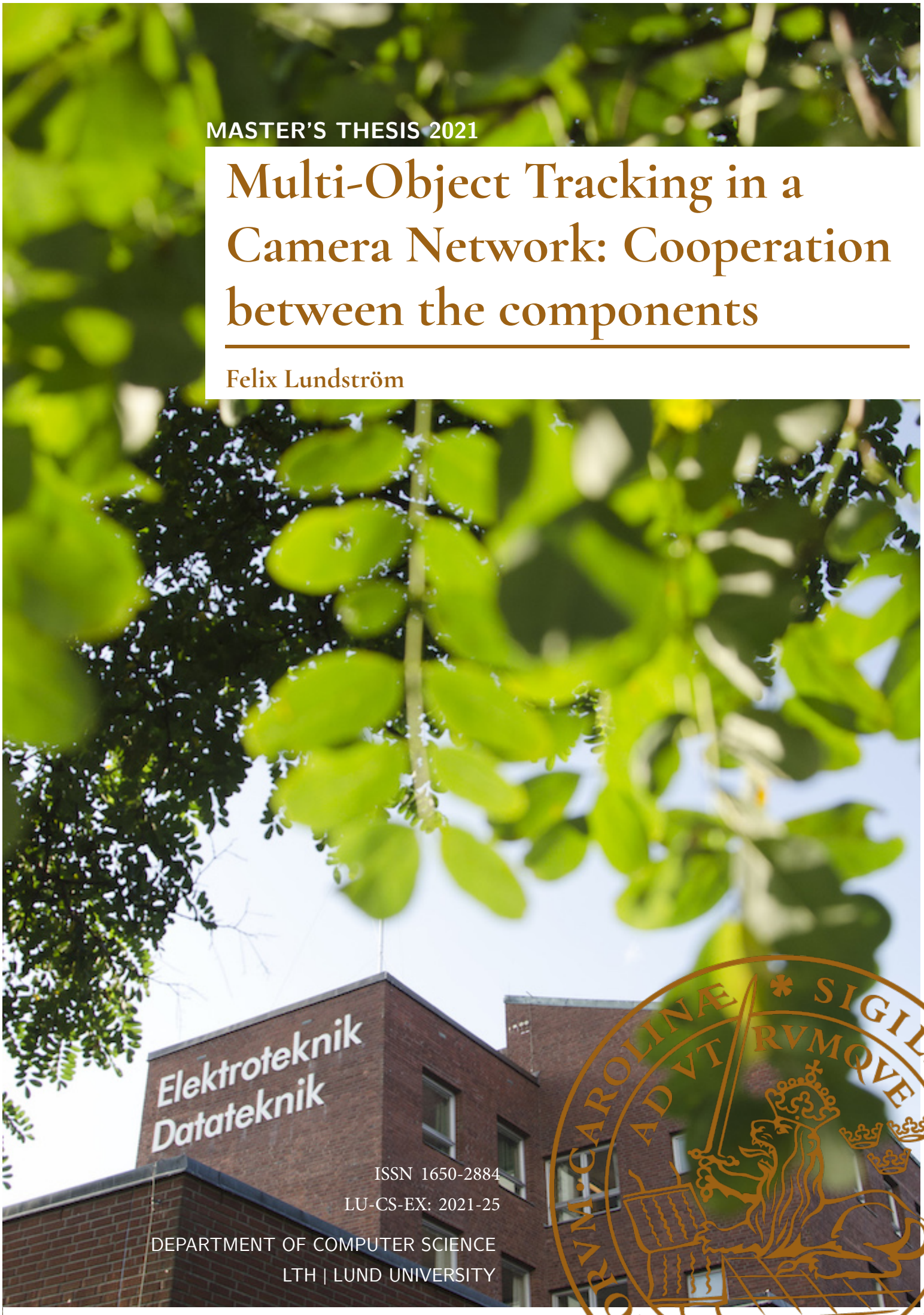

MASTER'S THESIS 2021

Multi-Object Tracking in a Camera Network: Cooperation between the components

Felix Lundström

ISSN 1650-2884
LU-CS-EX: 2021-25

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2021-25

Multi-Object Tracking in a Camera Network: Cooperation between the components

Spårning av flera objekt i ett kameranätverk: Samarbetet mellan komponenterna

Felix Lundström

Multi-Object Tracking in a Camera Network: Cooperation between the components

Felix Lundström
fe5713lu-s@student.lu.se

June 22, 2021

Master's thesis work carried out at Axis Communications AB.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Oskar Persson, oskar.persson@axis.com
Mikael Andreen, mikael.andreen@axis.com
Simon Molin, simon.molin@axis.com

Examiner: Elin Anna Topp, elin_anna.topp@cs.lth.se

Abstract

This thesis covers the topic of multi-target multi-camera tracking. Tracking objects across cameras can for example be used to prevent or analyse recordings of crime scenes. The solution can be divided into three different components: detection, tracking and re-identification. Detecting the objects, tracking them within each view and re-identifying appearing objects against a database. We focus on real-time tracking with open world re-identification. This means that we are not able to use future frames when tracking targets and we also need to figure out if an appearing object has been seen or not. Re-identification turned out to be the most difficult problem since objects might appear in different poses, lights and distances. We investigated if we can solve this problem by using principal component analysis (PCA). PCA is used to transform the features into stable ones over time. We did not find any improvements by adding PCA to the system.

Keywords: Multi-target multi-camera tracking, tracking, re-identification, real-time, open world, principal component analysis

Acknowledgements

First, we would like to thank Axis Communications AB for giving us the opportunity to conduct this thesis. We would like to thank our supervisors at Axis, Oskar Persson, Simon Molin and Mikael Andreen for providing us with invaluable help and discussions.

We would also like to thank our supervisor at the university, Pierre Nugues, for helping us to stay in the right direction to complete this thesis.

At last, we would also like to thank everybody involved in the FLET21 dataset. This gave us the possibility to test our tracking system on different scenarios.

Contents

Chapter 1
Introduction

Real-time object tracking between multiple cameras is an increasingly growing area (Ciaparrone et al., 2020). The difficulties with tracking objects in real-time between multiple cameras are varied and hard to overcome. But the technology and hardware recently have necessitated an investigation into if it is practically possible today. Among other things, we will look into the components of a multi-target multi-camera tracking (MTMCT). MTMCT consists of the following components: detection, tracking and re-identification, which is described further in Chapter 2. We will investigate how they work, both separately and together. In addition, we will analyse why the various approaches of the components interact differently with each other.

1.1 Context

Tracking objects across multiple cameras by analysing and processing videos is a widely used technique in many different application areas. It can be used to track lost kids in a mall (Abraham et al., 2021), monitor and analyse the environment in autonomous cars (Yurtsever et al., 2020) or crime scenes (McLean et al., 2013) or even to detect strange behaviour to prevent criminal activity (McLean et al., 2013). Trained machine learning models for detecting and tracking objects in videos have been used for a long time and have been dominated by deep learning models and neural networks for a while.

In recent years, we have reached sufficient performance with deep learning approaches to also be able to track objects between different cameras, both overlapping and non-overlapping. However, such approaches require significant resources. In this thesis, our task is to determine, whether it is viable to use in practice with reasonable memory usage and complexity, but still efficient enough to use in real-time (30 FPS) applications. By reasonable we mean that a general computer can run the application.

1.2 Case description

Axis Communications AB (which we will refer to as Axis from now on) was founded in 1984 and first focused on network technology to connect devices like printers, storage and scanner servers, etc. In 1996, Axis launched the world's first network camera, named AXIS Neteye 200 (Axis, 2021). After this breakthrough, Axis has mainly been focusing on developing network cameras.

Currently, Axis spends a lot of resources on writing algorithms and models for video processing in their cameras. Techniques like object detection and similar are now continuously developed at Axis, which also includes a lot of research to keep being on the forefront of the field. The interest of tracking people through multiple cameras has risen lately for different uses. One thing that is important to mention is that there exists a wide variety of algorithms to accomplish the same goal in every case. Most of the state of the art algorithms do now involve artificial intelligence such as deep learning and neural network models.

In this thesis, we investigated whether it is possible to track and re-identify persons within several cameras in real-time. This includes detecting all pedestrians within each camera, tracking them through the view and connecting each person to a unique identifier between all the cameras in the network. More specifically, we will use existing models for each component and connect them into a complete system. Figures 1.1 and 1.2 show examples of how two different cameras re-identify two persons at different times. Both figures contains three images, where the left most is an image of person 1, second of person 2 and the third with both person 1 and 2.

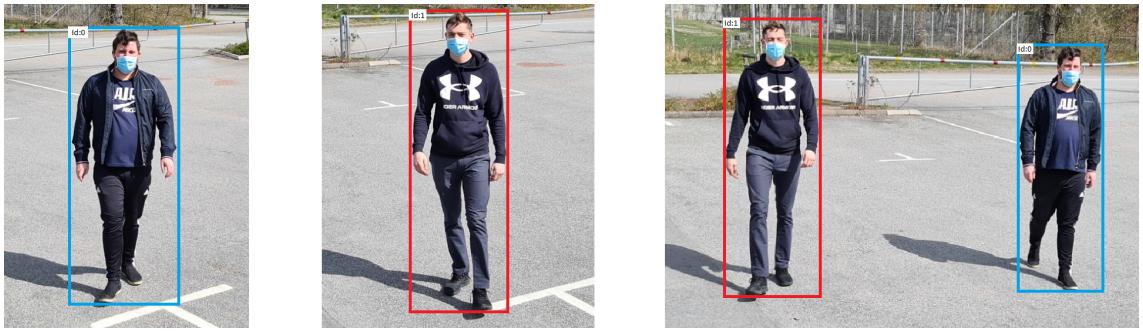

Figure 1.1: Example of frames from camera 1

Figure 1.2: Example of frames from camera 2

We considered some typical use cases provided by Axis to narrow down the scope of this task. This was essential for completing this thesis, otherwise its evaluation would be incomplete due to the time and resources available. The two most common cases are:

1. multi-channel products with 360ºview, mounted on street lights;
2. linked cameras in indoor corridors.

Furthermore we have limited the work to be within a real-time system that will be run on either the camera product or a general-purpose computer, i.e. no supercomputers. We are not allowed to assume that we know the camera topology, which means we can not use it while tracking between cameras.

