UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Sadji

Real-Time Soccer Player Localization and Tracking

William Alexander Stimpson-Karlsson

Master thesis in Computer Science, INF-3990, May 2024

Supervisors

Main supervisor:	Dag Johansen	UiT The Arctic University of Norway, Faculty of Science and Technology, Department of Computer Science
Co-supervisor:	Martin Rypdal	UiT The Arctic University of Norway, Faculty of Science and Technology, Department of Mathematics and Statistics

“I want to extend my sincere apologies to all 43 national FIFA affiliates
for my use of the term ’soccer’ instead of ’football’.
’Football’ is and will always be the correct term.”
–Me

Abstract

Advanced analysis tools leveraging invasive tracking technologies such as GPS and manual event tagging has become a global staple in top-tier soccer clubs for enhancing their strategical decision making and insight. These tools rely on precise coordinate data, with their effectiveness significantly enhanced when this data is produced in real-time. With the rapid advancement in computer vision and GPU technology, machine learning models and trackers have become efficient and accurate, enabling real-time production of precise coordinate data using conventional hardware.

In this thesis we will present Sadji, a real-time soccer player localization and tracking system utilizing conventional hardware. We employ state-of-the-art YOLOv8 deep learning models coupled with multi-object trackers for player detection and tracking. We utilize SuperPoint for keypoint detection and feature matching to produce homography matrices for accurate translation of player coordinates between what the camera observes and a 2D soccer field image.

Through a series of experiments and iterative design choices we gradually improve fps throughput of the system while maintaining a high level of accuracy. Using a combination of interpolation and resolution reduction for input images to SuperPoint, we achieve system throughput speed over 30 fps, while maintaining a comparable position accuracy to that of state-of-the-art GPS tracking solutions.

Acknowledgements

I would like to give a heartfelt gratitude to my supervisors Professor Dag Johansen and Professor Martin Rypdal for their invaluable guidance, knowledge and interest. Your passion in the soccer domain is truly inspiring and has been a tremendous inspiration and driving force for me. I would also like to thank Tor-Arne Schmidt Nordmo for productive dialogue and valuable input.

I want to thank my office mates for great discussions, memories and well timed lunch breaks. A recipe for success and happiness includes good friends. Thank you all for being ones.

Finally, I want to thank my family for their endless love and support throughout the writing of this thesis and my five years here in Tromsø.

Thank you.

Contents

List of Figures

List of Tables

List of Abbreviations

AGNN Attention Graph Neural Network

AI Artificial Intelligence

API Application Programming Interface

BBOX bounding box

BRIEF Binary Robust Independent Elementary Features

BRISK Binary Robust invariant scalable keypoints

CCTV Closed-circuit television

COTS common of the shelf

CPU Central Processing Unit

CSG Cyber Security Group

DAC Deep Adaptive Clustering

DAI distributed artificial intelligence

DDC deep density-based image clustering

EU European Union

FAST Features from accelerated segment test

fps Frames Per Second

GDPR General Data Protection Regulation

GNN Graph Neural Network

GPS Global Positioning System

GPU Graphics Processing Unit

HLS HTTP Live Streaming

HTTP Hypertext Transfer Protocol

JPEG Joint Photographic Experts Group

JSON Javascript Object Notation

KBHMG keypoint-based homography matrix generator

LPS Local Positioning System

MOT Multi-object tracking

ms Milliseconds

OML Optimal Matching Layer

openCV Open Source Computer Vision Library

ORB Oriented Fast and Rotated BRIEF

PNG Portable Network Graphics

POC proof of concept

R-CNN Region-based Convolutional Neural Network

RAM Random Access Memory

RANSAC Random sample consensus

SIFT Scale-invariant feature transform

SP SuperPoint

SURF Speeded-Up Robust Features

TCP Transfer Control Protocol

TIL Tromsø Idrettslag

UiT University of Tromsø

URL Uniform Resource Locator

UWB Ultra-Wideband

VIK Viking Fotballklubb

VR Virtual Reality

WSL Windows Subsystem for Linux

YOLO You Only Look Once

1

Introduction

Rory Smith's book, "Expected Goals"[56] explores the realm of soccer analytics and advocates for the potential quantification of the game. Recent technological advancements in computer vision and machine learning, exemplified by video surveillance systems and wearable Global Positioning System (GPS) tracking devices like STATSports[58] used in top leagues such as the English Premier League and German Bundesliga, amplify the significance of Smith's argument. These quantifications can now be computed more accurate, faster, and closer to real-time, aligning with the pace of modern analysis in soccer.

These technological advancements, combined with tools like Hudl[24] for comprehensive video analysis, have fundamentally reshaped how teams evaluate both collective and individual player performances. In this field of evolving technology, Smith's arguments of quantifying soccer in his book emerges as a key component, offering a sophisticated metric that transcends conventional statistics and coaching approaches. "Expected Goals" provides an understanding of goal-scoring opportunities based or backed by the wealth of data generated by modern soccer technologies.

An example of a sport which has undergone this type of revolution, detailed in the book "Moneyball: the art of winning an unfair game"[32] by Michael Lewis, using quantified sports data is baseball. During the early 2000s, Oakland Athletics shifted towards utilizing analytics during the scouting periods as opposed to traditional methods, which mostly relied on subjective evaluations regarding skill and ability. With this conscious decision to transition into

utilizing more analytics, the team experienced great success securing playoffs several times while being one of the teams in the league with the lowest overall spending. While baseball, because of the games more structured nature, may be simpler to use such quantified data to gain an advantage we believe that the argument still stands as showcased by the great success of top soccer clubs that incorporate such methods.

Smith's argument for the quantification of soccer aligns seamlessly with the current technological wave, creating a need of deeper analysis and strategic improvement within the sport of soccer. The best teams in the Premier League[1] do this by placing a paramount focus on quantification in their analysis and strategic plans before, during, and after games. By harnessing cutting-edge technologies, advanced analytics, and real-time data, they meticulously evaluate player performances, uncover opponent strategies or style of play, and optimize tactical decisions on the field. This intersection of soccer, data, and technology, serves as a guiding compass for top-tier teams navigating the intricate landscape of modern soccer, where quantification is becoming more and more essential to the recipe of success.

