MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Automatic Virtual Soccer Camera

Master's Thesis

ATTILA ZSÍROS

Brno, Fall 2023

MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Automatic Virtual Soccer Camera

Master's Thesis

ATTILA ZSÍROS

Advisor: doc. RNDr. Pavel Matula, Ph.D.

Department of Visual Computing

Brno, Fall 2023

MUNI
FI

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Attila Zsíros

Advisor: doc. RNDr. Pavel Matula, Ph.D.
Consultant: Mgr. Miroslav Krajíček

Acknowledgements

I thank my advisor doc. RNDr. Pavel Matula PhD. and consultant Mgr. Miroslav Krajíček for their kind and insightful advice during our meetings.

I thank my parents for their abiding support throughout my studies. I also cannot forget to thank my dear friends and flatmates, Ľuboš and Marek, for making the whole journey of writing this thesis just a bit more enjoyable and fun.

Abstract

Sports match recordings are essential for teams not only to stream their matches but also to identify areas for improvement through analysis of various events in the match. While larger clubs take the availability of their match recordings for granted, smaller clubs often do not have the human and financial resources to acquire them. However, given a static wide-angle recording of the full pitch, it is nowadays possible to emulate a human camera operator to some degree by cropping out the important region of each frame using computer vision techniques. These automatic virtual cameras are, however, primarily commercial black-box solutions. This thesis aims to develop a new, open prototype intended for soccer matches. The system uses a fine-tuned player-ball detector, a virtual pan-tilt-zoom (PTZ) camera, and a novel algorithm for estimating the camera PTZ based on the detections. The system is evaluated on a match recording from the Slovak First Football League by comparing the estimated footage with the footage from the broadcast camera. The results show that the system produces smooth camera movements with minimal jitter while keeping the action in the frame throughout all key moments of the match.

Keywords

virtual camera, image distortion, gnomonic projection, object detection, object tracking, kalman filter, particle filter, PID controller

Contents

Glossary **1**

1 Introduction **2**
- 1.1 Problem Description . . . 2
- 1.2 Contributions . . . 3
- 1.3 Thesis outline . . . 3

2 Related work **4**
- 2.1 Commercial Solutions . . . 4
- 2.2 Academic Research . . . 7

3 System Overview **9**
- 3.1 Pipeline . . . 9
- 3.2 Application . . . 9
- 3.3 Applicable Datasets . . . 11
- 3.4 Top Down Module . . . 12

4 Detector Module **14**
- 4.1 Player-Ball Detection In Literature . . . 14
- 4.2 Detector Model . . . 15
  - 4.2.1 Training . . . 15
  - 4.2.2 Training Results . . . 16
- 4.3 Detection Pipeline . . . 18

5 Cameraman Module **21**
- 5.1 Requirements . . . 21
- 5.2 Object Tracking using Kalman and Particle Filters . . . 22
  - 5.2.1 Object Tracking as Sequential Bayesian Filtering 23
  - 5.2.2 Using Kalman and Particle Filters for Object Tracking in Literature . . . 23
  - 5.2.3 Tracking the Players' Centroid and Estimating the Camera Target using Kalman and Particle Filters . . . 24
- 5.3 Camera Update Algorithm . . . 25

6 Virtual Camera Module **29**

6.1 Image Distortion 29
6.1.1 Optical Distortion 29
6.1.2 Perspective Distortion 30
6.2 Distortion Correction 30
6.2.1 Optical Distortion Correction 30
6.2.2 Perspective Distortion Correction 31
6.2.3 Pitch Tilt Correction 35
6.3 Movement Smoothing 36
6.3.1 Requirements 36
6.3.2 PID Controller 37
6.3.3 PI Controller 37
6.3.4 Parameter Tuning 38

7 Evaluation **39**
7.1 TrnavaZilina Dataset 39
7.2 Full Match Evaluation 41
7.2.1 Metric 41
7.2.2 Algorithm 41
7.2.3 Results 44
7.3 Clips Evaluation 44
7.3.1 Results 46
7.4 Technical Details 47
7.5 Discussion 48

8 Conclusion **50**
8.1 Future Work 50

Bibliography **53**

List of Tables

4.1 Universal detector training results. 16
4.2 Ball detector training results. 16

7.1 Mean IoU of different algorithm variants. 44
7.2 Clips evaluation results. 46

List of Figures

2.1 Examples of industrial sports cameras: Veo Cam 2 (left) and Pixellot (right). 4
2.2 Veo Cam 2 [2] Mounted on a Tall Tripod at the Center Line of a Soccer Pitch 6

3.1 The application pipeline. 10
3.2 SoccerTrack dataset sample. 11
3.3 Top-down view of the pitch (bottom) with its corresponding ROI (top). 13

4.1 `YOLOv8n_640` confusion matrix. 17
4.2 The detection pipeline. 19
4.3 Comparison of SoccerNet-Tracking (left) and TrnavaZilina (right) dataset image samples. 20

6.1 Example of barrel (left) and pincushion (right) distortion with annotated grid lines. 30
6.2 Visualization of the gnomonic projection. 32
6.3 Algorithm for virtual camera sampling with perspective distortion correction using only 4 points. 34
6.4 Pitch tilt correction demonstration. 36
6.5 Block diagram of the PI controller. 37

7.1 The TrnavaZilina recording setup. 40
7.2 Visualization of the evaluation metric: IoU of the top-down projected estimated (main) and reference (VAR) frames. 42
7.3 Calibration results of VAR full match recordings. 45
7.4 False positives *holding back* the camera. 49

Glossary

CNN Convolutional Neural Network

DL Deep Learning

field of view Is the extent of the scene covered by the camera lens. While it is often used interchangeably with the term *angle of view* (as is the case also for this thesis), they are not the same. The field of view refers to *the distance* (*in meters*) *covered by the projection at a certain distance* [1], and the angle of view refers to the angle (in degrees) of coverage

FOV field of view

FPS frames per second

IoU Intersection over Union

KF Kalman filter

LSTM Long Short-Term Memory

mAP Mean Average Precision

MOT Multi-Object Tracking

PF Particle filter

PTZ pan-tilt-zoom

rectilinear image Is an image produced using rectilinear lenses, i.e., straight lines in reality are straight also in the image.

rectilinear lens Is a lens preserving the straightness of lines.

ROI region of interest

SOTA state-of-the-art

1 Introduction

Sports match recordings are indispensable for teams not only for broadcasting their matches but also for coaches to analyze the match events and, with the aid of visualizations and drawings, to communicate their insights to the players. While large sports clubs take these tools for granted, smaller clubs often do not have the human and financial resources to acquire video recordings of their matches. This complicates not only the post-match analysis but also the scouting of young talents, who are often emerging from youth and semi-professional leagues.

With the increasing availability of high-resolution cameras and the advances in computer vision in recent years, it is nowadays possible to record the whole match on one wide-angle camera and crop out the important area of the play. Even though a handful of commercial solutions already exist, they are mostly black-box. The assignment for this thesis has been given by Goal Sport Technology, a company providing technology for sports stadiums, aiming to expand its portfolio with this kind of solution. Using a self-acquired dataset, this thesis applies techniques from various domains, such as projective geometry, control theory, probability theory, or computer vision, to develop a prototype for an automatic virtual soccer camera. The developed system is then evaluated by comparing the estimated footage to the broadcast footage of the same match.

1.1 Problem Description

Given a static video stream of a soccer match covering the full pitch, estimate a region of interest (ROI) for each frame, such that a video composed of these ROIs resembles a video recorded by a human camera operator.

The ROI should capture the *action* of the game in a *pleasing* framing. The *action* of the game includes the ball and the nearby players. For a *pleasing* framing, the ROI should imitate the movement of a real camera by changing its pan-tilt-zoom (PTZ) smoothly (avoiding jitter) and by correcting for the perspective distortion caused by the wide field of view of the frame. Since the input video is a video stream, only the current and the past frames are available for processing. The system should also provide a top-down view of the pitch with marked locations of players and the ball. While real-time performance is not required, the system should be kept as lightweight as possible.

1.2 Contributions

The contributions of this thesis include:

- A survey of related work on automatic virtual cameras.
- A player-ball detector trained on the SoccerNet-Tracking dataset.
- The Virtual Camera module with smooth movement across a rectilinear image, simulating the PTZ of a real camera.
- The Cameraman module for estimating the PTZ of the virtual camera based on the current situation on the soccer pitch.
- An evaluation algorithm for comparing the estimated and broadcast footage of the same match.

The Detector, the Virtual Camera, and the Cameraman modules are integrated into one standalone application attached to this thesis. The evaluation algorithm is also attached as a separate application.

1.3 Thesis outline

The thesis is structured as follows. Chapter 2 presents related work on automatic virtual cameras in industry and academia. Chapter 3 first gives a high-level overview of the developed system pipeline and its implementation. Next, it discusses the applicable datasets for this kind of application. Last, it briefly describes the Top Down module. The following chapters then give more details about the three main application modules: the Detector (Chapter 4), the Cameraman (Chapter 5), and the Virtual Camera (Chapter 6). In Chapter 7, the system is evaluated using a self-acquired dataset on a full-match recording and on highlight clips. Finally, in Chapter 8, the results are summarized and ideas for future work are presented.

2 Related work

Due to the large commercial interest in automatic sports cameras, many advanced solutions have been developed in the sports industry. Therefore, in this chapter, we first summarize the currently available commercial solutions and afterward give an overview of related academic work.

2.1 Commercial Solutions

Plenty of companies are already making this kind of product[1]. Under the motto *democratize the recording of soccer*, they primarily target amateur soccer clubs to allow them to stream their matches and training sessions without needing a camera operator. Some companies provide only software; others have also developed their hardware setups, which, as Chapter 6 shows, may be a great advantage when correcting image distortions. While there exists a large number of solutions, the following paragraphs focus on a small selection of them to give a picture about their most common features.

Once Once, the product of the Croatian company Once Sport, is a software-only solution. It accepts a video covering the full pitch as input and does the processing offline on the computer. Apart from automatically estimating the ideal crop-out, its software focuses on

1. Examples of companies: Once, Veo, Pixellot, Spiideo, Provispo, Playsight.

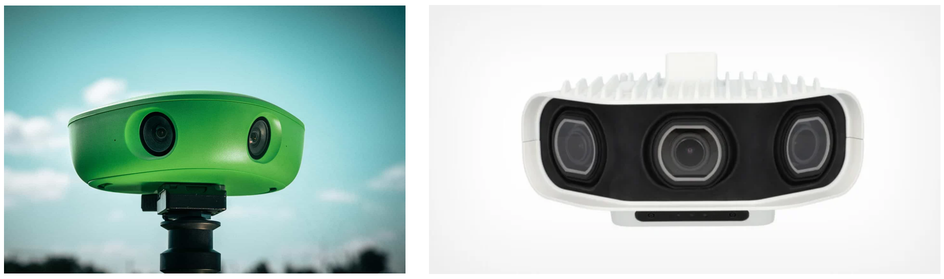


Figure 2.1: Examples of industrial sports cameras: Veo Cam 2 (left) and Pixellot (right).

tracking players and provides various tools for drawing and visualizations. Although it supports live streaming, e.g., on YouTube, it is only available if there is enough computing performance to process the video live. Once recommends using a single wide-angle camera, such as a CCTV camera or sports cameras like DJI Osmo Action 3, SJ8 PRO, or INSTA 360 RS. The advised placement is at least 6 meters above the ground, mounted on a tall tripod or light stand. However, since Once accepts only a single video stream, it does not provide stitching multiple videos of cameras placed next to each other. In such case, it recommends first stitching the videos using a tool like Actionstitch[2] and then processing the stitched video. The pricing is subscription-based at the cost of 32 EUR per month.

Veo The Copenhagen-based company Veo Technologies has developed a two 4K camera setup encased in a 3D-printed weatherproof box, called Veo Cam 2 [2] (see Figure 2.1). After mounting it on a tripod just outside the halfway line of the pitch and raising it above 4 meters (Figure 2.2), it can reach a 180-degree panoramic field of view of the entire pitch. Using the mobile app, Veo Cam 2 can be connected to the internet and live-stream a match (with a latency of 45 seconds).

However, in contrast to Once, the actual video processing and cropping are realized in the cloud, so an internet connection is necessary to preview the final video. After uploading the footage to their cloud platform and waiting for it to be processed, tools for cropping videos, drawing annotations, or tagging game highlights become available. Some highlights are already AI-suggested; the user can delete or modify them if desired. These highlights can then be saved into a player's profile, which can be easily shared by a link, e.g., for recruiting purposes.

Apart from video editing, the cloud platform offers a number of analytics, such as ball possession, shot map, pass strings, shots on goal, or game momentum. When multiple matches are already recorded, these statistics can be shown aggregated, giving the team valuable insights about their progress in a given area of their play. Being one of the most popular solutions in this field, they also have an extensive blog section on their website and even an educational platform called

2. https://actionstitch.com

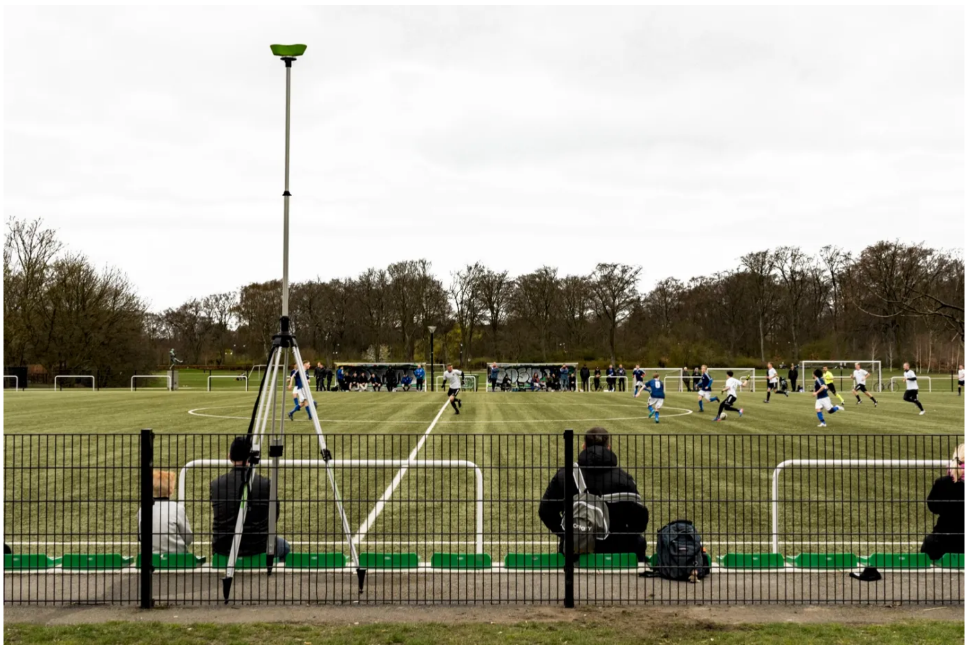


Figure 2.2: Veo Cam 2 [2] Mounted on a Tall Tripod at the Center Line of a Soccer Pitch

Veo Academy, with how-to guides to their products. Their business model is subscription-based, starting at 39 EUR per month, with a one-time expense for the camera of 999 EUR.

