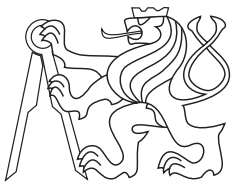
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

BACHELOR THESIS

ISSN 1213-2365

Floorball Player Tracking from a Top-View Camera

Miroslav Purkrábek

purkrmir@fel.cvut.cz

May 22, 2020

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

Bachelor Thesis of CMP, Czech Technical University in Prague,

Published by

Center for Machine Perception, Department of Cybernetics
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

Floorball Player Tracking from a Top-View Camera

Miroslav Purkrábek

May 22, 2020

BACHELOR'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 Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Floorball Player Tracking from a Top-View Camera

Bachelor's thesis title in Czech:

Sledování hráčů florbalu ve videu z horního pohledu

Guidelines:

1. Review current methods for player tracking, focus on those successfully applied to top-view camera.
2. Propose and implement a system for player detection and tracking.
3. Prepare data for training and evaluation of the method.
4. Evaluate the method and compare it to a baseline.

Bibliography / sources:

[1] Redmon Joseph, YOLOv3: An Incremental Improvement, 2018, arXiv:1804.02767
[2] Goodfellow and al, Deep Learning, MIT Press, 2016

Name and workplace of bachelor's thesis supervisor:

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

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **11.02.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

prof. Ing. Jiří Matas, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Author statement for undergraduate 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 useful critiques of this thesis. I also wish to thank my parents for their support and encouragement throughout my study. Moreover, I greatly appreciate the help in writing the thesis provided by my sister.

Abstract

In this thesis, we propose a new system for online person tracking by detection. The system focuses on tracking persons in top-view sports videos. A modified YOLO network serves as a person detector from top-view videos. A fine-tuned ResNet network works as a similarity metric allowing person reidentification by visual similarity. The proposed method achieves 94% mAP in the detection and 90% MOTA in the tracking on a new dataset from sports videos. The dataset is attached to the thesis along with a GUI application for data processing.

Keywords

online person tracking, person detection, person identification, person reidentification, player tracking, ResNet, sports videos analysis, top-view tracking, tracking by detection, YOLO

Abstrakt

Tato práce představuje nový sytém pro sledování osob ve sportovních videích. Systém se zaměřuje na sledování osob při pohledu shora. Pro detekci slouží upravená síť YOLO, dotrénovaná síť ResNet slouží pro porovnání podobnosti hráčů. Za pomoci vizuální podobnosti systém rozpoznává ztracené hráče. Navržená metoda dosahuje 94% mAP při detekci a 90% MOTA při sledování osob na novém datasetu ze sportovních videí. Dataset spolu s grafickou aplikací na zpracování dat je přiložen k práci.

Klíčová slova

analýza sportovních videí, online sledování osob, opakovaná identifikace osob, ResNet, rozpoznávání osob, sledování hráčů, sledování osob, sledování pomocí detekce, sledování z vrchního pohledu,YOLO

Contents

Abbreviations

AP average precision. 28, 29

AUC the area under the curve. 29

DCNN deep convolutional neural network. 12

DNN deep neural network. 21, 23

FN False negative. 28

FP False positive. 28

GPS global positioning system. 4

GUI graphic user interface. 8

ID identity. 18

IDP identification precision. 29

IDR identification recall. 29

IFF international floorball federation. 6

IoU intersection over union. 28

LDA linear discriminant analysis. 22

mAP mean average precision. 28

MOTA multiple object tracking accuracy. 30

MOTP multiple object tracking precison. 30

NBA The National Basketball Association. 4

NMS non-maxima supression. 14

OCR optical character recognition. 19

SSL Svenska Superliga; Swedish Super League. 7

TN True negative. 28

TP True positive. 28

WFC World Floorball Championship. 7

WFC-18 World Floorball Championship in Prague in 2018. 6

YOLO You Only Look Once. 12

1. Introduction

The use of modern technologies in all kinds of sports is growing. Statistics about each player's physical condition (distance run, maximum speed) through an individual's technical abilities (shooting stats, hold of the ball) to even team statistics are widely used in football, basketball, hockey, baseball, and other sports. Empirical data has a significant impact on team results and individual athlete performance if adequately applied to the training process. From the players' positions, we can also derive statistics useful for coaches. These are not only data about the physical condition but errors in tactics and various game patterns too.

We introduce an online person tracking system for top-view sports videos. The method tracks all players, goalies, and referees simultaneously with information from previous frames. With the program running in real-time, we could analyze a team's strategy during the match and adjust the tactic. As there are too many occlusions from side-view cameras, we use a ceiling-mounted top-view camera.

The system applies tracking by detection. A modified YOLO network acts as a top-view person detector, and a fine-tuned ResNet network works as a similarity measure. The memory in the tracking algorithm allows for person reidentification later in the match. We created a new dataset from floorball videos to train and test the method and a GUI app for data processing.

The rest of this chapter reviews the current methods used in sports and especially floorball. It shows the usage of data and the advantages it brings to coaches. The *Data* chapter presents data used in this thesis, along with the application created. The *Method* chapter explains parts of the proposed method and solutions to the problems faced. The proposed experiments test the method quality on a new dataset. The method gives competent results in person detection and person tracking in a highly constrained sequence. The implementation details and code used are mentioned in the last chapter.

1.1. Current work

Currently, most data about players are collected either manually or from sensors. Individual athletes use tracking methods based on position sensors like GPS or an indoor tracking system. GPS trackers are relatively cheap, but its precision is limited [15]. While indoor tracking systems are more precise than GPS, they are also more expensive. Both approaches are problematic about player comfort and, therefore, concentration on the performance. Nevertheless, these methods are the most common for players as they are simpler to set up than a more sophisticated tracking system.

Big teams and leagues use services of companies for player tracking and analysis. In each NBA hall, at least 13 cameras record the whole match since 2014. The NBA contracts a company [16] (first SportsVu, then Second Spectrum) to deal with data from videos. Premier League hires Opta [36] for the same reason. All businesses are secretive about their methods as well as data, so it is not clear how they process the videos.

In the Czech Republic, floorball teams have to collect a great deal of statistics about

each match in higher leagues. It usually takes four people to record information about shots and shift changes for each match. More people then transcribe hand-written notes to the electronic form. Czech Floorball experimented with indoor locators sewn into jerseys to collect statistics but it did not proceed further as the jerseys were too expensive. However, all teams in the top two men leagues and the top women league have to record each match on the video. The standards make it possible to extract statistics from videos and process them automatically.

Several computer vision tracking methods have been applied to the problem of sports video analysis. IoU tracker [3] depends only on the position of detections, Kalman filter, and Hungarian algorithm. Parson and Rogers [19] use traditional computer vision techniques for player detection and team assignment.

Other approaches are neural networks combining positional information along with image encoding. The original Siamese tracker [1] as well as SiamDW [29] profit from the siamese network architecture for image encoding. These networks require big datasets for learning different players. TADT [13] focus on learning features characteristic for the given class, which increases inter-class difference and helps recognize players from the same team. LightTrack [17] bases its tracking on pose estimation, but that shows poor results for top-view. There are also methods with promising results (for example, ATOM [6], DiMP [2]) with online tracking. However, these are solely focused on single object tracking and scaling for 14 persons (10 players, two goalies, two referees) would be computationally expensive, as is the case for [27].

All methods mentioned above aims to track visible players and forgets them once the player disappears for a long time. That is frequent in sports like hockey and floorball, where players change each minute. To avoid this, Yoon at al. [28] use jersey number detection. Numbers are not usually visible from top-view. Senocak and al. [24] presented visual tracking based on different body parts, but it requires enormous learning.

2. Data

In this chapter, we present data used for training, validation, and testing, as well as further experiments mentioned in chapter 4. Since this thesis focuses on floorball videos, we gathered data from floorball games. As mentioned in chapter 1, private companies do not share their data collected in similar team sports like basketball or hockey. Furthermore, most sports videos publicly available are either from side-view or from broadcast camera characterized by frequent view cuts, zooms, and camera movement. While these videos are widely used in TV and popular with viewers, they are also complicated for tracking because of occlusions.

In public datasets used for visual object detection such as PASCAL [7], COCO [14], or Open Images [11] there is a small number of sport-specific data. The datasets contain frames from broadcast cameras focused on a limited section of the playing field. Therefore, we retrieved videos from World Floorball Championship in Prague in 2018 (WFC-18). For each match, we have 4 videos.

One side-view camera records match above the central stand. The camera captures the entire pitch and sees most jersey numbers. As the tracking is done only on top-view videos, this video could be used for individual player identification by jersey number or as help for human annotator.

Two top-view cameras, one for each playing field half, are mounted on a screen hanging above the court. Top-views cameras capture most of the field but for a small stripe along far sides of the field. There is noticeable barrel radial distortion in images from top-view cameras. The view is ideal for tracking as there are not as many occlusions as in side-view videos. The camera's position introduces variations in player appearance as close players look different than those further away. Variations in shape bring confusion to the person model later created by the neural network.

The last camera captures match from above and contains the whole field in one image. This panoramic camera sees the whole playing field resulting in extreme radial distortion. As with the previous view, players differ whether seen bellow camera or in corners of the field.

All videos were provided by the Amden Company ([30]).

2.1. Videos

2.1.1. Technical parameters

A short analysis of videos from WFC-18 determined the use of the two top-views cameras and the side-view camera. We use the top-view cameras for player detection, and tracking and the side-view video serve for player identification during annotation.

Table 2.1 below shows properties of used videos. Videos from WFC-18 are not synchronized.

2.1.2. Representativeness

The WFC-18 data are characteristic for most floorball games in a technical aspect. The pitch, as well as other dimensions, are the same as in all official IFF floorball games,

Video	Video shortcut	Resolution	Framerate	Radial distortion
side-view	4k	3840 × 2160	25	-
top-view	KL	1920 × 1080	25	yes
	KP	1920 × 1080	25	yes

Table 2.1. Properties of used videos

there is the same number of players, referees wear standardized dress. IFF rules also standardize face-off marks and creases lines.

What differs is the playing surface. On WFC, there are only lines and marks used for floorball and a few advertisements. The lines could serve for match initialization by the line detection and further homography estimation. Unfortunately, the playing surface is not standardized yet, so even in Svenska Superliga; Swedish Super League (SSL) matches, various lines cover some playing fields hindering the automation of homography estimation.

The last technical detail worth mentioning is jersey colors. In top state leagues like SSL and Czech Superleague, jerseys are of all colors but gray (forbidden by the rules). However, national teams tend to have similar jersey colors, the most frequent colors being white, red, and blue.

Differences could impede generalization on all floorball matches in future work. Meanwhile, the same problems apply to other hall team sports like handball, basketball, or hockey on the international level, so generalizing to other sports could be without serious problems.

2.2. Floorball Annotator

We made a new tool for video preprocessing and annotation named Floorball Annotator for annotating data from WFC-18. This section describes the principal functionalities. Implementation details and screenshots are mentioned in chapter 5.1.

2.2.1. Frame extraction

The script included in Floorball Annotator extracts frames from videos with given framerate. It accounts for different framerates in side-view and top-views videos so that each camera can have different framerate. As the script extracts five frames per second, this is the minimal framerate for videos as well. Extracted frames are then saved in required directories, ready for annotation.

To start working with the Annotator, follow the installation script. This script installs required programs and libraries and runs first data extraction. The app runs on Linux, tested on a machine with Ubuntu 16.04.

2.2.2. Radial distortion and homography estimation

To annotate players in top-view videos with player identity, we use side-view video to help the user with identification since the side-view video is more suitable for player identification. The app maps points from top-view cameras to side-view using two visual transformations to make the workflow as smooth as possible.

Radial distortion is an image deformation caused by lens curvature. Barrel distortion means that points are displaced from their original position toward the center of the picture. Points that are farther from the center move more than those close. Barrel

radial distortion results in the square looking like a barrel (hence the barrel distortion). Radial distortion is the property of a single image.

Homography is the transformation between two planes. It is used to transform points from one plane (one image) to another. Homography estimation means finding a 3×3 matrix representing the transformation. The homography is the trait of a pair of images.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

To project points from one image to another, we need to estimate radial distortion and homography between views. The preparation script takes face-off marks and corners of the big and small creases marked by the user and estimates homography and radial distortion. Even though the program needs only 4 points for homography estimation, more points serve for better precision. The program returns estimated parameters in the terminal, and the user has to accept and rewrite them by hand to the configuration file. The default parameters are for the WFC-18 data.

Below is the algorithm for the automated radial distortion and homography estimation.

Algorithm 1: Algorithm for radial distortion and homography estimation

```
Result: Estimated homography and radial distortion
mark 12 points in each video;
for C in ∈ (−10^−6, 10^−6) with step 10^−8 do
    compute homography using C as radial distortion parameter;
    compute reprojection error;
    if error < smallest error then
        save homography as best;
        save radial distortion parameter C;
        save error as smallest;
    end
end
```

2.2.3. Annotation process

After preprocessing each video, the GUI tool starts automatically. In Floorball Annotator GUI (image 5.2), the user can see both top-views as well as side-view frames at the same time. It is possible to zoom images, create new annotations, delete wrong annotations, and see corresponding players from top and side view. Annotation consists of one point in the image plane along with player ID. Annotations are saved for each image after a change of frame. It enables more people to work on the same match simultaneously if these people do not annotate the same frame at once.

After the user finishes his task, scripts transferring from Annotator format to YOLO format and back are available for working in the human-in-the-loop process. Thereby, the user can annotate a small number of frames, let the tracking algorithm learn itself from given annotations, and track players for the rest of the match. The user then repairs the wrong annotations and continues in the cycle. The process ensures an efficient way to upgrade the whole algorithm quickly.

2.3. WFC-18 dataset

We annotated a modest dataset for training, validation, and testing. The dataset consists of 260 images from 10 different matches from WFC-18. 13 teams with 5 jersey colors are in annotated images. As videos from WFC-18 are not synchronized, we synchronized videos manually. The dataset, called the WFC-18 dataset, is divided into a train, test, and validation set.

Each player is annotated exactly once for both frames capturing the playing field. If the player is present on both frames, he is annotated only in frames with the corresponding half of the field. We annotated the player position as the projection of his center of gravity to the pitch plane. This definition accounts for standing player (point being center of the line connecting his feet) as well as players in unusual poses.

