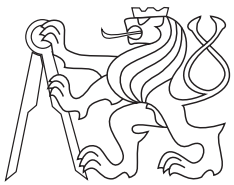
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

MASTER THESIS

ISSN 1213-2365

Discovery, tracking and redetection of floorball players from multiple cameras

Bc. Miroslav Purkrábek

purkrmir@fel.cvut.cz

May 20, 2022

Thesis Advisor: prof. Ing. Jiří Matas, Ph.D.

Master Thesis of CMP, Czech Technical University in Prague,

Published by

Center for Machine Perception, Department of Cybernetics
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Discovery, tracking and redetection of floorball players from multiple cameras

Bc. Miroslav Purkrábek

May 20, 2022

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Purkrábek Miroslav** Personal ID number: **465809**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence**

II. Master’s thesis details

Master’s thesis title in English:

Discovery, tracking and redection of floorball players from multiple cameras

Master’s thesis title in Czech:

Nalezení, sledování a redetekce hráčů florbalu z více kamer

Guidelines:

1. The supervisor has access to recordings of floorball acquired by multiple not fully synchronized cameras viewing the pitch from different angles. The recordings have been synchronized to within one frame precision, corrected for radial distortion and the ground plane has been registered.
2. The bounding boxes of players have been detected by the MMDet system. [1]
3. Implement a method for discovering new identities and tracking and redetection of players, possibly with the help of available open source components.
4. Using the results of 3., and information about camera calibration, propose a method for combining results in multiple cameras and create automatically or semi-automatically partial ground truth (reliable poses of some of the players).
5. Evaluate both the single-camera and multi-camera method on data with ground truth on a subset of the footage.

Bibliography / sources:

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark, 2019
[2] Szeliski, R. (2022), Computer Vision - Algorithms and Applications. , Springer, 2nd edition.
[3] Ponce J., Forsyth, D. (2011), Computer Vision - a Modern Approach

Name and workplace of master’s thesis supervisor:

prof. Ing. Jiří Matas, Ph.D. Visual Recognition Group, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **08.02.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

prof. Ing. Jiří Matas, Ph.D. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

Author statement for graduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date: . . . . . Signature: . . . . . .

I would like to express my sincere gratitude to my supervisor for his patient guidance, inspirational brainstormings, and valuable critiques of this thesis. Likewise, I greatly appreciate the help in writing the thesis provided by my sister. I also wish to thank the TJ Sokol Královské Vinohrady for their cooperation in data acquisition. And last but not least, this project would not have been possible without my parents and girlfriend and their support and encouragement throughout my study.

Abstract

This thesis proposes a new system for unsupervised person tracking by detection. The system focuses on tracking players in sports video, specifically floorball, using multiple cameras with different viewing angles. The new hard-negative mining technique leveraging time constraints enables us to train the identification network without labelled data. The proposed method generates tracklets with a low level of ID switches, and it is appropriate for generating labelled data for supervised training. Samples from the newly created dataset, which we used for evaluation, are attached to the thesis along with the code.

Keywords

automated sports analysis, computer vision, hard-negative mining, multi-view person tracking, person identification, sports videos, sports videos dataset, top-view camera, tracklets clustering, unsupervised clustering, unsupervised identification, unsupervised person tracking, videos synchronization

Abstrakt

Tato práce představuje nový systém pro sledování osob bez dozoru. System se zaměřuje na sledování hráčů ve sportovních videích, zejména florbalu, za použití více kamer z různých úhlů. Nový způsob výběru negativní třídy za použití časových omezení nám umožnil natrénovat identifickační síť bez anotovaných dat. Navržená metoda generuje tracklety s malým počtem výměn identity a je vhodná pro generování anotací. K práci přikládáme kromě kódu i vzorky datasesetu, který jsme pro tuto práci vytvořili.

Klíčová slova

automatická sportovní analýza, dataset sportovních videí, kamera z vrchního pohledu, shlukování bez dozoru, shlukování trackletů, sledování osob bez dozoru, sledování osob z více úhlů, sportovní videa, strojové vidění, synchronizace videí, výběr negativních vzorků

Contents

Abbreviations

CTL Centroids Triplet Loss. 31

DCNN deep convolutional neural network. 19

FN False negative. 36

FP False positive. 36, 40

fps framer per second. 12

GPS global positioning system. 9

ID identity. 7, 16, 40

IFF International Floorball Federation. 13

MOT Multiple object tracking. 7, 17, 19

MOT17 Multiple object tracking benchmark 2017. 19

MOT20 Multiple object tracking benchmark 2020. 19

MOTA multiple object tracking accuracy. 40

SKV TJ Sokol Královské Vinohrady. 8, 12

TN True negative. 36

TP True positive. 36

List of Figures

List of Tables

1. Introduction

The thesis introduces a new method for unsupervised tracking in sports videos from multiple cameras. Our system works specifically with four cameras, but it could use a different amount. We use the tracking-by-detection approach since the long-term tracking in sports videos requires re-detection and re-identification as players change regularly throughout the game. This thesis is a continuation of our previous work [30].

1.1. Problem description

Let us first clarify the main concepts of the thesis.

Object tracking is a computer vision task of assigning a unique identity (ID) to each set of detections representing a unique object. Multiple object tracking (MOT) is the task of tracking multiple objects in the video simultaneously. Unsupervised methods learn the structure of the problem without labelled data.

Our method is an unsupervised multiple object tracking algorithm. The algorithm runs in multiple cameras. Therefore, the method's inputs are synchronized videos, and the results are tracks of detected objects. The track is a set of detection (object locations) with the same ID. Inputs and outputs are further clarified in chapter 3, along with intermediate steps of the pipeline.

The proposed method tracks players in floorball videos. In the thesis, we refer to anyone on the playing field as a player. Unless explicitly emphasized, we do not differ between players, goalies and referees as they all behave similarly.

The main challenges of the task are re-detection and re-identification, tracking in crowded areas and memory consumption. When the player changes, he disappears from the video, and the algorithm has to recognize him once he returns. Crowded areas are challenging because of many occlusion hindering detection. And the memory consumption becomes a problem with the size of the video. The average floorball match takes around two hours.

1.2. Motivation

With the growing use of data in sports, clubs are looking for methods for automatic data collection while minimizing financial expenses. All sports matches are recorded on video, and using the computer vision approach for automated analysis brings only negligible costs.

Various sides use data in sports, and each is interested in different data. Players are interested in their statistics of shooting efficiency, rebounds or blocks. We can also collect statistics about the player's current health and fitness status, which is beneficial not only for the player but also for coaches and doctors. Then we have team statistics, technical analysis and tactical analysis of both the home team and the opponent. Detailed statistics about the game and players are also helpful in evaluating fouls by video referees. And last but not least, computer vision methods for automated video analysis are popular with fans for the visualizations and extended reality.

An ideal system would meet the following criteria:

- Easy to deploy. Sports clubs do not employ technicians, and coaches are sports-oriented experts. Therefore, the system should run automatically end-to-end for effortless deployment.
- An ideal system would run in real-time to provide results during the match. One step behind are methods evaluating a larger part of the match and providing results after each period.
- Able to track in the long-term. The system should be able to re-detect and re-identify a player after s long period (more than one hour) of absence.

We designed the method using floorball videos as we have access to the sports hall of the TJ Sokol Královské Vinohrady (SKV), playing the highest floorball league.

1.3. Thesis layout

The remaining paragraphs of this chapter are dedicated to the analysis of current solutions. The chapter 2 treats the problem of data. We first explain the shortcomings of public datasets and then proceed with the data we collected. The chapter 3 introduces the new method and the relevant theory. The chapter 4 depicts experiments and results we executed on our dataset. The chapter 5 briefly lists interesting implementation challenges and cites used libraries. We sum up the thesis in chapter 6 by repeating the most significant results and proposing directions for further research.

1.4. Current work

Here we analyze current solutions and state-of-the-art. We split the analysis into the more general one concerning sports and the specific one focusing on the computer vision approaches.

1.4.1. Situation in sports

Nowadays, sports teams collect a vast amount of various data about athletes. Individual sports focus more on statistics about physical and technical performance. Team sports like floorball also add tactical analysis. The data ranges from ball (or puck) speed to the study of the tactic of the whole team. Other big interests are fans and advertising. We can offer attractive visualizations or advertise products with extended reality with automatic game processing. All influential sports like basketball, hockey, soccer and others collect basic statistics like ice-time (or equivalent), shooting efficiency, number of rebounds and more. Players tracking is the underlying technology that would enable automatic evaluation. Many sports videos (floorball included) are also available for analysis, so research in this area promises considerable popularity.

The most prominent source of data is still human annotators. The biggest sports clubs have their teams of data analysts who collect data either with sensors or by manually processing videos. There are also automated solutions from private corporations like SecondSpectrum [52], Opta [50] and iSportAnalysis [47]. The highest leagues like NBA or Premier League employ these solutions to provide extensive data for fans and bookmakers.

Contrarily, there is no such solution available for minor sports like floorball. Each match in the Czech highest league needs at least four people to record statistics about

shots and changes. In the latest season, the Czech Floorball required recordings in digital format for the first time.

The most common approach to tracking is the usage of localization sensors. Outdoor sports take advantage of the global positioning system (GPS), with players wearing GPS receivers during the match. Lower leagues and minor outdoor sports employ sports trackers included in the smartwatches. They can provide approximate location and health data, for example, pulse. However, the precision of the GPS is limited to several meters [26], which is unsuitable for professional sports.

Furthermore, GPS trackers do not work indoors. The indoor trackers work similarly to the GPS but require local signal beacons. Teams can use off-the-shelf solutions like Kinexon [48] or open-source variants (for example, Navigine [49]).

The main disadvantage of tracking with sensors is the price. We need one sensor for each player, and the player is exposed to increased discomfort by wearing sensors on the body. Hardware tracking systems would be ideal for annotations generation for supervised training of the computer vision algorithm.

1.4.2. Computer vision approaches

Data

Let us first review datasets available for the training of computer vision algorithms. Even though our method is unsupervised, we want to illustrate the shortage of relevant data, which forced us to retort to the unsupervised method.

Most sports-related datasets for tracking focus on soccer [9], [7]. We believe that there are two reasons: money and the simplicity of tracking in soccer videos. Soccer is one of the biggest sports globally and is also a great business indirectly offering enough money for the research. The second reason is that tracking in soccer videos is easier than in other team sports. Players virtually do not change, so no re-detection is needed with the proper angle. The playing field is big, players are far apart, and the game is relatively slow. Lastly, we can use GPS trackers to generate at least noisy annotations. These features make soccer an ideal sport to start with tracking.

The image 1.1 shows the asymmetry between sports. It depicts the percentage of studies of the well-known sports. The image is taken from [25].

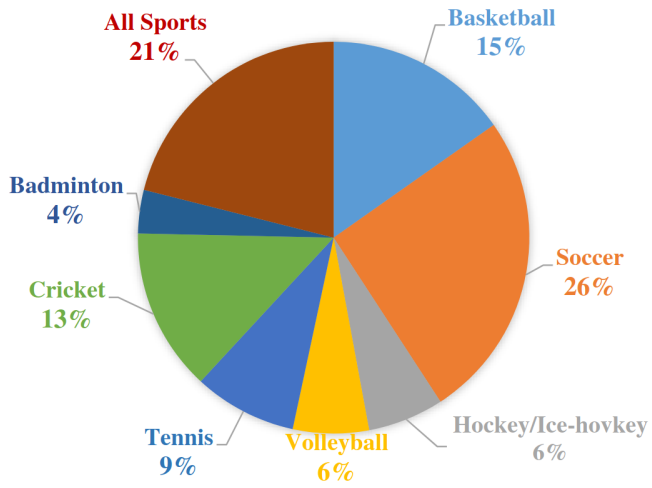


Figure 1.1. Percentage of studies in different sports. Image taken from [25].

Apart from sports videos, many datasets feature short sequences, such as MOT [21] or OTB50 [41]. Short sequences do not simulate real-life scenarios when tracking is needed for more than an hour. Datasets with long sequences focusing on long-term tracking like [24] are a relatively new area of research.

Lastly, there are datasets focusing on the person re-identification like Market-1501 [45] and DukeMTMC-reID [33]. We mention both throughout the thesis.

Sports analysis

Automated sports analysis is an open research area. The paper [25] reviews all current approaches in more detail. The direction of research corresponds to available datasets. There are articles concerning tracking in soccer [2], but also in basketball [11] and hockey [6]. Hockey is the most similar of these sports to floorball because of the game structure and shifts changes. But the hockey players look differently in principle. In hockey, players wear a lot of standardized equipment, including helmets, so there are no other means of identification than jersey numbers. In floorball, we believe the players are visually different, and we can identify them by their appearance.

There are also articles regarding the recognition of game situations in all these sports - basketball [23], hockey [34] and soccer [18]. Some papers like [43] employs multi-camera view as we do. The common problems in sports analysis are a shortage of data, jersey numbers and different game structures. We mentioned data shortage previously. Jersey numbers are the perfect feature for identification, but the number is often deformed and hard to read. In basketball and soccer, players wear jerseys with big clear numbers, while in hockey and floorball, numbers are smaller and hidden in a tangle of advertisements. Lastly, game structure differs significantly between sports. In soccer and basketball, player changes are limited and happen only when the game is interrupted. In hockey and floorball, players can change without limitations throughout the game.

State-of-the-art

In the last part, we mention the most used algorithms and state-of-the-art.

For detection, the algorithm with the best result is the DINO [42]. The previously mentioned papers often use the Faster R-CNN [32] or YOLO [31]. This thesis uses the newest variant of YOLO, YOLOX [12]. We are also aware of articles performing multi-view detection [40], [16] and tracking [36].

ByteTrack [44] is the state-of-the-art tracker on the MOT dataset. It tracks by detection, and we used it in the first stage of our pipeline. Another successful tracker is OC-SORT [5], state-of-the-art on the KITTI [13] dataset.

The identification networks usually employ ResNet [15] as a backbone. It also applies to the state-of-the-art Centroids [39] used in this thesis.

Automatic calibration

The last related research area is automatic camera calibration. It includes not only geometric calibration but also homography estimation. Several articles propose detecting the playing field (for example, [27]) to automate the tedious calibration process. We can estimate the radial distortion from straight lines with the detected pitch as in the [1] or directly compute the homography with radial distortion as in [20]. The website [46] gathers selected articles in this area.

2. Data

This chapter presents data used for training, validation, testing and other experiments described in chapter 4. First, we lay out the current situation regarding sports analysis and sports videos. We briefly touch on available datasets and the dataset used in our previous work. Then we continue by introducing a new data collection system designed for this thesis. The chapter closes with a section presenting collected videos.

2.1. Current situation

Automated sports analysis suffers from a shortage of data. Teams broadcast matches online in most sports out of obligation. Although some videos are available for free, these videos are suitable for fans and television. Broadcast videos feature a lot of zooms, cuts, camera movement and occlusions. Sports clubs capture more technical videos for tactical analysis or analysis of goalies. The technical videos are not available to people outside the club, and the viewing angle is not suitable for automatic analysis. As mentioned in chapter 1.4.2, private companies gather a vast amount of data in similar team sports, but they do not publicly share the data for further research.

Publicly available datasets for detection or tracking like PASCAL [8], COCO [22], Market-1501 or DukeMTMC-reID contain a small amount of sport-specific situations. We do not know about any publicly available dataset, let alone annotated.

The main distinctions between publicly available datasets and sports videos are jerseys, unified equipment, and characteristic constraints on players' movement. Players in unified jerseys pose a challenge for successful tracking as all players of the same team are very similar. Players use the same equipment like sticks and shoes given by a sponsor, which increases the similarity in the same team. On the other hand, opponent teams and referees wear distinct jerseys, so it is easy to distinguish players from two teams. Moreover, jersey numbers simplify identification. The last characterization of sports videos is players' movement. Players do not move randomly, and rules constrain the area of action. There is a limited number of players on the pitch, which again simplifies tracking.

Our previous work focused on the same topic using data provided by a private company. The data came from the World Floorball Championship in 2018, and we created and labelled a modest WFC-18 dataset. However, the dataset had several shortcomings.

First and foremost, videos were from only two different angles - top-view and side-view. The side-view video was too far, the players were small, and there were many occlusions. Second, the top-view video was split into two independent videos, capturing half of the pitch. Mapping players between halves and detecting players around the half-line proved difficult, and tracklets were disconnected when the player crossed the half-line. Third, there was no straightforward way to synchronize videos. The World Floorball Championship applies strict rules for used technologies, and videos were not captured for tracking. Lastly, the WFC-18 dataset had only small variability in teams and jerseys colours. It showed that most of the countries wear similar jersey colours.

We decided to take advantage of our access to the team playing in the Czech highest league (Livesport Superliga) and created a data collection system focused on tracking.

The next chapter introduces the system, following section describes created dataset.

2.2. Data collection system

With access to the sports hall of SKV in Prague, we designed a system of 5 cameras capturing the pitch from 5 different angles. Four cameras are along the edges of the pitch in heights between 4 to 10 meters. The fifth camera is on the ceiling directly above the throw-in area in the centre of the pitch. Throughout the thesis, we refer to all cameras by their respective cardinal directions (for example, W or WEST) in capitals. The ceiling-mounted camera is T or TOP.

2.2.1. Technical parameters

As is evident from the image, all cameras see the whole playing field, which was one of the crucial requirements for the system. The goal was to maximize the playing field area captured by all cameras while keeping the cost reasonably low. Using five different angles offers variability in views minimizing the number of false negatives caused by occlusions. The TOP camera in the height of 14 meters is exceptionally rare (most sports halls do not have such high ceilings) and could help track in crowded areas. However, as we explain in the chapter 3.5, we did not use the TOP camera in the final pipeline. The dataset presented in chapter 2.3 also contains videos from the TOP camera so that we could use them after the advancement of top-view detectors.

