FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Multiple Object Tracking in Top-View Camera Video-Sequences
Student:	Petr Šejvl
Supervisor:	Ing. Filip Naiser
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2022/2023

Instructions

Managers of complex buildings (like shopping malls, office space) deal with various tasks (e.g., minimization of wait time, queue prevention, advertisement placement). The pedestrian detection system, gender and age prediction, and inter-camera identity preservation are crucial to know the customer. In iC Systems.ai, s.r.o. we are developing such systems.
A student will improve our methods for pedestrian tracking in top-view video sequences. He will do tracking by detection and assignment.
At first, he performs a literature review on this topic. Next, he will design assignment cost functions (based on features like position, descriptor, detection score), implement them, and evaluate them on our datasets. He will also investigate neural network-based methods for assignment problem optimization and discuss their usage.
He will implement and design these methods while considering limited computation resources since they must run online on our sensors.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 28 December 2021 in Prague.

Bachelor's thesis

MULTIPLE OBJECT TRACKING IN TOP-VIEW CAMERA VIDEO-SEQUENCES

Petr Šejvl

Faculty of Information Technology
Department of Theoretical Computer Science
Supervisor: Ing. Filip Naiser
April 27, 2022

Contents

List of Figures

List of Tables

List of code listings

In the first place, I would like to thank my supervisor Ing. Filip Naiser. I want to thank him not only for his very inspiring advice regarding this thesis, but for all the time he has been my supervisor in iC Systems.ai, s.r.o. This collaboration really showed me how exciting a job can be.
I would also like to express my appreciation to Ing. Pavel Hrabák, Ph.D. for his valuable remarks and to Eliška Růžičková for her help with my English.
My thanks belong to my family for all the support throughout my studies.
Last but not least, I need to thank all my friends who helped me with my studies so much. I would like to name Terezie Hrubanová, Ondřej Staníček and Ladislav Floriš.

Declaration

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

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

In Prague on April 27, 2022 .....

Abstract

Multiple Object Tracking in video sequences has great scientific and commercial potential. Multiple Object Tracking helps for example with observing the movement of customers and monitoring the occupancy of buildings. In the thesis, we investigate how to accurately and effectively get pedestrians' trajectories from their detections. The thesis is situated in malls and office buildings. The system is optimized for counting pedestrians entering and leaving the camera view through the predefined areas.

In the thesis, we propose an assignment cost function based on machine learning and discuss the data association strategies. To get the best results possible, we propose heuristic functions, which we optimize and test together with the algorithm.

The output of this thesis is an effective and accurate algorithm for multiple pedestrian tracking in video sequences. The algorithm is developed for usage in iC systems.ai, s.r.o. software.

Keywords computer vision, multiple object tracking, pedestrian tracking, tracking-by-detection, data association, iC systems.ai, s.r.o

Abstrakt

Sledování pohybu objektů ve videosekvencích nabízí velký výzkumný i komerční potenciál. Jedná se například o sledování pohybu zákazníků a zaplněnosti budov. V rámci práce zkoumáme jak co nejpřesněji a nejeefektivněji získat z detekovaných pozic chodců v čase jejich správné trajektorie. Práce se zaměřuje na prostory obchodních center a kancelářských budov. Systém je optimalizován pro přesné započítávání příchodů a odchodů ze záběru kamery skrz předem definované zóny.

V rámci práce navrhujeme přiřazovací funkci založenou na strojovém učení a diskutujeme přiřazovací strategie. Pro dosažení co nejlepších výsledků pak přidáváme další heuristické funkce, které v rámci práce se zbytkem algoritmu odladíme a otestujeme.

Výstupem práce je pak efektivní a přesný algoritmus na sledování chodců ve videosekvencích určený k použití v rámci systémů firmy iC systems.ai, s.r.o.

Klíčová slova počítačové vidění, multiple object tracking, sledování chodců, tracking-by-detection, asociace dat, iC systems.ai, s.r.o

Acronyms

MOT	Multiple Object Tracking
RFC	Random Forest Classifier
TP	True Positive
FP	False Positive
FN	False Negative

Chapter 1
Introduction

Multiple object tracking (MOT) is one of the key tasks in computer vision with great academic and commercial potential. The use of MOT ranges from tasks like studying ants and their hierarchies and sports' video analysis to autonomous vehicles. The goal of MOT is to detect objects in a video and track their movement in time. One of the fields of use is tracking pedestrians in buildings, like shopping malls, office spaces, etc. This thesis focuses on these environments. Having the information about the movement of pedestrians gives our clients numerous possibilities. They may use it to improve their goods or advertisement placement, observe the occupancy etc. Namely the occupancy measurement was an important application during the Covid-19 pandemic. The main focus of this thesis is on the strategies for connecting pedestrains' detections into their tracks.

There are various approaches to performing the task of MOT. In this thesis, we follow the tracking-by-detection paradigm. First, objects, pedestrians in our case, are detected in the video. These detections are then arranged into the tracks of those objects. This way we perform the MOT task. In the thesis we have the system to obtain detections already available, so we focus on the organization of the objects rather than the detection process.

The data association in MOT brings multiple challenges. There may be occlusions between the objects, the system for detecting objects may fail by either missing an object, or detecting a non-existing one etc. This thesis tries to solve as many of those issues as possible and provide the best possible solution. To do that, we make great use of our domain knowledge of the problem and consider the state of the art approaches around MOT.

1.1 Objectives and Goals

The goal of the thesis is to propose methods for online multiple pedestrian tracking in top-view video sequences between defined entrances. We will be gradually getting observations in frames F_1, ..., F_n each containing a set of pedestrian detections D_1, ..., D_m. The goal is then to link the detections of the same pedestrians to their tracks T_1, ..., T_k and return them gradually once the tracked pedestrians leave the area of interest. Each of these tracks has to end in a predefined area (entrance). This is the main difference between our approach and most of the other MOT approaches, which do not work with entrances and only try to track the object over the view of camera.

To reach our goal, we need to focus on the proposition of the assignment cost functions to get the cost of joining a detection to a track and data association strategies based on the output of the assignment cost function. As a superstructure above the data association, we propose heuristic functions to prevent or support certain situations based on our domain knowledge.

Many of the methods mentioned above use some thresholds, like maximal cost for acceptable assignment, so the next objective is to optimize values of those thresholds.

Precision of proposed and implemented methods will be tested on our datasets. Our team of annotators provided over 75,000 annotations of pedestrians leaving the area of interest in 763 testing datasets. The precision is mostly tested by counting True Positive passages (TP - True Positive), missed pedestrians (FN - False Negatives) and extra pedestrians (FP - False Positive, this means the tracks of pedestrians, which the algorithm counted, but actually are not in the video). Another relevant metric is absolute counting error, which allows to balance the FP and FN. By balancing we mean a situation where a missed passage and an extra passage compensate for each other, so instead of two mistakes, no mistake is counted.

We will also investigate the state of the art literature on topic of the neural network-based [1] methods for assignment problem optimization. We will discuss the usage of those methods in our system. To fulfill the set objectives, this thesis is organized into the next 5 chapters:

- the Theory chapter introduces the key concepts needed for further description of the algorithm and discusses related work,
- the Method chapter is the most important one, it describes proposed methods and algorithms,
- in the Optimization chapter we focus on the optimization of algorithms described in the Method chapter,
- results are tested by experiments described in the Experiments chapter,
- the final Conclusion chapter provides a summary of the thesis and its contributions to the industry.

Chapter 2
Theory

This chapter covers the explanation of the key concepts used in this thesis and the related work on the subject of Multiple Object Tracking (MOT). We will also introduce the environment, in which we shall be tracking pedestrians' movement.

2.1 Area of Interest, Entrance

The Area of interest is the area where the target can be detected. It does not always make sense to track pedestrians over the entire area available from the camera. As an example we can take a camera, which is supposed to capture one floor but sees a bit of the lower floor too. Detecting and tracking pedestrians on the lower one would be useless and could even cause problems.

The Entrance marks the location where people can enter or leave the Area of Interest. The Area of Interest and the Entrance can be seen in the figure 2.1.

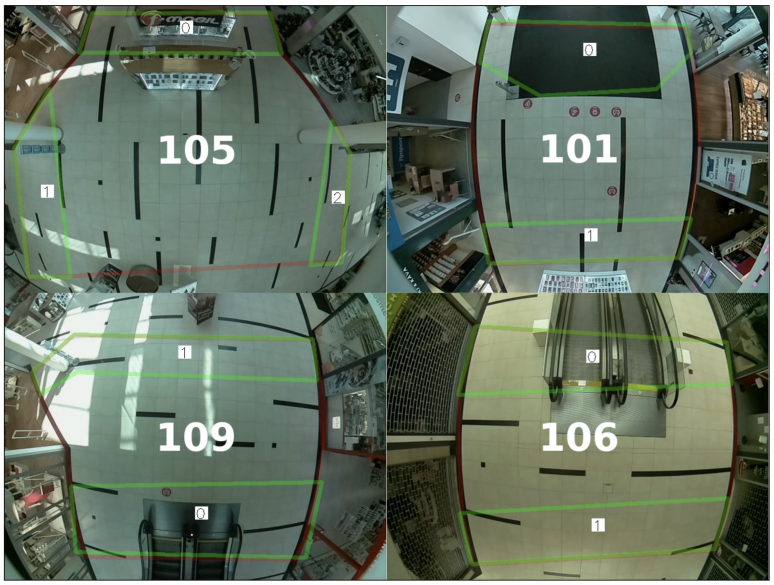


Figure 2.1 Example of the environment showing the Area of Interest (red) and the Entrance definition (green). Image source: [2].

2.2 Detection

A Detection is an object representing the target detected in a frame of the video sequence. It keeps some metadata, like:

- bounding box - coordinates of the rectangle around the detection,
- ground point - an estimation of the person's feet,
- descriptor - representation of the target look,
- other data, like gender, class, etc.

All the work around object detection is the work of my colleagues, mostly my supervisor Ing. Filip Naiser [3].

2.3 Track

A Track is a sequence of chronologically sorted detections [2]. In the ideal case, the track is constructed from all the available detections of one target and does not use any detections of other targets. Like a detection, a track also keeps some metadata. Most of them are aggregations of the track's detections. A Track should start in the area of entrance, but does not have to. To be counted and therefore marked as completed, the track has to end in an entrance (different from the start entrance, if start entrance is available). The start entrance is called the Source and the end entrance is called the Sink. Tracks which do not end in the entrance are called Fragments.

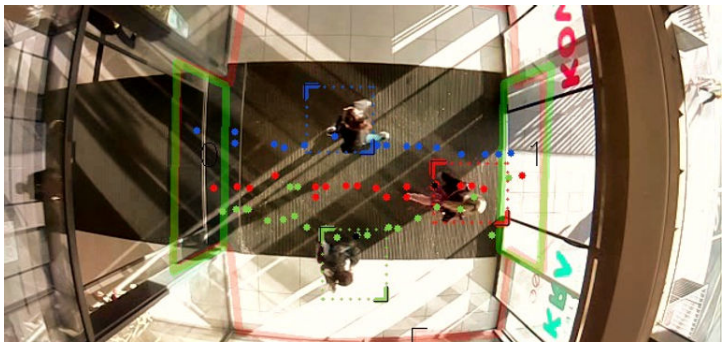


Figure 2.2 This figure captures 3 pedestrians' tracks. Dots represent the ground points (an estimation of the pedestrian's feet) and should follow the path they walked on. Rectangles represent each tracks' bounding box of the detection captured at frame F. Note that the image is directly from the camera used in our system, therefore its quality is quite low. Image source: [4].