1.3 Purpose

The purpose of this thesis is to analyse the different ways of detecting, tracking and re-identifying humans. Furthermore we will investigate how different algorithm for each component of the MTMCT affect the other components. This will provide a greater understanding of how the different components cooperate in the complete system.

1.3.1 Research questions

We can divide our problem into the following three questions:

1. Which commonly used components (detection, tracking and re-identification) can we apply to solve our task?
2. Why does specific combinations of components work better than others?
3. How is the different performance aspects affected when only comparing the most stable features to re-identify objects?

The motivation for **RQ1** is grounded in that we need to find existing components which can be used in the implementation.

Different models for each component prioritise different aspects of performance. This means that they will cooperate differently with the other components. We have to understand why specific setups are efficient and others are not. **RQ2**'s aim is to answer this.

When re-identifying objects, they might appear in different lights, angles, poses or distances. A person looks differently depending on these aspects. For example, if the re-identification model looks at colours, they may vary due to different light. We will try to solve this by finding and comparing the features that remain most stable through these different setups. This is the motivation for **RQ3**.

1.3.2 Approach

To begin with, we will start by researching the subject to see which methods are already discovered and which algorithms seem to be the most suitable. We will then continue by setting up a baseline version with existing models for detection, tracking and re-identification.

When we have a working baseline where everything works out together, we will create an evaluation pipeline to evaluate the system. When we are at this stage, we will analyse the results, upgrade the system and assess the design. This process will then be repeated until satisfaction or the time limit of the thesis has been reached.

1.4 Contributions

This thesis has been carried out in cooperation with Eric Turesson from Blekinge Institute of Technology. The work has been divided so that the first two research questions are similar while the third is completely separate and done individually.

During the work, all programming has been done by pair programming, where we have equally shared the programming and criticising role. All the analyses have been done together. The reason is that we believe that discussion brings further validity to the analysis since we can see different viewpoints and taking advantage of our different skills. We do not share the same thesis report, but there are similar parts in the reports. We wrote the introduction, results, discussion and conclusion individually. The result and discussion chapters are based on the same evaluations and results. The related works and method chapters are close to being identical with small differences. In the related works and method, we have tried to ensure that both of us have done the same amount of work. The process has been that we take turns writing and examining. The content of related works and method are not the exact same due to the fact that the third research question is different.

1.5 Outline

The thesis is structured in the following way: Chapter 2 introduces the reader to the tracking, detection and re-identification tasks and other relevant information. The metrics and datasets can be found in chapter 3. Chapter 4 details how the thesis was conducted. Chapter 5 presents the thesis's objective result and the objective analysis of the result. Chapter 6 presents the result and the significance of our results. In Chapter 7, you can find the conclusion of the thesis and discussions on future works for the area.

Chapter 2
Related work

MTMCT consists of three components: **detection**, **tracking** and **re-identification**:

1. Firstly, we need to search the frames for objects, which is handled by the **detection model**.

2. Secondly we need to keep track of each object within a sequence of frames, providing it with an unique id. To accomplish this task, a **tracking algorithm** is utilised.

3. Thirdly, we need to create connections between the targets across multiple cameras. This is achieved by using a **re-identification model**, which compares each target to a database with previously seen objects.

2.1 Detection

Detection algorithms identify all interesting objects of a specific class or multiple classes within an image. The detection algorithm takes an image as input, and the output is a list of bounding boxes discovered in the image. Each of these bounding boxes contains an object which the algorithm is trained to find. The bounding box is often a set of pixel coordinates and a confidence value on how sure the algorithm is that the bounding box contains a specific object. The convolutional neural network (CNN) is the state-of-the-art approach to deal with the detection task which outperforms previous methods. Some widespread CNN object detection methods are R-FCN (Dai et al., 2016), Faster R-CNN (Ren et al., 2017), SSD (Liu et al., 2015) and YOLOv3 (Redmon and Farhadi, 2018). A list of the detection algorithms which this thesis uses can be found in the following subsections.

2.1.1 You Only Look Once (YOLOv3)

YOLOv3 (Redmon and Farhadi, 2018) is a fast detection algorithm with decent accuracy, since real-time is an important factor in this thesis this makes YOLOv3 an incredibly suitable choice.

YOLOv3 capitalizes on previous work by Redmon and Farhadi (2017). YOLOv3 tries to predict coordinates and size parameters for its bounding boxes: t_x, t_y, t_w, t_h. To deal with boxes that are close to the boundaries of the image, offsets of c_x and c_y are used together with the previous bounding boxes prior width p_w and height p_h. This then creates the following formulas to calculate the bounding box's coordinates. e below is Euler's number.

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{2.1}$$

For each box, a class prediction is applied and determines a label for the box. YOLOv3 uses a feature extractor to extract the features and then using the features, it tries to predict a label.

YOLOv3 has the ability to classify a wide spectrum of objects. Examples could be cats, dogs, humans and many more. We have decided to only focus on humans so we have limited YOLOv3's amount of labels to only humans or, as YOLOv3 calls them, persons.

2.1.2 Faster Region-based Convolutional Neural Networks

Faster Region-based Convolutional Neural Networks (FRCNN) (Ren et al., 2017) are an improvement of the previously developed RCNN (Girshick et al., 2013). The goal of creating FRCNN is that it needs to increase its processing speed while retaining its accuracy.

FRCNN comprises two modules: a deep, fully convolutional network that proposes regions in an image that might contain objects of interest. The other module is called the detector. The detector takes the proposed regions and then tries to detect objects within the region.

2.1.3 YOLOv4

YOLOv4 is a one-stage object detection module developed by Bochkovskiy et al. (2020) which has continued the work of YOLOv3 (Redmon and Farhadi, 2018). YOLOv4 aims to optimise its speed and accuracy compared to YOLOv3. YOLOv4 improvement in accuracy and processing speed is respectable compared to YOLOv3 processing speed and accuracy. YOLOv4 strives to be able to be run and trained on consumer-friendly graphic cards.

YOLOv4 is built on YOLOv3 (Redmon and Farhadi, 2018). The improvement made in YOLOv4 is the introduction of several new modules.

2.2 Tracking

All tracking uses detection, but there is an important distinction between some approaches. Some handle the detection internally, while some need a detection component which provides them with detections as input for the tracking phase. Tracking using detection as input works by taking each frame and applying a detection algorithm which extracts bounding boxes. The tracking part can utilise these boxes to track objects.

Tracking is often divided into two phases. Prediction of object locations and the association with detection and the prediction (Bewley et al., 2016). For each new frame, the tracking algorithm does the following:

1. Detect objects which are of interest.
2. Predict new location for detected objects using previous frames.
3. Associate the detection with the prediction.

Some popular tracking methods are motion-aware multi-object tracking (Han et al., 2020), simple online real-time tracking (Bewley et al., 2016) and Brasó and Leal-Taixé (2020)'s algorithm. All of the mentioned tracking algorithms take advantage of several algorithms to combine, to create a stronger tracking algorithm.

The tracking component we used is Simple Online Realtime Tracking (SORT). Bewley et al. (2016) implemented a tracking system called simple online real-time tracking (SORT). As input, it takes the detection from a detection algorithm such as YOLO. Using the detection, SORT associates the detection to a bounding box containing the object which is tracked. With a Kalman filter, the optimal velocity is calculated and connected to the bounding box and used to predict the future frames bounding boxes.

Kalman Filter for tracking moving targets consist of estimating a state vector that comprises of attributes from the target, such as velocity and position. It is this state vector that tries to calculate the future position of the targets. Kalman Filter is an algorithm that uses series of known information observed over time to predict unknown information. In the case of tracking the unknown information is the future position of the targets.

The Hungarian algorithm is a combinatoral optimization algorithm that can solve assignment problems. Furthermore, it is a fast algorithm. Using Hungarian algorithm, an assignment cost matrix is calculated by applying intersection-over-union (IoU) to find the distance between the detection and the predictions. IoU is the result of dividing the overlapping area och two bounding boxes by the area of union. In addition, a minimum IoU is used to discard assignments where the overlap between the detection and the track is lower than the IoU_{min}. This ensures that a correct data association is conducted between the detection and the track.

2.3 Tracking across cameras

According to Hou et al. (2017), there exist mainly three ways to track human objects across multiple cameras:

- Human re-identification

- Camera linked model(CLM) based tracking
- Graph model (GM) based tracking

Below there will be a summary of each of the three ways to track objects across cameras.

2.3.1 Human re-identification

Hou et al. (2017) report that human re-identification focuses on extracting features from an object and distance metric learning. Simplified human re-identification works by saving the extracted features from an object in a database. Objects are provided by a tracking algorithm, which detects and generates bounding boxes for each object. When an unknown object enters a camera view, it extracts the features from that object. Then it compares it to all feature sets in the database to see if the object is a known object or a new object. The comparison could be made using Euclidean distance by calculating how close the features are to be the same, and this is the distance learning metric part. If the object is a match, it assigns the id it has in the database, otherwise generates a new id for the object and saves it.

Features

Hou et al. (2017) report that the most common features to extract are colour, texture, shape, global features, regional feature, patch-based features, and semantic features. The most prominent one is the colour. Hou et al. (2017) further explain the definition of these features and how to extract these features.

Distance Metric Learning

Euclidean distance is the most common way to measure the similarity distance between two feature vectors representing human objects. Recently, research points to that the Euclidean distance approach struggles with significant variation in illumination, pose, and viewpoints.

According to Hou et al. (2017), further research has been more directed towards learning an optimal metric model. The optimal metric model's purpose is to learn a linear transformation that can map the original feature space to a new feature space.

2.3.2 CLM based tracking

According to Hou et al. (2017), CLM uses available training data from the entry/exit points of two views to estimate the feature (Temporal-spatial and appearance) relationship between two views. CLM uses this to account for the difference between the views when calculating the feature distance between human objects.

CLM can be trained using both supervised and unsupervised training. The difference lies in the training dataset, which needs considerably more attention if supervised training is used. Therefore since the dataset demands increased sizes to achieve functional performance, unsupervised training has become more commonly used. It is a more realistic alternative and scalable to large-scale camera networks. CLM is mainly used in tracking humans across multiple static cameras.