The economically strongest clubs, in contrast to some of the weaker clubs, have abundant resources to spend on these advanced camera systems, video analysers, and player health analysers to improve their respective player and team performance as well as investigate in opposing team weaknesses. These clubs have a clear advantage in terms of manpower spending on both tagging the produced footprint data and install facilities that can produce footprint[2] data of high quality fast. The pipeline for tagging and producing said footprint data in videos can be illustrated as following:

1. If reliant on video, detect player and/or ball in video frame, either utilizing machine learning model or manually tag each player and ball in frame using trained personnel.

2. Convert said detection from frame coordinates to on-field coordinates. Utilizing GPS tracking devices such as vests would remove this step. Instead, one would only need to know the corner GPS coordinates of soccer field played on.

3. Use on-field coordinates for analysis such as style of play, space visualization, or passing network.

1. Premier League - top professional soccer league in England
2. Footprint data - positional player data

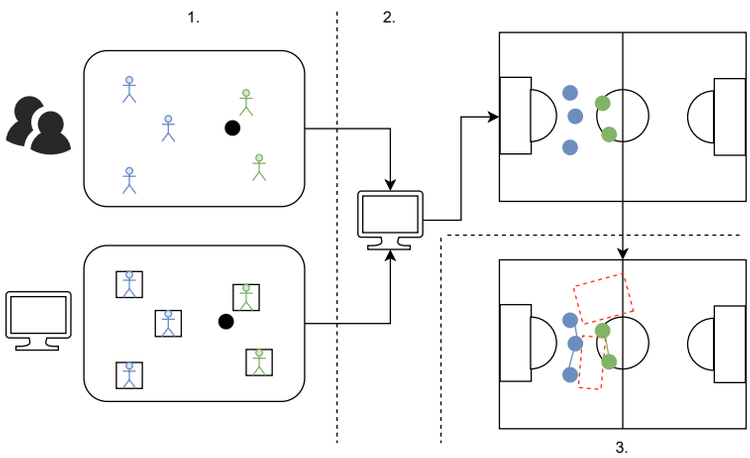


Figure 1.1: Simplified pipeline for tagging and producing footprint.

This thesis will focus on automating step 1 and 2 while preserving a high level of accuracy and speed utilizing video as input source to generate player coordinates/footprint data. The overall goal of the system is to get as close as possible to real-time using affordable measures that are suitable for *less economically strong* clubs to bridge the gap between them and the strong clubs. With recent advancements in the field of computer vision we hypothesize that it could be possible to accurately track objects in real-time using commodity hardware more suitable to the less strong clubs.

This project is carried out in connection with the Sárgut[10] and Guorrat[39] systems in the Cyber Security Group (CSG)[3] at UiT. The aim of the collaboration is to each work on their individual part of the illustrated pipeline where this thesis focuses on step 1 and 2, the two other theses focus on utilizing the produced player coordinates/footprint data from the system presented in this thesis for game analysis.

This introduction section will investigate further into motivation and existing systems that are available for similar use cases.

3. https://uit.no/research/csg

1.1 Motivation

As previously mentioned, economically strong soccer clubs have a major advantage in terms of resource spending on not only top tier players, but advanced technology and analysis staff, further increasing their advantage over the less economically strong clubs. Systems such as Hudl[24] or STATSport[58] are expensive and typically subscription based services, which inherently means teams themselves seldom "own" their footage exclusively. Some of these analysis services have long processing times, which works fine for a post game analysis but does little to nothing when analysis is wanted during half-time or during a game. Other drawbacks associated with these systems are often times lacking quality in their tagging, requiring manual work which again can be a cost problem for less economically strong clubs.

The precision of GPS systems are also questionable, as pointed out by this 2019 study on GPS precision in urban environments[37], when the use-case of said coordinates would be to compute advanced analysis regarding player formation, passing networks, and room detection where the accuracy level should be as high as possible. While these GPS tracking systems work well for calculating overall covered distance or speeds of a player, they are questionably sufficient for the needed footprint precision for a system which this project aims to create. GPS systems inherently also have the problem of not getting insight into opposing team positions, as team generally do not share such data between one another.

An alternative to STATSport GPS systems to track on field position of players is ZXY[45]. ZXY works by equipping each player with a belt capable of emitting radio signals received by on stadium installed towers. Position is computed by triangulating emitted signals.

Both these methods work well for detecting player coordinates on the field, but are both invasive in that each player must wear either a vest or a belt with a transmitter. ZXY is also reliant on pre-installed facilities that can capture the emitted radio signals, which could for resource scarce clubs be a problem.

Ownership and the right to delete personal data is also an interesting issue when it comes to the produced data from the GPS systems and recording systems. The General Data Protection Regulation (GDPR)[4] is a comprehensive data protection regulation that came into effect in the European Union on May 25, 2018. GDPR is specific to the EU, however its influence extends globally across the world, as it impacts organizations that handle the personal data of EU residents, such as Hudl or STATSport, regardless of where the organizations themselves are located. Now under the GDPR, individuals have explicitly the right to request the deletion of their personal data held by organizations. This

right, often referred to as the "right to be forgotten," empowers individuals to have their data erased under certain circumstances. Organizations must comply with such requests promptly, reinforcing individuals control over their own personal information and promoting data privacy. If these systems can guarantee such compliance is questionable at best. How the stored data is used by these companies is also not public as limiting the availability of algorithms helps maintain their competitive advantage as well as safeguard proprietary methods used. Such external services create a situation for a potential customer where their data is locked-in and the customers data is not fully manageable or controlled.

A lack of transparency regarding data and Artificial Intelligence (AI) usage poses significant problems. Regulations are being introduced to protect fundamental rights from high-risk AI. However a lack of transparency in organizational practices hinders accountability and undermines trust. Banned applications, such as those involving biometric categorization and emotion recognition, highlight the need for clear guidelines. Exemptions for law enforcement highlight the importance of strict oversight mechanisms and in detail description of how such systems function. Transparency requirements for AI systems, including publishing training data summaries and labeling manipulated content, aim to mitigate risks and promote responsible innovation. However, ensuring compliance and effective implementation remains crucial for upholding European values and safeguarding societal interests in the era of advancing technology. These new rules and regulations are apart of EU's new AI Act[42].