Pixellot A similar product comes from the company Pixellot. In May 2023, they released a new generation of their camera, the Pixellot Show S3 [3]. In contrast to their last generation and other solutions on the market, it uses three 4K cameras instead of two, producing a 12K video. Since all the processing is again realized in the cloud and not in real-time, such a large resolution is not an issue, and the video quality can only benefit from it. Pixellot is also being marketed as a solution suitable not only for soccer, but also for other sports, such as basketball, ice-hockey, baseball, softball, American football, rugby, volleyball, cricket, or handball.

To conclude, this is a competitive business segment with companies trying to set themselves apart by introducing new features and by increasing the resolution and quality of their videos. Nevertheless, we can find a few areas that they share. First, they provide their own video editors with tools for video trimming, annotation drawing, or highlight tagging. Second, they often provide two viewing modes: a broadcast view for a more detailed, zoomed-in view for spectators, and a tactical view, giving a broader overview of the happening on the pitch for coaches. Third, they extensively use AI for object detection, object tracking, and action-spotting for tagging and suggesting game highlights. While many of the solutions claim to succeed at separating the two teams from each other, tracking each individual player remains still a challenge.

The prices of these solutions vary widely; it was challenging in our research to conclude whether the price tag is directly proportional to the quality of the footage. This was mainly due to a lack of footage published by the companies, and we could not discard the bias of a companies showing only a showcase of the selected best footage on their websites. Also, while the more expensive solutions were publishing footage recorded at more semi-professional soccer stadiums, the cheaper ones were primarily recorded at amateur soccer pitches. However, what the footage across all solutions had in common was a relatively slow and stable camera movement.

2.2 Academic Research

In academic literature, little research has been conducted on developing automated camera systems. Here, we list the methods that are most related to our system. Not all of them, however, target sports applications. For example, Konstantinos et al. developed a web service for cropping (general) videos into a target aspect ratio [4]. To determine the target center of view, they use saliency maps of video frames. Another indicator of a region's importance might be the viewer's gaze, as proposed by Rachavarapu et al. [5]. Daigo et al. use gaze tracking in the context of sports broadcasting [6]. With the aim of creating an automatic pan control system, they move the camera by following the audience's face direction. The resulting images are obtained by sampling from a cylindrical panorama stitched from three cameras. While the system is evaluated on basketball footage, the authors claim that it is also applicable to other ball games. However, given the scale of a soccer pitch and that this thesis is using only a single camera (often placed behind the coaches), it would be hardly applicable in our context.

Another work demonstrated on a basketball game has been published by Chen et al. in a solution for controlling a real robotic camera [7]. The PTZ camera coordinates are estimated based on features derived from player tracking without the need to track the ball. A linear regressor is trained on feature representations of professional broadcast footage. In contrast, instead of using linear regressors and player positions only, Pidaparthy et al. detect also the optical flow and use it in a deep network regressor to determine the optimal camera PTZ in hockey scenes [8]. In his master's thesis, Bayrak presents another Deep Learning (DL) approach, using a Long Short-Term Memory (LSTM) as a regressor [9]. First, footage from two 4K cameras is stitched with the aid of ArUco markers. Next, a Convolutional Neural Network (CNN) extracts spatial features, which are then fed to the LSTM. Finally, a fully connected layer outputs the ROI center coordinates. However, approaches using regressors for PTZ estimation need quality training data obtained by experienced camera operators, which is often difficult to get.

During our research, we found the work of Gaddam et al. to be the most related to ours. In [10], they develop a distributed system of 5 cameras stitched together into a cylindrical panorama. Similar to [6], they created a virtual camera for sampling from the panorama to correct for perspective distortion. While in this paper, the camera is controlled by the user, in their later work, they attempt to fully automate the camera movement [11]. In contrast to [7] and [8], they focus less on determining the best ROI center and more on ensuring natural camera movement. Different camera control algorithms are evaluated in a user study. Since this thesis achieves perspective correction by sampling from a sphere instead of a cylinder, we find relevant also the work by Yu et al. [12]. They designed so-called Perspective Crop Layers (PCLs), which compensate for perspective distortion for improving DL model accuracy, as demonstrated on pose estimation. These PCLs can be inserted into existing CNN and MLP (Multilayer Perceptron) architectures without changing the number of parameters or the training process.

3 System Overview

As the main contribution of this thesis, an automatic virtual camera system was developed. This chapter first presents a high-level overview of the system pipeline that each input frame undergoes. Next, it provides information about the developed application and the datasets used during development. Last, it briefly explains the idea behind the application's Top Down module. The explanation of the remaining modules is left for the following chapters.

3.1 Pipeline

The input to the system is a static wide-angle video stream of a soccer match covering the full pitch. Each frame goes through a series of stages shown in Figure 3.1.

First, the Detector module detects the players and the ball. In the next stage, the Cameraman module uses the detections to estimate the appropriate PTZ of the virtual camera for the current situation on the field. Then, the Virtual Camera module calculates the ROI corrected for perspective distortion and crops out that region from the input frame. Based on the detections and the virtual camera viewing cone, the Top Down module renders a top-down (bird-eye) view that can be later used for analysis purposes. Finally, the virtual camera output frame is written to the output video, and the next frame is retrieved. This process is repeated until the end of the video stream.

3.2 Application

The application is written in Python[1], a popular language for prototyping in the computer vision community. Various libraries are used: OpenCV[2] for image preprocessing, drawing, and IO operations;

1. https://www.python.org/
2. https://opencv.org/

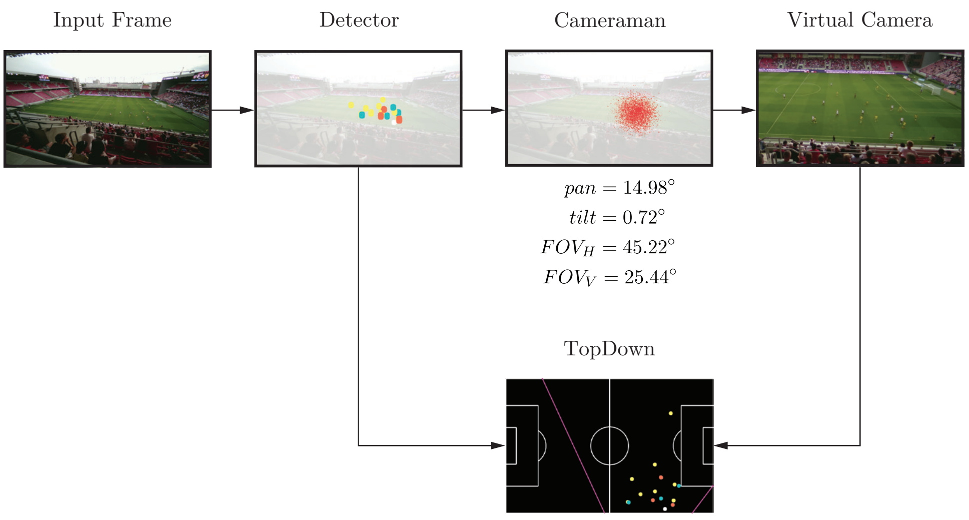


Figure 3.1: The application pipeline.

Numpy[3] for data structures and mathematical operations; SciPy[4] for statistics; and PyTorch[5] and Ultralytics[6] for DL training and inference.

The application accepts a soccer recording as input, automatically finds each frame's ROI, and outputs a new video consisting of these ROIs. There are two configuration files: the system and the dataset configuration file. The system configuration file provides options for debug visualizations or camera preferences, such as camera sensitivity. The dataset configuration file specifies the coordinates of the pitch corners, a necessary information for the Top Down module (see Section 3.4), and virtual camera options for the main camera default PTZ or the PTZ of the three cameras for the frame splitting in the detector pipeline (see Section 4.3).

The code is attached in the attachments of this thesis, along with sample videos from the evaluation dataset (presented in Section 7.1). For installation and usage instructions, refer to the `README` file and the rich code documentation.

3. https://numpy.org/
4. https://scipy.org/
5. https://pytorch.org/
6. https://www.ultralytics.com/

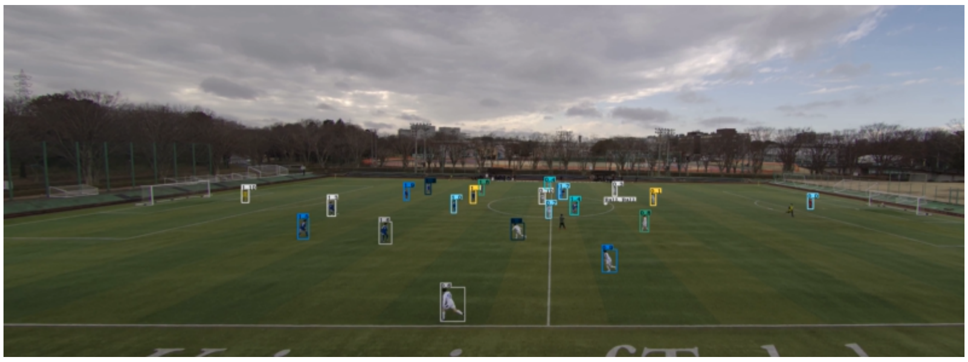


Figure 3.2: SoccerTrack dataset sample.

3.3 Applicable Datasets

During the development of this system, it was essential to have a soccer dataset with static footage covering the full pitch. However, we managed to find only a single suitable open dataset: the SoccerTrack dataset [13].

It is a Multi-Object Tracking (MOT) dataset, i.e., it contains bounding box annotations of players and the ball. However, the annotations were obtained in an unconventional way: the authors replaced the laborious manual annotation task with a semi-automatic annotation procedure using Global Navigation Satelite System (GNSS) player trackers and a bird-eye view. The 8K wide-angle main camera recording is thus complemented by GNSS coordinates from player trackers and a 4K drone video. The GNSS coordinates allow precise automatic drone video annotation of bounding boxes, which are then easily converted into the main camera view using homography (similar to our Top Down module in Section 3.4).

Even though the annotations are not used in this thesis, since our detector is trained on a different dataset (see Chapter 4), this dataset was used in the early stages of the application development. However, due to the low camera placement (see Figure 3.2) and no reference footage for evaluation, there was still a need for a different dataset. Since this was the only applicable open dataset of this kind, we decided to acquire our own as shown in Section 7.1.

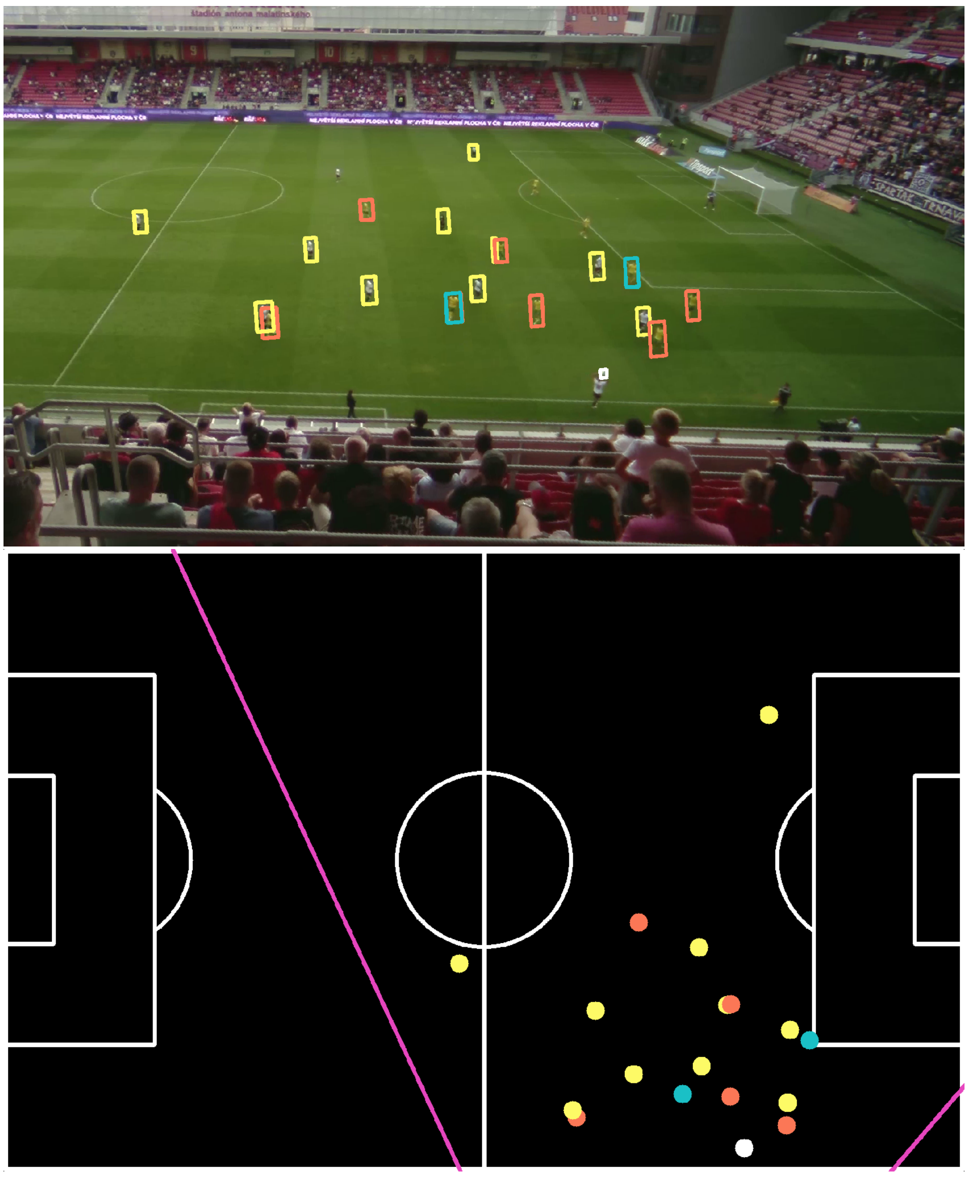
3.4 Top Down Module

The Top Down module uses data from the Detector and the Virtual Camera modules to render a schematic view of the soccer pitch with marked locations of players, the ball, and the viewing cone of the virtual camera. It is the least complex module in our application. Therefore, instead of devoting a whole chapter for the module description (as is done for the rest of the modules), this section briefly summarizes the method used for creating a top-down view.

The task is to find a mapping between two images (main and top-down) of the same 3D object (the pitch). If we simplify this task by assuming that the captured object is planar[7], we are then searching for a projective mapping between two images of a plane, i.e., a homography [14]. It is a $3x3$ matrix with 8 degrees of freedom. The OpenCV library function `findHomography` is used, where the input points are the pitch corner coordinates of both images. The function computes the homography matrix by minimizing the reprojection error, using methods such as least squares, RANSAC, or Least-Median robust method[8]. The returned homography then transforms points from the original to the top-down frame space.