Match	Period	Shortcut	Number of frames	train/val/test	IDs
FIN-SWE	1	AS3	20	train	no
	1	AS5	10	val	no
	1	KF3	10	train, val	no
	2	AS1	10	test	no
	2	KF1	4	test	no
CZE-GER	1	AS4	10	val	no
	1	AS6	10	train	no
	1	KF4	20	train,val	no
	2	AS2	20	test	no
	2	KF2	6	test	no
	2	AS22	100	test	yes
CZE-SUI	1	-	6	test	no
FIN-DEN	1	-	4	test	no
LAT-GER	1	-	4	test	no
SIN-JAP	1	-	4	test	no
SUI-LAT	1	-	6	test	no
SVK-SIN	1	-	4	test	no
SWE-NOR	1	-	6	test	no
THA-AUS	1	-	6	test	no

Table 2.2. Sequences used in the WFC-18 dataset

A review of the annotated dataset is in table 2.2. There is only one testing sequence for tracking - AS22. It is the only sequence with annotated player IDs; all others were used only for detection testing. All images were hand-picked. Most situations are well-arranged; players are easily separable. Most occlusions are two players from different teams. There are only game situations so that there are always at most 14 persons (5+1 from each team, 2 referees) on the field. Examples from the dataset are in images 2.1.

The training dataset is from two matches — CZE-GER and FIN-SWE. Finns wear white jerseys as well as Germans, Czechs wear red jerseys and Swedes have blue ones. Four classes are in the dataset — team A, team B, goalie, and referee. Whether we annotated a team as team A or team B depends on which half teams start the match — teams starting on the left half are annotated as team A.

Each match was extracted with framerate 5 frames per second. Shortcuts help orient in types of images. *AS* stands for annotated sequence, meaning that images from this

sequence are consecutive. *KF* stands for keyframes, meaning that we selected them as exemplary for a floorball match.

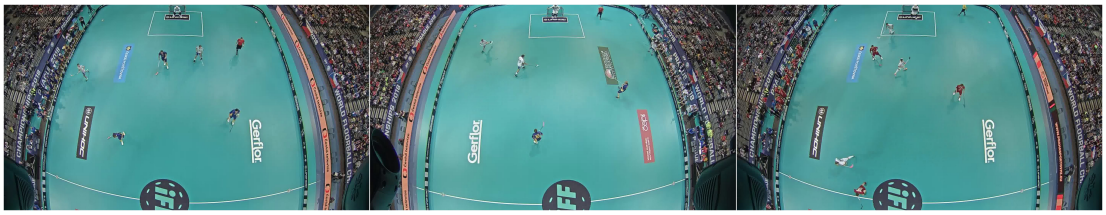


Figure 2.1. Example images from the WFC-18 dataset

3. Method

This chapter introduces a new method for player detection, classification, and identification, describes the algorithm and the issues we faced. All training and experiment were done with the WFC-18 dataset described in chapter 2.3.

3.1. Method overview

The proposed algorithm takes a top-view video sequence of a floorball match. If the field is divided into multiple videos (two in WFC-18 dataset), frames come consecutively. For each video, the algorithm also needs significant points on the court to estimate homography.

The program produces various outputs from tracking data in text files for each frame or in MOT Challenge format, through images with visualized bounding boxes to a projection to the court.

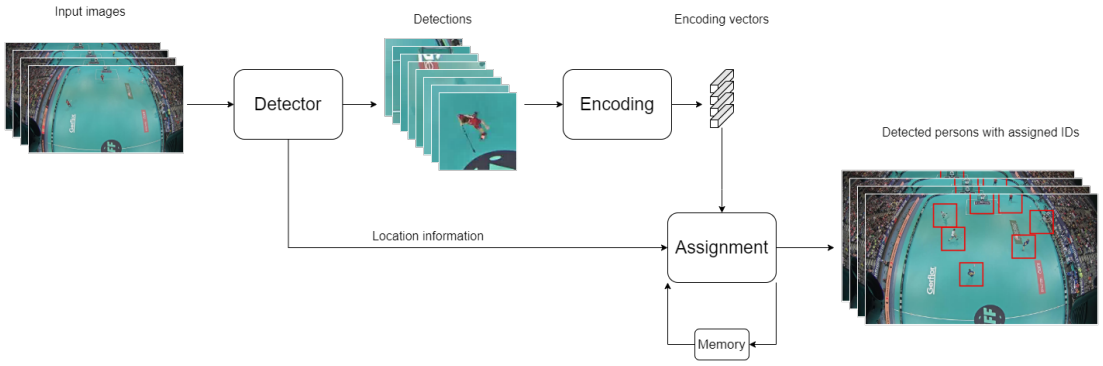


Figure 3.1. Overview of the proposed method

In the picture 3.1, the whole data pipeline is visualized. The program cuts the input video to the sequence of frames. Then it rescales frames to 416x416 required by YOLO. Each image is then run through the YOLO network for player detection. We use YOLO as a region proposal network, so the output is a cropped image of detected players. Each detection gets its ID based on information from the current frame and previous frames in the tracking part.

3.2. Detection

In the detection task, we aim at estimating the location of the object in a given image. The most straightforward approach is to estimate a bounding box, which is the tightest box that contains the detected object. Other approaches could be estimating masks for each object or pose for a human.

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. The situation is even worse in team

sports like floorball. To help with occlusions, we use top-view videos even as they are not as common as broadcast videos. Other frequent problems are variability in detected objects, view angle, and image transformation. As detected objects differ in shape, color, and size, it is harder to model this class. This also applies to a change of angle as human from side-view looks different than from top-view. Image transformation is present to some degree in all images. It could be a transformation of image plane by radial distortion or change of perspective. Objects in the image plane then change shape and size depending on their position in the image.

Figure 3.2. Examples of occlusions in broadcast side-view videos

Currently, deep convolutional neural network (DCNN) is the most used algorithm for visual object detection. Networks take images and predefined classes and return positions of detected objects, usually in the form of the bounding box. In most networks, the detection task is connected with classification. In this thesis, we differ these two as we use DCNN as a region proposal network for further processing.

For the detection part of the system, we chose the You Only Look Once (YOLO) detector in its third version, YOLOv3. The next section shortly explains YOLO and its features. The rest of the chapter addresses the problems we faced.

3.2.1. YOLO

When mentioning YOLO throughout the thesis, it is always YOLOv3. YOLOv3 introduces multi-scale predictions and better overall performance than the previous version. We chose YOLO because of its speed, performance, and its simplicity of use and training. The unofficial Python implementation of YOLO [34] is used for more effortless coding and quicker results. All training described bellow is fine-tuning from pretrained weights from the official YOLOv3 webpage [23]. As a base model, we edited YOLO pretrained on the COCO dataset with 80 classes. In the time of finishing this thesis, the newer version of YOLO (YOLOv4 [4]) was presented.

The following articles sum up features of the YOLO network necessary for this thesis. For a more detailed overview of the YOLO network, see the original paper.

Grid approach

From its name, YOLO process images in one forward run on all scales at the same time. Each image is run through multiple convolutional layers, effectively encoding its content. From this encoding, the YOLO layer estimates bounding boxes' shapes, positions, and each class probability. The network divides an image into smaller cells to speed up the detection process. Each cell then predicts bounding boxes independently.

Therefore, YOLO predicts the vast amount of bounding boxes and then erases most of them in the non-maxima suppression algorithm

After the image runs through convolutional layers, it ends up with encoding for each cell. The encoding describes the content of the part of the image belonging to the cell. The YOLO layer predicts bounding boxes' positions, shapes, and class labels from the encoding. Encoding for each cell is visualized in image 3.3 taken from [32]. The YOLO layer's output is bounding box position and shape, the class probability for each class, and detection objectness. Objectness is a probability that the bounding box contains any object.

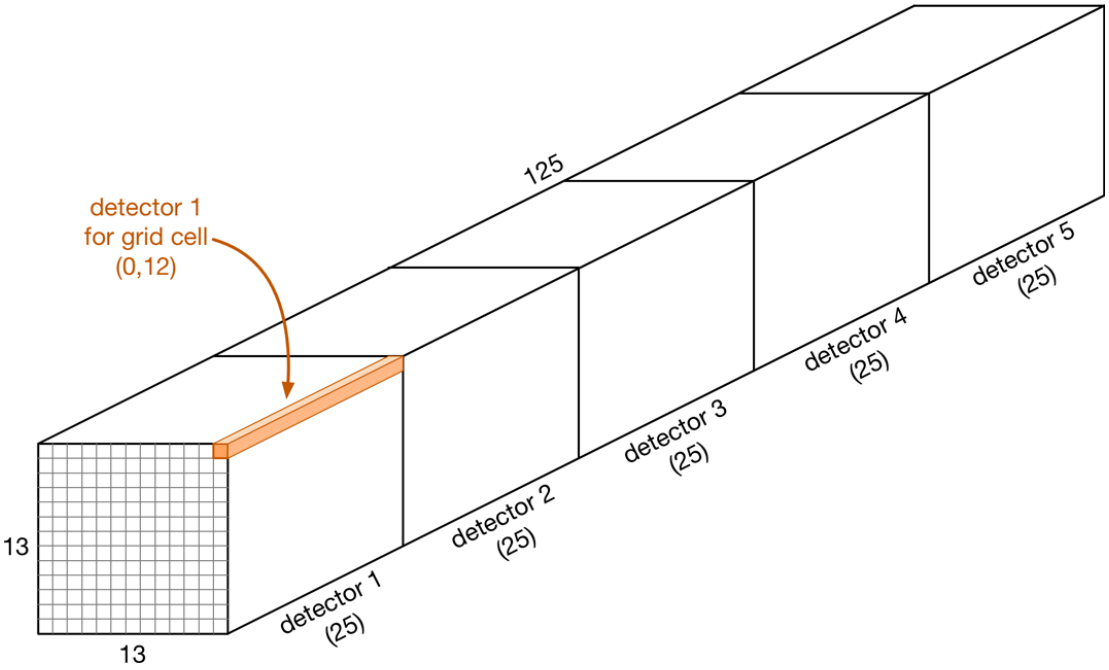


Figure 3.3. Visualization of the YOLO grid cell encoding. The picture taken from [32]

To predict bounding boxes, YOLO uses an approach with anchors. Predefined anchors are something as bounding box templates. Each cell fits multiple anchors to its content and then selects anchors with the best fit. During the learning process, YOLO estimates bounding box shape by editing anchors so that detection does not have to be the exact shape as the anchor. Each cell can output up to three bounding boxes.

YOLOv3 returns bounding boxes on three levels to account for multiple scales. The grid approach explained above is applied to the grid of size 13×13, 26×26, and 52×52. In effect, cells from a 13×13 grid should detect the largest objects, while cells from the finest grid should detect the smallest objects. Since each cell can reshape and resize its bounding box, the size discriminability is not granted as the smallest bounding box from 13×13 grid cell would be similar size as the biggest one from 26×26 grid cell. Encoding on each level has a different length. While in the finest grid encoding is of length 256, in the coarsest, it is 1024.

Table 3.1 shows the sizes of outputs on each level for better clarity. N_{max} stands for the maximal number of detections per given size.

Grid size	encoding length	N_{max} per cell	N_{max} per grid
13 × 13	1 024	3	507
26 × 26	512	3	2 028
52 × 52	256	3	8 112
overall			10 647

Table 3.1. Overview of the number of detections in different YOLO scales

Multilabel, losses

YOLOv3 introduces the possibility of multilabel detections. YOLO classes do not have to be mutually exclusive, and one bounding box can have more than one label. It is possible because instead of softmax above class probabilities, which require the sum of the probabilities to be one, YOLOv3 uses independent logistic classifiers learned with a binary cross-entropy loss function.

Although the YOLO backbone is ready for multilabel ground truth boxes, the program does not account for multilabel annotations. We modified the code for annotations loading, processing, and the non-maxima suppression algorithm.

Non-maxima suppression

Since the non-maxima supression (NMS) algorithm plays a significant role in this thesis, we explain how it is implemented in YOLOv3. Non-maxima suppression is an algorithm suppressing all but maximum values. In the YOLO case, it erases bounding boxes with the same or similar position and label. This algorithm is crucial for proper YOLO function as it suppresses most of the proposed detections.

In the original implementation, if boxes overlap by more than 0.4 IoU and have the same label, only the one with the highest class probability is preserved. It is one of the problems for multilabel predictions as it is not clear when boxes with more than one label match in the label.

3.2.2. Point to bounding box

The first problem we had to address was the ground truth format. YOLO network uses bounding boxes to estimate player positions. Since our goal is to track player location on the playing field, we annotated the position as a 2D point in the image plane. Therefore, we had to transfer point (projection of the center of the mass as described in chapter 2.3) to the bounding box for each player.

As mentioned in chapter 3.2.1, YOLOv3 divide each image to grids to cover different scales and then encode each cell from each grid to predict bounding boxes. Each cell predicts those bounding boxes with the center in the cell. As long as the bounding box center stays in the same cell as player encoding, the bounding box does not have to be tight around the player. We can choose a unified bounding box shape and size for all players and retrain YOLO to predict those boxes instead of tight-fitting ones.

We chose to use squared bounding boxes for simplicity. If two bounding boxes do not overlap, the learning algorithm ignores their distance as IoU equals zero. For the fastest learning, the bounding box has to be as big as possible. But as mentioned above, the bounding box too big would results in the cell encoding not only the player but also its surrounding. When players are too close, the same encoding would belong to both players, making them indistinguishable for further identification. It would also

effectively make detections from finer grids invalid as small cells could not predict big bounding boxes.

The experiment in chapter 4 compares detection performance for different bounding box sizes. In the result of the experiment in image 3.4, we chose to use square bounding boxes of size 150 and 200 for further progress. As we can see from image 3.5, boxes of size 150 and 200 always encompass players.

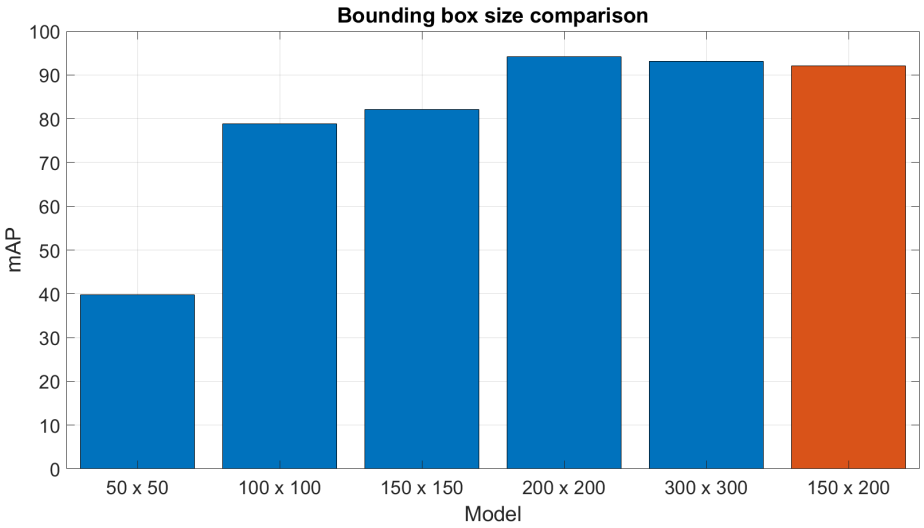


Figure 3.4. Comparison of the detection performance when using different ground truth bounding boxes

3.2.3. Half-line overlap

The next problem we addressed was the fact that the field consists of two images in our dataset. The same player could be detected twice, once in each image, as these images overlap around half-line. We annotated images so that only players on the relevant half are positive detections. The annotations are described in detail in chapter 2.3. The idea was to learn a mask to where to look for players implicitly. This approach worked for field boundaries, but it turns out that it does not detect player on either half losing its detection sometimes.