We will see in chapter 3.4 that using fisheye cameras complicates projection between cameras, but we prefered the full playing field coverage to non-fisheye cameras. Three cameras could see the whole pitch because of the fisheye lenses.

The table 2.1 sums up the technical parameters of all cameras and camera models for the system's reproducibility. We chose security cameras for their low cost and high durability. In the table, fps stands for frames per second.

Camera name	Resolution	Framerate	Fisheye lens	Hikvision model
EAST	3840 x 2160	20 fps	No	DS-2CD2685FWD-IZS
NORTH	2688 x 1520	25 fps	Yes	DS-2CD2T45G0P-I
SOUTH	2688 x 1520	25 fps	Yes	DS-2CD2T45G0P-I
TOP	2688 x 1520	25 fps	Yes	DS-2CD2T45G0P-I
WEST	3840 x 2160	20 fps	No	DS-2CD2685FWD-IZS

Table 2.1. Technical parameters of used cameras

As you can notice, cameras do not capture videos with the same framerate. Although the producer offers variable framerate settings, we prefer editing videos with the FFMPEG library [10] for its speed and precision.

2.2.2. Synchronization

To connect videos from different cameras in the pipeline, we need to synchronize videos with frame-level precision. In an ideal environment, all cameras capture the frame instantly and at the exact moment. Time synchronization means aligning videos such that frames from the exact moment are processed simultaneously. Unfortunately, cameras are not precise due to their price and technical limitations. The cameras do not capture video with a constant framerate, and when there is a quick action in the video, the cameras drop frames to preserve real-time capturing. The frame dropping results

in each camera having a different number of frames for the same time interval, posing a challenge in synchronization. Using the FFMPEG library solves problems with different framerates and the frame dropping. Frames are extracted based on the required constant framerate, and when a frame is missing, the nearest one is duplicated.

The resulting frames are not necessary from the exact moment, but the time difference between the two frames is small enough for the pipeline to work correctly. The size of the difference depends on the amount of dropped frames, and we observed that it is not greater than 0.2 seconds when using 20 fps. The difference could grow with the length of the video. Our most extensive experiment in chapter 4.4.5 worked with 30-minutes long video, and the difference was just 0.2 seconds. To achieve a more precise synchronization, we would have to synchronize videos regularly and not only at the beginning of the sequence.

So far, we have discussed the problem with variable framerate. Now we will focus on the synchronization itself. The frames are equipped with timestamps written directly to the image. We failed to synchronize videos using the timestamps as it is unclear how often the cameras update their internal clocks, and each camera can have a slightly different time.

We resorted to manual synchronization using light flashes visible in all cameras. We have to find the first frame where the light burst appears and align videos by them. Manual synchronization is by far the most laborious manual work in the pipeline and should be automated in the future work.

2.3. Videos

With the presented system, we collected videos from three different leagues. Unless stated otherwise, all matches are played three times 20 minutes according to the International Floorball Federation (IFF) rules for the highest leagues. The dataset offers variance in age, gender and lighting conditions. We collected 24 matches, resulting in more than 280 hours of videos.

In the table 2.2, you can see all recorded matches. Data comes from the men's highest league (Livesport Superliga, marked as MA in the table), women's second-highest league (marked as WA) and the highest league for boys under 19 (marked as U19). All matches consist of 5 videos from cameras described in table 2.1. Some Livesport Superliga matches also contain the sixth video - broadcast video for fans.

The videos are representative of one season in one hall. We can notice that the home team (SKV) is always yellow or black as those are their jersey colours. Correspondingly, the guest team usually wears blue or white, and the referees are red. The low colour variability is characteristic of each hall, and we would need to collect videos from multiple clubs to avoid it. The sports hall is multi-purpose, and there are many lines for different sports on the surface. The lines could help in geometric synchronization in chapter 3.4 but are not standard for the official surface.

This thesis used sequences from two matches - the U19 match in line 22 and the WA match in line 14.

2.4. Annotations

This short chapter focuses on the problem of annotations.

The complete annotation means labelling all players and their identities correctly. We could evaluate the detector, tracklets and tracks creation independently with the

#	Category	Home colour	Guest color	Referee colour	Notes
1	MA	Yellow	Red	Blue	TV Stream
2		Yellow	White	Red	TV Stream
3		Yellow	White	Red	TV Stream
4		Yellow	Blue	Red	TV Stream
5		Yellow	White	Red	TV Stream
6		Black	Green	Red	
7		Yellow	White	Blue	
8		Yellow	Red	Black	
9		White	Orange	Yellow	unofficial
10	WA	Black	White-green	Blue	
11		Black	Red	Yellow	
12		Yellow	Red	Blue	
13		Black	White	Red	
14		Black	Yellow	Blue	
15	U19	Yellow	White	Red	
16		Yellow	Black	Red	Sunshine
17		Yellow	Blue	Red	
18		Yellow	White	Blue	
19		Yellow	Blue	Red	
20		Yellow	White-green	Red	
21		Yellow	White	Blue	
22		Yellow	Blue	Red	
23		Yellow	Blue	Red	
24		Yellow	White	Blue	

Table 2.2. Created dataset of 24 matches

full annotation. Manual annotation for tracking is extremely expensive because the annotation should be independent of all tracking steps and sports videos are very long. Independence for tracking steps, we understand as the ability to process all steps independently. For example, we could train and evaluate the tracker even with the wrong detector.

Visual tracking is an easy task (for humans) in short sequences when the player does not leave the video. It showed that re-identification in one camera is challenging even for a human. We were able to identify players only with the personal knowledge of players and a deep understanding of the game. The labelling process would require identification from all cameras so that the annotator can choose the best viewing angle. But such a process is complicated and needs dedicated tools. We decided to generate annotations automatically.

As indicated in the chapter 1.2, we started the project as supervised tracking. The expected workflow was to obtain annotations semi-automatically and train the tracker (detector and identification network) on the data. But because of the mentioned reasons, we generated annotations fully automatically. The chapter 3 describes the process in detail. The automatic labels generation proved so successful that we dedicated ourselves to it in this work and left the supervised tracking for future research. We do not exclude the possibility of using the human-in-the-loop approach for correcting automatically created labels.

2.4.1. Testing sequences

But even for unsupervised tracking, we needed to annotate some data for evaluation. We describe annotated sequences here.

U19-SKV-MIL-08-01-2022-1st-period

We created a sequence of one period of the match to test the pipeline extensively. The video is 30 minutes long with a framerate of 15 fps. We decreased the framerate to lower the computational overhead. The sequence consists of 27 000 frames, approximately 1 300 000 single-camera detections and more than 10 000 tracklets. Even with a lower framerate than the original videos, we had to deal with memory problems when processing the sequence. Implementation details are mentioned in chapter 5. There are 34 players throughout the sequence - 15 for each team, two goalies and two referees. Throughout the thesis, the sequence is referred to as **1period**.

Completely annotating the sequence of such length would be expensive, so we opted for statistical evaluation. We fixed multi-camera detections and tracklets and used the sequence only for evaluating tracks creation which is the only step influenced by the length of the video. All other parts were evaluated using the 1 min sequence.

But even manual tracklets connection is too lengthy. We sampled the longest tracklets for each player. They make approximately 1% of all tracklets but on average cover 8% of the sequence. Therefore, when the long tracklets are correctly identified, the video looks good, and the resulting statistics would be mostly correct. However, as we will see in the chapter 4.4, the small tracklets are where the most errors occur, and the errors could propagate further in the video. Annotating long tracklets is, therefore biased evaluation, and we chose it for its cheapness.

One of the teams in the sequence has identical twins, which even the coach has problems distinguishing.

U19-SKV-MIL-08-01-2022-1st-period-1min

This sequence is the main testing sequence during development. It is the first minute of the previous sequence with the original framerate of 20 fps resulting in 1200 frames. There are approximately 65000 single-camera detections and almost 300 tracklets. The sequence captures one shift and one change, and we can see 24 players (10 players from each team, two goalies, and two referees). Throughout the thesis, the sequence is referred to as **1min**. The first part of the sequence capturing the first shift is referred as **1shift**.

The sequence serves for the evaluation of all steps of the pipeline. Single-camera and multi-camera detectors are evaluated statistically, and tracklets clustering is completely labelled. We randomly sampled frames and evaluated detectors on the sample. Results and details are in chapters 4.1 and 4.2. Similarly, we analysed randomly sampled tracklets for evaluation in chapter 4.3.

We identified all tracklets in the sequence to test the last step of the pipeline. Annotating the short sequence with less than 300 tracklets took around 2 hours. Again, details of the evaluation are in the chapter 4.4.

3. Method

3.1. Key concepts

Let us introduce and define key concepts used in this work now. Some of the definitions are taken from our previous work [30]. Here we define only terms used throughout this work crucial for tracking. Concepts introduced in this chapter but not used anywhere else are defined in relevant chapters.

The **player's location** is the projection player's centre of the mass onto the playing field plane. The definition account for various players' poses during the match.

Detection is the object's location in the image. Various output types range from bounding boxes (the smallest rectangle encompassing the object) to masks (pixel-wise object segmentation). We also use single-camera detections and multi-camera detections in this work.

Single-camera detections are both a bounding box and a player's estimated location. The bounding box is the standard output of the detection algorithm, whereas the estimated location is suitable for further manipulation.

Multi-camera detection is the player's location estimated from multiple single-camera detections.

A Player's **identity (ID)** is a unique tag, a number in this work, assigned to each detection. The only defining property of the ID is that all detections of one player have the same ID, and no two detections of two players have the same one. The ID is a virtual concept and is not mapped to real-world identities (names). The mapping is beyond the scope of this work. Later in the text, we use abbreviations for single-camera ID (SC_ID) and multi-camera ID (MC_ID). For SC_ID, the defining properties apply only in the space of one camera, while for MC_ID, it applies in the space of all cameras.

Tracklet is a set of detections. A tracklet should contain only one player (detections with the same ID). We distinguish between single-camera tracklets and multi-camera tracklets composed of single-camera and multi-camera detections.

A set of tracklets form a **track**. Although we could again distinguish single-camera and multi-camera tracks, we use only multi-camera ones in this work. Therefore, we always mean a multi-camera track when referring to a track without specification.

Identification is the process of assigning an ID to each detection. We can also assign an ID to the whole track (tracklet), meaning all detections from the track (tracklet) have the given ID.

Re-identification is assigning an already used ID to new detections. This process is crucial in long-term tracking, especially in sports, as players reappear regularly.

Radial distortion is an image deformation caused by lens curvature. Barrel distortion means that points are displaced from their original position toward the centre of the picture. Points that are farther from the centre move more than those close. Barrel radial distortion results in the square looking like a barrel (hence the barrel distortion).

With all key concepts defined, we can continue by introducing our method.

3.2. Pipeline overview

This chapter introduces a new method for unsupervised player detection, identification and tracking in a multiple-camera setup described in chapter 2.2. We depict not only the final algorithm but also the issues we faced and unsuccessful approaches.

The proposed algorithm takes multiple videos of a floorball match from different angles. The algorithm also needs videos to be synchronized in time, calibration matrices and homographies between cameras.

The program's main output is tracking data in the MOT Challenge format. Byproducts are datasets for identity training and projection of player positions onto the playing field. Furthermore, as one of the algorithms used in the pipeline is a single-camera tracking-by-detection neural network, the proposed pipeline could provide feedback for fine-tuning this network.

In the picture 3.1, the whole pipeline is visualized; the data flow from top to bottom in the direction of arrows. The only inputs are four cameras in orange colour on the top edge of the image. Blue labels describe intermediate outputs. We need synchronized videos for tracking, so the first step is *Synchronization And Calibration*. The rectangle is grey as the process is not fully automated. For a detailed explanation, see chapter 3.4. Next, we process each synchronized video by the Multiple object tracking (MOT) algorithm and receive single-camera detections. The MOT algorithm runs independently on all four cameras, and only two images are visualized for clarity. Details of this step are described in chapter 3.3. Single-camera detections and geometric calibration are necessary for geometric verification, which is the following pipeline step depicted in chapter 3.5. We can construct tracklets as described in chapter 3.6 with geometrically verified multi-camera detections. The last chapter explains the last two steps - ID learning and tracks creation. These steps are in the same section for their possible connection to the iterative approach described in chapter 3.7. Images used in the visualization 3.1 are further explained in their corresponding chapters.

Since tracking in sports videos with regular players changes (as is the case of floorball) requires a considerable amount of re-detection and re-identification, we chose the tracking-by-detection approach. This approach also leverages the re-identification of the player after a cluster during the game. While we introduce new methods, most of the pipeline uses existing algorithms.

3.3. Single camera detection and tracking

The detection task has some common problems. First and the major is detecting objects hidden behind another called occlusion. The occlusions occur regularly in a public dataset for detection as COCO or PASCAL. Other frequent problems are variability in detected objects, view angle, and image transformation.

Sports videos are characterized by a significant amount of occlusions and shape changes. During the match, players appear in uncommon poses. On the other hand, a game clearly defines where a player can be and limits the number of players on the pitch.

Our cameras also suffer from extensive radial distortion as their viewing angle is 180°. The distortion is visible in the picture but does not deform players locally. The players are more rotated than deformed, and the detection algorithms did not have any problems with the distortion. Therefore, we do not straighten images before running the detection. Radial distortion compensation is used only in projecting detections to

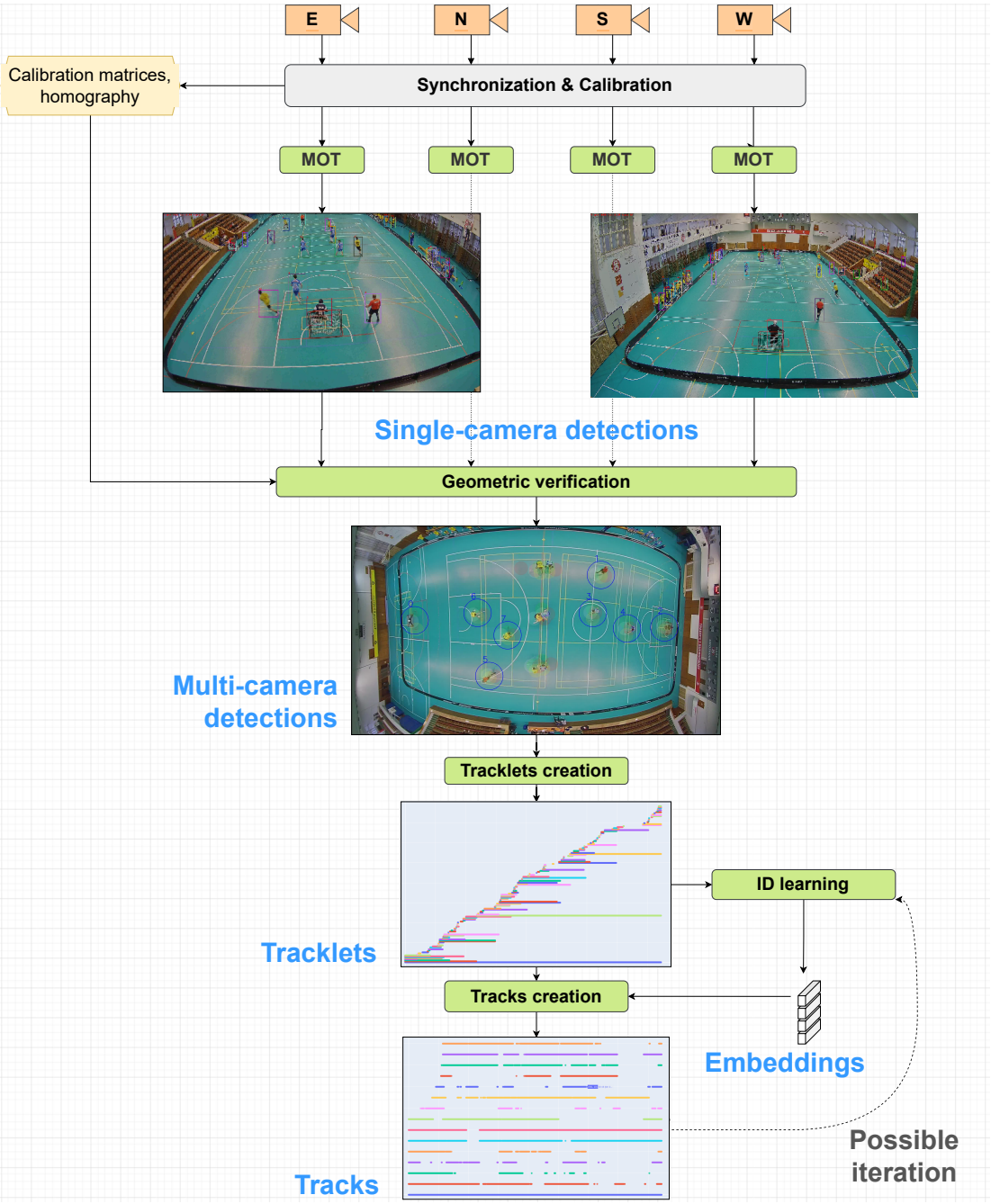


Figure 3.1. Visualization of the pipeline. Green rectangles are automated steps, and the grey one is manual. Inputs are orange cameras. Visualization of intermediate steps are explained later.

the common space as described in chapter 3.5.

Currently, deep convolutional neural network (DCNN) is the most successful algorithm for visual object detection. Networks take images and predefined classes and return detected objects' positions. In this thesis, the used detection algorithm has only one class (human), so the detection is the same as classification.