For the bounding boxes of detections, the point to be transformed is the center point of the bottom edge of the bounding box. For the viewing cone, we take the four corners of the virtual camera ROI in the original frame space, clip them to the pitch edges (for numerical stability), and transform these clipped ROI corners to the top-down view. This creates a characteristic trapezoid of the horizontal field of view of the camera, as seen in Figure 3.3.

7. In reality, however, depending on the stadium, the center of the pitch is usually located higher than the corner points, making the pitch rather conic than planar. Nevertheless, these elevation differences are small relative to the pitch size and the camera distance to the pitch, allowing us to treat the surface as planar in this thesis.
8. https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html



The top-down view (bottom) shows the viewing cone of the virtual camera ROI (violet), the players (teal), the referees (yellow), and the ball (white). Note that the misdetections of players with yellow jerseys marked as referees is because the referees in detector training data were mostly yellow (see Section 7.1).

Figure 3.3: Top-down view of the pitch (bottom) with its corresponding ROI (top).

4 Detector Module

The Detector module aims to detect the players and the ball, which is essential information for the following stages in the application pipeline. This chapter first discusses relevant player-ball detection approaches. After fine-tuning a state-of-the-art (SOTA) object detector, it discusses the training results and trained model deficiencies. The last section of this chapter presents the detection pipeline that uses frame-splitting for converting input frames into a suitable detector format.

4.1 Player-Ball Detection In Literature

In literature, player and ball detection have been studied extensively. The methods can be divided into classical and deep learning methods.

The classical methods commonly use a combination of background subtraction using the Gaussian Mixture Model, feature extraction using the Histogram of Oriented Gradients, and classification using Support Vector Machine [15, 16]. For ball detection in particular, circular object detectors, such as the circular Hough Transform, can be utilized [17]. However, due to the large variability of grass textures and patterns in soccer, DL methods often outperform the classical ones.

DL-based object detection has been traditionally done by 2-stage detectors, such as R-CNN, where the first stage is responsible for region proposal and the second stage for classification [18, 19]. However, single-stage detectors, such as YOLO [20] or SSD [21], have shown that both stages can be unified and achieve better performance and high accuracy at the same time. While there are some specialized methods for detecting small objects at high speed in soccer [22] or in tennis [23], there has been rapid progress in object detection in recent years [24]. Thus, we decided to use a SOTA general-purpose object detector developed by Ultralytics—YOLOv8[1]—in a transfer learning setting[2].

1. https://github.com/ultralytics/ultralytics
2. *Transfer learning and domain adaptation refer to the situation where what has been learned in one setting ... is exploited to improve generalization in another setting.* [25]

4.2 Detector Model

YOLOv8 provides a high inference speed in a user-friendly interface with models for object detection and tracking, instance segmentation, image classification, and pose estimation. This subsection shows how we adjusted the general object detector model to soccer scenarios.

The object detection model is pre-trained on the COCO dataset[3] on 80 object categories, such as *person*, *car*, *bicycle*, or *sports ball*. If a broadcast soccer video is run on this model, all detected players in the image are assigned the label *person*, and the network may also detect other potentially unwanted objects in the image, such as spectators or staff. Our goal is to train the network to differentiate between players, goalkeepers, referees, and the ball and, if possible, to minimize the detection of other objects. Thus, we fine-tuned this object detection model by training it on an open MOT dataset as follows.

4.2.1 Training

YOLOv8 is available in different model sizes. Considering this thesis's emphasis on inference speed and time constraints, we decided to work only with the smallest, *nano* model, further denoted as `YOLOv8n`. In this section, we describe the setup and results of the training.

Dataset We used the SoccerNet-Tracking[4] dataset provided by SoccerNet[5]. It is a MOT dataset of broadcast soccer footage, i.e., the camera is following the action by panning, tilting, and zooming by a camera operator. The dataset comprises 30-second annotated clips, which are split into 57 train and 49 validation clips, making a total duration of 28.5 minutes for the training set and 24.5 minutes for the validation set[6].

3. https://cocodataset.org/
4. https://github.com/SoccerNet/sn-tracking
5. SoccerNet is a group organizing open challenges for soccer video understanding, see https://www.soccer-net.org/
6. There are 94 more clips that the organizers keep for challenge purposes.

Table 4.1: Universal detector training results.

Model	mAP@50	mAP@50-95	Precision	Recall
`YOLOv8n_640`	0.65318	0.37711	0.78737	0.60251
`YOLOv8n_960`	0.68649	0.40958	0.81395	0.64109

Table 4.2: Ball detector training results.

Model	mAP@50	mAP@50-95	Precision	Recall
`YOLOv8n_640_ball`	0.29939	0.09803	0.5098	0.30877
`YOLOv8n_960_ball`	0.41768	0.14373	0.58763	0.42506
`YOLOv8n_1280_ball`	0.4542	0.15999	0.61717	0.47156

Technical setup The training was run on a remote computing server equipped with AMD EPYC 7702P 64-Core CPU, 512GB RAM, and NVIDIA A100 80GB GPU.

Parameters The model was trained with a batch size of 32 with the SGD (Stochastic Gradient Descent) optimizer with momentum and variable learning rate. The rest of the parameters were left default as listed on the Ultralytics website[7] and attached to this thesis.

4.2.2 Training Results

First, we trained two universal detectors for all object classes (i.e., ball, player, goalkeeper, and referee) with varying input image sizes: one with 640 pixels and one with 960 pixels on the longer edge. From the resulting matric values listed in Table 4.1, it is visible that the larger image size increases the overall quality of detections. However, the model struggles most with detecting the ball, as the confusion matrix shows in Figure 4.1. This is inevitable, given the small ball size and its fast movement speeds with motion blur; however, it is still important for our application to have the best ball detector possible. Therefore, we also trained a specialized ball detector with the intention to be used alongside the universal detector.

7. https://docs.ultralytics.com/usage/cfg/#train

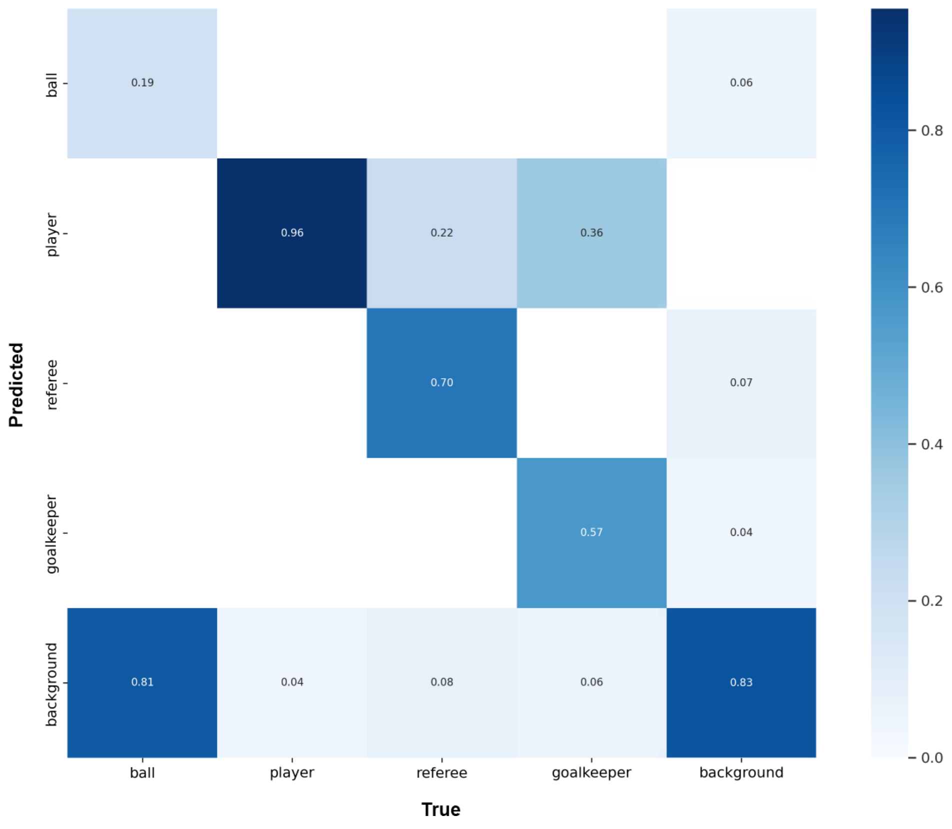


Figure 4.1: `YOLOv8n_640` confusion matrix.

For the ball detector, we trained three models with different image sizes, ranging from 640 to 1280 pixels. The mAP@50 column in Table 4.2 shows that while increasing the image size from 640 to 960 pixels is a significant improvement (39.7%), increasing it further to 1280 means only a slight improvement (8.7%).

Discussion From the results and our observations during the model inference on sample videos, we can conclude that the trained models provide overall satisfactory player detections with rare false positives outside the pitch, given the small model size and small training set.

One source of problems for the universal models is the label assignment. It is expected, since players in one match might wear the same color as a referee wears in another match. Even the ball is not always white but can be orange in winter conditions. Therefore, the next parts of the application pipeline should account with this detector deficiency and not rely on the label assignment, e.g., by treating all detections (except for the ball) as players, as done in Chapter 7. Another solution might be using a player clustering technique as presented in [26] and [27].

We can also conclude that it is worth using a larger input image size than 640 pixels. However, increasing the image size comes for the cost of worse model generalizability (leading to overfitting) and increased computational complexity. Thus, as the results also indicate, the a size of 960 pixels might be the optimal image size for our detector.

4.3 Detection Pipeline

Despite having a trained object detector, it is not yet ready for use with the original full-pitch frame as input. Thus, we introduce the detection pipeline, as shown in Figure 4.2 and described in this subsection.

First, it is critical for our application to avoid the non-relevant detections of objects outside of the pitch. Even though the trained detectors can achieve this to some degree, we ensure it by masking out everything except for the pitch. Moreover, some user-defined margins are added to exclude the pitch areas near the borders where managers and referees usually stand. These margins can be also used to add some area outside the pitch to detect players near the pitch borders.

Second, we want to adjust the detector input frames to resemble the training data as closely as possible. Since we use a detector trained on regular broadcast footage, our original wide-angle frame cannot be considered a suitable input image, mainly because the player and the ball sizes between the training and the inference data differ widely (see Figure 4.3).

Therefore, the frame gets split into three other frames, which, combined, cover the full pitch. These frames represent the frames of three virtual cameras with predefined PTZ coordinates, dependent on the used dataset.

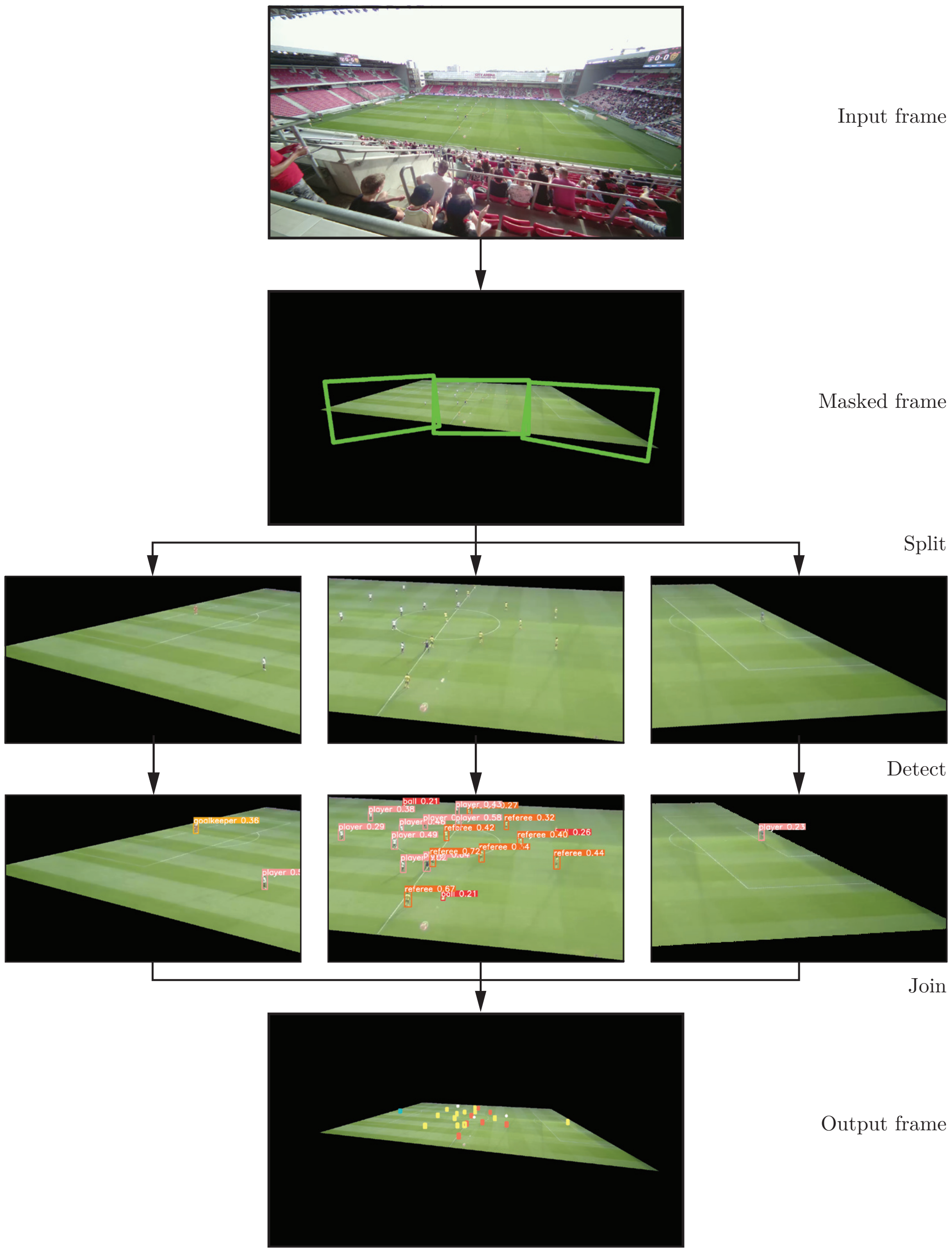


Figure 4.2: The detection pipeline.

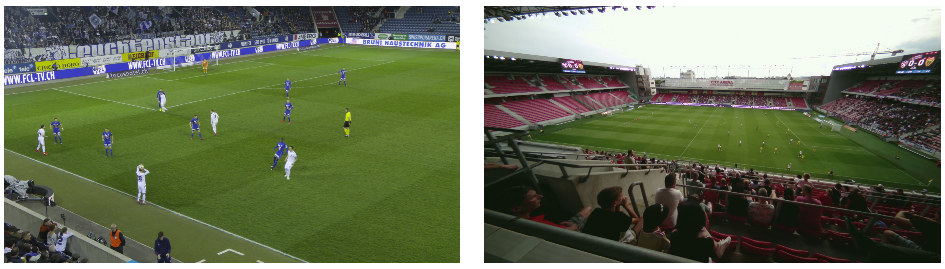


Figure 4.3: Comparison of SoccerNet-Tracking (left) and TrnavaZilina (right) dataset image samples.

