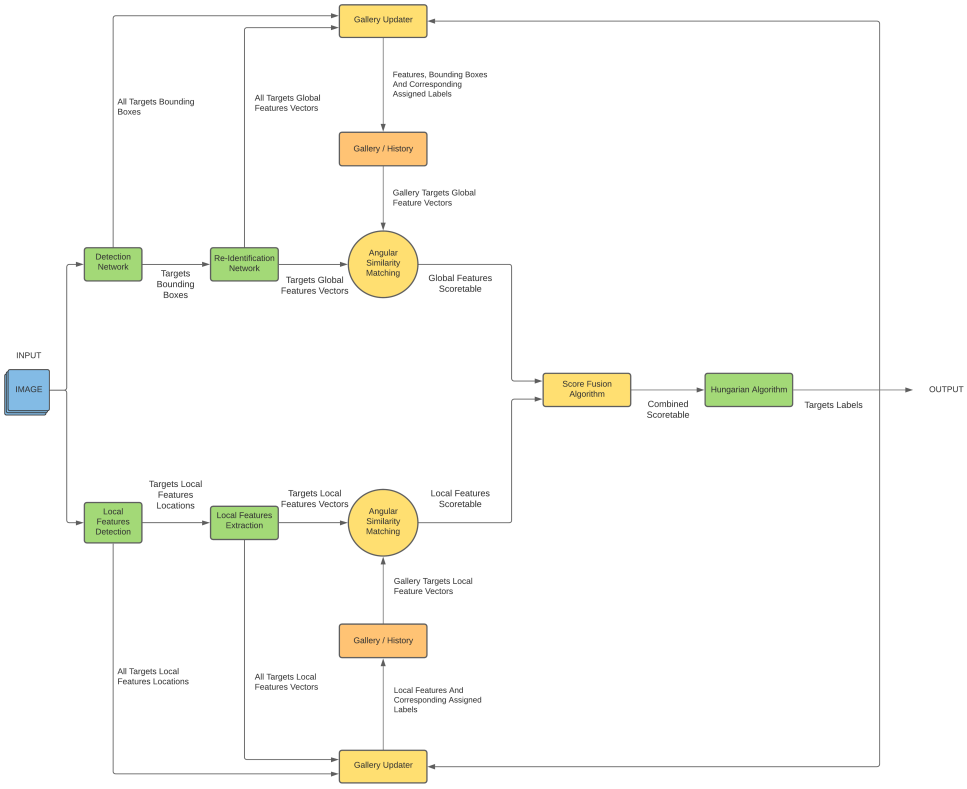

Fig. 4.5 Overview of an architecture that combines the *global* and *local* appearance descriptor scores in order to predict the labels (identities)

ture for an online tracker, in order to maximize the throughput of the system. Segmenting the bodies of the detected targets may easily become a burden in this sense, as one would need to design and train a segmentation network ad hoc (and possibly generate the necessary dataset too) and then another expensive network to generate the descriptors from the segmented areas, so it may be more practical to find an alternative to the description of full body parts.

One possible cheap solution for the extraction of local regions inside the targets bodies is to exploit the fact that many modern object detection network architectures can provide with no additional cost both the coordinates of each detected target, but also the landmarks corresponding to the joints of the skeleton (when the target is a human).

By using the landmark points to locate the features to extract, there is a double advantage: the landmarks are always inside the body of the target, so it is possible to extract features that are not influenced by background pixels noise, plus the landmark points are topologically marked, so that it is possible to confront the features extracted from a landmark of a target (e.g. the right shoulder) only with features extracted from the corresponding landmark on another target. This greatly reduces the overall number of confrontations necessary to perform the matches and avoids noisy matches.

As for the features to be extracted at such locations, the simplest thing is some type of patch description: by selecting suitable size patches centered on the detected skeleton landmarks there are many possible ways to encode the patch appearance information (texture or color or both) into some kind of feature vector (for instance using patch color hashing or a neural network descriptor).

The specific ways the patches were sized and described are presented in detail in section 4.4.

4.3.3 Score Fusion

When dealing with two independent types of description (local and global) for the appearance of each target, with the goal of making a correct label (identity) assignment during tracking, it is necessary to find a way to incorporate both descriptors information in the decision making.

Two main approaches are possible:

1. The information can be integrated at the descriptor level
2. The fusion can be performed at the score level

In order to fuse the descriptors in a meaningful way it is possible to create a neural network architecture that takes the local and global descriptors as input and outputs a unified descriptor which incorporates all the information.

The other possibility is to perform independent confrontations and matching for the local and global descriptions, obtaining two independent *score tables* and then use a deterministic or stochastic algorithm to fuse the scoretables into a unified one.

There are pros and cons for both approaches, so some consideration must be made. The fusion of the descriptors information can be obtained by a specifically designed and trained neural network, which with an appropriately designed dataset can in theory obtain good performances; the problem is that, given the way of extracting the local descriptors for a given target as proposed above, the number of visible skeleton landmarks for that target is influenced by external factors (such as motion blur, resolution and partial occlusion), leading to a descriptor with variable size (in the extreme case no landmark is visible for a given target). There is no easy way to design a non biased neural network with a variable input size.

If the information fusion is performed at the score level instead, it is possible to first combine all the local information of the confrontations for a given target in a single scoretable, and then find a way to fuse the unified local scores with the global scores computed independently.

The details of the specific custom designed score fusion algorithm used are given in section 4.4.

4.4 Workflow

As explained previously, this work is not intended to directly compete with the state of the art algorithms in terms of performance metrics, but the idea is to demonstrate the principle that with the help of local appearance description it is possible to boost the performances (in terms of number of identity switches) of those algorithms that solely rely on a global appearance descriptor.

Referring to figure 4.5 in this work Detectron2[45] has been chosen to perform the detection of the targets, as well as the extraction of all the visible skeleton landmarks for the targets. Regarding the global appearance description, different networks have been tested (see chapter 5).

The use of Kalman filter as a mean of guaranteeing the spatio-temporal coherence of the targets' trajectories has been added to the final workflow as it as proven through experiment to avoid mismatches of targets with similar appearance (see figure 4.6).

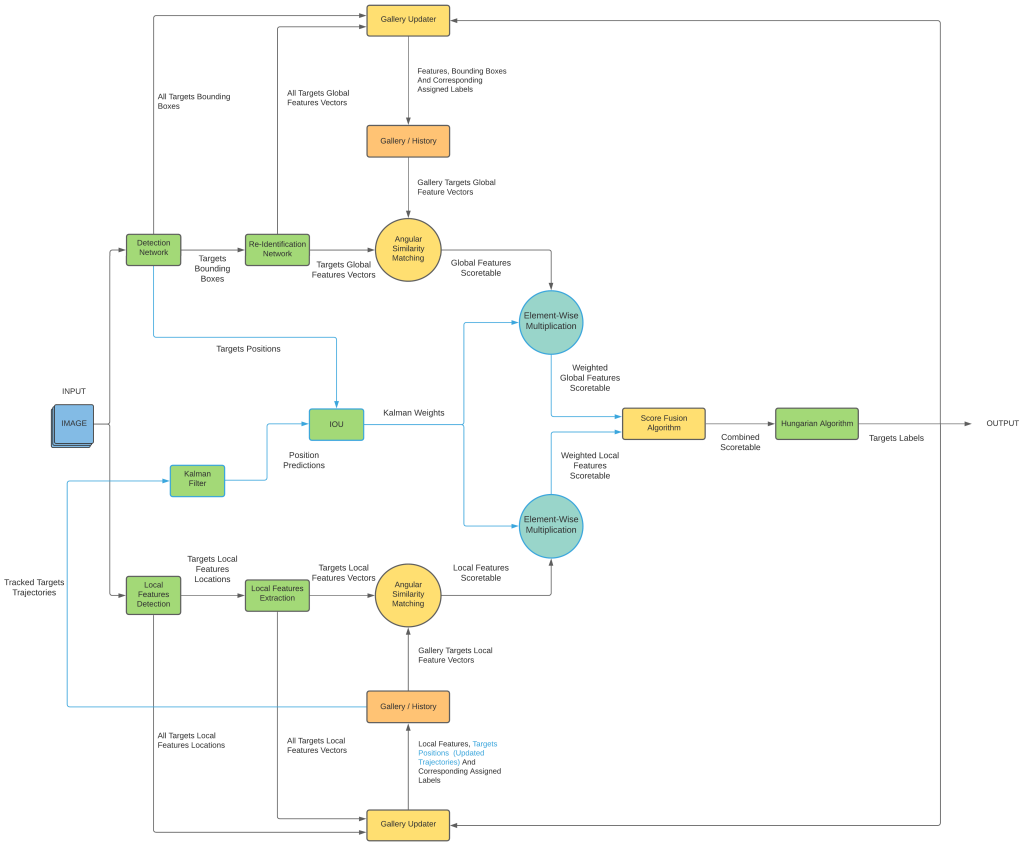

Fig. 4.6 The same architecture as in figure 4.5, but with the addition of the Kalman filter based validation system for guaranteeing spatio-temporal coherency of the trajectories

For the global description of the targets, different deep network architectures have been tested, with the intention of proving the generality of the proposed solution.

