FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Pedestrian reidentification in camera system
Student:	Erik Hulmák
Supervisor:	Ing. Filip Naiser
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of winter semester 2021/22

Instructions

Managers of complex buildings (like shopping malls, office space) are dealing with various tasks (e.g., minimization of wait time, queue prevention, advertisement placement).

The pedestrian detection system, gender and age prediction, and inter-camera identity preservation are crucial in order to know the customer. In
iC Systems.ai, s.r.o. we are developing such systems.

A student is going to deal with identity perseverance among multiple cameras. At first, he performs a literature review on this topic. Then, he will design a solution for cameras with and without a field of view intersection. Next, the student will design and train a neural network. This network will encode the given image crop into a descriptor vector. The student will use this vector for measuring similarity between pedestrian image crops. He will also design inter-camera identity matching. This matching process will take into account locational and time consistency.

References

Will be provided by the supervisor.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 24, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Pedestrian Re-Identification in a Camera System

Erik Hulmák

Department of theoretical computer science
Supervisor: Ing. Filip Naiser

January 7, 2021

Acknowledgements

Firstly, I would like to express my very great appreciation to my supervisor Ing. Filip Naiser, for his unbelievably patient guidance, faith, and enthusiasm. We had many valuable discussions, not only about theory but also about the proper mindset. I would also like to thank my friends and family for their warm support during the studies and my dog Yllo for being a ball of infinite optimism. Finally, I would like to give thanks to my figurants, namely Tomáš Pěnkava, Jan Jirák, Sára Luštková and Jiří Hulmák for fun time during recoring the demonstrations in the Krakov shopping center in Prague.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. I further declare that I have concluded an agreement with the Czech Technical University in Prague, on the basis of which the Czech Technical University in Prague has waived its right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act. This fact shall not affect the provisions of Article 47b of the Act No. 111/1998 Coll., the Higher Education Act, as amended.

In Prague on January 7, 2021 …………………

Czech Technical University in Prague
Faculty of Information Technology
© 2021 Erik Hulmák. All rights reserved.
This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Hulmák, Erik. *Pedestrian Re-Identification in a Camera System.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Sledování pohybu osob a jejich Re-Identifikace je zajímavá oblast výzkumu s širokou aplikací. Cílem této práce je rekonstrukce trajektorie průchodu lidí komplexními prostory, jako jsou například obchodní centra nebo kancelářské budovy. Představujeme dynamický algoritmus vhodný pro paralelizaci, co sleduje pohyb lidí v systému kamer. Oblast záběru dvou kamer se může, ale nemusí, překrývat. Nejdříve získáme deskriptory pomocí Siamských sítí a následně hledáme vhodná spojení mezi tracky pomocí Hungarian algoritmu. Všechna data byla sesbírána v obchodním centru Krakov v Praze. Zároveň představujeme algoritmus pro tvorbu objemných datasetů bez nutnosti heavy labelingu vhodný pro inter-camera tracking.

Klíčová slova počítačové vidění, multi-object inter-camera tracking, smart dataset, dynamické spojování tracků, re-identifikace osob, udržení identity

Abstract

Pedestrian tracking and their Re-Identification is an exciting area of research with a wide range of applications. This work aims to reconstruct multiple objects' trajectories through spacious complexes, such as shopping malls or office buildings. We present a dynamic algorithm suitable for parallelization that tracks multiple objects between cameras with a possible field of view intersection. First, we obtain image descriptors with a Siamese Network. We use descriptors to associate tracks into trajectories with the Hungarian algorithm. We built the whole project on our dataset collected in the Krakov shopping center in Prague. We also present an algorithm for creating large scale datasets suitable for inter-camera tracking without the need for heavy labeling.

Keywords computer vision, multi-object inter-camera tracking, smart dataset, dynamic track association, person re-identification, identity perseverance

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Expectations about Artificial intelligence (AI) went lately through the roof yet still with a noticeable fear. Incorporating AI in business processes positively correlates with an understanding of how such models work [1]. Whether managers of large complex buildings (like shopping malls, office spaces) can see smart systems effects is beyond doubt. They are aware of how significant influence AI may have on, for example, a suitable shop or advertisement placement, queue prevention, the number of employees, or social distancing in a crisis like the COVID-19. Even with common sense, we can notice that different groups of people tend to have different habits. Some people prefer simplicity, and they go shopping with a clear view of what they want. For a different group of people, the mall provides a convenient place for hanging out.

Even with this basic knowledge, we can estimate better shop placement. E.g., sports and groceries close to the entrance so that customers do not have to go far for staple items. After careful observation of customer's behaviors and traits like their movements, age group, or gender, we quickly realize that problems become suddenly solvable.

To gather these pieces of information, we have to consider the sheer vastness of observed places and install enough cameras with an often intersected field of view. Now we need to make this set of isolated cameras work as one giant organism. However, this is associated with another crucial question: how to preserve a moving customer's identity among these isolated sets of cameras?

In IC Systems.ai, s.r.o. (iC), we develop such smart systems. We help accumulate data for the managers mentioned above, surveillance companies, or anyone who needs to track multiple people. We gathered the data for pure statistics. So saving absolute identification is not our objective. Whenever we use the term ”identification,” we only mean it from a local perspective. It is impossible to match customers across multiple visits, and it is not even our intention to do so. We do not do facial recognition. Our method is based just

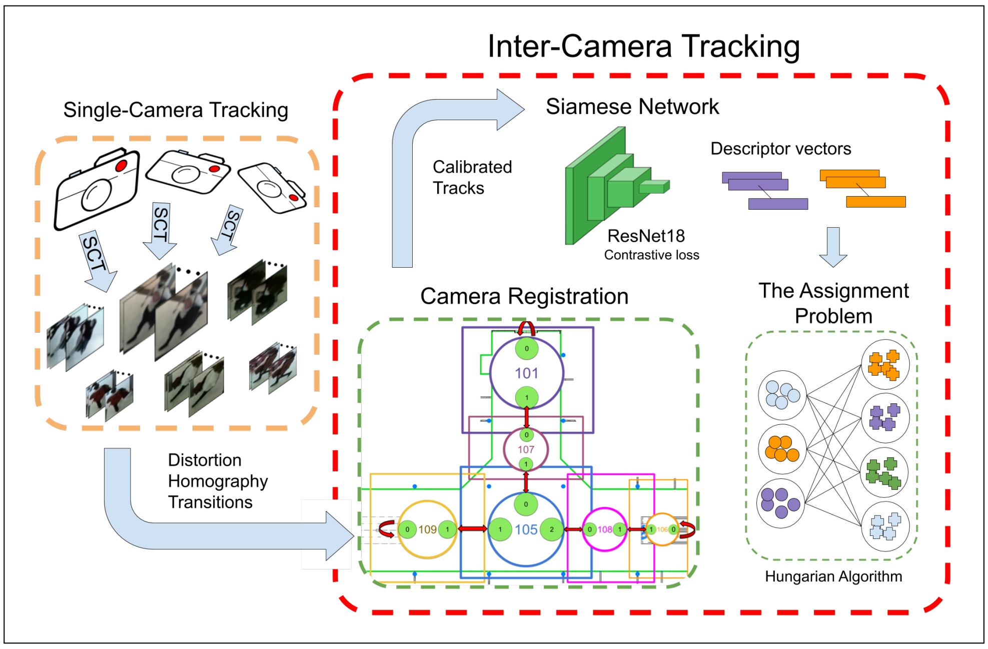

Figure 1.1: The pipeline of Inter-Camera Tracking.

on color and shape. If, for example, someone takes their jacket off on a toilet, we cannot match him to his previous self. This approach respects the GDPR. Please note that each person published in this thesis is a contracted actor.

This thesis's most significant advantage is that it is driven by demands and offers excellent experience opportunities. To handle everything well, We have to study theory properly and test our knowledge in practice with strictly limited hardware resources. This combination is a perfect presupposition for effective learning. The project is beneficial for enhancing various services and event organizing so that our work can be instrumental. We do believe the outcome will be helpful and favorably appreciable.

1.1 Objectives and Formulation

The goal is to perform multi-target *inter-camera tracking* and Re-Identification in reasonable time and space. Assume we have a system of m cameras $C^1, C^2, ...C^m$ with possible view overlaps. Each camera makes observations $O^1, O^2, ..., O^i$. Observations are tracks of pedestrians $p_1, p_2, ..., p_k$ in the context of a single camera. We use the term observations only if we want to differentiate between tracks from a person's trajectory and observations(tracks) from a camera. Trajectory of person p_i consists of n tracks T^1, T^2, T^n.

This project focuses on the re-identification of pedestrians and re-creation of their trajectories in a system of m cameras. Our input is a set of cameras

$C^1, ...C^m$ with their observations $O^1, ...O^m$. The objective is to successfully re-identify pedestrians and link their tracks into trajectories.

1.2 Overview of the Thesis

This project consists of three stages covered in 5 chapters. Each chapter analyzes stages from a different viewpoint.

In the first *Smart Dataset* stage, we describe our method of obtaining a representative dataset. The second stage, *Image Descriptor*, explores the problem of embedding individual image crops and measuring similarity between tracks. The third stage, *Inter-Camera Assignment Problem*, focuses on track association and person trajectory reconstruction.

Every stage is divided into multiple chapters that altogether cover theory, implementation, experiments, and results.

- Chapter 2 analyzes the theory of Siamese Networks, track association and person Re-Identification overall.
- In Chapter 3, we describe the environment in the Krakov shopping center in Prague. Cameras, installation, Inter-Camera calibration, and nature of our footage.
- Chapter 4 introduces methods and implementations of the *Smart Dataset*, *Image Descriptor*, and the *Inter-Camera Assignment Problem.*
- In Chapter 5, we cover experiments, tests, evaluation methods, and results.
- In Chapter 6, we summarize the whole pipeline and discuss future improvements.

CHAPTER 2

Theory

2.1 Track, Detection

Detection is a small area defined by an image crop or a bounding box containing most of the target's body. Every detection stores metadata, such as timestamp, score (confidence that the crop contains a person), and estimated local coordinates. How we use local coordinates for obtaining global coordinates is explained in section 3.4.

Track is a chronologically sorted sequence of detections with various attributes: camera id, source gate id, exit gate id, and track id. An example of tracks from multiple cameras is depicted in figure 2.1. To obtain these sets of tracks, or if you wish observations, it is necessary to solve a Single-camera tracking (SCT) first. Supervisor Ing. Filip Naiser has already implemented this problem. Let us briefly outline the process of SCT.

2.2 Single Camera Tracking

The SCT's essence is to generate a set of tracks O^i with a single camera C^i. C^i is placed on the ceiling while objects move under the field of view simultaneously. They can come and leave at any time point through specified gates. The job is to detect objects precisely as possible, register coordinates, timestamps, and gates that they passed. If we link individual detections into tracks, we successfully achieved SCT.

2.2.1 Target Detection

The first objective would be the detection of potential targets.

There is a possibility to use the state-of-the-art Real-Time object detector YOLOv4 [2] or an older approach Faster R-CNN [3]. Nevertheless, these models exist for much more general purposes. They can identify and detect hundreds of classes, and their training is time consuming. They demand

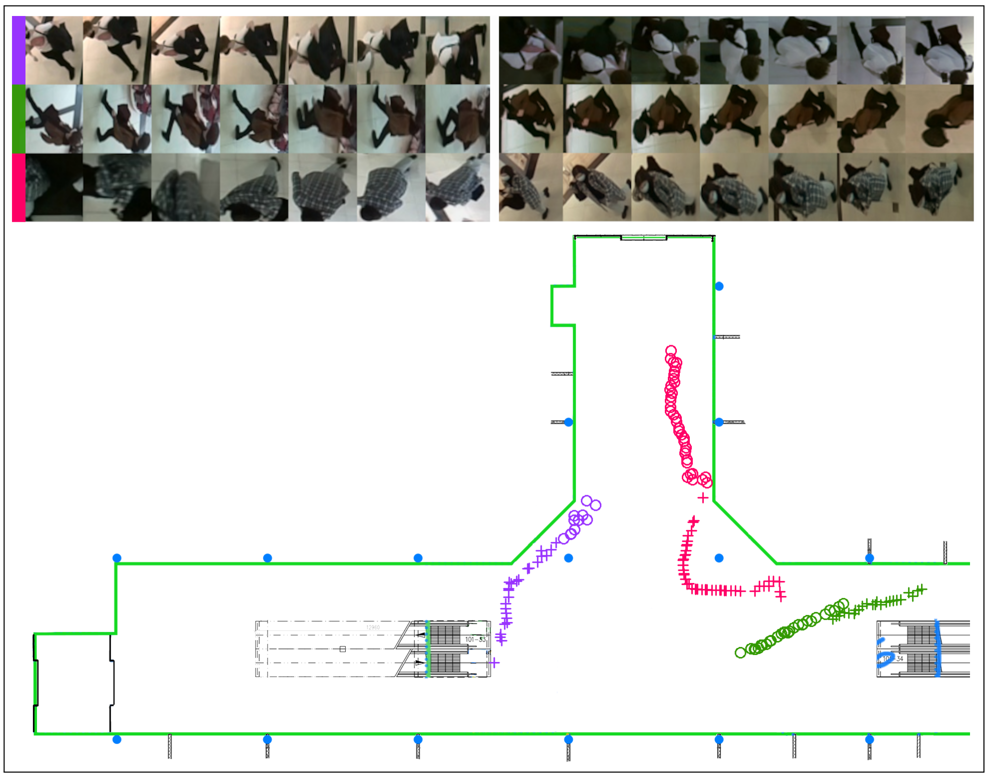

Figure 2.1: Example of detections arranged in tracks that forms a person's trajectory. Each displayed person consists of two tracks from two different cameras (+ and ◯).

long and heavy computations with high memory requirements and enormous datasets that are not feasible in our environment.