2.4 Pedestrian Counting

To handle the occupancy task, we need to be able to count the leaving and arriving pedestrians. When the pedestrian's tracks can not be further expanded and the last detection lays in an entrance E, which is different from the source entrance, we can count one passage through the entrance E. Whether this means that the pedestrian came or left depends on the entrance and is left for further processing. We count the number of pedestrians passing from one entrance to another.

The current occupancy level is then counted as the number of arriving pedestrians minus the number of leaving. The system's accuracy is tested using datasets, where the passages were annotated by our team. For each passage there is an annotation at the very moment the passage happened.

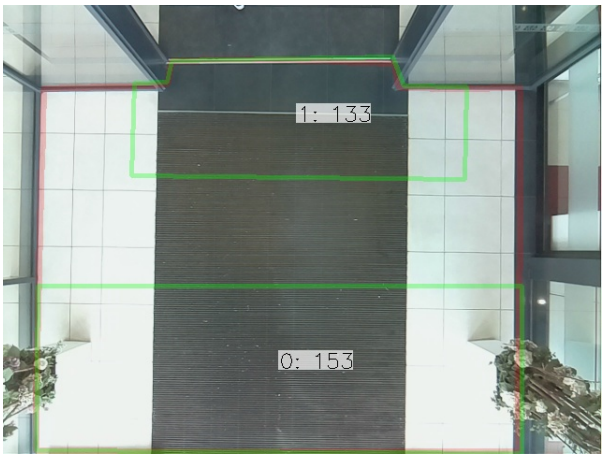


Figure 2.3 In this picture you can see an example of the final counts after the video was processed by our algorithm. The number on the left side of the colon represents the entrance ID. On the right side of the colon is the amount of observed passages.

2.5 Related Work

Multiple Object Tracking (MOT) has recently been the focus of many researchers. There have been various approaches proposed for solving the task based on different assumptions about it. The MOT can be viewed as an online task, an offline task, we may do the detection-based tracking, or even a detection free tracking [5]. An extensive literature review about the various aspects and approaches to MOT has been published by Wenhan Luo et al. [5].

Drew Linsley et al. even came up with an MOT algorithm [6], which does not rely on the appearance at all and performs the tracking only based on the motion model.

2.5.1 MOTChallenge as a Signpost for the State of the Art Literature

The MOTChallenge [7] is an online benchmark to evaluate the MOT algorithms. It comes with a few different datasets to test the algorithm efficiency. For most of the state of the art algorithms there is also a paper available describing them, or even their implementation. We discuss a few of those papers in this section.

Measuring the accuracy of MOT algorithms is not a straightforward task and there are multiple possibilities for doing it. To help with the issue, Christoph Heindl has implemented a Python library based on the previous research [8].

One simple but powerful trick is described by Yifu Zhang et al. [9]. For most of the MOT algorithms, if our confidence about the detection's correctness is low (meaning it might actually just be a noise), the association part will ignore it.

ByteTrack [9] comes with a two round association, where only high confidence detections are considered in the first round. For the second round, even the low confidence detections may

be used to extend tracks which were not extended in the first round. Note that the second processing also has some threshold, limiting using significantly weak detections.

Most of the state of the art is moving towards the usage of Neural Networks [1], which we shall discuss in the very next subsection.

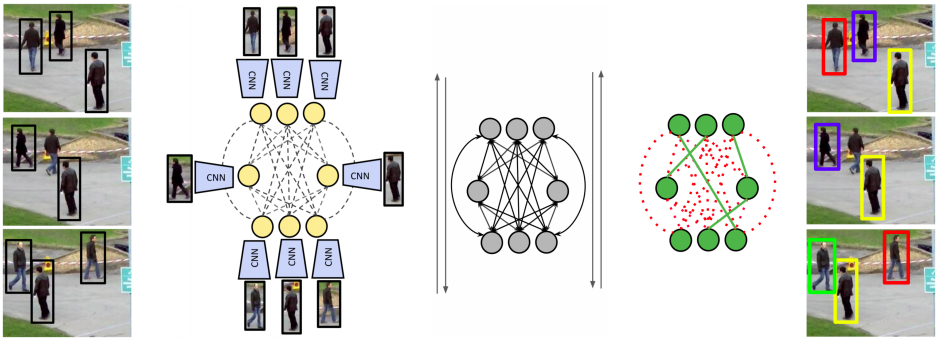
2.5.2 Neural Networks for Multiple Object Tracking

The state of the art in Multiple Object Tracking (MOT) has recently been moving towards the usage of Neural Networks. There are numerous possibilities to incorporate the Neural Networks into the task. To the best of our knowledge, the approaches can be divided into those performing object detection and data association separately and to those performing it jointly.

We provide a literature review of both of these approaches and then consider their usage in our system.

2.5.2.1 Neural Networks for Data Association

One approach which uses the tracking-by-detection paradigm with keeping the detection and association parts separated was proposed by Guillem Brasó and Laura Leal-Taixé [10]. The solution described in this paper benefits from the fact that the MOT in the tracking-by-detection paradigm can be naturally formulated as a multipartite graph. On this graph is then performed neural message passing. Based on the results of neural message passing, the edges of the graph are classified into active and inactive. Active edges represent the valid associations. Note that the classification may produce conflicts, like assigning one detection to more tracks. Those conflicts must be resolved after the classification.



(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

Figure 2.4 Learning a Neural Solver for Multiple Object Tracking - overview of the method. Image source: [10].

This approach has a few promising advantages. It can reason over N frames at once, so the approach is not frame to frame greedy. Moreover, final edge classification this way considers all active pedestrians and can adapt to the situation. On the other hand, reasoning over more frames brings disadvantages too.

One problem is that to reason over N frames, we need to have detections from them. This means that the online tracker has to run with some time delay. Another, perhaps more significant disadvantage, could be the time demand. The message passing brings more operations and computations. Also, the algorithm described in this paper classifies each edge N times and aggregates the classification results, which could in time yield demands unbearable for the system with strictly limited computation resources.

A slightly different approach compared to the one mentioned above is introduced by Jiawei He et al. [11]. Instead of conventionally modeling the graph between progressing tracks and

detections in the next frame, capturing possible associations, they model two graphs G_1 and G_2 and then perform graph matching between them.

G_1 is a complete graph, where the vertices are progressing tracks. G_2 is a complete graph too, but its vertices are the detections in the frame following the frame of progressing tracks. To find the best associations, the algorithm performs a neural graph matching between G_1 and G_2. The most similar vertices from G_1 and G_2 then represent a pair of a track and a detection, which shall be associated together.

Creating graphs between the objects in the same frame helps to capture a lot from the context. The positive effect of the context processing can be also seen in our RFC models' feature importances, see figure 3.3. The overview for this algorithm is captured by figure 2.5.

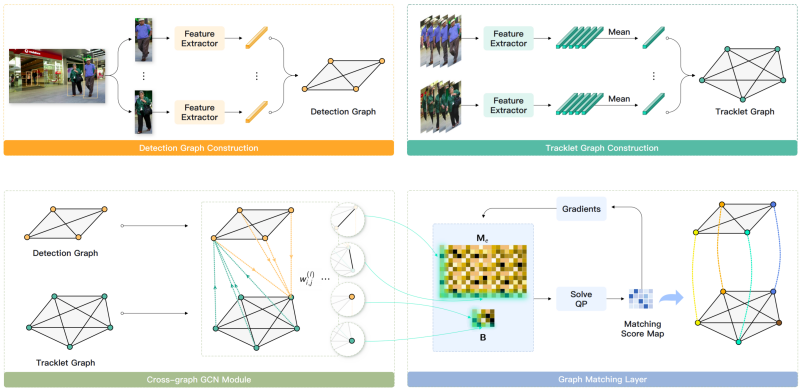


Figure 2.5 Learnable Graph Matching: Incorporating Graph Partitioning with Deep Feature Learning for Multiple Object Tracking - overview of the method. Image source: [11].

2.5.2.2 Solving the object detection and data association jointly

Some of the state of the art solutions base their approach on solving object detection and data association jointly. It is a powerful tool, because those two components are clearly dependent on each other [12].

Including tentative tracking results can help with object detection by providing information about the next expected detection. The data association then benefits from the improved detection system [13]. The algorithm described in [13] builds upon this strategy.

An interesting alternative to tracking by detection paradigm is described in [14]. Their *tracking by attention* relies on the usage of transformers.

2.5.2.3 Usage for iC Systems.ai, s.r.o.

Based on the knowledge from the state of the art literature, there is no doubt that the Neural Networks are a good step for iC Systems.ai, s.r.o. too. Even though using them for data association solely like in [10] could improve our results, the time demands would most likely be unbearable. Even if we could deal with the computation complexity, the recent state of the art mostly solves the object detection and association jointly and therefore it makes a better sense to aim our future steps there.

The [11] brings enlightenment about the importance of context information. Even though we do not follow their approach with graph matching, we are taking an inspiration there trying to work with the context more.

Solving object detection and data association jointly is a promising approach. Sadly, it has not been possible to integrate it into our recent architecture [15], but it is a subject of discussion for future development.

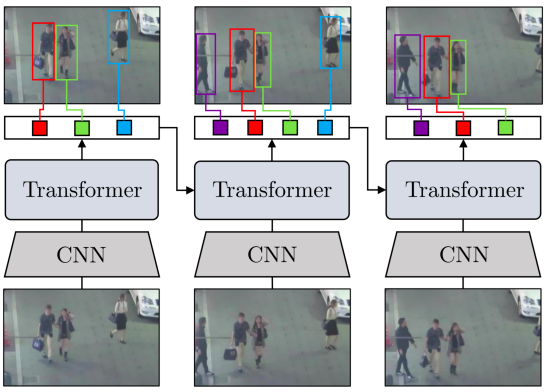


Figure 2.6 TrackFormer: Multi-Object Tracking with Transformers - overview of the method. Image source: [14].

Chapter 3
Method

This chapter gives an overview of all implemented methods used in order to handle the Multiple Object Tracking (MOT). Firstly we describe the basic idea of the algorithm, then we dig deeper to investigate its key components. The Algorithm Overview describes how will all the pieces fit together. We provide a step by step description of our approach to MOT.

The Assignment Cost Function section discusses how to take two detections in consecutive frames and get the cost for their belonging to the same pedestrian. The next section, the Detection Association Problem then uses these assignment costs to actually create the tracks.

Heuristic function is the last part of this chapter. We discuss the usage of our domain knowledge. Some of the heuristics proposed there are generic for MOT, while the rest makes sense only for the counting problem.

3.1 Algorithm Overview

The main functionality is encapsulated in a class called RFCPrasator [1]. The core method *process next frame* takes a list of detections and processes them. These new detections can be either associated with the progressing tracks, or start the new tracks themselves. The method also notices if some tracks have just finished and returns them in such cases. To reach this goal, the method must gradually update some members of RFCPrasator.

Essential members kept by RFCPrasator are:

- *active tracks* - a list of progressing tracks,
- *unfinished track fragments* - a list of tracks for which neither progressing or finishing is possible without the use of special heuristics discussed later,
- other metadata like configurations, environment info etc.