Recent advancement in object detection models such as YOLOv8[26] are able to accurately detect players and classify them based on for example teams affiliation in video in real-time. Such machine learning models combined with systems for tracking said detections through multiple frames also exist, such as ByteTrack[65] or DeepSORT[64]. These systems are called Multi-object tracking (MOT) and work by locating and following multiple objects detected by a machine learning model in a sequence of video frames. These new systems combined open up many new possibilities for running advanced software on common of the shelf (COTS) hardware that is more accessible to lower income clubs. Off the shelf graphics cards have significantly improved with increased processing power, advanced architectures supporting features like real-time ray tracing, significantly higher memory capacities, and more efficient manufacturing processes, resulting in better performance capabilities and pricing.

1.2 Problem Definition

Quantification of soccer is becoming more and more important to complement the type of analysis that top tier soccer clubs are conducting in major sporting organizations. Thus it is decisive that teams can extract relevant data fast and of high quality. Existing solutions for teams today, especially those of lower economical strength, rely on third-party solutions to capture, analyze, and for storage. This in turn, can lead to time delays, high costs, and uncertainties on the overall quality of produced analysis. The most accurate solutions also rely on invasive devices such as GPS-vests for producing positional data, which often is only shared on a per team basis meaning that the opposing team will not have access to half the player coordinates. This thesis shall create a system overcoming these problems by using video stream or clips as input together with object detection models and Multi-object tracking (MOT) to accurately map player field position in the video. Then translate their video frame coordinates utilizing keypoint detection and matching algorithms to a user provided birds-eye view or top-down view of a soccer field (see figure 1.1) in real time.

The thesis problem definition is the following:

> ***This thesis aims to develop a system capable of automatically and precisely detect and track soccer players in video footage. Utilizing computer vision, the system will be capable of determining the team affiliation of each player. The goal is to promptly and accurately map players to their respective on-field coordinates, with the goal of real-time speeds. The research within this thesis will explore the YOLOv8 deep learning models for the precise detection of players and their teams in video content using MOTs for tracking. Additionally, the investigation will look into methods for accurate and efficient spatial coordinate mapping of soccer players from video to a top-down view image of a soccer field.***

Investigation of given statements above will be done in the following order:

- Examine what related work exists to become acquainted with the current state of the domain.
- Define functional and non-functional requirements for the proposed system.
- Design and implement system with provided defined requirements.

- Evaluate system by conducting a series of experiments on different parts of it. Experiments will also serve as a foundation for design choices that were made throughout the design/implementation process.

1.3 Methodology

The framework detailed in "Computing as a Discipline"[13] serves as a fundamental building stone for understanding the many-sided field of computing, highlighting important subjects and methodologies within the domain of computer science. The report details three distinct paradigms with each paradigm offering unique perspectives and methodologies for the advancement of computational knowledge and practice.

1.3.1 Theory

Rooted in mathematical principles, the theoretical paradigm in computer science revolves around the formulation and validation of mathematical relationships. At its core, theory serves as the bedrock which the structure of computational science is built on, providing a systematical framework for the conceptualization and analysis of computational phenomenas. The paradigm is comprised of the following four steps:

- *Definition* - At the outset, theory involves the accurate characterization objects that are under study. This step lays the groundwork for subsequent theoretical inquiries by stating clear boundaries and definitions.

- *Theorem* - Building upon the defined objects, step two entails hypothesis, showcasing potential relationships and properties among the defined objects.

- *Proof* - Theory demands empirical validation through the process of proof to find truth. Through reasoning and deduction, one can validate the proposed relationships.

- *Interpretation* - interpretation and analysis of the achieved results. This step is about exploring the theoretical findings in a larger picture.

1.3.2 Abstraction

Contrasting with the theoretical nature of the previous paradigm, abstraction approaches with a more pragmatic and empirical way for computational questions. Abstraction uses experiments to simplify complicated computer questions into easier models for which we can study. The abstraction paradigm contains four stages:

- *Hypothesis* - Drawing inspiration from theoretical insights, first step of abstraction begins with formulating a testable hypotheses aimed at detailing computational phenomena. These hypotheses serve as guidelines for empirical investigations.

- *Prediction* - This stage is about making models that can predict how computer systems will behave. These models will help us analyze and test things before they happen.

- *Experimentation* - Once we have our predictive models, this stage begins by executing experiments to explore computer phenomenas. These experiments check if our theories are right and make our models better, helping connect theory with real-world use cases.

- *Analysis* - Using the data produced in the following stage, abstraction carefully investigates the results. This step involves analysing what we observe in the experiments with what we expected from our theories from stage 2, helping us improve our computer models.

1.3.3 Design

Leading to the combination of theories and practical evidence, the design paradigm exemplifies the practical creation of computational principles to address real-world problems. It is rooted in engineering principles, where design represents the culmination of theoretical abstraction into tangible systems and solutions. The design paradigm has four stages:

- *Requirement Specification* - Design commences with a meticulous specification of system requirements, detailing the functional and non-functional specifications that dictate behavior and performance.

- *System Specification* - Building on the requirements, design proceeds to articulate comprehensive system specifications, outlining the architectural blueprints and design constraints for the system development.

- *Implementation* - With specifications in hand, design transitions into the system implementation, detailing the actual construction and realization of computational systems.

- *Testing/Experimenting* - The final stage of design is about testing and validation of the constructed systems against predefined specifications. Through systematic testing methodologies, design choices to ensure the reliability, robustness, and efficiency of computational systems in fulfilling their intended purposes.

In summary, the design paradigm emphasizes the pragmatic application of computational principles, bridging the theoretical and empirical paradigms to generate tangible solutions to real-world challenges. The methodology used in this thesis has its roots in the design paradigm, embracing the theoretical insights and empirical findings to inform the development of a functioning system.

1.4 Scope and Limitation

This section will cover the scope of the thesis, detailing what limitations are predefined for the design and implementation of Sadji[4]. By listing what boundaries and limitations the system operates in and identifying the constraints, we aim to help give a comprehensive understanding of the thesis objective and constraints.

- Video/stream source will be assumed as functional in the pipeline defined in section 4.2. Sadji does not initiate any recording and operates as a passive component that initiates once a user of the pipeline wishes to start the coordinate production pipeline. Sadji will be capable of utilizing any video/stream source which conforms to OpenCV's video formats.

- Sadji assumes that it is a part of a functioning pipeline where, in the future, it is part of a pipeline where recording and usage of the produced coordinates is functional. See figure 3.1 for illustration. System position is highlighted in red.