The three frames are fed to the detector in a batch, which allows the detection of these three frames to be run in parallel on a GPU. The resulting bounding boxes of players, referees, and balls are finally joined into one final array of detections represented in the original frame pixel coordinates.

The system allows the user to define areas in the frame that should be excluded from the detection either by using the pitch mask margins mentioned above or by defining such rectangular regions in the configuration file. This filtering is done right after the join step.

5 Cameraman Module

The creative part of the developed system lies in this module. It aims to estimate an ideal framing, such that it not only contains all the necessary objects (the ball and the nearby players) but is also pleasing to watch. This is a challenging task that requires to exploit as much information from the input frame as possible.

This chapter starts by listing our requirements for this module. Next, the tracking problem is formulated as a Bayesian filtering problem. Last, the full algorithm is presented with a detailed description of each step.

5.1 Requirements

A naive approach would be to keep the ball in the center of the frame constantly. While this would be sufficient for interplay in the middle of the pitch, as we analyze the different scenarios of a soccer game, we quickly find cases where it would not provide satisfactory results. For example, we do not want the ball to be framed in the center during a corner kick, when one team fully occupies the opponent's half, or during a counterattack, where it is desired to pan the camera in the direction of the attack proactively. Consequently, our goal is to target the camera center somewhere between the ball position and the players' centroid[1].

This is, however, only under the assumption that the ball is detected. Nevertheless, as shown in Chapter 4, it is often not the case since ball detections are unreliable, missing or containing false positives due to occlusions, high speed, bad lighting, or other factors [29]. In such cases, we want to at least estimate the ball position in a way that considers past measurements by utilizing the ball's direction and velocity. The same applies to the players' centroid. In other words, we expect the solution to be robust to noisy or missing ball and player detections.

1. An even more suitable information would be to know the movement vector of each player, as in [28]. This would, however, require tracking each player's movement, which is a complex problem on its own, given the visual similarities of teammates [29]. During our development, we tried using two state-of-the-art trackers ([30], [31]), but they did not produce satisfactory results on our dataset.

Apart from the camera orientation, another essential factor of pleasing framing is the zoom level, where it is a trade-off between a more zoomed-in (immersive) view or a zoomed-out (tactical) view. The zoomed-in view is usually preferred for broadcasts because it provides a more detailed view for the fans. The zoomed-out view captures the majority of the players most of the time, giving valuable insights to coaches. It is also sometimes preferred among the fans, giving them more room for game analysis than the immersive view. Because both views have their advantages and disadvantages, many commercial solutions provide both views for the user to select (as shown in Section 2.1). Nevertheless, even in the tactical view, having a too-wide field of view (FOV) is not desired because it would mean capturing empty parts of the field. In our case, since the zooming is done digitally, the input footage resolution limits the maximum zoom level, and thus, we aim to implement a more tactical view.

The final product of this kind should be able to do a live match broadcast. Thus, the input footage is considered a video stream, i.e., only the past frames are available when processing a given frame.

5.2 Object Tracking using Kalman and Particle Filters

The requirements mentioned above indicate that there are two essential features to track: the ball position and the players' centroid. Since we are concerned with the problem of tracking, we employ well-established tracking approaches: the Kalman filter (KF) for tracking the players' centroid, and the Particle filter (PF) for estimating the camera target by fusing multiple measurements, including the ball position and the players' centroid.

This section first formulates the problem of tracking as a Sequential Bayesian filtering problem, introducing the KF and the PF. Next, it summarizes their usage for object tracking in literature. Finally, it explains their properties and our reasoning behind choosing them for tracking the players' centroid and the camera target.

5.2.1 Object Tracking as Sequential Bayesian Filtering

The problem of tracking can be defined as the task of recursively estimating the state x from measurements z [32, 33]. This task can be transferred into the Bayesian context, which aims to construct the state probability density function at time t given all past measurements as

$$p(x_t|z_{1:t}) \propto p(z_t|x_t) \int p(x_t|x_{t-1}) p(x_{t-1}|z_{1:t-1}) dx_{t-1}, \tag{5.1}$$

where $p(z_t|x_t)$ is the last observation likelihood and $p(x_t|x_{t-1})$ is the state transition probability.

The algorithm proceeds in two steps: *predict* and *update*. The predict step uses a system motion model for predicting the next state, accounting for some random noise due to unknown disturbances affecting the system. The update step uses the gathered measurements to correct these predictions. If no measurement is available at the current step, only *predict* is called. Due to its recursive nature, only the last state and measurement must be stored, making it a lightweight algorithm.

There are several techniques for solving Equation (5.1) using various assumptions. The two most common algorithms are the KF [34] and the PF [35]. The KF assumes that the system motion model is linear and the noise is multivariate Gaussian distributed. The PF, on the other hand, does not use such assumptions and is therefore suitable for non-linear systems with unknown noise distribution. Also known as the Sequential Monte Carlo method, bootstrap filtering, or survival of the fittest, the PF approximates the true distribution using a set of particles as weighted samples of the distribution.

5.2.2 Using Kalman and Particle Filters for Object Tracking in Literature

In literature, the KF and the PF are used extensively for tracking. Chen et al. [36] used two KF trackers for a single object (table tennis ball) in horizontal and vertical directions separately. Hamuda et al. [37] used one Kalman filter per detection in a Multi-object tracking scenario, where the Hungarian algorithm is used for associating detections with their trackers. Even some DL-based object trackers use the KF internally, achieving SOTA results for pedestrian tracking [31, 30].

Pérez et al. [38] used the PF in combination with histogram matching, which provided more robust detections. Due to its robustness to fast-moving objects and occlusions, it has also been used multiple times for ball tracking [39, 40].

5.2.3 Tracking the Players' Centroid and Estimating the Camera Target using Kalman and Particle Filters

In our application, we employ the Kalman filter for tracking the players' centroid and the Particle filter for tracking the camera target by fusing multiple measurements, including the ball position and the players' centroid.

Players' Centroid Tracking Using the KF As shown in the camera update algorithm below (Section 5.3), we calculate the players' centroid in each frame. This centroid is used not only in assigning weights to the particles in the PF but also in calculating the PF control input by providing information about the players' mean direction and velocity. Obviously, for estimating the players' mean velocity, it is important to reduce the noise from the player detections caused by, e.g., bad lighting conditions. Since the player detections are relatively stable compared to ball detections (i.e., the centroid does not move as quickly as the ball), we assume the measurement noise for the centroid to be Gaussian distributed. Therefore, we track the players' centroid state (i.e., its position and velocity) using the KF with the constant velocity model (for a detailed explanation of the KF constant velocity model, refer to [41]).

Camera Target Tracking Using the PF In contrast to the players' centroid, the ball is a small, fast-moving object, often occluded by players, resulting in very noisy detections and frequent false positives. Consequently, we not only cannot assume Gaussian distributed error, but due to false positives, we also want to support multimodal distribution for the system state. Thus, the PF is a more suitable approach for ball tracking than the KF [40].

While we could use the PF for ball tracking by weighting the particles based on their distance to the ball detection and then use another tracker for estimating the camera center target as the weighted mean of the players' centroid and the ball position (considering the camera target requirements stated above), we decided to use a single PF for both as follows.

The tracked state of the PF is the camera center target; the PF update step is called whenever a ball measurement comes; and the error measure for the particles is based on their distance to the ball d_{ball} and to the players' centroid $d_{centroid}$ as

$$Error = \alpha d_{ball} + (1 - \alpha) d_{centroid}, \tag{5.2}$$

where α is a user-defined constant deciding whether we want the camera to be aimed more at the ball or the centroid. The larger the error, the lower the weight assigned to the particles. The particles are always normalized to yield a sum of 1. To prevent the weight degeneracy problem of the PF [42], we use *systematic resampling* (for a detailed description, refer to [43]).

One of the reasons for choosing the PF for camera center estimation is the dynamically changing variance of the particles, reflecting the confidence of the filter. The more frequently we update the filter with the detection matching the predicted state, the lower the variance of the particles will be (i.e., the more confident the filter will be in its estimates); consequently, the less it will be influenced by sporadic outliers (i.e., false positive detections).

5.3 Camera Update Algorithm

The input to the algorithm are bounding boxes of player and ball detections. Both the PF and KF are initialized by the players' centroid at the start of the application.

1. **Discard Goalkeeper Detections**

   First, we want to discard the goalkeepers from further processing. We achieve this simply by discarding the left-most and the right-most players[2].

2. A more sophisticated approach would be to use a detector that differentiates between players and goalkeepers.

2. **Calculate the Players' Centroid**

   Second, we calculate the players' centroid in the original frame space, which is valuable information in the succeeding steps. For example, we use it directly in the third step for calculating the control input to the particle filter.

3. **Calculate the Particle Filter Control Input**

   The control input is a vector along which the particles are moved during the predict step. This control vector is formed as a sum of two vectors: the players' movement vector (1) and the players' center vector (2).

   (1) The players' movement vector represents the direction and the speed of players' movement, created by subtracting the previous players' centroid from the current centroid. Its purpose is to proactively move the camera during a counterattack when all players are moving in a similar direction.

   (2) The players' center vector prevents the PF from drifting away when no ball detection is available for some time. It achieves this by slowly moving the particles toward the players' centroid after the PF variance exceeds a threshold. We assume that this is our best estimate in such a situation.

4. **Update the Particle Filter**

   If there are no ball detections, only the PF predict step is called. Otherwise, we update the PF with the ball detection centers and the players' centroid as follows.

   For each ball detection, we assign a weight to each particle based on its distance to the ball and the players' centroid. The maximum weight will be assigned to a particle that lies on the line connecting the ball and the players' centroid. We use an $\alpha \in [0, 1]$ that determines the position on the line that will be assigned the highest weight. Using the α, we can decide whether we want the ball to be in the frame's center or whether to center the frame at a point between the ball and the players' centroid. Afterward, the weights are normalized, and the particles are resampled using *systematic resampling* [43].

5. **Update the Kalman Filter**

   Next, we perform the predict step of the KF for estimating the players' centroid, and if there are any player detections, we also proceed to the update step.

6. **Calculate the Zoom Level** The zoom level is determined by taking the maximum[3] of two calculated FOVs: FOV based on the PF variance (1) and FOV based on the players bounding box width (2).

   (1) Intuitively, when the ball has not been detected for some time, our uncertainty of its true position increases, and we expect the camera to zoom out. In the opposite scenario, when there are frequent ball detections (from a similar location), we have the confidence to zoom in. This is what we attempt to do in this step, where, similarly as in step 3, we utilize the PF variance to observe whether the ball has been detected in the past few frames. Specifically, we define a zoom range for the virtual camera (its minimal and maximal allowed FOV) and a variance range for the PF (a minimal and maximal variance); we map these values on each other so that the minimal FOV is reached at the minimal variance, and vice-versa. The zoom level is then a linear interpolation within this range based on the PF variance.

   (2) The second contributing factor aims to correct the cases when the camera is too zoomed out, i.e., when the PF variance is high due to a lack of ball detections. The idea is that if all players are in the frame, the camera should not zoom out further. In other words, if no ball is detected, we want to fit all players into the frame and not zoom out more (as would be the case if relying only on the zoom calculated from the PF variance). We achieve this by calculating a bounding box encapsulating all player detections (except for the goalkeepers) with some added margins for a more pleasing framing. The width of the bounding box (in pixels) is then used for calculating the target FOV for the virtual camera.

3. A more conservative approach than taking the maximum of the two zoom estimates would be to take their average.

7. **Update the Virtual Camera**

   In the last step, we perform the actual update of the virtual camera, i.e., set the target of its PI controller (see Section 6.3). The target x and y are the coordinates of the weighted mean of all PF particles, and the zoom level (focal length) is derived from the estimated FOV. However, before the PI controller update, we first verify if the ROI defined by these new PTZ coordinates fits the original frame bounds. We update the PI controller only if this test is passed.

6 Virtual Camera Module

The Virtual Camera module aims to define a ROI around a target center point estimated by the Cameraman module and to transition between the current and the next target smoothly. A naive way of defining a ROI based on its center would be to just use a rectangle of a given aspect ratio. However, this would give poor results because it does not compensate for distortions caused by wide-angle lenses. This chapter first describes the problem of image distortion and ways of correcting them. Next, it presents the perspective distortion correction algorithm used in our application. Finally, it introduces the PI controller for achieving smooth camera movement.

6.1 Image Distortion

This section describes the two types of distortion that images can suffer from: optical distortion and perspective distortion [44].

6.1.1 Optical Distortion

Optical (also referred to as *curvilinear*) distortion is an optical error caused by curvilinear lenses. It makes physically straight lines appear curvy in the image. The most common types include the barrel and the pincushion distortion, shown in Figure 6.1. In barrel distortion, image magnification decreases from the optical axis towards the corners and is commonly seen in wide-angle lenses, where a wide FOV relative to the sensor size needs to be *squeezed*. In pincushion distortion, on the other hand, the magnification increases towards the corners and appears mainly in zoom lenses, where a small angle of view has to be *stretched* to fit the sensor. Curvilinear images appear unnatural because human vision does not bend lines. Lenses preserving the straightness of lines, making the images look natural, are called rectilinear lenses. Consequently, optical distortion is sometimes also defined as the deviation from rectilinear projection.

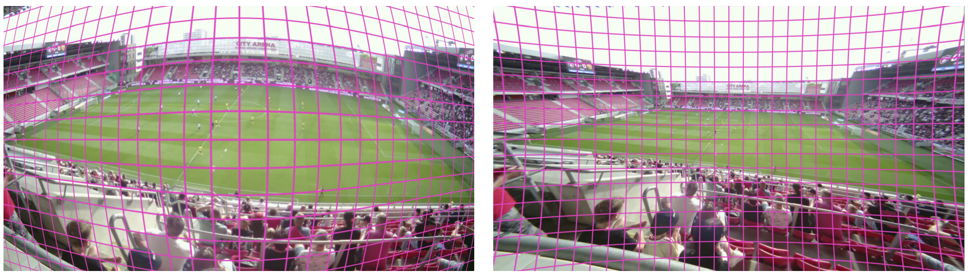


Figure 6.1: Example of barrel (left) and pincushion (right) distortion with annotated grid lines.

6.1.2 Perspective Distortion

Perspective (also referred to as *geometric*) distortion is the deviation of the observed image from the image captured with a normal lens[1] [45]. This results in warping objects in the image so that they appear smaller as their distance from the camera increases and appear (disproportionately) larger when placed too close to the camera. Thus, it is not an issue of optics but rather of the position of the camera relative to the captured subject. In other words, the wider the FOV of a lens, the more its image deviates from the normal FOV, and the more are the objects in the image warped [44].