2.3.3 GM based tracking

Hou et al. (2017) informs that GM based tracking uses graph modelling techniques to generate a solvable GM using detection, tracks or trajectories as input. An optimisation solution is used to deal with data association across cameras. According to Hou et al. (2017), GM based tracking can handle complex scenes relatively well, but it struggles with the association of object across cameras.

2.4 Re-identification

We used TriNet and the Attention Generalised Mean Pooling with Weighted Triplet Loss (AGW) re-identification algorithm in this thesis. A summary of the re-identification methods we used can be found in the following sub-sections.

The feature extraction from one object in different environments might extract features which varies a lot under different circumstances. This problem might be solved by Principle Component Analysis (PCA) and is explained in the third sub-section.

2.4.1 TriNet

Re-identification is an essential part in being able to track across several cameras. Using Hermans et al. (2017) plain convolutional neural network (CNN) together with a triplet loss enables a re-identification application that has the ability to extract features from a human object.

This method works by enhancing the CNN's training by using triple loss. As an effect, end-to-end learning between an input image and a desired feature space is possible. Furthermore this optimises the network and causes an additional metric learning step obsolete. Then, simple Euclidean distance can be used to compare the features.

2.4.2 AGW

Ye et al. (2020) describe a re-identifcation method called AGW. AGW is an improvement on Luo et al. (2020) work. It is built upon a ResNet50 (He et al., 2016) as a backbone, which is a 50 layer Residual Network. In addition it utilises three improvement methods. First, a non-local attention block (Wang et al., 2018) produces the weighted sum of the features at all positions. Next a generalised-mean pooling (Radenovic et al., 2019) provides fine grained instance retrieval and lastly the weighted regularisation triplet loss part generates hard triplets for AGW to use.

2.4.3 Principle Component Analysis

Wold et al. (1987) describes how PCA can reduce the dimensions of data to contain as stable data as possible. PCA analyses the features over time to create new features in smaller dimension to represent the objects. These new features are hopefully more stable over time and can be trained on use-case specific data.

PCA creates new uncorrelated variables (principal components), where it tries to put maximum information in the first component, the remaining maximum information in the next component an so on. This is reduced into solving an eigenvalue/eigenvector problem, and the new variables are defined by the dataset at hand. To make the PCA more adapted to the specific use-case, you can feed it with varied data to find a feature space that are stable through these variations.

2.5 State-of-the-art Multi-Camera tracking

The following algorithms are the state-of-the-art real-time deep learning algorithms used for tracking and re-identifying across multiple cameras. The methods provide information on how state-of-the-art tracking methods are constructed and information we can use when constructing our tracking system.

2.5.1 A distributed approach for real-time multi-camera multiple object tracking

Previtali et al. (2017) detail a MTMCT system named PTracking. PTracking works by utilising Distributed particle filters (DPF), which is often used in developing tracking algorithms. Particle filtering is done by placing out possible future positions (particles) for the object. It weights each particle depending on the possibility of being correct, according to the known information. The particles are then resampled and new particles are created with the weights in consideration and so on. This enables the system to predict the location of the object, and can then use that to track objects in the views.

The three main components of PTracking are the following:

- The general algorithm (PTracking)
- The clustering algorithm (KClusterize).
- The data association.

KClusterize is a clustering method without the need of knowing the number of clusters. KClusterize is also an efficient algorithm that is not computationally heavy.

The data association assigns ID's to each object using the following features:

- Direction
- Velocity
- Position

Furthermore, PTracking uses an HSV (Hue, saturation and value) colour model to re-identify already known objects. The model contains HSV colour histograms for the already known objects to use when comparing them.

2.5.2 Real-Time Multi-Target Multi-Camera Tracking with Spatial-Temporal Information

Zhang and Izquierdo (2019) detail a tracking system composed of detection, tracking in a single view and a re-identification algorithm to handle cross camera object association.

For detection, they use the OpenPose algorithm (Cao et al., 2019). The false-positive rate for OpenPose is too high. A lightweight RFCN (Chen et al., 2018) is applied to refine the bounding box. The result is a drastic reduction of false positives.

It uses a Kalman filter for tracking, which is then sent to a re-identification network to connect the targets in multiple cameras. This network uses Deeply-Learned Part-Aligned Representations re-identification algorithm (Zhao et al., 2017) to extract features for the cross camera object association, which decomposes the human body into regions which are discriminative for person matching.

2.5.3 Real-Time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification

Chen et al. (2018) describe a method that uses R-FCN detection algorithm (Dai et al., 2016) together with a Kalman filter to predict future positions of the detected objects. To solve the association across cameras, a re-identification method is used. More specifically Szegedy et al. (2015) GoogLeNet is used.

Chapter 3
Datasets and Metrics

3.1 Datasets

This section contains a short summary of the datasets used for evaluating the different components. All datasets except for FLET21 provides ground truths. These ground truths are manually annotated by the authors of the datasets. These ground truths are used to compare the results against.

3.1.1 Detection datasets

For detection evaluation, we have used two datasets. The first is the COCO dataset (Lin et al., 2014). We have modified it a bit as we only evaluated persons and the COCO dataset has a wide array of classes. COCO is a commonly used dataset for evaluating detection.

3.1.2 Tracking datasets

For tracking, we utilised the MOTChallange 2015 dataset (Leal-Taixé et al., 2015), which is a collection of datasets with various characteristics. This dataset collection was chosen because it subjects the tracking algorithm to various difficulties. Examples of the difficulties could be that the objects are occluded, walks in groups or crosses path with other objects.

3.1.3 Re-Identification datasets

Re-identification uses Market1501 (Zheng et al., 2015) dataset. Market1501 is a commonly used dataset for training and evaluating re-identification methods.

DukeMTMC-reid is a subset of the complete DukeMTMC (Ristani et al., 2016) dataset. This subset is specifically designed for training and evaluating a re-identification.

3.1.4 Dataset for the complete system

For evaluating the complete system, we created a couple of videos representing the use cases. We called it FLET21. The dataset consists of several camera views which contain a specified amount of people.

3.2 Metrics

This section contains a short summary of the metrics used for evaluating the components and the complete system.

3.2.1 Detection metrics

Padilla et al. (2021) describe some common metrics used when evaluating detection. There are the following: Average precision (AP), average recall (AR), true positives, false positives and missed detection. Some of them are self-explanatory.

AP is a conventional object detection evaluation metric where AP is the regression and classification accuracy. AP is often determined using intersection over union (IoU) which describes how good the bounding box fits within the ground truth bounding box. For example, AP_{50} says that at least 50% of the bounding box fit within the ground truth bounding box. AR is used to measure the detection algorithms ability to detect an object in an image. AR1 describes how good it is at detecting an object if there is one object in the image and AR10 is how good it is at detecting an object if there are 10 objects in the image.

True positives (TP) describes the number of correctly identified objects. False positives (FP) describe the number of incorrect detection of objects that do not exist or misplaced an existing object's detection. False negatives (FN) describes the number of objects which were not detected. True negatives (TN) is not possible to measure, since it represents the objects that are not detected and does not exist.

3.2.2 Tracking metrics

There are many metrics for evaluating tracking, and not all of them are essential for this thesis. A selection of the most relevant metrics is therefore necessary.

The metrics used for evaluating the tracking are from Ristani et al. (2016)'s paper where the following metrics are used: Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), ID F1 score (IDF1), ID precision (IDP), ID recall (IDR), Recall, Precision, TP, FP, FN, ID switches (IDSW) and ID switch recall (IDSWR):

MOTA describes the overall performance of the tracking.

MOTP describes the average dissimilarity between the trackers output and the ground truth.

IDF1 is the number of correctly identified detections over the average number of labels in the ground-truth and the computed detection.

IDP is the ID precision measuring the fraction of ground-truth person detections, that are correctly assigned to a unique person ID.

IDR measures the probability that the id is correctly labelled.

IDSW is the number of times the tracked objects switch id in total. Lastly,

IDSWR is the ratio for how often each object switches id.

3.2.3 Re-Identification metrics

For re-identification, there are two commonly used metrics for evaluating re-identification. Mean average precision (mAP) and cumulative match characteristics (CMC-k) where k is the rank. The rank specifies how many of the top most possible matches we look at. mAP is the average over all the precisions of the queries. Precision is the ability of the model to only identify the relevant objects. CMC-k represents the probability that a correct match appears in the top k ranked retrieved results.

We also have the check for existence (CFE) accuracy metric, which describes how good the re-identification algorithm is at determining if a person exists in the database or not. The evaluation is done using a query and gallery set. The gallery contains images of people. The query set contains images on both people that exist in the gallery set and some that is not yet seen.

This evaluation is done by creating a gallery containing a set number of people. The query set contains images of people in the gallery set.

Creating a query with persons we know exist in the gallery and persons we know do not exist. This result in a percentage value.

3.2.4 Metrics for the complete system

Here we will look at the processing speed (FPS), tracking and re-identification performance. There are no direct metrics for evaluating a complete tracking system over multiple cameras, so here we had to construct a metric for evaluating the complete system's performance.

Chapter 4
Method

The work has been divided into five separate but closely related stages. For a complete picture of the workflow, Figure 4.1 shows the different stages and how they relate.

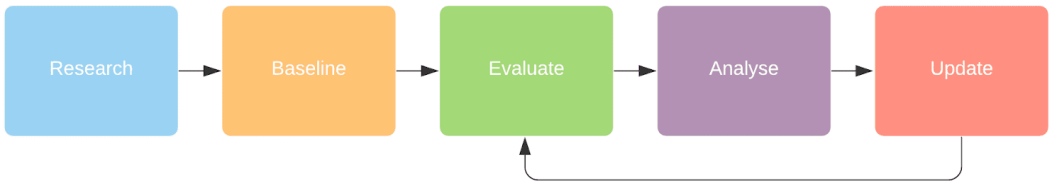

Figure 4.1: Thesis workflow

In the first stage, we performed a literature review to examine real-time MTMCT. We researched the components we decided to use in our MTMCT system. The system was chosen from extensive research into similar use cases.

Both state-of-the-art and more frequently used approaches of each component have been researched. They both contain important information to be able to construct an MTMCT system properly.

The inclusion criteria are that they must have been published within 10 years, written in English or Swedish, and available through known digital libraries such as IEEE and arXiv. Common keywords used for exploring the libraries are “re-identification”, “Real-Time”, “Multi-target multi-camera tracking” or “MTMCT”, “Tracking algorithms”, “detection algorithm" and many more keywords. These mentioned keywords has been combined together in various combinations.