- Sadji assumes that the detection model capable of also classifying players is pre-trained. System defaults to the detection model capable of player detection only and not team classification.

4. **Sadji** - The word for "*position*" in North Sámi.

1.5 Context

This thesis is integrated in a collaboration with the CSG at University of Tromsø (UiT). CSG is a research group focused on providing new knowledge and innovative distributed system technologies in computer science. Their multidisciplinary research spans various faculties, with an emphasis on creating new knowledge, tools, and innovative technologies in different disciplines such as for example sport surveillance systems and AI.

CSG collaborated on a case study quantifying soccer using positional data in 2018[44], comparing radio-based wearable positioning data systems with GPS systems. The findings revealed the ability to detect anomalies, identify trends, and offer valuable insights for individual players and team performance development. CSG's research in the computer science domain also focuses on constructing scalable, efficient, fault-tolerant systems for real world applications such as sports analysis. They actively deploy these systems in real-world settings, encouraging user involvement and following an open publishing and distribution policy for their software. Mearka[59], published in 2023 by Alexander Torkelsen is one example of such a system utilizing computer vision technologies to detect soccer players in video clips combined with manual input for event captures.

Key personnel at CSG have been part of research in the distributed systems and AI domain for decades. StormCast[21][22] published in 1988 presents a novel prototype for a distributed artificial intelligence (DAI) application focusing on presenting severe storm prediction. The paper recognized the importance of accurate predictions for local fishing communities in northern Norway, introducing a novel solution for gathering challenging aspects of local weather forecasting such as cloudiness and local topography deviations effects. The paper presents limitations of existing solutions at the time and proposes DAI for gathering local weather forecast information before sending those local summaries to an expert user (experienced meteorologist).

CSG's involvement in sports science includes systems like Bagadus, Muithu, Darkon and Mearka, Bagadus [20] as mentioned combines a sensor system, soccer analytics annotations, and video processing. Muithu [27] integrates real-time coach notations with video sequences. Darkon[28] is a video analysis system designed to assist teams in meeting their video analysis requirements. In tandem with a concurrently developed tagging system, Dárkon facilitates filtering based on soccer field positions, event type, and event outcome, enabling users to locate specific events or create playlists.

This thesis will contribute to the ongoing efforts by the CSG at UiT by exploring and deploying a system that can produce accurate on field player coordinates

in real-time. The coordinates produced will be essential for future systems that can take advantage of produced coordinates in AI-based analysis for the identification of high-level patterns and trends that emerge throughout a game of soccer. Such a digital twin coach can empower coaches to make informed decisions, optimize their strategies and enhance player performance.

A digital twin coach, equipped with this kind of capability, operates as an impartial observer, unaffected by the intensity of the game. It could potentially offer unparalleled insights into various game aspects, from player positioning and movement patterns to team dynamics and tactical formations by looking at for example space in between opponent formation lines. By solely relying on observations, it provides a comprehensive and unbiased view of the match, supplementing coaches with analysis about game-play dynamics in real-time and tailor strategies without being influenced by the heat of the moment.

Post-match analysis facilitated by the digital twin coach offers detailed reports and visualizations based purely on collected coordinates. This allows coaches to objectively review key moments, assess player performances, and pinpoint areas for improvement. By integrating advanced data analytics with real-time player tracking, the digital twin coach emerges as an invaluable tool for enhancing coaching effectiveness and driving success on the soccer field, all while maintaining an objective perspective uninfluenced by the emotional highs and lows of a soccer game.

1.6 Outline

Brief description of the chapters of this thesis:

- Chapter 2 will detail about technologies used to build Sadji. The section will also investigate existing systems and concepts in the current state of the domain and some of the systems referenced in chapter 1.

- Chapter 3 will detail the requirements, both functional and non-functional for the proposed system.

- Chapter 4 presents the overall design of Sadji showcasing the individual parts of the coordinate production pipeline. Implementation details are also included with references from evaluation section 5 on why specific design/implementation decisions were chosen.

- Chapter 5 showcases achieved results from deploying Sadji with various different configurations. It also contains implementation testing

throughout the development process of Sadji.

- Chapter 6 will use achieved results from chapter 5 on evaluation for discussion and investigate Sadji using pre-defined requirements from chapter 3.

- Chapter 7 will conclude the thesis and give a summarization of the Sadji system and present future work and integration in new iterations.

2 Background

There are primarily two methods of implementing tracking systems in soccer. The first method involves non-invasive technologies, like Closed-circuit television (CCTV). This process involves capturing the soccer game and then either simultaneously or in post-processing use computer vision technologies to detect players and other objects. These systems rely on visual data as input to track and monitor players without equipping the players with any devices.

Another method entails invasive technologies, such as GPS vests, which require individuals to wear devices that continuously transmit their location data to nearby antennas or via satellite. While non-invasive systems offer flexibility and ease of implementation, invasive technologies provide precise and real-time location information but may raise privacy concerns due to their direct monitoring approach.

2.1 Non-invasive Tracking Technologies

In this section we will investigate some non-invasive tracking technologies and techniques which are prevalent in various fields, including surveillance, sports analytics, and object detection. These technologies offer sophisticated solutions for monitoring and analyzing movements and activities without direct physical contact. Non-invasive tracking technologies utilize camera sources to monitor and analyze movements (without physical contact). The resolution

and positioning of the camera are crucial factors determining the accuracy and effectiveness of the tracking system. Low camera placement could potentially result in multiple tracking elements being occluded. Similarly, a camera source with low frame-rate and resolution could result in missing important information.

2.1.1 YOLO

You Only Look Once (YOLO)[48] is a well known object detection algorithm known particular in the deep learning community for its speed and accuracy. Traditional object detection algorithms generally use region proposal techniques for the generation of bounding boxes or regions that are likely to contain objects in an image. These generated bounding boxes are then passed on to a classifier which determines if an object is contained in the bounding box or not, and if so, which class the contained object belongs to. Region proposal techniques help with narrowing down the search space, while classification algorithms accurately classify objects within the proposed regions. This approach is commonly used in object detection systems like Faster Region-based Convolutional Neural Network (R-CNN) or Region-based Convolutional Neural Network (R-CNN). However, this process can be unsuitable for real-time applications due to its inherently computational expensive operations.