6.2 Distortion Correction

To be able to simulate a realistic PTZ of a camera on a tripod, both types of distortions must be corrected.

6.2.1 Optical Distortion Correction

Removing optical distortion from an image is sometimes referred to as image undistortion. It essentially means turning it into a rectilinear image, i.e., straightening the curved lines that are straight in reality. For this, we need to know the distortion coefficients of the lens. A

1. An image produced by a normal lens resembles the FOV and magnification of human vision, thus appearing naturally, without space compression or bending lines [45].

procedure for finding these coefficients is called *geometric camera calibration*, also called *camera resectioning*. Apart from the lens distortion coefficients, it also yields the intrinsic and extrinsic parameters of the camera, used, e.g., for pose estimation in robotics or navigation systems. A well established approach has been proposed by Zhang [46], which is a part of many computer vision libraries and toolboxes, such as OpenCV or Matlab Camera Calibration Toolbox[2].

Our application assumes that every input frame is a rectilinear image. Thus, before executing the application on a dataset, the input images must be undistorted, e.g., by camera calibration. Also, even if the dataset is already rectilinear, camera calibration can be used to obtain the focal length and, consequently, the FOV of the camera, which is needed for correcting the perspective distortion (as shown in the following section). The FOV can be derived from the focal length as

$$FOV = 2\arctan\frac{H}{2f}, \tag{6.1}$$

where H is the sensor size and f is the focal length. In the case of the virtual camera, H can be set arbitrarily. Our application uses 36 mm to emulate the focal length values to a full-frame camera.

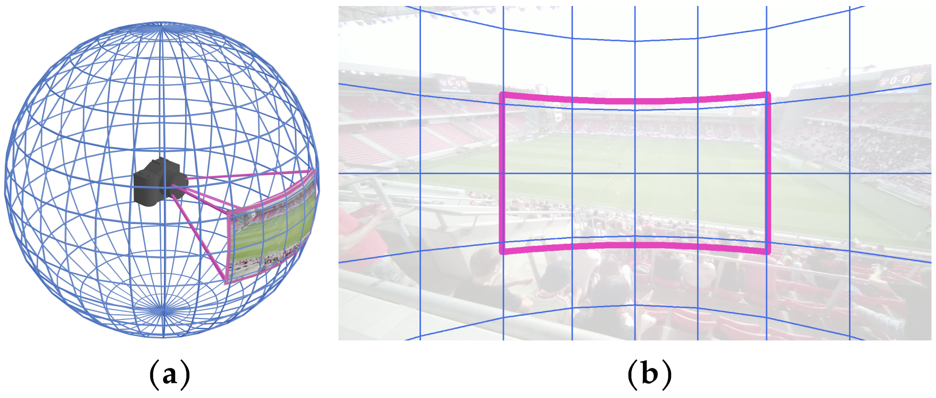
6.2.2 Perspective Distortion Correction

While optical lens distortion correction essentially transforms the dataset into a correct input format for our application, in our case rectilinear, perspective distortion correction handles the simulation of the PTZ effect. It leverages the rectilinear property of images by sampling from their projection scheme: the *gnomonic* projection, also referred to as *gnomic*, *rectilinear*, or *tangent-plane* projection.

Gnomonic Projection

The gnomonic projection is a perspective projection of a sphere from the center of the sphere onto a tangent plane [47]. This resembles the case of capturing a scene around the camera and projecting it on a

2. https://www.mathworks.com/help/vision/camera-calibration.html



(a) When taking a picture with a camera, we are essentially projecting a portion of a sphere (our scene) onto a plane (camera sensor). If the lens is equirectangular and spherical, the points on the sphere are related to points on the tangent plane by gnomonic projection.
(b) Rectilinear image with marked sphere grid lines projected using gnomonic projection. The image now serves as a tangent plane that touches the sphere by its center at latitude and longitude $(0, 0)$. Notice the stretching of grid squares towards the edges of the frame, mimicking the behavior of rectilinear lenses.

Figure 6.2: Visualization of the gnomonic projection.

flat sensor, as shown in Figure 6.2a. Figure 6.2b shows how these grid lines get deformed under the gnomonic projection[3].

The transformation equations [48] for projecting a point on a unit sphere (given by its longitude λ and latitude ϕ) onto a tangent plane

3. This is similar to the problem that cartographers face when creating world maps. The large number of world map projections shows that it is impossible to map a sphere onto a plane without introducing any distortion. For example, some projections prefer preserving angles for the cost of distorting image size and shape, like the Mercator projection, suitable, e.g., for navigation purposes; other projections put emphasis on preserving area sizes for the cost of distorting shapes and angles, like the Gall-Peters projection.

centered at λ_0, ϕ_1, are given by

$$x = \frac{\cos\theta \sin(\lambda - \lambda_0)}{\cos c}, \tag{6.2}$$

$$y = \frac{\cos\theta_1 \sin\theta - \sin\theta_1 \cos\theta \cos(\lambda - \lambda_0)}{\cos c}, \tag{6.3}$$

where $\cos c$ is the angular distance of the point (x, y) from the center of the projection, given by

$$\cos c = \sin\theta_1 \sin\theta + \cos\theta_1 \cos\theta \cos(\lambda - \lambda_0). \tag{6.4}$$

Panning and tilting the camera in this context means moving the tangent point on the surface of the sphere in the opposite direction as the desired camera orientation. This is key for correcting the perspective distortion. In the next section, we present the full algorithm.

Perspective Distortion Correction Algorithm

The deformed grid lines in Figure 6.2b already suggest that the expected shape of the ROI is a trapezoid. When the camera is pointing at the center of the frame, the ROI is a rectangle, and by panning or tilting, it turns into a trapezoid to compensate for the rectilinear image stretching. Since the mapping between the input frame and the ROI frame is linear, it can be expressed by a homography matrix. Similarly to Section 3.4, the following algorithm uses four corner points to calculate the homography matrix used for obtaining the final ROI.

Figure 6.3 shows a schematic overview of the algorithm. It uses two coordinate spaces: the screen space and the spherical space. The screen space is used for sampling from the input image. It has a normalized variant, ranging from 0 to 1, and a non-normalized variant, ranging from 0 to the resolution of the operated image. The spherical space represents the points on the sphere, needed for working with the gnomonic projection. Even though the spherical coordinates lie in intervals $[-\pi, \pi]$ and $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for horizontal and vertical direction, respectively, the algorithm uses only the portion of the sphere's surface that is covered by the FOV of the lens used for capturing the input footage.

The algorithm starts by generating four corner points in normalized screen coordinates. Next, they are transformed into the spherical

Virtual Camera Sampling Algorithm with Perspective Distortion Correction

Input: Rectilinear image, Lens FOV, Virtual Camera PTZ coordinates.
Output: ROI crop-out of the input image, based on the Virtual Camera FOV.

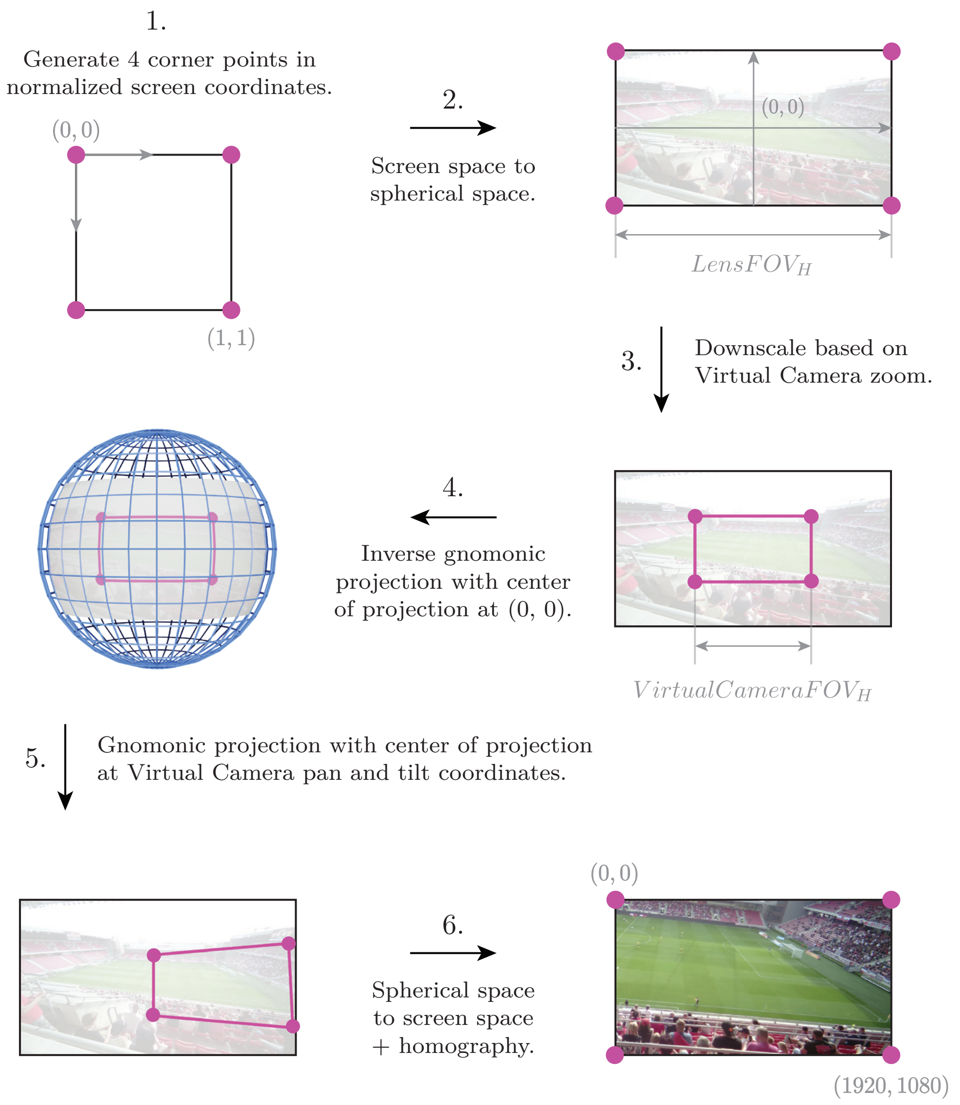

1.
Generate 4 corner points in normalized screen coordinates.
(0, 0)
(1, 1)
2.
Screen space to spherical space.
(0, 0)
$LensFOV_H$
3.
Downscale based on Virtual Camera zoom.
4.
Inverse gnomonic projection with center of projection at (0, 0).
$VirtualCameraFOV_H$
5.
Gnomonic projection with center of projection at Virtual Camera pan and tilt coordinates.
6.
Spherical space to screen space + homography.
(0, 0)
(1920, 1080)

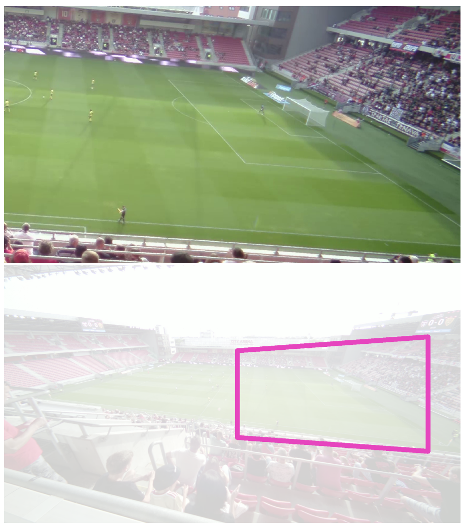
Figure 6.3: Algorithm for virtual camera sampling with perspective distortion correction using only 4 points.

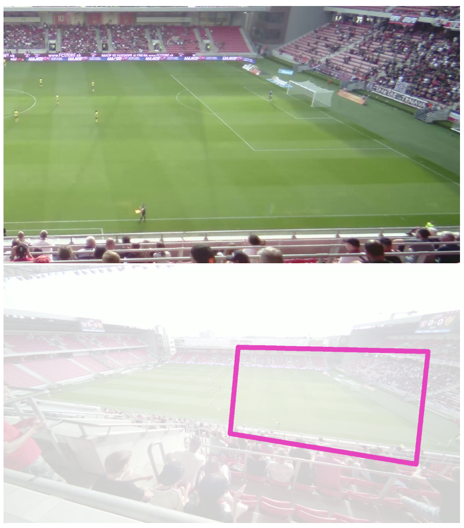
space, where they now represent the corner points of the input image, i.e., the top-left point has coordinates $(-\frac{LensFOV_H}{2}, -\frac{LensFOV_V}{2})$ and the bottom-right point has coordinates $(\frac{LensFOV_H}{2}, \frac{LensFOV_V}{2})$. In the third step, the points are scaled down to match the virtual camera FOV given by the current zoom level. These coordinates are then mapped onto the sphere's surface by the inverse gnomonic projection with the center of projection at $(0, 0)$. In step 5, they are mapped back onto a plane by the gnomonic projection, but this time with the center of projection at the pan and tilt coordinates of the virtual camera, which yields the expected trapezoid that is stretched towards the frame corners. In the last step, these spherical coordinates are transformed into screen coordinates, ranging from 0 to the original frame resolution. What remains is to map these four points to the corner points of the output frame using homography. The coordinates of the output frame corner points depend on the desired resolution: our application uses the standard Full HD resolution (1920 x 1080 pixels). The homography matrix relating the four projected points to the corner points of the output frame is calculated using the OpenCV function `findHomography`. Applying this matrix to the input frame yields the desired ROI crop-out.

In summary, this algorithm allows cropping out the ROI based on its center in the input image. Since the mapping between the source and the output image is linear, no line bending is employed. As a result, the obtained ROI is still a rectilinear image, but due to the lower FOV it covers, the image looks more natural to the eye.

6.2.3 Pitch Tilt Correction

Looking closely at the output image in Figure 6.3, it can be observed that the pitch is slightly rotated. The undesired effect of pitch rotation gets more pronounced at higher pan levels. We address this problem in step 4 of our algorithm, where we move the tangent point of the inverse gnomonic projection up (or down, depending on the dataset), effectively tilting the pitch outward or toward the camera. Figure 6.4 shows the effect of pitch tilt correction on our dataset.


(a) Before Correction


(b) After Correction

The frames are shown with their respective ROIs in the original frame.

Figure 6.4: Pitch tilt correction demonstration.

6.3 Movement Smoothing

This section presents the method used to achieve a smooth camera transition from one target to another. First, requirements for camera movement are listed. Next, the PID (proportional–integral–derivative) controller family is presented, which forms the basis for the PI (proportional–integral) controller used in this thesis. Finally, the influence of individual controller parameters is described and technical details are given.

6.3.1 Requirements

The transition is expected to progressively accelerate at the beginning of the movement and decelerate when approaching its end, with minimal overshooting. Furthermore, the transition speed should be high enough to follow dynamic scenarios, such as counter-attacks. The transition properties should be easily adjustable by the user.