We have researched different methods to evaluate tracking systems and their components. This enables a correct way of evaluating the tracking system and its component to increase the results’ validity.

Standard datasets used for evaluating the components and the MTMCT system were also researched. The reason is that we need datasets that represent the use cases we were given. There is also the need for commonly used datasets that can grant legitimacy to our evaluations. This stage will lay the foundation to be able to answer the research questions dictated in Section 1.3.1.

The second stage consists of utilising the knowledge gained from the research stage. Here we take the knowledge gathered and implements a MTMCT baseline using open-source projects. The result is a baseline that can track objects in a multi-camera system. This stage provides answers for **RQ1** since we have to create a working baseline for our use cases.

The third stage is to evaluate the system created from the second stage. This evaluation will be the basis for comparing future improvements. The result from the third stage is a list of performance values for each of the components and also for the complete tracking system.

The fourth stage's task is to carry out a comprehensive analysis of the evaluation stage's information. This information will show how the components work together, which components are bottlenecks. The information gained from the analysis will then result in one or more component and parameters which needs to be modified. Furthermore, we will execute a more thorough analysis of why the component works the way it does with the other components. For example, why does inadequate detection accuracy affect the tracking component in a certain way? At this stage, we are laying the groundwork for **RQ2**. Moreover we will investigate the metric which is described in Section 4.3.1.

The fifth stage, the update stage, takes the information from the analysis to determine which part we should update. We continuously loop over stage three to five until either the time limit is reached or sufficient performance is achieved.

4.1 Implementing a Baseline

Figure 4.2 provides an overview of the baseline and how the future system will work.

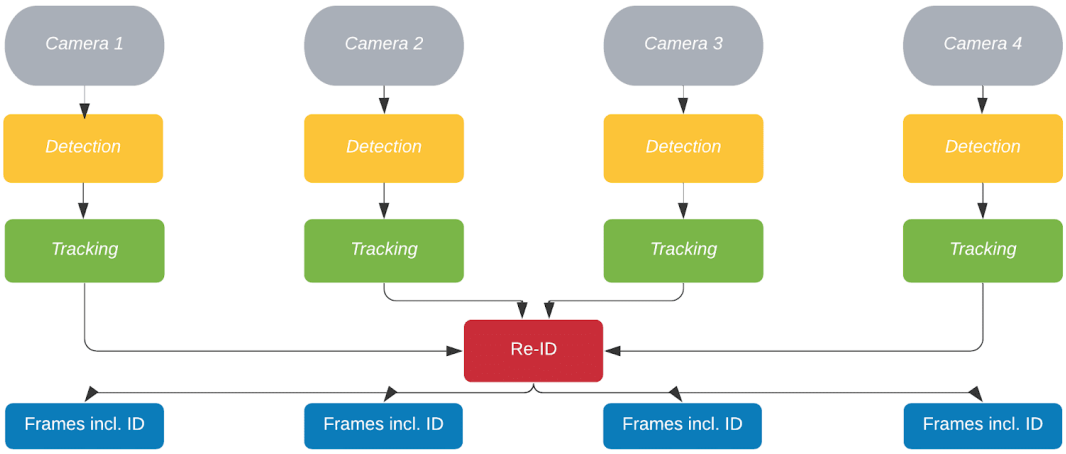

Figure 4.2: Complete MTMCT system

After researching, we have concluded that we should divide the components as in Figure 4.2. The reason for this division, instead of having some parts merged together, is that we

now have greater control over the components. It is easier to replace components, change the parameters and optimise the way all components work together. With this division, we can find implemented models for each component and create a baseline to start from. In addition we have the ability to individually evaluate the components which would not be possible in a system where the components are built in together.

4.1.1 Detection

For detection, we use YOLOv3, which is discussed in Section 2.1.1. The reason to use YOLOv3 is that its focus lies in achieving real-time detection and still having decent accuracy. Since we decided that real-time is a crucial aspect, we chose to prioritise real-time over accuracy.

YOLOv3 is commonly used due to its ease of use and implementing. There exist pre-trained models for the implementation, which drastically reduced the time to implement it. There are similar works from which we could gather inspiration when implementing it. These mentioned aspects are the reason for choosing YOLOv3 and not later versions. The later ones are more experimental versions, which do not provide good and tested support or models.

YOLOv3 takes a frame as input and produces bounding boxes for each object found in the frame. These bounding boxes are then sent to the tracking algorithm. In the baseline, we apply YOLOv3 on every frame since it is relatively fast. However, it is possible to only run the detection every n-th frame.

4.1.2 Tracking

Simple Online Real-Time Tracker (SORT) was used as the tracking component. Section 2.2 summarises how SORT works. The reason for choosing SORT is grounded in that it is commonly used when implementing a baseline. SORT was explicitly designed to be a baseline and a testbed for future trackers. Furthermore, SORT is easy to implement. The tracker's downside is that it has difficulties handling occlusion and cannot keep a consistent id when an object disappears and reappears. This downside is irrelevant in our case due to the re-identification algorithm handling this.

SORT takes the bounding boxes the detection algorithm produces and generates an id for the bounding boxes. SORT then tries to predict future positions for the bounding boxes to keep a consistent id for the targets.

4.1.3 Re-Identification

The TriNet re-identification algorithm, which is discussed in Section 2.4.1 was used as the re-identification component. TriNet was chosen based on its ease of implementation and that it had available trained models. The performance of TriNet, while not the best, is sufficient for the baseline.

TriNet receives the targets from each of the camera views and extracts features from them. It uses the Euclidean distance to measure the distance between the object features and all the database features by utilising a threshold to determine if the object's features are close enough to features in the database. This enables us to say whether they are the same object or not. If

the object has no match in the feature database, it stores it with a new ID. Re-identification is constructed to allow for only applying it every n-th frame.

4.2 Evaluation

We have created an evaluation pipeline to enable us to quickly replace the components in the tracking system. This frees up more time to change out components. The evaluation pipeline is constructed to evaluate the components independently and also to calculate the metric for the complete system. The metric for the complete system is Weighted Averaged Metrics (WAM) which tells us roughly how good the tracking system is.

4.2.1 Detection

Section 3.2.1 describes the metrics used in the evaluation of the detection algorithm. Furthermore, Section 3.1.1 describes the dataset used in the evaluation of the detection models.

The evaluation for detection consists of applying the detection algorithm on a considerable amount of images and then comparing the output against the ground truth. The ground truth were manually annotated by the authors of the dataset. This produces accurate performance values for the detection algorithm, which describes its ability to detect objects.

The evaluations were conducted on the COCO dataset, one of the most commonly used datasets when evaluating detection models.

4.2.2 Tracking

Section 3.2.2 describes the metrics used in the evaluation of the tracking algorithm. Furthermore, Section 3.1.2 describes the datasets used when evaluating the tracking models.

The evaluation of the tracking algorithm consists of several evaluations on different datasets. The reason is that we want to evaluate the tracking algorithm on datasets containing different difficulties such as occlusion, number of people and image difficulties. For each of the datasets, a list of detections is provided. The tracking algorithm receives these detections, predicts their future positions and assigns them an id. The output is then compared against the ground truth for the dataset. The ground truth were manually annotated by the authors of the dataset. This result in accurate performance values. Since we test the tracking on different datasets with different complexities, we receive performance values which reflect how the tracking handles complex situations.

4.2.3 Re-identification

Section 3.2.3 describes the metrics used in the evaluation of the re-identification algorithm. Furthermore, Section 3.1.3 describes the datasets used for evaluating the re-identification models.

The re-identification uses two datasets which are common datasets to evaluate on. The re-identification algorithm is applied on a gallery and a query set which are images of persons extracted from the dataset. We have modified the sizes of the sets to reflect our use cases

more accurately. The evaluation consists of matching the persons from the query and gallery sets. The result is then compared against the provided ground truths. The result consists of accurate performance values for the re-identification algorithm. Testing it on more than one dataset ensures that it has no bias towards a dataset.

When evaluating the re-identification model, we will evaluate both with or without dimension reduction methods. There are two different approaches of dimension reduction we experimented with and found out whether it improves or worsens the metric values. The first approach is to simply train the re-identification algorithms to use less features, the other involves the usage of PCA to reduce the feature dimension.

4.2.4 Complete System

Section 3.2.4 describes the metrics used in the evaluation of the complete system algorithm. Furthermore, Section 3.1.4 provides a description of the datasets for the complete system evaluation.

The development testing consists of running the complete tracking system on two camera views and monitoring the frame rate. The system is constructed to calculate the frame rate in runtime and presents the average frame rate when finished executing. Moreover, we conduct a manual evaluation of the result on the finished video sequence. Here we evaluate its ability to keep the ids consistent over several cameras.

Due to Covid-19 and GDPR, the difficulty to gather suitable material to test the complete system properly was increased. To gather performance values that reflect the use cases we were provided, we would have to create four camera views where approximately 30 persons moved around. We had to create this ourselves. Due to GDPR, we wont be able to show any images.

We created a two-view camera network with roughly 10 persons. This might hurt the tracking system's validity since we cannot accurately test it on its intended number of people. We reasoned that this only hurts the performance values and not the ability to determine the effect of the different components on the complete tracking system.

The argument for this is that the only difference we will see is the metric change in magnitude. The metrics values still represent the performance of the components and can still be used. We have named our dataset FLET21. In FLET21, there are also other smaller videos which test different scenarios. These videos are created with 1-2 camera views and about 5 persons.

As mentioned in Section 3.2.4, there is no specific metric for a MTMCT system. Therefore we had to use our WAM metric to evaluate the MTMCT.

4.3 Analysis

The fourth stage encompasses analysing the evaluation result. This means that we go over each of the MTMCT system components and conduct quantitative and qualitative analyses. We looked at the complete tracking system and conducted a quantitative and a qualitative evaluation. Lastly, using the information from the analysing stage, we initiated the update stage.

The quantitative analysis consists of calculating the metrics for each component and then calculating the WAM for the complete system. For a more thorough explanation on how WAM is constructed and how to compare WAM values, see Section 4.3.1.

The qualitative evaluation is conducted by examining the output videos after the tracking system has been applied. Here we will look at how it applies ids and tracks the objects across cameras. We also look at how the environment (light, occlusion and other disturbances) affect the MTMCT system.