As for the local feature description, many different solution have been tested, but

all of them were based on the same principle: the features were extracted from small patches, each centered in one of the skeleton landmarks of the target. The dimension of such patches scales in accordance to the dimension of the target (see chapter 5) in order to achieve patch content coherence. Experiments showed landmarks situated in the head area of the targets to be unreliable, due to their similarity (typically the heads look similar, particularly when the resolution is low due to the sensor to target distance), even across different targets, while those extracted around landmarks situated on the torso, arms and legs are more likely to present variance in both color and texture for different targets. Regarding the description of the contents of the extracted patches the focus was on encoding either texture or color content of the patch (or possibly both), thus methods such as patch hashing, Delta-E based color content description and DNN-based descriptors have been tested.

The global and local appearance information of a specific target are then fused at the score level (which means they are first confronted independently), by means of a custom designed algorithm aimed at minimizing the combined local and global matching error rate.

It is to note that experimental results have proven that the global appearance model works better when the target is not occluded, and in such cases the local appearance model does not provide a positive contribution to the solution, but in fact can sometimes worsen the performances.

In order to solve this predicament, the strategy that has been devised is quite simple: at the moment of detection, each target is marked as either **dominant** or **overlapped**: the target is dominant if its bounding box does not intersect any other target's bounding box or if it is the closest to the camera among all the bounding boxes that overlap with it (i.e. its bounding box is the biggest), and is marked as overlapped otherwise. When deciding the weight of the contribution of the global and local descriptors for the score level fusion, the two cases (dominant and overlapped) are considered separately.

The implementation details of the workflow will be addressed in depth in chapter 5.

CHAPTER 5

Implementation

This chapter illustrates the implementation details of the workflow outlined in chapter 4. All the sections that follow will refer to figure 5.1. .

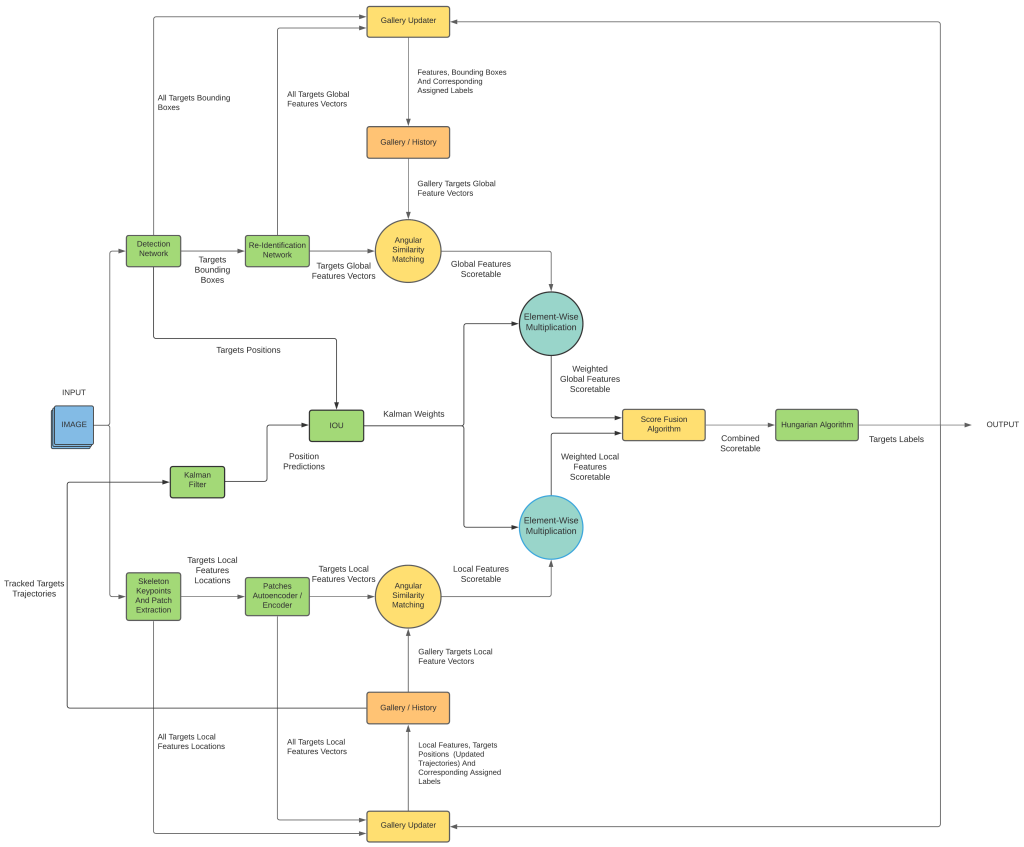

Fig. 5.1 An overview of the execution pipeline of the proposed system

5.1 Target ROI and Landmarks detection

As mentioned in 4.4, the chosen approach in order to extract both skeleton landmarks and regions of interest for all the targets in a single shot is **Detectron2**.

The official implementation by Facebook Research has been adopted and the pre-trained model and weights **R50-FPN** (model ID: 137849621) from the *COCO Person*

Keypoint Detection Baselines with Keypoint R-CNN was selected because it had the lowest inference time while keeping the accuracy of the detections and landmarks very high.

For the purpose of this thesis, among all the 80 different classes present in the MS COCO Dataset[28] only the *Person* class has been considered. In order to raise the selectivity of the network and avoid partial detection (that is detecting only part of a target that is in fact fully visible) or reflections and non human targets (such as mannequins inside shops), the value for the class confidence threshold has been raised to 0.70, which in MOT20 dataset sequences proved to be the best value (the default was 0.50).

5.2 Re-Identification Networks and Global Appearance Description

Regarding the *Re-Identification Network* block in the aforementioned figure 5.1, two different networks have been tested in the context of this application. Both of them have been deployed using the pre-trained models and weights provided in the selected implementations.

5.2.1 DG-Net

The first Re-Identification architecture tested in the system is a baseline method described by Zhedong Zheng et Al.[51], implemented using Pytorch[1], obtaining, with the model and weights used $Rank@1 = 87.74\%$ and $mAP = 69.46\%$ on the Market1501 dataset[50].

A brief description of the details of DG-Net can be found in section 3.9.2.

5.2.2 FD-Gan

The second Re-Identification architecture tested is FD-Gan[11], implemented using Pytorch[2].

The model used is *Market1501_baseline_model*, which is build upon a Resnet50 backbone, trained on the Market1501 dataset, reaching $mAP = 77.7\%$ and $Rank@1 = 90.5\%$ on the same.

The inner workings of FD-Gan are described in section 3.9.3.

5.3 Patches Extraction and Local Appearance Description

As already explained in Chapter 4, different methodologies to obtain local appearance descriptors for the parts of the targets have been tried in this work and all such methods have in common the fact that they describe the visual contents of patches extracted in

[1]Implementation used for this work available at: https://github.com/layumi/Person_reID_baseline_pytorch
[2]Implementation used for this work available at: https://github.com/yxgeee/FD-GAN

correspondence of the detected skeleton landmarks for the specific target (i.e. the center of each patch is one of the skeleton landmarks detected).

This choice is due to the fact that not only it allows the extraction of meaningful local appearance content for a given target, but also because it makes possible to label the content with their respective area of origin (since landmarks are labelled, we know from which specific area of the targets' body each patch was extracted), thus keeping a topological order of the patches.

This enables the algorithm to confront each patch only with patches representing the same area of the body, therefore reducing the number of total confrontation needed and avoiding cross-matching of appearance content from different body regions (meaning it cannot match, for instance, a patch from the shoulder with a parch from the knee).