Computational resources are in IC limited. Therefore they had to come up with a completely different algorithm. They combine a sliding window, AdaBoost [4], and a CNN classifier. The sliding window and the AdaBoost first suggest places with plausible targets. Such positions are called proposals, and their boundaries are defined with bounding boxes. Then the CNN classifier decides whether there is or is not a person. The whole image is compressed into a set of small image crops. This leads to faster run time and a cheaper training process with modest data set. In this thesis, we are not going to explain the target detection further. Let us instead focus on the next topic, which is closely related to the assignment.

2.2.2 SCT-Image Descriptor

After the target detection, it is necessary to encode crops into descriptor vectors. There are two crucial requirements for the descriptor. If we encode two crops into descriptor vectors, we can measure distances between these vectors. The distance tells us how similar these two crops are.

It is desirable to obtain rather short vectors to save computational time during SCT. For this task, IC historically used the Color Names algorithm [5, 6]. It is a fast and easy baseline method, which works as a lookup table that encodes RGB colors into 11 categories. The authors introduced bins based on large scale research of image database. The more sophisticated approach would be using a Siamese Network (for example, the Twofold Siamese Network [7]).

A visual descriptor for SCT can be the same one as a visual descriptor for ICT. We tried to use the locally trained SCT-descriptor for ICT, but we did not meet any success. Therefore, we expect improvements in SCT with the use of the ICT-descriptor. In section 2.3.1, we discuss the topic of image descriptors in greater detail.

2.2.3 Track Construction

The final stage of SCT is linking detections into continuous tracks. Engineers in IC use descriptor distance as the cost function and they formulate the task as a spatio-temporal matching problem. For a better result than greedy matching, they use the Hungarian algorithm [8]. When formulated correctly, it is a proven stable solution resulting in continuous sequences.

The whole pipeline is as follows: first, they detect all interesting objects from a frame. With the use of the image descriptor, they perform matching with all detections from the next frame. This approach results in simple matching, which can be further improved with more sophisticated methods. They, for example, implemented a motion model for position prediction with the Kalman filter [9]. They also solved many real-world problems, e.g., dropping frames, incomplete detections, or strong shadow silhouettes.

2.3 Inter Camera Tracking

The problem of inter-camera identity matching, as it is described in section 1.1, is discussed quite frequently. We consider ICT as a composition of two crucial sub-problems: image descriptor 2.3.1 and ICT Assignment problem 2.3.2. The problem of ICT has been previously done. Appearance similarities can be covered with handcrafted descriptors [10], or learned by Siamese networks with triplet loss and hard sample mining [11, 12], center loss and label smoothing [13], or fully unsupervised online learning [14]. The inter-camera assignment problem is solvable with a static correspondence matrix and the Hungarian algorithm [8, 15, 16], more dynamically without the inter-camera calibration [17, 18] or correlation clustering association [19, 11].

2.3.1 ICT-Image Descriptor

In computer vision, visual descriptors encode interesting visual features of the image contents into a vector of numbers. They can either describe images generally or describe essential (local) pieces of information such as the shape, color, or texture.

Our interest is leaning towards global descriptors. They describe objects as a whole. There are many different approaches, but we usually classify them into two main categories, called handcrafted and learned features. Handcrafted descriptors are usually manually predefined on rather small datasets, but they require expert knowledge. For many years, SIFT [20] has been among the most popular handcrafted feature descriptors. On the other hand, learned descriptors extract features through learnable models. And thanks to the deep learning progress, they are becoming mainstream.

Descriptor Properties Suitable for Person Re-Id

Given an arbitrary pair of two objects, we want their descriptors to be close if both objects belong to same class. On the other hand, we want descriptors to be distant if the objects come from different classes. The distance can be considered as resemblance. Objects can be compared relative to others, or they can be compared absolutely to some threshold value. Both approaches are, in our case, relevant.

Then, we are able to encode object into a point in multidimensional space. To spare a computational time, we want rather short descriptors. We suppose that the vector length should not be greater than 64.

Depending on our metric of choice, we might face the curse of dimensionality. Luckily, we can use methods like [21].

Color Names - A First Solution

Color Names [5, 6] encodes RGB colors into 11 categories: black, blue, brown, grey, green, orange, pink, purple, red, white, and yellow. With the use of such a lookup table, we classify each pixel in an image. Then we calculate a normalized histogram of each color name. Then we can compare two images encoded with this technique.

Color Names are due to their simplicity, suitable baseline method. In the spatio-temporal domain, we matched the majority of pairs within a single camera correctly with just color features.

The environment of ICT conspicuously differs from SCT in such a manner that each camera brings a different viewpoint, deformation, or illumination. It increases demands for robustness. Our visual descriptor should be invariant to such changes. Multiple cameras also make the domain for association significantly larger. And absolute color comparison becomes less relevant. A demonstration of the manifestation of different camera influences is depicted

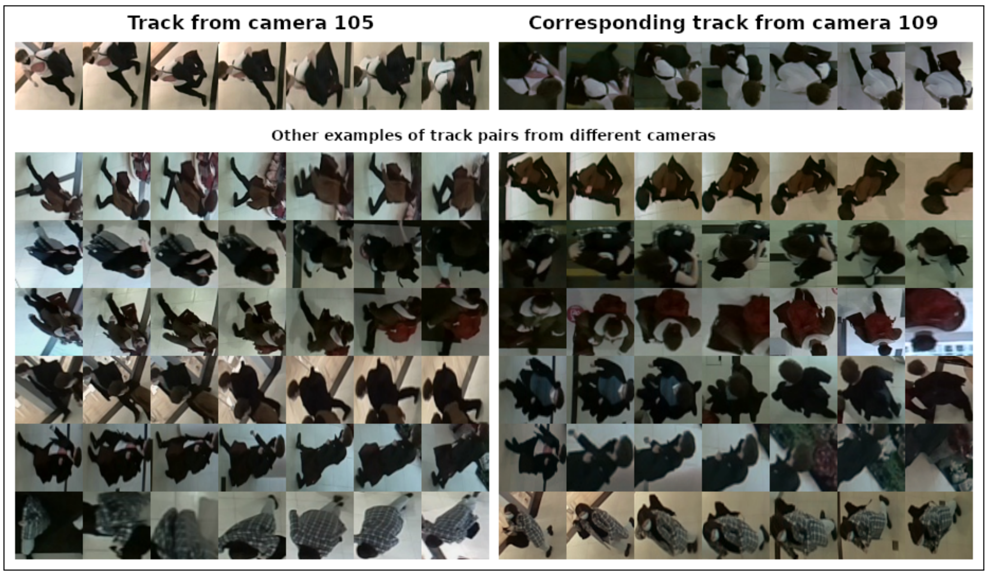
Track from camera 105
Corresponding track from camera 109
Other examples of track pairs from different cameras

Figure 2.2: Comparison of observations from different cameras.

in figure 2.2. Comparison of the Color Names, our learned descriptor and random descriptor is visible in table 5.3.

However, it is possible to overcome the difficulty of having different colors between cameras. One solution might be using a Camera Transfer Function in form of a 2-bin RGB quantization matrix as authors in [10, 22] presented. Topic of color transfer is also used in [14], or described in greater detail in [23]. Nevertheless, we are going to take a step towards learned descriptors and leave the idea of color transfer for future work.

Learning Features with Convolutional Neural Networks

CNN are a special sub-type of neural networks commonly applied in computer vision [24]. They generally consist of convolutional layers, pooling layers, and fully connected linear layers. Examples of convolutional neural networks that may be usable in our thesis are [25, 26, 27]. During writing this thesis, all of the mentioned models performed reasonably well on the COCO dataset in Object Detection problem [28, 29].

Generating an image descriptor is often with the use of the Siamese networks, which have been generally used for similarity learning [30] and one of the applications is real-time object tracking [7], or face recognition [31]. A Siamese network is special type of CNN used for similarity learning. When we feed the network with two inputs x, y, we apply an identical function f on both inputs. Then we can measure the similarity of $f(x)$ and $f(y)$ with, for example, Euclidean distance, and we can use this as a loss function.

For training a Siamese network, we can either use supervised or semi-supervised, and even unsupervised methods for person Re-Id learning [32]. Various loss functions exist too [33, 34, 35, 13]. During our research we developed a special variation of triplet loss the *cluster loss* (sec: 4.2.2). There is also a possibility to combine these approaches with the curriculum learning [36] which is also a good practice for person re-identification problem [12]. To prevent vanishing/exploding gradients, we can use gradient clipping [37] and Kaiming weight initialization [38]. We can limit the latent space for descriptors to n dimensional unit sphere with the softmax function. Or favorably use influence of different batch sizes [39] (beginning at batch size=1). Different sources like [13] advises to warm-up the network and to adjust learning rate during the training. Authors in [13] demonstrate that specific combinations of loss functions may help significantly.

2.3.2 Inter-Camera Track Assignment Problem

The next part of this thesis is to reconstruct pedestrian trajectories. To do so, we have to link corresponding tracks into sequences so that pedestrians' identities are preserved. As formulated in section 1.1, the result should obey rules of logic, e.g., realistic transitions between cameras, no teleportation, and no multiple occurrences of one person in different places at the same time.

Authors present an interesting solution in [15, 16], where they try to maximize the joint linking probability with the Hungarian algorithm [8] for minimal weight matching. The solution is focused on two cameras and then extended to the problem with multiple cameras. First, they define a correspondence matrix $\mathbf{H}$ of size $(2m + 2n) \times (2m + 2n)$ as follows:

$$\mathbf{H} = \left[\begin{array}{cc|cc} \mathbf{A}_{m\times m} & \mathbf{B}_{m\times n} & \mathbf{E}_{m\times m} & -\infty_{m\times n} \\ \mathbf{C}_{n\times m} & \mathbf{D}_{n\times n} & -\infty_{n\times m} & \mathbf{F}_{n\times n} \\ \hline \mathbf{G}_{m\times m} & -\infty_{m\times m} & \mathbf{0}_{m\times m} & \mathbf{0}_{m\times n} \\ -\infty_{n\times m} & \mathbf{K}_{n\times n} & \mathbf{0}_{n\times m} & \mathbf{0}_{n\times n} \end{array}\right] \tag{2.1}$$

where the parts of the matrix are as follows:

A	targets leave and return back to Camera a
B	targets leave and return back to Camera b
C	targets leave Camera b and enter Camera a
D	targets leave and return back to Camera b
E	targets terminate in Camera a
F	targets terminate in Camera b
G	new targets initialized in Camera a
K	new targets initialized in Camera b

Then authors apply the Hungarian algorithm [8] and process the result.

We based our method on the minimal weighted matching, but we perform matching on a dynamic graph structure, not on a static matrix with all matching possibilities (described in section 4.3). Our graph is also very sparse, which means that it is not optimal to use the Hungarian algorithm since it accepts only fully connected bipartite graphs. The Hungarian algorithm is sufficient for this project, but we can use a much more sophisticated algorithm as in [40]. Our approach does not surpass the algorithm proposed in [15]. However, we benefit from incorporating various heuristics that make the computation much more efficient.

2.3.2.1 Cost Function

For building a track assignment problem, we need a cost function. The result should combine spatial, temporal, and appearance similarity between tracks. Each detection has global coordinates, which can be used for trajectory estimation. Trajectory comparison can be used only with overlapping cameras, as described in section 4.1.2. Future work may take into account patterns of motion, walk analysis, or temporal prediction, as presented in [17].

For measuring temporal similarity, we can estimate a transition duration (sec: 3.4). And then, concerning the transition duration d, for tracks t_i, t_j, determine the cost as $d - \text{time\_difference}(t_i, t_j)$. If the cost exceeds a certain limit, we consider the cost to be infinity.