4.3.1 Weighted Averaged Metrics (WAM)

The quantitative evaluation consists of calculating $\text{WAM}_{component}$. Equation 4.1 describes how to calculate WAM for detection, Eq. 4.2 is for tracking and last Eq. 4.3 describes re-identification. The values which we receive from these represent the performance of the components and the complete system. The higher the value, the better the performance.

$$\text{WAM}_{Detection} = AP_{50} \tag{4.1}$$

$$\text{WAM}_{Tracking} = \frac{MOTA + MOTP}{2} \tag{4.2}$$

$$\text{WAM}_{Re-identification} = \frac{mAP + CMC_{rank\_1}}{2} \cdot CFE \tag{4.3}$$

For the complete system, we summarise the $\text{WAM}_{component}$ from each of the components.

$$\text{WAM} = \text{WAM}_{Detection} \cdot \text{WAM}_{Tracking} \cdot \text{WAM}_{Re-identification}. \tag{4.4}$$

What is important to mention here is that WAM is heavily biased towards our use cases. This causes the metric to be hard to compare to other tracking systems with other use cases. The WAM value only holds meaning within our use cases and exists to enable an easy way of determining if the system performance changes.

4.4 Update

4.4.1 1st Iteration: Re-identification

To accomplish this, we used PCA which is explained in Section 2.4.3.This was to let the program do a deeper analysis of the features stability and create new, fewer features. The PCA was performed with features extracted from a video we recorded of a person walking in different light, background, poses and distances. The hope was to create fewer and simpler features, adapted to our use case. We tried this for 8, 16, 32 and 64 features.

4.4.2 2nd Iteration: Detection

Based on the results from the first iteration, we chose to update the detection. We decided on using FRCNN (Ren et al., 2017) which focuses more on accuracy and correct bounding boxes, which we felt was needed to improve the overall performance.

4.4.3 3rd Iteration: Detection

Based on the result from the second iteration, we decided on changing the detection one more time. The reasons are the disappointing result from FRCNN (Ren et al., 2017). We chose to implement YOLOv4 (Bochkovskiy et al., 2020) which shows respectable increase in accuracy and processing speed compared to both FRCNN and YOLOv3 (Redmon and Farhadi, 2018).

4.4.4 4th Iteration: Re-identification

We wanted to see how the re-identification is affected when lowering the feature dimensions when training the TriNet re-identification model.

4.4.5 5th Iteration: Re-identification

Based on the results in Section 5.3.1, we decided on testing a re-identification algorithm with higher performance metrics in the mAP and rank-1 fields. Therefore we chose the Attention Generalized mean pooling with weighted triplet loss (AGW) which has a respectable increase in both the mAP and the rank-1 metrics. It is also a state-of-the-art baseline which was developed and used in Ye et al. (2020) to compare against current state-of-the-art re-identification methods.

Chapter 5
Results

In this chapter, we begin with presenting the results from evaluating the detection, tracking and re-identification models independently. We continue by presenting the results for evaluating the complete MTMCT system both quantitatively and qualitatively. Lastly, we will talk about how the component cooperate and affect each other.

5.1 Detection

The detection models are evaluated on the COCO dataset described in Section 3.1.1. Since we are only interested in detecting pedestrians, we modified the evaluation to only detect persons within the COCO dataset. We present tables of metric values for each detection algorithm we implemented in our system.

The metrics we produced are AP with different intersection over union (IoU) and AR with different numbers of ground truth objects. These metrics are described in Section 3.2.1. Since the tracker can handle some flaws from the detection stage, AP_{50} is the most interesting metric value and is the only one that has impact on the WAM metric.

Table 5.1 shows the metrics provided by the evaluation tests from each of the detection algorithms. Table 5.2 shows the performance in the aspect of processing speed. The reason for calculating the processing speed on FLET21, is that we wanted to evaluate it on a representation of our use-case. In the COCO dataset, there are always only at most one object to be detected per image. This is why we use FLET21 for evaluating the processing speed, since we want to know how fast one average frame is calculated. Also, the resolution on the videos are higher, which is more likely the case when used in practice. Each section also includes three bounding boxes for one person created with the given detection model. These bounding boxes are picked manually from a video but should represent what it usually looks like. The images are from the same video where one is taken from behind, one from the side further away and one from the front.

Table 5.1: Detection models evaluated on the COCO dataset. The metrics are explained in Section 3.2.1 and the dataset can be read about in Section 3.1.1

Model	Dataset	AP	AP_{50}	AP_{75}	AR1	AR10	AR100	WAM
YOLOv3	COCO	0.3575	0.5808	0.3905	0.1638	0.3954	0.3953	0.5808
Faster RCNN	COCO	0.1796	0.4036	0.1294	0.1072	0.2192	0.2230	0.4036
YOLOv4	COCO	**0.4519**	**0.6487**	**0.5171**	**0.1823**	**0.4783**	**0.4884**	**0.6487**

Table 5.2: The time it took for each of the detection algorithms to process one frame in FLET21

Detection algorithm	Processing speed (s)
FRCNN	0.3350
YOLOv3	0.0740
YOLOv4	0.0592

5.1.1 YOLOv3

The metric values for YOLOv3 on the COCO dataset are presented in Table 5.1. It receives the second highest values of the three detection models evaluated in all of the metrics. In Table 5.2, we can see that YOLOv3 also comes in the second place in the aspect of processing speed, which is calculated with the reason explained above. Figure 5.1 visually shows three examples of bounding boxes on a person detected and created by YOLOv3. As we can see, the bounding boxes are relatively accurate. There is some extra space at some places and some small parts of the person are missing in the middle image.

Figure 5.1: Three different images with bounding boxes detected by YOLOv3 applied to it.

5.1.2 Faster RCNN

Table 5.1 presents the metric values for Faster RCNN on the COCO dataset. Table 5.2 shows the processing speed of the model, evaluated on the FLET21 dataset with the reason explained

earlier. As we can see, Faster RCNN underperforms in both the precision/recall and processing speed aspects. Figure 5.2 visually shows three examples of bounding boxes on a person detected and created by Faster RCNN.

In these examples, the FRCNN misses a lot in some bounding boxes and very often has much extra space around the actual objects. The bounding boxes change very much between every frame, which is harder to present in images. The middle image in Figure 5.2 shows one example when FRCNN both miss a huge part of a person and got considerably extra space on one side. This frame is not the same as when showing examples from YOLOv3 and YOLOv4 in Figures 5.1 and 5.3 respectively. This is because FRCNN was not capable to detect the person in that moment at all.

Figure 5.2: Three different images with bounding boxes detected by FRCNN applied to it.

5.1.3 YOLOv4

YOLOv4's results from the evaluation on the COCO dataset is shown in Table 5.1.The processing speed per frame, which is evaluated on the FLET21 dataset with the reason explained before, is shown in Table 5.2. These results show that YOLOv4 gives the superior performances on all the independent evaluations. Figure 5.3 visually shows three examples of bounding boxes on a person detected and created by YOLOv4. As we can see, the bounding boxes fits the object almost perfectly. It does not include space between the person and the bounding box, neither does it miss much of the person. The bounding boxes between each frame in the tested videos are very similar for the objects. This makes the boxes float very smoothly in the video.

5.2 Tracking

In Tables 5.3 and 5.4, we present the results generated by evaluating SORT according to Section 4.2.2. The number of calculated metrics was too large, which is the reason it is divided into two tables. The datasets in the tables are all of the datasets in the MOTChallenge, which is a collection of varying datasets representing different situations för multi-target tracking.

Both tables show varying values on the different datasets. For example, the MOTA for TUD-Stadtmitte dataset is 0.7171 while MOTA for the Venice-2 dataset is 0.1855. When

Figure 5.3: Three different images with bounding boxes detected by YOLOv4 applied to it.

looking into these two datasets, we can see that Venice-2 includes much more occlusion and is more crowded. The overall WAM value for the datasets was 0.5635, which is calculated as described in Section 4.3.1.

When running our system without the re-identification on the FLET21 dataset, we can see that SORT struggles with occlusion and reappearing objects. Another problem for SORT is that it can not track two persons overlapping each other. This explains the bad MOTA value on the Venice-2 dataset. The process time for SORT on our FLET21 dataset was an average of 0.0014 seconds per frame.

Table 5.3: SORT tracking algorithm evaluated on the dataset mentioned in the dataset column. See Section 3.2.2 for information on the metrics and see Section 3.1.2 for information regarding the datasets used. This is part 1 of 2 tables. See Table 5.4 for part 2.

Dataset	MOTA	MOTP	Recall	Precision
ADL-Rundle-6	0.3809	0.7489	0.5752	0.7624
ADL-Rundle-8	0.2860	0.7106	0.4426	0.7579
ETH-Bahnof	0.4471	0.7433	0.6486	0.7743
ETH-Pedcross2	0.4538	0.7480	0.5188	0.9078
ETH-Sunnyday	0.5915	0.7438	0.7750	0.8186
KITTI-13	0.1929	0.6855	0.3858	0.6918
KITTI-17	0.6018	0.7227	0.6706	0.9234
PETS09-S2L1	0.6189	0.7368	0.7500	0.8738
TUD-Campus	0.6267	0.7368	0.6852	0.9425
TUD-Stadtmitte	0.7171	0.7523	0.7448	0.9751
Venice-2	0.1855	0.7341	0.4246	0.6476
Overall	0.3989	0.7283	0.5596	0.7918

Table 5.4: SORT tracking algorithm evaluated on the dataset mentioned in the dataset column. See Section 3.2.2 for information on the metrics and see Section 3.1.2 for information regarding the datasets used. This is part 2 of 2 tables. See Table 5.3 for part 1.