The most natural solution for this would be to use one camera for the whole court. The camera would introduce even more significant radial distortion and would make players look smaller, resulting in harder detections. Another approach would be to use a hard-coded or variable (detected) mask. The mask would have to be identified for each camera. Or detections could be filtered when connecting two images by a non-maxima suppression algorithm in the real-world coordinates.

3.2.4. Anchors

YOLOv3 uses anchors for estimating bounding box shape. As our bounding boxes are square, YOLO has to choose anchor similar to square and try to fit it to square ground truth during learning. As with half line problem, this approach works very often. Estimated boxes are never square, but since we only need its center for player location, small disturbances are not a problem. Similarly, when YOLO fits the rectangular bounding box instead of square one during detection, it is not a problem for location

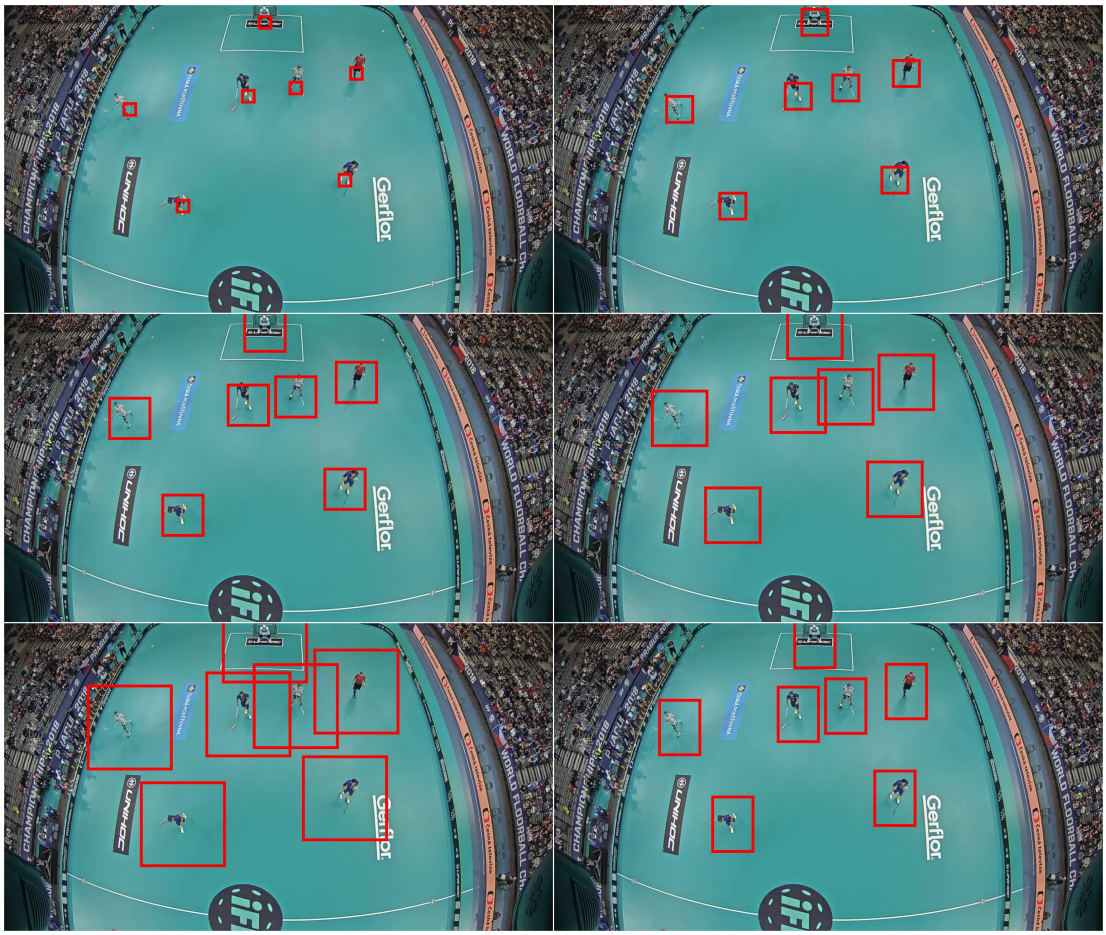


Figure 3.5. Visualization of different bounding boxes' sizes

estimation. However, it is a problem for algorithms using IoU, primarily non-maxima suppression, and performance evaluation.

One way to solve this problem would be a bigger dataset that would retrain YOLO always to use square-like bounding boxes. We tried editing anchors to square values because we don't have dataset this big at the time of writing this thesis. The experiment in chapter 4.2.2 compares detection performance depending on used anchors. From the experiment, it seems that removing anchors have little to none effect on detection performance. The examination of individual images revealed that while double detections caused by rectangular bounding boxes disappeared, new false detections along playing field edge appeared.

3.2.5. Non-maxima supression

More aggressive suppression

We made several enhancements in the non-maxima suppression algorithm. The first alteration is based on knowledge of floorball videos. In top-view videos, players do not tend to be too close to each other. It allows us to suppress detections overlapping for more than a certain threshold. The threshold depends on the position of camera and type of videos. For the WFC-18 dataset, we experimentally chose the threshold 0.6. As our goal is to track players, it is better to lose players for a couple of frames than

have too many false detections. In this thesis, we call this modification of the NMS algorithm *huge overlap NMS*.

Before this modification, it was common for one player to have 2 or 3 bounding boxes. After the adjustment, most of the players have only one bounding box. We also tried to limit the number of players from each team as it is in the game. This was of no help as detection from the same player had higher class probability resulting in losing part of the players and having double detections for others.

Multilabel suppression

Here, we introduce alternations to the NMS algorithm made because of classification. We mention only modifications in the NMS. The next chapter describes the whole classification process.

As explained in chapter 3.2.1, introducing multilabel detections requires a change in the NMS algorithm. Since the code used is not ready for correct multilabel detection (to have a bounding box with more than one label), we chose using more bounding boxes for each person. Details of this approach are explained in chapter 3.3.1.

The NMS algorithm ignores *human* class when suppressing a huge overlap to account for label *human*. Each player can have more than one bounding box as long as one of them is of *human* class. NMS still works for suppressing detections of the same label. This NMS modification is used in experiments for 5 classes classification and tracking. In an ideal case, each player has exactly 2 bounding boxes - one of class human, one of the other class. In this thesis, we call this modification of the NMS algorithm *double label NMS*.

3.3. Classification

The classification task is assigning the predefined class to the detected object. Usually, both tasks are done simultaneously in current neural networks. In this thesis, we differentiate between these two tasks as we use detection and classification network for detection only.

In the classification task, the primary challenge is to define classes so that it is easy to make a model of each class. More general classes are harder to learn but could be applied to more scenarios. More specific classes are easier to learn but can not be readily generalized. YOLO learns classes by learning cell encoding such that logistic classifiers for each class can predict class from this encoding.

Only class *person* resembles floorball players in the COCO dataset. We defined four classes in the WFC-18 dataset. Goalies from both teams are in one class as they differ significantly from other players and resemble each other. Also, from goalie's area of movement is evident that these two identities can not be mistaken. As for team selection, we selected team from left half (for the first period) of the field as team A. This selection should have been random, but it showed up later that both teams selected as team A (FIN, GER) wear white jerseys. We later tried to address this problem, but from such a small training dataset, jersey colors would always be the problem. Chapter 2.3 discusses jersey colors in the WFC-18 dataset.

3.3.1. Human class

After learning YOLO for the four classes mentioned above, we tested the newly trained network on the THA-AUS match, both countries being rather exotic in floorball. We

purposefully selected match differing in jersey colors as much as possible from training matches. After examination of first frames from this match, it was apparent that the classification between team A and team B was inferior. Also, many players on opening face-off were not detected at all. Thinking that the problem is an insufficient generalization, we came up with a new classification approach, including fifth class - *human*.

The YOLO backbone is ready for multilabel classification, but ground truth processing is not as explained in chapter 3.2.1. Inspired by ground truth from the Open Images dataset [11], we decided to implement the fifth class by double detection. Each player would have one bounding box labeled by its class and one labeled as *human*. It would result in twice as many bounding boxes, two for each player. For this, we also modified the NMS algorithm and used *double label NMS* as described in chapter 3.2.5.

The experiment 4.3 presents a comparison between different classification methods.

3.3.2. Team templates

As we see in chapter 4.3.2, introducing the fifth class does not improve the detection score of the whole algorithm. We suspect that the problem is in low variability in training data. Also, trying to fit a new team into one of two classes is a significant constraint.

As a result, we propose training YOLO to recognize more than 2 teams. Crating numerous team classes for different colors would raise variability in team colors. Before each match, detection on one image would show which two templates are the most similar to actual teams, and the rest of the match would use only these two templates (along with class goalie and referee) to detect and classify players.

3.4. Identification

After correct player detection, we need an algorithm to identify individual players. When working flawlessly, the algorithm should identify a player even without observing him for a long time. Identification indicates assigning a unique ID number, not a real-world name. This ID number should be unique for the whole match (or video sequence) for each player. When the algorithm detects player for the first time, it assigns him an unassigned ID, when observed again, this ID should not change. Below is a simple structure of the desired algorithm.

It is common for players to play at the beginning and then at the end of the match only. In a time when the player is not playing, the identification algorithm should not forget his ID.

To remember a player across multiple frames, we need an encoding describing each player. The identification algorithm then comes down to find proper encoding, which would be descriptive enough as well as not too memory consumptive. For encoding to be descriptive enough, it should give proper values for inter- and intra-class distances. The encoding needs to fulfill the following two conditions.

First, all samples of the same class should be closer than samples of other classes. We call this condition **class discriminability** further in the text.

Second, all samples of the same individual should be closer than a sample of other individuals. For the rest of the thesis, we call this condition **individuals' discriminability**.

The class discriminability is solved with classification done in the detection part as it means to classify detection correctly. The second property is crucial for individual

Algorithm 2: Ideal algorithm for ID assignment

```
Result: All players with assigned ID by their identity
detect persons in video frames;
for each frame in video sequence do
    for each detection in frame do
        find last player occurrence;
        if player in previous frame then
            assign the same ID
        end
        if player in older frame then
            assign the old ID
        end
        if player never seen then
            assign new ID
        end
    end
end
```

player discriminability.

The main obstacle in finding this encoding is players' similarity. Players from the same team not only wear similar clothes but usually also shoes and floorball sticks, as they are from the same sponsor. The encoding then has to find small dissimilarities between players like beard or tattoos. One approach would be to use jersey numbers and optical character recognition (OCR) software. We did not use this approach as numbers are not visible for a long time and not from top-view videos. It remains as a possibility to use OCR software for side-view videos.

We aim for an online identification algorithm that could be used during a match if detection and identification are fast enough. Online means that identification is dependent only on previous frames. It is possible to annotate the first frames of the match as initialization.

3.4.1. YOLO encoding vectors

When searching for proper encoding, YOLO anchor encoding comes to mind as the first candidate. As outlined in chapter 3.2.1, YOLO divides an image into cells and encodes each cell. The YOLO encoding should facilitate class discriminability condition when classification works right as YOLO classify detections using logistic classifier on this encoding. It remains to test the individuals' discriminability.

The nearest neighbor algorithm requires fixed-length encoding. However, YOLO returns detections on three scales with three different encoding lengths. Testing revealed that more than half of the detections were done in a 52×52 cell grid. To normalize the length, we disabled detection on different scales and used encoding of length 256 from the 52×52 scale.

Nearest neighbor

The nearest neighbor refers to classification (or here tracking) by predicting class based on class according to the nearest sample. We use Euclidean distance in the space of YOLO encoding of length 256.

The nearest neighbor algorithm is ideal for simple testing of individuals' discriminability. If YOLO encoding works as expected, the nearest neighbor of selected encoding from the previous frame is encoding from the current frame. The simple single-player tracking by YOLO encoding distance implements this idea. The tracking algorithm is outlined below.

Algorithm 3: Nearest neighbor YOLO encoding single player tracking

```
Result: sequence of frames with marked tracked player
detect persons in video frames;
select player from the first frame;
for each frame in video sequence do
    for each detection in frame do
        compute euclidean distance in encoding space from tracked player in
         previous frame;
    end
    mark detection closest to the player from previous frame;
end
```

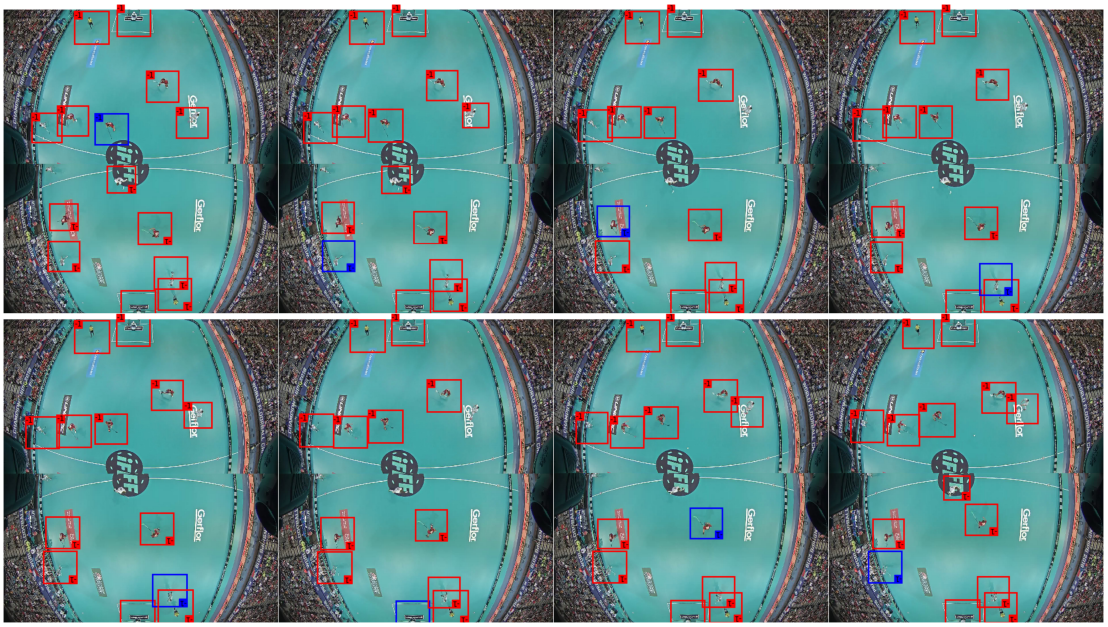


Figure 3.6. Example of tracking by the nearest neighbor algorithm using the YOLO encoding. Blue is the tracked player.