For the detection part of the system, we chose the ByteTrack [44] algorithm. The ByteTrack is a single-camera MOT) algorithm using YOLOX [12] detector. The following section shortly explains ByteTrack and its features. The rest of the chapter addresses the problems we faced.

3.3.1. ByteTrack

At the time of writing this thesis, the ByteTrack is the best-performing algorithm for benchmarks MOT17 and MOT20 [21] with publicly accessible code [51]. It is easy to use and fast to employ. That is why we use it for the detection part of the pipeline.

ByteTrack is a tracking-by-detection method. It relies on precise detections provided by YOLOX. ByteTrack is modular, so a better-performing alternative can easily replace the detection network in the future. Furthermore, as our final pipeline does not leverage tracking information from single-camera tracking, we can directly use YOLOX for detection and reduce running time. We believe the information from the single-camera tracker could prove helpful in future work, and we opted to keep it. Even with obsolete single-camera tracking, ByteTrack runs at 20 fps, sufficient for real-time video analysis.

ByteTrack is available under MIT license on the official GitHub repository.

3.3.2. Top camera behavior

Most detectors are trained on side-view data as those are common. This fact becomes a problem for the TOP camera. The camera sees players directly from above, which is a never encountered viewing angle during YOLOX training. Therefore, pretrained YOLOX performs poorly in the TOP view camera. Moreover, the viewing angle is different for players in the centre of the pitch and those at the sides. As we can see in image 3.2, the most challenging part is the central circle where the camera sees only the head and shoulders. Also, players in the lower parts of the image are turned upside down, further complicating the detection.

The naive solution to the upside-down problem is rotating the image and running the detection multiple times on rotated images. This solution can further be improved, but it does not solve the central circle challenge. We would need to retrain the detector on annotated bird-view images to detect players from above. Since we do not have annotations for our data and obtaining them is expensive, we decided not to use the TOP camera in the pipeline. No information from the TOP camera is used in this work, and images serve only for visualization.

The image 3.3 shows the input of the stage. Each bounding box represents a single-camera detection. The small number on the top of the bounding box is the single-camera track number generated by the ByteTrack.

3.4. Geometry

To ensure the multi-camera tracking, we need to project detections (or tracklets) into a shared space. This space can be virtual as described in 3.5. We opted for the space of the TOP camera for more comprehensible visualization. Having a ceiling-mounted camera

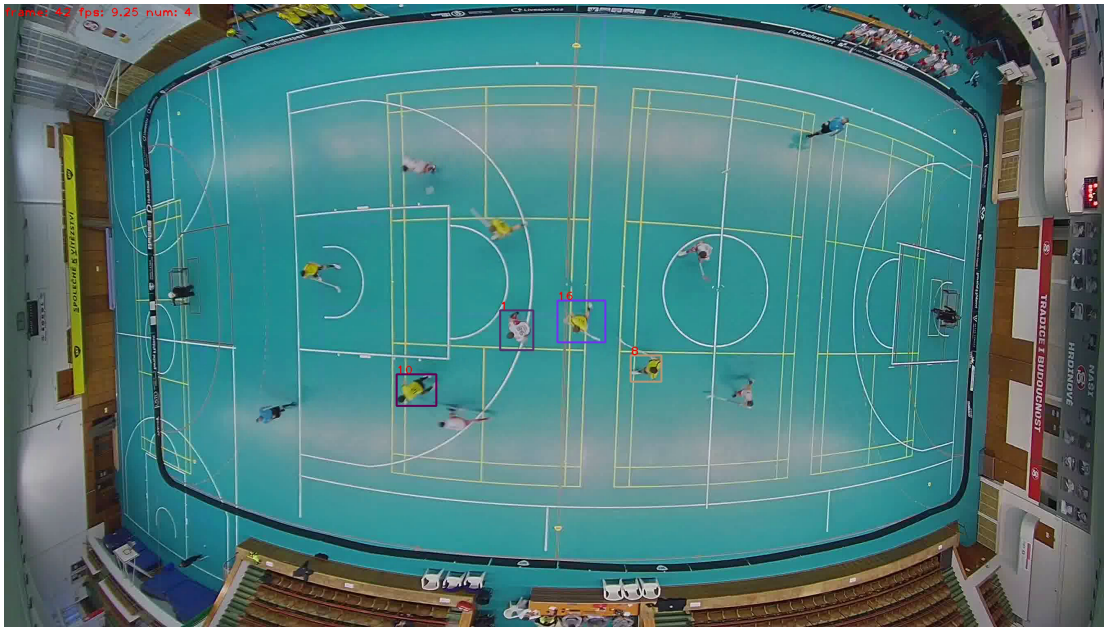


Figure 3.2. Example of detections from the TOP camera. Notice the amount of false negatives.

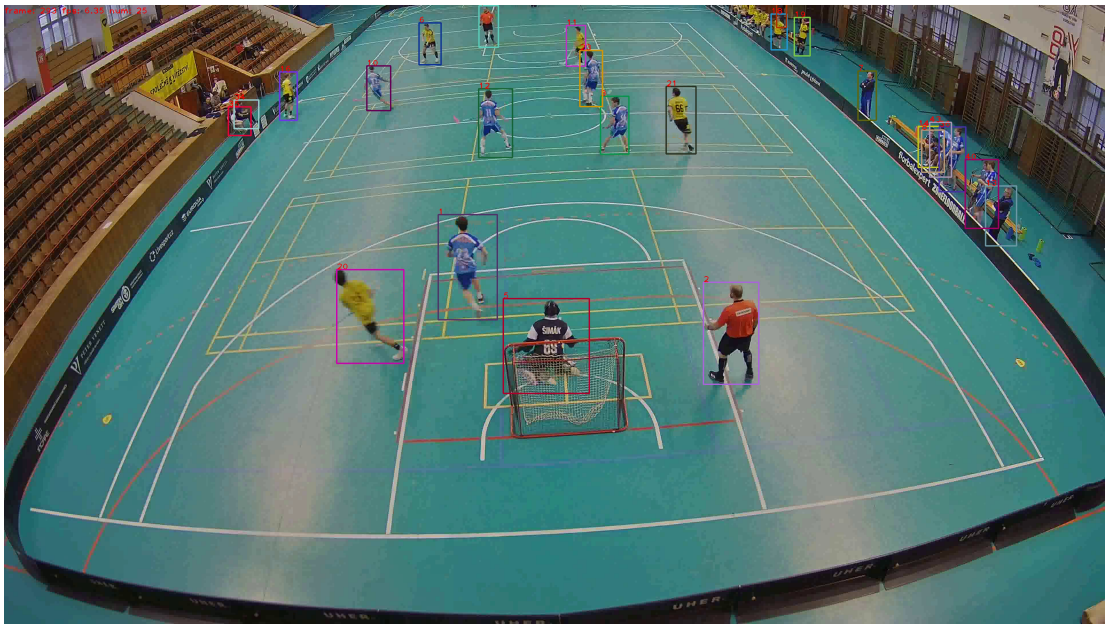


Figure 3.3. Example output of the single-camera detector. Small numbers are identities assigned by ByteTrack.

is unique to our project, so this option is usually not available. Another alternative would be a virtual drawing board commonly used by coaches. We used this approach in our previous work [30].

Regardless of the chosen shared space, we need to project detections to it. It is not necessary to project tracklets back to the original image for tracking. However, re-projection could help fix single-camera detections and tracking.

The simplest projection between two images (side-view and TOP cameras in our case) is homography. Homography is the transformation between two planes. The playing field and its image in each camera represent planes for projection. Homography estimation means finding a 3×3 matrix representing the transformation. We can project any point on the pitch from one image to another with homography only when images are without radial distortion.

One of the more refined approaches for detection projection would be point triangulation in the 3D space. Triangulation would permit us to reconstruct the player's pose in 3D, leading to a better understanding of the scene. The reconstruction would be computationally expensive and would require point-to-point matching. We rejected this approach, acknowledging that it is a possible direction for further research.

3.4.1. Radial distortion

Before computing homography, we had to straighten images by correcting the radial distortion. Although radial distortion is present in all cameras, the fisheye cameras on longer pitch sides proved problematic. As we can see from the image 3.4, the camera is not distorted symmetrically. While horizontal lines are almost straight, vertical lines are subject to strong barrel distortion. We opted for the TOP camera as the example image because straight lines are clearly visible there.

We believe the sensor in the fisheye cameras is symmetrical, but the camera manufacturer straightens the image along the horizontal axis by software in the camera. We contacted the company but could not confirm the suspicion.

Chessboard calibration

First, we tried to calibrate cameras using the standard approach with a perpendicular chessboard. This approach is implemented in the OpenCV [3] and computes distortion and calibration parameters based on the detection of the known pattern. It requires several images of the same pattern from different angles or distances. Since all cameras are relatively far from the playing field (at least 10 meters from the centre), the pattern should also be far enough from the camera. The first challenge comes as the TOP camera is mounted 14 meters above the pitch, and the chessboard would have to be large for precise detection. Since all fisheye cameras are the same model, we tried to calibrate the more accessible ones (NORTH and SOUTH) and use the parameters for the TOP camera. The results were unsatisfactory.

The OpenCV library provides a new model specifically for the fisheye camera. The model can work with higher polynomials but relies on symmetrical distortion. Interestingly, the fisheye model proved more suitable for non-fisheye cameras with symmetrically distorted images.

Manual grid calibration

To solve the issue with chessboard detection, we manually a square grid on the pitch and manually marked corners of the grid in all images. We then treated points as

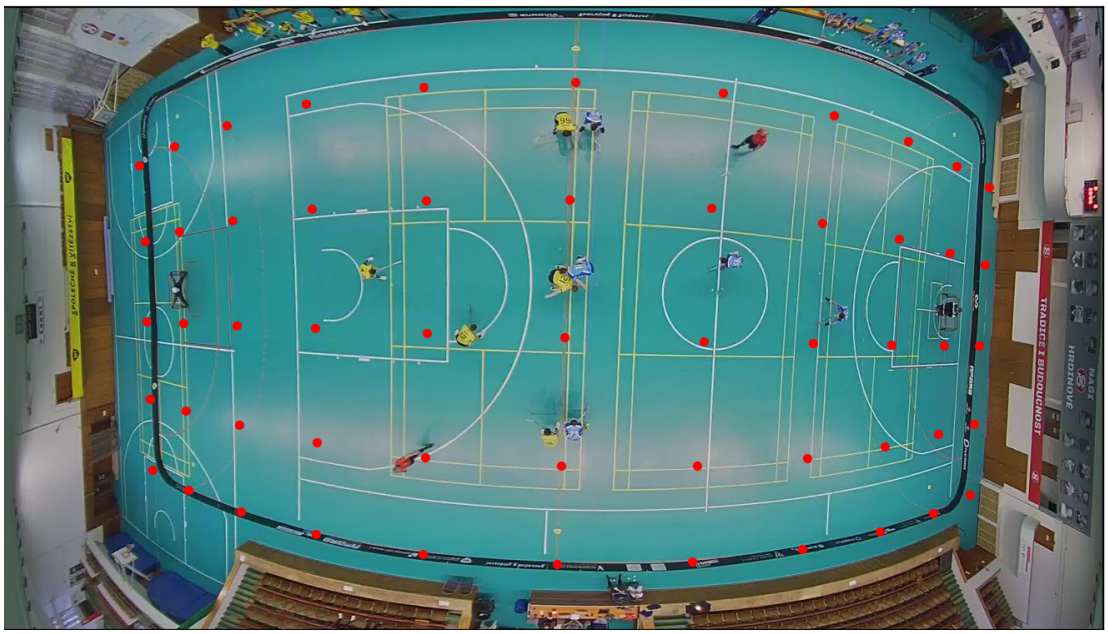


Figure 3.4. Distortion of the TOP camera. Red dots are manually labelled grid points.

detected chessboard patterns and proceeded with the OpenCV calibration module.

As shown in the image 3.5, lines are straight inside the grid but still distorted outside. We judged this straightening sufficient for the projection.

All cameras are calibrated using a manually created grid. Non-fisheye cameras use the fisheye module of the OpenCV, while fisheye cameras do not.

Further research

Camera calibration proved a complex problem not easily done using off-the-shelf solutions. In the subsequent research, we propose automating the playing field detection process as mentioned in [27]. With precisely detected pitch, we can estimate both camera calibration and homographies. Further, automatic pitch detection in every frame would connect moving broadcast cameras with static ones used in this work.

3.4.2. Homography

With radial distortion fixed, we can continue with homography estimation. Again, the OpenCV library provides an off-the-shelf function to estimate homography between two sets of points. Our first step was to use the solution. The pitch is designed to be as clear as possible, so it is no surprise that points on the playing field plane are not discriminate enough to create clear correspondences. Since we already had the manual grid from the calibration, we used the same points for homography estimation.

As points for the calibration are manually selected, and the radial distortion model is approximate, point-to-point correspondences suffer from significant errors. Estimating the homography by standard methods was not precise enough. To achieve better precision and even fix part of the distortion problem, we came up with an approach of local homographies. From a grid of manually created points, we created a grid of local homographies defined by the nearest square from the grid. Each square in the grid represents a local homography. When projecting a point, we find the nearest square of the grid and use appropriate homography.

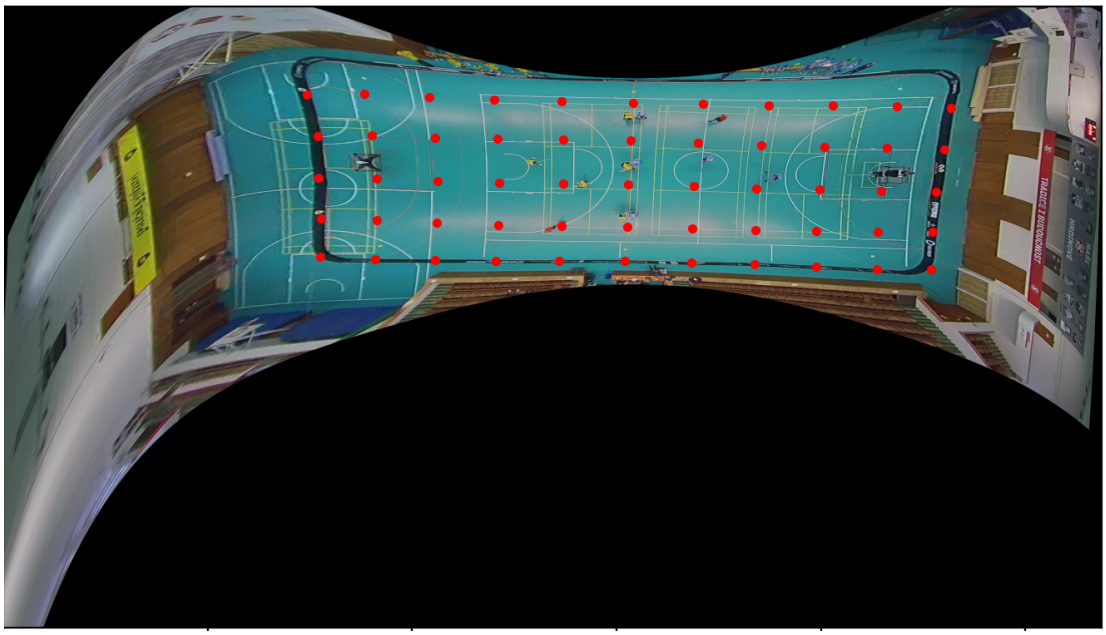


Figure 3.5. Undistortion of the TOP camera. Red points are manually labelled grid points. Notice the curved line in the top left corner of the pitch.

The approach accounts for nonlinearities in the radial distortion, partially fixing the error made by the wrong distortion estimation. For points inside the grid, the projection is almost perfect. Points outside of the grid have more significant errors as the local homography works only in the neighbourhood of the square. The solution would be to enlarge the grid to cover the whole pitch.

The picture 3.6 visualizes the local homographies approach.

3.5. Multi camera detection

Once we have single-camera detections from individual cameras and geometric models for projecting between each camera and the shared space, we can proceed with multi-camera detection.

Multi-camera detection means locating the player in the shared space, in this work, the space of the TOP camera. Since homography can only project points on the playing field plane, multi-camera detection is a single point corresponding to the detected player's location.

Similarly, we need a single point describing the single-camera detection, which we can project to the shared space. This point should be as close as possible to the true player's location. In the early draft of our project, we used the centre of the bottom edge of the bounding box as an approximation, and it worked surprisingly well. It is far from the player's location and would be insufficient as an approximation when using only one camera. The approximation is biased toward the camera, so the approximated point is always closer to the camera than the location. These biases cancel out when using a set of perpendicular cameras and averaging the individual location estimations. Visualization of the location are in the image 3.7.