Dataset	IDF1	IDP	IDR	IDSW	IDSWR
ADL-Rundle-6	0.4338	0.5044	0.3805	75	1.3
ADL-Rundle-8	0.3263	0.4426	0.2518	103	2.3
ETH-Bahnof	0.6094	0.6684	0.5599	67	1.0
ETH-Pedcross2	0.5353	0.7360	0.4206	77	1.5
ETH-Sunnyday	0.7244	0.7447	0.7051	22	0.3
KITTI-13	0.4381	0.6118	0.3412	16	0.4
KITTI-17	0.7108	0.8448	0.6135	9	0.1
PETS09-S2L1	0.3510	0.3800	0.3262	102	1.4
TUD-Campus	0.6065	0.7203	0.5237	6	0.1
TUD-Stadtmitte	0.7347	0.8482	0.6479	10	0.1
Venice-2	0.3243	0.4094	0.2684	57	1.3
Overall	0.4589	0.5541	0.3916	544	9.7

5.3 Re-Identification

The results we present in this section are retrieved from experiments on both the Market1501 and the DukeMTMC-reid datasets respectively. In our experiments, we provide three different metric values: mAP, CMC-1 and CFE. To match our use case, we ran the experiments on 15 randomly generated subsets of 40 people where half are previously seen and exist within the re-identification database. We did this to represent a kind of steady state for the system. We ran this with 10,000 iterations to get a statistically trustworthy result, where we take the average values from each iteration.

We also present the results for different approaches of performing the re-identification step. We have some variation of how many features we look at when comparing an unknown object with a person in the database. These features are converted with a feature dimension reduction method. The comparison between a feature list with the database is done by computing the euclidean distance. To decide if the detected person is included in the database we use a threshold. Only new targets within a view are re-identified.

5.3.1 TriNet

In Tables 5.5 and 5.6, we can see the metrics produced when testing TriNet on the Market1501 and the DukeMTMC-reid datasets. These are the two datasets presented in 3.1.3. This re-identification model was trained on the Market1501 dataset and was trained to extract 128 features. We used PCA, explained in Section 2.4.3, to reduce the feature dimension to 64, 32, 16 and 8 features. The PCA was created with a video of a person representing our use case where the light, pose, angles and distance varies.

Table 5.5: Metrics for the TriNet re-identification model on Market-1501. 128 features is the original extracted features while 64, 32, 16 and 8 features are created by the PCA. The thresholds used to determine if a person exists in the gallery is 8 for 128 features, 7.5 for 64/32 features, 5.5 for 16 features and 3 for 8 features.

Features	mAP	CMC-1	CFE	WAM
128	**0.9850**	**0.9944**	**0.9937**	**0.9835**
64 (PCA)	0.9748	0.9909	0.9910	0.9740
32 (PCA)	0.9714	0.9901	0.9917	0.9726
16 (PCA)	0.9350	0.9765	0.9845	0.9409
8 (PCA)	0.7977	0.8953	0.9442	0.7992

Table 5.6: Metrics for the TriNet re-identification model on the DukeMTMC-reid dataset. 128 features is the original extracted features while 64, 32, 16 and 8 features are created by the PCA. The thresholds used to determine if a person exists in the gallery is 8 for 128 features, 7.5 for 64/32 features, 5.5 for 16 features and 3 for 8 features.

Features	mAP	CMC-1	CFE	WAM
128	0.4708	0.6108	**0.8603**	0.4652
64 (PCA)	**0.4753**	**0.6148**	0.8550	**0.4660**
32 (PCA)	0.4707	0.6134	0.8549	0.4634
16 (PCA)	0.3954	0.5432	0.8223	0.3859
8 (PCA)	0.2776	0.3785	0.7218	0.2367

As we can see, reducing the number of features did not improve the stability and performance in our case. Since we did not see any significant improvements, we did not go on with a qualitative analysis with PCA. Since the model was trained on Market1501, the metrics produced with the Market1501 dataset are not as interesting as the ones with Duke. This is because when we train a model on some data, it gets familiar to this kind of data. When we run the tests on similar data, it will more like perform better on these tests since it is trained to do so. The DukeMTMC-reid dataset then represent the general case since it is unknown to the model.

In Table 5.7, we present the metric values for the TriNet model that where trained to extract 64 and 8 features instead of 128. As we can see, no improvement were accomplished in this attempt to find simpler features.

Table 5.7: TriNet re-identification algorithm evaluated on Market1501 and DukeMTMC-reid datasets. The model is trained to extract 64 and 8 features instead of 128.

nbr. features	Dataset	mAP	CMC-1	CFE	WAM
64	Market	0.9750	0.9916	0.9811	0.9647
64	DukeMTMC-reid	0.4632	0.5937	0.7159	0.3783
8	Market	0.9496	0.9779	0.9523	0.9178
8	DukeMTMC-reid	0.4441	0.5524	0.6723	0.3349

5.3.2 AGW

In Tables 5.8 and 5.9, we find the resulting metrics created by the AGW re-identification model on the Market-1501 and the DukeMTMC-reid dataset respectively. Just as TriNet, AGW was trained on the Market-1501 dataset and extracts 2048 features per object. We used PCA to reduce the feature dimension to 1024, 512, 256 and 128 features. The PCA was created on the same video as the one for TriNet, where it can follow a person through different circumstances. We can see that adding feature dimension reduction to the AGW with PCA did not improve the metrics at all. We were not able to retrain AGW with less features due to that it AGW is quite heavy to train.

Table 5.8: Metrics for the AGW re-identification model on the Market-1501 dataset. 2048 features is the original extracted features while 1024, 512, 256 and 128 features are created by the PCA. The thresholds used to determine if a person exists in the gallery is 17 for 2048 features, 15 for 1024 features, 13 for 512 features, 10.5 for 256 features and 9 for 128 features.

Features	mAP	CMC-1	CFE	WAM
2048	**0.9854**	**0.9965**	**0.9941**	**0.9851**
1024 (PCA)	0.9741	0.9944	0.9933	0.9777
512 (PCA)	0.9656	0.9930	0.9903	0.9698
256 (PCA)	0.9511	0.9911	0.9869	0.9583
128 (PCA)	0.9253	0.9865	0.9108	0.8707

Table 5.9: Metrics for the AGW re-identification model on the DukeMTMC-reid dataset. 2048 features is the original extracted features while 1024, 512, 256 and 128 features are created by the PCA. The thresholds used to determine if a person exists in the gallery are 17 for 2048 features, 15 for 1024 features, 13 for 512 features, 10.5 for 256 features and 9 for 128 features.

Features	mAP	CMC-1	CFE	WAM
2048	**0.6466**	**0.7728**	**0.9026**	**0.6406**
1024 (PCA)	0.5476	0.7066	0.8894	0.5578
512 (PCA)	0.5293	0.6965	0.8891	0.5449
256 (PCA)	0.5035	0.6779	0.8782	0.5187
128 (PCA)	0.4661	0.6523	0.8699	0.4865

5.4 Complete system

In this section, we will present the evaluation done with the complete MTMCT system on the FLET21 dataset. We present both the qualitative and quantitative analyses described in Section 4.2.4. The distance between two objects in all tests were computed and represented in Euclidean distance since it was recommended by the authors of the re-identification models. The evaluation was done within every iteration of the MTMCT on three different sub-datasets from FLET21.

1. ROOM: Recorded inside a boxing gym with only four persons walking through a camera view in a line with some space between. The videos are with the same view but in different times and the persons are walking in another order.

2. COR: Dataset representing an indoor corridor with two cameras in different directions placed about 2 meters from the ground. Recorded outdoors because of the circumstances of the ongoing pandemic with Covid-19. This dataset was created to represent the indoor corridor case mentioned in Section 1.2

3. STREET: Created outdoors on a bigger area to represent the street view on the lamp-posts case mentioned in Section 1.2. Some persons are walking through the view very far away and some are closer to the cameras. Also this dataset has two views.

Table 5.10 shows us the resulting WAM and FPS from all the different setups on the FLET21 dataset. This is the quantitative analysis done on the complete system. We can see that we no MTMCT system managed to achieve the goal of 30 FPS.

In the coming sub-sections we present the results from the qualitative evaluation done on all the different sets of components.

5.4.1 Baseline (YOLOv3 – SORT – TriNet)

The baseline is an MTMCT system consisting of the following components:

- Detection: YOLOv3

Table 5.10: The WAM value for each of the MTMCT system and the FPS they manged on the FLET21

MTMCT system	WAM	FPS
YOLOv3-SORT-TriNet	0.1532	4.1548
FRCNN-SORT-TriNet	0.1068	1.9035
YOLOv4-SORT-TriNet	0.1716	4.3934
YOLOv4-SORT-AGW	0.2088	4.1050

- Tracking: SORT
- Re-identification: TriNet

When evaluating the system on the ROOM dataset, the system correctly re-identified three out of four persons. There was one person who switched id when turning in both views. On the COR and STREET datasets, it did not manage to re-identify any of the persons across the different camera views. The bounding boxes from the detection model were sufficiently good except for some extra space around some objects. The tracker worked well except for the cases when persons cross each other. The tracker only keeps one of the targets and sometimes lets the wrong person keep that track id. When a person walks out of a camera, and another person walks in at almost the same place and approximately the same time, the new person steals the track when entering the view. This is because SORT only uses spatial and temporal aspects when tracking.

5.4.2 FRCNN – SORT – TriNet

This MTMCT system consists of the following components:

- Detection: FRCNN
- Tracking: SORT
- Re-identification: TriNet

The version of MTMCT system with FRCNN as detection algorithm did not re-identify any of the targets between cameras. The bounding boxes where very inaccurate where it often included things in the background as part of the persons.

5.4.3 YOLOv4 – SORT – TriNet

This MTMCT system consist of the following components:

- Detection: YOLOv4
- Tracking: SORT
- Re-identification: TriNet

YOLOv4-SORT-TriNet got the same performance on our qualitative analysis as the same system with YOLOv3 instead. The only difference is that the bounding boxes are a bit more accurate. This does not help the re-identification in our test cases, but helps the tracking for each target to become smoother. As we can see in Table 5.10, this is the fastest system even though it is not real-time with our implementations.

5.4.4 YOLOv4 – SORT – AGW

This MTMCT system consist of the following components:

- Detection: YOLOv4
- Tracking: SORT
- Re-identification: AGW

This system, where AGW is used instead of TriNet as re-identification model, is the one with best results on the metrics. Still, this does not show in our qualitative analysis. It did not re-identify any target across the cameras.

5.5 Cooperation between components

The three different detection algorithms used (YOLOv3, FRCNN and YOLOv4) have very different performance.

FRCNN creates much more varying bounding boxes for the same object, which affects both the tracking and re-identification algorithms. The tracking algorithms are having a hard time interpolating the target between frames with detections, which makes the tracks irregular and harder to follow. Since both the re-identification algorithms (TriNet and AGW) are struggling with different backgrounds, too large bounding boxes from the detection algorithms decrease the re-identification performance.