Each descriptor can be represented as a point in n-dimensional space. Properly working descriptor should project descriptors of identity in proximity with descriptors representing the same identity. Since we are matching sets of detections (i.e., tracks), we use cluster distance metrics. We consider the following distance metrics for appearance similarity of two sets of tracks T^a, T^b, and embedding function f.

- **Maximum:** $max\{d(f(t_a), f(t_b)) : t_a \in T^a, t_b \in T^b\}$
- **Minimum:** $min\{d(f(t_a), f(t_b)) : t_a \in T^a, t_b \in T^b\}$
- **Mean:** $mean\{d(f(t_a), f(t_b)) : t_a \in T^a, t_b \in T^b\}$
- **Centroid:** $\|c_a - c_b\|_2^2$ where c_a, c_b are centroids of $f(T^a), f(T^b)$
- **Ward's:** $\frac{|T^a||T^b|}{|T^a|+|T^b|}\|c_a - c_b\|_2^2$ where c_a, c_b are centroids of $f(T^a), f(T^b)$

The final cost function returns a weighted sum of these similarity measurements.

Chapter 3

Data

3.1 Environment

All experiments took place in the Krakov shopping center in Prague, a medium-sized building with three levels and more than seventy shops. We installed and configured around 40 sensors within the area to capture all exits, entrances, shops, and main routes within our reach. Sadly, as a consequence of the COVID-19 outbreak, the overwhelming majority of shops ended closed. The lockdown followed, and people without necessary exceptions had to stay at home. We resolved the situation with six cameras near the largest grocery store, the most frequented place possible, to collect enough data. These six sensors cover all possible intersection cases, and people have to pass them if they want to reach the supermarket. Image 3.1 approximately outlines the area of the first-floor plan, where all presented experiments took place.

3.2 Camera Description

We situate our sensors approximately 5 or 6 meters above the ground. They reliably meet all strategic places. It is desirable to cover a wide area with just one camera, so we use special lenses for a field of view extension. As shown in the following example 3.2, each image is under strong radial distortion.

Sensors have defined virtual gates, which determines if a person leaves or enters the area. We define gates as a sequence of coordinates representing a polygon. They are also necessary to specify which transitions make sense in the real world. These specifications are making the track matching problem significantly easier. As polygons, we also define camera masks useful for marking the region of interest.

Table 3.2 defines transitions within our environment. Each transition is a connection between two exits of two, not necessarily different, cameras. We also specify the average duration of each transition. This value is easily obtainable, as described in section 3.4.

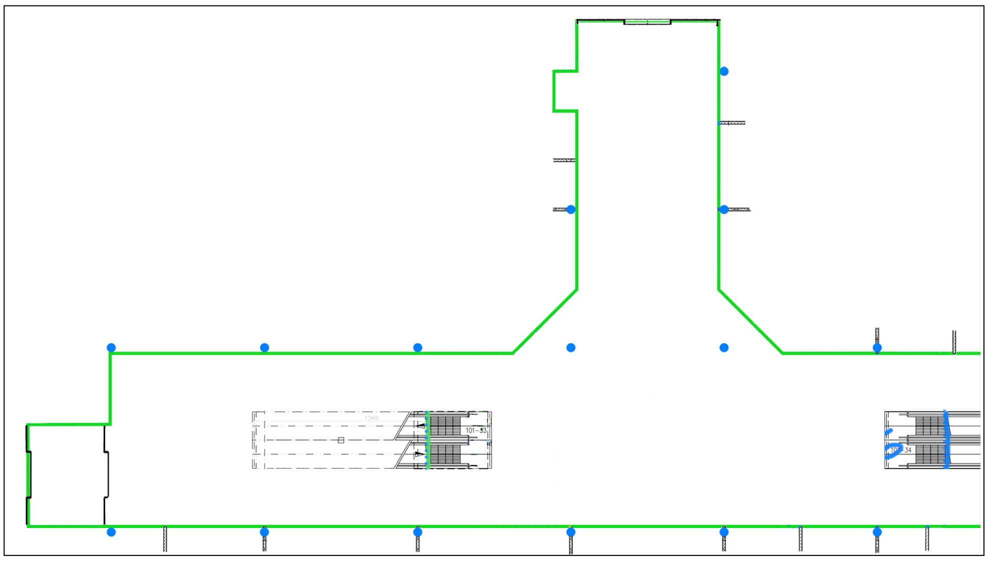

Figure 3.1: Plan of OC Krakov's first floor.

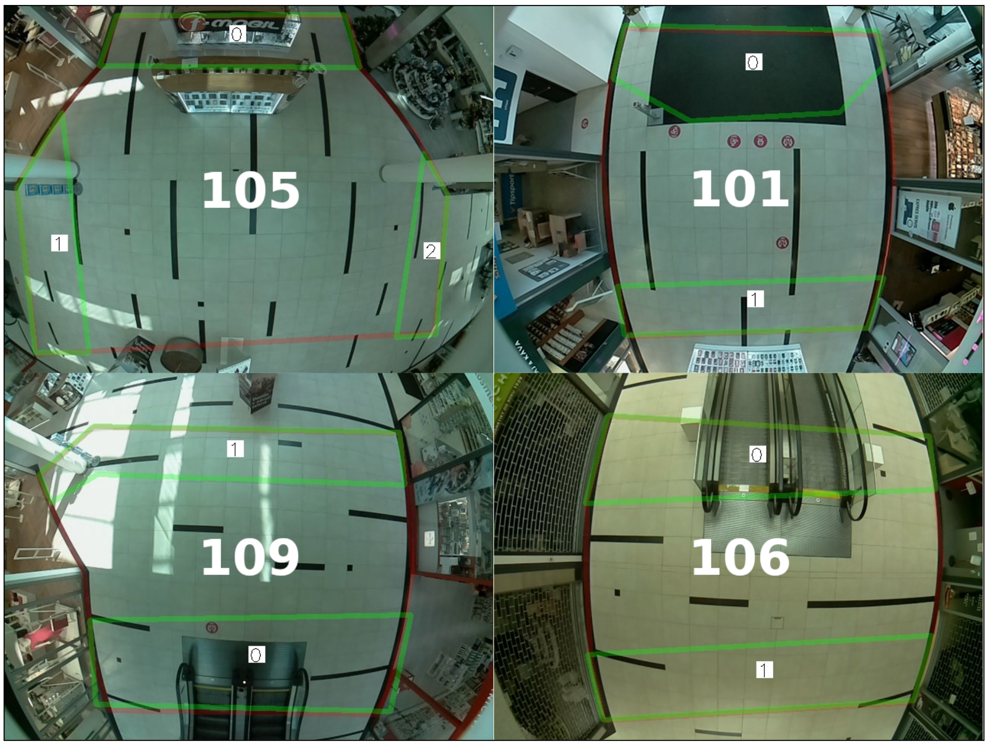

Figure 3.2: Influence of distortion on a selection of 4 cameras. Example of camera mask (red) and gate definition (green).

cam1_id	cam1_exit	cam2_id	cam2_exit	duration[s]
101	1	107	0	-1
101	0	101	0	2
107	1	105	0	-1
105	1	109	1	-1
109	0	109	0	2
105	2	108	0	-1
106	1	108	1	-1
106	0	106	0	2

Table 3.1: Bidirectional transitions between cameras. Leaving and entering the same camera is estimated to 2 seconds. The transition duration is negative for overlapping cameras because the pedestrian appears in the second camera before leaving the first camera.

Notice that only three transitions are leaving and entering the same camera. We estimated the duration of such transitions to two seconds. Pedestrians can not always leave and enter the same camera/exit. For example, when the person leaves on escalators or crosses from camera C^a to overlapping camera C^b. In such a case, the person is observed first by camera C^b, and then he can enter C^a again.

Camera Configuration

Mandatory parameters are height, cam id, frames per second, gate coordinates, camera mask, transitions, lens type for distortion compensation, and homography anchors as ground points for the global coordinate system. We are going to use the points later on to register cameras into the floor plan. Distortion compensation and homography anchors are elaborated on in sections 4.1.4 and 4.1.3.

Camera Output

Cameras record at a resolution of 640x480 pixels with five frames per second. Pixels have three channels, each of which has 255 possible values. We represent images as arrays with the shape 640x480x3. We can record at 1920x1080 resolution with 60 frames per second. However, we had to choose between accuracy and computational cost. In this thesis, we work with the RGB color model.

Further research can explore different formats. There is a possibility that the contribution of texture is higher than that of color [41]. Since our cameras record in the YUV420 format, and then they extrapolate to the RGB format,

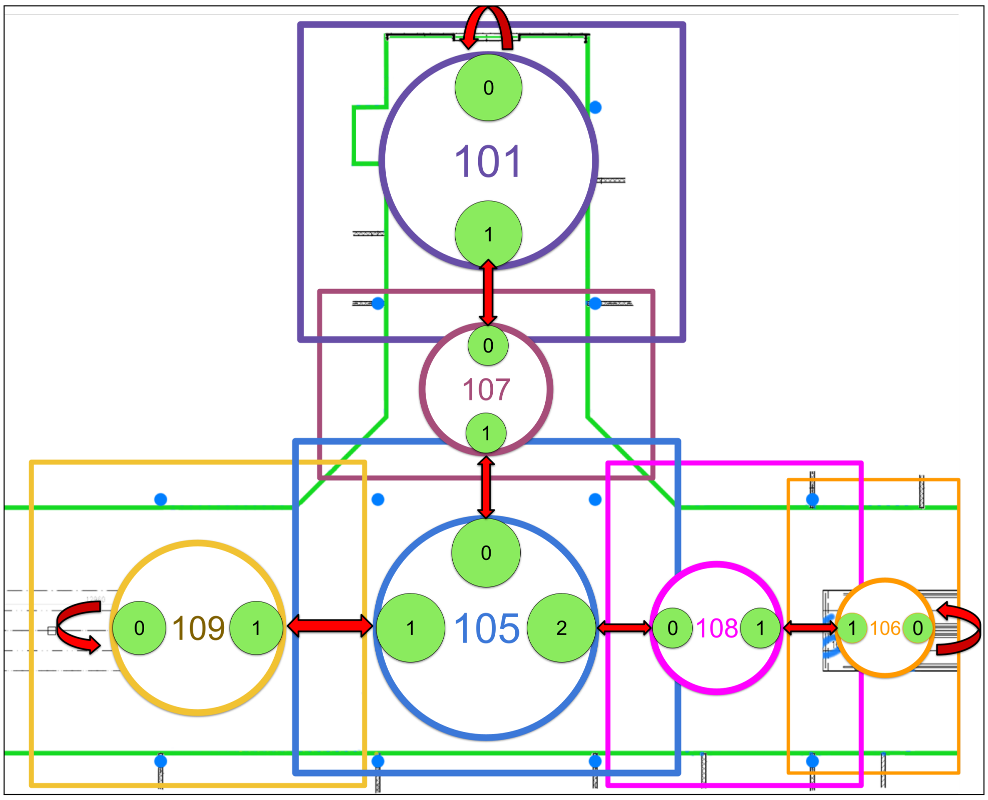

Figure 3.3: Floor plan transition scheme. Each rectangle approximates the field of the camera's view. Green circles are exits, and red arrows are transitions.

there is information redundancy. Using the YUV format might lead to faster inference with the same accuracy.

3.3 Dataset

For appearance similarity learning, we need pairs of tracks with the same person from two different cameras. For spatio-temporal predictions, it is necessary to register coordinates and timestamps for each detection.

Our dataset is strongly dependent on the testing environment based in the Krakov shopping center. Despite the COVID-19 crisis, we were able to gather enough data. Our setup consists of six cameras deployed in a ”T” shape. People can access monitored space with three entrances, one of which is a moving walkway. Figure 3.3 shows the distribution of cameras with possible transitions, and figure 4.3 accurately presents how the footage overlaps. There is always an overlay between two adjoining cameras, but we can simulate a gap by turning off some of the cameras situated in the middle.

We recorded our train set and test set between 06.08.2020 and 05.09.2020, which takes approximately 368 hours of raw video. This footage is not by any means sufficient because our goal is to deliver an all-around product. For better generalization, the data should cover all conditions. For example, in the winter people usually wear coats. Coats are not as colorful, and they also look different than a casual summer outfit. This may result in lower accuracy overall.

3.4 Coordinate System, Distances

Each camera is like a separate universe with its coordinate system, different distortion, and scaling. However, we want to measure distances and navigate in the shopping center with one unified coordinate system. We have to make a model for each sensor that translates coordinates from the local universe to the absolute/global universe (sec: 4.1.4).