When transferring single-camera detections to multi-camera ones, we also solve the problem of false positive detections. We assume that false positives are independent between cameras, and we could eliminate them by voting. We need to decide the

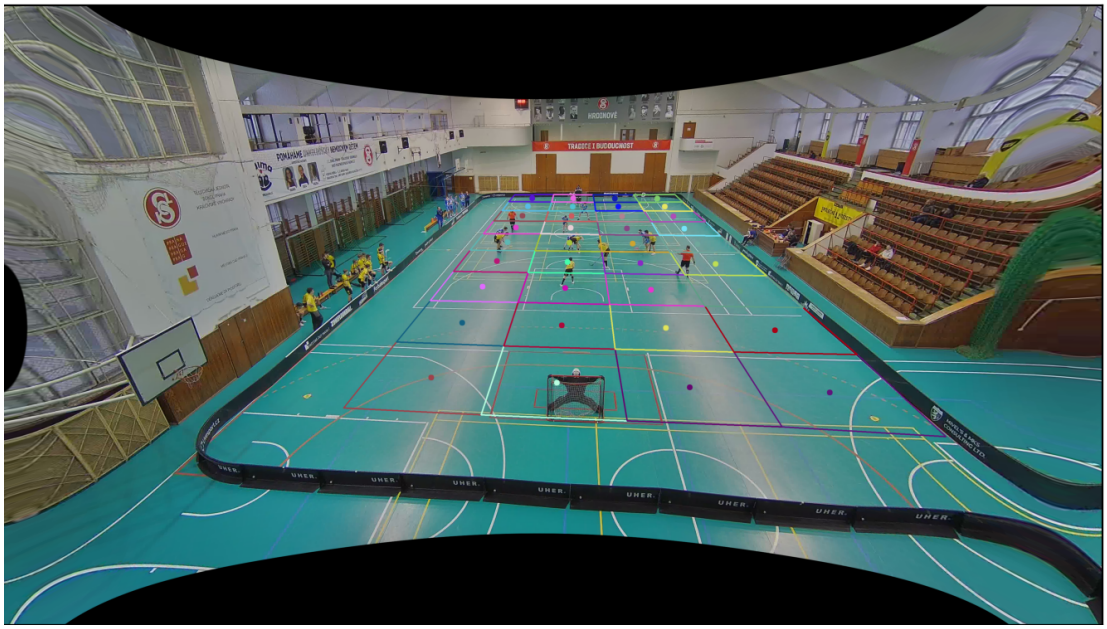


Figure 3.6. Visualization of the local homographies approach. Each square represents one homography. Points are centres of squares.

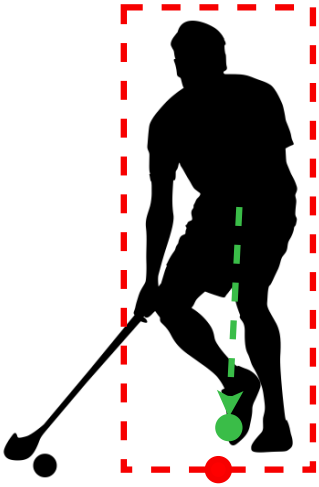


Figure 3.7. Visualization of the player's location. True player's location in green, our estimate in red. Silhouette of the player taken from [53].

number of votes (or percentage of all cameras) sufficient for a multi-camera detection. If we set this threshold too low, we would not eliminate false positives. If we set the threshold too high, we would lose detections and have too many false negatives. With the assumption of independent false positives, we chose the threshold to 2 cameras as we judge the probability of two false positives co-occurring and at the same place small.

We also employ hysteresis thresholding on geometric verification to decrease the number of false negatives. Suppose only one camera detects a player where one was verified (detected by more than one camera) in the previous frame. In that case, we keep the detection, even though the information is only from one camera. Hysteresis thresholding is only in the forward direction, as situations where backward hysteresis would be beneficial, are rare. With this approach, we can process frames sequentially in real-time. The hysteresis thresholding is the only source of false positives in the multi-camera detection, as we will see in chapter 3.5.

The last issue to solve is players clustering. Players tend to cluster in a small area during the game, especially in front of the net. Detecting players in such crowds is extremely hard because of many occlusions. Having more than one player from the same team in such clusters is common, so team classification helps only a little. We decided to ignore these clusters and not detect players in them. Strictly speaking, it brings a lot of false negatives to the process as players spend approximately 50% of the time in clusters (number taken empirically from our data). However, we still have information about the occurrence of the cluster from single-camera detections. The information is available for further processing, but we do not use it in this work.

To avoid detecting clusters, we had to define a distance when the player is safely far from all other players. As will be evident in the section 3.7, choosing a long distance in the first iteration of the pipeline is advantageous for obtaining a clean dataset for training the identification network.

Lastly, single-camera detections are filtered by manually created masks to avoid false detections outside of the pitch.

3.5.1. Clustering algorithm

The algorithm 1 defines the process of geometric verification.

In the algorithm 1, we use *AgglomerativeClustering()* function from library SciPy [35]. The clustering is thresholded by the fixed distance, as described above.

Results of the clustering are visualized in the picture 3.8. Each single-camera detection is shown as a coloured transparent circle of size $\frac{threshold}{2}$, where *threshold* is the discussed minimum distance between detections. Both single-camera and multi-camera detections are represented by a circle instead of a more common bounding box to emphasize that detection in the shared space is only one point. We do not know anything about the size or shape of the detected player in the shared space. Multi-camera detection circles have a radius of size *threshold* and an ID of the corresponding tracklet from the next step.

In the multi-camera detection phase, we did not fully leverage the YOLOX detections and ByteTrack identities. As mentioned previously, ignoring crowded areas creates false negatives hindering the tracking. The side-cameras have data about players in the crowd, and we can compute the number of players in each group. Further, the ByteTrack in each side camera assigns an identity to each detection so we can connect two tracklets through the crowd by voting from side cameras. The approach is not independent as there are cases of the same ID switches in two or more cameras, and with only four side cameras, it corrupts the voting. The approach would combine well with

Algorithm 1: Geometric verification of single-camera detections

```
Data: Single-camera detections, geometric relations between cameras
Result: Multi-camera detections
for each frame do
    clusters ← AgglomerativeClustering(detections_in_frame);
    multicamera_detections ← ∅;
    for each cluster in clusters do
        if no two detections from the same camera then
            if at least 2 detections then
                multicamera_detections += cluster_centre;
            else
                if detection nearby in previous frame then
                    multicamera_detections += cluster_centre;
        else
            Cluster, ignore;
```

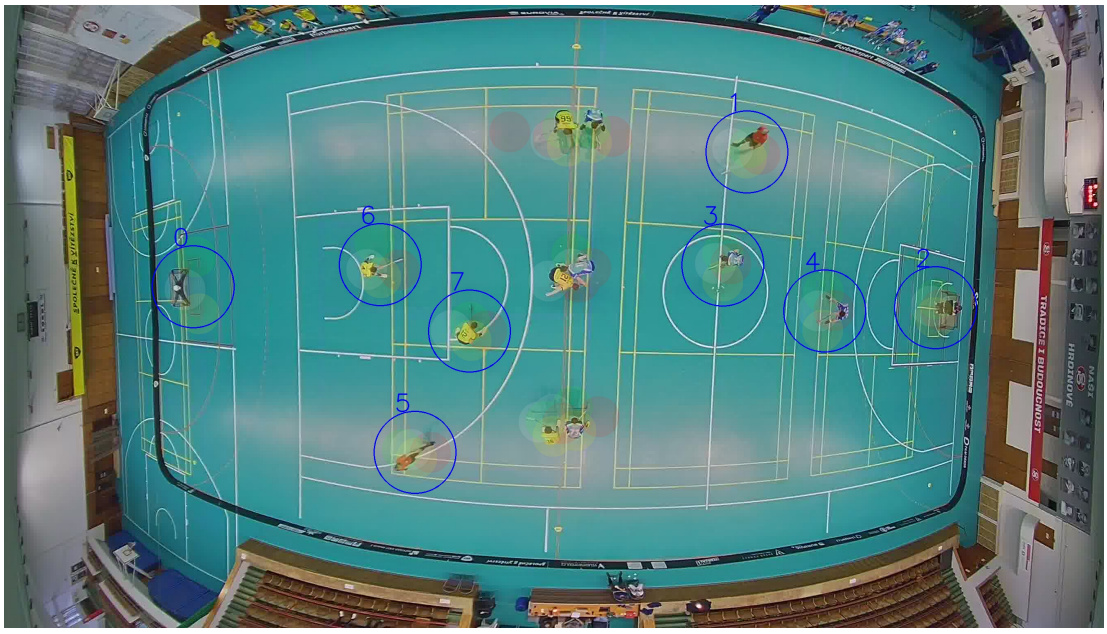


Figure 3.8. Example output of the multi-camera detector. Numbers above detections are tracklets IDs from the following step.

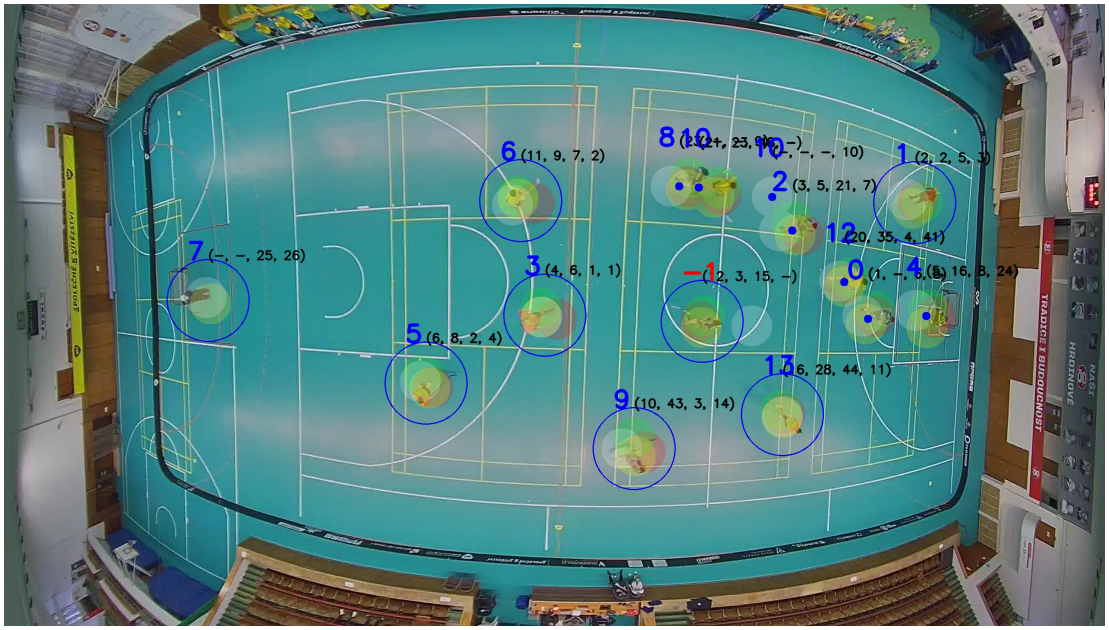


Figure 3.9. Visuzalization of the tracking with data from side cameras. Blue circles are multi-camera detections defined previously. Blue dot represents detected player in a crowded area. Blue numbers represents multi-camera IDs, black numbers depicts identities from side cameras. The red number signifies error in voting.

the similarity metric defined later in chapter 3.7.2. We conducted some experiments in this direction, but we do not have publishable results when writing this thesis. The image 3.9 shows an example of correct detections in a crowded area using the data from side cameras. A circle around the player signifies the standard multi-camera detection described previously, and a blue dot represents a detection in the crowd. Black numbers above players are identities from side cameras.

3.6. Multi camera tracking

Tracklets are obtained by clustering detections according to the constraints. Common constraints are:

- **Time constraint**: any two detections from the same tracklet cannot occur simultaneously.
- **Position constraint**: distance between positions of two consecutive detections is limited by the maximal speed of the player in the real world.

The following paragraph discusses our approach to tracklets creation.

3.6.1. Careful approach

We decided that the re-identification algorithm has to be a crucial part of our method and designed tracklets creation in such a way to ensure tracklets clarity. The algorithm 2 leverages only time and positional constraints in the most straightforward way. It process frames sequentially, so the time constraint cannot be violated. It does not attempt to re-identify players as that would be the task of the latter part of the pipeline. Once the tracklet is lost in a crowd, it creates a new one, resulting in many short

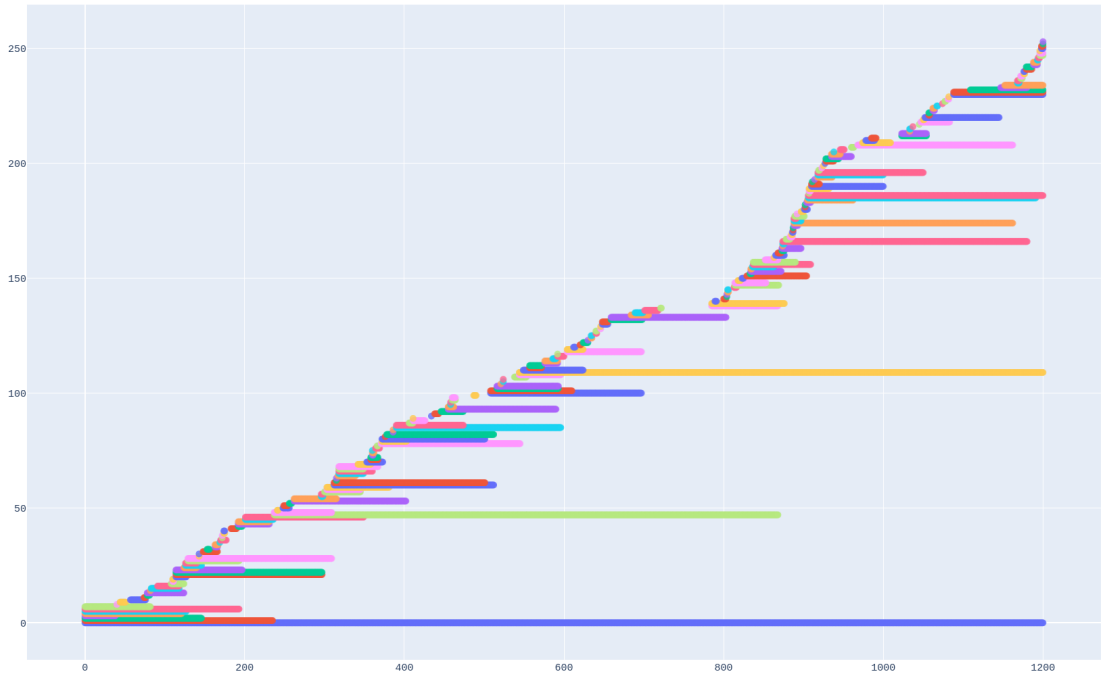


Figure 3.10. Example output of the tracklets creation step. The x-axis represents time, y-axis identity. A point in the graph means that the tracklet with identity Y (y-axis) is in frame X (x-axis).

tracklets. There are no ID switches inside tracklets, and the only issue for identity learning arises from false positive multi-camera detections.

Algorithm 2: Careful approach to tracklets creation

```
Data: Multi-camera detections
Result: Multi-camera tracklets
prev_detections ← ∅;
for each frame do
    frame_detections ← alldetectionsinframe;
    HungarianAlgorithm(prev_detections, frame_detections);
    for each unassigned detection do
        assign new id;
    prev_detections ← frame_detections;
```

As mentioned in the algorithm 2, the core of the clustering is assigning identities between frames. We used the Hungarian algorithm [19] with thresholding implemented in the SciPy library [35]. The cost of the algorithm is the euclidean distance between detections.

The image 3.10 from the visualization of the whole pipeline 3.1 shows individual tracklets in the sequence. The graph represents occurrences of tracklets in the time. Each dot in the graph represents a tracklet (y-axis) occurring in the corresponding frame (x-axis). Since tracklets are not connected, once the tracklet ends, there are no new dots on the line. New tracklets appear in time, resulting in the diagonal shape of the graph.

3.7. Player identification, re-detection

This chapter describes the last step of the pipeline - tracks creation. The previous step ensures that all tracklets are pure, so there are no ID switches inside any tracklets. Therefore if we connect tracklets correctly, tracks will also be pure. The only question remain is how to connect tracklets correctly.

In this thesis, we use both terms *tracks creation* and *identitfication* for the same process. We create tracks by clustering tracklets which are without ID switches. Identification in computer vision means connecting two images of the same person while distinguishing two images of different persons. By correctly connecting tracklets, we effectively identify the player.

The following paragraphs describe the tracks creation algorithm. The rest of the chapter is dedicated to details of the algorithm and our experiments.

3.7.1. Tracks creation

The algorithm 3 defines the time-sensitive centroids-linked agglomerative clustering used for track creation. We used the centroids-linkage to speed up the computation and save memory. It allows us to represent each cluster by only one vector, significantly reducing the distance matrix. The algorithm clusters (merge) tracklets based on the similarity metrics and occurrence in time. Two tracklets cannot be merged if they occur in the same frame. It is similar to the time constraint from the tracklets creation part (chapter 3.6).

Algorithm 3: Tracks creation by time-sensitive agglomerative clustering

Data: Multi-camera tracklets, visual similarity metric, max distance threshold
Result: Multi-camera tracks
$distance\_matrix \leftarrow SimilarityMetric(tracklets)$;
$diag(distance\_matrix) \leftarrow \infty$; `Distance to self is` ∞
for *each tracklet* **do**
  for *each frame where the tracklet is* **do**
    for *each other_tracklet in frame* **do**
      $distance\_matrix[tracklet, other\_tracklet] \leftarrow \infty$; `Time-sensitive`
      $distance\_matrix[other\_tracklet, tracklet] \leftarrow \infty$;
for *number of tracklets* **do**
  $min\_dist \leftarrow min(distance\_matrix)$;
  $min\_tracklet1, min\_tracklet2 \leftarrow argmin(distance\_matrix)$;
  if $min\_dist = \infty$ **then**
    return ; `No tracklets merge possible`
  if $min\_dist >$ *max distance threshold* **then**
    return ; `Distance is to big, finish`
  $MergeTracklets(min\_tracklet1, min\_tracklet2)$;
  $UpdateDistanceMatrix(distance\_matrix)$;

The pseudocode describes the algorithm in more detail. Once we construct the distance matrix, we keep it updated after every merge. The algorithm stops once there are no tracklets to be merged or if we reduced the number of tracklets sufficiently (given by threshold). The thresholding becomes useful during the player's change. We know

that there is always the same number of people for the sequence of one shift, and we can merge tracks aggressively (set the threshold to infinity). With shifts exchange, we have to threshold the clustering, or we would connect tracklets of two different players to the same track. We could also threshold the clustering by maximal distance as we did in the experiment 4.4.5, but it did not improve the clustering.