The *process next frame* method works in 7 phases. In the list below we provide a brief overview of these phases. Core phases will be discussed later in more detail:

1. preprocess obtained detections,
2. filter *active tracks* to relevant groups (like *active tracks*, *unfinished tracks fragment*, etc),

[1] The name RFCPrasator has a bit of a funny story. One of our colleagues once translated the word "tracker" to Czech as "trasátor". So the next attempts took an inspiration there and the very next version was Called Trasator. Unfortunately, we had no success with Trasator and the algorithm has gradually transformed into the recent version, RFCPrasator.

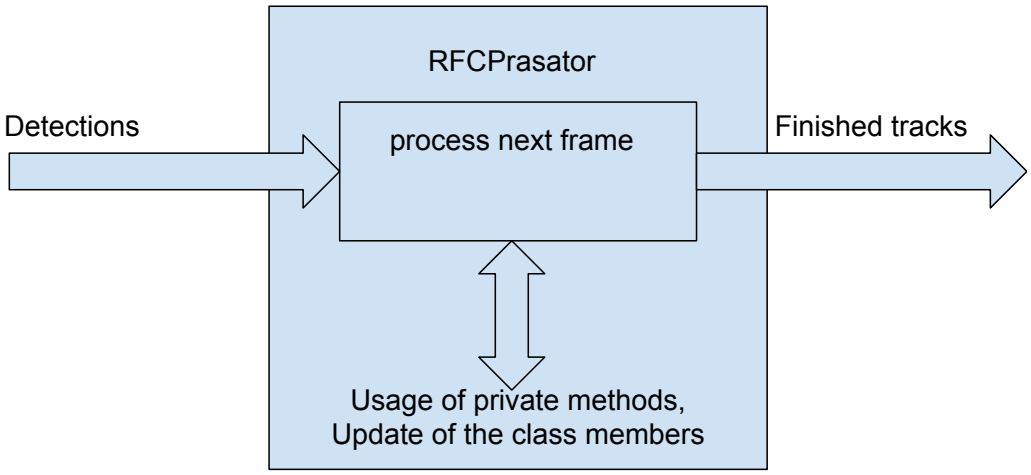


Figure 3.1 A brief overview of the algorithm architecture.

3. context update,
4. perform the data association of the obtained detections and *active tracks*,
5. with some of the detections which failed to be associated we shall initiate new tracks,
6. tracks postprocessing,
7. tracking postprocessing, this step is related to some heuristics like the one from [9].

The return value of this method is a list of *finished tracks*. This list contains all the tracks that were marked as finished in step 2 of *process next frame*. This way the system can gradually obtain the finished tracks from RFCPrasator and handle the online MOT. The pseudocode for this method can be seen in 3.1.

Code listing 3.1 Process Next Frame Pseudocode

```
def process_next_frame(self, detections):
    detections = self.process_obtained_detections(detections)
    finished_tracks = self.remove_old_tracks()
    self.update_context(detections)
    unused_detections = self.extend_existing_tracks(detections)
    self.start_new_tracks(unused_detections)
    self.postprocess_tracks()
    finished_tracks += self.postprocess_tracking()
    return finished_tracks
```

3.2 Assignment Cost Function

Perhaps the most important part of the whole data association process is the assignment cost function. The input for this function is a track T, a detection D in the following frame and all the other detections in the following frame. The output of this function is the cost for associating the detection D with the pedestrian tracked by track T. By providing this cost, the assignment cost function allows us to do the re-identification of the pedestrian from track T in the following frame F of the detection D. The track is constructed based on this assignment cost function. Good assignment cost function will help with tracking by doing two things:

1. cost for the correct assignment should be much lower than cost for association between different pedestrians,

2. we can set a cost based threshold, above which we consider the association as wrong.

The re-identification cost can depend on numerous features. Most of the features are based on the comparison between the last detection D_{last} of track T and the detection D we are trying to associate with T. Other features are capturing the context.

From the comparison of D and D_{last} we perceive their geometrical distance, difference of the appearance descriptors and others. The *context dependent features* describe the context, under which the assignment happens. We are working with some metadata about T, its expectation of the next detection whereabouts and other detections in the frame of D.

The quantity of the features makes it hard to handcraft a relevant assignment cost function, so it is relevant to use a pre-established framework. My supervisor Ing. Filip Naiser proposed to use the Random Forest Classifier (RFC) [16], to solve the task for us.

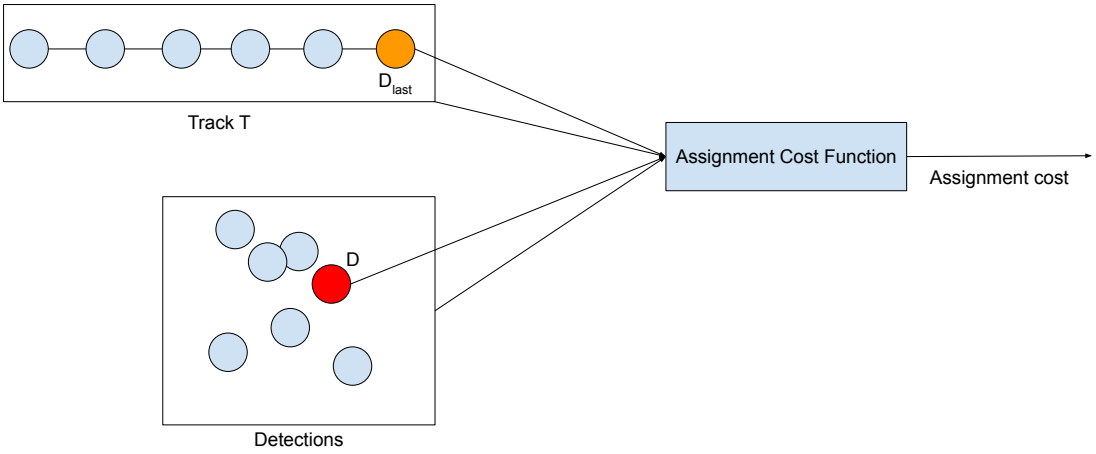


Figure 3.2 Diagram of the Assignment Cost Function input output architecture. Most features are based on the last detection D_{last} of track T and new possible assignment detection D. We also add context information about the scene in the frame of detection D and several track T related features.

3.2.1 Random Forest Classifier

To train RFC models, our annotator team has provided required annotations. The total count of the annotations goes up to 93,830.

The algorithm does not strictly require to associate the track with a detection from the very next frame. If a track is not matched with any detection from the very next frame F, it can wait and try to find the match in the frames F + 1, ..., F + 4. This is a very useful tool for situations where the detector for some reasons fails.

Since the situations are different based on the frame gap, we have trained a separate RFC for each frame gap.

You can check the accuracy of the RFCs based on the frame gap in table 3.1.

The RFC from Sklearn library [17] holds a member called *feature importances*. See figure 3.3 to learn more about the feature importance of our model.

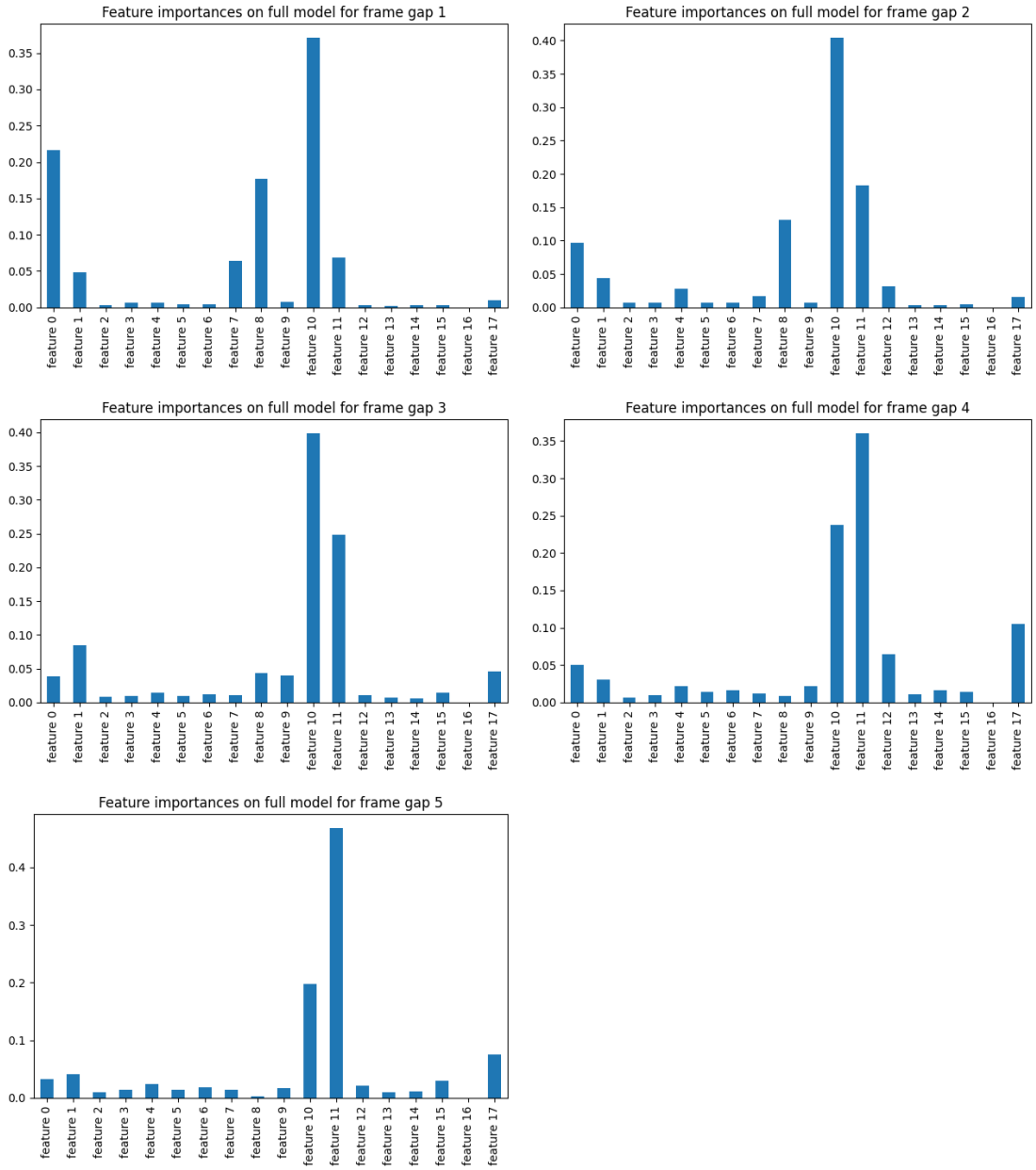


Figure 3.3 Captures the feature importance in RFC trained for each frame gap. You can see how some importances change with the rising frame gap. Some of the most relevant features due to those importances are:

- feature 10 captures the distance to the nearest detection different from the one we are trying to assign,
- feature 11 captures the distance between the track's prediction for the next detection's position and the nearest detection different from the one we are trying to assign,
- feature 0 is the distance between the track's last detection and the detection we are trying to assign,
- feature 17 represents the absolute difference between the confidence of the track's last detection and the potential new detection,
- feature 4 is there for the difference in appearance.

It is interesting to see that the appearance seems to be quite irrelevant. Another interesting thing to see is that as the frame gap increases, the motion prediction becomes progressively important.