This approach should work at least as long as detection and classification work. From the sequence 3.6, it is clear that the nearest neighbor approach failed. This could be because the distance should not be Euclidean. Or because the YOLO encoding does not yield any information about an individual.

Logistic regression

To test whether YOLO encoding vectors carry information about individuals' discriminability, we trained logistic classifier for identity recognition using vectors from two images. A logistic classifier is an algorithm classifying new samples based on logistic

regression on the training dataset. It accounts for nonlinearities in distance metric as opposed to Euclidean distance. The training was done in a one-against-all way with the NumPy module [18] for Python. A similar approach is used in YOLO for classification from anchor encoding.

With this enhancement, the nearest neighbor algorithm with a logistic classifier instead of a Euclidean distance produced similarly poor results.

Deep neural network

Lastly, the experiment with a fully connected deep neural network (DNN) visualized in image 3.7 reached no improvement as well. We extracted the encoding vectors of 10 players from 200 frames long sequence from the FIN-SWE match. Vectors were from all three scales and padded by zeros to normalize length 1024, to keep as many vectors as possible. The learning was done using triplet loss from [26].

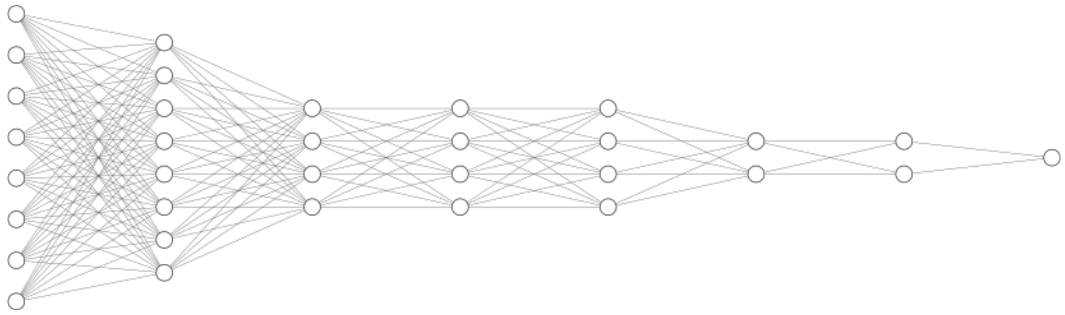


Figure 3.7. Visualization of the deep neural network for YOLO encoding vectors. Each dot represents 128 nodes. Input is of size 1024; Output is of size 128.

Player	class	number of vectors
player 1	team A	182
player 2	team A	117
player 3	team A	137
player 4	team A	48
player 5	team A	107
player 6	team B	122
player 7	team B	113
player 8	team B	120
player 9	team B	136
player 10	team B	116

Table 3.2. List of YOLO encoding vectors used for training of the DNN

After learning for 200 epochs, the distance between the same individual samples was similar to the distance between different individuals samples. Retraining the network

for different margins (2, 20, 200) and different distance metrics (L1, L2) delivered no improvement. Images 3.8 offer a visualization of the learning progress. This visualization was done performing the linear discriminant analysis (LDA) implemented in [21] on transformed vectors. LDA was computed on each epoch again. Visualized are two most discriminant components for each epoch.

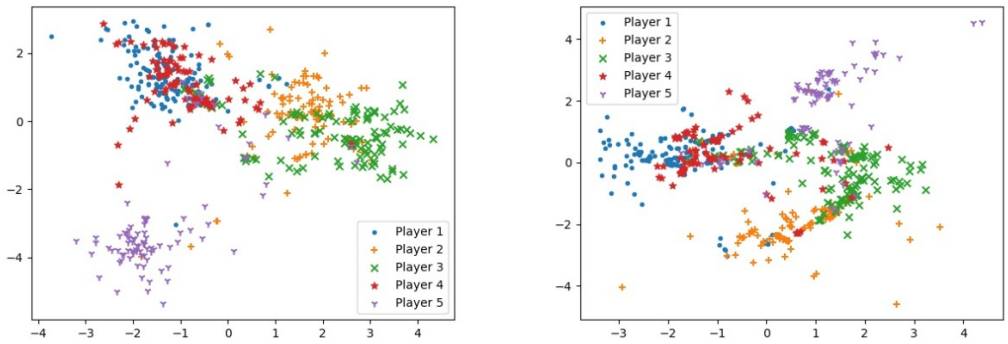


Figure 3.8. Visualization of the learning process of the YOLO encoding. Vectors of length 128 are projected to 2D space using the LDA. Images shows visualization before and after the learning.

Summary

From these experiments, we assume that YOLO anchor encoding does not yield any information about individuals. YOLO backbone differs between classes and encodes anchors, so that same class samples are close and different class samples are far regardless of the organization in class but does not differ individuals from the same class.

3.4.2. ResNet

After YOLO cell encoding did not prove useful, we extracted information directly from the image. The ResNet network [9], in its elementary version of 18 hidden layers referred to as ResNet 18, encodes images of players. ResNet is a deep convolutional neural network that encodes an image to a vector of length 1000. Using triplet loss as in chapter 3.4.1, we trained ResNet to encode player images to satisfy individuals' discriminability. A training dataset consisted of cropped and rescaled images of 10 players from two teams. As these are the same players as for YOLO encoding, the overview is in table 3.2. We used ResNet with weights pretrained from PyTorch [20].

Images 3.9 offers a visualization of learning. They visualize the first two components of LDA, as in chapter 3.4.1. The first two images visualize the train set; the last image visualizes the test set. We can see that players start to separate into clusters after several epochs. It is also evident that the encoding does not differentiate classes. It is because in the loss function is no part representing classes but only individuals. Inconsistency in visualization is caused by LDA being computed for visualization after each epoch.

We experimented with training on as few as 3 images of 5 players with similar results. The ResNet used in the method was trained on all ten players, but this experiment shows promising results for possible match initialization in the future.

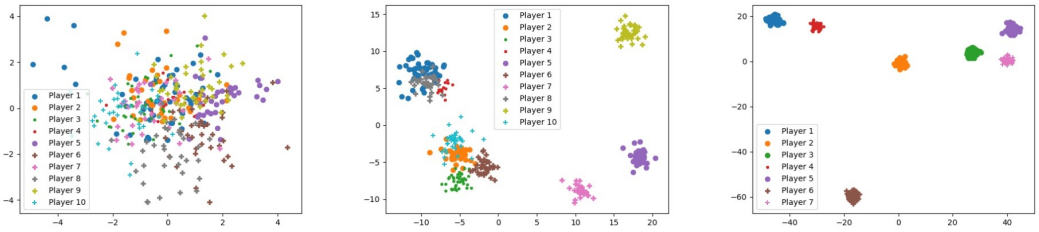


Figure 3.9. Visualization of the learning process of the ResNet 18 encoding. Vectors of length 1000 are projected to 2D space using the LDA. Players visualized as circles are from team A, crosses are from team B. Left picture is visualization before learning, the middle one is after 39th epoch, the right one after the learning.

After implementing ResNet as a detection processing tool after the detection algorithm, we tested whether LDA transformation brings any improvement in tracking. As seen in table 3.3, LDA is useful only for visualization as it makes tracking results worse.

Method	IDF1	IDP	IDR	MOTA
human only 200 with LDA	29.3	30.3	28.3	90.1
human only 200 without LDA	50.1	51.9	48.4	90.1
human only 150 with LDA	14.9	14.6	15.1	84.6
human only 150 without LDA	26.1	25.7	26.6	84.6

Table 3.3. Effect of using LDA in the similarity tracking

The method without LDA is used as a similarity metric in the final algorithm. After detection, each player's cropped image is extracted and resized to fit into the required ResNet 18 input. The output from ResNet 18 is then used as a similarity encoding for assignment.

3.4.3. Assignment

Lastly, the experiment with a fully connected deep neural network (DNN) visualized in image 3.7 reached no improvement as well. We extracted the encoding vectors of 10 players from 200 frames long sequence from the FIN-SWE match. Vectors were from all three scales and padded by zeros to normalize length 1024, to keep as many vectors as possible. The learning was done using triplet loss from [26].

The Hungarian algorithm [10] implemented in [25] solves the assignment problem. Input to this algorithm is cost matrix; output is the assignment. The value in an *n-th* row and *m-th* column of the cost matrix symbolize the cost of assigning *n-th* detection from the previous frame to *m-th* detection in the current frame. Assigning detections signifies that these detections contains the same player and have the same ID.

The cost of the assignment consists of two parts - position cost and similarity cost. The position cost is the distance in pixels between two detections in the image plane. Position cost 100 means that two detections are 100 pixels apart. It is a Euclidean distance of centers of bounding boxes.

The similarity part is the euclidean distance in the space of ResNet encoding vectors. The more similar two images are, the closer they are in this space.

Images from two cameras

The division of the playing field into two images introduces a problem in position cost. The program stacks two images vertically so that the top edge of the right half is connected with the bottom edge of the left half. This would solve the problem with two frames entirely if images made the playing field exactly. As there is an overlap of halves in the frames, there remains a small gap in location cost. Also, the radial distortion, which is not removed in images, introduces nonlinearities in real-world coordinates.

Cost functions

The position and similarity metrics have to be transformed to represent cost. In theory, the similarity metric would return a value between 0 and 1. This value could then be interpreted as a probability of detections being the same player. To form cost, we would then take negative value or value one minus probability. It would make sense as the probability of the same player is not linear. When the two detections are 5 pixels apart, we can be sure that this is the same player. On the other hand, when detections are 1000 pixels apart, it is clear that these detections are of the same player. The same logic applies to the similarity metric.

The logistic classifier is suitable approach for transforming distance to probability. Position and similarity metrics need different classifiers since the metrics are not normalized. Because of the lack of time, we opted to use functions simulating logistic regression. Instead of fitting logistic regression, we experimentally estimated the threshold for each part of the cost. The thresholds are 100 for location and 200 for similarity. Resulting functions are defined in equations 3.1 and 3.2 and are plotted in the image 3.10. Logistic regression visualized is already reversed so that max value means 0 probability and otherwise.

$$f(p) = \begin{cases} \text{p}, & \text{if } p < 100 \\ 1000, & \text{otherwise} \end{cases} \tag{3.1}$$

$$f(s) = \begin{cases} \text{s}, & \text{if } s < 200 \\ 900, & \text{otherwise} \end{cases} \tag{3.2}$$

The position part and similarity part are then summed without weights to form the assignment cost. After the Hungarian algorithm returns the cheapest assignment, identities are assigned if the cost for each assignment is under threshold 1000. When the cost is higher, we assume that this detection does not have its counterpart in the current frame and works with it as unbalanced detection. The threshold is chosen so that it forbids jumps bigger than 100 pixels in location.

Player memory, new IDs

After detections with counterparts in current frames are assigned, it rests on processing unassigned detections from both current and previous frames. Detection from the previous frame without assignment is saved into memory for later reassignment. Unassigned detections from current frames are compared with detections in memory, and the ID with the smallest similarity cost is assigned. If no detections are in memory, a new ID is assigned.

This approach allows redetecting lost detections based on the similarity metric. When there are more unassigned detections in the current frame, it assigns old IDs using a

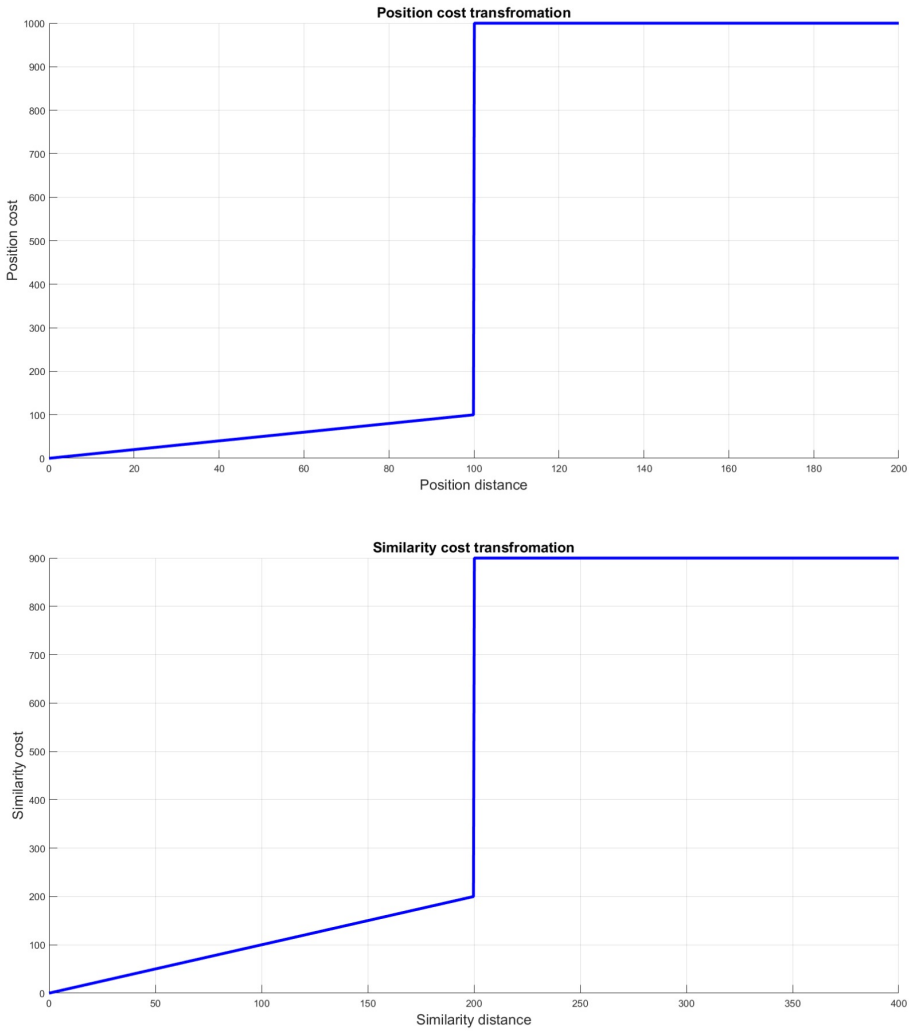


Figure 3.10. Plots of nonlinear transformations of location and similarity cost

greedy algorithm, which is sub-optimal. As the algorithm saves the lost detections only, in memory are the latest encoding vectors. If the player is redetected after a long time, significant rotation, or other significant visual transformation, the similarity metric could fail.