YOLO differs from such traditional methods in that YOLO directly predicts bounding boxes or regions and classes for multiple objects in an image simultaneously within a single neural network. An image fed to YOLO is divided into a grid and the system will then for each cell within the grid, predict bounding boxes and class probability. This process is illustrated in figure 2.1 from the *You Only Look Once:Unified, Real-Time Object Detection*[48] paper.

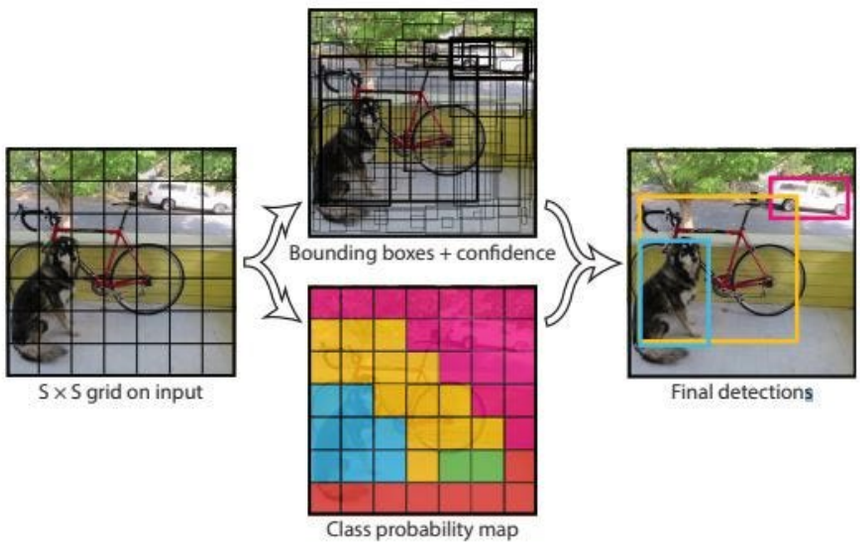


Figure 2.1: YOLO detection pipeline[48].

YOLO achieves faster inference time while maintaining a high level of detection accuracy by eliminating separate region proposal and classification steps making it a good alternative to systems in need of real-time performances.

YOLOv8

At the time of writing this thesis, YOLOv8 [26] is the newest iteration of the YOLO object detection algorithm. YOLOv8 builds on previous iterations and offers state-of-the-art performances in both speed and accuracy. One of its new key features has been on optimizing accuracy-speed trade-offs making this iteration more suitable for systems in need of incorporating object detection tasks in real-time execution speeds.

2.1.2 Ultralytics

Ultralytics[1] [25] is an organization specializing in computer vision and deep learning technology developing state-of-the-art software tools and libraries to seamlessly train and run different deep learning models. Its open-source with their primary focuses being providing accessible and high-performing solutions for image classification, object detection, instance segmentation, and

1. URL to GitHub: `https://github.com/ultralytics/ultralytics`

pose estimation. The Ultralytics framework allows for user-friendly deployment and training of custom deep learning models. Training and testing datasets follow a user-friendly labeling structure enabling an efficient manual labeling process. Ultralytics also enables seamless integration with different multiple object trackers such as ByteTrack[65] or BotSort[5].

2.1.3 Multiple Object Tracking

Multi-object tracking (MOT) is a field within computer vision that deals with the task of simultaneously detecting and tracking multiple objects in video sequences. MOTs deal with problems such as identifying and maintaining different trajectories of detected objects between sequences of frames and associating disappearing and reappearing detections. MOTs can be deployed in various applications such as sport surveillance systems, autonomous vehicles [47], underwater monitoring [34], or social distance monitoring [7]. MOTs deploy various techniques to overcome challenges with occlusion, blending backgrounds, and varying appearances such as size change or rotation. Some of these techniques include feature association, motion estimation, and other data association algorithms such as nearest neighbor [18].

ByteTrack

ByteTrack[65] is a MOT that introduces a new data association method by including low scoring bounding boxes for detections. Typically in MOTs, detections are categorized into low and high scoring objects. High scoring detection generally have little to no occlusion within them and are above a given threshold set by the MOT. Low scoring detections are usually occluded. ByteTrack includes both high and low scoring detections and works by:

- Associate detection with high scoring tracklets. Tracklets that do not match a high scoring detection box are unmatched.

- Associate the low scoring detection boxes and the unmatched tracklets from step 1 to recover objects in low scoring boxes. The ByteTrack paper [65] illustrates this sequence in Algorithm 1 pseudo-code.

As highlighted in the paper, ByteTrack shows great performance metrics on conventional hardware when compared to other state-of-the-art MOTs at the time.

BotSort

BotSort[5] combines both appearance and motion information to better predict corresponding bounding boxes. This combined with camera-motion compensation which means that unlike common MOTs they are able to associate bounding boxes that don't necessarily overlap between frames. They have also designed a new Kalman filter[29] that is able to more accurately encapsulate the detected object.

2.2 Invasive Tracking Technologies

In this section we will investigate the domain of invasive tracking technologies. These tracking devices typically operate by equipping players with signal emitting devices such as GPS, Local Positioning System (LPS) or Ultra-Wideband (UWB). The context of usage is what drives selection of the different systems. GPS works for outdoor tracking while is limited for indoors tracking. LPS works for indoor monitoring but cannot transmit data over large areas like GPS can.

2.2.1 Global Positioning System

GPS provides a wide coverage for determining the location by triangulating signals with satellites in orbit. It offers positioning capabilities on a global scale making it versatile for outdoor usage. Its main drawback is that its functionality is limited for indoor or urban areas as these areas create an obstruction for the line-of-sight between the transmitter and satellites.

2.2.2 Ultra-Wideband

UWB is a positioning technology which utilizes short-duration pulses of radio frequency energy to accurately determine the location of objects in three-dimensional space. It works by measuring the time it takes for the pulses to travel between a transmitter and multiple receivers installed within the tracking area. UWB is highly accurate achieving centimeter precision as showcased in a experimental study of a UWB for industrial usage [55]. Some drawbacks with UWB are initial setup costs which can be high as pre-installed receivers is a condition for the system to function. Areas with dense obstructions or other interference sources can also challenge accuracy of the system.