The presented clustering algorithm does not ensure the position constraint from the chapter 3.6.

We did not define the distance function in the algorithm. Next, we describe the distance function and justify its construction.

3.7.2. Distance function

We are looking for a distance function which will return a single number for any two given tracklets (or tracks). For the clustering algorithm to work correctly, the distance of any two tracklets with the same player must be shorter than the distance of any two different players. In other words, we want tracklets with the same player as close as possible while tracklets of different players as far as possible.

The distance function would compose of two parts - the similarity metric and the time metric. Therefore, we define the tracklet representation as a tuple $(\overline{e}, t)$, where $\overline{e}$ stands for the tracklet's embedding and t for time (set of frames) when the tracklet is present. The tracklet's embedding is the centroid of embeddings of respective detections hence the centroids-linkage in the clustering algorithm.

Equation 3.1 defines the distance function d and its respective similarity and time metrics S and T.

$$d\big((\overline{e_1}, t_1), (\overline{e_2}, t_2)\big) = S(\overline{e_1}, \overline{e_2}) + T(t_1, t_2) \tag{3.1}$$

$$S(\overline{e_1}, \overline{e_2}) = \|\overline{e_1} - \overline{e_2}|_2 \tag{3.2}$$

$$T(t_1, t_2) = \begin{cases} \infty, & \text{if } t_1 \text{ and } t_2 \text{ overlaps} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The time metric is zero unless the tracklets overlap. Two overlapping tracklets cannot be merged, and their distance is infinity. The similarity metric is the euclidean distance between two embeddings.

The following paragraphs explain the construction of the embedding space.

3.7.3. Similarity metric

The single-camera detection embedding is a vector of numbers characterizing the detected bounding box. Triplet Loss [37] is popular to learn such an embedding space. The backbone neural network encodes the image into the embedding vector such that the required distances are appropriate in the embeddings space.

The problem is that training such a network would require a vast amount of labelled data. The proposed method is unsupervised, so no manual labelling is necessary. With a perfectly working similarity metric, we could skip tracklets creation and cluster multi-camera detections directly into tracks based only on the two constraints and the similarity metric. Therefore, our goal is to train such a network to run in real-time.

The standard identification networks create embedding for each detection, but we only need one representative for the whole tracklet (or track). In theory, we only need one characteristic detection of the player for the tracklet to identify it correctly. Finding

a distinctive image (characteristic detection) is an open question, and we recommend it for further research. In this work, we used the mean embedding of all detections in the tracklet as a tracklet representation previously referred to as $\overline{e}$.

3.7.4. Pretrained model

The most straightforward way is to use a model pretrained on another dataset. We chose the CTL-Model presented in [39]. The model uses ResNet50 [15] as a backbone for the encoding and proposes a new Centroid Triplet Loss. We selected this algorithm as the best-performing one on the Market-1501 [45] dataset with the open-source code. The first experiment used the pretrained model without any edits to compute embeddings.

Detailed results of the experiment are in the chapter 4.4.2. Not surprisingly, the model performs worse without fine-tuning described later on. But even without any training on floorball-specific data, the model can distinguish two players to some degree.

3.7.5. Frame-based Triplet Loss

We designed a new hard-negative mining technique for triplet selection to compensate for the lack of annotated data for identity training. To explain it in detail, let us first revisit the CTL-Model and its Centroids Triplet Loss (CTL).

The CTL-Model uses ResNet50 as an encoding backbone but employs a new Centroid Triplet Loss. The standard Triplet Loss works with the triplet (a, p, n), where a stands for anchor, p for positive example and n for negative example. The positive example is pulled closer to the anchor during the training, while the negative one is pushed further. We require n to be the closest negative example while p is the furthest positive. Selecting n and p is one of the biggest challenges, and it's called hard-negative mining.

The Centroid Triplet Loss simplifies the process and speeds it up by working with the triplet (a, c_p, c_n). Here a is still the anchor, but c_p and c_n are centroids of the positive and negative class. Therefore we do not look for the hardest positive and negative samples but for the hardest classes. It speeds the process as there usually are significantly fewer classes than instances. The loss is also more robust to outliers as we work with centroids instead of extreme cases. According to the paper, this elegant simplification of Triplet Loss brings significant improvement.

In the published code, the images are processed in batches. Each batch is a subset of identities (classes) randomly selected for training. The identity can appear in more than one batch. A fixed number of representatives are again randomly sampled in the training process for each identity. The network then finds a triplet (a, c_p, c_n) for each instance in the batch. For clarity, the process is described in the pseudocode in algorithm 4.

Algorithm 4: Batch creation in the original CTL-Model paper [39]

```
Data: Labeled single-camera detections
Result: Batch
batch_images ← [];
batch_identities ← SampleWithoutRepetition(identities);
for each identity in batch_identities do
    identity_instances ← SelectInstancesOfIdentity(identity);
    batch_images += SampleWithoutRepetition(identity_instances);
```

We can generate an unlimited amount of short tracklets from our dataset. The tracklets are labelled, but we know the label is incorrect. We assume that all tracklets are pure (containing only one player) from experiments. Selecting a positive instance is an easy task as we know that instances in the same tracklet are of the same identity (class). We could lose a variety when selecting instances from short tracklets and overfit the learning quickly. Nevertheless, most tracklets are composed of detections from all four cameras, and we assume that perpendicular viewing angles will ensure the variety.

The only remaining problem is finding a negative instance (or, in the case of Centroid Triplet Loss, negative class). Disconnected tracklets often represent the same player but at different times, so we cannot rely on tracklets' IDs for negative class selection. We leverage the fact that detection comes from synchronized cameras of the sports video. Two co-occurring tracklets are each other's negative class as the player cannot be detected twice in the same frame. It is one of the two reasons we used a longer distance threshold in chapter 3.6. The second one is to ensure that all training images contain precisely one player. The image 3.11 shows one of the rare examples when the bounding box captures more than one player due to the occlusion. Such images are noise for the training process and should be avoided.

Figure 3.11. Example of the bounding box with two players confusing the learning.

In the code, we edited batch creation. We create batches such that all identities in the batch are from the same frame. It guarantees that selecting any of the different labels is truly a negative example. Again, edited batch creation is described in algorithm 5.

Algorithm 5: Frame-based batch creation

Data: Labeled single-camera detections, frame number for each detection
Result: Batch
batch_images ← [];
batch_frames ← $SampleWithoutRepetition(frames)$;
for *each frame in batch_frames* **do**
    $frame\_identities \leftarrow SelectIdentitiesInFrame(frame)$;
    $sampled\_identities \leftarrow SampleWithoutRepetition(frame\_identities)$;
    for *each identity in sampled_identities* **do**
        $identity\_instances \leftarrow SelectInstancesOfIdentity(identity)$;
        $batch\_images \mathrel{+}= SampleWithoutRepetition(identity\_instances)$;

The training dataset consists of tracklets longer than 19 frames, and we use 15% of tracklets for testing and 85% for training.

The original training process selects the best model by the retrieval precision in the test set. We cannot use the metric as the data is noisy. Tracklets of the same player have different IDs, and when the network correctly pushes them together, the evaluation worsens. It leads to overfitting in the training process when the network finds one positive example for each query image but then finds random images as the next nearest. Such an embedding is not suitable for our system, and we manually select the latest model before overfitting occurs. To decide empirically, we manually review random images from the retrieval and check for signs of overfitting. Examples of the correct retrieval and overfitting are in the images 3.12 and 3.13.

Figure 3.12. Example of the correct retrieval. The left image is query, red and green rectangles are false and true according to the noisy labels.

Figure 3.13. Example of the retrieval of the overfitted model. The left image is query, red and green rectangles are false and true according to the noisy labels.

Detailed results are provided in chapter 4.4.2. The network correctly learns representations of players on the testing sequence. Visualization of the output of this step is in the image 3.14. The graph shows tracks in the same way as tracklets in the previous chapter, but now we can see fewer tracks with re-detections.

3.7.6. Going beyond one match

So far, we have done the unsupervised tracking on one short sequence. But the purpose of the system is to track players during the whole match lasting 2 hours. Our last algorithms focus on expanding knowledge from short videos to the entire game or sequences from a different one.

Multiple shifts learning

We noticed that unsupervised tracking works well for sequences without player changes. Time constraints are crucial for tracking in short sequences, and the similarity metric does not need to work well as most conflicts are between players of different teams. Our idea is to manually collect short sequences capturing all players appearing in the match. The structure of sport helps us as it is common for teams to change players at the same time, and teams do not use more than 15 players in one game. It is then enough to collect three sequences with different shifts. The processing of one match is summarized in the following steps:

1. Manually create sequences of (at most) three shifts
2. Create tracks for players in their respective sequences by the proposed system

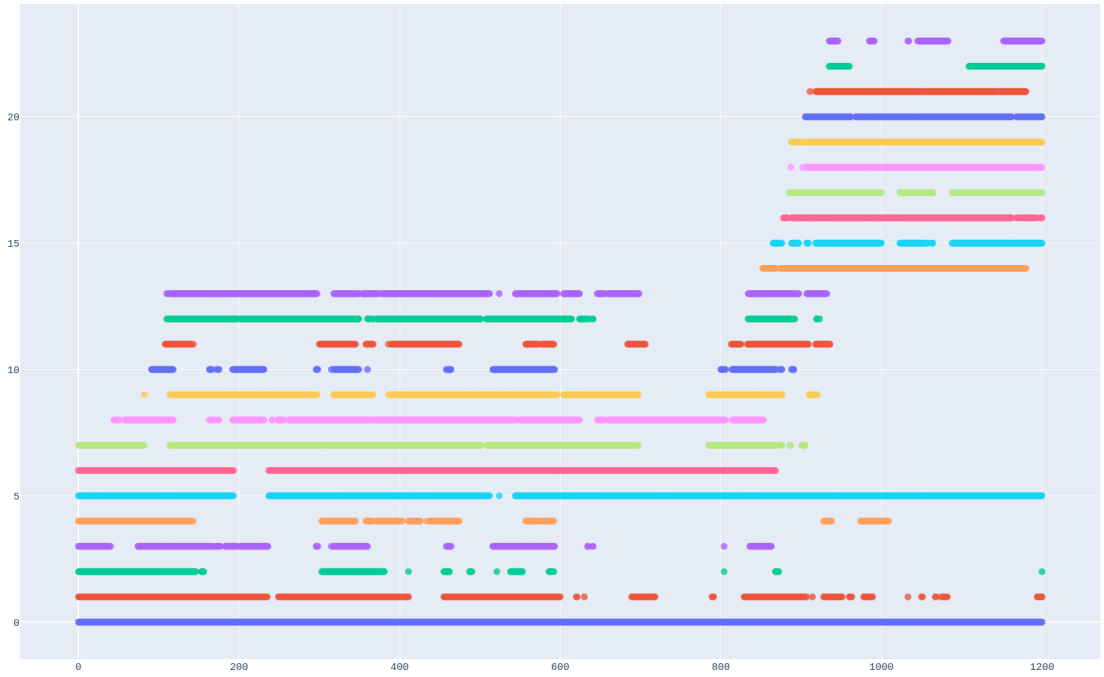


Figure 3.14. Example output of the tracks creation step. As in the previous step, the x-axis represents time and the y-axis identity. A point in the graph means that the tracklet with identity Y (y-axis) is in frame X (x-axis).

3. Combine sequences to create one dataset with all 34 players from all sequences
4. Learn the identification network using the Centroids Triplet Loss
5. Run the system on the whole match but using the CTL-Model as the identification network

The method requires perfect tracks for each sequence at the beginning of the process. When there are no ID switches in tracks, the dataset for CTL-Model training will not be noisy, and we can use the standard training. The CTL should be robust to the noise, but it is not clear to which degree.

In the future, we suggest using the human-in-the-loop approach for fixing incorrect tracks in the sequences. A human would correct the noisy data created by unsupervised tracking. Correcting noisy data is less expensive than creating annotations manually, and correct tracklets could be used for standard identity learning.

The proposed process is visualized in the image 3.15 for clarity. Experiment with the approach is in chapter 4.4.5.

Transfer learning

The perfectly working identity network would recognize players without seeing them previously. Such a network could process all matches without any (or little) training and would run in real-time. We need to create a floorball-specific dataset general enough for the network to learn to identify players without dictionary-like knowledge. The dataset is out of the scope of this thesis, and we recommend creating it in the following research.

We tested the ability of the network to generalize between matches in the experiment 4.4.6. We chose the most challenging transfer from women's (WA) to men's (U19)

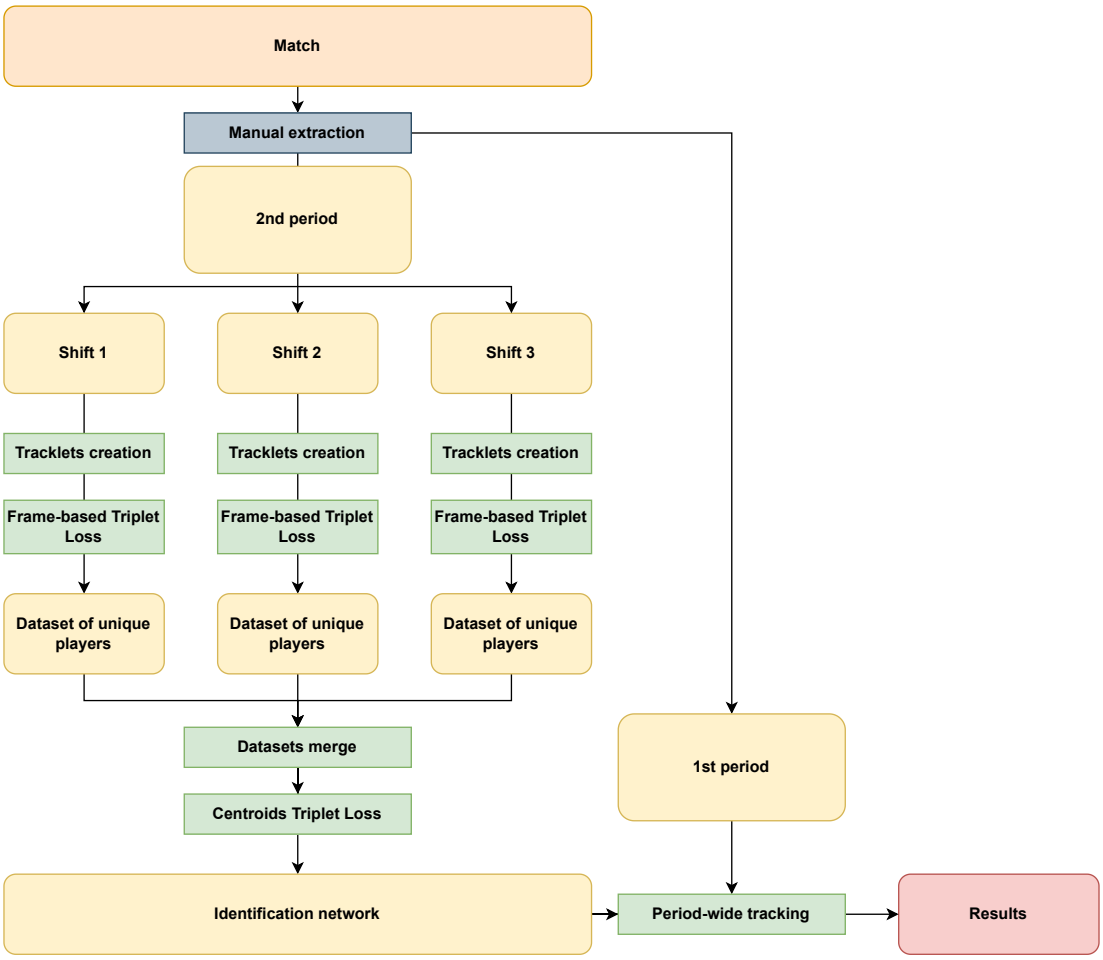


Figure 3.15. Diagram of the Multiple shifts learning experiment. Orange input is the video of one match. Yellow rectangles are outputs of intermediate steps, green ones automated steps. The result is emphasized in red.

match with jerseys' colours change. We conclude that the network did not transfer any knowledge between games, and a more extensive dataset would be required.

4. Experiments

This section describes experiments and tests mentioned in the chapter 3. It depicts the qualitative and quantitative results and briefly touches on the data.

The section is divided by the stages of the proposed pipeline. For each stage, we define and explain metrics used for evaluation, report results, and analyze extreme cases, both positive and negative.

4.1. Single-camera detector

The first evaluated part of the pipeline is the single-camera detector. As explained in the chapter 3.3, we did not alter the existing solution (ByteTrack) but merely used it on our data.

4.1.1. Metrics

We used standard detection metrics and took definitions of our previous work [30]. Since the detector uses only one class, definitions were adjusted appropriately.

True positive (TP) is the number of detections labeled by correct class. It means how many selected detections were selected correctly.

True negative (TN) is a number of detections marked as negative correctly. This metric is impossible to compute in computer vision as there is an infinite number of possible bounding boxes.

False positive (FP) is a number of detections marked as positive when it should be negative. Since we have only one class, the metric computes the number of bounding boxes where is no player.

False negative (FN) stands for all objects that were not detected.

Precision is the ratio of the number of true positives to the number of all detections. Precision 100% means that all detections were true positives.