Table 3.1 RFC accuracy.

Frame gap	Train set accuracy	Test set accuracy
1	0.942	0.941
2	0.907	0.905
3	0.868	0.871
4	0.841	0.840
5	0.815	0.811

3.3 Detection Association Problem

The assignment cost function provides us with essential values for the data association. Having the assignment costs, we still need to decide which association should be active and which not. This section talks about this problem and how to solve it. Firstly, we shall formulate the problem and then describe two possible solutions for it.

3.3.1 Detection Association Problem Formulation

The input for the problem is a set of progressing tracks T_1, ... T_n and a set of detections D_1, ..., D_m in some frame following frame F (for each T_i is $Frame(T) < F$). For each reasonable pair of track and detection the cost for their association is provided. By reasonable pair we mean a pair fulfilling handcrafted conditions, such as being geometrically close enough (this threshold is mainly to speedup the whole process) etc.

The output is an association of tracks with detections, where each track t_i is associated with at maximum one detection and vice versa. This association should be as good as possible in optimizing some criteria, two of them are discussed further in this chapter.

We discuss two approaches, each optimizing a different criterion. The first one uses the greedy approach and the second one uses the Hungarian Algorithm [18].

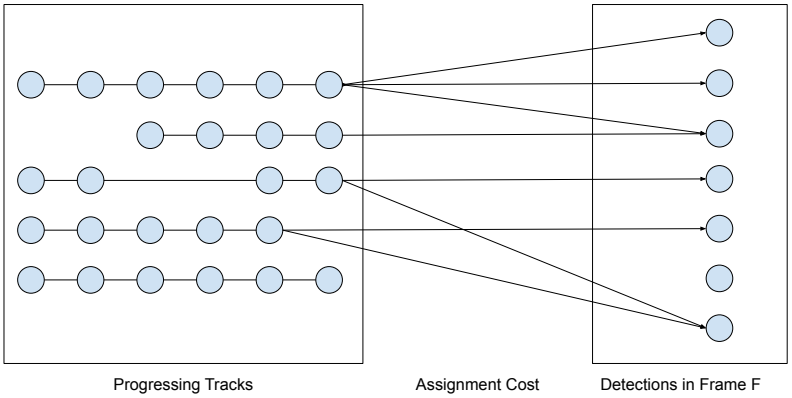
See figure 3.4 for graphical illustration of the Detection Association Problem. The proposed methods are later compared in section 5.3. Their difference is also presented in figure 3.5.

3.3.2 Greedy Association

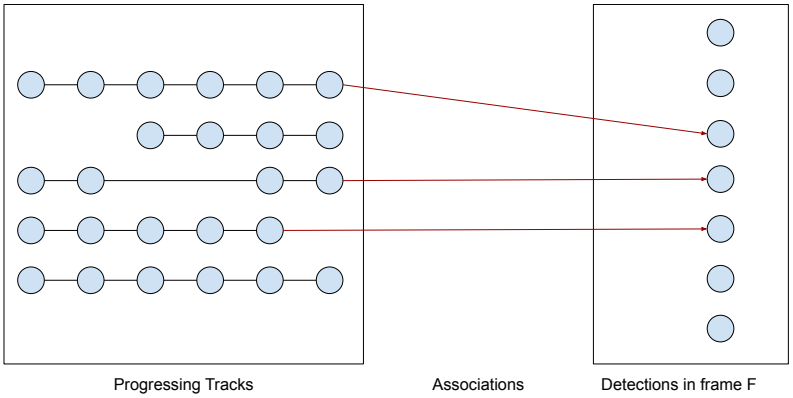
The Greedy association algorithm was the first one used in iC Systems.ai, s.r.o. and has been proposed and implemented by my supervisor Ing. Filip Naiser [19]. This subsection is going to describe it.

The greedy approach is based on processing the edges based on the cost from the least to the most costly. The advantage of this approach is in its simplicity, only a sorting algorithm is required to sort the edges by their cost. As this can be done in $O(n * log(n))$, it is also asymptotically efficient. As the best possible association is always preferred, this approach should yield some really strong tracks.

The disadvantage is an ignorance towards worse association. Consider a situation where you can do either one very good association and one slightly worse, but completely miss the third possible association. On the other hand, you could also do all 3 associations, but miss the best one and include the worse one. Greedy association would follow the first situation. It is hard to argue which approach is better. To follow the second situation resulting in all doing all the three associations, we present an association using the Hungarian Algorithm in the very next subsection.



(a) This bipartite graph represents possible input for the Detection Association Problem. The arrows between Tracks and Detection show possible association and are weighted by the costs.



(b) Output for graph 3.4a could look like this. The reds arrows represent the final active associations.

Figure 3.4 Graphical representation of the Detection Association Problem

3.3.3 Association Using Hungarian Algorithm

The Hungarian Algorithm can solve the data association problem optimally in O(N^3), where N is the number of the nodes [18]. Optimally means that the overall sum of costs for associations is as small as possible, while associating as many nodes as possible (note that optimal does not have to mean right for our Detection Association Problem). The advantage there may be the preference of optimal association. This should yield rather more cooperation during the association.

The disadvantage may be the likelihood of the loss of strong associations in favor to have more weaker ones and a worse asymptotic time complexity.

3.4 Heuristic Functions

Knowing the environment and common situations there allows us to handcraft a series of heuristic functions. Such functions can reason above the assignment cost function and the subsequent association to improve the precision of the algorithm.

Over the time spent on the thesis, we have come up with many different heuristics. We discuss all the developed heuristics in this section and later in 5.1 prove their impact in experiments.

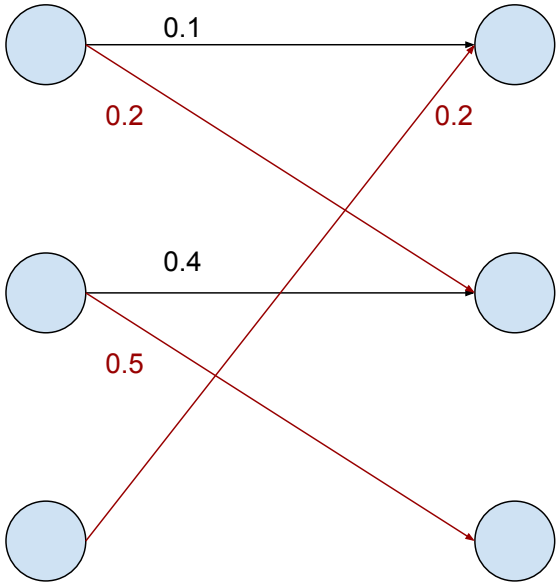


Figure 3.5 In this figure we present the difference between the greedy approach and the use of the Hungarian Algorithm. Consider a situation described by the graph. The greedy approach would make the associations noted by black arrows active, while the Hungarian algorithm would use the red.

3.4.1 Repetitive Processing

We have proposed two heuristics which prosper from some kind of repetitive processing. Repetitive processing may help to see things missed in a one-way greedy approach. The downside of this is increased computational complexity and a risk of some falsely positive observations.

3.4.1.1 Two Rounds Tracking

This heuristic was designed as a compensation for the greedy frame-frame approach of our Tracking algorithm. The observation shows that some of the True Positives (TP) are getting lost because stronger TPs overshadow them and it is not possible to finish the weaker ones. Clearing out the detections of strong tracks then makes it possible to finish even the weaker ones.

After every constant period of frames, all residual detections are frame by frame passed to the next instance of RFCPrasator to perform tracking on them. The finished tracks from this tracking are added to those from the original tracking and counted normally.

If a reasonable constant is used, first level tracking has enough time to separate the tracks from the residual detections and the time gap for second level tracking is negligible.

3.4.1.2 Usage of Low Confidence Detections

This heuristic is based on the ByteTrack paper published by Yifu Zhang et al. [9].

The approach first uses high confidence detections to perform normal tracking with starting new tracks etc. After this step, it comes back to work with low confidence detections. All the tracks for which the association in the first step failed can be associated in this second step. Detection Association is the only thing that happens in the second step.

The main advantage described in the ByteTrack paper is a bigger chance to use the occluded (and therefore low confidence) detections too. As our goal lies in keeping the track between entrances and we do not care about missing a small number of detections on the way, this

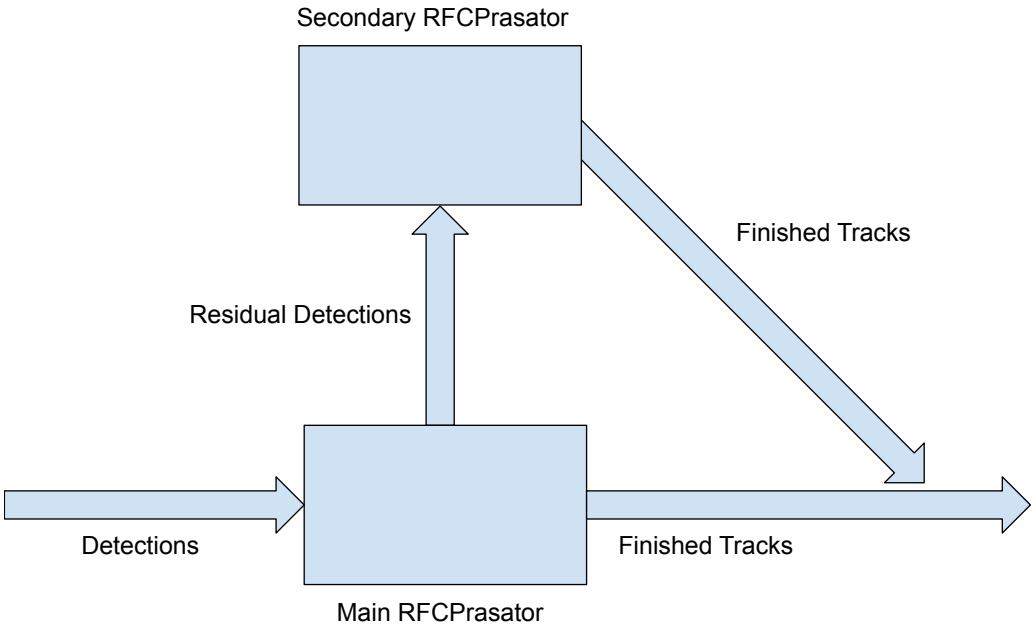


Figure 3.6 Architecture of the heuristic performing a Two Round Tracking

heuristic did not bring much improvement for our solution. The ByteTrack paper is also discussed in related work section 2.5.

3.4.2 Threshold Based Heuristics

This subsection batches together some simple heuristics based only on a comparison with a threshold.

3.4.2.1 Flen Threshold

This discriminative heuristic enforces a strict preference of some tracks to others in the detection association process. The preference is based on the track's length and is designed to disadvantage both short and long tracks.

It is reasonable to disadvantage them, because the short and long tracks may be just noises and could damage the creation of TPs.

3.4.2.2 Max Track Costs

The tracks keep metadata, mostly some aggregations of the measurements on detections it is created from. This data gives us the possibility to evaluate the quality of the track. Based on the quality we can decide whether the track is acceptable or not. The data includes:

- scores of detections used in a track,
- the cost of detection associations,
- the cost of appearance differences between the pairs of consecutive detections in the track.

For each item, we provide a threshold above which the track will not be finished and therefore will not be counted.