Average Crossing Duration

With our single-camera setup, we can estimate the average walking speed. Detections were taken in 0.2[s] intervals. To prevent small deflections caused by frame drops, lags, or inaccurate position estimation, we also measured the distances over more distant pairs. With correct homography, velocity, distance, and time, we can easily derive the mean crossing time for each transition.

Our interest in future work might be directed towards estimating inter-camera space-time probabilities using Parzen windows as presented in [17].

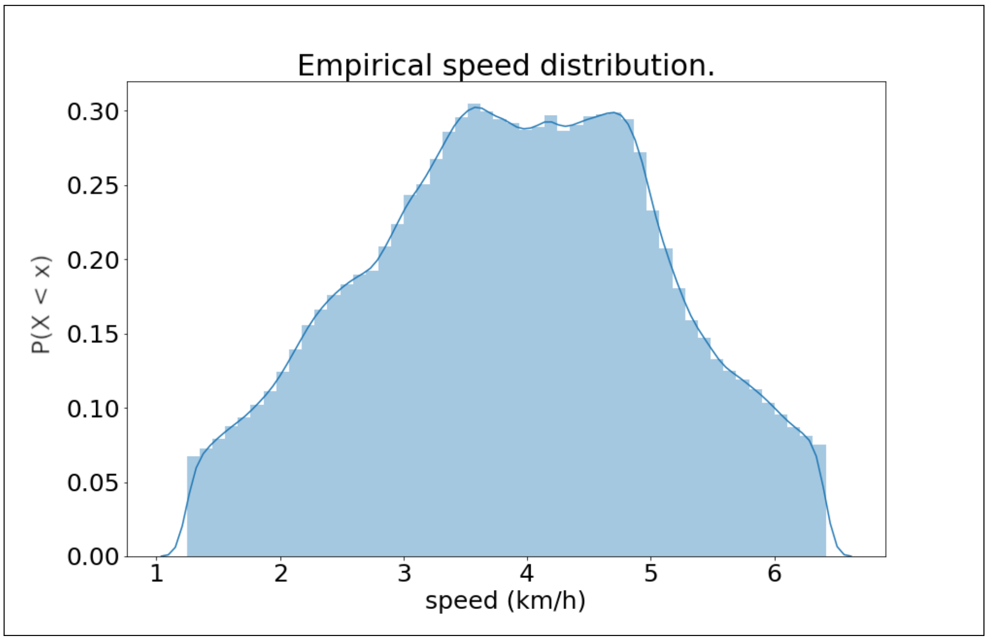

Figure 3.4: Empirical speed distribution.

Chapter 4

Method

4.1 Smart Data Set

The straightforward approach to create the dataset is to annotate the video by a human. He would have to find all detections in every video sample, and then he would have to link these detections into tracks. He would have to document all the metadata like timestamp and exits, camera id, and match corresponding tracks across the whole camera system. We indeed could automate the process as much as possible, but working simultaneously among more than two sensors is just too complicated for a single person to achieve. Moreover, we want at least 10000 tracks, where each of which has approximately 30 detections with extra attributes.

In order to avoid this labor, we introduce an algorithm that makes the whole process exceptionally easy.

The idea

The Smart Dataset can be entirely or partially generated. The algorithm has to be highly reliable, and the result should be mostly correct. Our first strategy was naive, and we thought it would be possible to generate the whole dataset automatically. Later analysis revealed that the task is not as simple as it looks.

The idea is to rely entirely on geometry. We installed cameras so that they overlap heavily. Then we run SCT and classify tracks. With a massive view overlap, we can measure the average distance between all overlapping tracks. Each detection has a ground point. It is a pair of coordinates that estimates the approximate position of the person's feet. If we could somehow compare these coordinates between two different sensors, we could match corresponding tracks based on the distance. However, in order to measure the distance, we have to synchronize time and unify geometry.

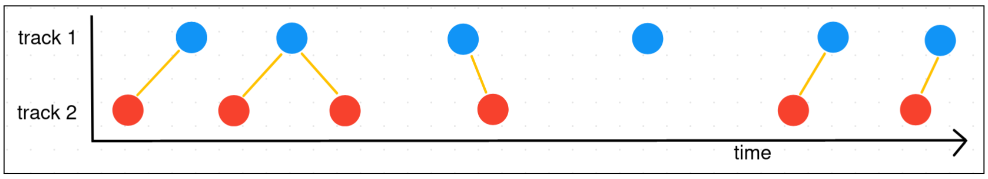

Figure 4.1: Smart Dataset - detection matching example.

4.1.1 Detection Association by the Timestamp

First, we have to find detections taken at a similar time, and then we can measure the distance between them. To create these pairs, we have to take into account multiple problems. The first one is that the images are not taken at the same time nor in equal intervals. The second problem is that we experience occasional frame drops. Approximately every 600 frames camera stops recording for half a second. These small frame drops are problematic because we have to measure the location as precisely as possible. It means that it might sometimes be the best solution to ignore some detections or generate an imaginary point in between them. Scheme 4.1 displays an example of proper association.

4.1.2 Measuring Trajectory Distance Between Two Tracks

With proper association within an overlapping interval, we are almost ready to measure the trajectory distance. We only have to choose a suitable metric for measuring the distance between detections. Multiple options were considered, for example, the mean distance of max n distances, mean distance of min n, where $n \in \langle 1, |distances| \rangle$. However, experiments showed that the average Euclidean distance between pairs is suitable, and other methods are not worth mentioning. Algorithm 1 describes the process of measuring the positional distance between two overlapping tracks. How we acquired coordinates is described in detail in section 4.1.3 and 4.1.4. Part of the SCT phase is smoothing track coordinates with the Kalman filter [9]. This process improves the accuracy of the PositionalTrackDistance algorithm.

4.1.3 Radial Distortion

One way to capture as large area as possible with just one camera is to use a lens with a wide field of view. This approach also has its downsides. The most obvious one is that the original straight lines do not remain straight in the projection. An example of what effect radial distortion may have on a square grid can be seen in figure 4.2. Strong warp makes a notable difference if we want to measure distances precisely.

Algorithm 1: PositionalTrackDistance

```
input: Tracks $t_1$, $t_2$, max detection timestamp difference $\delta$
begin ← $max(t_1[0].timestamp, t_2[0].timestamp)$
end ← $min(t1[|t_1| - 1].timestamp, t_2[|t_2| - 1].timestamp)$
overlap1 ← $[d \mid d \in t_1 \wedge$ begin $\leq d.timestamp \leq$ end$]$
overlap2 ← $[d \mid d \in t_2 \wedge$ begin $\leq d.timestamp \leq$ end$]$
distances ← []
for $d1 \in$ overlap1 do
    differences ← $[|d1.timestamp - d.timestamp| \mid d \in$ overlap2$]$
    if |differences| $> 0$ then
        x ← overlap2$[argmin($differences$)]$
        if x $\leq \delta$ then
            distances.$append(\|d1.coordinates -$ x$.coordinates\|_2^2)$
if |distances| $> 0$ then
    Result: $mean($distances$)$
else
    Result: $-\inf$
```

Radial distortion is not a linear transformation. Thus homography (sec: 4.1.4) does not work here. We had to calibrate the camera first. Both lens distortion calibration and intrinsic camera parameters are estimated using OpenCV's calibration pipelines. All of our sensors use the same camera and the same lens, so we have to calibrate them only once. Quality control is done before shipping to catch a low-quality lens or other issues.

We followed the process of camera calibration with the OpenCV library [42]. Figure 4.2 presents a simple comparison of the calibrated image from our sensor 101.

4.1.4 Perspective Transformation Between Two Planes

We want to register six separate cameras into the floor plan. The objective is to find a transformation H from camera coordinates to global/plan coordinates. The homography matrix H is a 3x3 matrix, and its dot product returns the perspective transformation between the camera view and the floor plan.

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.1}$$

For each sensor, we had to select points carefully, that concurrently with their floor plan counterparts, form a calibration for the homography. The

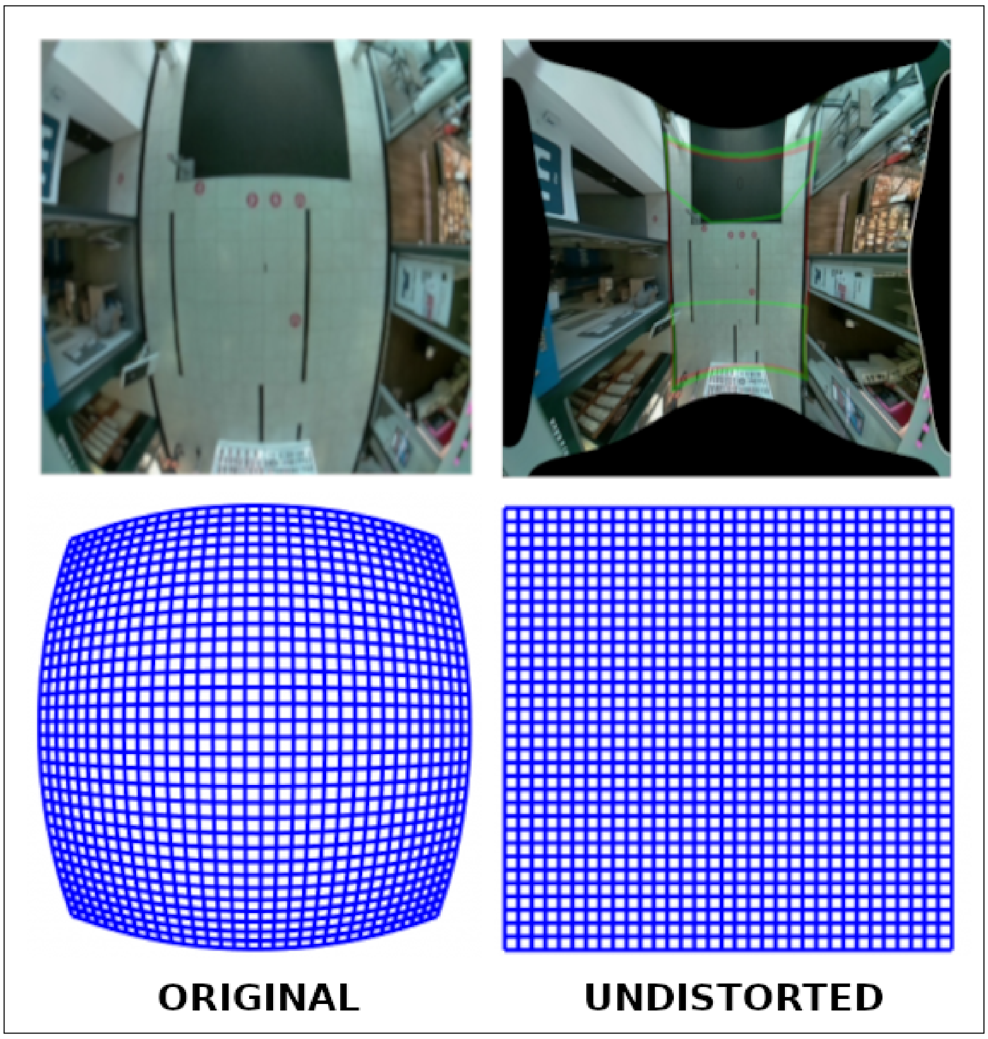

Figure 4.2: Effect of radial distortion - comparison with a square grid.

findHomography function from the OpenCV gets the most accurate transformation when it minimizes the back-projection error described in [42].

With an undistorted image and the homography, we are ready to register all cameras into the floor plan (fig: 4.3). Our algorithm 1 for measuring track distances is complete.

4.1.5 Matching Tracks

With previously defined distance and coordinate transformations, we are ready to describe the matching algorithm 2. A reader may consider using more advanced data structures, but the naive variation suffices for our purposes.

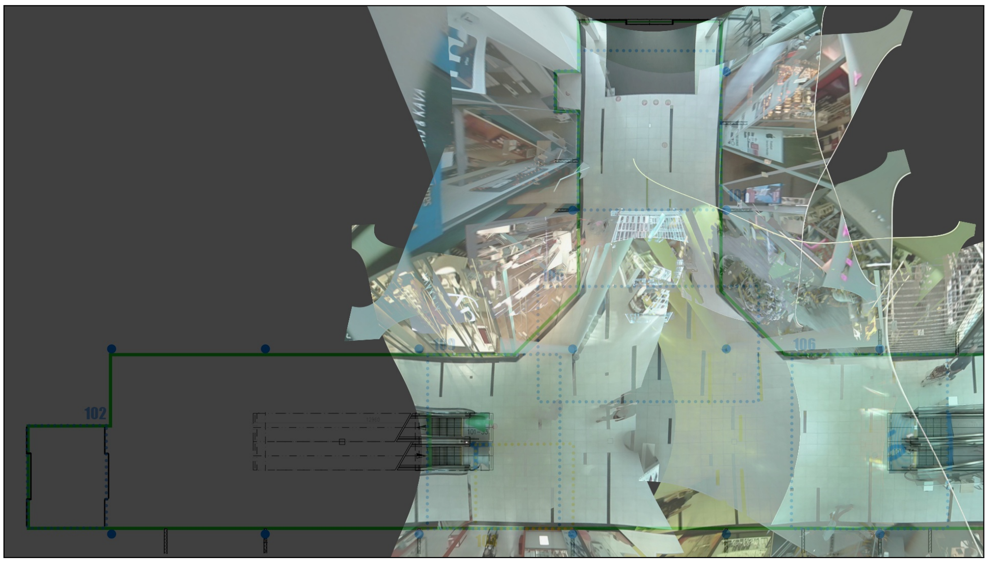

Figure 4.3: Rectified and registered camera view projection into the floor plan.