The other detection algorithms, YOLOv3 and especially YOLOv4, create much more regular and precise bounding boxes which improves the re-identification performance. The tracking algorithm (SORT) does a good job tracking targets until they get occluded or disappears for a short while. When it loses track of a person when occluded or overlaps with another person, it forces the system to re-identify the lost target. Lastly, when two targets cross each other, one may take over the other persons track id. Since we do not want to assign the same id to different targets within one view, we might let the wrong person take over the id. This forces the person, who the id actually belongs to, to re-identify and switch to another id. This corrupts the gallery of known persons, i.e. the database.

Chapter 6
Discussion

We start by discussing the complete system's performance and how the system worked out in the end. We continue by discussing every components performance and how it affects the MTMCT system. We also talk about our WAM metric and the results of reducing the features dimensions in different ways. Lastly, we also include the ethical aspects and the validity of this thesis.

6.1 Complete system

The complete MTMCT system consists of three parts, detection, tracking and re-identification. We only used already implemented models from github and connected them into one system. The detection component sends all detections for each frame to the tracker, the tracker uses these detections to update each track and remove duplicates. From here, we have bounding boxes with tracker ids. Every time a new tracker id appears, the detections for this track are used by the re-identification model to connect the track id to a person.

We got a working system, but did not reach the performance we wished for. The biggest problem is the re-identification part since the models seem to include a big amount of the background into the features. Also the light and pose will make features which depend on colour vary for the same person. It is also very hard to distinguish if one person already exists in the database or not. These problems are further discussed in Section 6.5

6.2 Weighted Averaged Metrics

The WAM metric we introduced in this thesis is a metric created for our specific use case. We chose the most important metrics for an overall quantitative evaluation for our system. This does not represent the reality, but might still give us a hint of if it performs well or not at all.

We chose to look at the following metrics for the computation of WAM:

- Detection: AP
- Tracking: MOTA, MOTP
- Re-identification: mAP, CMC-1, CFE

This generates the best kind of accuracy through the system as we could think of in an quantitative evaluation. The WAM value for re-identification seems to be kind of misleading. As seen in Section 5.3, AGW gets a WAM value of 0.6406 ond the DukeMTMC-reid dataset while TriNet gets a WAM value of 0.4660 as highest. In our qualitative evaluation, AGW do not work as well as TriNet on our test videos. AGW gets higher mAP, CMC-1 and CFE than TriNet, which means that even the standard metrics are misleading in this case. The reason for this might be that TriNet works better for our use-case. In our use-case, the background is many times pretty similar. Even though we can see that the background is affecting the feature extraction in the qualitative tests, it might have less impact. In the quantitative evaluations on the DukeMTMC-reid dataset, the TriNet might struggle more with different backgrounds than AGW. This would also imply that TriNet is better at extracting features when the background does not differ.

6.3 Detection

The coming two sections will discuss the results of the evaluation on the detection models. We include how the performance of the detection algorithms affect the complete MTMCT system. The discussion includes the aspect of our use case and how we used and implemented the models.

6.3.1 Detection performance

We started with YOLOv3 as the detection algorithm in the baseline. This model was the first model in the system to replace since we wanted as accurate bounding boxes as possible for the re-identification model. The reason why is described in the next section. YOLOv3 had a decent FPS of 0.0592s and created good enough bounding boxes for the tracker.

When replacing YOLOv3 with FRCNN, the results were sadly disappointing since we expected better accuracy than YOLOv3. FRCNN received much worse metric values, both quantitatively and qualitatively. We continued by replacing FRCNN with YOLOv4, which performed sufficiently enough for our MTMCT system.

6.3.2 The detection performance effect on the MTMCT

The performance of the detection model affects both the tracker and the re-identification model. If the detection model misses a considerable amount of bounding boxes for a target, the information which is fed to the tracker becomes limited and might make it lose track of the target.

YOLOv3 and YOLOv4 were sufficient enough for the tracker. It did not miss a substantial amount of detections per target, with exception to when something was occluded. The re-identification model would have better possibilities for improvement if it gets partly occluded objects as input. Anyhow, these situations with much occlusion seem to be easier to solve with a better tracker model instead.

Since the re-identification algorithms seems to depend to a considerable degree on the background when comparing objects, YOLOv3 might be including too big amount of space around the targets. YOLOv4 on the other hand, should not impact the tracker or re-identification very much in any negative way with its sufficiently accurate bounding boxes.

FRCNN missed too many detections, created inaccurate bounding boxes and could not detect objects far away. This limited the input feed to both the tracker and re-identification models, which decreased their and the whole MTMCT systems performance.

Another interesting part is whether we want detections from far away in our case. The detections from far away might make it hard for the re-identification model to extract good features due to too low resolution. The low resolution often results in less details and merged colors, which seems to extract generalised features. We noticed in the analysis that many detections matches with these low resolution persons. Still we want to detect and be able to track persons far away from the camera as well, which results in high requirements of the cameras resolution. One could also keep tracking people when they appear in low resolution, but not save new features extracted from them.

6.4 Tracking

This section will describe and discuss the results of the tracking algorithm SORT and how it affects the complete MTMCT system. We talk about what we think the tracker does well and what performance we would have wanted to see. Just as for the detection section above, we will include the aspect of our use case when discussing.

6.4.1 Tracking performance

The tracker got decent values on the metrics for the datasets which did not include too populated areas or occlusion. The tracker is stable as long as people do not cross each other's paths or get occluded. When people cross each other's paths, one target often steals the other's id and the other gets a completely new id. These flaws are critical in our case, but it is within what we expected from such a simple and fast algorithm. We did not have the time to replace this component since we prioritised changing the detection and re-identification models instead. We also thought that putting some work into configuring the way the models work together was more important than replacing the tracker.

6.4.2 The tracking performance effect on the MTMCT

Since re-identifying seems to be the weak spot in this kind of systems, we would need to be able to rely on the tracker in each view. If the tracker changes id, this will force the re-identification algorithm to re-identify the target. If this goes wrong, we might gather inaccurate data to the new target in the database. This will decrease the performance on the

re-identification model when comparing with seen objects in the future, due to corrupted data in the database.

When an object steals another person's tracking id, this will not trigger the re-identification model to re-identify the target. This means that it will continue gathering extracted features to the database from the wrong person. Because of the fact that SORT is struggling with occluded objects, the re-identification cannot gather information on the targets when they are occluded. This makes it harder to re-identify the target when it becomes occluded in the future.

6.5 Re-identification

Re-identification across the cameras is the absolute hardest part of the MTMCT system. When an already seen person appears in a new camera, this will often be in a completely different angle, light, environment, pose and distance. To compare this person with the known persons within the database is hard as it is. In our case, we also need to consider the open-world version. This means that the model firstly has to consider if the detected person exists in the database or not. The performance of these aspects is included in the coming sections, where we will also discuss whether feature dimension reduction is useful or not.

6.5.1 Re-identification performance

According to the tables shown in Section 5.3, we see that AGW receives better test results on the metrics. These metrics do not show a fair picture of the reality, since we actually found out that TriNet performs better in the qualitative tests. We do not really understand why AGW performs worse, since it has better mAP, CMC-1 and CFE than TriNet on the tests. The reason might be that AGW does not match our use-case as well as TriNet does, or it gets too affected of the detection or tracking flaws in the system.

Both TriNet and AGW are struggling with the difference in background when extracting features. The distance between a person and the ones in the database seems to be very affected by the background of the bounding boxes. We tried to reduce the feature dimensions in different ways to solve this in hope for extracting simpler and more stable features over time. This is further discussed in Sections 6.5.2 and 6.5.3. We did not have the time to experiment with segmentation of the bounding boxes to simply delete the background, which we might think can make a big difference.

In our implementation of the system, we are using a threshold to determine if a person exists in the database or not. This seems to be a bad way of doing it and does not work as well as one might think. Unfortunately, we were not able to implement anything that works better.

We were not able to get the test data we wanted to for FLET21 due to the ongoing pandemic. The datasets do not really represent the real cases since we could not gather the amount of people needed or being inside for some tests. We were not able to use the cameras from Axis which it is intended to run on, since we were working from home and had to use our mobile phones.

6.5.2 PCA

When choosing to implement PCA into our system, the thought was to reduce the features into fewer and more stable features. This PCA was created with videos where the angles, light, distance and poses differ to make the features independent of these factors. As far as we tested on our models, this did not improve the performance on either TriNet or AGW.

The PCA would hopefully contain the most significant and important features for each person. We tried to record the videos in environments and from heights representing our use-case to redirect the features towards optimal ones for our system. Probably both TriNet and AGW extracted as good features as they could before we applied the PCA, or we would need more material to train the PCA on to make it more reliable.

6.5.3 TriNet trained to extract less features

We also tried to train the TriNet model to extract less features from the bounding boxes. The idea was to represent a person more roughly by simple features to distinguish between persons more easily. Just as with PCA, this did not improve the performance in any way.

6.5.4 Re-identification performance affect on the MTMCT system

Since tracking within one view is an already solved problem, the re-identification is the component which has biggest impact on MTMCT. Not only the model, but also the way it is used. If the re-identification gets one re-identification wrong, this will corrupt the dataset and will probably lead to either generalised targets in the dataset or several ids for the same target. Both these problems will set off a snowball effect.

With a good re-identification, one can even correct the trackers flaws. If the tracker switches id for example, the re-identification model might connect the new tracking id to the same re-identification id. The re-identification model will not solve this if a target steals another ones when crossing each other for example. At least not in the way we implemented the system, since we only re-identify the persons when the tracker finds a new track.

6.6 Ethical aspects

Processing videos, or more specifically, tracking persons can be used for both good and bad purposes. Our work was to investigate the components of a MTMCT system, how we can connect them in an efficient way and how we can manipulate the features to make it more stable. We are focusing on tracking persons to study or prevent crimes, find lost persons and any other security based purposes. Still, this kind of technology can be used to violate other peoples privacy in an immoral way as well. As much as we would like to, we could not be able to prevent this from happening.

When recording the data for the FLET21 dataset, we made sure every actor were well aware of us recording. They all also had to agree to being a part of the videos. All images within this thesis contains only two persons who cooperated when writing this thesis, Felix

and Eric. Moreover, all recorded videos for evaluation will be deleted as we are done using them.