Recall is the ratio of the number of true positives to the number of all positive samples. Recall 100% means that the detector found all positive samples

$$precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

We do not have annotations for single-camera detections, so we cannot compute intersection-over-union for the bounding box to evaluate its correctness. Fortunately, the pretrained YOLOX detector does not make errors on the verge of TP and FP. The

bounding box always clearly captures a person or not, as illustrated in the images 4.2 and 4.1.

We used the statistical approach mentioned in chapter 2.4 to evaluate the single-camera detectors. We randomly sampled 60 frames from a sequence and manually evaluated detections in the sample. For a fair comparison, we used the same frames from all cameras. Every situation is easier for different cameras, so using the same frame for all cameras would favour some cameras over others. The effect should diminish with enough random frames, and we could compare cameras.

4.1.2. Evaluation

The table 4.1 shows evaluation on all side videos (cameras EAST, NORTH, SOUTH and WEST) of the 1min sequence.

Camera	TP	FP	FN	Precision	Recall
EAST	298	0	9	1.00	0.97
NORTH	306	2	25	0.99	0.92
SOUTH	326	0	7	1.00	0.98
WEST	309	5	8	0.98	0.97
Overall	1239	7	49	0.99	0.96

Table 4.1. Results of single-camera detections in all side cameras. We evaluated the detection statistically on 60 randomly sampled frames.

The detection results are impressive from all four cameras. The camera NORTH has the most false negatives because it often does not detect goalies. We speculate that the goalies are far, and the fisheye lens rotates them significantly. The YOLOX detector is not suitable for detecting rotated persons, as we saw in the case of the TOP camera. Except for goalies, the only source of false negatives is heavy occlusion. There are only a few false positives in all cameras, and most of them are from camera WEST. The images 4.2 and 4.1 shows examples of false negatives from the NORTH camera and false positive from the WEST camera.

4.2. Multi-camera detector

The multi-camera detector builds upon the results of the single-camera detector. Chapter 3.5 explains that creating multi-camera detections is the same as geometrically verifying detections from side cameras by voting. Merging information from 4 independent detectors fixes some errors while making new ones.

4.2.1. Metrics

We used the same metric and evaluation technique as for the single-camera detectors. We took advantage of the TOP camera and evaluated the sequence visually from the TOP camera's frames.

4.2.2. Evaluation

The table 4.2 sums up the results of the multi-camera detection. We used the sequence 1min from chapter 2.4 for evaluation. We used the same 60 frames as in the previous chapter.

Figure 4.1. Example of false positive single-camera detection. The error emphasized in red circle.

Figure 4.2. Example of false negative single-camera detections. The error emphasized in red circle.

Metric	Value
Sequence name	1min
Number of frames	1200
Number of evaluated frames	60
TP	551
FN	307
FP	0
Measured precision	1.00
Estimated precision	**0.9804**
Measured recall	**0.6421**

Table 4.2. Results of multi-camera detection. We evaluated the detection statistically on 60 randomly sampled frames.

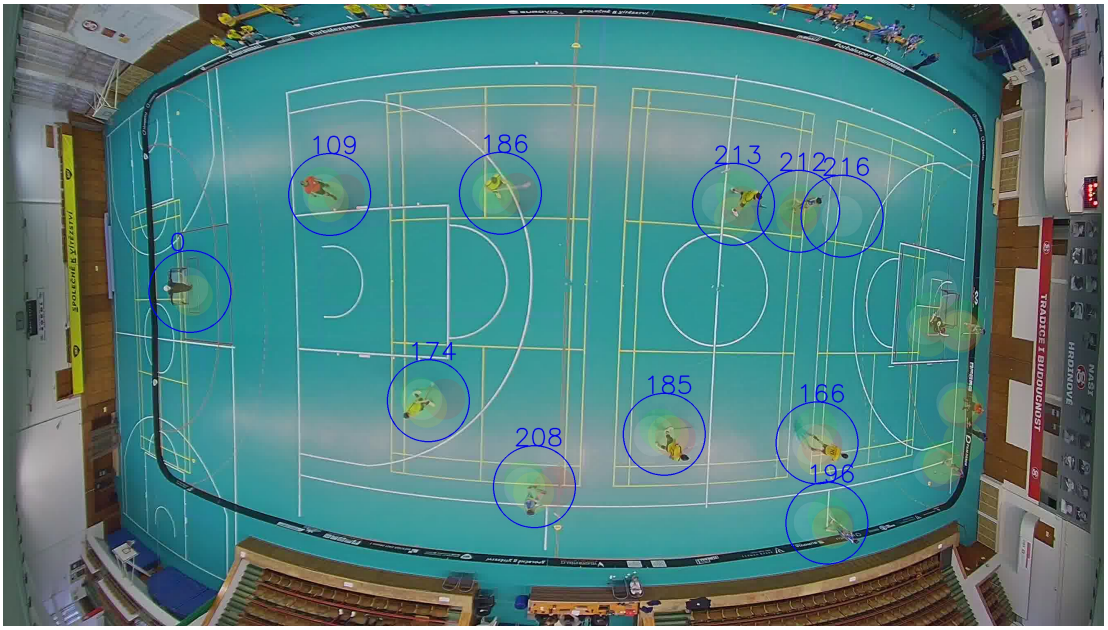


Figure 4.3. Example of a false positive multi-camera detection. The error is detection with ID 216.

The first thing we notice is perfect precision. It is no surprise, as we designed the system in such a way to trade precision for recall. With precision nearing 100%, we can guarantee that subsequent tracklets always contain at least one player, and training of the identification network is possible. False-positive detections would cause noise in the training data, which could be impossible to overcome.

However, precision 1 is misleading as false positives are in the sequence. With statistical evaluation, we did not sample any frame with false positives. When single-camera detections of previously detected player move away from each other to the point of separation (defined by the distance threshold used in geometric verification), false-positive detection appear. The phenomenon occurs primarily along the edges of the pitch, where at least one camera is too far to estimate the player's location. An example of such a situation is in the image 4.3. The problem happened eleven times during the 1-minute long sequence. Therefore, we can estimate precision to be 98.04%.

The second notable issue is the vast amount of false negatives. Again, this is caused by the design of the verification algorithm. When players are too close, we cannot

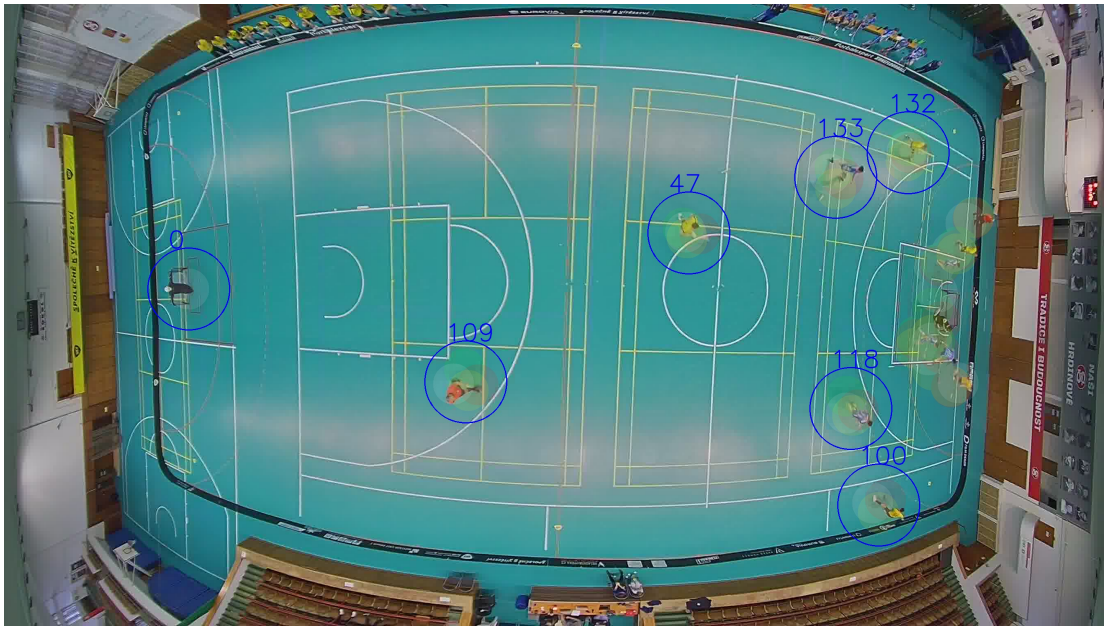


Figure 4.4. Example of a crowded area resulting in false-negative multi-camera detections.

reliably decide how many players are there or identify them from pictures. To retain clarity of the dataset for embedding training, we opted for ignoring clusters of players hence the number of false negatives. But it is not true that we would not have any information about the false negatives. Instead, we have the data from at least one of the cameras and choose to ignore it. The last column of the table recounts players not detected in any of the cameras. It is promising that we did not find any such player during the minute of the sequence. With an appropriate algorithm for dealing with clusters, combining cameras with different viewing angles could reach almost perfect precision and recall. We highly recommend it as the direction of further research.

On the other hand, multi-camera detection works well on easily separable players. When the game stretches and the players fill the playing field evenly, all players are detected, which is a substantial advantage for tracklets clustering.

4.3. Tracklets creation

The third stage connects multi-camera detections to the multi-camera tracklets. Let us first define used metrics.

4.3.1. Metrics

Since our method is unsupervised and we do not have annotations, we cannot use standard tracking metrics like IDF1 or MOTA. We defined the following metrics measuring the progress of tracklets creation.

False positive (FP) tracklets denotes tracklets with no player. FP tracklet is composed of FP detections.

identity (ID) switches are situations when players change in one tracklet. The tracklet should contain only detections of one player.

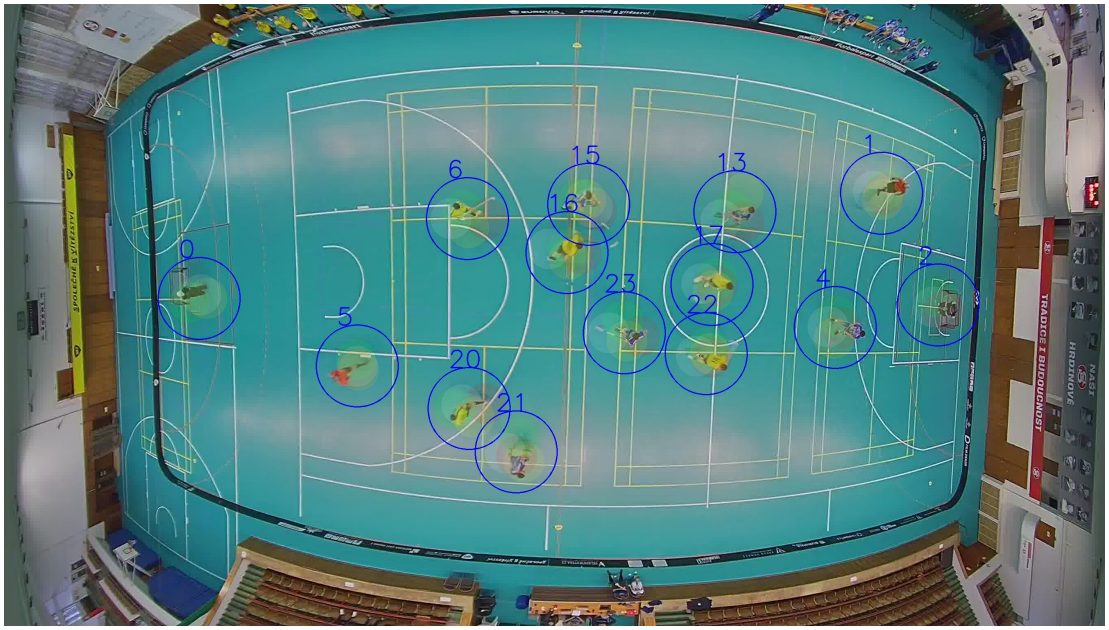


Figure 4.5. Example of multi-camera detection without errors. Detecting all players helps in the time-sensitive agglomerative clustering.

The tracklet length is the number of detections forming the tracklet. We give both median and average tracklet length as both react differently to the distribution change.

As explained in chapter 3.6, our primary focus is to minimize the number of ID switches for a clear dataset for identity training. ID switches would make impossible training described in chapter 3.7 because we would not be able to find a reliable positive sample. Even one ID switch could contaminate the tracklet such that the training is impossible. The worst-case would be tracklet uniformly divided between at least two players. Contrarily, one false detection does not have to break the training procedure as we use the Centroids Triplet Loss, which is more robust to outliers. Therefore, minimizing ID switches is not the only way to the clean dataset but the most reliable one.

Similarly to detections, we evaluate the false positives and ID switches statistically. We randomly sample tracklets from a sequence and check for false positives or ID switches. Naturally, a false-positive tracklet may arise only from false-positive detections. We mentioned in the previous chapter that false-positive detections are rare and never two consecutive. We could filter false-positives tracklets by erasing all tracklets of length two, but we would also lose some true-positive ones.

4.3.2. Careful approach

The table 4.3 reports results from the 1min sequence. We evaluated the sequence on 25 (10%) randomly sampled tracklets.

The approach is successful in minimizing ID switches. The method is cautious when not sure and splits a tracklet instead of trying to connect two tracklets. The tradeoff between tracklets lengths is visible, but we still have a sufficient amount of long tracklets for the training. The difference between the median and the average length tells us that there are many short tracklets. The longest tracklet spans the whole sequence, and it

Metric	Value
Sequence name	1min
Number of tracklets	254
Number of evaluated tracklets	25
FP	0
ID switches	0
Median tracklet length	6.00
Average tracklet length	48.39
Maximal tracklet length	1200.00

Table 4.3. Results of the tracklets creation. The step was evaluated statistically on 25 tracklets. Notice the number of false positives and ID switches.

is one of the goalies.

Similarly, as with multi-camera detections, the statistical evaluation did not find any false positive tracklets because of the random sampling. We can expect that eleven false-positive detections would create eleven false-positive tracklets of length one.

4.4. Tracks creation

The final step in the pipeline is clustering tracklets into tracks. The clustering algorithm is the centroids-linked agglomerative clustering described in chapter 3.7.1 in all experiments in this chapter. The only difference is the encoding upon which is the similarity metric computed.

4.4.1. Metrics

We use metrics similar to the previous step.

False-positive tracks are tracks containing an already tracked player. Each player should have precisely one track, and every additional track is deemed as a false positive.

Track length is the number of tracklets composing the track. For tracks, we are not interested in the number of detections making the track as the step cluster tracklets regardless of their length.

The number of ID switches is the same as for tracklets - ID switches are situations when players change in one track. The track should contain only detections of one player.

Coverage tells us how many detections are in the track. We differ between absolute coverage (divided by the number of frames in the sequence) and relative coverage (divided by the number of frames in the ground truth track). The absolute coverage cannot be 1 when the player disappears from the sequence. Contrarily, the relative coverage should always be 1 if the ground truth is available. The relative coverage is not defined when no annotation is present.

The track length is valuable for evaluating the clustering algorithm but does not tell us if the tracking "looks good". Track length increases with every two clustered tracklets, but the coverage increases with the length of clustered tracklets. The coverage

puts more weight on underlying tracklets lengths than the track length as it is more suitable for visual evaluation.

We annotated all tracklets in the 1min sequence as explained in chapter 2.4. The 1min sequence is the main testing sequence, and we compute all metrics exactly. The other sequence (1period) is too long for manual annotation, and we evaluate it statistically as in the previous steps.

4.4.2. Datasets comparison

The first experiment depicts the usage of pretrained off-the-shelf models. The [39] provides two pretrained models, one for the Market-1501 dataset and the second for the DukeMTMC-reID dataset. In addition, we trained one model from random initialization ("ours" in the table), and the last model is the Market-1501 model fine-tuned on our data (Ours+Market-1501) using the Frame-based Triplet Loss as described in 3.7.5. Detailed results from the 1min sequence are in the table 4.4. In all tables, med, avg and max stands for median, average and maximum.

Dataset		Market-1501	DukeMTMC	Ours	Ours+Market-1501
Number of tracks		24	24	24	24
FP		12	11	8	**7**
ID switches		43	48	26	**25**
Length	med	10.50	11.00	9.00	8.00
	avg	10.58	10.58	10.58	10.58
	max	27.00	27.00	33.00	33.00
Relative coverage	med	1.22	1.00	1.00	1.23
	avg	1.46	1.22	1.10	1.20
	max	2.39	2.10	2.13	2.13
Absolute coverage	med	0.43	0.47	0.32	0.32
	avg	0.43	0.43	0.43	0.43
	max	1.00	1.00	1.00	1.00

Table 4.4. Results of the tracklets clustering in the 1min sequence with the same model trained on different datasets. We compare Market-1501 [45], DukeMTMC-reID [33], Ours and combination. The best results are emphasized in bold.

Not surprisingly, the models trained on our specific dataset perform better than off-the-shelf models. The combination of the Market-1501 and Ours datasets performs the best (values highlighted in bold). We also assume that the combined model would generalize better, using it in all further experiments.

Let us also comment on the results of different metrics and explain abnormal numbers. We know that the sequence captures the movement of 24 players, and we stop the agglomerative clustering when reaching 24 clusters. By design of the algorithm, we cannot have fewer tracks than players in the sequence. The number of players is always known in the match, and we can use the information in every sequence. We provide the number of tracks in each experiment since it is possible to have more than the lower limit when no more clustering is possible because of the time constraints. We will see this happening later on.

But even with the number of players known, we can get false positive tracks. By definition above, a false-positive track is one containing a player already included in a different track. The seven false positives can mean that seven players have two tracks each, one player has eight tracks or anything in between. Each false positive must

create at least one ID switch when the tracking algorithm returns the correct number of tracks.