2.2.3 VICON

VICON [61] is a system that provides optical motion capture technology widely used in sports biomechanics, animation, and Virtual Reality (VR). Their system consists of several high-resolution cameras placed in strategically positions around a capturing area, using reflective markers placed on athletes or objects to track with accuracy. VICON systems capture real-time data such as movement which allows for detailed analysis of kinetics and kinematics. Some drawbacks with the VICON system are occlusions problems associated with markers line of sight being obscured by limbs and clothe-wear. They are also primarily used indoors due to their reliance on lightning conditions needing to be controlled.

2.3 System Specific Technologies

This section will cover some of the technologies used to create the Sadji system. Some of the technologies have already been covered in section 2.1 and 2.2.

2.3.1 OpenCV

Open Source Computer Vision Library (OpenCV) [9] is a widely used opens-source library for image and computer vision tasks. OpenCV was originally developed by Intel in 1999 and has since become a standard tool in computer vision. It offers a wide range of functionalities, from basic image manipulation to more complex tasks such as object detection and recognition. OpenCV is available as an import library in Python from bindings in the original *C++* library offering all of the functionality seamlessly through Python scripts.

OpenCV in Python supports a wide range of different image formats and provides data structures for representing images and their associated metadata (JPEG, PNG, etc). It offers functions for performing basic operations like resizing, cropping, and filtering, as well as more advanced techniques such as image stitching or feature detection. OpenCV also has an extensive collection of pre-trained models and algorithms, including homography estimation, machine learning-based methods like Haar cascades for object detection, Speeded-Up Robust Features (SURF) and Scale-invariant feature transform (SIFT) for feature extraction. OpenCV also has an active community and comprehensive documentation making it easy to understand and maintain for new developers. Cross-platform support and compatibility with other popular libraries and frameworks such as NumPy, SciPy, and TensorFlow, makes OpenCV a good tool for researchers and practitioners in various different domains including sport

science, autonomous vehicles, or augmented reality.

2.3.2 Homography

Homography matrices are common in computer vision technologies for representing transformations between different perspectives or views of a common area. They describe mapping between two cameras or images that record or photograph a common plane. Typically, they are used in scenarios where images are taken from different angles or perspectives for example in sport broadcasting or surveillance systems. Some common areas where homography matrices are used in image stitching, panoramic image creation, or augmented reality.

How they work

Homography matrices represent the translation between coordinates in one image to coordinates in a different image in what is called a perspective projection. Perspective distortion happens when a three-dimensional scene is projected onto a two dimensional image plane. Homography matrices encode this transformation, allowing accurate manipulation and representation of the distorted perspective. Figure 2.2 shows a use case of perspective projection of a frame from a video clip to its destination in a panorama image.

Figure 2.2: Perspective project of frame in video clip to panorama image.

A homography matrix is represented by a 3x3 matrix and given n corresponding coordinates:

$$\begin{aligned} \mathrm{p}_i &= [x_i, y_i, 1]^T \quad \text{in the first image,} \\ \mathrm{p}'_i &= [x'_i, y'_i, 1]^T \quad \text{in the second image.} \end{aligned}$$

between two different images can be compFuted by:

$$\begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i'x_i & x_i'y_i & x_i' \\ 0 & 0 & 0 & -x_i & -y_i & -1 & y_i'x_i & y_i'y_i & y_i' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0.$$

1. Solve the overdetermined linear system $Ah = 0$ using for example singular value decomposition. The solution vector h corresponds to the elements of the homography matrix H.

2. Reshape the solution vector h into a 3×3 matrix H and ensure that the last element of H is normalized to 1.

Cv2.findHomography is an OpenCV function that computes the homography matrix from a set of corresponding points in two images. It utilizes Random sample consensus (RANSAC) or Least Median of Squares to estimate the homography, which is useful for tasks like image registration, object tracking, and panorama stitching in computer vision systems.

2.3.3 Random Sample Consensus

RANSAC is an iterative algorithm that is used to estimate model parameters for a data set that contains outliers. It randomly extracts subsets of the original data to estimate model parameters and consequently quality checks each model with new parameters by counting number of non-outliers. This process is done several times until a model is chosen with the least amount of outliers. The model which has the fewest amount of outliers will best fit the original data.

2.3.4 Keypoint Detection and Matching Algorithms

This section writes in short detail about different algorithms explored and used for keypoint detection and matching algorithms.

ORB

Oriented Fast and Rotated BRIEF (ORB) [51] is an efficient feature detection and description algorithm that couples the Features from accelerated segment test (FAST) keypoint detector with Binary Robust Independent Elementary Features (BRIEF) descriptor. ORB is commonly used in real-time application such as augmented reality or object tracking because of its speed and scale/rotation invariance. ORB works by generating binary descriptors which in return reduce computation time while still being able to function accurately.

SURF

Speeded-Up Robust Features (SURF) [8] employs integral images for fast computation and uses a combination of Gaussian and box filters to efficiently detect keypoints. Similar to ORB, SURF is a good fit for systems in need of speed and scale/rotation invariance such as image stitching and object recognition.

SIFT

Scale-invariant feature transform (SIFT) [35] works by identifying keypoints based on high-scale space and extracts descriptors by utilizing histograms of rising orientations. It is very robust to variation in scale, rotation, illumination as well as view-point changes making it a good match for many different imaging conditions.

2.3.5 SuperPoint

SuperPoint: Self-Supervised Interest Point Detection and Description [14] presents an alternative to traditional methods for point detection and description. It is a fully convolutional neural network which is designed to handle operations on fully scaled images having the ability to generate interest points together with a fixed-length descriptor in one forward pass.

The architecture of SuperPoint starts with an encoder responsible for processing and reducing the extent of dimensions of the input image. This encoder is shared. Next step after the encoder is that the network splits into two parts or two decoders where each one of them specializes in a specific task. One of the decoders specializes in interest point detection while the other specializes in interest point description. This splitting allows for communication which differs from traditional methods. The interest point detector produces

a probability of "point-ness" for each pixel in the input image meaning each pixel in the networks output represent some likelihood of being important point in the original image. Unlike typical dense prediction networks with encoder-decoder pairs and upsampling, SuperPoint's explicit decoder reduces computational overhead. The descriptor of the neural network creates a compact representation, which is called a description, for each pixel in a smaller version of the original image. To do this, the descriptor decoder generates a grid of descriptions at specified points and then fills the gaps using bicubic interpolation. Then, it normalizes the descriptors to ensure consistency. This process helps with identifying features in images efficiently.

The loss function evaluates how accurately the neural network detects interest points and creates corresponding descriptors. It combines two intermediate losses:

- One for interest point detection.
- One for descriptor creation.