Tests have shown that all landmarks situated on the head of the target are bound to produce unreliable matching results, due to the average resolution and scale of target in real world applications, as well as the presence of motion blur and the similarity of content in such patches for different targets. So for the purpose of this thesis, among all 17 landmarks extracted by Detectron2, the 5 on the head area are ignored, leaving only six point couples: shoulders, elbows, wrists, thighs, knees and ankles. These 12 areas present content that is potentially distinctive for each target in both texture and color range.

In order for the patches (which are square) to be extracted at the correct scale with respect to the dimension of the target, a procedure has been devised, based on polynomial fitting: for each different typology of patches (meaning separately for each couple such as left/right shoulder and left/right elbow), the size is computed by regressing a second degree polynomial, associating the area of the bounding box of the target to the desired length of the side of the patch; this was done on a custom, purposely designed dataset. The dataset has been produced by using Detectron2 generated bounding boxes and landmarks and hand designed patches centered in such landmarks for 600 target at different scale from images coming from MOT17 and MOT20 sequences which are not used for evaluating the system in the testing phase (detailed information about this custom dataset can be found in section 6.2.1).

Regarding the description of the patches content, three different strategies were tested. The first is image hashing applied to the patches, in particular color hashing, as it considers both color and texture of the patch. Since, as seen in section 3.3, image hashing has been used as a method for image retrieval, it was a candidate in order to confront the appearance of different patches; unfortunately the results were poor, due both to the reduced dimension of the patches, as well as the difficulty to reliably model (also in the case of color hashing) the colors in the patches.

Fig. 5.2 Examples of the application of the automatic patches extraction: each patch is centered around the visible skeleton landmarks, and is scaled accordingly to the procedure described above, which maximizes the probability that all the pixels of the patch belongs to the target and not the background, while still retaining the largest amount of target pixels as possible

The second one is based on the Delta-E color distance metric: each patch is resized to a standard dimension of 32×32 pixels and then converted to CIELab color space, as it is one of the closest to be perceptually uniform, therefore the most suitable for describing color differences from the human point of view. Then the confrontation between two targets is done keeping the topological coherence (only between patches representing the same area in the two targets) via the following procedure: the two patches are confronted by computing the Delta-E pixel-wise, then the average of the results of such confrontation is considered as a measure of the color difference between the two patches. The difference between two targets is then represented as a single value, given by the average of the patch-wise Delta-E distance (just described above) of all the visible patches of said targets. This solution does not capture differences and similarity in the textures of the patches, yet still considers not only the average color difference between two patches, but since it is done first in a pixel-wise fashion, it also takes into account the similarity in the distribution of the colors of the confronted patches. This method has proven to bring marginal performance improvements in the testing phase with respect to the standard tracking by detection method using just the global descriptor for the targets (see chapter 7).

The third strategy is based on a deep neural architecture: a model of a patch is created by training a Convolutional Auto Encoder (CAE) over patches extracted from the same custom dataset used for the polynomial fitting (refer to 6.2.1). It is not a new idea to use a CAE to extract a compressed representation of patches in order to confront the visual (appearance) similarity between them[26]. The patches are again resized to a standard dimension of 32×32 pixels. The CAE is trained to reproduce the patch itself, so, by exploiting the structure of the architecture, the encoder part of

the trained network will contain a compressed representation of the content of the patch itself, encoded in a numerical vector, which can be considered a descriptor for both the texture and color content. The compression factor obtained at the end of the encoder in the final architecture is $12\times$. This compression factor was chosen by trying to filter out as much noise as possible while still retaining as much useful information as possible. Given the dimension of the patches and their level of content, after testing various layer sizes, numbers and configurations, the chosen architecture is shown in figure 5.3.

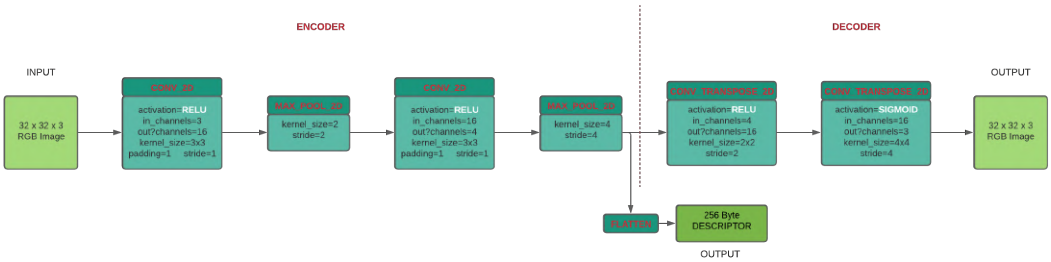

Fig. 5.3 The architecture of the autoencoder. The pooling in the encoder part is designed to generate a compression of $12\times$ with respect of the size of the original data.

When confronting two patches, the similarity measure used is the cosine similarity between the two descriptors. Again the similarity of two targets is computed as the average distance for each visible patch across the two targets considered. This solution is the one that selected among all three as it has provided the best performance improvements respect the global-features-only tracking algorithm.

5.4 Kalman Filters and Spatial-Temporal Matching Coherence

When dealing with data coming from a static camera, it is possible to take advantage of the fact that there is no relative camera motion in order to predict with a certain degree of accuracy the position of each target in the next frame.

By exploiting this fact the mismatch of different targets can be avoided in many cases even if they have a similar appearance. In this way it becomes possible guaranteeing a certain degree of coherence of the targets' tracks at a spatial and temporal level, reducing the tracking noise.

Such feature, in the context of static camera footage, is easily achievable by applying well known algorithms such as Multiple hypothesis tracker (MHT), Joint Probabilistic Data Association Filter (JPDAF) or Alpha Beta Filter (ABF) and Kalman Filter (KF).

In this direction, different solutions have been tested. The first one was based on the use of Alpha Beta Filters, but it was quickly discarded due to the lack of robustness in the presence of missing observations for the state of a given target, which made the prediction rapidly drift away from the actual position of the target.

The second solution was instead based on Kalman Filters: a separate KF was instantiated for each target; at each frame, the detected bounding boxes are confronted via IOU with the bounding boxes of each tracked target, projected to the predicted (by the KFs) positions, thus obtaining a table of scores, with each score representing the likelihood of each target possible association in the current frame in terms of position. This approach is very similar to that of SORT[4]. The scoretable is then sent to the scores fusion algorithm described in Section 5.5.

This solution was in the end also discarded in favor of a third alternative, which was experimentally more robust: the KFs are instantiated and used in a similar fashion as previously described, but instead of producing an independent score table, they are used to produce a table of weights with just two possible values (0 and 1). In the IOU confrontation, carried on in the same way as described for the second method, if the value of the IOU of the two bounding boxes is greater than zero, the weight is set to 1, otherwise is set to 0.

The weight table is then multiplied element-wise with the scoretables generated by both global and local descriptor confrontations, thus forcing to zero the similarity scores of those confrontations deemed impossible on a spatial-temporal level by the Kalman Filter. The two newly weighted scoretables obtained can then proceed to the score level fusion. This can be viewed as a relaxation of the constraint of the minimum viable IOU value in order for a possible match to be considered, thus making up for some of the detection inaccuracies. In practice the Kalman filters are used only to validate the possibility of an identity confrontation, the possible confrontations are performed on the basis of the appearance descriptors.

As a side note, the reason why the third method outperformed the second method is certainly influenced by the fact that the score level fusion works well only when all the algorithms in the ensemble provide similar performances, while in the case of the aforementioned second method, the Kalman filters scores proved to be way less reliable than both the global and local appearance scores. By using the Kalman filter as a spatio-temporal coherency validation method for the target positions with respect to the trajectory, the information is still used, without biasing the score fusion algorithm.

5.5 Score-Level Appearance Information Fusion

There are many possible strategies to use both the global and local appearance information as well as the information obtained by the Kalman Filters. As mentioned before, one way is to perform fusion at the descriptors level, obtaining a single descriptor with both local and global information and then perform a single confrontation between each

couple of targets.

In the case of the local descriptors selected for this work, this type of solution would have proven difficult, since the local descriptor, in the case of a detected target with no landmark clearly visible (i.e. due to visual obstruction), may be nonexistent. Furthermore, the local appearance confrontation score between two targets can be nonexistent in another type of situation: None of the landmarks visible in the first target is visible in the second target, leading to the impossibility of the confrontation.

Thus the logical conclusion is to evaluate the similarity of two targets' appearance models independently for the global and local descriptions at first, obtaining two separate similarity scores for each couple of targets, and then fusing the obtained similarity scores in a unique score for the same confrontation.