Algorithm 2: SmartDatasetPositionalMatching

```
   input: Tracks $\mathsf{T}$, Transitions $\mathsf{X}$, max track distance $\varepsilon$, max detection
          timestamp difference $\delta$
1  sets ← []
2  for $x \in \mathsf{X}$ do
3      $C^1_{out} \leftarrow$ {tracks that left cam1 with exit1}
4      $C^2_{in} \leftarrow$ {tracks that entered cam2 with exit2}
5      sets.$append((C^1_{out}, C^2_{in}))$
6      $C^1_{in} \leftarrow$ {tracks that entered cam1 with exit1}
7      $C^2_{out} \leftarrow$ {tracks that left cam2 with exit2}
8      sets.$append((C^1_{in}, C^2_{out}))$
9  matches ← []
10 for set1, set2 ∈ sets do
11     for $t_1, t_2$ : $overlapping\ pairs \in$ set1 × set2 do
12         if $0 \leq PositionalTrackDistance(t_1, t_2, \delta) \leq \varepsilon$ then
13             matches.$append((t_1, t_2))$
   Result: matches
```

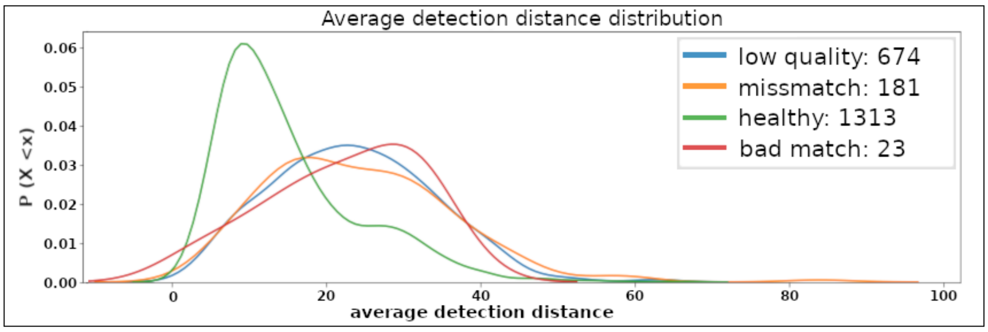

Figure 4.4: Average track distance distribution in SmartDatasetPositional-Matching.

One of the most effective implementations of algorithm 2 uses an interval tree for filtering overlapping pairs at line 11. This approach has a time complexity of $\mathcal{O}(n \log n)$, for our purposes, is the cross product fast enough because we always get a limited number of tracks, and the process ends quickly. Our implementation runs in $\mathcal{O}(n^2)$.

To obtain satisfactory results, we have to fine-tune the upper bound for the maximum detection time difference δ and the maximum track distance ε. For δ is the threshold of approximately 0.25. More in-depth analysis revealed that the best value for ε lies somewhere around the number 15 as depicted in figure 4.4, where we try to separate healthy matches from corrupted remainders.

The biggest flaw of this setup is that we often detect so-called ghost tracks. These are usually relatively short tracks (shorter than 5 detections). The SCT falsely classified various objects as pedestrians. The most frequent false positive objects are car wheels, plants, or posters. We can easily detect defective tracks by their length.

Another problem arises with noisy tracks. When a person passes behind various objects (E.g., plants), it may be unclear where the pedestrian is and how he looks. We want to find a way to measure the quality of a track and find a satisfactory standard. Favorably via CNN classifier (sec: 2.2.1), we already have a method to measure a single detection quality. This technique assesses a value that declares person occurrence probability in an image crop. Our threshold for the classification is 0.5. Further exploration confirms that the best strategy is measuring the average of the lowest 5 scores from both tracks combined. Noisy detections are more likely to get low scores close to the threshold. We use a heuristic, that if the average of the lowest five values is unsatisfactory, we know that the track has poor quality. The upper bound lies a bit under 0.6 (fig: 4.5 serves only for an intuition).

Finally, we have a model for positional matching. Next, we developed tools for manual selection with track previews. We want to have the dataset as clean as possible, which means that there has to be someone who decides whether

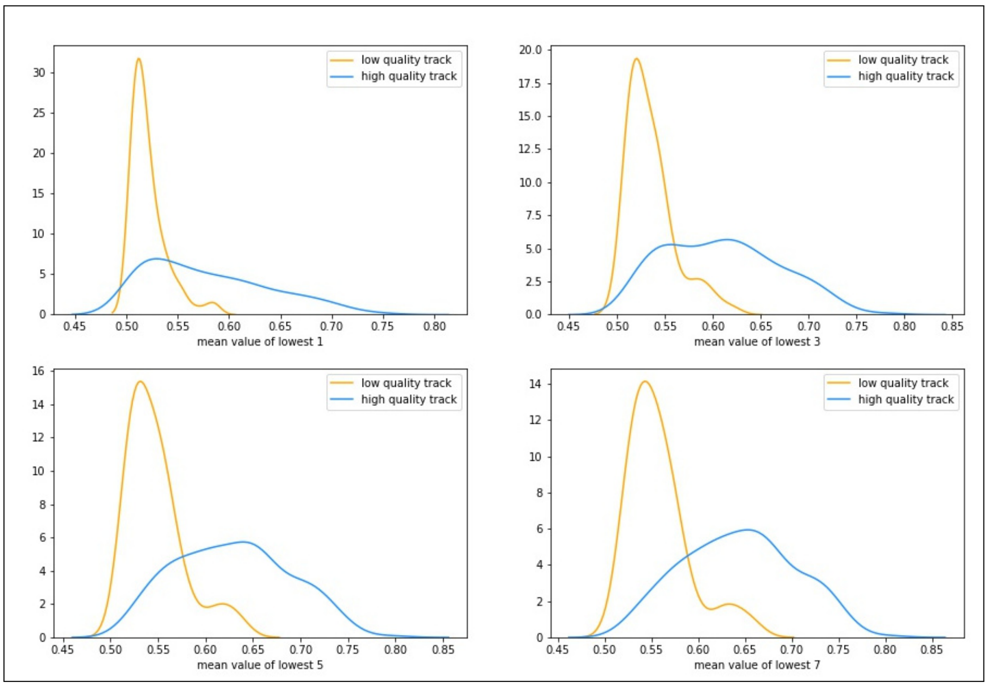

Figure 4.5: Track quality distribution according to the CNN classifier score. The y axis represents a percentage.

a positional match is correct or false. In the end, we got nearly 6500 track pairs from approximately 368 hours of raw footage without heavy labeling.

4.2 Image Descriptor

4.2.1 Neural Network Architecture

Authors in [12] propose to use a pre-trained network for image classification as [25, 26, 27]. We chose Resnet18 as a reliable and straightforward baseline architecture, and its efficiency allows us to conduct more experiments in a short period of time. In future work, we consider using EfficientNet [25]. The default Resnet18 model accepts images with dimensions at least (224, 224, 3). However, our image crops are of size (64, 64, 3). So we slightly changed the network so that it accepts our input.

The output of our model is a vector of 32 numbers. We assumed that using the softmax function for the output would be beneficial because all descriptors would be situated on a unit sphere of 32 dimensions. Experiments revealed the contrary (tab: 5.1.4).

We use adaptive learning rate optimizer Adam [43], gradient clipping [37], and we initialize weights with the Kaiming Normal method [38], also known as He initialization.

All methods were implemented with the use of the PyTorch library [44].

4.2.2 Loss Function

A suitable loss function is arguably a crucial part of the whole training process. Therefore a considerable part of our work consists of experimenting with various loss functions and sampling methods. We treat person re-ID more as a ranking problem, so we use losses suitable for this kind of approach. Our first bet falls upon the triplet loss [33] as it is used in the FaceNet [31]. Next, we tried a different approach with the quadruplet loss [34] and even our cluster loss 4.2.2. And then, we experimented with the contrastive loss [35]. Future work may include compensation of triplet loss drawbacks with the center loss [13].

Triplet Loss

As mentioned in the FaceNet paper [31], we want to ensure that an image crop $x_i^a(anchor)$ is closer to all images $x_i^p(positive)$ of the same person than it is to any crop $x_i^n(negative)$ of any other person.

$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2$ for all triplets from a dataset, where α is a minimal margin between positive and negative pairs. For the embedding of an image x into a d-dimensional space, we use the notaion $f(x) \in \mathbb{R}^d$. The triplet loss function can be expresed as

$$L = \sum_{i}^{N} max\Big(0, \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\Big) \tag{4.2}$$

The authors then appeal to correct triplet selection. They argue the hard triplet selection is crucial for a successful result. These propositions ensure fast convergence because we feed the network with triplets, which significantly violates the constraint.

Cluster Loss

Our objective was to assimilate the training to the real application as much as possible. Input for ICT is a sequence of detections that can be represented as image crops. Our goal is not to learn a similarity between detections. It is to compare the similarity between tracks. Thus, we came with an idea to represent a cluster of positive crops as a single object. We represent each person's cluster as its geometric center $\phi(x) \in \mathbb{R}^d$.

$$\phi(x)_j = \frac{1}{N}\sum_{k=0}^{N}(x_{kj}) \tag{4.3}$$

So the cluster loss function can be expressed as

$$L_c = \sum_{i}^{N} max\left(0, \|\phi(f(x^a)) - \phi(f(x^p))\|_2^2 - \|\phi(f(x^a)) - \phi(f(x^n))\|_2^2 + \alpha\right) \tag{4.4}$$

Where x^a is the anchor cluster, x^p is the positive cluster, and x^n is the negative cluster. Similarly to the triplet loss 4.2.2, we propose using hard triplet mining.

Quadruplet Loss

Authors of the quadruplet loss [34] argue that the triplet loss suffers from a weaker generalization capability. They introduce a method that results in smaller intra-class variation compared to the triplet loss. They added a constraint to the triplet loss, which demands a margin between unrelated image pairs. The authors claim that the quadruplet loss outperforms the triplet loss on the testing dataset.

Contrastive Loss

A Contrastive loss [35] takes only two input images x_i^a and x_i^b. These two crops could be either of a similar or different person encoded in a binary variable Y defined as 0 if x_i^a is similar to x_i^b, 1 otherwise.

$$L = 0.5 * \left(Y * \|x_i^a - x_i^b\|_2^2 + (1 - Y) * max(0, \alpha - \|x_i^a - x_i^b\|_2)^2\right) \tag{4.5}$$

Consider α as a minimal margin between dissimilar data points. This approach pulls positive points inwards to the margin circle, whereas negative pairs are pushed out of the margin zone. A method like this gives us a decision boundary so that if $\|x_i^a - x_i^b\|_2^2 < \alpha$, we assume that the pair is of a similar class.

4.2.3 Training Method

The basic idea was to start with more comprehensive and general samples and then changing the training method so that the problem becomes harder and harder. This approach is inspired by curriculum learning [36].

It is desirable to ensure fast convergence via all to all triplet loss approach described in [31]. Then we train the network with random sampling as a form of warm-up. Random sampling also ensures maximal person variance within a mini-batch, and finally, we use the hard sample selection. We assume that the online hard triplet selection is more difficult within one epoch, so our objective is, to begin with, offline selection and then switch to the online selection.

We implemented each loss function mentioned above with random and hard sample selection implemented in its data loader. We have an option to perform hard sample selection, both offline and online (or semi-online). Offline hard sampling means we compute responses at the beginning of each epoch and online hard sampling means we compute responses at the beginning of each batch.

4.2.3.1 Data Pre-Processing

The average track length is approximately 33 detections. For precise similarity measure, we do not need every detection from the track. We took advantage of geometry knowledge, and we simplified the dataset significantly.

We flipped all image crops so that a pedestrian head lies within the first quadrant of the image. We calculate the rotation with knowledge of a camera's optical angle and relative position of the camera and the pedestrian.