3.4.2.3 Distance and Count Limits

Distance and Counts Limits is a special type of Threshold Based Heuristic and therefore we devoted a special subsection to it. We are using multiple limits to ensure the quality of the track before we can finish it. Each track has to gain a satisfying value for each of those values, before it can be finished and counted up:

- minimal traveled distance,
- minimal count of detections the track consists of,
- minimal triangle distance, which measures the distance between the start and the middle of the track and between the middle and the end, see figure 3.7,
- the distance between the start and the end.

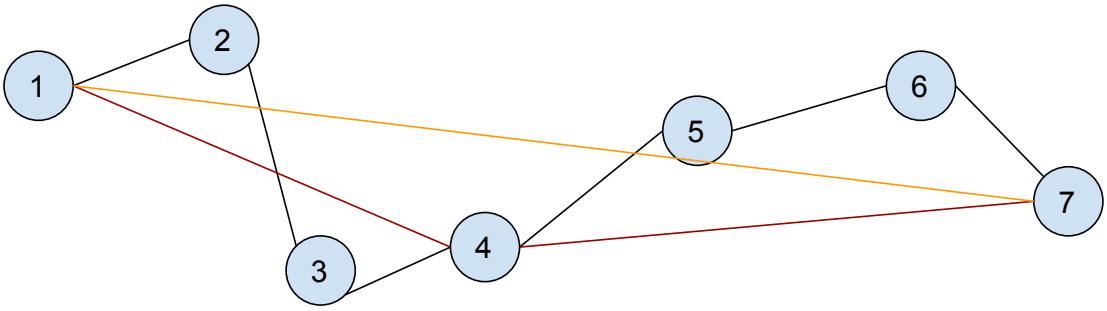


Figure 3.7 This figure presents the distances we follow in the Track. The traveled distance is the sum of the distances between each consecutive detections (black lines). The triangle distance is shown by the red line. The distance between the start and the end refers to the orange line.

We set three other limits. The first assures that the distance between the objects to be associated is limited. The second one handles the distance between the new object and its expected whereabouts, see 3.4.4 to learn more.

The third one takes action when the distance between the start and the end decreases. In such case we enforce the track to finish.

The last mentioned heuristic brings some controversies, as it may theoretically produce problems, rather than improvements. This can be seen in figure 3.8.

3.4.3 Tracks Without the Source Entrance

The original intention was to track the pedestrians between the entrances. After a while of development, we realized that some mistakes arose because it was not possible to begin a track in the entrance (there may be occlusions etc.). To compensate for this problem, we decided to not strictly require the track to start at the entrance, but rather anywhere in the area of interest.

However, such tracks are not ideal and may cause problems. We are therefore proposing two limitations for such tracks. If a track without a source wants to enter the entrance (meaning most likely its sink entrance), it has to have traveled a certain distance in the no-entrance zone first. This should prevent counting of pedestrians just walking around the entrance. The second limitation works similarly to the first, only it processes the amount of the detections instead of the traveled distance.

The tracks without the source entrance are also limited just like the other tracks.

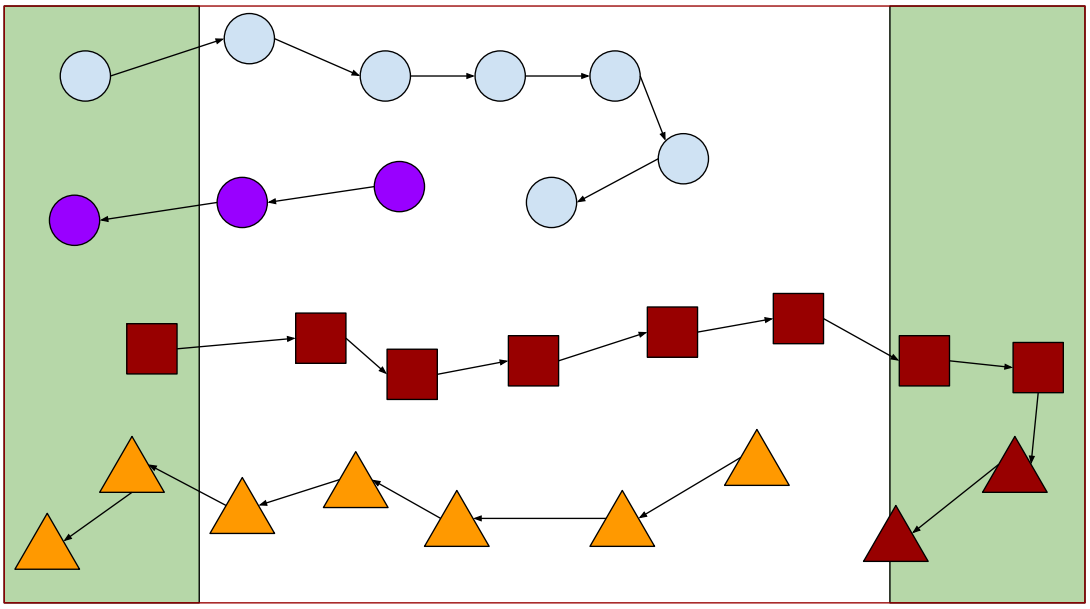


Figure 3.8 The Diagram captures both situations when the heuristic preventing the return success and fails. The red border represents the area of the interest where we track the objects. The green rectangles are the entrances. The same pedestrians are presented with the same shape, while the shape's color shows us the identity assigned by the algorithm.
The pedestrian presented by the circle goes from the left to the right and then returns. In this case, the heuristic fails by disconnecting the pedestrian tracks.
Bellow this scenario we have a square track going also from left to right and leaving the area of the interest there. At the very same moment, a triangle pedestrian comes from the right to the left and the algorithm may falsely assign it the same identity. If this would happen, the heuristic would successfully intervene in disconnecting the square from the triangle.
Overall, the heuristic was proved to be efficient by numerous experiments.

3.4.4 Predictions for the Next Detecion, Kalman Filter

If we observe a pedestrian for a while, we can do some predictions of their next movements. Simply taking the pedestrian's mean direction, mean velocity and the last position could do the trick. On the other hand, such a simple approach could not quickly react if the pedestrian stops or turns around.

My supervisor Ing. Filip Naiser proposed using the Kalman Filter [20] to tackle this problem. He provided the needed implementation and then I integrated it into our solution.

The Kalman Filter is, to the best of our knowledge, mostly used to compensate for mistakes in measuring. Based on the previous measures and recent measure, it provides an estimation of where the recent measure should be. The position of the recent measure can be estimated based on this prediction.

We use the Kalman Filter only for its predictions. The distance between the obtained detection (measure) and its predicted whereabouts is useful for the RFC model and for limiting associations. If the distance is too big, we will not allow the association.

3.4.5 Static Detection Filter

Mostly with older versions of the Detector, we had to deal with static FP detections. One characteristic of such Detections is that they occur with a big frequency while staying in the

same place.

To get rid of them, we are saving the detections for a constant amount of frames (e.g., 50). Then we take the next detections from a few frames as a test (e.g., 10). If we find a place, where the detections are occurring too often with too big intersections, we consider it to be a source of static FPs.

For the next period of observing the train and the test detections, detections around the places marked as a source of static FPs are treated differently. Such detections can either carry the information about the intersection with the source of the static detections, or can even be ignored.

Note that even though the filter was meant to filter out static FPs, it can not differentiate between the FPs and TPs, so it filters everything static.

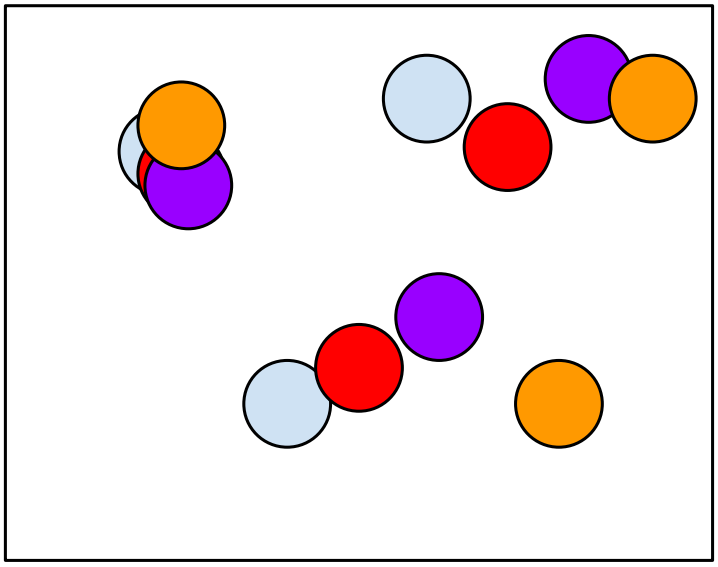


Figure 3.9 The picture presents Detections observed over 4 frames, with each frame having a different color. The top left batch shows how the static Detections could behave.

3.4.6 Joining of Track Fragments

Heuristics may sometimes prevent the correct associations to happen. This may cause a track to fall apart. We have developed a method to detect such a situation and merge those track fragments back together.

The method is based on collecting the assignment costs of all possible associations between the end of the first track and start of the second one, differences in the appearance descriptor and the position predictions.

3.4.7 Less Strict Measures Based On Camera Height

From our observations, we found that especially on the cameras high above the ground, tracks sometimes get lost just before reaching the entrance. Those problems arise most likely because the higher the camera is, the wider the area it captures is and the longer tracks are produced. The longer tracks are more problematic, because as the length of the track increases, a chance of an occlusion or a detector failure increases too.

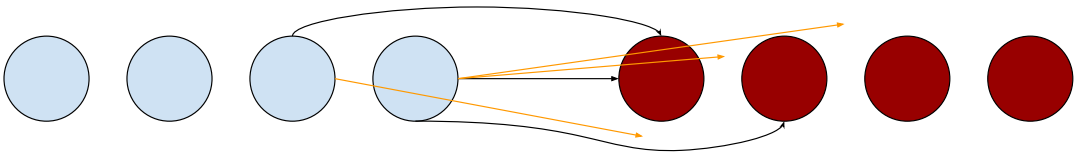


Figure 3.10 The figure captures a track which fell apart in half. To test the homogeneity of the fragments, we count assignment costs (black arrows), distances from predictions (prediction is represented by orange arrow) and the appearance differences (captured by black arrows). You can see that the connections are not dependent only on the edge between detections of the fragments.
Obtained values are then aggregated into a single value and based on it the fragments are either joined, or not.

We are therefore relaxing the necessity of ending in the entrance to ending near the entrance. The distance to the end is based on various tuned constants and the height of the camera.

3.4.8 Frame Gap Penalty

We have observed, that preference of the associations in strictly consecutive frames over the associations with frame gaps is reasonable. To capture the preference, we are adding a penalty to the association cost based on the size of the frame gap.

Except for disadvantaging such association in the data association part of the algorithm, we are also adding a threshold for maximal cost with the penalty counted.

3.4.9 Considering Older Entrance

Using this heuristic allows tracks to be finished outside of the sink entrance, if they passed such entrance previously. The heuristic is presented by figure 3.11.

3.4.10 Track Groups

This heuristic uses a feature developed by my colleague. The feature decides which pedestrians are walking together in a group. After getting this group, we can look at the unfinished track fragments which went along with the group. If fragments are clear and follow the group well, it is likely that they were not counted due to some occlusion in a crucial part.

Based on this knowledge, we can count an extra track to the entrance where the group went. See the 3.12 to get a picture about the situation.