We use the track length primarily to measure progress during development. The longer track is not necessarily a better one, but the length of the tracks should be approximately uniformly distributed except for referees and goalies.

The relative coverage should be one. Numbers below one signify that not all tracklets of the track were correctly clustered. Contrarily, numbers above one mean more tracklets in the track than should be according to the annotation. Similarly, the absolute coverage tells us how long part of the sequence the track covers. The absolute coverage cannot exceed one, as that would mean more frames than available in the sequence. True absolute coverage depends on the characteristics of the sequence (number of shifts, duration of shifts, occurrences of clusters of players). Goalies and referees can have absolute coverage nearing one, while other players can have between 0.5 and 0.2.

From the analysis above, we can deduce that the clustering works as expected. Starting from more than 250 tracklets, the amount of false positives and ID switches is relatively low. The median and average lengths indicate that track lengths are approaching uniform distribution. The high relative coverage signifies that the algorithm clustered too aggressively and merged tracklets of different players.

We also provide visualization of the tracks in image 4.7. The graph represents occurrences of tracklets in the time. Each dot in the graph represents a tracklet (y-axis) occurring in the corresponding frame (x-axis). For reference, we also show the visualization of the annotated sequence in the image 4.6.

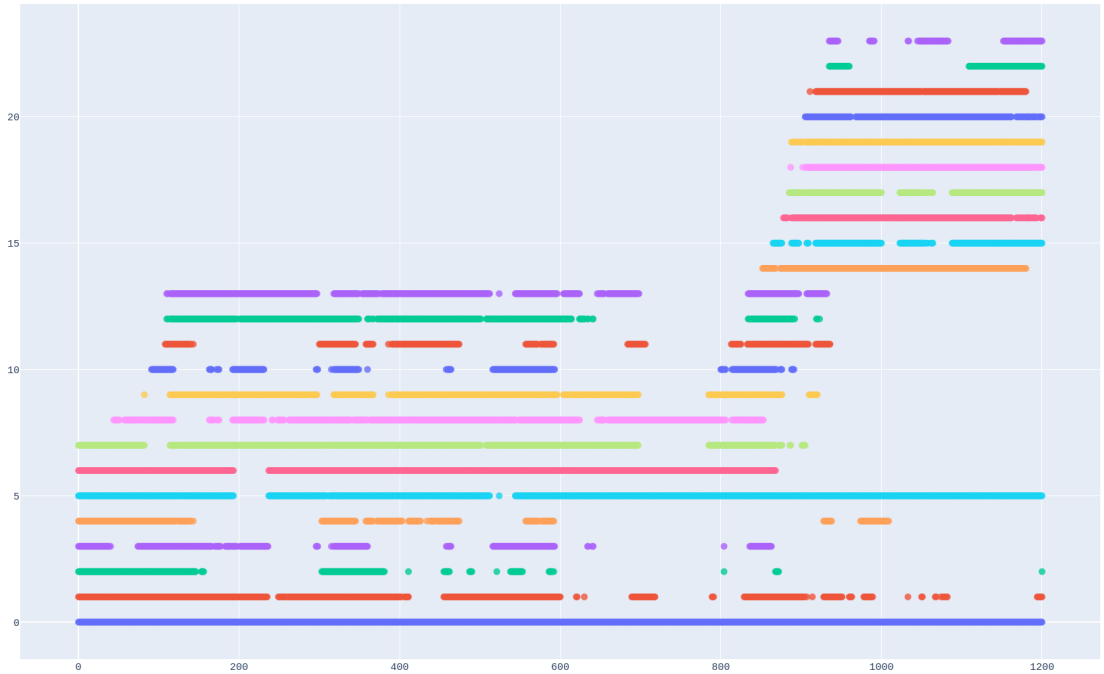


Figure 4.6. Visualization of the ground truth tracks for sequence 1min.

We can distinguish between two players' shifts in the ground truth image. Eight players are detected in the first frame and then six more before the frame 200. These 14 players (5 players for each team, two goalies and two referees) form the first shift of the sequence. Around frame 850, we see ten more players appearing - a new shift coming to the game. Between frames 800 and 1000 is the trickiest part of the sequence when both shifts are present on the pitch. The last thing to mention is four tracks

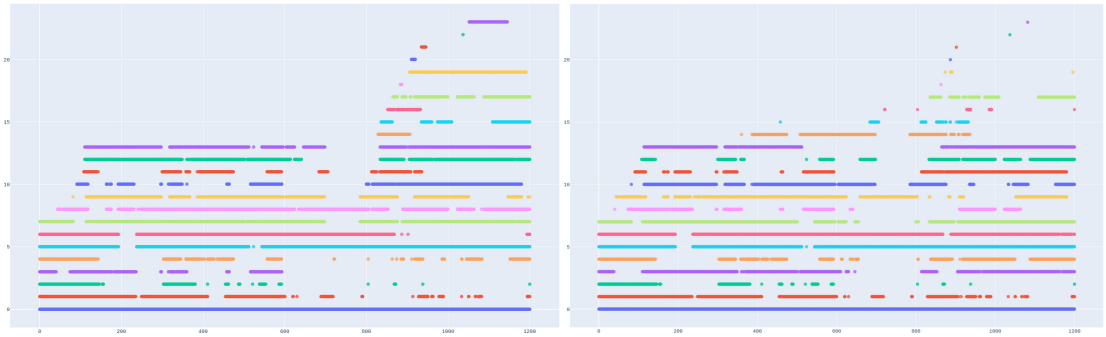


Figure 4.7. Visualization of the clustering based on datasets Ours+Market-1501 (left) and Market-1501 (right).

spanning the whole sequence at the bottom of the graph. The lowest track in dark blue is a goalie detected throughout the sequence. This track consists of only one tracklet and cannot be merged with anything else while enforcing time constraints. Therefore, all experiments will have this one player correctly tracked. The other goalie is the dark green track, third from bottom. As the game flows around him, he is not detected in large part of the sequence. The last two long tracklets are two referees - the second and sixth from the bottom in red and light blue, respectively. In all experiments on the 1min sequence, we would like to see a similar structure.

Now we can qualitatively compare models from the experiment. The image 4.7 visualizes the Market-1501 and Ours+Market-1501 models. Although both models force tracklets to continue even after the shifts change, the expected structure is more evident in the left image (Ours+Market-1501). The right image shows most long tracks with several short ones as we described from the quantitative results.

We show only two visualizations as the DukeMTMC-reID and Ours models are similar to the Market-1501 and Ours+Market-1501 ones, respectively.

4.4.3. Distance function performance

The previous experiment's result poses the question of whether the similarity metric improves the tracking and how significantly. We propose the results of two experiments to vindicate its usage in the final pipeline.

Table 4.5 shows the comparison of tracking with different parts of the distance function defined in chapter 3.7.2. The first column is tracking with time constraints but without the similarity metric. Conversely, the second column shows results of tracking without time constraints while using only the similarity metric. And finally, the last column is the usage of both parts of the distance function.

The first and most important result is that a combination of both metrics brings the best results. Using only time constraints results in the worst numbers in all metrics. Interestingly, using only the similarity metrics results in fewer ID switches. Ignoring time constraints produces much longer tracks. It is because the algorithm can cluster two most similar players even when appearing simultaneously. This bug produces several long tracks with few ID switches but more false positives.

Visualization comparison in images 4.8 and 4.9 confirms our conclusions. The right image depictingmarket-1501 clustering without time constraints shows a few long tracks and many small ones. The issue is not obvious from false positive and ID switches metrics only, but we can see it from the distribution of track lengths.

Distance function		Time	Similarity	Both
Number of tracks		24	24	24
FP		11	16	7
ID switches		90	22	25
Length	med	10.00	1.00	8.00
	avg	10.58	10.58	10.58
	max	34.00	171.00	33.00
Relative coverage	med	1.07	0.54	1.23
	avg	1.05	0.46	1.20
	max	1.51	1.00	2.13
Absolute coverage	med	0.36	0.05	0.32
	avg	0.43	0.15	0.43
	max	1.00	1.00	1.00

Table 4.5. Comparison of tracking in the 1min sequence when clustering with different distance functions. The model was trained on the Ours+Market-1501 dataset.

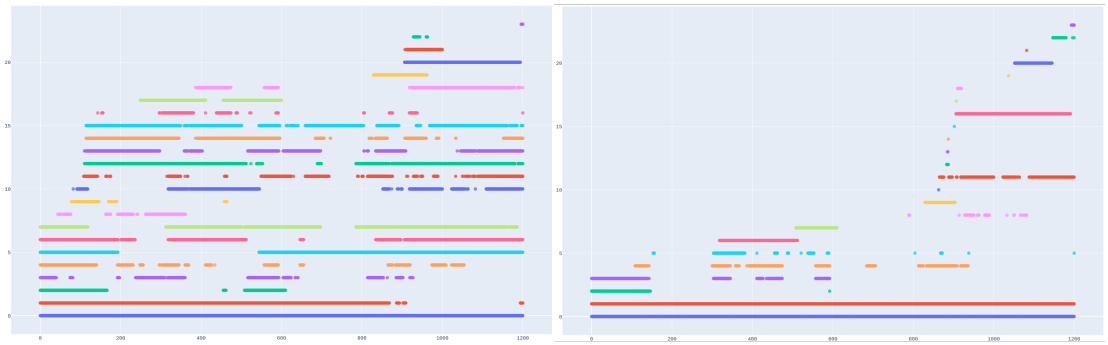


Figure 4.8. Visualization of the clustered tracks using the time metric (left) and similarity metric (right)

4.4.4. Clustering space and short tracklets

When developing the algorithm, we visualized similarity embeddings in the 2D using the Self Organizing Nebulous Growths (SONG) [14]. We noticed visible structures for two teams and four players (goalies, referees). This experiment shows clustering after the embedding vector is reduced to 2D using the SONG. It brings computational overhead for computing the SONG kernel, but we can transform new detections quickly once the kernel is available. The 2D vectors are smaller in memory, and all other operations are quicker than the entire 2048 vectors.

Table 4.6 provides the results of the experiment in column SONG. The performance is worse than clustering in the original space, and we do not recommend using dimensionality reduction techniques unless speed is the primary concern.

When analyzing errors of the multi-camera detector and tracklets creation, we deduced that all false-positive tracklets consist only of one false-positive detection. We can eliminate them by ignoring all tracklets of length one. The experiment is also in the table 4.6.

Again, we can observe the deterioration of the results. It can be caused by the evaluation method, which does not ignore eliminated tracklets and evaluates them as ID switches explaining the significant increase in the metric. When looking at the visualization in image 4.10 (left), we notice that the results are similar to the Full

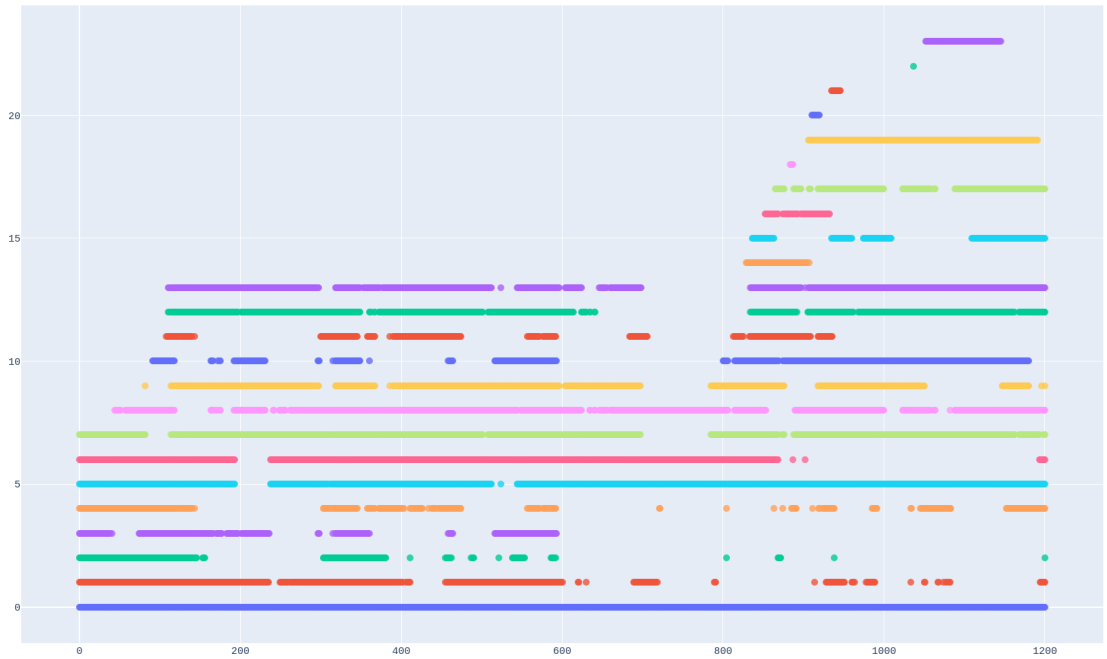


Figure 4.9. Visualization of the clustered tracks using the full distance function with both similarity metric and time constraints.

version.

We did not use the short tracklets elimination in the final pipeline, but it remains possible when the long and most visible tracklets are the primary focus.

The right part of the image 4.10 visualizes the clustering in the compact space using SONG.

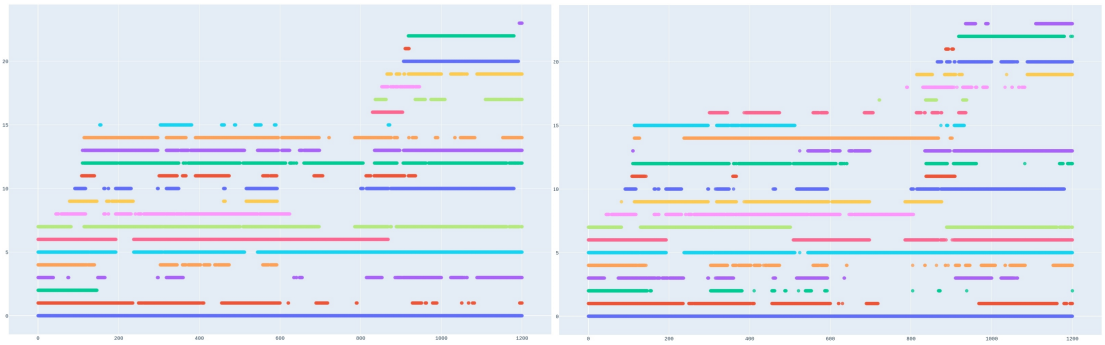


Figure 4.10. Visualization of the clustered tracks when removing short tracklets (left) and with clustering in the compact space using SONG (right)

4.4.5. Multiple shifts learning

The penultimate experiment investigates the problem of tracking on a larger scale. As described in chapter 3.7.6, our idea is to train the identification network on three distinct shifts and then use the network for period-wide clustering. We manually selected three shifts from the second period of the match, ran the clustering algorithm with Frame-based Triplet Loss separately and created a dataset with known identities. We trained the identification network on the dataset with Centroids Triplet Loss and used it for

Clustering space		Remove short tracklets	SONG	Full
Number of tracks		24	24	24
FP		9	8	7
ID switches		35	32	25
Length	med	8.00	8.00	8.00
	avg	8.58	10.58	10.58
	max	23.0	33.0	33.0
Relative coverage	med	1.00	1.00	1.23
	avg	1.09	1.08	1.20
	max	2.12	2.13	2.13
Absolute coverage	med	0.28	0.39	0.32
	avg	0.41	0.41	0.43
	max	1.00	1.00	1.00

Table 4.6. Comparison of tracking in the 1min sequence with different modifications. The first column is clustering when ignoring tracklets of length one, and the second column is clustering in the space reduced by the SONG. [14]

tracking in the first period of the match. For clarity, the diagram 3.15 in the chapter 3.7.6 visualizes the process.

We evaluated the final CTL-learned network on the three sequences from the first period - 1shift, 1minute and 1period. The 1period sequence contains the 1minute sequence, which in turn contains the 1shift sequence. The results in table 4.7, therefore, compare the algorithm's effectiveness with the growing length of the sequence.

Sequence		1shift	1min	1period
Number of players		14	24	34
Number of tracks		14	24	54
FP percentage		0.00	0.28	0.37
ID switches percentage		0.11	0.13	0.04
Absolute coverage	med	0.69	0.28	0.15
	avg	0.65	0.41	0.17
	max	1.00	1.00	0.97

Table 4.7. Results of tracking on different sequences. To compare sequences of different lengths, we use normalized metrics. See the text for more details.

To compare different sequences, we use relative metrics and ratios. Instead of the number of false positives, we provide the percentage of FP (ratio of FP tracks to all resulting tracks). Similarly, we changed the number of ID switches by the percentage of possible ID switches. The ID switch is possible for each tracklet, so the number is the ratio of ID switches to all tracklets.

The annotation is available for the two short sequences which were evaluated empirically. The 1period sequence is too long to annotate and was evaluated statistically similarly to detectors and tracklets creation. Details of the annotation are described in chapter 2.4.

The first uncommon number is the number of tracks in the 1period sequence. Although we set the threshold to the number of players (34), the algorithm could not merge more than 54 tracks due to time constraints. It means that some merges had to be incorrect, causing conflicts in the later clustering phases. When inspecting the issue, we noticed that the original three shifts were also incorrectly clustered as there

are more tracks than players. Unfortunately, we do not have ground truth for the individual shifts, and we can analyze only visualizations in the image 4.11. Apart from the higher number of tracks, we did not find anything suspicious.