6.7 Validity

The metrics gathered from the quantitative evaluations are done with publicly available datasets, modified to represent our use case. This should give a valid representation of the values.

However, due to Covid-19, we were not able to create the data for qualitative evaluation as we wanted to. The FLET21 dataset contains fewer people than we wanted to and some use-cases were not able to be recorded in good places. For example, the corridor case had to be recorded outside. This made a big difference in the light, more varying background and was overall harder for the system to work as intended.

Since we only used already implemented components, and in the way the authors described how to use them, some validity lies in how they implemented them. Furthermore the way we connected the components and how we used the re-identification seems to work as intended.

Chapter 7
Conclusions

During this thesis, we found that some components are definitively more difficult than others. For example detection and tracking within a view is an already solved problem, especially the detection models.

We did not prioritise to replace the tracker since we were struggling more with the re-identification model and how to use it in the best possible way. The tracker works well except for the cases when people cross each other's paths or get occluded. If we were allowed to assume the camera topology to be known, the tracker would have been worth to replace at an early stage.

The next coming paragraphs answers both **RQ1** and **RQ2**. The detection algorithm used in our final MTMCT system, which performs as good as we need, is YOLOv4. YOLOv4 is probably both accurate and fast enough to use for this purpose if optimised.

The tracker used, SORT, is probably fast enough but lacks too much accuracy when people are occluded. The tracker needs to handle occlusion for the re-identification to work, since it needs to gather data for persons when they are occluded as well. We thought about some other solutions to this problem, which might be possible to be solved with the re-identification model. This is discussed in the next Section 7.1, which is about future work.

Furthermore, if the tracker switches id and creates a new track, the system is forced to re-identify this target. This will hopefully re-identify to the same target, but since re-identification seems to be the most difficult part we want to rely on the tracker as much as possible. In an even worse case, one person might steal another person's track when crossing each other. This person will now take another person's id and extract false features to it. Since we want to rely on the tracker, we do only re-identify persons which are considered new persons by the tracker.

The re-identification models we used for our system were TriNet and AGW. AGW achieved superior metric values from the quantitative evaluation than TriNet, while TriNet seems to work better in practice during our qualitative evaluations. The only reasonable reason for this is that TriNet is better adapted for our use-case, alternatively that our recordings in FLET21 dataset are not as extensive and reliable as they need to be. The re-identification

algorithm is the heaviest model to run. There might be possible improvements on how often to extract features, how to store them and how to compare them. Both models seem to be struggling with the background of the bounding boxes. This is the reason for replacing the detection algorithm to a more accurate one, to avoid extra background.

Regarding **RQ3**, we came to the conclusion that reducing the feature dimension did not improve our models. We tried to train the TriNet model to extract fewer features, which might be simpler and easier to separate people from each other. This did not improve the performance in any way except for a small cut in processing time when comparing feature vectors. We also found that transforming features with PCA, neither improved the performance significantly in any case. We created our PCA with recordings of persons through different circumstances and our hope was to create features that is independent of different light, angles and distances.

7.1 Future Work

One of the most important things to solve is when the tracker lets a person steal another persons tracking id. A way to solve this might be to not rely on the tracker when two persons are too close to each other, or to skip the Kalman's filtering in these occasions. If we always re-identify the persons when they are very close, this might solve the problem. It can be interesting to experiment with this to find an effective approach. Alternatively replace the tracker to a more stable one which can use the direction the objects are moving. The tracker might be replaced due to bad performance during occlusion anyways.

There are some attempts to improve our re-identification phase that we did not have the time to experiment with. Since the models we use for re-identification extract features that depend on the background, one option could be to segment and mask the persons within the bounding box. This would simply delete the background and only extract from the actual person.

Our tracking model is struggling with occluded objects, and this might also be possible to be solved by the re-identification model. If the re-identification could split the target into parts like legs, torso and head, this might be a good approach.

When checking if the person already exists in the database or not, we are not using a threshold for the Euclidean distance between the feature vectors. It would be interesting if there is any clustering algorithm who could do a better work at deciding this.

Lastly, one could use the knowledge of the camera topology in the network, which we were not allowed to assume. But might be possible in many cases. One way may be to make it a requirement for the one who installs it to describe the topology to the MTMCT system in some way. Another approach could be to let the system have an initialisation phase where the system learns the camera topology. The system could then create weights for the re-identification phase depending on where the person was last seen and where it appears.

References

Abraham, N. S., Rajan, R. A., George, R. E., Gopinath, S., and Jeyakrishnan, V. (2021). Finding missing child in shopping mall using deep learning. In Suresh, P., Saravanakumar, U., and Hussein Al Salameh, M. S., editors, *Advances in Smart System Technologies*, pages 477–482, Singapore. Springer Singapore.

Axis (2021). Axis history. `https://www.axis.com/about-axis/history`. Accessed: 2021-03-26.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468.

Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

Brasó, G. and Leal-Taixé, L. (2020). Learning a neural solver for multiple object tracking. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6246–6256.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chen, L., Ai, H., Zhuang, Z., and Shang, C. (2018). Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

Ciaparrone, G., Luque Sánchez, F., Tabik, S., Troiano, L., Tagliaferri, R., and Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *ArXiv*, abs/1605.06409.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Han, S., Huang, P., Wang, H., Yu, E., Liu, D., Pan, X., and Zhao, J. (2020). MAT: motion-aware multi-object tracking. *CoRR*, abs/2009.04794.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. volume 2016-, pages 770–778. IEEE.

Hermans, A., Beyer, L., and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737.

Hou, L., Wan, W., Hwang, J.-N., Muhammad, R., Yang, M., and Han, K. (2017). Human tracking over camera networks: a review. *EURASIP journal on advances in signal processing*, 2017(1):1–20.

Leal-Taixé, L., Milan, A., Reid, I. D., Roth, S., and Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016;2015;). *SSD: Single Shot MultiBox Detector*, volume 9905, pages 21–37. Springer International Publishing, Cham.

Luo, H., Jiang, W., Gu, Y., Liu, F., Liao, X., Lai, S., and Gu, J. (2020). A strong baseline and batch normalization neck for deep person re-identification. *IEEE Transactions on Multimedia*, 22(10):2597–2609.

McLean, S. J., Worden, R. E., and Kim, M. (2013). Here's looking at you: An evaluation of public cctv cameras and their effects on crime and disorder. *Criminal justice review (Atlanta, Ga.)*, 38(3):303–334.

Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., and da Silva, E. A. B. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics (Basel)*, 10(3):279.

Previtali, F., Bloisi, D. D., and Iocchi, L. (2017). A distributed approach for real-time multi-camera multiple object tracking. *Machine vision and applications*, 28(3-4):421–430.

Radenovic, F., Tolias, G., and Chum, O. (2019). Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668.

Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., and Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37–52. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

Ye, M., Shen, J., Lin, G., Xiang, T., Shao, L., and Hoi, S. C. H. (2020). Deep learning for person re-identification: A survey and outlook. *CoRR*, abs/2001.04193.

Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. (2020). A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469.

Zhang, X. and Izquierdo, E. (2019). Real-time multi-target multi-camera tracking with spatial-temporal information. *2019 IEEE Visual Communications and Image Processing (VCIP), Visual Communications and Image Processing (VCIP), 2019 IEEE*, pages 1 – 4.

Zhao, L., Li, X., Wang, J., and Zhuang, Y. (2017). Deeply-learned part-aligned representations for person re-identification. *CoRR*, abs/1707.07256.

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

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-06-10

EXAMENSARBETE Multi-Object Tracking in a Camera Network: Cooperation between the components
STUDENT Felix Lundström
HANDLEDARE Pierre Nugues (LTH), Oskar Persson, Simon Molin and Mikael Andreen (Axis)
EXAMINATOR Elin Anna Topp (LTH)

Följa och återidentifiera människor mellan kameravyer

POPULÄRVETENSKAPLIG SAMMANFATTNING **Felix Lundström**

Att bearbeta videor från bland annat övervakningskameror blir allt mer vanligt. I detta arbete undersöker vi om det är möjligt att följa och återidentifiera människor mellan olika kameror i realtid.

Vad gör vi om vi inte hittar vårt barn som gått vilse i köpcentret? Skulle det inte vara skönt om vi kunde ta reda på vart han eller hon tagit vägen? Liknande scenarier tillhör de problem som detta arbete är gjort för att lösa. Ett annat användningsområde kan vara att analysera övervakningskameror på brottsplatser. I vilket fall som helst, så kan det finnas stor nytta av att kunna följa och återigenkänna personer mellan olika kameror.

I bilderna kan vi se ett exempel på när två personer känns igen på olika bilder. Målet är att varje person ska ha samma färg på rutorna oavsett bild, men även samma unika id-nummer.

Uppgiften kan delas upp i tre olika faser: upptäcka personer i en bild, följa varje person inom en kameravy och känna igen personerna mellan de olika kamerorna. Allt detta ska dessutom ske i realtid! Detta betyder att man inte kan använda sig utav delar av videon som sker i framtiden. Detta gör det ännu svårare för systemet att hinna samla tillräckligt med information om en person innan denne behöver identifieras igen.

Att följa personer i endast en kameravy är något man redan kan göra relativt effektivt. Däremot att återidentifiera personer på ett bra sätt, utan att känna till kameratopologin, är ännu ett olöst problem. Dessutom ser varje person väldigt olik ut beroende på vinkel och ljus. Här fick jag försöka sätta en gräns för hur lik en person behöver vara en annan för att säga att det är samma person.

Jag har undersökt hur de olika faserna i systemet kan fungera tillsammans och hur deras prestanda påverkar övriga komponenter. Jag kom fram till att återidentifieringsfasen är den delen som kräver mest fortsatta studier. Jag testade varje del av systemet för sig och även hur de fungerar tillsammans med övriga komponenter.

När man extraherar information från upptäckta personer, så kan man få med egenskaper som varierar beroende på omständigheterna. T.ex. så ser personer i regel väldigt olika ut från olika vinklar, eller om det är mörkt eller ljust. Jag testade om man kan förbättra prestandan genom att transformera egenskaperna i databasen till några som är stabila över tid, oberoende av olika parametrar. PCA tillförde inga förbättringar i den utsträckningen jag testade. Däremot lyckades jag bygga en fungerande grund för ett komplett system som lyckas utföra uppgiften.