3.4.11 Multiple Association Scores

To obtain a better association cost, we can use all the relevant means. As there are more RFCs trained based on frame gaps, we can use them all to get better cost for association with the new detection. We can then reason over those values and take perhaps the mean of them, the best of them, or any other aggregation. See figure 3.13.

The downside of this heuristic is its computational inefficiency, as the repeated RFC computations are quite expensive.

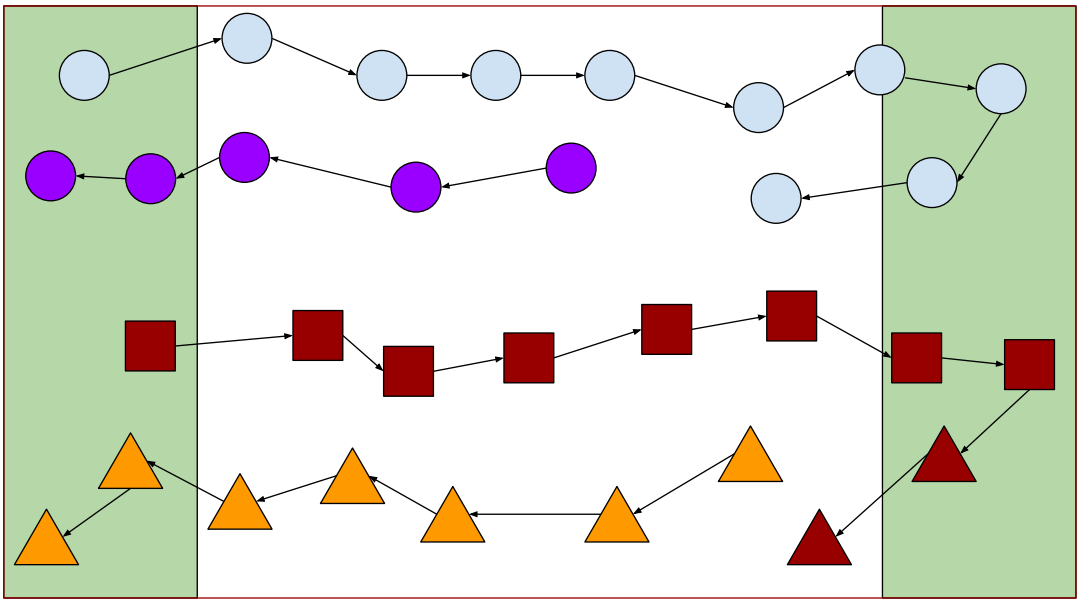


Figure 3.11 If we slightly adjust figure 3.8, we can present the benefits and the problems of considering older entrances. Again, the red border represents the area of interest and the entrances are denoted by green rectangles. The real identity of the pedestrians is presented by the shapes and the identity in context of the algorithm by colors.
In the case of the blue track, letting it finish as if it belonged to the right entrance would be wrong. On the other hand, if you look down on the red one, the heuristic would be beneficial.
This heuristic brought us more problems than advantages, so we do not use it anymore in our system. Still we believe that it is worth mentioning.

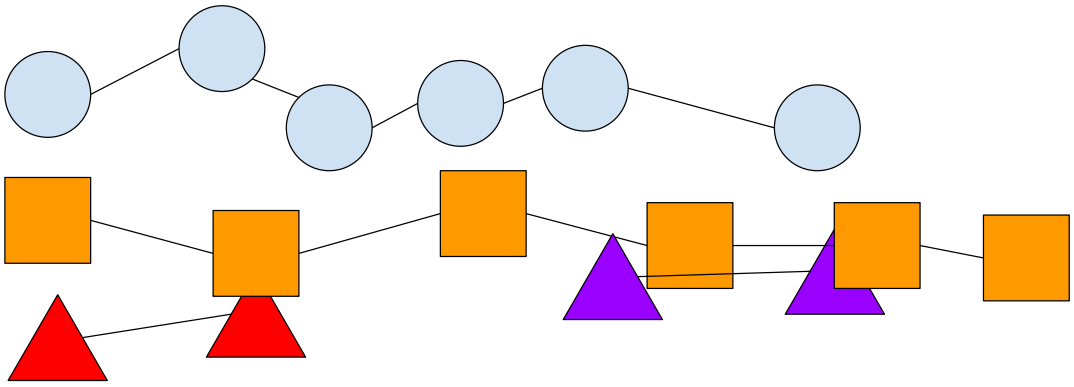


Figure 3.12 You see the circle and square tracks walking along. Sometimes there appears a triangle track. With the knowledge about the group, we can assume the fragments of the triangle track are a part of it and count one extra track.

3.4.12 Letting Finished Tracks Compete

Some heuristics may force tracks to end, while they still could be progressing. Detections which would be used for their expansion may be prone to mistakes, like starting a new track and even finishing it in combination with some other noises.

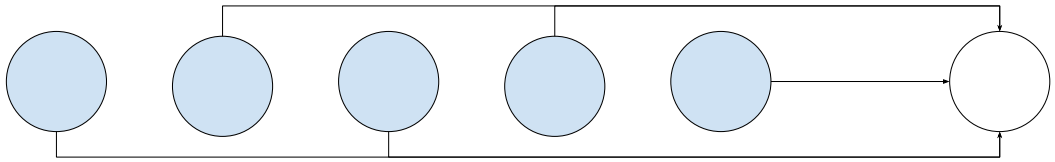


Figure 3.13 The arrows on the figure capture the counting of assignment costs for the association with the new Detections. Results are then aggregated to provide more precise values.

We arguably need to finish the tracks, but yet we do not want to let their residual detection cause any problems. To prevent such problems, all the finished tracks may compete for a few frames about the detections in the scene. But even if we find a good detection for some of such tracks, neither of them will be expanded anymore. If a detection to extend such a track is found, it cannot be used for any other track or functionality.

Chapter 4

Optimization

In this chapter we discuss how to optimize all the proposed methods and algorithms together to get the most accurate outcome. Most of the methods use some internal constants and we need to find the optimal values from them. The task is furthermore complicated, as the optimal values of those constants are tightly dependent on each other.

We will discuss three approaches for the task in the order they were developed and used.

We refer to the concrete combination of constant values used together in the algorithm as the *config*.

For all the proposed methods, we evaluate and store the tracking from the detections. The stored tracking results are then compared with annotations provided by our team of annotators to evaluate the accuracy.

4.1 Grid Search

The first attempted optimization method was the grid search. This method is based on the predefined values for each of the optimized constants. Say we have constants C_1 ..., C_n. For each of them we define some values V_1 ..., V_m (note that the number of values defined for constants can individually vary in practice).

Then the grid search takes each possible value V for every constant C and tests it with all the possible values of other constants. This way, every possible *config* is created and tested. See figure 4.1.

While the grid search worked well, there were two problems. The grid search can use only user defined values, so if the optimal value is somewhere in between them, the grid search would miss it.

The second and more significant problem is the time complexity. Let us say that A_1, ..., A_n refers to the amount of predefined values for the constant C_1, ..., C_n respectively. This means that the total number of configs to be tested is $\prod_{i=1}^{n} A_i$. The growth of the *configs* number is therefore exponential and becomes quickly unbearable for practical use.

4.2 Genetic Algorithm

To overcome both issues mentioned with the grid search, my supervisor Ing. Filip Naisser suggested to use the genetic algorithms. An overview of genetic algorithms usage for similar problems can be seen in the paper written by Annu Lambora et al.[21].

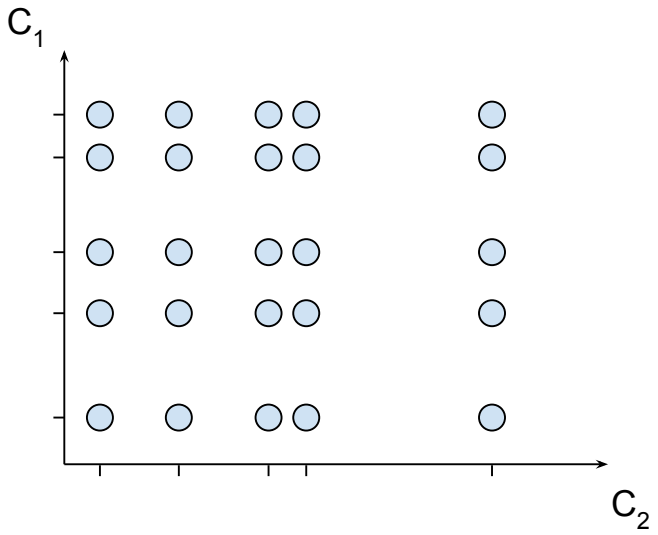


Figure 4.1 Assume we have only two constants to optimize values for. For both constants C_1 and C_2 we define 5 values, V_1 ..., V_5. The possible configs in this optimization are denoted by the dots in the graph.

Genetic algorithms proved to us to be effective during the optimization of the detection system. I took this implementation made by my supervisor and extended it to the optimization in the data association part of MOT.

This approach is based on randomly generating a number of *configs* with values from a reasonable range. Those configurations are tested on randomly picked subsets of our datasets. After each config has gained a score, we take the best one and test it on all our training datasets and store the result for further investigation.

As all the configs have been assigned some score, we can compare them and decide which ones are worth *selecting* for further processing and which are not. Some of the configs may *mutate* (slightly change the value of some constants) with a given probability, or *mate* with each other (mix values of the constants together).

We refer to the processing of testing, selecting, mutating and mating as one *generation* of the genetic algorithm. The output of the generation is its best config with the test result on all our testing datasets. The process is captured by figure 4.2.

Even though the genetic algorithm is less dependent on the possible value's predefinition, the computational complexity still remains challenging. It takes about 10 minutes to test one config on all our testing datasets, which is too much if we want to have a reasonable amount of configs and run enough generations. This is why we, as mentioned in this section earlier, firstly evaluate all the configs only on a subset of our training datasets. We talk about such subset selection in the next subsection.

4.2.1 Smart Datasets Subsetting

The recorded datasets vary in some important aspects. Their length ranges from 5 to 60 minutes and the number of annotated passages goes from 0 up to 1712. This imbalance needs to be compensated by some kind of smart subsetting. Consider a case with simple random subsetting, where we would choose a datasets' subset with a length of 10. Each dataset would be captured by a different camera. In this case, one dataset can contain for example 90 % of all the annotations

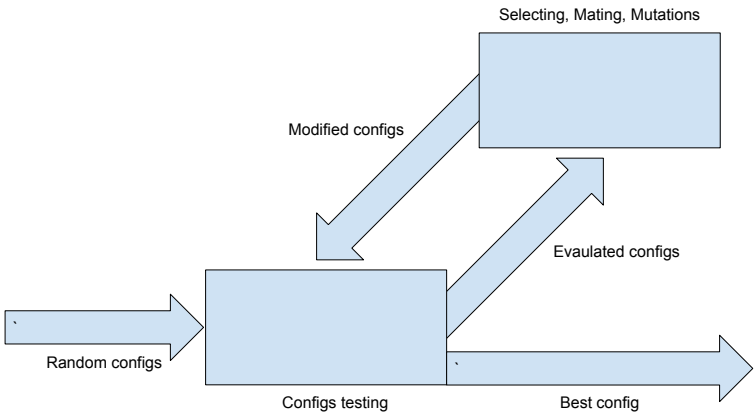


Figure 4.2 The figure captures the process of optimization using the genetic algorithm.

present in the subset. The generation tested on such subset could easily overfit for the scenery captured by the dominant camera.