When a pedestrian approaches a camera, he is first observed from the front, then from above, and then back. Each angle is different, and it may be harder to recognize the same person from two different angles. Therefore we use only the first third of the track for training and similarity measure. In future work, we can train multiple descriptors for different angles and consider multiple similarity measurements.

4.3 Inter-Camera Track Assignment Problem

4.3.1 Building an Assignment Problem

Section 2.3.2 outlined the solution presented in [15, 16], where they build a static correspondence matrix. Our method comes with a more dynamic approach. First, we have to build an assignment problem (alg: 3).

The effectiveness of algorithm 3 is essential for a fast construction process. We index tracks by camera_id and exit in our implementation, so the selection can be one in constant time. We select only *PartitionA*, and then we use an **interval tree** for quick candidate selection. We register tracks into the interval tree by the timestamp, and then we search for intersections that intersect track t_a with an extra margin.

The *GetCost* (alg: 4) function respects similar constraints as our fast candidate selection. As an input, *GetCost* accepts two tracks t_a, t_b recorded with cameras C^a, C^b. Let transition x be defined as: $(cam\_id1 = C^a, exit1 = a1, cam\_id2 = C^b, exit2 = b1, duration = d)$

GetCost should return cost even for tracks that leave and enter the same camera if defined in the transition config. However, there are cases when it is not possible to return with the same exit. For example, when a pedestrian leaves the area on escalators, or there are two overlapping cameras. If two cameras overlap, the pedestrian has to arrive in the second camera's field of

Algorithm 3: BuildTrackAssignmentProblem

```
  input: Tracks $\mathsf{T}$, transitions $\mathsf{X}$
1 $\mathsf{G} \leftarrow$ empty graph
2 for $x \in \mathsf{X}$ do
3     $\mathsf{PartitionA} \leftarrow$ {tracks that left cam1 with exit1} $\cup$
        {tracks that left cam2 with exit2}
4     $\mathsf{PartitionB} \leftarrow$ {tracks that entered cam1 with exit1} $\cup$
        {tracks that entered cam2 with exit2}
5     for $t_a, t_b \in \mathsf{PartitionA} \times \mathsf{PartitionB}$ do
6         $\mathsf{cost} \leftarrow GetCost(t_a, t_b)$
7         if $\mathsf{cost} \neq \infty$ then $\mathsf{G}.add\_edge(t_a, t_b, \mathsf{cost})$
8 $\mathsf{res} \leftarrow SplitIntoConnectedComponents(\mathsf{G})$
  Result: res
```

Algorithm 4: GetCost

```
  input: Tracks $t_a$, $t_b$ that either satisfy transition $x$, or they were
         recorded with the same camera
1 $\mathsf{res} \leftarrow \infty$
2 if $(t_a.sink = a1 \wedge t_b.source = b1) \vee (t_b.sink = b1 \wedge t_a.source = a1)$
   then
3     $time_x \leftarrow$ time when the first track ended
4     $time_y \leftarrow$ time when the second track began
5     if $time_y \in < time_x + d - \varepsilon, time_x + d + \varepsilon >$ then
6         $\mathsf{res} \leftarrow \alpha * |time_y - time_x + d| + \beta * AppearanceDistance(t_a, t_b)$
          if $time_y < time_y$ then
7             $\mathsf{res} \leftarrow \mathsf{res} + \delta * PositionalTrackDistance(t_a, t_b)$
  Result: res
```

view, and only then can he return to the first camera. If two cameras overlap, it means, that $time_y \leq time_x$. In such a case, we consider geometry as a factor, and we use the *PositionalTrackDistance* algorithm. Positional distance is a potent tool that can help us significantly. In this thesis, we do not use this distinguisher. The dataset was made solely with the use of geometry so that the result would be heavily biased. The same goes for the time distance since our benchmark (sec: 5.2.1) sets timestamps randomly within a defined time range.

In *GetCost*, we use three fine tuners α, β, δ. They all work as weights that emphasize a particular measurement. Another parameter is ε. It is a margin that defines a time window in which we consider tracks to be relatable. The ε is a constant in our project, but it might carry a different value for each transition in real settings.

Our observation of the track assignment problem revealed that the graph G is relatively sparse. Plus, we can assign each track to a small set of tracks that satisfy strict constraints. It means that G has many independent components. There are at least $|X|$ components (for each transition one) and multiple sub-components. Each one can be computed in parallel, and the result does not depend on other components. When a component closes, we can run the assignment algorithm 5. It means that we can continually perform the ICT, and we do not need to interrupt the process or wait until the end of the day.

4.3.2 Track Assignment Problem Solution

Algorithm 5: TrackAssignment

```
input: Independent components $c_1, c_2, ..., c_n$ of the track assignment
       problem G
1  q ← priority queue (min heap)
2  for connected component $c_i \in$ G do
3      If needed, set undefined edges of $c_i$ to $\infty \Rightarrow c_i$ is fully connected
        bipartite graph
4      matching ← $MinimalWeightMatching(c_i)$
5      for $m \in$ matching do
6          q.heap_push($m$, priority = cost of $m$ in $c_i$)
7  seen ← $dictionary\{x : 0 \mid x \in$ G.$nodes\}$
8  res ← []
9  while $|$q$| \neq 0$ do
10     $m$ ← q.$pop()$
11     if seen[$m$[0]] $< 2 \wedge$ seen[$m$[1]] $< 2 \wedge m.cost \neq \infty$ then
12         res.$append(m)$
13         seen[$m$[0]] ← seen[$m$[0]] + 1
14         seen[$m$[1]] ← seen[$m$[1]] + 1
Result: res, seen
```

The cycle at line 9 (alg: 5) ensures that each track has at most two assignments (one prior and one further in time), we call this process an elimination. However, our dataset is created in such a manner that each track has exactly one assignment. Thus we implemented a slightly different elimination, where the condition at line 11 is as follows: seen[m[0]] $< 1 \wedge$ seen[m[1]] $< 1 \wedge m.cost \neq \infty$. This change will undoubtedly impact test results, but due to the complexity of this work, we did not have enough resources to create a new dataset.

We chose this elimination algorithm since the min-heap prioritize pairs with the lowest cost first, benefiting from the ICT-Descriptor's good relative distance comparison property.

Chapter **5**

Experiments

5.1 Image Descriptor

The crux of our problem is arguably training a well-performing descriptor. Hence we conducted a series of experiments revealing the best possible training method for our task.

5.1.1 Image Descriptor Evaluation Method

Our ambition is to deliver a solution that performs reasonably well in a real environment. To obtain representative results, we designed an evaluation method that resembles the real application.

The *PairPercentileTest* (alg: 6) approximates a normalized rank (percentile) of an anchor-positive distance against all anchor-negative distances.

Algorithm 6: PairPercentileTest

```
  input: Track associations $\mathsf{TA}$
1 $\mathsf{ok} \leftarrow 0$
2 for $t_i, t_j \in \mathsf{TA}$ do
3     $\mathsf{anchor} \leftarrow \mathit{random} \in t_i$
4     $\mathsf{positive} \leftarrow \mathit{random\ positive} \in t_j$
5     $\mathsf{negative} \leftarrow \mathit{random\ negative} \in \mathsf{TA} \setminus (t_i, t_j)$
6     $\mathit{pos\_d} \leftarrow \|\mathsf{anchor} - \mathsf{positive}\|_2^2$
7     $\mathit{neg\_d} \leftarrow \|\mathsf{anchor} - \mathsf{negative}\|_2^2$
8     if $\mathit{pos\_d} < \mathit{neg\_d}$ then $\mathsf{ok} \leftarrow \mathsf{ok} + 1$
  Result: $\mathsf{ok}/|\mathsf{TA}|$
```

We also designed a second test *ClusterPercentileTest*, that works similarly to the *PairPercentileTest* with just a slight change, that we do not measure distances between detections, but distances between clusters with the same metric as in the cluster loss (sec: 4.2.2). However, we discovered that the results of both tests follow the same pattern. The *ClusterPercentileTest* returns slightly better scores thanks to higher robustness towards outliers. We do not consider the Cluster results essential for the training, so we do not present them. However, we mention this method for completeness.

5.1.2 Baseline Configuration

Overview of Training Options

Gradient clipping - Prevention against vanishing/exploding gradients, optimization performs more reasonably near sharp areas of the loss surface.

Descriptor size - It is desirable to obtain relatively short vectors. We experimented with a length of 32 and 64.

Softmax - Situates output on a unit sphere of n dimensions. In our case $n = 32$ or $n = 64$. A universe where descriptors live, has boundaries with the softmax function.

Warm-up - Warming up/pretraining the network may result in better performance.

Flipped detections - All crops are flipped so that the pedestrian head lies within the image's first quadrant.

Track cropping - For learning and similarity measurements, we use only the first third of a track.

Weight initialization - Kaiming weight initialization prevents vanishing/exploding gradient problem.

Batch size - Batch size has a great influence on the training process. We may experiment even with batch size=1.

As a baseline setup, we used the ResNet18 [27] that encodes 64x64x3 images into a vector of length 32. We flipped detections, and we cropped the last two-thirds of all tracks. As suggested in [31], our first epoch is sampled with all possible positive combinations to ensure fast convergence. Another way to ensure fast convergence is to use small batches [39]. We use batches of size 60. As an optimizer, we use Adam [43] with an initial learning rate of 3.5 * 1e-4. We use the Kaiming weight initialization [38] and gradient clipping for the reasons mentioned above.

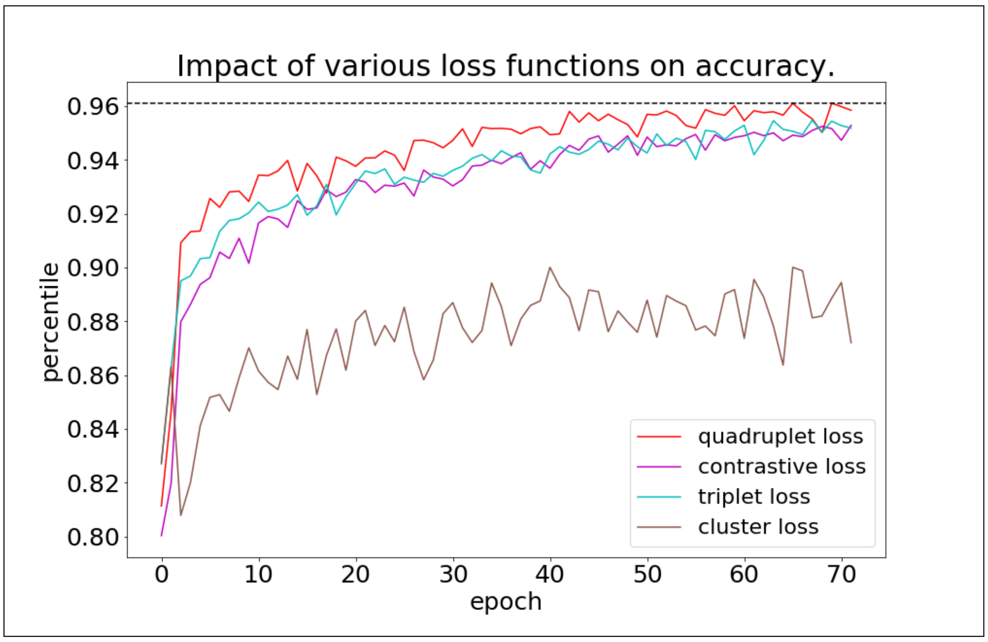

Figure 5.1: Comparison of various loss functions with the baseline configuration. Percentile defined with algorithm 6.

5.1.3 Loss Function Comparison

With subsequent experiments, we aim to discover a suitable loss function. We considered the following options: triplet loss, contrastive loss, quadruplet loss, and cluster loss (sec:4.2.2). Baseline ResNet parameters were pre-trained with all possible positive combinations.

The comparison revealed that the quadruplet loss worked best for our baseline setup. The quadruplet, triplet, and contrastive losses perform similarly in our environment. We expect a shrink of differences in the following training. Nevertheless, due to our resources, we were not able to continue with the training further.

5.1.3.1 Hard Sampling and Noisy Dataset

Sadly the use of hard sample mining leads to lower performance than random sampling. We assume that the cause is noise in our Smart Dataset (sec: 4.1). We obtained the dataset without heavy labeling, and it allowed us to generate an outstanding amount of images, however, with lower quality.