To address the problem of the possibility of nonexistent local descriptors, the solution of choice is a convex combination of the local and global scores. As explained in section 3.7, this is the basic principle behind the weighted averaging. Convex combination is a weighted sum where all coefficients are non-negative and sum to one. This allows solving also for the extreme case where the weight of the local appearance confrontation score (when no landmarks were visible for at least one of the targets) can be set to zero, while the weight for the global can be set to one.

In this case the system needs two weights: one for the global descriptor and one for the local descriptor.

In order to understand the way the specific weights for the convex combination were selected it is first necessary to explain the context in which they were selected. Since avoiding biases was a necessity, the Kalman filters output were disabled during the process (i.e. every confrontation was deemed valid independently of the spatial coherence). This has made possible performing the selection of the convex combination weights on sequences with the moving camera, so this was done on the training sequences from MOT17 dataset (see 6.2.2). As explained at the end of chapter 4, it is to be noted that the weights of the convex combination are calculated independently for the two possible cases, where the target in question is dominant or overlapped. The procedure used to determine the weight for the dominant and overlapped cases is the same, but is carried out separately.

In order to generate the training data needed to select the optimal weights, the system was first used to produce outputs (both targets identities and corresponding associated score) for global and local description independently. The identity association for this purpose was done using a *naive* strategy: for each detection in the current frame, the identity was associated in an independent manner from all other detection in the same frame, by just picking the maximum similarity score in the corresponding row of the scoretable (i.e. the assignment problem at each frame is treated as multiple instances of one-to-many assignment, instead of a single many-to-many assignment).

The identities for the history view are updated as described in section 5.7 with the corresponding ground truth label value. This update strategy allows for duplicate identities and is the product of a one-to-many confrontation, rather than a many-to-many one. This approach is necessary in order to evaluate the raw performances of the global and local descriptors without external influences from Kalman filters or the complexity of the scene: this way what is being assessed is the capability of the global descriptor to match the past global descriptor of the same target according to the ground truth identity, and the same is done for the local descriptor.

With these outputs saved, they are checked against the ground truth, and all possible combination of weights for the weighted average of the scores, at 0.01 intervals; then the combination that produces the minimum number of errors is selected. Along that, another hyperparameter is selected in the operation: a threshold on the weighted average value for deciding whether the target need to be considered a match with an existing target, or a new identity needs to be created.

As already stated, this procedure is performed separately for target marked as dominant and for target marked as overlapped, thus at inference time each row of the scoretables will be weighted by the corresponding value depending on the status (dominant or overlapped) of the target under consideration.

5.6 Identity Association

For the identity association task (as in figure 5.1) the selected strategy was simply the application of the Hungarian algorithm on the single scoretable resulting from the fusion of the global and local scoretables. This choice is due to the fact that the Hungarian algorithm ensures that each associated label (or identity) at a given frame is unique, that is no identity duplicate is possible with this strategy.

Once the identities for all the detected targets of a given frame have been associated, the system can now update the trajectories and the global and local models for all those target in the **history view**, which will be described in the next section.

5.7 Target Incremental Model And Update Strategies

The **history view** (or HV) is the name assigned to a data structure that contains, for all the individual target detected in all the frames before the current one, all the necessary data to perform the identity association task.

This means that it contains, for each target:

- the **identity** of the target, which is represented by a numerical label
- the **trajectory** of the target (composed by couples of positions at each frame and corresponding frame number)

- the **global descriptor** of the target
- the information to create the **local descriptor** of the target
- the target's associated **Kalman filter**

All the information on the single target in the list above will be referred to as the target's model. At the end of the inference step for each elaborated frame, once the identities for each current target have been associated, a procedure is called, whose task is to save the targets' models to the history view. At this point a distinction is necessary between two cases:

1. Target model initialization
2. Target model update

The initialization occurs when the system decides that it is seeing that particular target for the first time, so a new identity need to be created. In the first frame all detected targets go through this procedure; the other condition when this happens is when the selected identity association for a specific target detected in the frame done by the Hungarian algorithm has a score[3] which is below a certain threshold. The value of this threshold is selected alongside the weights of the convex combination for the weighted averaging, and it is the value which minimizes the number of wrong identity associations over the training data.

The target model update occurs only on already initialized targets (targets already seen in previous frames), when the Hungarian algorithm associates an identity for the target in question and the corresponding fused score is above the aforementioned threshold. It is important to note that not all the components of the history view of a specific target are updated at every frame.

In fact the global model of the target is updated only in two cases: if the target is dominant on its overlap group, or if its visibility has improved with respect to the frame when its global model was last updated.

Regarding the local model, it follows an incremental update strategy: each time the target in question is matched and marked to be updated, all visible local patches of the target are extracted; if no patch corresponding to the specific skeleton landmark in question is present in the history view of the target, the patch is added to the pool for the particular landmark of the target. Each target has a pool of a maximum of n instances (or patches) for each of the 12 possible landmarks, so if the target has already present in its history view patches for a specific landmark in question, they are updated

[3]The score in question is the result of the score level fusion step.

with a FIFO strategy. In this way the history view always stores for each landmark of the target up to the last n appearances of the corresponding patch; at each new frame, in order to perform the confrontations, a local descriptor has to be produced, so a new patch has to be generated for each landmark, which should contain the information accumulated in all the sightings registered. To do so, all the patches of the pool of a particular landmark of the target are modelled using a mixture of gaussians (MOG) approximation, producing a single patch, with informations representative of all the sightings in the pool. Such patch is then used to extract the local features by using either the encoder part of the autoencoder as previously described, or the Delta-E based information. The value of n has been chosen experimentally and was set at 20.

As stated before, to each target in the history view is associated a Kalman filter. Every time a target in the HV is matched to a detection, the new position is used to update the KF. If a target in the HV is not matched in a particular frame, the filter has to deal with a *missing observation*. The update strategy is designed such that after m consecutive missing observations, the target is marked as *dead*. The targets that are marked as dead are moved from the **alive person pool** to the **dead person pool**. The choice of m was empirically set to 30 frames because experiments on the dataset have shown that over that threshold the KF prediction drifts too much from the actual position of the target.

It should be noted that this value is arbitrary, based solely on the performances of the KF when dealing with missing observations, and was not decided via an optimization procedure in order to avoid biasing the value for either one of the implemented strategies (i.e. only global features, or global plus local features strategies).

A target is also marked as dead when it goes out of frame. The dead person pool is kept for a possible multi-camera expansion of the system.

CHAPTER 6

Datasets

This chapter contains the description of all the datasets used throughout this work.

The MOT Datasets are a core component of this thesis, because they are used in two different ways: the first one to select the hyperparameters of the system, the second one to define custom made datasets based on the sequences of MOT Datasets that are used to train specific part of the final pipeline.

The next section will be dedicated to the MOT challenge datasets and their metrics, while 6.2 illustrates custom made dataset for training the specific sub-tasks related to the introduction of the local appearance model, as well as some hyperparameters of the algorithm.

6.1 The MOT Datasets

This section will present the most widely used datasets for evaluation of Multiple Object Tracking algorithms, especially focused on pedestrians tracking: the *MOT Challenge* datasets.

At the moment this thesis is written, the MOT Challenge comprises the 2DMOT15[25], the MOT16 and MOT17[29] and the MOT20[9] datasets. Each dataset is a collection of short sequences, associated with a *ground truth* (*GT*) file, which contains information about the location and the identity of each target, among many other data.

Since the most used among all the datasets is the MOT16 dataset, and given the fact that all the datasets listed above use similar metrics for the evaluation [1], the following section will describe in synthesis all there is to know about MOT16.

6.1.1 MOT16 Dataset

The MOT16 dataset[29] is still today the most used dataset for evaluating multi-person tracking algorithms. The goal of the dataset is to present a fair benchmark for the evaluation of multi-target tracking methods, bringing forward the strengths and weaknesses of state-of-the-art multi-target tracking methods.

[1] All the algorithms presented in 2 are evaluated on the MOT16 dataset.

The benchmark is composed by 14 sequences, 7 train sequences and 7 test sequences; each of them has an associated file containing the detection ground truth (with coordinates of the bounding boxes), but only the train sequences have an associated ground truth file containing the labels for tracking. This was done in order to avoid biasing the results of the tests by making impossible to train networks or tune parameter directly on the testing data.