One possible approach to tackling the issue is to have weighted mistakes. This approach brings some uncertainties. Let us say we have one dataset with 6 annotated passages and one with 600 annotated passages. How should there be now a mistake on the first dataset weighted regarding the second? Giving it 100 times greater significance, to even out the ratio of the annotations, could be problematic (mostly with overfitting for the smaller dataset). For it is unclear what the weight function should be, we are taking a different path.

First, we randomly choose a cameras subset of a given size. Then we pass all the datasets recorded by those cameras and find the one with the most annotations and save their number. Now we go camera by camera and randomly choose their datasets, until they have at least the same number of annotations as the most significant camera, or all the available datasets were chosen.

The proposed approach helps us to balance the datasets subsetting and does not corrupt the information value of the revealed mistakes in any way. Note that the algorithm could work just the same to compensate for the video length. We capture the algorithm by a pseudocode 4.1.

Code listing 4.1 Smart Subsetting Pseudocode

```
def subset_datasets(count_to_pick, all_cameras):
    random_cameras = get_random_cameras(count_to_pick, all_cameras)
    m = get_max_annotation_amount(random_cameras)

    datasets = []
    for camera in random_cameras:
        tmp_m = 0
        datasets_tmp = randomly_shuffle(camera.datasets)

        for dataset in datasets_tmp:
            if tmp_m >= m:
                break

            datasets.append(dataset)
            tmp_m += dataset.amount_of_annotations

    return datasets
```

4.3 Random Search

Random search is the simplest method we have proposed for the optimization. In general, the random search is only about generating a random config and then testing it. In our solution, we first test the config on testing datasets. If it performs well enough there, we test it on the training datasets too and store the result from this testing.

We compare the effectiveness of the genetic algorithm and the random search in the section 5.2.

Chapter 5
Experiments

This chapter covers the experiments performed to evaluate the precision of the proposed methods. It is dived into 3 sections - Heuristics, Optimization and Association Method.

In the Heuristics section, we discuss the impact of the implemented heuristic functions. Since many of the heuristics depend on each other and with respect to the great amount of them, we are not testing them separately, but rather batch them into some logical units.

In the second section we discuss the approach used for the optimization.

Lastly, the association method section compares the effectiveness of both proposed assignment methods. Namely we compare there usage of the greedy assignment and the usage of the Hungarian Algorithm.

We observe four error types. Number of False Positives (FPs), number of False Negatives (FNs), counting error and metric error. The counting error is given by an absolute difference of counted passages and annotated passages for each entrance.

The metric error works with FPs and FNs and is given by a figure 5.1.

$$FPs + FNs + \frac{(FPs + FNs) * |FPs - FNs|}{GTs}$$

Figure 5.1 The metric error is counted from the FPs and FNs. It a sum of FPs and FNs, to which we add the same sum multiplied by the absolute difference of FPs and FNs and divide it by the count of annotations (GTs).

5.1 Heuristics

To capture the contribution of the heuristic functions, we need to test the algorithm's accuracy with and without them.

First, we have batched the heuristics we want to test into 4 distinct groups G_1, ..., G_4. Heuristics are gradually placed into groups by their complexities, so G_1 contains the simplest heuristics, while G_4 contains the most complex ones. If some heuristics tightly depend on each other, they are always placed in the same group.

We start the experiment with a limited version of the algorithm, which can use heuristics only from the G_1 group. Such a truncated version has given some time to find the best configuration using our genetic algorithm for optimization. The goal of the optimization is to minimize the metric error.

After the time for optimization, we add the heuristics from the next group into the algorithm. The improved algorithm then also has some time to search for the optimal configuration, while its starting configurations are based on the values found in the previous search. The new starting configurations' values are randomly sampled around the previous results with the help of the normal distribution implemented by [22].

This way we gradually incorporate all the heuristics and observe how the accuracy changes. See figure 5.2 capturing the results during the search for the best accuracy. This figure captures the optimizations separately as the groups were added back. The figure 5.3 shows all the progression together and you can see there that the usage of our heuristic functions has a positive impact.

5.1.1 Heuristic Functions Grouping

First group contained only the very essential functionalities - it sets a few thresholds only: maximal distance (described in section 3.4.2.3) and the maximal frame difference for acceptable association, minimal association probability, the maximal distance from the prediction and the minimal detections' confidence.

Second group comes with a usage of some natural tracking properties. We are limiting the tracks based on their scores (section 3.4.2), deciding at what track's length to setup the Kalman Filter [20] and use it differently based on frame differences (Kalman Filter was discussed in section 3.4.4). We are also adding penalties for associations skipping frames (see 3.4.8 section). Arguably the most important heuristic added into this group is the differentiating based on camera heights (pursued in section 3.4.7).

Third group takes advantage of more complex functionalities. We are working with tracks which start out of the entrance area (as described in section 3.4.3), trying to use older entrances (section 3.4.9) and limiting the tracks' returning (one of heuristics described in 3.4.2.3). We also try to use the multiple association scores (section 3.4.11) and we test the ByteTrack [9] heuristic in this group (ByteTrack for our solution describes section 3.4.1.2).

The last one, **Fourth group**, uses the most complex heuristic. We add here the FP filter (section 3.4.5), group usage and fragment joining (respectively sections 3.4.10 and 3.4.6). The group also takes advantage of two round tracking (section 3.4.1.1).

5.2 Optimization Method

In this section we compare the efficiency of genetic algorithms for optimization to the efficiency of a simple random search, both described in Chapter 4. If you look at the third and fourth group in figure 5.2, you can see there that the criterion metric quite struggles to improve and the convergence oscillates. Furthermore, we are getting good results already in the first generations.

Because of these observations, we have proposed a random search method to have a baseline for comparison with the genetic algorithm usage.

To compare it with a genetic algorithm based optimization, we run the random search as described in section 4.2. The random configurations are generated the same way as for the third group in a previous section and the random search was provided approximately the same amount of time.

We have decided to use the third group, because the solution with the fourth group is computationally expensive and the number of values included in this group starts making things confusing. The solution used by the iC Systems.ai, s.r.o in practice uses only heuristics up to the third group too.

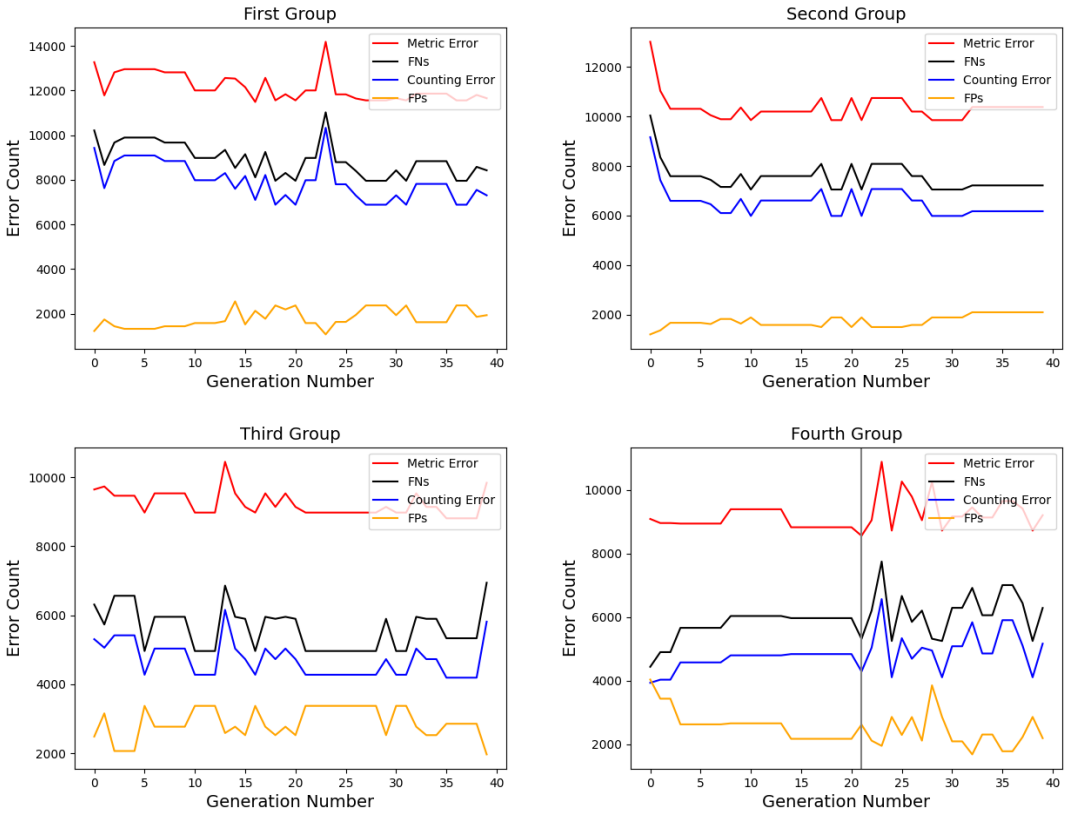


Figure 5.2 The figures capture progressions of the search for optimal configuration with limited usage of the heuristic functions separately. The search was done with a genetic algorithm and each group got 40 generations with a goal to minimize the metric error given by 5.1.
Only with the fourth group, we had to modify the experiment during its run. The time demands were high and the improvements were slow, so we decrease the number of training datasets and added more randomness. The moment of this stepup is denoted by the gray vertical line.
The entire optimization process took approximately 146 hours on our server with 32 core CPU.

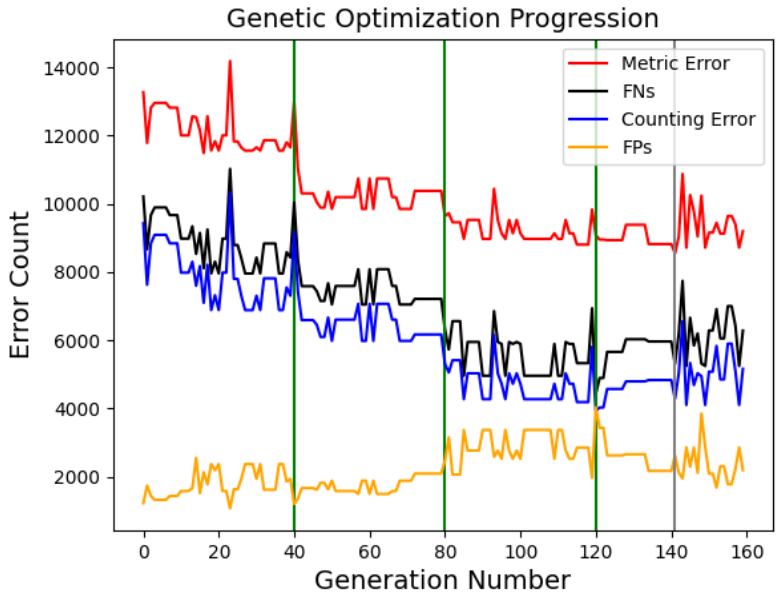


Figure 5.3 This figure captures the progression of optimization for all the groups together. The green vertical line denotes a moment, when the functions from the next group were added. The gray vertical line shows a moment, where we had to modify the fourth experiment slightly to make the computation faster and try to help the progression by bringing more randomness.
You can see the positive impact of heuristic functions, as the metric error 5.1 is gradually decreasing.