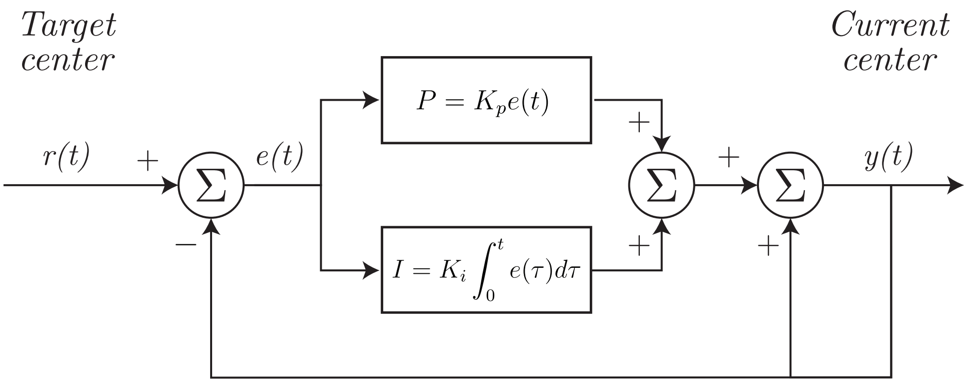


Figure 6.5: Block diagram of the PI controller.

6.3.2 PID Controller

This task comes from the field of control theory, where the aim is to drive a system state towards a target level. A well-established controller family is the so-called PID controller family. It is ubiquitously used in industry to regulate temperate, flow, pressure, speed, or other process variables [49].

The PID controller is a closed feedback control loop that keeps the actual output of a process as close to the target as possible [49]. It achieves this by calculating the error $e(t)$ (the difference between the actual output and the target) and relating this error to the controller output $y(t)$ by a transfer function. This transfer function consists of: the proportional term P, the integral term I, and the derivative term D (hence the name PID controller).

6.3.3 PI Controller

This thesis uses only the proportional and integral terms, i.e., the PI version of the PID controller family. The transfer function is then defined as

$$e(t) = r(t) - y(t), \tag{6.5}$$

$$y(t) = y(t) + K_p e(t) + K_i \int_0^t e(\tau) d\tau, \tag{6.6}$$

where $r(t)$ and $y(t)$ are the target and the actual output (camera center coordinate) at time t, respectively; K_p and K_i are the coefficients for the proportional and integral terms, respectively. This transfer function is also shown in block diagram in Figure 6.5.

6.3.4 Parameter Tuning

The K_p and K_i coefficients must be set appropriately for the PI controller to work correctly, each having its distinct effect on the camera's behavior.

The proportional coefficient K_p allows the user to regulate the responsiveness of the camera. By directly depending on the instantaneous value of the error, it also provides a gradual slowdown. However, in case of target change, using only the P term would result in sudden movement changes, harming smoothness.

The integral coefficient K_i addresses this issue by adding a slow reaction factor to the controller. Before changing the direction of motion, it gradually slows down, resulting in a smoother behavior. Also, by accumulating the error, it increases the maximum speed, which is suitable for capturing counter-attacks. For a detailed description of these coefficients and ways of tuning them, refer to [49].

Even though the integral term causes some minor overshooting, which could be compensated by the derivative term, achieving the highest precision possible is not our aim and, thus, we decided to omit the derivative term by setting K_d to 0. Furthermore, the use of the derivative term increases the controller's sensitivity to noise, making the system less stable, which would be more harmful to our purposes than minor overshooting.

Our application uses three separate PI controllers: two for the ROI center point (x and y coordinates) and one for the zoom (focal length of the virtual camera). Advantages of this approach include low-performance overhead and customizability by the user. This solution meets the listed requirements and produces satisfactory results.

7 Evaluation

It is not clear how to approach the evaluation of our system. A labor-intensive way could be manually setting the camera PTZ coordinates for each frame by a person and comparing it to the estimations. However, as already discussed in this thesis, people have different framing preferences, and thus, these PTZ coordinates could differ widely, even among experienced camera operators. Given this subjective nature of the evaluation of such a system, one could also conduct a user study, as, e.g., in [11].

Nevertheless, we decided to utilize the broadcast footage provided with our evaluation dataset (presented in Section 7.1). We compare the estimated frames (referred to as the **main** camera) with the broadcast video frames (referred to as the **VAR** camera). For the metric, we compute the average Intersection over Union (IoU) of their top-down projected frames.

This chapter first presents the evaluation dataset and the technical setup. Second, it describes the method and the results of full match evaluation. Third, it uses a second evaluation method: pass/fail manual inspection of 30-second clips of selected match highlights. Last, it draws conclusions from the obtained results in the discussion section.

7.1 TrnavaZilina Dataset

With the help of Goal Sport Technology, we recorded the entire soccer match between FC Spartak Trnava and MŠK Žilina in the Slovak First Football League. It was the sixth round of the Niké Liga on 03.09.2023 at 17:30 at Anton Malatinský Stadium in Trnava.

The description of the recording setup (shown in Figure 7.1) follows. The camera used for recording was the Aida UHD-100A[1]. It was equipped with a wide-angle, rectilinear lens by Theia Technologies[2]. This setup provided a horizontal and vertical angle of view of 115 and 83 degrees, respectively[3]. We used the Blackmagic UltraStudio

1. https://aidaimaging.com/uhd-100a/
2. https://www.theiatech.com/lenses/183/?lens=ML183A
3. While this information can be acquired by camera calibration, we obtained it from the manufacturer's product page.

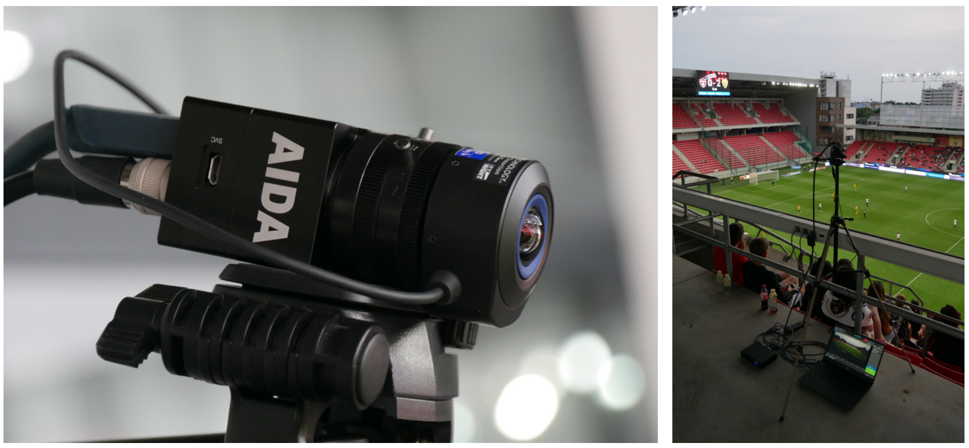


Figure 7.1: The TrnavaZilina recording setup.

4K Mini[4] for recording a 4K (3840 x 2160px) video in 30 frames per second (FPS) and bitrate 7710 kbps encoded with the H264 codec.

The Slovak Football Association kindly provided the original broadcast footage from the primary (VAR) camera, however, for evaluation purposes only. Thus, for legal grounds, the full-length footage from neither the VAR camera nor the main camera could be enclosed in the attachments of this thesis. Only several 30-second clips from the main camera are attached for testing purposes[5].

This is a challenging dataset for our algorithm especially due to the player jersey colors: one team is wearing yellow jerseys and the other is wearing white jerseys. In the training dataset of our detector, yellow is a typical color for the referee and white for the ball. As shown previously in Figure 3.3, this may lead to incorrect label assignment for objects of these colors. To address this, all detected object classes (except for the ball) are treated as players in our experiments. Furthermore, some parts of the dataset contain strong lens flares due to direct opposite sunlight. Their bright color and strong contrast cause false positive detections, harming the Cameraman module estimation quality. To avoid this at the time of recording the dataset, it is recommended to use

4. https://www.blackmagicdesign.com/products/ultrastudio/techspecs/W-DLUS-11
5. See the clip `p0_shot4.mp4` for a delightful bicycle kick

lens hoods. For the sake of evaluating our system, we are discarding the false positive detections by predefining the areas containing the lens flares; these areas are then discarded in the detection pipeline (see Section 4.3).

7.2 Full Match Evaluation

To compare two views of a scene, it is necessary to convert the frames into a common coordinate system. We decided to use the top-down view as this common space, where, after projecting the two views, we get two finite sample sets of pixels, where a pixel belongs to a set iff it has a value greater than zero.

7.2.1 Metric

Next, we want to find a metric that lets us compare multiple algorithms over the full match. If we used only the intersection, an over-conservative algorithm that stays fully zoomed out throughout the whole match would achieve the highest score. Another metric could be the difference between the two sets. This would penalize the over-conservative case; however, in case of no overlap, the difference would be the same as for an optimal framing. Thus, we want to also penalize areas that are not overlapping. A metric combining both types of penalization is the IoU, or the Jaccard index. It is a commonly used similarity measure and is defined as follows:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \tag{7.1}$$

where A and B are finite sample sets.

7.2.2 Algorithm

The evaluation algorithm takes two videos (main and VAR) as input and outputs their mean and median IoU.

1. **Export frames**

   Export a frame every 10 seconds both from the main and the VAR footage.

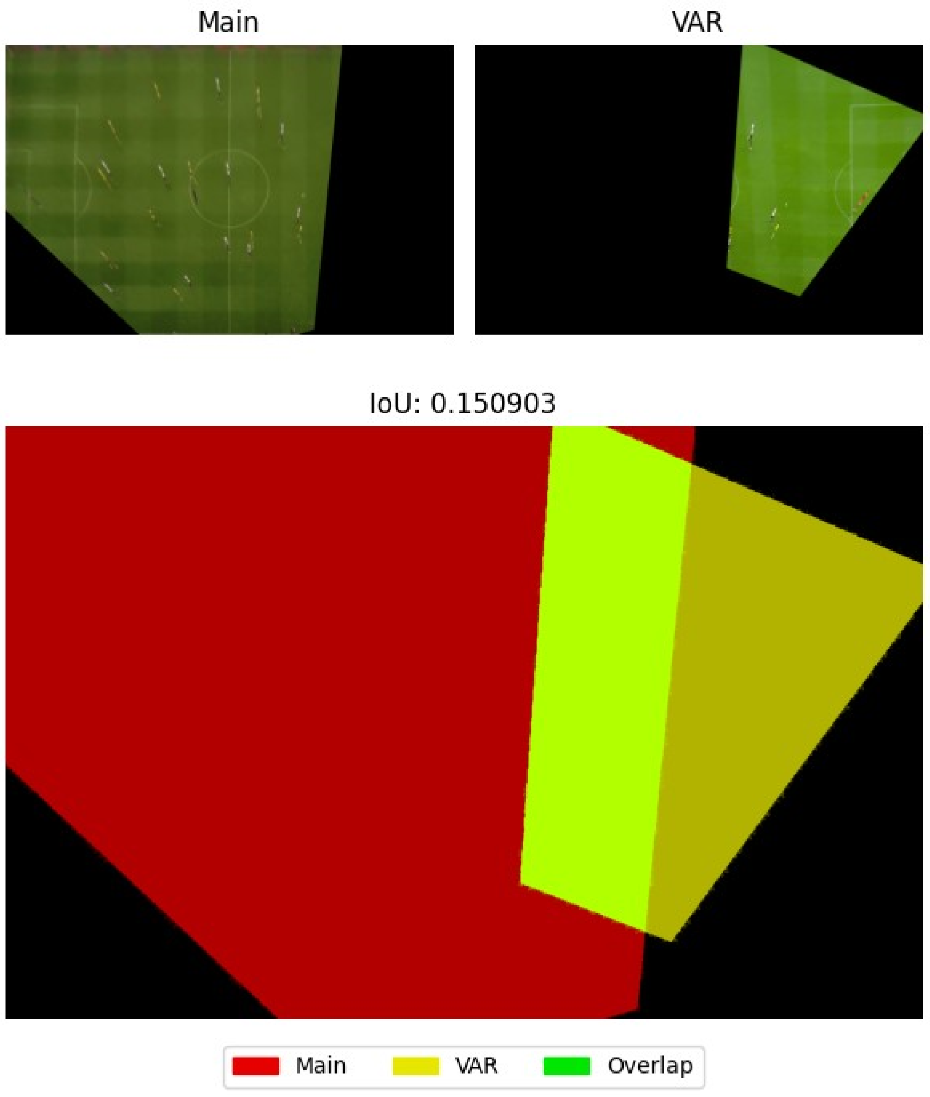


Figure 7.2: Visualization of the evaluation metric: IoU of the top-down projected estimated (main) and reference (VAR) frames.

2. **Top-down transform**

   Transform these frames into a unified top-down view. We use the homography matrix mapping points from the screen space to the top-down space (described in more detail in Section 3.4).

   For the main frame, we utilize the homography matrix of the original frame at the time of cropping the ROI (see step 6 in Figure 6.3).

   For the VAR frame, however, we do not have the homography matrix yet. To obtain it, we first need to calibrate the camera, i.e., to determine the camera position and rotation with respect to the pitch. For this, we use the TVCalib method [50]. TVCalib takes the broadcast frames as input and outputs the intrinsic and extrinsic parameters of the camera, including the homography matrix and the segment reprojection loss (described in [50]). We use this loss to discard low-quality (invalid) calibrations from the evaluation. To assess the calibration quality, we use the segment reprojection loss. We assume that an invalid calibration has a significantly higher or lower loss than most of the losses. To find the threshold values for discarding outliers, we use the boxplot lower and upper whiskers as follows:

   $$t_{low} = Q_1 - 1.5IQR, \tag{7.2}$$
   $$t_{high} = Q_3 + 1.5IQR, \tag{7.3}$$

   where t_{low} and t_{high} are the lower and the upper threshold, respectively; Q_1 and Q_3 are the first and the third quartile, respectively; and

   $$IQR = Q_3 - Q_1. \tag{7.4}$$

   Figure 7.3 shows examples of valid and invalid calibrations with their respective loss values.

3. **Overlay**

   Overlay the main and the VAR projected frames over each other and calculate their IoU.

Table 7.1: Mean IoU of different algorithm variants.

Algorithm	First half	Second half	Full match
Single detector	**0.3696**	**0.3390**	**0.3548**
Ball detector	0.3600	0.3337	0.3472
Imgsz640	0.3539	0.3257	0.3402
N1000	0.3695	**0.3390**	**0.3548**

7.2.3 Results

Table 7.1 shows the full match evaluation results for four outputs of our system using the TrnavaZilina dataset.

Single detector This is our reference algorithm; the rest of the evaluated algorithms are only variants of it, differing only in a single parameter. This algorithm uses a single detector for both the players and the ball, with an image size of 960 pixels and 500 particles for the Particle Filter. For a complete list of parameters for this variant, refer to the thesis attachments.