The authors provide standardized ground truth data, evaluation metrics and scripts so that all methods are compared under the exact same conditions. Among the metrics defined there are:

- TP (true positive): an actual annotated target (the IOU between the detection and the GT of the target must be > 0.5)
- FP (false positive): a false allarm
- FN (false negative): a missed target
- IDSW (identity switch): a mismatch error between the algorithm annotation an that of the GT
- MOTA (Multi-Object Tracking Accuracy): is the combination of three sources of errors, and is defined as:
  $MOTA = \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t}$
  where t is the frame index and GT is the number of ground truth objects
- MOTP (Multi-Object Tracking Precision): is the average dissimilarity between all true positives and their corresponding ground truth targets, defined as:
  $MOTP = \frac{\sum_{t,i}(d_{t,i})}{\sum_t c_t}$
  where c_t is the number of matches in frame t and $d_{t,i}$ is the bounding box overlap of target i with its assigned ground truth object
- MT (Mostly Tracked): a target successfully tracked for at least 80% of its life span
- PT (Partially Tracked): a target successfully tracked for between 20% and 80% of its life span
- ML (Mostly Lost): a target successfully tracked for less than 20% of its life span
- IDF1: The ratio of correctly identified detections over the average number of GT and computed detections
- PRECISION: Ratio of $TP/(TP + FP)$
- RECALL: Ratio of correct detections to total number of GT boxes

6.2 Custom Datasets

Since sequences from MOT15 and MOT16 have been used for evaluating the system, for the sake of avoiding the introduction of biases, all the training and hyperparameters selection tasks have been performed on sequences coming from MOT17 and MOT20.

The reason for this choice is the fact that the use of the Kalman filters in the elaboration pipeline assumes sequences taken from a static camera; in fact, without introducing complications such as integrating SLAM (Simultaneous Localization And Mapping) algorithms into the workflow, it is impossible to make use of the Kalman position predictions for the targets in order to guarantee a certain degree of spatial coherence in the detections.

Given the fact that the majority of static camera sequences are found in MOT15 and MOT16, they have been selected for the evaluation of the system, while MOT20 contains only moving camera sequences, thus it has been relegated to training and hyperparameter selection tasks, which do not require the use of Kalman filters.

6.2.1 Local Features Extraction and Description

The first task requiring the use of a training dataset is the target's local features extraction and description task. This task is explained in detail in chapters 4 and 5, while this section will outline the structure of the dataset used for training the sub-parts of the system that will perform those tasks.

Using the sequences MOT20-03, MOT20-01, MOT17-05-SDP, MOT17-08-SDP and MOT17-10-FRCNN as a basis, a total of 120 random frames from each (equally distributed) have been selected. The choice of the specific sequences was done in order to obtain the greatest variability in the targets' dimension with respect to the point of view.

For the extraction task the idea is to retrieve some patches from the target in correspondence of the skeleton landmarks for shoulders, elbows, wrists, hips, knees and ankles. Each frame was first processed by Detectron2 in order to locate all the targets and the skeleton landmarks, then square patches centered around each landmark were selected by hand with a dimension such that each patch would cover the most number of pixels of the target without including any pixel outside the target silhouette. This new dataset is used to fit a second degree polynomial that gives the length of the side of each patch with respect to the area of the bounding box of the detected target.

For the description task the goal is to have a number of correctly sized patches to train a small autoencoder to reproduce the extracted patches. In order to guarantee the greatest varability in the training patches appearance, the extraction algorithm, has been run on a total of 1600 random frames coming from all the sequences in MOT20 and from MOT17-05-SDP, MOT17-07, MOT17-08-SDP, MOT17-10-SDP, MOT17-11-

SDP, and MOT17-12-SDP, and all the patches extracted from these frames have been used to train the autoencoder. It is to be noted that the autoencoder is an unsupervised learning structure, which does not require any ground truth or labelling procedure on the training data, but simply learns to reproduce in output what it has been given as input.

6.2.2 Score Level Fusion

The selection of the weights for the averaging procedure of the global and local score fusion described in section 5.5 was done on the sequences MOT17-05-SDP, MOT17-10-SDP and MOT17-11-SDP. The ground truth used for the task was the original one provided by the dataset (for each target identity the bounding boxes and all the information on his trajectory), no custom annotation was done to this purpose.

CHAPTER 7

Results

Before presenting the results it is necessary to address the metrics used to evaluate the task and their meaning. There are a lot of different metrics used to rank multiple-object-tracking tasks, but generally the most commonly used based on the number of True Positives (TP), False Positives (FP) and False Negatives (FN) identity association for the targets. Other commonly used metrics use as well as the number of Identity Switches (IdSw) too. The Multi-Object Tracking Accuracy (MOTA) and the Multi-Object Tracking Precision (MOTP) are two example of widespread ranking approaches, whose description can be found in Chapter 6. It is common practice to consider the MOTA index as the most meaningful measurement for describing the performances of a MOT algorithm, since it combines the three principal sources of errors: the number of FP, FN and IdSw in a single index.

Unfortunately the MOTA is not well suited for the evaluation of the improvement achieved by the introduction of the local model in the tracking-by-detection paradigm.

In fact, the number of TP, FP and FN are mostly related to the performances of the detection algorithm which in this experiment, is the same (Detectron2), with the same parameters for both the methods (Only Global Model and Global+Local Model).

The emphasis in this thesis was to evaluate the improvements related to the IdSw that can be obtainead by mixing global and local descriptors.

The number of IdSw mainly depends only on the capability of the system of associating the correct identity to the targets, independently from the detection performances.

This means that the MOTA metrics will be less perturbed by changes in the number of IdSw than by changes in the detection performances.

Being the goal of this thesis (as previously stated) that of evaluating the improvements coming by the combined use of local and global approaches during the MOT, the most meaningful metrics to consider is plainly the number of identity switches. In any case the next section will nonetheless present the full set of metrics suggested in the MOT Challenge for all the different versions of the proposed system as described in Chapter 6. It is necessary to remember that the goal of this work was not to obtain the best absolute results in terms of MOTA or another parameter, but to prove that carefully extracted features describing local areas of the targets can help improve the system by

reducing the number of IdSw for the cases of partial occlusion of the targets, leading to less noisy tracks.

The performance values showed in the next sections were obtained on a standard desktop with Intel i9-9900K CPU, 32GB DDR4 RAM and NVidia GTX 2080Ti GPU with 11GB of memory. It should be noted that the processes were not memory-bound, but were computationally intensive on the GPU. The system execution pipeline was not optimized by splitting the sub-tasks in multiple threads, but all the operations on a frame were done sequentially; this was done in order to avoid the non-deterministic nature of multi-threading systems.

Before looking at the overall performances of the tracking system it could be useful to analyze the performances of the single sub-parts of the system separately.

7.1 Detection performances

As explained before, throughout this work the detection framework used was the official implementation of Detectron2, using the pretrained model ResNet50-FPN from the model zoo (model ID 137849621), trained on Microsoft COCO for both the detection (bounding boxes) and the keypoints (skeleton landmarks). The Facebook AI Research team declares for this model an inference time of $0.066s/im$, a box average precision of 55.4% and a keypoint average precision of 65.5%.

It is to be noted that in my implementation every hyperparameter was left at its default value, with the exception of the confidence threshold for the detection which was raised from 0.50 to 0.70 (this was done in order to increase the rejection rate for false targets). In fact this increase in the value of the confidence threshold had a great impact in reducing the number of false positive detections. False positive detections are ROIs detected by the system even if they are not present in the ground truth; as an example the reflections on showcase glass of an actual target or a mannequin could be detected as targets by Detectron but they should not have been detected. This false positives negatively affect the metrics that depend on them (i.e. the MOTA), but they also increase the probability of identity switches. This is due to the fact that the false positive detections can have a very similar appearance to that of an actual target, and there is the risk that the position may coincidentally be validated by the Kalman filter as spatio-temporal coherent, especially when the target is close to the reflection surface.

7.2 DG-Net and FD-Gan performances

Regarding the Re-Identification networks, both FD-Gan and DG-Net were used with the pretrained weights made available in the repositories used (see 5.2.2 and 5.2.1). Since no changes were made to the pretrained models, this section will report the performance

analysis values directly from the original papers, in particular the mean average precision (mAP) and the $Rank@1$ accuracy for the Market1501 and the DukeMTMC-ReID datasets:

Framework	**Market1501**	**DukeMTMC-ReID**
DG-Net	mAP=94.8, Rank@1=86.0	mAP=86.6, Rank@1=74.8
FD-Dan	mAP=77.7, Rank@1=90.5	mAP=64.5, Rank@1=80.0

Table 7.1 Performances of DG-Net and FD-Gan as reported by the respective publications

In order to have a meaningful measurement for the inference time of the two architectures, the values were calculated based on the execution on the actual hardware (described above) utilized in evaluating the whole tracking system. DG-Net showed an average inference time of $0,007$ seconds per target, while FD-Gan $0,0024$ seconds per target. This means that it is possible for DG-Net to extract the global appearance model of 143 targets per second, and for FD-Gan as many as 416 targets per second. So the two network could run real-time @30fps (frames per second), if the average number of targets in the frame does not exceed 4.7 and 13.8 respectively. This calculation, however, only takes into account the global features extraction step, and not the time needed for the confrontation with the features of the target in the history view and the execution time for the Hungarian algorithm, as a result the reported values do not represent the total inference time for the processing of a frame.