For interest point detection, the loss measures the difference between predicted and actual interest points in the images using a cross-entropy loss function. This penalizes the network more for incorrect predictions, ensuring it focuses on accurate detection. The loss function evaluates descriptor creation by comparing predicted descriptors to the ground-truth descriptors using a hinge loss function. This penalizes larger errors in descriptor creation, but within certain margins, preventing excessive penalties for small deviations. By balancing these losses and minimizing the combined score, the network adjusts its predictions to improve both interest point detection and descriptor creation accuracy.

2.3.6 SuperGlue

SuperGlue: Learning Feature Matching with Graph Neural Networks [54] is a method for enhancing feature matching. It provides improvement for both accuracy and robustness between keypoints that match in images by integrating a Graph Neural Network (GNN) into the SuperPoint architecture (as detailed in 2.3.5).

SuperGlue consists of two components: The Attention Graph Neural Network (AGNN) and the Optimal Matching Layer (OML).

The AGNN component is responsible for learning and reasoning about correspondences between different keypoints detected within different images.

It uses a graph neural network architecture and works by treating keypoints in images as nodes in a graph. It uses attention mechanisms to aggregate information between keypoints that are neighbors in the graph. This enables AGNN to capture both semantic similarities and spatial relationships between keypoints.

The OML is the second component and is responsible for computing the best correspondences between keypoints in pairs of images. Is uses learned embedding of keypoints from said image pair and a learned similarity metrics to determine the best matches. The component employs a differentiable mechanism to solve an assignment problem. This mechanism is targeted at maximizing the overall similarity between two keypoints in an image pair while ensuring each one of the keypoints is paired with at most only one other keypoints in the other image. This is what allows for robust and accurate matching between keypoints even when there are occurrences of noise or occlusions.

SuperPoint and SuperGlue have a library extension for Python which simplifies deployment of the system [60]. The library enables researchers or developers to integrate the functionality of both into, for example, a computer vision pipeline for keypoint extraction and matching between images.

2.3.7 HTTP

Hypertext Transfer Protocol (HTTP)[17] is the foundational protocol for communication over Internet. It enables transfer of information between a client and a server, operating on a request/response basis. It functions over Transfer Control Protocol (TCP) and IP and employs various methods for communications such as GET, POST, and PUT requests to perform different actions such as deletion, updating, or retrieving data between a client and a server.

2.3.8 Web API

Web Application Programming Interface (API) serves as an intermediate in software for communication and interaction between components or servers typically adhering to protocols such as HTTP. It works by abstracting away complex implementation details, promoting modularity and interoperability facilitating a powerful and flexible way for developers to efficiently setup communication in software. Flask[19] is a common Python web framework that facilitates web API development.

2.3.9 Hudl

Hudl is a commonly used industry platform for sports and performance analysis. Hudl offers a comprehensive set of tools for coaches, athletes and analysts for facilitating and capturing sports play, health monitoring, and individual performance data. Hudl is the main recording and analysis software used in the Norwegian soccer league Eliteserien[57]. For this thesis, we have been granted access to Hudl content through our collaboration with Norwegian Eliteserie team Tromsø Idrettslag (TIL) who play in the top Norwegian soccer league for Tromsø.

2.4 Related Work

Earlier work has been done by the CSG research group at UiT using machine learning techniques to automate the process of tagging soccer players and soccer balls in videos. Mearka[59] provides player detection using machine learning models that rely on CPU usage and not GPU, which significantly increases the overall time of inference. This thesis aims at producing ground truth (player and ball position) faster by utilizing GPU power for running machine learning models for inference. The thesis focuses more on tracking a player over several frames using multi-object tracking[65] for said machine learning models to track players over the course of a half-time or individual set pieces. The main goal is to implement a system that is convenient to run on mid-tier computer hardware using pre-installed cameras at the different soccer stadiums. The hardware more specifically are 30-series NVIDIA graphic cards and high amounts of memory (up to 128 GB), system hardware specifications are listed in chapter 4 on evaluation.

2.4.1 Mearka

Mearka, which is a distributed soccer tagging system using conventional hardware components, uses YOLOv4 models for inference on the video footage of soccer games to detect players. This thesis also uses the YOLO framework, but uses newer versions for faster, more accurate and more with more functionality compared to previous versions. The speed of the inference is limited to the speed of the CPU because Mearka does not utilize GPU power to run the models. These earlier models also have no user friendly way of incorporating multi-object tracking such as ByteTrack to keep track of objects. Newer YOLO version such as YOLOv8 have support for such trackers and is what this thesis uses to achieve real time inference with accurate tracking over multiple frames. The newer models also have easier support for using the GPU instead of CPU

by tweaking inference parameters.

Through the Mearka app, coaches can manually tag events (save a timestamp) live during a match using their mobile phones, to create a highlight or smaller video of the uploaded match using the timestamp. In post-processing, machine learning models are run to create player position metadata which can be used for further analysis. This thesis aims at producing this said player and ball metadata closer to real-time so that analysis may be performed during a match.

2.4.2 Muithu

Muithu[27] provided a cost-effective alternative to resource-intensive tagging systems that traditionally required a team of several individuals to tag events over the course of a match or training session. This traditional approach can be expensive and result in tagging irrelevant events, typically performed by someone other than the head coach. In contrast, Muithu utilized affordable consumer cameras and a mobile app, allowing the head coach to tag events in real-time. This thesis also focuses on using cost-effective alternatives which the smaller clubs are more likely to obtain than the big clubs.

Involving the head coach in the tagging process ensured that only relevant events were recorded. With the ability to preview a situation before tagging, precision and recall rates neared 100 percent. The user-friendly Muithu app enabled the coach to tag "who" and "what" within five seconds. After recording, a timeline displayed all tagged events, and specific video segments for each event could be extracted. Each tag included an offset within the recording, streamlining the workflow, which was significantly faster than manual retrospective review.

This thesis builds on similar concepts as Muithu. The idea is that the system is able to accurately extract all ground truth from the video stream or file of a soccer match after or during a game and pass this metadata on. How the metadata is used in analysis is outside the scope of this thesis, but it is important to mention to provide context. This analysis based on the produced metadata is then provided to the coaches through some user interface and can be used as a supplement for game decisions such as substitution or formation changes. On the premise that the ground truth is accurate a coach could also be presented with analysis about a players total run distance, passes, sprints, position in formation lines, etc. With such insight a coach can make game changing decisions in real-time.