Below is the whole identification algorithm.

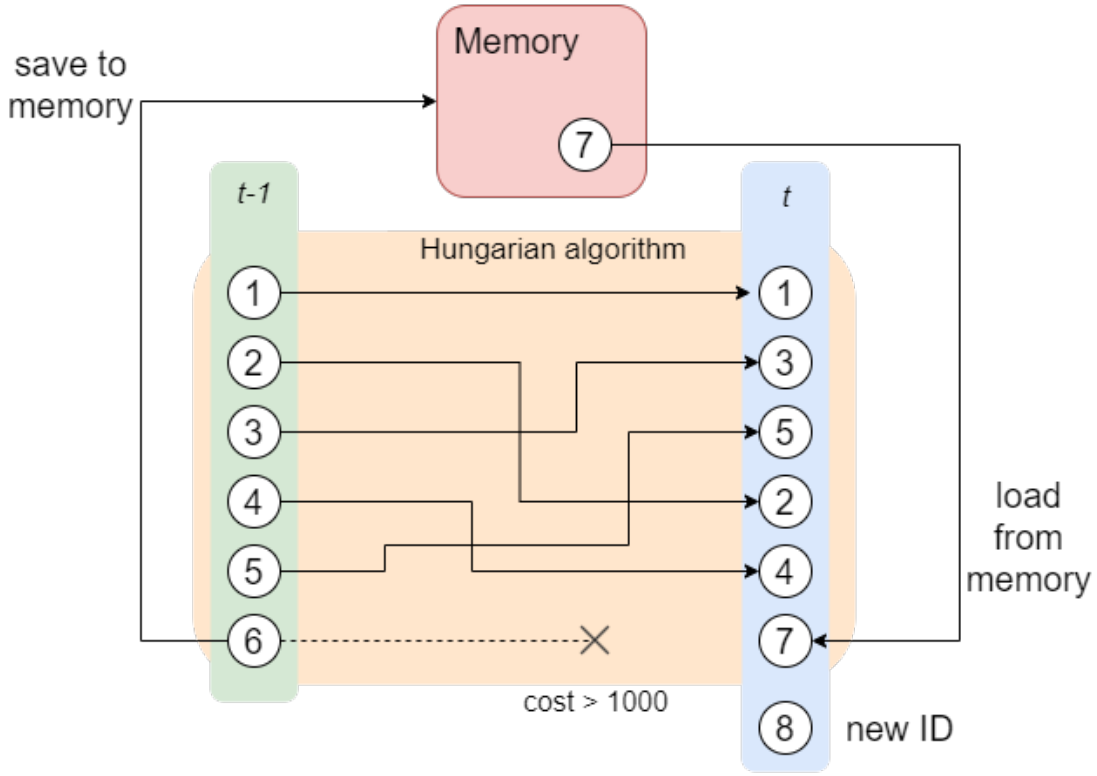


Figure 3.11. Possible situations in the assignment problem. Orange rectangle signifies the assignment by the Hungarian algorithm. Detections in the green and blue rectangles are from the frame *t-1* and *t*, respectively. ID 6 from the frame *t-1* is not assigned as the assignment cost is higher than the threshold.

Algorithm 4: Implemented tracking algorithm

```
Result: Each detection with assigned ID
detect persons in video frames;
for each frame in video sequence do
    compute location cost matrix;
    threshold location cost matrix;
    compute similarity cost matrix;
    threshold similarity cost matrix;
    sum similarity and location cost;
    compute assignment by Hungarian algorithm;
    for each detection in frame do
        if assignment cost ≤ 1000 then
            assign ID by Hungarian algorithm;
        end
        if assignment cost > 1000 OR unassigned then
            if any detections in memory then
                assign the most simillar ID from memory;
            else
                assign new ID;
            end
        end
        save unassigned detections from previous frame;
    end
end
```

4. Experiments

4.1. Metrics

4.1.1. Detection

In this thesis, we use mean average precision (mAP) as the measure of detection quality. Before explaining mAP computation, we introduce basic metrics. Below are the definitions used.

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. It could either be that to correct detection was assigned the wrong label or that there is no object to be detected.

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

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

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

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

All metrics are used per class so that a positive sample from one class is a negative sample from others (assuming exclusive classes). To determine whether detection is the same as ground truth, we need to define the similarity of two bounding boxes. We use intersection-over-union. **Intersection-over-union** (IoU) is a metric telling us how much two bounding boxes overlap. It is a ratio of area common for both bounding boxes to the area covered by the boxes together. IoU 1 means that bounding boxes are identical; IoU 0 means that bounding boxes do not overlap. In the image 4.1 is a visualization of this metric taken from [22].

For the rest of this chapter, we assume that two bounding boxes overlap with IoU 0.5 and above.

The average precision (AP) evaluates the classification. **Average precision** is defined as the area under the recall-precision curve. In definition below, r stands for

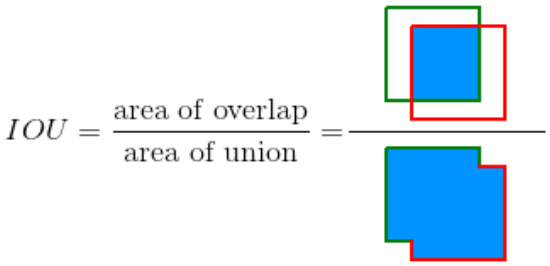


Figure 4.1. Visualization of the intersection-over-union metric. Image taken from [22]

recall, and $p(r)$ stands for precision as the function of recall. Since precision and recall are normalized between 0 and 1, AP 1 is the best result.

$$AP = \int_0^1 p(r)dr \tag{4.3}$$

The average precision (AP) rates the classification. **Average precision** is defined as the area under the curve (AUC). In definition below, r stands for recall, and $p(r)$ stands for precision as the function of recall. Since precision and recall are normalized between 0 and 1, AP 1 is the best result.

All detections are sorted by confidence, and the evaluation script computes precision and recall for each detection (taking into account only previous detections). The AUC is shown in image 4.2. We use the all-data interpolation method used in VOC Challenge since 2010. For more details, see [7].

After computing AP for each class, the mAP is a weighted average of APs, with weight being the number of samples.

The program from [22] evaluates detection and classification performance. For more details, see the documentation on its GitHub webpage.

All experiments were done with training on the train set from the WFC-18 dataset for 100 epochs. We present training parameters in table 4.1 below for possible replication of our results.

Parameter	value
Number of epochs	100
Batch size	10
Learning rate	0.001
Layers decay	0.0005
Momentum	0.9
Input image dimensions	416 × 416

Table 4.1. Learning parameters used for in experiments to fine-tune the YOLO network

4.1.2. Tracking

Metrics from MOT Challenge, as defined in [12], rate the tracking performance.

IDP and **IDR** are precision and recall for identities. The meaning is the same as explained in the previous chapter about detection metrics.

IDF1 is the harmonic mean of precision and recall. The formal definition is in the

equation 4.4, and it summarizes precision and recall.

$$IDF1 = 2 \cdot \frac{IDP \cdot IDR}{IDP + IDR} \tag{4.4}$$

IDs used is the number of identities used to track players in the sequence. This metric is not a measure of quality itself but shows us the effect of memory and reassignment. Since our test sequence tracks 14 individuals, any number beneath 14 means that and an individual was not detected nor tracked. Any number above 14 then implies that one or more individuals were assigned with more than one ID.

Multiple object tracking accuracy (MOTA) is measure combining FP, missed targets, and identity switches

$$MOTA = \overline{m} + \overline{FP} + \overline{mme}. \tag{4.5}$$

Where $\overline{m}$ is the ratio of misses

$$\overline{m} = \frac{\sum m}{\sum gt}, \tag{4.6}$$

$\overline{FP}$ is the ratio of false positives

$$\overline{FP} = \frac{\sum fp}{\sum gt}, \tag{4.7}$$

and $\overline{mme}$ is the ratio of mismatches in the sequence

$$\overline{mme} = \frac{\sum mme}{\sum gt}. \tag{4.8}$$

We do not use multiple object tracking precison (MOTP) as it is metric focused on bounding box precision, and our goal is to track position and not find the tightest bounding box.

For the computation of the metrics mentioned above, we used the official MOT evaluation Matlab script from [31].

4.2. Detection performance

4.2.1. Bounding box size

This experiment shows the effect of bounding box size. The annotation consists of only one point in the image plane, but YOLO requires rectangular bounding boxes as input. All created bounding boxes have a center in an annotated point. The following table 4.2 presents a comparison of a square and rectangular bounding boxes. All examples were trained with default anchors and learning parameters from chapter 4.1.

From table 4.2, we can see that the bigger the bounding box, the more accurate detection. This is no surprise as a bigger bounding box means a more significant overlap for the wrong detection and improved learning, as outlined in chapter 3.2.2. The rectangular shape does not bring any improvement and has even worse results than square bounding boxes.

Bounding box size	mAP
50 × 50	39.73%
100 × 100	78.72%
150 × 150	82.10%
200 × 200	94.15%
300 × 300	93.14%
150 × 200	92.03%

Table 4.2. Comparison of the detection performance for different bounding boxes' sizes

4.2.2. Anchor shape

Here, we show the influence of anchors on detection performance. As described in chapter 3.2.1, YOLOv3 uses 9 anchors of various sizes and shapes. Since our bounding boxes have the same size and shape, it should be easier to learn a better detector with anchors being the same size and shape as bounding boxes. Therefore, we trained YOLO with anchors edited so that it equals the bounding box and without classification (only one class in the ground truth).

Model	mAP
human only 150	88.31%
human only 150 noAnchors	83.79%
human only 200	94.15%
human only 200 noAnchors	93.01%

Table 4.3. Comparison of the detection performance with and without using anchors

As seen in table 4.3, removing bounding box shape from the learning process does not bring any significant improvement. On such a small dataset, YOLO can learn bounding box shape in 100 epochs.

4.2.3. Region proposal

This chapter presents the results of detection overall. We trained YOLO on only one class serving then as a region proposal network. Classification and identification would have to be done in a separate pipeline, for example, using the ResNet network. Although these results were already presented in previous experiments, this chapter examines detection performance in detail.

Method	mAP
YOLO	1.40%
human only 200	94.15%
human only 150	88.31%

Table 4.4. Results of detection with one class

In table 4.4, we can see a comparison with untrained YOLO. Untrained here stands for YOLO pretrained on the COCO dataset. As expected, untrained YOLO shows significantly worse results than the same network trained for my dataset.

When examining testing images, we can see that most errors are around half-line when a player is transitioning from one image to another. The wrong annotation causes this. For more about this problem, see chapter 3.2.3. Other than this, advertisements

on playing field cause detection problems as some players standing on them are not detected. The last sources of errors are false detections in crowded areas in front of the goal, which are the hardest situations for detection as well as tracking in top-view videos. Crowded situations and double detection (one person with two labels) are the only source of false positives.

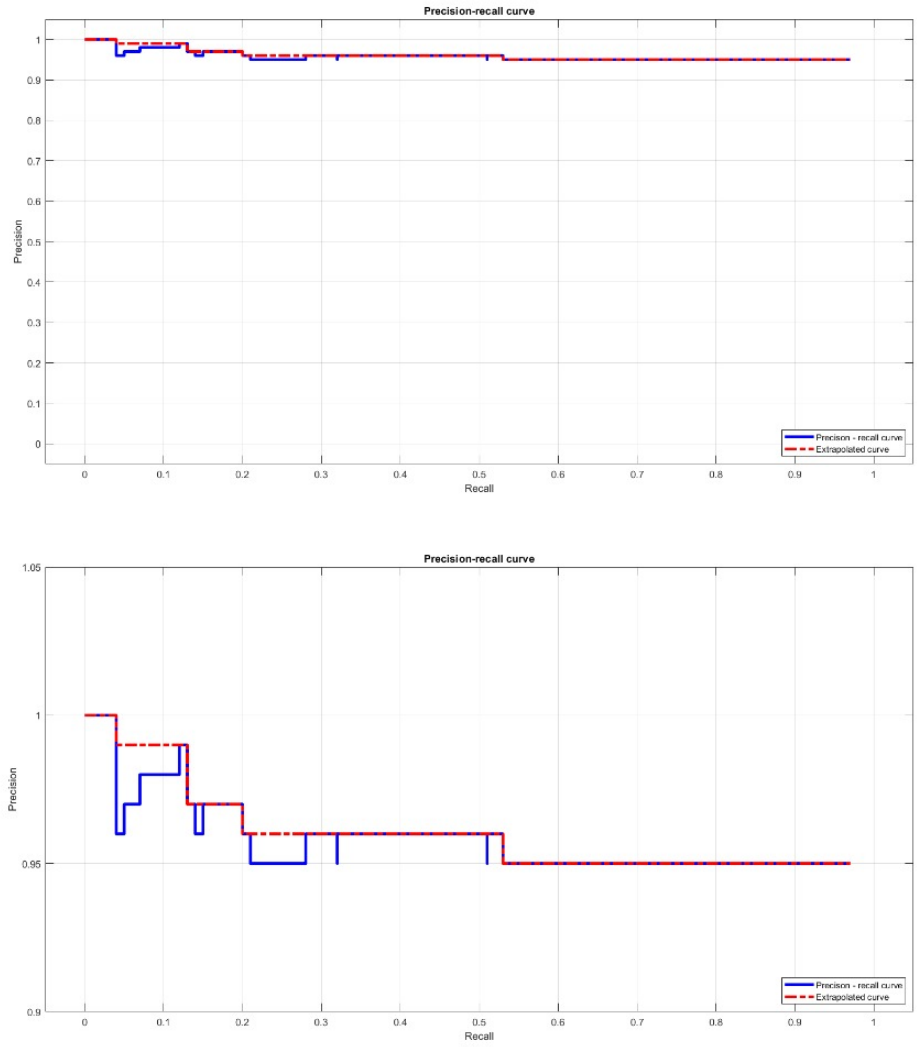


Figure 4.2. Precision - recall curve of the human only 200 model. Bottom picture shows the detail of the top picture.

4.3. Classification performance

4.3.1. 4 class classification

Next, we present results for the network trained for detection and classification. The first situation shows the network trained to recognize 4 classes - two teams, goalies, and referees.