We noticed two different types of defects in our dataset. The first one, we call noise. These are detections that do not contain any person. The second one we call mismatch. It is a false detection assignment that usually propagates until the end of the track. With various heuristics, we were

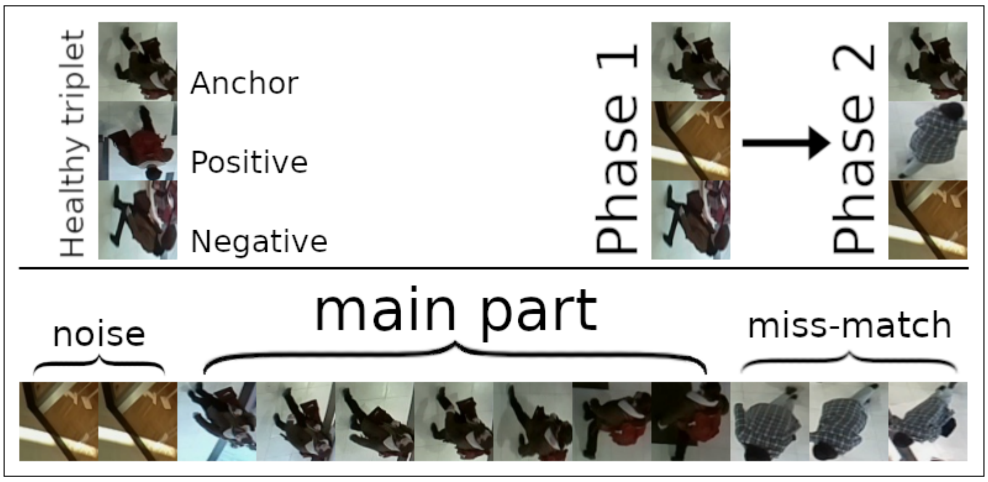

Figure 5.2: Example of noisy track and it's influence on hard sampling.

able to detect some malicious detections. Nevertheless, the majority of errors remained undetected.

In figure 5.2, we depicted two phases that occur during hard sample training. The noise is distant from all detections throughout the dataset at the beginning. In the first phase, the hard positive sampling pulls these noises closer. Therefore the noise detections are generally close to all other detections. This problem escalates in a cycle of pushing noises away with hard sampling and pulling noises in positive sampling. In the second phase, noises are closer to anchors than miss-matches are. Hence we minimize distances between two different pedestrians as is depicted in figure 5.3. A similar principle applies to the cluster loss, but in a slightly different manner. Among others, the cluster loss minimizes the distance of outliers from the center. Moreover, outliers are often noisy detections. Another reason that causes lower performance might be that we did not found a suitable training method.

By the time we started experimenting, supervisor Ing. Filip Naiser considerably improved the SCT. We had an option to repair the dataset and eliminate the majority of noises. However, we decided to eliminate more significant flaws first and execute more experiments due to the lack of time.

5.1.4 Training Method

Table 5.1.4 revealed, that the gradient clipping has no influence on the accuracy. Surprisingly enough, we did not meet much success with the softmax function, even with a longer descriptor vector. On the other hand, the warm-up proved to be beneficial.

Because of how time-consuming the training is, experiments from table 5.1.4 were conducted with just the quadruplet loss. However, we made a quick

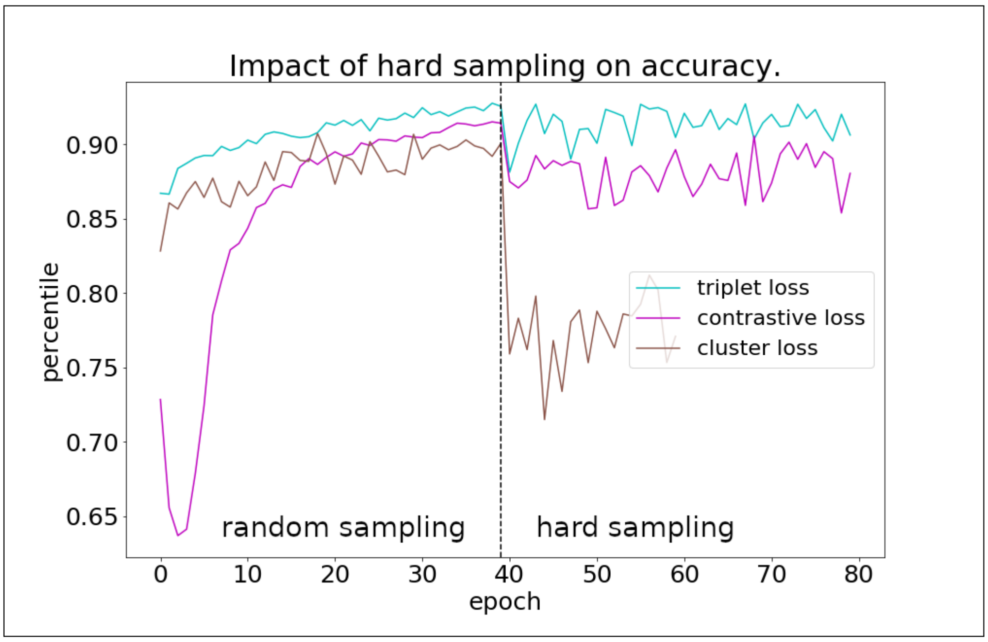

Figure 5.3: Impact of hard sample mining on accuracy. The dataset is pre-processed (sec: 4.2.3.1).

comparison of how training options influence other loss functions (fig: 5.4). Note that figure 5.4 is only to acquire intuition over the matter. Nevertheless, we can notice recurring patterns similar to table 5.1.4.

5.1.5 t-SNE Projection

One way to get insight to multidimensional data is by reducing the number of dimensions. The t-SNE [45] is a practical non-linear technique for visualizing large-scale and high-dimensional data.

Figure 5.5 outlines the descriptors of two tracks that contain the same person concentrate in a cluster. It seems that clusters are not mixed up, so we should differentiate them. It turns out that descriptors of two tracks with the same person are relatively mixed, which is a desirable outcome. It suggests that in latent space, they concentrate on one area evenly.

Summary

Experimental results revealed that the baseline method with warmup performed the best. We were able to exceed the 0.97 percentile on the validation dataset with just random sampling. We believe that it is possible to outperform our result on a cleaner dataset with hard sample mining and various improvements (i.e., loss function, different network architecture, augmenta-

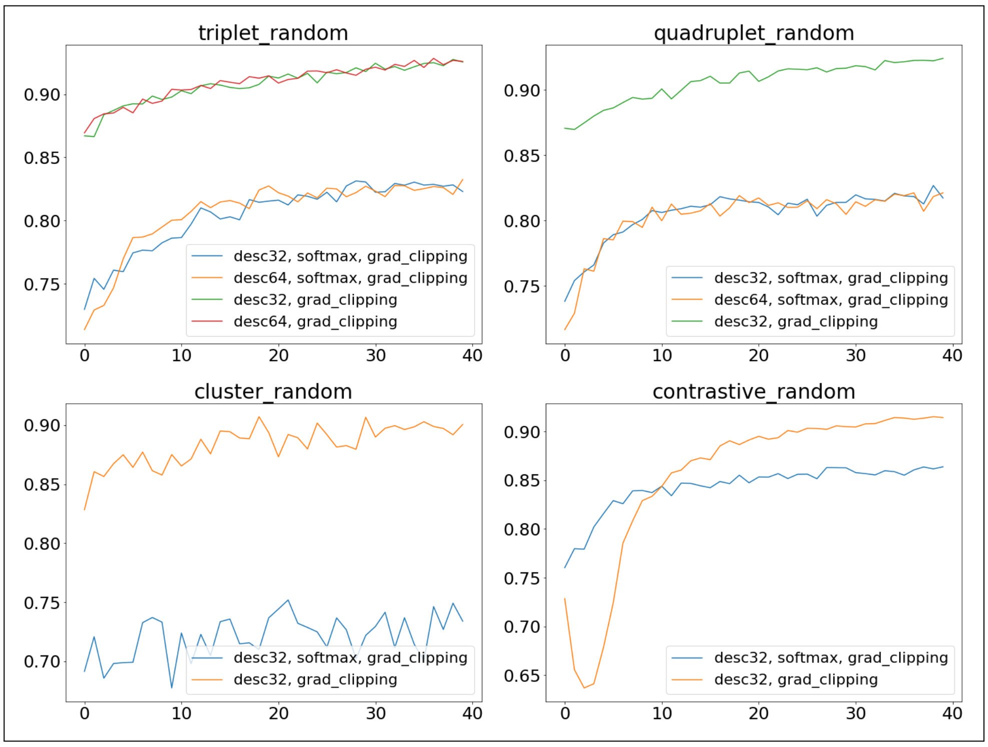

Figure 5.4: Performance of loss functions with different training options on the preprocessed dataset (sec: 4.2.3.1). The y axis represents a percentile.

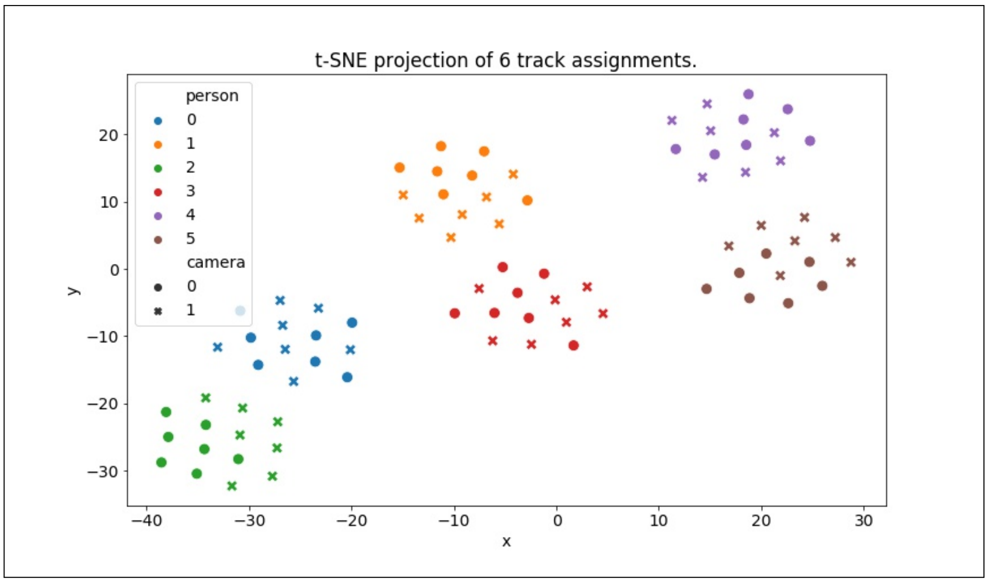

Figure 5.5: t-SNE projection of random tracks with our best descriptor. (percenitle: 0.97)

model	epoch 5	epoch 20	epoch 40
baseline	0.922	0.941	0.950
	positive effect on accuracy		
warm-up	-0.025	-0.002	+0.014
	neutral effect on accuracy		
no gradient clipping	-0.003	+0.004	+0.004
desc64	-0.002	-0.003	-0.002
no weight init	-0.012	-0.014	-0.006
	negative effect on accuracy		
softmax64	-0.182	-0.128	-0.119
softmax32	-0.116	-0.088	-0.078
full track	-0.020	-0.014	-0.015
unflipped	-0.007	-0.011	-0.004

Table 5.1: Comparison of training methods with quadruplet loss. Each row represents a change of the baseline configuration.

tion, or tricks described in [13, 12]). Moreover, we found the influence of crop flipping and track cropping beneficial. Therefore we look forward to better data preprocessing.

In the future, we want to estimate the pedestrian's perspective (front, back, or top view). We want to train separate descriptors for each perspective and imply in the track association process.

On the other hand, we could not prove the effect of gradient clipping [37] and Kaiming weight initialization [38] on accuracy. However, we believe in their importance since we experienced exploding gradient issues, especially during hard sampling methods. Longer descriptors also did not prove to be useful.

The softmax function resulted in significantly lower performance both in its short and long descriptor form. We assume that it is possible to reach reasonable results with the softmax function. However, we did not found a suitable training configuration.

Our best configuration (baseline with warm-up) had the best results. Therefore we call this setup an ICT-Descriptor, and we use it in the following results.

5.2 Inter-Camera Track Assignment Problem

5.2.1 ICT-Evaluation Method

ICT-Benchmark

To simulate a real environment, we designed a benchmark that randomly distributes track pairs in a time interval. A track is a sequence of chronologically sorted detections. The time interval between detections is 0.2 seconds. The benchmark accepts a set of assigned tracks (crossings) and duration d so that the longest crossing is shorter than d. Each track pair respect a transition defined between two cameras with a transition time. For each pair of tracks t_i, t_j from cameras C_i, C_j with transition-duration x_d, we execute following steps:

1. estimate crossing duration $c_d = (|t_i| + |t_j|) * 0.2 + x_d$
2. randomly pick t_i beginning $t^0 \in \langle 0, d - c_d \rangle$
3. calculate transition beginning $t^1 = t_0 + |t_i| * 0.2$
4. with respect to the order, linearly distribute detections from t_i in interval $\langle t^0, t^1 \rangle$, and detections from t_j in interval $\langle t^1 + x_d, c_d \rangle$

For duration d, e chose a value so that each track has approximately four assignment candidates.