Figure 4.11. Visualization of the clustered tracks for the 3 shifts used in the Multiple Shifts experiment

We tried to set up a distance threshold for the clustering in addition to the number of players. The idea is to increase the number of tracks to lower the percentage of ID switches. The approach will work if the time conflicts causing merges occur at the end of the clustering. Results are in table 4.8, where no distance threshold is marked as NaN.

Distance threshold		15	30	NaN
Number of tracks		571	54	54
FP percentage		0.94	0.37	0.37
ID switches percentage		0.04	0.04	0.04
Absolute coverage	med	0.00	0.15	0.15
	avg	0.02	0.17	0.17
	max	0.97	0.97	0.97

Table 4.8. Results of tracking with thresholding the maximum clustering distance. Notice that lowering the threshold only decreases the quality of the tracking. The NaN value stands for no threshold applied.

We infer from the results that the complicated merges do not occur at the end of the clustering. The method must merge two similar players playing in different shifts. Interestingly, the sequence contains identical twins, but a brief visual analysis shows that their tracks are not merged. Perhaps it is because they play in the same shift, but we cannot back up the claim with data.

4.4.6. Transfer learning

The last experiment verifies the generalization of the identification network. We trained the network on the sequence from one match (women's match, different jersey colours) and evaluated it on the 1min sequence from the U19 game. Table 4.9 provides results for the 1min sequence and the shorter 1shift sequence.

The first column depicts the original approach (evaluation on the same sequence as training) for each sequence, and the second one represents transfer learning. We see that the network generalizes poorly on a different game. The results in the last column are comparable to results from the first experiment in this chapter (table 4.4). The transferred knowledge brings the same results as the model trained on the Market-1501 dataset. Since our method fine-tunes the Market-1501 model, we assume that training on the WA sequence did not improve the original model. This can be connected to the fact that the training sequence from the WA match is also incorrectly clustered, as we

Sequence		1shift		1min	
		U19 → U19	WA → U19	U19 → U19	WA → U19
Number of players		14	14	24	24
Number of tracks		14	15	24	24
FP		0	4	7	12
ID switches		4	14	25	43
Length	med	12.50	12.00	8.00	9.00
	avg	11.29	10.53	10.58	10.58
	max	23.00	19.00	33.00	27.00
Relative coverage	med	0.99	1.00	1.23	1.12
	avg	0.99	0.98	1.20	1.17
	max	1.00	1.22	2.13	2.13
Absolute coverage	med	0.69	0.61	0.32	0.28
	avg	0.65	0.61	0.43	0.41
	max	1.00	1.00	1.00	1.00

Table 4.9. Results of the transfer learning. We show original tracking (trained on the same game as tested) and tracking with transferred knowledge (trained on a different match than tested) for each sequence.

can see in the images 4.12 and 4.13. The WA sequence captures two shifts with one change which is not evident from tracks. Again, we can not verify the clustering quality since we do not have the ground truth for the WA sequence.

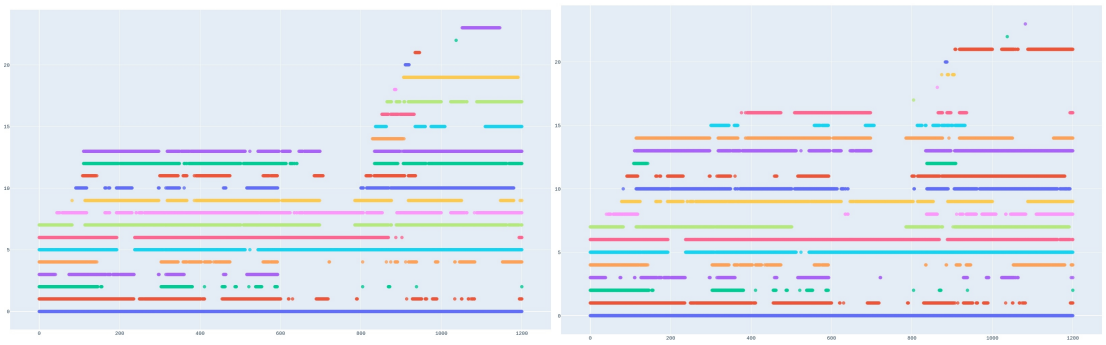


Figure 4.12. TBD: Visualization of the clustered tracks when model trained on the U19 1min sequence (left) and model trained on different match (right)

4.5. Experiments summary

The experiments proved that the method works in limited situations. Two main challenges are detecting players through heavy occlusion and identifying the re-detected player. Although we tested these parts of the pipeline separately, both issues are linked as a reliable identification network would help resolve clusters of players in multi-camera detection.

We ignored clusters of players in the multi-camera detection tasks resulting in many short tracklets in the next step. The biggest problem of the multi-camera detector is the vast amount of false negatives, which remain open for further research.

The tracklets clustering steps work reliably on the same sequence. We could not generalize the network, and the off-the-shelf models perform equally to our generalized

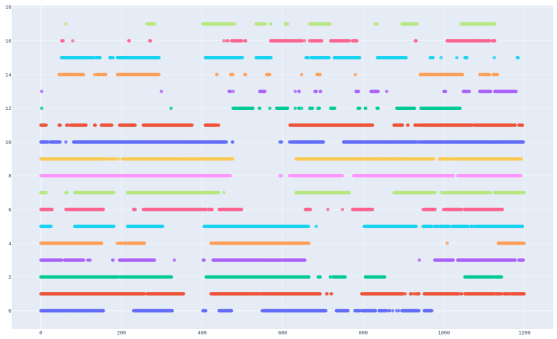


Figure 4.13. Visualization of the clustered tracks in the WA sequence. Even though the sequence captures two shifts, we cannot distinguish them from the graph.

model. Interestingly, the model trained on a public dataset improved the tracking compared to the clustering with time constraints only.

4.5.1. Speed

We did not mention the speed of the algorithm yet. As mentioned in the chapter 1.2, the ideal system would run in real-time to give results during the match. The proposed method can not run this fast as the last step (tracklets clustering) requires all tracklets before starting the algorithm and is therefore unsuitable for direct deployment in a match.

One modification allowing near real-time execution would be training the general network on manually selected shifts from the beginning of the match, as we did in the experiment in chapter 4.4.5. After initial training, the system would not cluster tracklets as we did but assigned an identity to the tracklet based on the nearest neighbours in the encoding space. We can still apply time constraints during the assignment. The initial training should be only short for the system to work during the match. Our experiments showed that the network learns basic team-level recognition in a couple of epochs, so this approach looks promising. However, we did not test this approach.

Step	1min	1period
Single-camera detection (per camera)	10 fps	10 fps
Geometric verifiction and tracklets creation when using 4 cameras	19 fps	17 fps
Tracks creation	33 fps	3 fps

Table 4.10. Speed of individual pipeline steps for sequences 1min and 1period

The table 4.10 shows the speed of individual steps in the pipeline for the 1min and 1period sequences. Provided values are not precise as the performance depends on the used hardware, primarily the amount of memory. The table is only evidence that all but the last step can, in theory, run in real-time. The slow decline in speed for geometric verification is caused by the implementation and processing of big tables. We must highlight that the agglomerative clustering does not scale well and is not recommended for longer sequences.

5. Implementation

This chapter quickly mentions libraries and functions used in the implementation. We justify the selection of some approaches and clarify them. For details, see the enclosed code.

5.1. Geometry

All geometric computations stand upon the OpenCV library. We use it for homography estimation, camera calibration and visualizations. The most peculiar part is the estimation of the barrel distortion. The OpenCV, since version 3.0, offers a model designed for fisheye cameras. Ironically, we achieved better results with the fisheye model used for our non-fisheye cameras while using the standard camera model for the fisheye cameras. The selection of the model is hard-coded in the algorithm.

5.2. Annotations

All data outputs are stored in the CSV format in our custom format. We use the CSV as it is memory efficient and machine-readable. In the app, the data are represented as Pandas [38] Dataframe for its easy manipulation. The Pandas are suitable for relatively small tables, and experiments on the 30-minutes long sequence caused noticeable slowdowns. For 2-hour videos of one game, we recommend considering using more efficient libraries like Vaex [4].

5.3. Embeddings

The most challenging part regarding memory efficiency is storing and working with detections embeddings. We compute the embedding of length 2048 for each detection in the sequence. The 30-minutes long video contained more than 700 000 detections, and the Numpy [28] file was almost 8GB. After decreasing the substantial false-negative rate in multi-camera detection, a better detector could easily reach numbers above 1 000 000. The standard computer cannot operate with such big files, and we used the *mmap* feature of the Numpy library to save the embeddings on the disk instead of the memory. The approach solved the problem so we could conduct our experiments, but it would not scale well for longer videos. The only solution we see is to compute more compact tracklets embeddings or select only several detections as representatives of the tracklet.

5.4. Algorithms

Here we cite libraries used for various algorithms. Details of each algorithm are available in the documentation of the corresponding library. The Hungarian algorithm [19] is from the SciPy [35] library. The SciPy also provides a function for hierarchical clustering used in the multi-camera detector to cluster single-camera detections. We did not use

any off-the-shelf function for tracklets clustering to control the process and preserve original IDs for evaluation.

5.5. Visualizations

To visualize our results in interactive plots, we used the Plotly [17] library. All videos are processed or created by the FFMPEG [10]. Dimensionality reduction techniques T-SNE and SONG are from respective libraries Scikit-Learn [29] and SONG [14].

6. Conclusion

We proposed a new method for unsupervised tracking from multi-view sports videos. The approach ignores crowds and has a high false-negative rate in the detection phase. The modularity of the system ensures easy modifications of individual parts.

The most significant contribution is the unsupervised tracklets clustering using the similarity metric and time constraints. To train the similarity metric with unlabelled data, we propose a new method for hard-negative mining.

The player re-detection and re-identification rely heavily on the quality of the detector. With a sufficient amount of frames with a low false-negative rate, we can track players in one change with few ID changes. Connecting tracklets between shifts is challenging, and our system can perform it under ideal conditions. Most issues arise from two similar players playing the same position in two consecutive shifts.

We show that the system generalizes well in the same match but does not transfer learned knowledge between games.

There are still steps missing for the deployment of the system. The time synchronization and geometric calibration are only semi-automatic processes and are laborious. Further, they require staff with knowledge of computer vision techniques. Missing detections from crowded areas are causing fragmented tracks and low coverage.

We also collected a new dataset of multi-view floorball videos with light flashes for synchronization. The tracking app, visualizations, and short sequences from the dataset are enclosed in the attached CD. The full dataset will be available offline in the Center for Machine Perception.

6.1. Future research

Throughout the thesis, we referenced many possible directions of the subsequent research. The area of automated sports analysis is rich and relatively new in computer vision, and it offers many unexplored areas.

With automatic pitch detection, we could automatically estimate the barrel distortion and homographies. The most challenging task here is time synchronization, which could be one possible direction of future research.

Using the top-view camera detections and information of single-camera trackers could resolve the problem of crowded areas. With a lower false-negative rate of the detector, tracklets would be less fragmented. Tracking data from individual cameras would bring more information to the identification process.

Also, representing the tracklet by the centroid of all its detections seems suboptimal. Each tracklet contains several significant detections characterizing the player uniquely, like the visible jersey number. Automatically selecting the representation of the tracklet is another possible research direction.

The last suggestion is employing the proposed method for labelled data generation. The system can generate many tracks from multiple matches. A human annotator can filter identity-incoherent ones, and the resulting dataset would be suitable for the standard Triplet Loss training. We believe it is possible to train the general players' identification network from tracks from multiple matches.

Bibliography

[1] Luis Alvarez, Luis Gomez, and J. Rafael Sendra. Algebraic Lens Distortion Model Estimation. *Image Processing On Line*, 1:1–10, 2010. https://doi.org/10.5201/ipol.2010.ags-alde. 10

[2] Thulasya Banoth, Mohammad Farukh Hashmi, Zong Woo Geem, and Neeraj Bokde. Deepplayer-track: Player and referee tracking with jersey color recognition in soccer. *IEEE Access*, 10:1–1, 01 2022. 10

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 21

[4] Maarten A. Breddels and Jovan Veljanoski. Vaex: big data exploration in the era of gaia. *Astronomy & Astrophysics*, 618:A13, oct 2018. 52

[5] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking, 2022. 10

[6] Alvin Chan, Martin D. Levine, and Mehrsan Javan. Player identification in hockey broadcast videos. *Expert Systems with Applications*, 165:113891, mar 2021. 10

[7] Anthony Cioppa, Silvio Giancola, Adrien Deliege, Le Kang, Xin Zhou, Zhiyu Cheng, Bernard Ghanem, and Marc Van Droogenbroeck. Soccernet-tracking: Multiple object tracking dataset and benchmark in soccer videos, 2022. 9

[8] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, jun 2010. 11

[9] Na Feng, Zikai Song, Junqing Yu, Yi-Ping Phoebe Chen, Yizhu Zhao, Yunfeng He, and Tao Guan. Sset: a dataset for shot segmentation, event detection, player tracking in soccer videos. *Multimedia Tools and Applications*, pages 1 – 22, 2020. 9

[10] FFmpeg Developers. ffmpeg tool (Version be1d324) [Software]. https://ffmpeg.org. Online; Accessed: October 2019. 12, 53

[11] Xubo Fu, Kun Zhang, Changgang Wang, and Chao Fan. Multiple player tracking in basketball court videos. *J. Real-Time Image Process.*, 17(6):1811–1828, dec 2020. 10

[12] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021, 2021. 10, 19

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 10

[14] Saman Halgamuge. Self organizing nebulous growths (song), 12 2019. 6, 46, 48, 53

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 10, 31

[16] Yueh-Cheng Huang, Chin-Wei Liu, and Jen-Hui Chuang. Using fisheye camera for cost-effective multi-view people localization. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3248–3252, 2021. 10

[17] Plotly Technologies Inc. Collaborative data science, 2015. 53

[18] Ali Karimi, Ramin Toosi, and Mohammad Ali Akhaee. Soccer event detection using deep learning, 2021. 10

[19] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955. 28, 52

[20] Zuzana Kukelova, Jan Heller, Martin Bujnak, and Tomas Pajdla. Radial distortion homography. pages 639–647, 06 2015. 10

[21] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*, April 2015. arXiv: 1504.01942. 10, 19

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 11

[23] Long Liu. Objects detection toward complicated high remote basketball sports by leveraging deep cnn architecture. *Future Generation Computer Systems*, 119:31–36, 2021. 10

[24] Abhinav Moudgil and Vineet Gandhi. Long-term visual object tracking benchmark, 2017. 10

[25] Banoth Thulasya Naik, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. A comprehensive review of computer vision in sports: Open issues, future trends and research directions. *Applied Sciences*, 12(9):4429, apr 2022. 4, 9, 10

[26] National Coordination Office for Space-Based Positioning, Navigation, and Timing. GPS Accuracy. https://www.gps.gov/systems/gps/performance/accuracy/. Online; Accessed: March 2020. 9

[27] Xiaohan Nie, Shixing Chen, and Raffay Hamid. A robust and efficient framework for sports-field registration. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1935–1943, 2021. 10, 22

[28] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. Online; Accessed April 2020. 52

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 53

[30] Miroslav Purkrábek. Floorball player tracking from a top-view camera, 2020. 7, 16, 21, 36

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 10

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015. 10

[33] Ergys Ristani, Francesco Solera, Roger S. Zou, R. Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV Workshops*, 2016. 6, 10, 43

[34] Kanav Vats, Mehrnaz Fani, David A. Clausi, and John Zelek. Puck localization and multi-task event recognition in broadcast hockey videos, 2021. 10

[35] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 25, 28, 52

[36] Tsaipei Wang, Chih-Hao Liao, Li-Hsuan Hsieh, Arvin Wen Tsui, and Hsin-Chien Huang. People detection and tracking using a fisheye camera network. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–5, 2021. 10

[37] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, June 2009. 30

[38] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. 52

[39] Mikolaj Wieczorek, Barbara Rychalska, and Jacek Dabrowski. On the unreasonable effectiveness of centroids in image retrieval, 2021. 10, 31, 43

[40] Hao Wu, Xinxiang Zhang, Brett Story, and Dinesh Rajan. Accurate vehicle detection using multi-camera data fusion and machine learning. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3767–3771, 2019. 10

[41] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. pages 2411–2418, 06 2013. 10

[42] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022. 10

[43] Kailai Zhang, Ji Wu, Xiaofeng Tong, and Yumeng Wang. An automatic multi-camera-based event extraction system for real soccer videos. 23(2):953–965, may 2020. 10

[44] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box, 2021. 10, 19

[45] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015. 6, 10, 31, 43

[46] Awesome Sports Camera Calibration. https://github.com/cemunds/awesome-sports-camera-calibration#real-time-camera-pose-tracking. Online; Accessed: May 2022. 10

[47] iSportAnalsis website. https://www.isportsanalysis.com/. Online; Accessed: May 2022. 8

[48] Kinexon, player tracking. https://kinexon.com/technology/player-tracking/. Online; Accessed: May 2022. 9

[49] Navigine website. https://navigine.com/open-source/. Online; Accessed: May 2022. 9

[50] Opta website. https://www.statsperform.com/opta/. Online; Accessed: May 2022. 8

[51] Papers with code. https://paperswithcode.com/. Online; Accessed: May 2022. 19

[52] SecondSpectrum website. https://www.secondspectrum.com/index.html. Online; Accessed: May 2022. 8

[53] Unihockey PNG 4 (silhouette of the player). https://pngimage.net/unihockey-png-4/. Online; Accessed: May 2022. 4, 24

A. Contents of the attached CD

purkrmir-Master_Thesis.pdf	the text of this thesis
tracking_app/	folder with the code
tracking_app_data/	folder with example sequences and results