This approach worked well on the same or similar matches as the training ones. After testing on different matches (teams and referees with different jersey colors), not

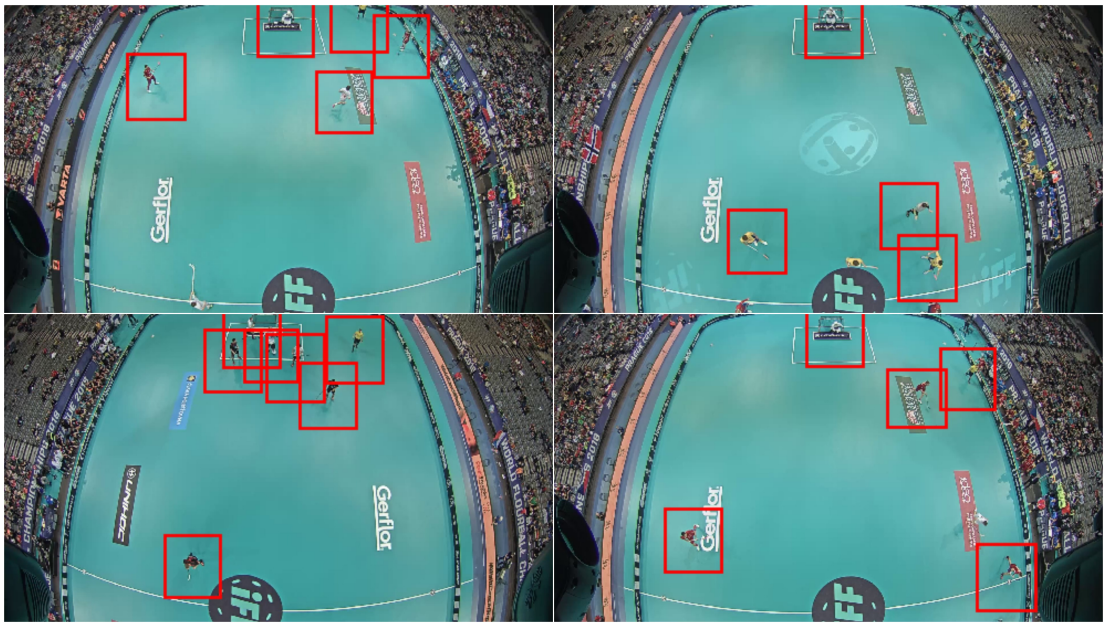


Figure 4.3. Examples of detection with one class. From top left - successful detection, a player on faceoff not detected, false detection in the crowd, a player not detected on the advertisement.

Method	mAP	AP_{teamA}	AP_{teamB}	AP_{goalie}	AP_{ref}
150 × 150	78.40%	66.40%	71.16%	82.91%	95.28%
200 × 200	76.72%	59.57%	71.54%	78.8%	96.81%

Table 4.5. Classification performance for 4 class approach

only classification went down, but also detection was incorrect as the network could not identify players with sufficient confidence. It is because of overfitting team classes. Since YOLO could not assign a new team to any of the pretrained teams, it ignores it completely. It also caused substantial misclassification errors. Examples of classification are in image 4.4.

4.3.2. Human class

As the previous technique could not classify new matches correctly, we changed the training method and added a new class called *human*. All persons have two bounding boxes - one for its label, one labeled as human. The goal was to learn classification as well as detection when classification confidence is too low. This experiment takes advantage of the double detection suppression mentioned in chapter 3.2.5.

Method	mAP	AP_{teamA}	AP_{teamB}	AP_{goalie}	AP_{ref}	AP_{human}
4 class 150	78.40%	66.40%	71.16%	82.91%	95.28%	-
4 class 200	76.72%	59.57%	71.54%	78.8%	96.81%	-
5 class 150	48.97%	23.99%	26.52%	34.65%	90.25%	69.45%
5 class 200	56.06%	40.05%	39.67%	33.36%	86.41%	80.78%

Table 4.6. Results of detection with 4 and 5 classes

It is clear from the result that introducing the fifth class did not improve classification

but worsened it. We believe it is because the YOLO backbone is built for multilabel detection. The YOLO tries to estimate one bounding box with two labels. Meanwhile, we force to learn it to estimate two identical bounding boxes, each with one label. The generalization to other matches by the fifth class seems complicated, and, therefore, we propose using the team templates approach.

From image 4.5, we can see how the YOLO is estimating *human* class for all players but goalies. The classification performance declined, and the detection is still not as good as for human-only detection. This approach is then not useful for tracking nor for classification.

4.4. Tracking performance

4.4.1. Assignment methods

Below is a comparison of different methods for the assignment algorithm, as outlined in chapter 3.4.3. We show examples where the cost matrix was assigned with the similarity metric, the distance metric, or the sum of these two and the improvement with memory introduction.

Method	IDF1	IDP	IDR	MOTA	IDs used
human only 200 sim	50.1%	51.9%	48.4%	90.1%	30
human only 200 dist	75.5%	78.3%	73.0%	90.1%	30
human only 200 sum	78.2%	81.0%	75.6%	90.1%	41
human only 200 memory	84.4%	87.4%	81.6%	90.1%	15
human only 150 sim	26.1%	25.7%	26.6%	84.6%	41
human only 150 dist	65.4%	64.4%	66.6%	84.6%	41
human only 150 sum	64.5%	63.4%	65.6%	84.6%	51
human only 150 memory	75.4%	74.2%	76.7%	84.6%	18

Table 4.7. Comparison of different assignment methods

We can see from the table that introducing the similarity metric from ResNet 18 brings no improvement. When the tracking algorithm does not have any memory, it is heavily dependent on excellent detection. When detection fails even only in one frame, a new identity is created. The similarity metric is useful for assigning even between previous detections with the memory of previously detected players. The algorithm is more robust for detection failure and could even detect the same player across multiple changes. The example of the tracking is in the image 4.6.

4.4.2. Tracking results

This experiment presents a comparison with one of the state-of-the-art tracker - DiMP. As DiMP is a single object tracker, we ran it as many times as persons in the testing dataset, each initializing with a square bounding box of a given size. The first annotation is then the same as ground truth.

It seems in table 4.8 that DiMP has inferior results. Negative MOTA result means that there were more errors (FP and FN rate) than ground truth bounding boxes. Ground truth bounding boxes size and shape cause this. As DiMP estimates bounding box shape and size, it estimates smaller boxes than ground truth, and that is classified as false positive. Also, since initialization is by ground truth boxes, which are not tight

Method	IDF1	IDP	IDR	MOTA	IDs used	FP	FN
DiMP 100	42.6%	42.6%	42.6%	-14.01%	14	399	399
DiMP 150	28.9%	28.9%	28.9%	-35.1%	14	473	473
DiMP 200	27.0%	27.0%	27.0%	-44.9%	14	507	507
human only 200 memory	84.4%	87.4%	81.6%	90.1%	15	11	58

Table 4.8. Comparison of the DiMP tracker with my method

fit for a person, tracker often ends up tracking advertisement or structure on the playing field. Again, examples are in the image 4.7.

4.5. Field projection

The system projects detected players with assigned IDs back to the playing field. It estimates homography similarly as in chapter 5.1.3 from 12 hand-clicked points. The following images offer the visualization of projection to side-view video and the diagram of the playing field. Coaches widely use these diagrams, so this projection is a close connection between offline handwritten tactics and video analysis.

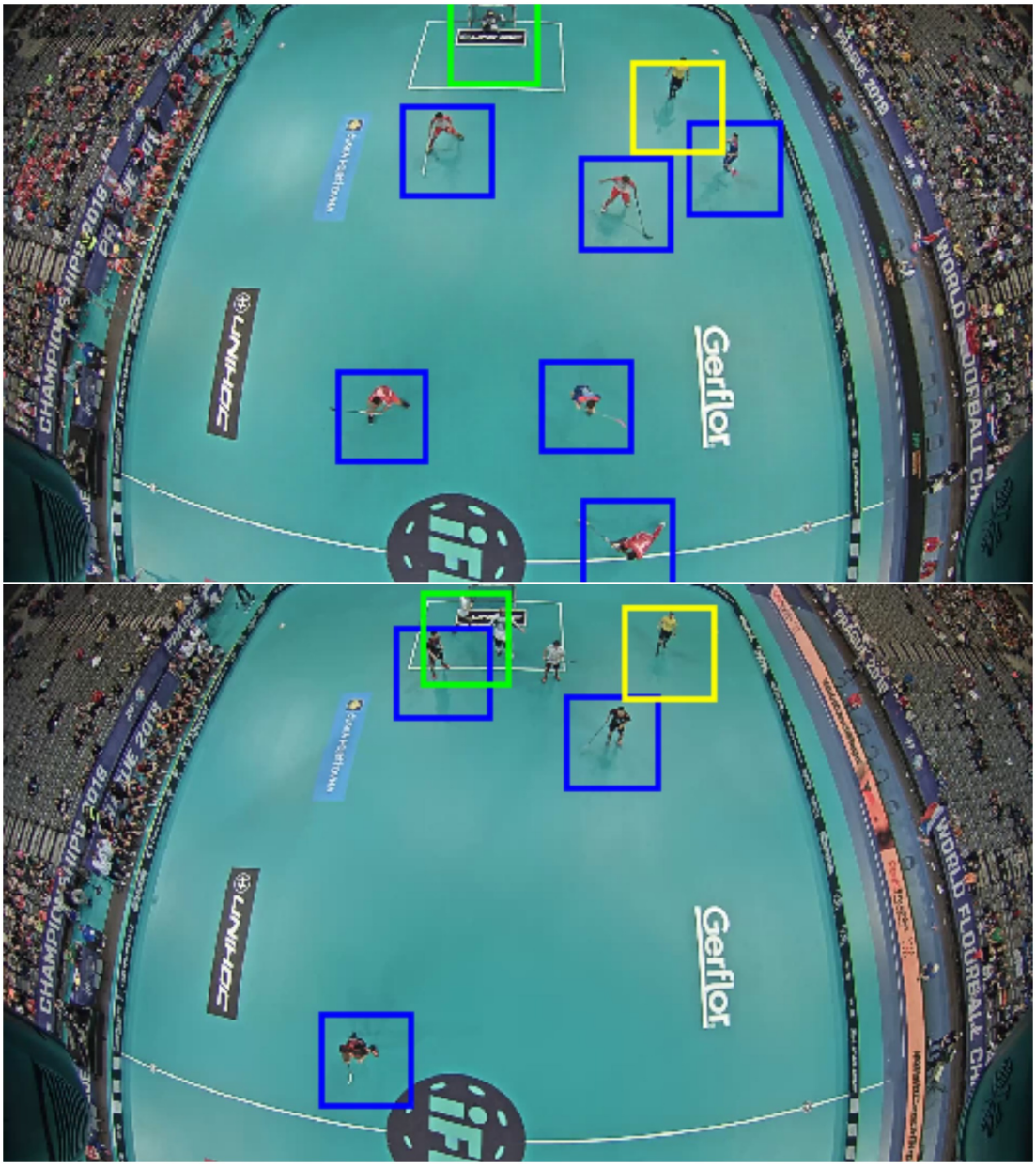


Figure 4.4. Examples of classification with 4 class approach. Top to bottom - misclassification, false negative

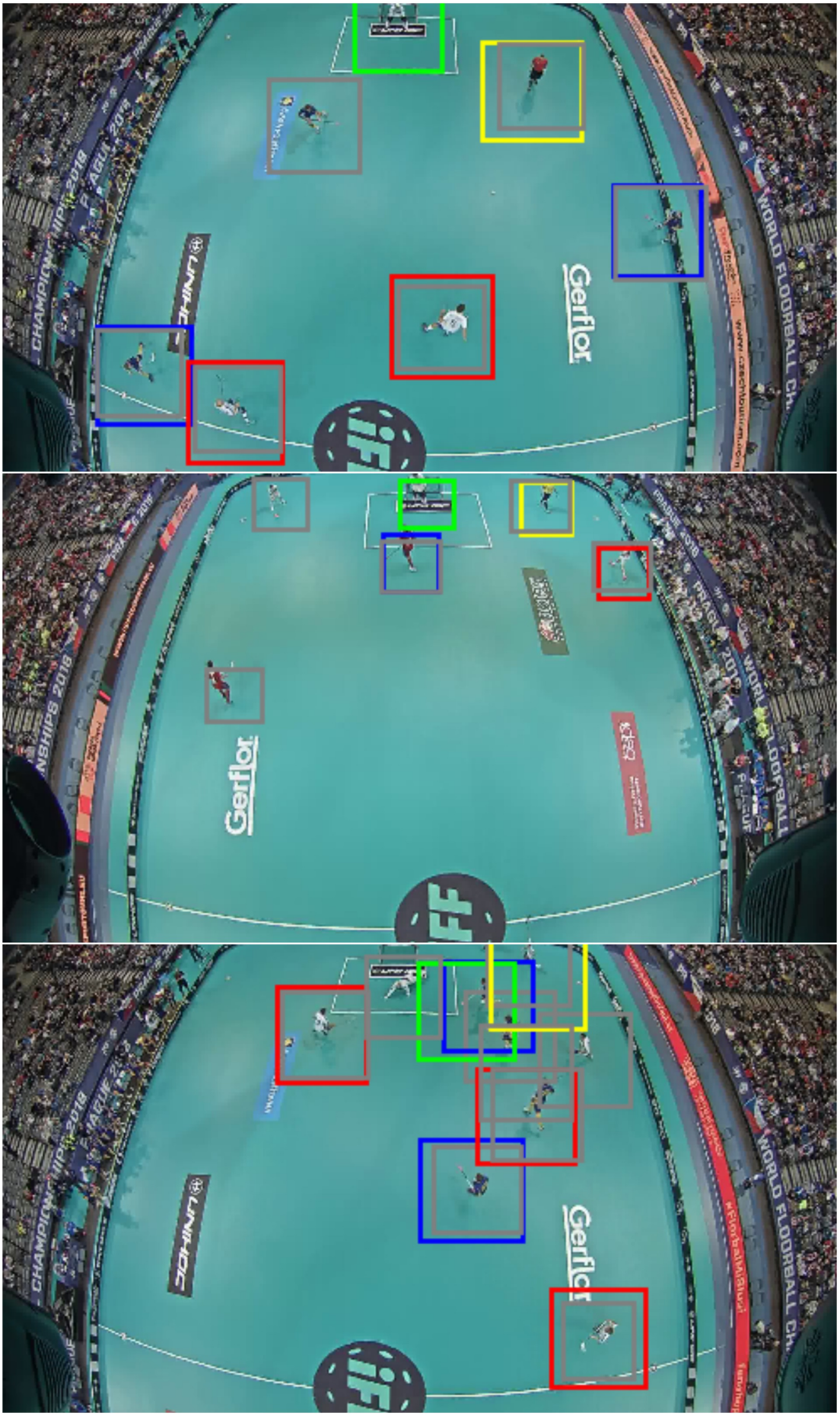


Figure 4.5. Examples of classification with the fifth class. From top to bottom - successful classification, players with one label, false detections