Figure 7.1 illustrates the weakness of tracking by detection systems relying solely on global features to model the appearance of the targets (the labelling in the pictures was obtained using DG-Net for the extraction of global descriptors).It is apparent that due to the occlusions the target loses its tag and a new one is erroneously attached to it (i.e. an IdSw occurs). For comparison, figure 7.4 shows the improvements introduced by using both the global (again by DG-Net) and CAE-based local descriptors in the same exact conditions.

Figure 7.2 illustrates the changes on the descriptors of two target during a partial overlap of the two, with respect to when they are not overlapping. From the figure it can be seen that the occluded target in particular is seriously affected by the overlap, as its global appearance descriptor has suffered heavy changes, leading to a poor match with the correct gallery target, leading to an identity switch. Note that the aforementioned figure contains visual representations of the descriptors, and for ease of visualization they have been scaled so that the differences and similarities are noticeable.

Fig. 7.1 A sequence of (cropped) images from the test set, with labels assigned by the system using only the global features (with DG-Net). Note that target 3 (bounding box in pink), during the partial occlusion suffers from mislabelling, caused by an identity switch.

7.3 Performances of the CAE-based local features

This section illustrates the results of the local features extracted by the autoencoder whose architecture is reported in figure 5.3.

Since the feature quality is dependent on the correct extraction of the patches, it is useful to first evaluate the performances of the automated patch extraction system. As a first thing, the patches are centered on visible skeleton landmarks, which are reported to have an average precision of 65.5% for the model used. In order to have a more complete view, the IOU (intersection over union) of the automatically extracted patches and the corresponding ground truth hand-drawn patches was calculated on the test set of the custom-created patches dataset, obtaining a mean value of 0.86.

Before reporting the Rank@1 metrics values for the local descriptors over the test set, it can be useful to look at figure 7.3 to have an idea of how the implemented autoencoder architecture is able to reconstruct the contents of a patch. Note that even if the hue or the saturation in the reconstructed patches are visually different with respect to the original inputs, the reconstructed patches are nonetheless quite similar among them. On the other hand, for the re-identification of the target what matters is the similarity between the descriptions extracted rather than their fidelity to the original patches. Confirming this, the quantitative analysis of the obtained matches shows that the descriptors are good in discriminating between the different patches. During the early stage of the autoencoder implementation, the fidelity of the reconstruction to the original patch has been pursued by using different color spaces, such as HSV and RGB, in the hope to find

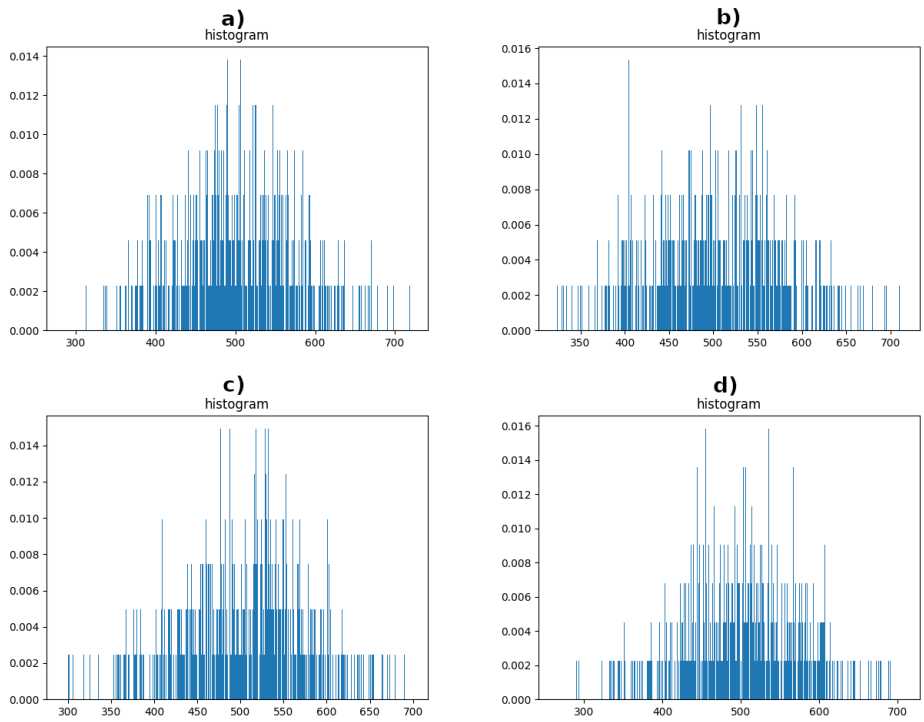

Fig. 7.2 Overlapping influence on the descriptors, referring to the scene in figure 4.3. a) target $\#3$ when not occluded, b) target $\#3$ during partial occlusion, when the ID-switch happens (mislabelled as $\#4$), c) target $\#1$ when not occluded, d) target $\#1$ during the same partial occlusion described for b). The descriptors are extracted with DG-Net. The descriptors are depicted as histograms, where each bin in the horizontal axis represent a position in the tensor, and the values int the vertical axis have been scaled (multiplied by a constant) in order to make the visual interpretation easier

a better representation. Unfortunately all these efforts proved to be ineffective, resulting in worse results, both visually and quantitatively.

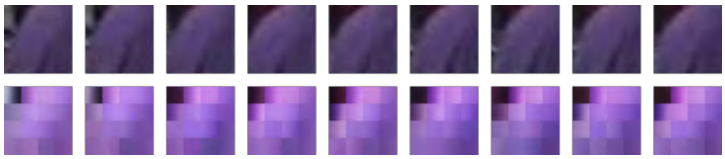

Fig. 7.3 The top row shows a sequence of skeleton landmarks patches extracted from the same target consecutive consecutive in the test set. The bottom row illustrates the corresponding reconstructions done by the autoencoder.

For sake of completeness, the performances of the descriptors generated by the autoencoder have also been evaluated independently from the system, by using the $Rank@1$ metrics. On the test set (as described in chapter 6), the autoencoder with the final architecture and weights, achieved a $Rank@1 = 83\%$.

For contrast, the Delta-E-based local appearance description method showed a $Rank@1 =$

59% on the same test set.

In [52], the author remarks the fact that score level fusion with averaging does perform well only if all the learners of the ensemble achieve similar performances individually. This means that, since FD-Gan and DG-Net achieved $Rank@1$ performances between 74.8 and 90.5 (see table 7.1), the Delta-E-based method is not suitable for the score fusion, while the CAE-based approach is a good choice.

7.4 Computational Cost associated with the Assignment Problem

For the assignment problem, the chosen solution was to run the Hungarian algorithm on the unified scoretable (combination of the global and local scoretables, validated by the Kalman filters as explained in chapter 5).

As stated in chapter 3, the Hungarian algorithm performs the best assignment by maximizing the total assignment score for the current frame, but its complexity is $O(n^4)$ ($O(n^3)$ in the implementation used in this thesis). Since the algorithm assumes a square $n \times n$ matrix, while the scoretables are rectangular $n \times m$ matrices, the so called padding technique is used to reshape the original scoretable.

In the whole test dataset used (see chapter 6), the average number of targets present in a single frame changes from as few as 4 to more than 50, depending on the sequence. The number of targets present in the gallery (history view) can be much larger, especially for scenes where the flow of targets is fast and constant and many targets enter and leave the frame quickly. In the test set the peak is 136 alive (see section 5.7) targets present in the history view, and in real applications can be much more.

This means that the execution time for the frame can greatly vary depending on the number of targets present both in the frame and in the history view.

In tracking application throughput is usually less important than accuracy, and this is the reason Hungarian algorithm was chosen for the assignment task instead of a sub-optimal heuristic solution.