Track Assignment Test

As an evaluation metric, we chose the following formulas.

TP = correctly assigned pairs

FP = falsely assigned pairs

FN = missing assignments

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

The conception of the project puts emphasis mainly on *Precision*. We are not concerned about unassigned tracks as much as we are about falsely assigned tracks. Nevertheless, we believe in the importance of this rate, so we measure it as *Recall*.

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

5.2.2 Track Distance Metric

Distance metric between two tracks can make a big difference in the final assignment. Therefore we analyzed the influence of different measuring techniques on Precision. We considered six basic techniques enumerated in table 5.2.2. The Euclidean distance between centroids of tracks outperformed other variants consistently.

metric	precision	recall
l2 centroids	0.854	0.459
cross product mean	0.841	0.455
min5	0.827	0.451
min	0.827	0.451
ward	0.820	0.449
max	0.774	0.435

Table 5.2: Influence of different metrics on the inter-camera assignment problem.

Summary

The track assignment algorithm analyzes the correspondence graph and runs the Hungarian algorithm for each independent component. It notably reduces the computation time and allows us to reconstruct trajectories efficiently. In the future, we would like to replace the Hungarian algorithm with an algorithm utilizing a high sparsity of our correspondence graph. We expect that this might lead to dramatic speed improvements.

5.3 ICT-Results

We successfully designed and implemented the *inter-camera multi-object tracking*. We collected and annotated our dataset in the Krakov shopping center. To our best knowledge, it is the only available top-view inter-camera tracking dataset (6 cameras, XY track pairs). On this dataset, we get the following results:

PairPercentileTest

In PairPercentileTest (alg: 6), ICT-Descriptor significantly outperformed the baseline Color Names descriptor (tab: 5.3).

desc variant	percentile
ICT-desc	0.97
Color Names	0.69
random desc	0.50

Table 5.3: Comparison of the ICT-Descriptor, Color Names descriptor, and randomly generated descriptor in the *PairPercentileTest*.

ICT-Benchmark

In the *inter-camera multi-object tracking*, we achieved reasonable results (precision: 0.83) with the ICT-Descriptor on the ICT-Benchmark (tab: 5.4).

desc variant	precision	recall
ICT-desc	0.83	0.45
Color Names	0.38	0.26
random desc	0.18	0.15

Table 5.4: Comparison of the ICT-Descriptor, Color Names descriptor, and randomly generated descriptor in the ICT-Benchmark (sec: 5.2.1).

Chapter 6

Conclusion

The author single-handedly implemented the Smart Dataset, collected the dataset, developed, tested, and tuned the whole ICT pipeline within ten months, which took nearly 950 hours of work.

We recorded 368 hours of video in the Krakov shopping center in Prague. We introduced the Smart Dataset, an algorithm for annotating large scale footage without heavy labeling. With the Smart Dataset, we associated approximately 13000 tracks into 6453 pairs from different cameras. Our dataset consists of 436043 image crops.

We trained a Siamese neural network useful for obtaining both SCT and ICT descriptors. We presented a cluster loss function suitable for pedestrian Re-Identification. With careful study of various optimization techniques, we were able to maximize the network's performance (tab: 5.3) despite having a noisy dataset, uncalibrated RGB cameras, and low resolution.

We presented a dynamic algorithm for the track assignment problem using the Hungarian algorithm and min-heap. We implemented the ICT-benchmark that simulates an environment for testing *inter-camera multi-object tracking.*

Finally, we have developed and implemented an effective algorithm for the *inter-camera multi-object tracking problem.* We search for independent components in a correspondence graph, which allows continuous and parallel computation. Furthermore, the whole ICT pipeline performs well on the ICT-Benchmark (tab: 5.4).

We believe that future work increases performance even further. In particular, more sophisticated training methods, different descriptors for approaching and leaving pedestrians, better network architecture, a cleaner dataset for effective hard sample mining, image augmentation, spatio-temporal predictions and motion models, the influence of improved detector in SCT, the substitution of the Hungarian algorithm with better, a faster algorithm capable of using sparse graphs, or changing the assignment algorithm as a whole.

Chapter 7

Acronyms

AI	Artificial intelligence.
CNN	Convolutional neural network.
iC	iC Systems.ai, s.r.o..
ICT	inter-camera tracking.
SCT	single-camera tracking.

Bibliography

1. RANSBOTHAM, Sam; KIRON, David; GERBERT, Philipp; REEVES, Martin. Reshaping business with artificial intelligence: Closing the gap between ambition and action. *MIT Sloan Management Review*. 2017, vol. 59, no. 1. Available also from: `https://search.proquest.com/openview/83d554491afeb2435c6c2e386821c60c/1`.
2. WANG, Chien-Yao; BOCHKOVSKIY, Alexey; LIAO, Hong-Yuan Mark. *Scaled-YOLOv4: Scaling Cross Stage Partial Network*. 2020. Available from arXiv: `2011.08036 [cs.CV]`.
3. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*. 2015, pp. 91–99. Available also from: `http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf`.
4. HASTIE, Trevor; ROSSET, Saharon; ZHU, Ji; ZOU, Hui. Multi-class adaboost. *Statistics and its Interface*. 2009, vol. 2, no. 3, pp. 349–360. Available from DOI: `https://dx.doi.org/10.4310/SII.2009.v2.n3.a8`.
5. VAN DE WEIJER, Joost; SCHMID, Cordelia; VERBEEK, Jakob; LARLUS, Diane. Learning color names for real-world applications. *IEEE Transactions on Image Processing*. 2009, vol. 18, no. 7, pp. 1512–1523. Available also from: `https://ieeexplore.ieee.org/abstract/document/4982667`.
6. NOVAK, Carol L; SHAFER, Steven A, et al. Anatomy of a color histogram. In: *CVPR*. 1992, vol. 92, pp. 599–605.
7. HE, Anfeng; LUO, Chong; TIAN, Xinmei; ZENG, Wenjun. A Twofold Siamese Network for Real-Time Object Tracking. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

8. KUHN, Harold W. The Hungarian method for the assignment problem. *Naval research logistics quarterly*. 1955, vol. 2, no. 1-2, pp. 83–97.

9. WELCH, Greg; BISHOP, Gary, et al. *An introduction to the Kalman filter*. Citeseer, 1995.

10. MONTCALM, Trevor; BOUFAMA, Bubaker. Object inter-camera tracking with non-overlapping views: a new dynamic approach. In: *2010 Canadian Conference on Computer and Robot Vision*. 2010, pp. 354–361.

11. RISTANI, Ergys; TOMASI, Carlo. Features for multi-target multi-camera tracking and re-identification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6036–6046.

12. ALMAZAN, Jon; GAJIC, Bojana; MURRAY, Naila; LARLUS, Diane. Re-id done right: towards good practices for person re-identification. *arXiv preprint arXiv:1801.05339*. 2018.

13. LUO, Hao; GU, Youzhi; LIAO, Xingyu; LAI, Shenqi; JIANG, Wei. Bag of Tricks and a Strong Baseline for Deep Person Re-Identification. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2019.

14. LEE, Young-Gun; TANG, Zheng; HWANG, Jenq-Neng; FANG, Zhijun. Inter-camera tracking based on fully unsupervised online learning. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 2607–2611.

15. CAI, Yinghao; MEDIONI, Gerard. Exploring context information for inter-camera multiple target tracking. In: *IEEE Winter Conference on Applications of Computer Vision*. 2014, pp. 761–768.

16. KUO, Cheng-Hao; HUANG, Chang; NEVATIA, Ram. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In: *European Conference on Computer Vision*. 2010, pp. 383–396.

17. JAVED, Omar; RASHEED, Zeeshan; SHAFIQUE, Khurram; SHAH, Mubarak. Tracking across multiple cameras with disjoint views. In: *Computer Vision, IEEE International Conference on*. 2003, vol. 3, pp. 952–952.

18. JAVED, Omar; SHAFIQUE, Khurram; RASHEED, Zeeshan; SHAH, Mubarak. Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. *Computer Vision and Image Understanding*. 2008, vol. 109, no. 2, pp. 146–162.

19. DEMAINE, Erik D; EMANUEL, Dotan; FIAT, Amos; IMMORLICA, Nicole. Correlation clustering in general weighted graphs. *Theoretical Computer Science*. 2006, vol. 361, no. 2-3, pp. 172–187.

20. LOWE, David G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*. 2004, vol. 60, no. 2, pp. 91–110.

21. INDYK, Piotr; MOTWANI, Rajeev. Approximate nearest neighbors: towards removing the curse of dimensionality. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 1998, pp. 604–613. Available also from: `https://doi.org/10.1145/276698.276876`.

22. GILBERT, Andrew; BOWDEN, Richard. Incremental, scalable tracking of objects inter camera. *Computer Vision and Image Understanding*. 2008, vol. 111, no. 1, pp. 43–58.

23. REINHARD, Erik; ADHIKHMIN, Michael; GOOCH, Bruce; SHIRLEY, Peter. Color transfer between images. *IEEE Computer graphics and applications*. 2001, vol. 21, no. 5, pp. 34–41.

24. SERMANET, Pierre; EIGEN, David; ZHANG, Xiang; MATHIEU, Michaël; FERGUS, Rob; LECUN, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*. 2013.

25. TAN, Mingxing; LE, Quoc V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*. 2019.

26. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*. 2017, vol. 60, no. 6, pp. 84–90.

27. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

28. LIN, Tsung-Yi et al. *Microsoft COCO: Common Objects in Context*. 2015. Available from arXiv: `1405.0312 [cs.CV]`.

29. *papers with code - coco test-dev benchmark (object detection)*. 2020. Available also from: `https://paperswithcode.com/sota/object-detection-on-coco"`.

30. BROMLEY, Jane; BENTZ, James W; BOTTOU, Léon; GUYON, Isabelle; LECUN, Yann; MOORE, Cliff; SÄCKINGER, Eduard; SHAH, Roopak. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*. 1993, vol. 7, no. 04, pp. 669–688.

31. SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

32. YE, Mang; LI, Jiawei; MA, Andy J; ZHENG, Liang; YUEN, Pong C. Dynamic graph co-matching for unsupervised video-based person re-identification. *IEEE Transactions on Image Processing*. 2019, vol. 28, no. 6, pp. 2976–2990.

33. DONG, Xingping; SHEN, Jianbing. Triplet Loss in Siamese Network for Object Tracking. In: *The European Conference on Computer Vision (ECCV)*. 2018. Available also from: `http://openaccess.thecvf.com/content_ECCV_2018/html/Xingping_Dong_Triplet_Loss_with_ECCV_2018_paper.html`.

34. CHEN, Weihua; CHEN, Xiaotang; ZHANG, Jianguo; HUANG, Kaiqi. *Beyond triplet loss: a deep quadruplet network for person re-identification*. 2017. Available from arXiv: `1704.01719 [cs.CV]`.

35. HADSELL, Raia; CHOPRA, Sumit; LECUN, Yann. Dimensionality reduction by learning an invariant mapping. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006, vol. 2, pp. 1735–1742.

36. BENGIO, Yoshua; LOURADOUR, Jérôme; COLLOBERT, Ronan; WESTON, Jason. Curriculum learning. In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48.

37. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron; BENGIO, Yoshua. *Deep learning*. MIT press Cambridge, 2016. No. 2.

38. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

39. WILSON, D Randall; MARTINEZ, Tony R. The general inefficiency of batch training for gradient descent learning. *Neural networks*. 2003, vol. 16, no. 10, pp. 1429–1451.

40. AXIOTIS, Kyriakos; MADRY, Aleksander; VLADU, Adrian. Circulation control for faster minimum cost flow in unit-capacity graphs. *arXiv preprint arXiv:2003.04863*. 2020.

41. BUHRMESTER, Vanessa; MÜNCH, David; BULATOV, Dimitri; ARENS, Michael. Evaluating the Impact of Color Information in Deep Neural Networks. In: MORALES, Aythami; FIERREZ, Julian; SÁNCHEZ, José Salvador; RIBEIRO, Bernardete (eds.). *Pattern Recognition and Image Analysis*. Cham: Springer International Publishing, 2019, pp. 302–316. ISBN 978-3-030-31332-6.

42. BRADSKI, G. *Camera Calibration and 3D Reconstruction*. 2020. Available also from: `https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html`.

43. KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014.
44. PASZKE, Adam et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. Available also from: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.
45. MAATEN, Laurens van der; HINTON, Geoffrey. Visualizing data using t-SNE. *Journal of machine learning research*. 2008, vol. 9, no. Nov, pp. 2579–2605.