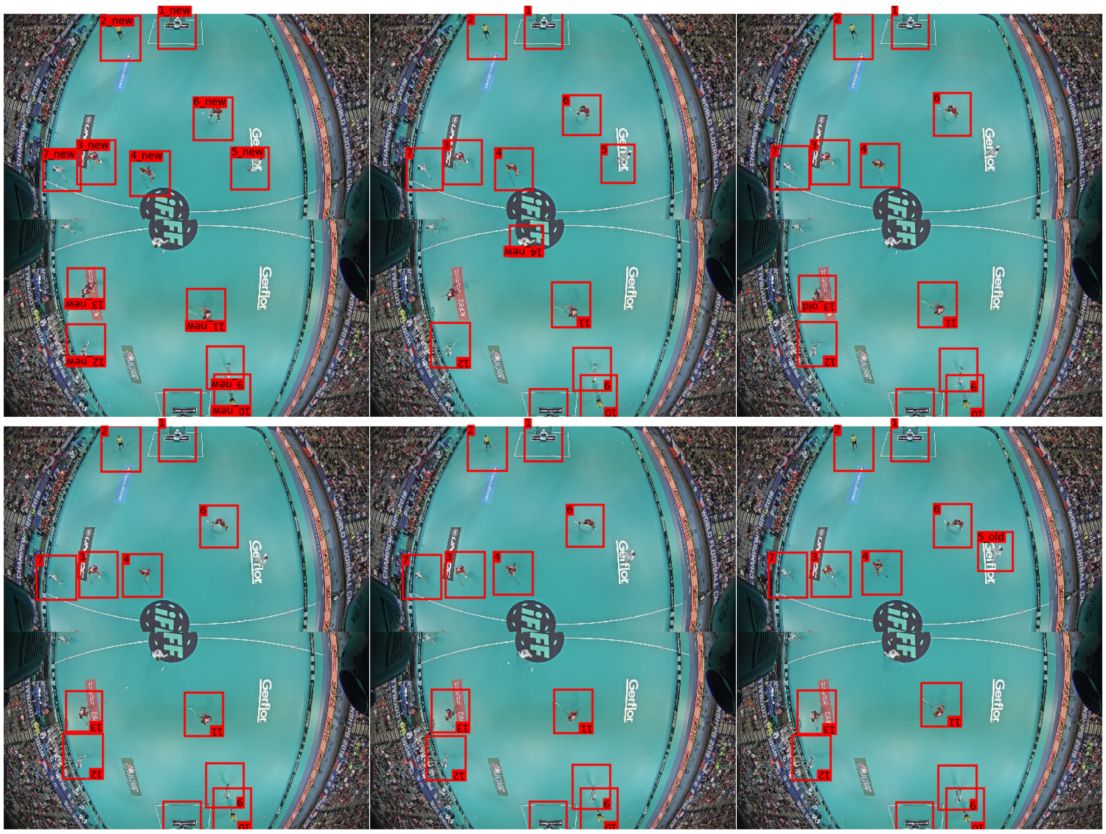


Figure 4.6. Example of the tracking by the assignment with memory. We can see problems with detection around the half-line and successfully redetected player with ID 5.

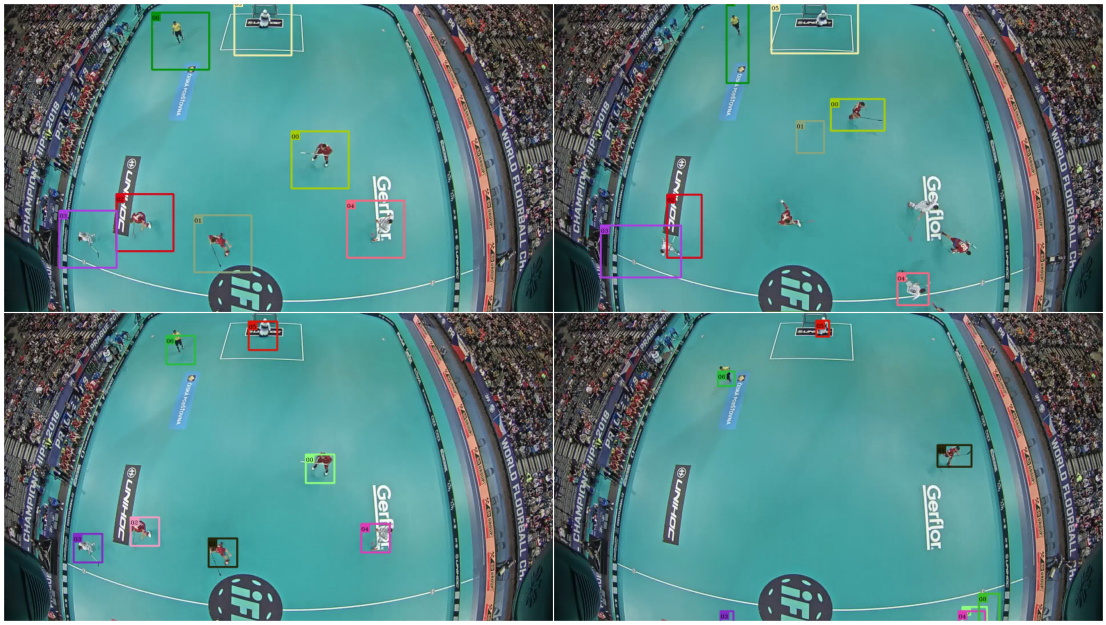


Figure 4.7. Visualization of tracking by the DiMP tracker. The first row shows the initial frame with bounding boxes of size 200. The second image is tracking after 39 frames. The second row shows the same situation with bounding boxes of size 150.

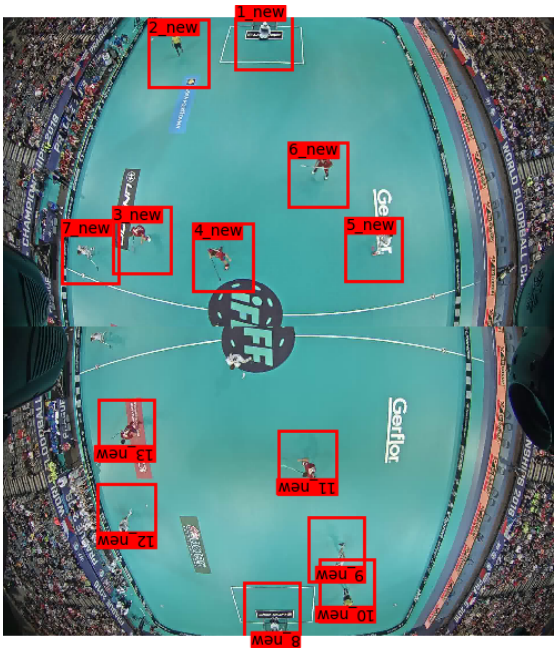


Figure 4.8. Visualization the projection to the playing field - detections from the top-view camera

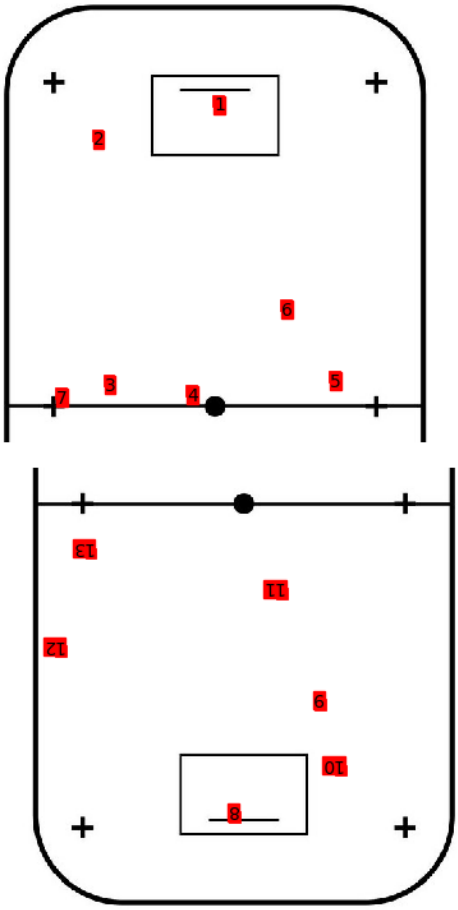


Figure 4.9. Visualization the projection to the playing field - detections projected to floorball board

Figure 4.10. Visualization the projection to the playing field - detections projected to the side-view camera

5. Implementation

This chapter describes the implementation details of both the Floorball Annotator app and the image processing pipeline. It emphasizes the code and libraries used. The major modifications in logic are listed in chapters 3 and 2.

5.1. Floorball annotator

The Floorball Annotator app is written in Python. The central part employs the QT library for Python - PyQT in version 5 [33]. It is itself a standalone app using packages available in Python either in default or via Pip or Anaconda and video processing app called FFmpeg [8]. The source code for class QtImageViewer is from [35]. The app folder includes an installation script providing automated installation. The app runs on Linux only and was tested on Ubuntu 16.04.

5.1.1. Installation

The *install.sh* script is available to install the app. This script installs all needed programs as Python 3, FFmpeg, pip, and Python packages. The script does not search for programs if not found in known repositories via the apt-get tool.

The app expects data organized in the folders so that each match has its folder, where the app expects 3 folders called KL, KP, and 4k for videos from different views. Image 5.1 displays the expected structure. We recommend using a short and unique name for the app names created frames and annotations by the video name.

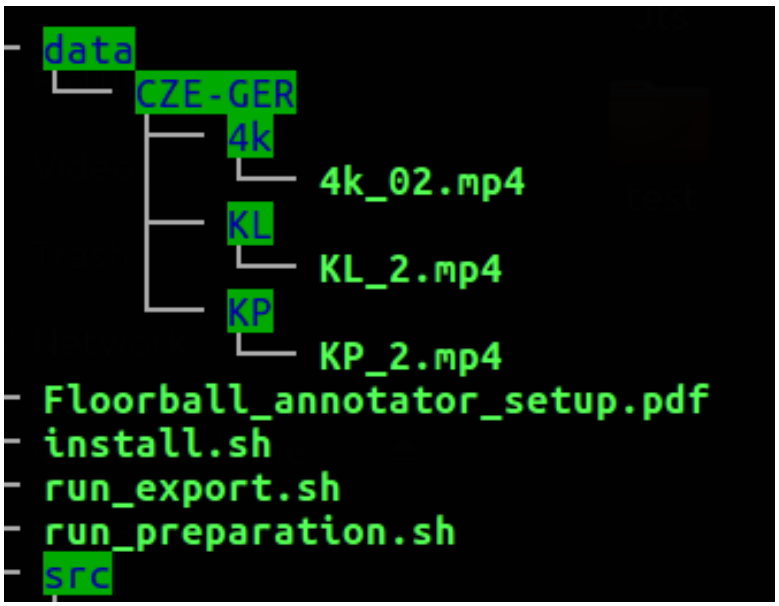


Figure 5.1. Folder structure required by the Floorball Annotator app. Videos names should be short for better frames naming.

5.1.2. Scripts

Two bash scripts are ready to start and finish the match. The first of them is *run_preparation.sh* script. This script handles free bash software called FFmpeg. It retrieves frames from videos and saves them appropriately. In default settings, it extracts the whole video selecting 5 frames per second. The script also computes radial distortion and estimate homography for each video. The created configuration file holds all information about the match.

The last parameter missing is synchronization. Since this app can not synchronize videos automatically, the user has to write adequate starting frames to the configuration file. The default value is the first frame for all videos.

When all parameters are set up, the main Floorball Annotator app is launched. Top-views with annotations are on the right side, on the left side are control buttons and side view. Annotation is possible only in top-views videos. When a point is marked, it appears in side-view for control. Zoom is possible by right click, movement between images is possible by buttons or keyboard shortcuts. All annotations are saved upon image change.

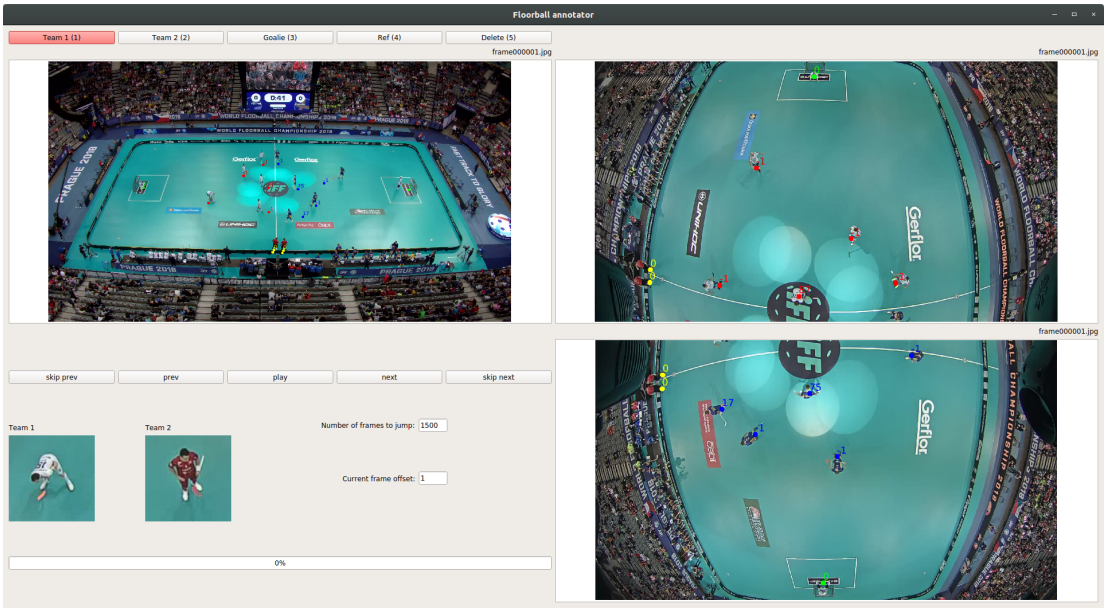


Figure 5.2. Screenshot of the Floorball annotator app

When annotations are ready, script *run_export.sh* extracts all text files and zip them along with the configuration file. The whole match then can be annotated and then send to another machine without the need to send images or videos. With the configuration file, it is then possible to reconstruct the annotated video. The design allows multiple users, each annotating a different match. They have to have access to the shared data folder to collaborate on the same match. Since annotations are loaded after each image change, more people can work on the same match as long as they do not access the same image.