5.2.1 Optimization Using Genetic Algorithm

The optimization using this approach succeeded in finding a configuration, which yielded 4189 counting errors and a metric error of approximately 8806.

The genetic algorithm ran for 40 generations and found this best configuration in the 36th generation. Therefore we can assume that finding it took about $\frac{9}{10}$ of the 32 hours that the optimization ran in total, which means approximately 28,8 hours.

5.2.2 Optimization Using Random Search

The random search was used as described in the section 4.3. To have a relevant comparison with the genetic algorithm based approaches, we ran it approximately for the same amount of time - 32 hours. The best configuration found by this random search has 4745 counting errors and a metric error of approximately 8920.

The best configuration was found in the 241st attempt. This attempt was preceded by 21 attempts, which passed the test and were evaluated on all the training datasets. Such a process takes about 12 minutes. The rest 219 attempts failed the test, which takes about 1,5 minutes. Therefore we can assume that the configuration was found in about 7,5 hours.

5.2.3 Comparison

The usage of the genetic algorithm for optimization brought better results. Even though finding the best result took longer using the genetic algorithm, one configuration comparable to the best from the random search was found in the 6th generation. This configuration made 4274 mistakes

in counting and about 8973 mistakes on the metric. Since computing those 6 generations took about 2,4 hours, both methods are at least similar on the time demand.

Furthermore, the fact that the genetic optimization found the best results in the late phase of its run, brings a potential for further improvements. If we had to let the experiment run for a few more generations, it could have still improved.

Another interesting thing to see is that the genetic algorithm optimization performed much better on the counting error, which was not the direct subject of the optimization. See table 5.1. This is a nice thing to see for the genetic algorithm optimization. Arguably it happens because the generations were tested only on the subsets of datasets. This testing prevented optimization to overfit and resulted in the better counting error.

Table 5.1 The table captures the results of 4 different best configurations found by each optimization method.

Method	Rank	Metric Error	Counting Error
Genetic	1	8806	4189
Genetic	2	8973	4274
Genetic	3	9137	4726
Random	1	8920	4745
Random	2	8941	4840
Random	3	9102	5158

5.3 Association Method

In the section 3.3, we have discussed two association methods. This section is devoted to their comparison.

In the previous section, we ran the random search with use of the Hungarian Algorithm [18] and the heuristics from the third group. To compare the greedy approach with the usage of the Hungarian Algorithm, we ran the algorithm with the greedy association in the very same conditions.

With the help of the same seed for generating pseudo-random numbers, it is possible to test both association methods with the exact same configurations.

We ran the random search, generating same pseudo-random configurations as in the previous sections, and tested their accuracy with use of the greedy association approach. The results for the configurations which passed the testing phase can be seen in figure 5.4.

Using the Hungarian Algorithm proved to be better for almost all of the configurations, including the best one.

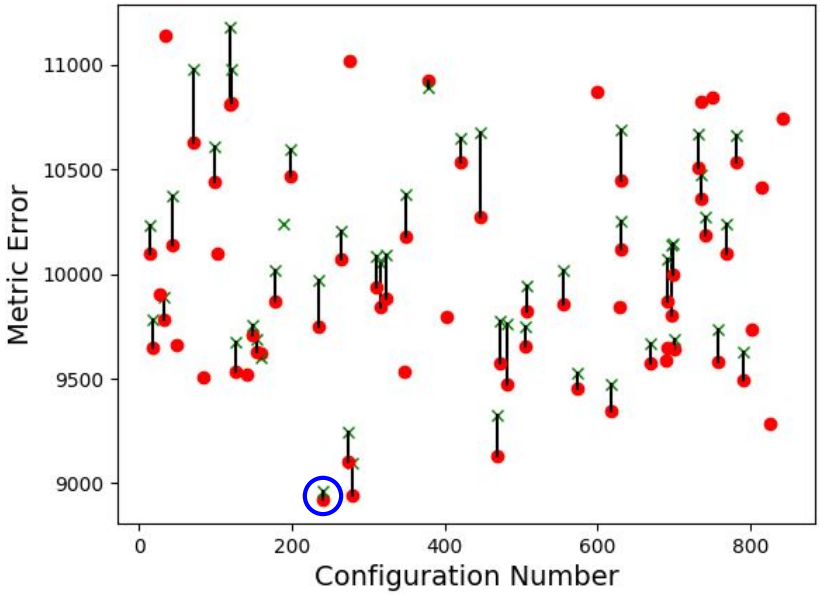


Figure 5.4 In the figure you can see the comparison of the usage of the Hungarian Algorithm and the greedy approach for the data association. For both of them we ran the random search, generating the same pseudo-random configurations.
In case the configuration passed the test with both approaches, the results are linked.
The results of using the Hungarian Algorithm are denoted by the red circles, the greedy approach by the green crosses. The best config found by both approaches is highlighted in a blue circle.

Chapter 6
Conclusion

The thesis set out to improve the methods for pedestrian tracking in top-view video sequences used by the iC Systems.ai, s.r.o. To achieve this goal, we needed to propose an assignment cost function, use a relevant data association method and incorporate some heuristic functions. For the first one mentioned, an assignment cost function, we have also further investigated the neural network-based frameworks. This chapter summarizes how we met the set goals.

This thesis proposed methods to efficiently solve the data association part of the Multiple Object Tracking (MOT) and successfully evaluated them.

The literature review directed the author's steps regarding the recent development and has opened some opportunities for the future development. The author considered the usage of Neural Networks [1] for MOT in iC systems.ai, s.r.o. in the Chapter 2.

All the methods for pedestrian tracking in top-view video sequences are described in Chapter 3. The author has designed numerous features for the assignment cost function and used them together as an input of the Random Forest Classifier (RFC) [16]. The success rate of the cost assignment function is captured by 3.1 and the feature importance by 3.3.

The author contributed also by integrating the Hungarian algorithm [18] as a superior data association method.

A significant part of the author's contribution are the heuristic functions in Section 5.1. These functions offered a great improvement thanks to the author's domain knowledge. The author proved his ability to adapt, as the detection system was changing during the development phase and the following data association was required to solve different problems (eg. one detector version was producing a high number of static False Positives, another version had problems rather with False Negatives, etc.).

In the Chapter 4 the author proposed different methods to find the best configuration for the implemented algorithm. Their use arguably brought the algorithm's accuracy close to the limits.

Testing of all the relevant methods is captured by the Chapter 5. There, the author illustrates the contribution of the proposed and improved methods.

The time efficiency of the final algorithm is sufficient to run online with respect to provided computation resources.

Please note that to respect the company policy about the code publishing, the implementation is not publicated with the thesis, but is available at request after the NDA signature.

Bibliography

1. BISHOP, Christopher M; NASRABADI, Nasser M. Pattern recognition and machine learning. In: Springer, 2006, vol. 4, pp. 225–290. No. 4. ISBN 9780387310732.
2. HULMÁK, Erik. *Pedestrian Re-Identification in a Camera System* [online]. 2021 [visited on 2022-04-06]. Available from: `https://dspace.cvut.cz/bitstream/handle/10467/92890/F8-BP-2021-Hulmak-Erik-thesis.pdf?sequence=-1&isAllowed=y`.
3. NAISER, Filip. *Detector Implementation for iC Systems.ai, s.r.o.* 2022.
4. *iC Systems AI, s. r. o. website* [online] [visited on 2022-04-06]. Available from: `https://www.icsystems.ai/`.
5. LUO, Wenhan; XING, Junliang; MILAN, Anton; ZHANG, Xiaoqin; LIU, Wei; KIM, Tae-Kyun. Multiple object tracking: A literature review. *Artificial Intelligence* [online]. 2021, vol. 293, p. 103448 [visited on 2022-04-12].
6. LINSLEY, Drew; MALIK, Girik; KIM, Junkyung; GOVINDARAJAN, Lakshmi Narasimhan; MINGOLLA, Ennio; SERRE, Thomas. Tracking Without Re-recognition in Humans and Machines. *Advances in Neural Information Processing Systems* [online]. 2021, vol. 34 [visited on 2022-04-17].
7. *MOT Challenge* [online] [visited on 2022-04-12]. Available from: `https://motchallenge.net/`.
8. HEINDL, Christoph. *py-motmetrics* [online] [visited on 2022-04-17]. Available from: `https://github.com/cheind/py-motmetrics`.
9. ZHANG, Yifu; SUN, Peize; JIANG, Yi; YU, Dongdong; YUAN, Zehuan; LUO, Ping; LIU, Wenyu; WANG, Xinggang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *arXiv preprint arXiv:2110.06864* [online]. 2021 [visited on 2022-03-29].
10. BRASÓ, Guillem; LEAL-TAIXÉ, Laura. Learning a neural solver for multiple object tracking. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* [online]. 2020, pp. 6247–6257 [visited on 2022-03-29]. Available from: `https://arxiv.org/pdf/1912.07515.pdf`.
11. HE, Jiawei; HUANG, Zehao; WANG, Naiyan; ZHANG, Zhaoxiang. Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* [online]. 2021, pp. 5299–5309 [visited on 2022-03-29]. Available from: `https://openaccess.thecvf.com/content/CVPR2021/papers/He_Learnable_Graph_Matching_Incorporating_Graph_Partitioning_With_Deep_Feature_Learning_CVPR_2021_paper.pdf`.

12. WANG, Yongxin; KITANI, Kris; WENG, Xinshuo. Joint object detection and multi-object tracking with graph neural networks. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* [online]. 2021, pp. 13708–13715 [visited on 2022-03-29]. Available from: https://arxiv.org/pdf/2006.13164.pdf.

13. WU, Jialian; CAO, Jiale; SONG, Liangchen; WANG, Yu; YANG, Ming; YUAN, Junsong. Track to detect and segment: An online multi-object tracker. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* [online]. 2021, pp. 12352–12361 [visited on 2022-03-29]. Available from: https://openaccess.thecvf.com/content/CVPR2021/papers/Wu_Track_To_Detect_and_Segment_An_Online_Multi-Object_Tracker_CVPR_2021_paper.pdf.

14. MEINHARDT, Tim; KIRILLOV, Alexander; LEAL-TAIXE, Laura; FEICHTENHOFER, Christoph. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702* [online]. 2021 [visited on 2022-04-05].

15. NAISER, Filip. *Disccusion on the Topic of Joint Object Detection and Association for iC Systems.ai, s.r.o.* 2022.

16. BREIMAN, Leo. Random forests. *Machine learning* [online]. 2001, vol. 45, no. 1, pp. 5–32 [visited on 2022-04-08]. Available from: https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf.

17. *sklearn Random Forest Classifier* [online] [visited on 2022-04-08]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.

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

19. NAISER, Filip. *Greedy Detection Association Implementation for iC Systems.ai, s.r.o.* 2020.

20. KALMAN, Rudolph Emil. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering* [online]. 1960, vol. 82, no. Series D, pp. 35–45 [visited on 2022-04-09]. Available from: http://160.78.24.2/Public/Kalman/Kalman1960.pdf.

21. LAMBORA, Annu; GUPTA, Kunal; CHOPRA, Kriti. Genetic algorithm-A literature review. In: *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)* [online]. 2019, pp. 380–384 [visited on 2022-04-17].

22. *random.gauss* [online] [visited on 2022-04-17]. Available from: https://docs.python.org/3/library/random.html#random.gauss.