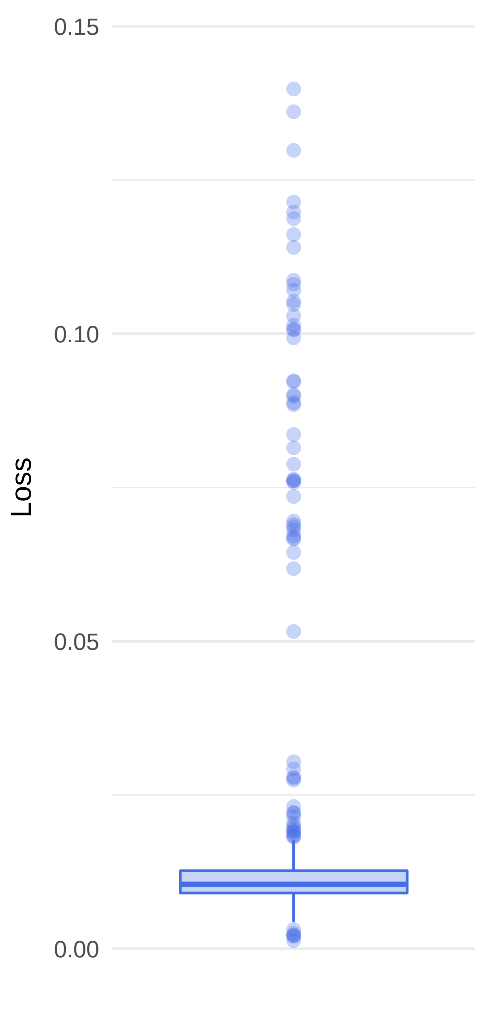
Ball detector This variant tests the effect of using a specialized ball detector `YOLOv8n_960_ball` (described in Chapter 4) instead of a single, universal detector for all object classes. The resulting mean IoU is lower than in the case of a single detector.

Imgsz640 Decreasing the resolution of the detector input image from 960 to 640 pixels is demonstrated by this algorithm variant. As expected, it resulted in a drop in the mean IoU.

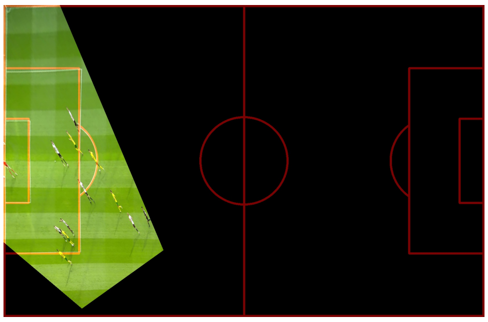
N1000 This algorithm variant uses 1000 PF particles instead of 500, which did not result in improved mean IoU.

7.3 Clips Evaluation

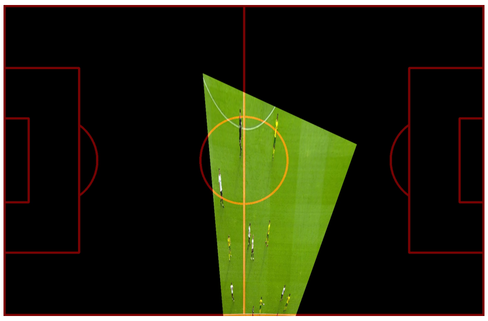
Apart from focusing on the match as a whole, we wanted to test whether all key moments are captured well. In particular, whether all goals and shots are in frame and whether the algorithm can follow fast counterattacks.



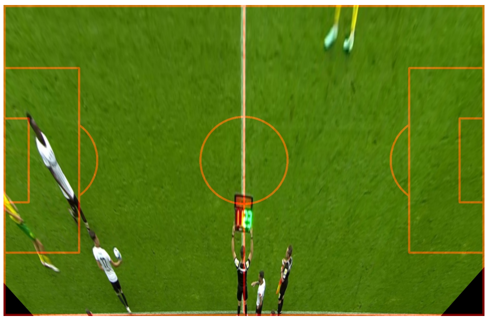
(a) Boxplot



(b) Loss: 0.0091 (OK)



(c) Loss: 0.0607 (NOK)



(d) Loss: 0.0012 (NOK)

Subfigure (a) shows the calibration loss boxplot of the full match used for determining the lower (t_{low}) and upper (t_{high}) threshold for discarding invalid calibrations. In this case, $t_{low} = 0.0034$ and $t_{high} = 0.0188$.
The rest of the subfigures show examples of top-down views of a valid calibration (b), invalid calibration with loss above the upper threshold (c), and invalid calibration with loss below the lower threshold (d).

Figure 7.3: Calibration results of VAR full match recordings.

Table 7.2: Clips evaluation results.

Clip	**Single detector**	Ball detector	Imgsz640	N1000
p0_shot1	Pass	Pass	Pass	Pass
p0_shot2	Pass	Pass	Pass	Pass
p0_shot3	Pass	Pass	Pass	Pass
p0_shot4	Pass	Pass	Pass	Pass
p0_zoom	Fail	Fail	Fail	Fail
p1_counter1	Fail	Fail	Pass	Fail
p1_goal1	Pass	Pass	Pass	Pass
p1_goal2	Pass	Fail	Pass	Pass
p1_shot1	Pass	Pass	Pass	Pass
p1_shot2	Fail	Fail	Fail	Fail

Prefix p0 denotes the period of the game—0 for the first and 1 for the second. The suffix denotes the situation captured by the recording—whether it is a goal, a shot, a counterattack, or a clip testing the adaptability of the algorithm for various zoom requirements.

We thus decided to export 10 clips capturing such scenarios. 9 of them are around 30 seconds long and capture 6 shots, 2 goals, and 1 counterattack. There is one clip of length 1 minute and 21 seconds (p0_zoom), which captures a part of the game with variable requirements for zoom: zoom in to capture the game in the rear part of the pitch, then zoom out to cover an attack, and finally capture the shot at the end of the action.

We wanted to look at these clips with the eye of a spectator, for whom it is essential to see the ball constantly. Thus, we processed each clip and observed whether the ball was always in the frame. If it was, it passed our test, otherwise, it failed.

7.3.1 Results

Table 7.2 shows that the algorithms passed the majority of cases. All shots and goals are framed well, capturing both the active player and the goal with the goalkeeper.

The deficiencies of the algorithm emerge in three clips: `p0_zoom`, `p1_counter1`, and `p1_shot2`. During dynamic parts of the game, when the ball is played from side to side, the camera sometimes struggles to follow the ball, resulting in losing it from the frame for a few moments. Even though, in these cases, the camera kept most of the players in the frame, and the ball eventually reappeared, we consider these clips to be failed estimates.

7.4 Technical Details

Initialization The initial PTZ coordinates of the algorithm were initialized so that both goals were in the frame. This fact may be reflected in the mean IoU of the clips, where in the first few seconds, the camera is panning from the center to the pitch.

Performance The estimations were run on a remote computing server equipped with AMD EPYC 7702P 64-Core CPU, 512GB RAM, and NVIDIA A100 80GB GPU. The computation time for two match periods run in parallel was 3 hours and 44 minutes (approx. 6.5 FPS on average). The computation of one frame took approx. 100 ms, where the bottleneck did not lie in the detection phase, which took only 25 ms thanks to the small detector, but in the image sampling operations.

Metric Limitations Note that, for this particular dataset, the two cameras are neither mounted on the same tripod nor placed near each other (the main camera is standing approximately 15 meters apart from the VAR camera). Furthermore, the broadcast camera uses a more zoomed-in framing for a more immersive effect for the viewers, whereas the main camera aims to use a more zoomed-out framing, as mentioned in Section 5.1. Consequently, the IoU could not be precisely equal to 1 even if the estimated ROI closely matched the reference ROI.

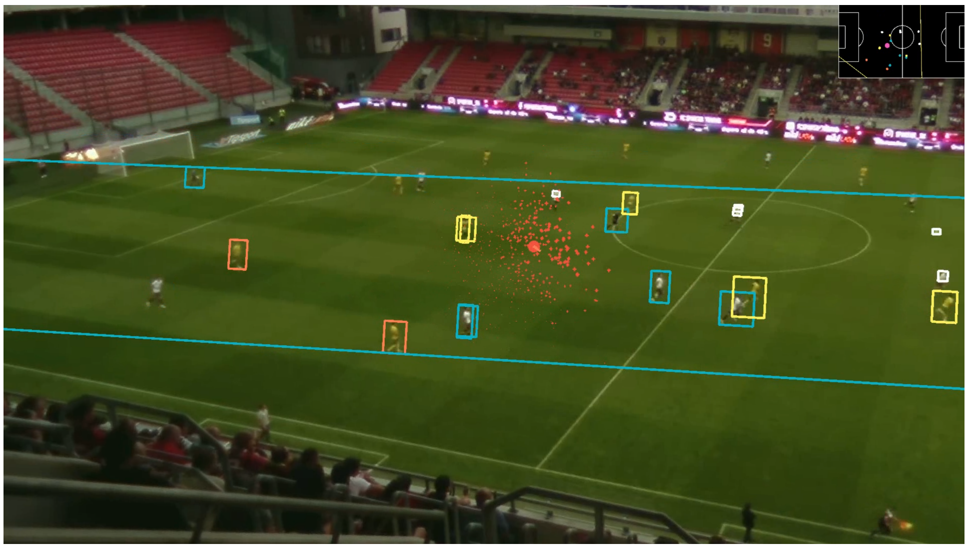
Nevertheless, this is not an issue since we require a relative measure that allows us to compare algorithm variants on the same dataset. The same holds for the interpretability of the IoU: by looking at the number, it is not easy to infer whether the algorithm performs poorly, but it serves well for comparison purposes. It also allows detecting large deviations from the broadcast frame, which might be the case for especially poorly performing algorithms.

7.5 Discussion

From the presented results and our observations while browsing the estimated footage, we can conclude that the algorithm performs well in most situations. It moves the camera smoothly, capturing all key moments and following the important part of the game most of the time. Even in the case of failed clips, the system managed to recover swiftly. Since doubling the number of particles in the particle filter did not lead to any significant improvements, we can consider leaving the number of particles at 500, or even decreasing it with decreasing variance (as in [40]).

The most significant observable difference between the estimated and the reference footage is the decreased image quality. While some companies argue that *the recordings are not meant for professional TV broadcasting, and the reduced video quality format works really well on devices with smaller screens* [51], the low resolution combined with poorer light conditions negatively impacts also the detector. This brings more false positives into the Cameraman module, which we found to be the most harmful factor for estimation quality.

For example, the clip p1_counter was obtained in the second half of the game, when the scene was darker than in the first half, and thus, the image was noisier and blurrier (see Figure 7.4). This resulted in the detection of more false positives caused by players in white jerseys, which *held back* the camera from following the actual ball movement (even though the ball was constantly detected). As shown in Table 7.2, Imgsz640 is the only variant that managed to keep the ball in frame for this clip. This is due to the lower number of false positive detections produced by the detector when given an image of a lower resolution. The negative impact of false positives can also be observed in the separate ball detector variant, which performs worse than the single detector algorithm in the mean IoU and the clips.



The false positives are *holding back* the camera from following the counterattack. Notice that the ball (white) is correctly detected near the right edge of the frame, but the particles (red) are attracted towards their nearest ball detections.
Note: The yellow, orange, and teal boxes correspond to detections of players, referees, and goalkeepers, respectively. Since many boxes are labeled incorrectly, we treat them as one class in our algorithm. The figure also shows the bounding box encapsulating all players (with added margins) used for estimating the zoom level as explained in Section 5.3.

Figure 7.4: False positives *holding back* the camera.

8 Conclusion

In this thesis, we developed an automatic virtual camera system that accepts a rectilinear video as input and makes a crop-out of each frame to produce a video with an automated PTZ. This is achieved by detecting the players and the ball on the field, estimating a framing based on the detections, and emulating the smooth PTZ of a real camera by compensating for perspective distortion. The detections are also used to render a top-down view of the detection for analysis purposes. The system allows easy adjustment of parameters, making it customizable for various datasets and user preferences.

Furthermore, we developed an evaluation algorithm that performs camera calibration on the broadcast video and compares the estimated video to the broadcast by computing the IoU of their top-down projected frames. While this metric is not an absolute measure of quality, it can serve to compare different algorithm variants to each other and to detect large deviations from the broadcast frame.

The results show that the system performs well in most situations, producing smooth movements with minimal jitter and keeping the ball in the frame throughout all key moments of the match. The main bottleneck of the system lies in the detector, which harms the estimation by producing false positives. This results in unnatural movements and impairs the camera's ability to follow the game action in some dynamic scenarios.

8.1 Future Work

There is still a lot of room for further exploration in future work. This section suggests possible areas for improvement, ranging from the input frame to the evaluation stage.

Input Frame To achieve a higher output frame resolution, two or more cameras could be rigged together, each covering its respective field of view. This would be, however, for the cost of increased computational complexity due to an added stitching phase.

Frame Splitter A desired feature for a production-ready product of this kind is deriving the Frame Splitter PTZ parameters automatically, i.e., without manually positioning the three virtual cameras for each dataset. Furthermore, non-maximum suppression (NMS) could be implemented for the overlapping areas of virtual camera frames to avoid duplicate detections. One can also consider using more than three Frame Splitter windows for a more zoomed-in input frame for the detector, utilizing the fact that the detections are computed in a batch in parallel.

Detector The detector could be undoubtedly improved by further training on different datasets, using different parameters or base models (e.g., a larger version of YOLOv8[1] or a different architecture [22]). However, given the uniform background of a soccer pitch, there is more room for optimization. For example, background modeling could identify moving objects as blobs, and then a deep learning classifier could assign labels to these blobs, as proposed in [52]. Also, detection clustering ([26, 27]) could be used not only to discard unwanted detections (i.e., coaches, referees, spectators, and staff) but also to cluster players into teams, which could be a useful piece of information in the Cameraman module. Discarding unwanted detections could also work with the fact that the spectators and coaches tend to be static when compared to the players, as proposed in [53].

Player and Ball Tracking For improved tracking, optical flow could be leveraged as in [26]. Also, the ball location could be predicted based on players' behavior on the pitch [54].

Cameraman For the Cameraman module, we suggest two main improvements.

First, the algorithm currently does not account for the situation when the ball is in possession of the goalkeeper. In case the ball is not detected, the goalkeeper is also unlikely to be included in the detections (due to discarding the extreme detections as the first step of the algorithm). As a result, the ball might get out of frame, and

1. https://github.com/ultralytics/ultralytics

thus, the algorithm should avoid discarding the goalkeeper detection in this situation.

Second, there is a trade-off between camera sensitivity and smoothness in the tracking and PI controller parameters. A sensitive camera can adjust to dynamic scenarios, such as counterattacks, for the cost of being sensitive to noise, leading to unnatural camera movements; the opposite applies to an overly smooth camera. Instead of laborious tuning of these parameters, we suggest dynamically changing them based on the dynamics of the play, using smooth settings for static play and sensitive settings when rapid movement is detected. For analyzing the evolution of the game, one can use the decomposition of individual situations in the game, as in [55], or the players' motion field, as in [56].