5.1.3. Radial distortion and homography estimation

For radial distortion computation, the simple GUI tool in *preparation_tool.py* script guides the user to select a player from each team as a template for annotation. The

user also clicks on points to compute homography and radial distortion. The script then estimates the homography as well as radial distortion, as outlined in chapter 2.2.2.

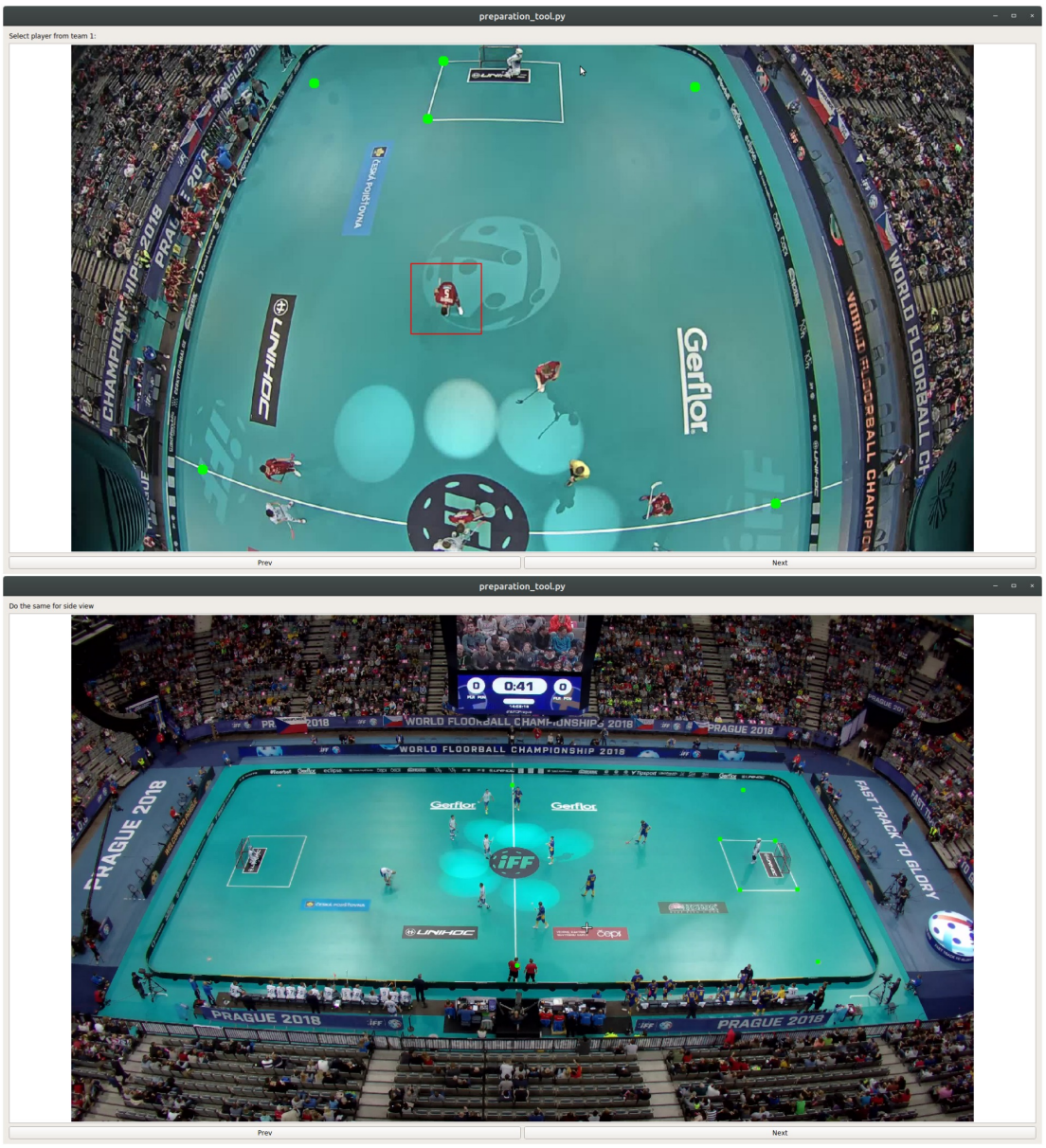


Figure 5.3. Screenshot of the preparation tool

Function from the OpenCV library for Python [5] estimates the homography. It uses default all-points method as points are manually selected, and this method achieved better results than RANSAC. It uses the basic formula for transformation in 5.1 to estimate radial distortion. Estimating radial distortion then comes down to determining the coefficient C. The program iterates through all sensible possibilities of the coefficient. It computes homography and reprojection error for each of them. Ultimately, it selects the homography and radial distortion coefficient with the smallest reprojection error.

$$x_u = x_c + \frac{x_d - x_c}{1 + C \cdot r^2} \tag{5.1}$$

where:

x_d = distorted point
x_u = undistorted point
x_c = center of the image
r = distance of distorted point from center of the image
C = distortion coefficient

5.1.4. Future use

We chose the format different from YOLO and other networks. Our annotation format is more user friendly and contains information as player number (and therefore ID) and origin flag. Origin flag is a groundwork for the human-in-the-loop annotations approach and is signaling where the annotation originated. The human-in-the-loop annotation method is summarized in chapter 2.2.3. The user annotates only a small part of the data, runs the sequence through the tracking network, and then checks and fixes the annotations. It allows faster videos annotating. The origin flag is also the first step to translate top-view annotations to side-view videos. With a well-functioning top-view detector and tracker, we can project top-view detections to side-view ground truth by the computed homography. This way, we can create a big side-view dataset without the need for annotating the whole sequence. Origin flag would then differ between human-annotated ground truth and automatically annotated ground truth.

```
1 CZE-GER_00011_KL
2 classID, x_center, y_center, number, originFlag
3 0, 1105.438972, 545.781585, 1, KL
4 0, 765.481799, 839.486081, 2, KL
5 0, 492.591006, 765.481799, 3, KL
6 1, 293.704497, 823.297645, 9, KL
7 1, 1299.700214, 788.608137, 10, KL
8 2, 906.552463, 78.629550, 0, KL
```

Listing 5.1 Example of an annotation made by the Floorball Annotator app

5.2. Image processing pipeline

The python port of YOLOv3 is a core of the image processing network. The majority of the code is from public repository [34] with YOLO implemented in PyTorch. We made several modifications to the code. This section explains only the new modifications. For more details about YOLO model implementation, see the original repository documentation. Not all alterations described below are available in the final version of the code enclosed to the thesis, though all are in the git history.

5.2.1. YOLO encoding vectors

The first change is in YOLO anchor embedding vector extraction. The network also returns the vectors along with detections and loss. Since the model outputs a vast

number of detections and then deletes most of them in the non-maxima suppression process, the system returns all vectors and deletes them in NMS. It retrieves 3649 ($= 13^2 + 26^2 + 52^2$) vectors of various lengths for each image. The vectors are stored in a list of lists to account for a different number of detections and different vector lengths. There could be (and often is) more detections than vectors as one cell could return up to three detections with identical anchor encoding. After NMS, there is one vector for each detection. If more detections are from the same cell, their encoding vector is duplicated to match. Vectors are sorted in the list in the same order as detections. Keeping the order is crucial to keep the vector connected with proper detection.

5.2.2. Detections processing

The next alternation is the way detections are processed. We designed a data container for image and detections since the program performs complex operations with detections in the tracking part. Each image is represented by the *ImageWithDetections* class, which contains a list of *Detection* class items. This structure allows us to process each detection on its own and make quick modifications in the data flow. As each detection and image is processed alone, it significantly slows down the tracking and detection saving process. In the development phase, we do not focus on algorithm speed. However, in this data structure, it is simpler to change separate parts of the system.

The network could output detections for each image for human-in-the-loop workflow, images with visualized tracking, or tracking in the MOT Challenge format. For debugging and development, the network could also produce cropped detections and projection to the playing field. All output options are available in parameters.

5.2.3. ResNet, Hungarian algorithm, LDA

The ResNet 18 network [9] encodes detection for tracking. We chose ResNet with 18 hidden layers for its size and speed. We believe that a deeper network would result in enhanced encoding. The ResNet pretrained weights are from PyTorch zoo [20].

The Hungarian algorithm function [10] comes from SciPy module [25], and we use the default settings.

Linear discriminant analysis (LDA) for encoding visualization is from the Scikit-learn library [21]. The LDA learns for each epoch again, resulting in not consistent representation. As we are using LDA only for visualization, it is not a problem.

6. Conclusion

We proposed a new system for online person tracking by detection. The system focuses on tracking persons in top-view sports videos. The proposed system is modular and can be easily updated with the latest research results.

To detect persons in sports videos, we modified and retrained the existing YOLO network to detect persons in top-view videos. The system attempts to classify persons into teams and predefined classes.

The developed system tracks detected players based on visual similarity and location on the pitch. A ResNet network fine-tuned on sports videos encodes detected players for visual tracking. Memory in the tracking algorithm allows person reidentification based on visual similarity. We tested the use of YOLO cell encoding for a person representation without success.

Experiments performed on a new floorball dataset show excellent results in top-view person detection. Modified YOLO achieves 94.15% mAP on the testing dataset for person detection. The only source of false positives are crowded situations and doubled detections. The classification works on validation set but decreases on the test set for poor generalization. The tracker has decent results and, on a short constrained sequence, achieves MOTA above 90%. It outperforms the state-of-the-art single person tracker on top-view videos.

The created dataset and the GUI app for automated data processing are enclosed in the attached CD.

6.1. Future work

Performed experiments indicated several drawbacks. The proper non-maxima suppression algorithm is crucial for the accurate detection and tracking of persons. We suggest using only a soft non-maxima suppression algorithm in the detection process and then suppressing more detections in the real-world coordinates to increase the detection performance.

To improve classification, we offer using team templates. The templates would bring smaller constraints with only small initialization by hand in each match.

As for tracking, optimal assignment along with using logistic regression for assignment cost would improve the assignment of the players and usage of the lost IDs memory. The creation of an internal model of each player could also improve reidentification.

We are confident that solving these challenges would bring significant improvement and get us closer to the production of a tool helpful for coaches and sports teams. We believe that the method would generalize to sports with similar player movements like basketball, handball, or hockey. The tool would find application in the highest leagues as well as national teams. The method uses multiple independent parts. The modular approach lets us change only one part should the latest research brings essential advance. It is especially true for person detectors and person encoding, which evolves with remarkable speed.

Bibliography

[1] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-Convolutional Siamese Networks for Object Tracking. *arXiv e-prints*, page arXiv:1606.09549, June 2016. 5

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. *CoRR*, abs/1904.07220, 2019. 5

[3] Erik Bochinski, Tobias Senst, and Thomas Sikora. Extending iou based multi-object tracking by visual information. In *IEEE International Conference on Advanced Video and Signals-based Surveillance*, pages 441–446, Auckland, New Zealand, November 2018. 5

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 12

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 43

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate Tracking by Overlap Maximization. *arXiv e-prints*, page arXiv:1811.07628, November 2018. 5

[7] 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, June 2010. 6, 29

[8] FFmpeg Developers. ffmpeg tool (Version be1d324) [Software]. `https://ffmpeg.org`. Online; Accessed: October 2019. 41

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 22, 45

[10] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955. 23, 45

[11] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018. 6, 18

[12] 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. 29

[13] Xin Li, Chao Ma, Baoyuan Wu, Zhenyu He, and Ming-Hsuan Yang. Target-Aware Deep Tracking. *arXiv e-prints*, page arXiv:1904.01772, April 2019. 5

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 6

[15] National Coordination Office for Space-Based Positioning, Navigation, and Timing. GPS Accuracy. https://www.gps.gov/systems/gps/performance/accuracy/. Online; Accessed: March 2020. 4

[16] NBA Media Ventures, LLC. NBA announces multiyear partnership with Sportradar and Second Spectrum. https://pr.nba.com/nba-announces-multiyear-partnership-sportradar-second-spectrum/. Online; Accessed: March 2020. 4

[17] Guanghan Ning and Heng Huang. LightTrack: A Generic Framework for Online Top-Down Human Pose Tracking. *arXiv e-prints*, page arXiv:1905.02822, May 2019. 5

[18] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. Online; Accessed April 2020. 21

[19] Scott Parsons and Jason D Rogers. Basketball player tracking and automated analysis. 2014. 5

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 22, 45

[21] 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. 22, 45

[22] Sergio Lima Netto Rafael Padilla and Eduardo A. B. da Silva. Survey on performance metrics for object-detection algorithms. 2020. 28, 29

[23] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv e-prints*, page arXiv:1804.02767, April 2018. 12

[24] A. Senocak, T. Oh, J. Kim, and I. S. Kweon. Part-based player identification using deep convolutional representation and multi-scale pooling. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1813–18137, 2018. 5

[25] 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. 23, 45

[26] 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. 21, 23

[27] Hefeng Wu, Yafei Hu, Keze Wang, Hanhui Li, Lin Nie, and Hui Cheng. Instance-Aware Representation Learning and Association for Online Multi-Person Tracking. *arXiv e-prints*, page arXiv:1905.12409, May 2019. 5

[28] Y. Yoon, H. Hwang, Y. Choi, M. Joo, H. Oh, I. Park, K. Lee, and J. Hwang. Analyzing basketball movements and pass relationships using realtime object tracking techniques based on deep learning. *IEEE Access*, 7:56564–56576, 2019. 5

[29] Zhipeng Zhang and Houwen Peng. Deeper and Wider Siamese Networks for Real-Time Visual Tracking. *arXiv e-prints*, page arXiv:1901.01660, January 2019. 5

[30] Amden company. https://amden.cz. 6

[31] MOT Challenge 2015 DevKit. https://bitbucket.org/amilan/motchallenge-devkit/src/default/3. Online; Accessed: April 2020. 30

[32] One-stage object detection. https://machinethink.net/blog/object-detection/. Online; Accessed: May 2020. 13

[33] PyQT Python wiki. https://wiki.python.org/moin/PyQt. Online; Accessed: October 2019. 41

[34] PyTorch-YOLOv3. https://github.com/eriklindernoren/PyTorch-YOLOv3. Online; Accessed: October 2019. 12, 44

[35] QtImageViewer class. https://github.com/marcel-goldschen-ohm/PyQtImageViewer. Online; Accessed: October 2019. 41

[36] Statistics Explained. https://www.premierleague.com/stats/clarification. Online; Accessed: March 2020. 4

A. Contents of the attached CD

purkrmir-Bachelor_Thesis.pdf	the text of this thesis
Annotator/	the Floorball Annotator project
PyTorch-YOLOv3/	the YOLO network with identification algorithm project
data/WFC-18/	folder with the dataset