7.5 Performances on the MOT Challenge

As explained before, all the version of the tracking system presented rely on the use of Kalman filters to obtain tracks that are spatially and temporally coherent. This means that the system could not be tested on the sequences of the MOT Challenge that are not taken from a static camera. In order to maximize the dimension of the set, all the static sequences from both MOT15 and MOT16 dataset were used for the evaluation.

Figure 7.4 illustrates the results of the labelling strategy for both the global (by DG-Net) and CAE-based local descriptors on a sequence from the MOT16 dataset. It is easy to see that the use of the local features reduces significantly the number of IdSw in case

Fig. 7.4 The same sequence of images as in figure 7.1, with labels assigned by the system using the combination of local and global features. Note that target 3 (bounding box in light green), does not suffers from mislabelling in the same conditions.

of partial occlusions between targets.

Architecture	Precn	Recall	MT	PT	ML	FP	FN	**IDsw**	MOTA	MOTP	Hz
FD-Gan Only	92.8	69.3	60	75	53	3818	36667	5838	57.2	77.1	5.9
DG-Net Only	92.7	69.2	59	76	53	3810	36796	7204	55.0	77.0	5.7
FD-Gan+DeltaE	92.7	69.3	58	75	53	3817	36676	9766	52.7	76.8	5.6
DG-Net+DeltaE	92.7	69.2	60	74	53	3823	36682	10339	51.1	77.1	5.3
FD-Gan+CAE	92.7	69.2	58	76	53	3810	36669	**714**	63.5	77.0	5.3
DG-Net+CAE	92.6	69.2	57	75	53	3823	36682	1517	60.5	77.0	5.1

Table 7.2 Performances of all the proposed tracking algorithm on the test set (described in chapter 6) according to the metrics in the official API of the MOT Challenge

Table 7.2 illustrates the performance evaluation according to the official MOT Challenge API over the test set as described in chapter 6.

7.6 Discussion

As stated at the beginning of this chapter, the ID-switches metrics is the most relevant one for illustrating the improvements due to the introduction of the local features and score level fusion with respect to the use of the global features only.

The influence that ID-switches have on the overall performances of the tracking system is manifold:

1. the number of ID-switches has a direct impact on the MOTA value

2. The fewer the number of ID-switches, the less noisy (fragmented) are the tracks; this is useful in one eventual post-processing step devoted to link the tracklets into complete trajectories, or something similar to what's done in [1] (see chapter 2), In fact, having fewer better, less fragmented tracklets would make the linking task easier.

3. Reducing the number of ID-switches improves the performances and usefulness of the Kalman filter spatio-temporal coherence validation. This happens because each mislabelling results in a possible missing observation of the correct label and the possible generation of a new label (and the related Kalman filter).

In fact, every time a new label is wrongly introduced, a new Kalman filter is generated, which in turn generates a prediction for the position at the next frame, thus propagating the error to successive frames.

Even if the ID-switch does not generate a new label, but instead an existing identity is mismatched and assigned to the wrong target the mislabelling can be propagated to the subsequent frames.

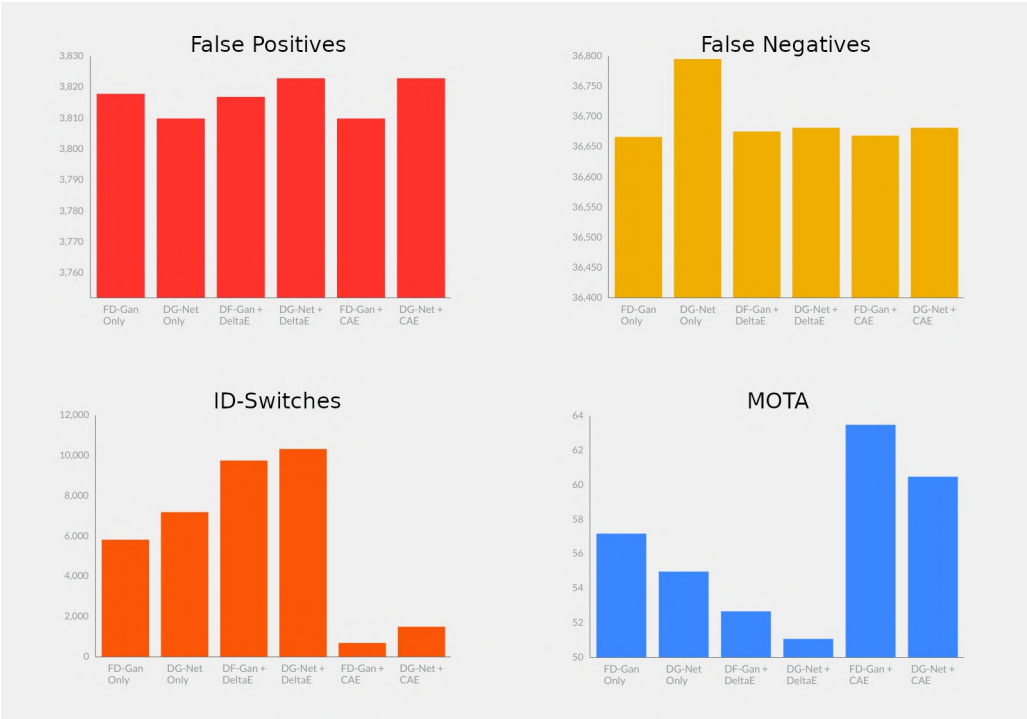

Fig. 7.5 Comparison of metrics between the six confronted architectures with reference to table 7.2. The false negative and false positive values are more or less constant between the different architectures, while MOTA and in particular ID-Switches are hugely impacted by the introduction of local appearance features (positively for the CAE-based, negatively for the Delta-E-based)

In table 7.2, by looking at the *IDsw* column, it is possible to see that the average Delta-E difference between query and gallery targets is not a representative measurement for the local appearance of the targets, as it increases the total number of identity switches instead of decreasing it. It is evident that the number of IDsw greatly increases with respect to the results obtained by using solely the global appearance model. This is why we can conclude that the Delta-E-based descriptor introduces too much noise in the description and significantly reduces the MOTA score.

This is partly due to the fact that, as explained in section 7.3, the low value on the $Rank@1$ metrics makes this approach unsuitable for the score fusion algorithm, if compared to the more accurate DG-Net and FD-Gan.

The features extracted by the CAE-based system on the patches instead, managed to reduce the total number of ID-switches by an average of 83.4%.

Furthermore the results table 7.2 highlight a huge improvement in the MOTA metrics thanks to the CAE-based local features, from 57.2 to 63.5 for FD-Gan, from 55.0 to 60.5 for DG-Net. Remembering that MOTA is an index derived from three different sources of error, FP, FN and ID-swithces (see chapter 6), by looking at the numbers in the aforementioned table, the number of false positives and false negatives were barely affected by the introduction of the local appearance model (i.e. the CAE-based one), thus the great majority of the improvement is due to the reduction in the number of ID-swithces (a visual representation of this is shown in figure 7.5).

In the light of these facts, the increased execution time (which, moreover, is in itself marginal[1]) due to the introduction of the local appearance description, is completely justified by the improvements on meaningful indices such as MOTA and ID-swithces. In Addition, as stated before, the elaboration of each frame was run in sequence, without parallelization, in order to evaluate the execution speed without the non-determinism introduced by multi-threading, and this means that the frame-rates achieved (see last column of table 7.2) can be further increased simply by introducing a multi-threaded programming approach.

[1] 10.1% average increase for the FD-Gan architecture and 10.5% for the DG-Net architecture (the values are deduced from the *Hz* column in table 7.2

CHAPTER 8

Conclusions and Future Developments

In this thesis a novel baseline for improving traditional tracking-by-detection algorithms is proposed. One important contribution was the introduction of local appearance features that respect topological coherence; this means that the system only confronts features coming from the same body part, for instance right shoulder with right shoulder, see section 4.3.1. The aim of this was to reduce the total number of identity switches, particularly in case of partial occlusion (i.e. when a target is partially occluded by another target).

Chapter 7 has proven the benefits of carefully designed local features, showing that they can greatly reduce the noise on the tracks by reducing the number of identity switches significantly.

The basic idea behind the CAE-based local appearance model was to have a description of the appearance of each visible local part of the target, by extracting both color arrangement and texture information, using a light-weight architecture in order to keep the computational burden as low as possible. From the results section it is evident that the proposed local appearance description (i.e. the CAE-based model) succeeds in its task without requiring a huge amount of resources, barely affecting the execution time.

By analyzing the obtained results, it is evident that the reduction of the number of identity switches has a huge impact on the overall performance of the tracking system, by directly improving the MOTA score and also making the job of the Kalman filter easier since it reduces the noise of the tracks.