As an alternative approach, the algorithm could be replaced (or augmented) by a deep learning regressor, as in [9]. This approach would, however, require a large amount of training data and provide less control over the camera characteristics.

Evaluation Given the subjective nature of the evaluation of such a system, user studies could be conducted for better quality assessment of individual algorithms, as in [11].

Performance As a performance optimization, every n-th frame could be skipped from processing, especially in broadcasts using 60 FPS. Another considered optimization is suggested in [40], where the number of PF particles varies with the PF variance.

This system also has the potential to allow the user to choose a tracked target, e.g., instead of tracking primarily the ball, tracking a particular player. This could be achieved either by an interactive selection of a player tracklet, given a reliable player tracker, or by selecting the player number, given a reliable jersey number classifier [57].

Bibliography

1. *Field of View - PanoTools.org Wiki* [https://wiki.panotools.org/Field_of_View]. 2023. Accessed: 2023-12-09.
2. *Veo Camera: Elevate your game with AI-Powerede Video Capture* [https://www.veo.co/product/camera]. 2023. Accessed: 2023-12-09.
3. SHAULOFF, Dor. *Introducing Show S3: the Best Soccer Camera - Pixellot* [https://www.pixellot.tv/blog/introducing-show-s3-reinventing-live-sports-streaming/]. 2023. Accessed: 2023-12-09.
4. APOSTOLIDIS, Konstantinos; MEZARIS, Vasileios. A Fast Smart-Cropping Method and Dataset for Video Retargeting. In: *2021 IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 2618–2622. Available from DOI: 10.1109/ICIP42928.2021.9506390.
5. RACHAVARAPU, Kranthi Kumar; KUMAR, Moneish; GANDHI, Vineet; SUBRAMANIAN, Ramanathan. Watch to Edit: Video Retargeting using Gaze. *Computer Graphics Forum*. 2018, vol. 37, no. 2, pp. 205–215. Available from DOI: https://doi.org/10.1111/cgf.13354.
6. DAIGO, Shinji; OZAWA, Shinji. Automatic Pan Control System for Broadcasting Ball Games Based on Audience's Face Direction. In: *Proceedings of the 12th Annual ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2004, pp. 444–447. MULTIMEDIA '04. ISBN 1581138938. Available from DOI: 10.1145/1027527.1027634.
7. CHEN, Jianhui; CARR, Peter. Mimicking Human Camera Operators. *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*. 2015, pp. 215–222. Available from DOI: 10.1109/WACV.2015.36.
8. PIDAPARTHY, Hemanth; ELDER, James. Keep Your Eye on the Puck: Automatic Hockey Videography. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 1636–1644. Available from DOI: 10.1109/WACV.2019.00179.

9. BAYRAK, MEHMET. *An auto video shooting system for soccer games*. 2020. MA thesis. Bahçeşehir University.

10. GADDAM, Vamsidhar; LANGSETH, Ragnar; LJØDAL, Sigurd; GURDJOS, Pierre; CHARVILLAT, Vincent; GRIWODZ, Carsten; HALVORSEN, Pål. Interactive Zoom and Panning from Live Panoramic Video. In: 2014. Available from DOI: `10.1145/2578260.2578264`.

11. GADDAM, Vamsidhar; EG, Ragnhild; LANGSETH, Ragnar; GRIWODZ, Carsten; HALVORSEN, Pål. The Cameraman Operating My Virtual Camera is Artificial: Can the Machine Be as Good as a Human? *ACM Transactions on Multimedia Computing, Communications, and Applications*. 2015, vol. 11, pp. 1–20. Available from DOI: `10.1145/2744411`.

12. YU, Frank; SALZMANN, Mathieu; FUA, Pascal; RHODIN, Helge. PCLs: Geometry-aware Neural Reconstruction of 3D Pose with Perspective Crop Layers. *arXiv preprint arXiv:2011.13607*. 2020.

13. SCOTT, Atom; UCHIDA, Ikuma; ONISHI, Masaki; KAMEDA, Yoshinari; FUKUI, Kazuhiro; FUJII, Keisuke. SoccerTrack: A Dataset and Tracking Algorithm for Soccer With Fish-Eye and Drone Videos. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3569–3579.

14. SZELISKI, R. *Computer Vision: Algorithms and Applications*. Springer International Publishing, 2022. Texts in Computer Science. ISBN 9783030343729. Available also from: `https://books.google.cz/books?id=QptXEAAAQBAJ`.

15. AMIN, Jiten. Soccer Player Tracking System. *International Journal for Research in Applied Science and Engineering Technology*. 2018, vol. 6, pp. 3455–3461. Available from DOI: `10.22214/ijraset.2018.3727`.

16. DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, vol. 1, 886–893 vol. 1. Available from DOI: `10.1109/CVPR.2005.177`.

17. POPPE, Chris; BRUYNE, Sarah De; VERSTOCKT, Steven; WALLE, Rik Van de. Multi-Camera Analysis of Soccer Sequences. In: *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2010, pp. 26–31. Available from DOI: 10.1109/AVSS.2010.64.

18. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. Available from DOI: 10.1109/CVPR.2014.81.

19. CIOPPA, Anthony; DELIÈGE, Adrien; MAGERA, Floriane; GIANCOLA, Silvio; BARNICH, Olivier; GHANEM, Bernard; VAN DROOGENBROECK, Marc. Camera Calibration and Player Localization in SoccerNet-v2 and Investigation of their Representations for Action Spotting. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2021, pp. 4532–4541. Available from DOI: 10.1109/CVPRW53098.2021.00511.

20. REDMON, Joseph; DIVVALA, Santosh Kumar; GIRSHICK, Ross B.; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*. 2015, vol. abs/1506.02640. Available from arXiv: 1506.02640.

21. LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. SSD: Single Shot MultiBox Detector. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 21–37. ISBN 9783319464480. ISSN 1611-3349. Available from DOI: 10.1007/978-3-319-46448-0_2.

22. KOMOROWSKI., Jacek; KURZEJAMSKI., Grzegorz; SARWAS., Grzegorz. FootAndBall: Integrated Player and Ball Detector. In: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 5: VISAPP*. SciTePress, INSTICC, 2020, pp. 47–56. ISBN 978-989-758-402-2. ISSN 2184-4321. Available from DOI: 10.5220/0008916000470056.

23. HUANG, Yu-Chuan; LIAO, I-No; CHEN, Ching-Hsuan; IK, Tsi-Ui; PENG, Wen-Chih. TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications. *CoRR*. 2019, vol. abs/1907.03698. Available from arXiv: 1907.03698.

24. ZOU, Zhengxia; CHEN, Keyan; SHI, Zhenwei; GUO, Yuhong; YE, Jieping. Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*. 2023, vol. 111, no. 3, pp. 257–276. Available from DOI: 10.1109/JPROC.2023.3238524.

25. GOODFELLOW, Ian J.; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. http://www.deeplearningbook.org.

26. MAVROGIANNIS, Panagiotis; MAGLOGIANNIS, Ilias. Amateur football analytics using computer vision. *Neural Computing and Applications*. 2022, vol. 34, no. 22, pp. 19639–19654. ISSN 1433-3058. Available from DOI: 10.1007/s00521-022-07692-6.

27. THEAGARAJAN, Rajkumar; PALA, Federico; ZHANG, Xiu; BHANU, Bir. Soccer: Who Has the Ball? Generating Visual Analytics and Player Statistics. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1830–18308. Available from DOI: 10.1109/CVPRW.2018.00227.

28. KIM, Kihwan; GRUNDMANN, Matthias; SHAMIR, Ariel; MATTHEWS, Iain; HODGINS, Jessica; ESSA, Irfan. Motion fields to predict play evolution in dynamic sport scenes. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 840–847. Available from DOI: 10.1109/CVPR.2010.5540128.

29. CUI, Yutao; ZENG, Chenkai; ZHAO, Xiaoyu, et al. SportsMOT: A Large Multi-Object Tracking Dataset in Multiple Sports Scenes. 2023. Available from arXiv: 2304.05170 [astro-ph.IM].

30. ZHANG, Yifu; SUN, Peize; JIANG, Yi, et al. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. 2021. Available from arXiv: 2110.06864 [astro-ph.IM].

31. AHARON, Nir; ORFAIG, Roy; BOBROVSKY, Ben-Zion. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. 2022. Available from arXiv: `2206.14651 [astro-ph.IM]`.

32. ARULAMPALAM, M.S.; MASKELL, S.; GORDON, N.; CLAPP, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*. 2002, vol. 50, no. 2, pp. 174–188. Available from DOI: `10.1109/78.978374`.

33. MILAN, Anton; REZATOFIGHI, Seyed Hamid; DICK, Anthony R.; SCHINDLER, Konrad; REID, Ian D. Online Multi-target Tracking using Recurrent Neural Networks. *CoRR*. 2016, vol. abs/1604.03635. Available from arXiv: `1604.03635`.

34. KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. 1960, vol. 82, no. 1, pp. 35–45. ISSN 0021-9223. Available from DOI: `10.1115/1.3662552`.

35. DOUCET, A.; GODSILL, S.J.; WEST, M. Monte Carlo filtering and smoothing with application to time-varying spectral estimation. In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*. 2000, vol. 2, II701–II704 vol.2. Available from DOI: `10.1109/ICASSP.2000.859056`.

36. CHEN, Wei; ZHANG, Yu-Jin. Tracking Ball and Players with Applications to Highlight Ranking of Broadcasting Table Tennis Video. In: *The Proceedings of the Multiconference on "Computational Engineering in Systems Applications"*. 2006, vol. 2, pp. 1896–1903. Available from DOI: `10.1109/CESA.2006.4281948`.

37. HAMUDA, Esmael; MC GINLEY, Brian; GLAVIN, Martin; JONES, Edward. Improved image processing-based crop detection using Kalman filtering and the Hungarian algorithm. *Computers and Electronics in Agriculture*. 2018, vol. 148, pp. 37–44. ISSN 0168-1699. Available from DOI: `https://doi.org/10.1016/j.compag.2018.02.027`.

38. PÉREZ, P.; HUE, C.; VERMAAK, J.; GANGNET, M. Color-Based Probabilistic Tracking. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 661–675. ISBN 9783540479697. ISSN 0302-9743. Available from DOI: 10.1007/3-540-47969-4_44.

39. LIANG, Dawei; HUANG, Qingming; LIU, Yang; ZHU, Guangyu; GAO, Wen. Video2Cartoon: A System for Converting Broadcast Soccer Video into 3D Cartoon Animation. *IEEE Transactions on Consumer Electronics*. 2007, vol. 53, no. 3, pp. 1138–1146. Available from DOI: 10.1109/TCE.2007.4341597.

40. DURUS, Murat. Ball Tracking and Action Recognition of Soccer Players in TV Broadcast Videos. In: 2014. Available also from: https://api.semanticscholar.org/CorpusID:39166258.

41. BAISA, Nathanael L. Derivation of a Constant Velocity Motion Model for Visual Tracking. *CoRR*. 2020, vol. abs/2005.00844. Available from arXiv: 2005.00844.

42. ELFRING, Jos; TORTA, Elena; MOLENGRAFT, René van de. Particle Filters: A Hands-On Tutorial. *Sensors*. 2021, vol. 21, no. 2, p. 438. ISSN 1424-8220. Available from DOI: 10.3390/s21020438.

43. BOLIC, M.; DJURIC, P.M.; HONG, Sangjin. Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing*. 2005, vol. 53, no. 7, pp. 2442–2450. Available from DOI: 10.1109/TSP.2005.849185.

44. JACOBSON, Ralph; RAY, Sidney; ATTRIDGE, Geoffrey G; AXFORD, Norman. *Manual of Photography*. Routledge, 2013. ISBN 9780080510965. Available from DOI: 10.4324/9780080510965.

45. FARIS-BELT, A. *The Elements of Photography: Understanding and Creating Sophisticated Images*. Focal Press, 2011. ISBN 9780240815152. Available also from: https://books.google.ne/books?id=dQzS5spOpL8C.

46. ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2000, vol. 22, no. 11, pp. 1330–1334. Available from DOI: 10.1109/34.888718.

47. PEARSON, F. *Map ProjectionsTheory and Applications*. Taylor & Francis, 1990. ISBN 9780849368882. Available also from: `https://books.google.sm/books?id=0D0gcaGB2hMC`.

48. *Gnomonic Projection – from Wolfram MathWorld* [`https://mathworld.wolfram.com/GnomonicProjection.html`]. 2023. Accessed: 2023-11-20.

49. GOODWIN, G.C.; GRAEBE, S.F.; SALGADO, M.E. *Control System Design*. Prentice Hall, 2001. ISBN 9780139586538. Available also from: `https://books.google.cz/books?id=7dNSAAAAMAAJ`.

50. THEINER, Jonas; EWERTH, Ralph. TVCalib: Camera Calibration for Sports Field Registration in Soccer. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 1166–1175. Available from DOI: `10.48550/ARXIV.2207.11709`.

51. O'HEAR, Steve. *Veo uses AI to let amateur soccer clubs video matches without the need for a camera operator | TechCrunch* [`https://techcrunch.com/2017/08/30/veo/`]. 2023. Accessed: 2023-11-28.

52. KAMBLE, P.R.; KESKAR, A.G.; BHURCHANDI, K.M. A deep learning ball tracking system in soccer videos. *Opto-Electronics Review*. 2019, vol. 27, no. 1, pp. 58–69. ISSN 1230-3402. Available from DOI: `https://doi.org/10.1016/j.opelre.2019.02.003`.

53. RANGAPPA, Shreedhar; LI, Baihua; QIAN, Ruiling. Tracking and identification for football video analysis using deep learning. In: 2021, p. 6. Available from DOI: `10.1117/12.2586798`.

54. AMİRLİ, Anar; ALEMDAR, Hande. Prediction of the Ball Location on the 2D Plane in Football Using Optical Tracking Data. *Academic Platform Journal of Engineering and Smart Systems*. 2022, vol. 10, no. 1, pp. 1–8. Available from DOI: `10.21541/apjess.1060725`.

55. ARIKI, Yasuo; KUBOTA, Shintaro; KUMANO, Masahito. Automatic Production System of Soccer Sports Video by Digital Camera Work Based on Situation Recognition. In: *Eighth IEEE International Symposium on Multimedia (ISM'06)*. 2006, pp. 851–860. Available from DOI: `10.1109/ISM.2006.37`.

56. KIM, Kihwan; GRUNDMANN, Matthias; SHAMIR, Ariel; MATTHEWS, Iain; HODGINS, Jessica; ESSA, Irfan. Motion fields to predict play evolution in dynamic sport scenes. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 840–847. Available from DOI: 10.1109/CVPR.2010.5540128.

57. VATS, Kanav; FANI, Mehrnaz; CLAUSI, David A.; ZELEK, John S. Multi-task learning for jersey number recognition in Ice Hockey. *CoRR*. 2021, vol. abs/2108.07848. Available from arXiv: 2108.07848.