Furthermore, the choice of the score-level fusion strategy proved to be effective, when all the learners in the ensemble show similar performances (see section 7.3), and ultimately, in this particular context was the only viable option, due to the fact that the local appearance description is variable in size, as explained in the previous chapter.

Lastly, the selective update strategy for the global appearance descriptor and the incremental update strategy for the local one, proved to be helpful in handling the different cases (dominant and overlapped), making the most of the strengths of the two descriptors.

8.1 Future Developments

This thesis was intended as a baseline, and the aim was to prove that local features can overcome some of the flaws of using a solely global appearance model of the targets.

This base concept can surely be improved by designing a better local features extraction pipeline that can represent another future development task. In parallel, a more interesting possibility would be to use this features indirectly to improve the quality of the global appearance model. From this point of view, one can imagine to insert a learning mechanism for local features as an improvement of an existing network architecture.

As an example, in chapter 2 an overview of the FairMOT architecture was presented, which is faster than most architectures and still presents state of the art performances. The benefits of FairMOT are related to the fact that its global appearance model is less affected by background noise respect to the traditional tracking-by-detection algorithms. This is obtained via a multitask learning strategy where two heads are responsible for the separate tasks of detection an appearance modelling, while sharing a common backbone.

According to the previous observations, such an architecture can be improved by adding a third head for the extraction of local features, opening two new, different possibilities. the first one is to use the local features directly, by performing an information fusion either at the score level or at the feature level. The second one is to use the feature indirectly, meaning that by training the third head of the network for the local features extraction.

As stated in [7], one of the advantages of a multi-task learning (MTL) approach is the fact that by exploiting similarities of the different tasks to be learned can improve the learning efficiency and the prediction accuracy for every task, respect to when the models are trained separately.

MTL models usually consist in a backbone of layers shared by all the different tasks (usually the lower levels of the network) to which the specialized ”heads” are attached, one for each different task to be learned. The shared layers are trained with the goal of being able to extract meaningful features for the whole set of different tasks without being polarized to any of them.

This feature of MTL can be exploited by adding a third head to the FairMOT architecture to learn the local appearance features. This means that the shared layers will also learn to extract local features, which will be propagated also to the global appearance description head, hopefully making it more robust to partial occlusions.

This means that with a careful design of the architecture and a correct training strategy, the Re-Identification head would become more robust against partial occlusions, even if at inference time the local features are not used, thus keeping the execution time

unchanged.

Figure 8.1 presents the general idea of the hypothesized architecture deriving from the original FairMOT implementation. In the case the changes forced by the presence of the new head during the training step on the pre-existing Re-Identification head proves to be an insufficient improvement for the robustness against partial occlusions (when using the Re-Identification head alone at inference time), there is still the possibility to use the local appearance features head in the same way it was used in this thesis, by combining it with the global description through the score-level weighted averaging, obtaining a single, unified scoretable.

The first solution is however preferable to the latter because having more heads active at inference time means that more computational resources are required, both for the extraction of the local features and for the score fusion step.

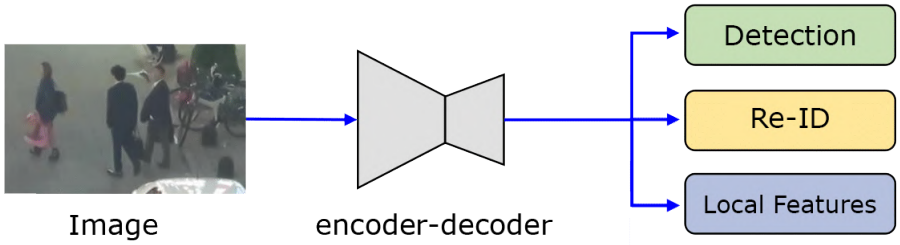

Fig. 8.1 Possible modified FairMOT architecture: with reference to the original scheme (see figure 2.1), a third head for the local appearance modelling was added

Furthermore, this work experimentally proved that the proposed use of the Kalman filter is beneficial to the overall performances of the system, Unfortunately, the Kalman Filter introduces a constraint: the camera must remain still. A possible workaround is to integrate this solution with an ego-motion estimation system for the camera, such as a simultaneous localization and mapping (SLAM) one, in order to extend the use of this paradigm to scenes coming from a moving camera.

REFERENCES

[1] Maryam Babaee, Ali Athar, and Gerhard Rigoll. Multiple people tracking using hierarchical deep tracklet re-identification, 2018.

[2] S. Bae and K. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1218–1225, 2014.

[3] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles, 2019.

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, Sep 2016.

[5] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking, 2020.

[6] Laanoja Risto Buldas Ahto, Kroonmaa Andres. Keyless signatures' infrastructure: How to build global distributed hash-trees. In Hanne Riis Nielson and Dieter Gollmann, editors, *Secure IT Systems*, pages 313–320, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[7] Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, 1997.

[8] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Deep convolutional autoencoder-based lossy image compression. In *2018 Picture Coding Symposium (PCS)*, pages 253–257, 2018.

[9] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes, 2020.

[10] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints, 2015.

[11] Yixiao Ge, Zhuowan Li, Haiyu Zhao, Guojun Yin, Shuai Yi, Xiaogang Wang, and Hongsheng Li. Fd-gan: Pose-guided feature distilling gan for robust person re-identification, 2018.

[12] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, Dec 2016.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking, 2020.

[17] Paul R. Kalata. - target tracking systems: a survey. In *1992 American Control Conference*, pages 832–836, 1992.

[18] Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[19] Shyamgopal Karthik, Ameya Prabhu, and Vineet Gandhi. Simple unsupervised multi-object tracking, 2020.

[20] Fouad Khelifi and Ahmed Bouridane. Perceptual video hashing for content identification and authentication. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):50–67, 2019.

[21] Neal Krawetz. Looks like it. `http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html`, 2011.

[22] Neal Krawetz. Kind of like tht. `http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html`, 2013.

[23] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[24] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955.

[25] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking, 2015.

[26] Marco Leonardi and Annette Stahl. Convolutional autoencoder aided loop closure detection for monocular slam. *IFAC-PapersOnLine*, 51(29):159–164, 2018.

[27] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[29] Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking, 2016.

[30] Mizuho Nishio, Chihiro Nagashima, Saori Hirabayashi, Akinori Ohnishi, Kaori Sasaki, Tomoyuki Sagawa, Masayuki Hamada, and Tatsuo Yamashita. Convolutional auto-encoder for image denoising of ultra-low-dose ct. *Heliyon*, 3(8):e00393, 2017.

[31] NIST. Secure hash standard. federal information processing standard. fips-180-1, 1995.

[32] International Commision on Illumination. Iso/cie 11664, cie 1976 l*a*b* colour space.

[33] Nikolai Proskuriakov, Boris Yakovlev, and N Arkhangelskaia. Features of the cie76, cie-94, and cie-2000 methods that affect the quality of the determining process of the image average color. *Journal of Physics: Conference Series*, 1791:012107, 02 2021.

[34] Z. Pylyshyn and R. Storm. Tracking multiple independent targets: evidence for a parallel tracking mechanism. *Spatial vision*, 3 3:179–97, 1988.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[37] R. Rivest. Mit laboratory for computer science and rsa data security, inc. 1992.

[38] C. Schooler. Optimal - filters for systems with modeling inaccuracies. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11:1300–1306, 1975.

[39] Jack Sklansky. Optimizing the dynamic parameter of a track-while-scan system. *RCA Laboratories, Princton, N.J*, June 1957.

[40] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2, 1999.

[41] Zhenjun Tang, Y. Dai, and X. Zhang. Perceptual hashing for color images using invariant moments. *Applied Mathematics and Information Sciences*, 6:643S–650S, 04 2012.

[42] ISO/TC 130 Graphic technology. Iso 12647.

[43] Dirk Tenne and Tarunraj Singh. Optimal design of --() filters.

[44] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017.

[45] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[46] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *International Conference on Computer Vision*, 2015.

[47] Chunbo Xiu and Xuemiao Su. Composite convolutional neural network for noise deduction. *IEEE Access*, 7:117814–117828, 2019.

[48] Min Yang and Yunde Jia. Temporal dynamic appearance modeling for online multi-person tracking. *Computer Vision and Image Understanding*, 153:16–28, Dec 2016.

[49] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking, 2020.

[50] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, 2015.

[51] Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. Joint discriminative and generative learning for person re-identification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[52] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman Hall, 2012.