2.4.3 Quantified Soccer Using Positional Data: A Case Study

This case study from 2018[44] highlights the impact a system capable of accurately tracking and positioning of players over the course of a training session or match has. This thesis aims at producing metadata which can be used to create such object performance data on players or teams to quantify how a player or team is performing and what measures need to be taken to improve using only video as a source and not invasive tools such as tracking vests.

The motivation for not using invasive tools is that such tools only produces position data for the team incorporating them and not the opposing team. Most radio-based positional systems only works on a teams home stadium where they have installed ground-stations that can pick up or register the radio signals. Video is not reliant on such prerequisites on modern stadiums where video systems such as Hudl (which is available to all with a Hudl subscription) or Live Soccer Streams.

2.4.4 Pixel2Field

Pixel2Field[43] is a system designed to automatically convert distorted sports video frames into precise scaled 2D field maps. Unlike existing methods at the time that often rely on invasive player-worn sensors or manual calibration, Pixel2Field exploits sports field characteristics and key-points detection for a fully automated process.

In the undistorting phase, the system estimates radial distortion coefficients by utilizing sports field properties, eliminating the need for camera parameters. The homography recovery phase employs key-points detection for automatic recovery of the homography transformation. Experimental results using a professional soccer game dataset illustrate the system's efficiency, demonstrating straightened borderlines and accurate mapping of player positions.

Pixel2Field's fully automatic nature opened up promising opportunities for real-time sports analytics with a birds-eye-view 2D map. The paper suggests potential applications on mobile platforms, hinting at the prospect of real-time sports analytics coupled with different types of battery optimization techniques. Pixel2Field represents a significant advancement in transforming distorted video frames in sports into actionable 2D field maps for enhanced analytics. This thesis uses proposed homography methods for calculating player coordinate coupled with newer and improved methods for finding key points between a large panoramic image and the frame to be translated.

2.4.5 Bagadus

Bagadus[20] from 2014 is a novel prototype system designed for soccer analytics in real-time. It combines several advanced technologies, including a sensor system, a soccer analytics annotations system, and a camera array video processing system, to offer comprehensive insights into live soccer matches. The system application is split into three parts:

- Video subsystem - made up of multiple small cameras that record a soccer field which combined cover the entire soccer field with overlapping segments as to detect common features for stitching.

- Tracking subsystem - which uses ZXY[45]. ZXY is a sensor-based solution that works by equipping each player with a radio belt emitter where the signals produced can be picked up by installed radio towers. The ZXY Sport Tracking system captures player data such as position, heading, and speed at 10 Hz, along with additional statistics like total distance covered.

- Analytics subsystem - allows quick registration of predefined events or textual annotations on tablets or mobile phones during a live match. The recorded events are stored in a database, enabling automatic extraction and presentation alongside relevant video footage. This modernizes and simplifies the game analysis and could replace the traditional pen and paper analysis methods used by coaches.

Bagadus was one of the first systems of its kind in the domain, laying the foundations for later commercial solutions such as Hudl. This thesis draws a lot of inspiration from this system, in particularly the first and second parts of the split system, and aims to improve the overall efficiency using machine learning approaches and the complete removal of invasive positioning equipment.

2.5 Summary

In this section we have showcased different non-invasive tracking technologies, exploring YOLO and the newest YOLOv8 version available combined with MOTs such as ByteTrack. Section 2.2 details what invasive tracking technologies are common in the domain, where GPS is commonly used for its global scale and versatility. In section 2.3 we detail what technologies are used to implement Sadji, further detailing SuperPoint and SuperGlue as these are major components in the implementation. We have also introduced OpenCV, which is a Python extension tool for images and videos. Lastly, we have investigated

related work in section 2.4, highlighting existing work done at the CSG at UiT The Arctic University of Norway.

3

System Requirement

This chapter will detail the requirement specifications in accordance with the design paradigm stated in section 1.3.3. Section 3.1 will detail the specific functionalities and features that Sadji will provide the user. Section 3.2 will describe how Sadji should perform with regards to speed, accuracy, automation, performance on common of the shelf (COTS) hardware, and data ownership.

3.1 Functional Requirements

Functional requirements should specify what the system provides a user with. It should clearly state that, with specific provided input from a user, how the system behaves in accordance with provided input (see section 1.3.3). The outline for the functional requirements are:

- **Non-invasive tracking** - Sadji shall support tracking of players on a soccer field in video without using invasive technologies such as GPS.

- **Detection model** - Sadji shall support different Yolov8 detections model for detecting/classifying objects that the user wishes. This is to accommodate a users preference with regards to detecting players, their team affiliation and/or the soccer ball.

- **Video source** - Sadji shall support different video sources, whether it is

a stream source URL or a normal video file (*.mp4*, *.avi*, etc.).

- **Frame-panorama mapping** - Sadji shall be able to accurately translate frames from the video source into a panoramic image of the same area that the video source is capturing.

- **Coordinate translation to user provided soccer image** - Sadji shall be able to map player coordinates from video frames to their location on a user provided image of a football field. This shall happen at real-time speeds.

- **Formatted metadata for user specification** - Sadji shall be able to aggregate metadata depending on how the end-user wishes them to be. This shall change depending on the video source that the user has submitted.

- **Panoramic image creation** - Sadji shall be able to create a panoramic image from the video source that a user provides.

- **Configurable** - Sadji shall be designed so that it can be configured to a user's specified requirements. This is so that a user of the system can alternate between different configurations to extract the most out of the system in terms of accuracy and speed.

3.2 Non-functional Requirements

Non-functional requirements shall detail what characteristics or qualities the proposed system provides.

- **Real-time** - Sadji shall be able to perform in real-time, translating player coordinates from what the video source monitors to where that player is on the soccer field at any given time. This is to adhere with user requirements that coordinates shall be produced during, for example, a soccer game or training, and there is therefore need of real-time speeds.

- **Accuracy** - Sadji shall produce accurate soccer-field coordinates regardless of optimal camera specifications such as resolution or camera placement.

- **Automatic coordinate production** - Sadji shall be able to automatically produce player coordinates from a video source provided that panorama image, 2D soccer field image, and corresponding mapping keypoints

between them are provided.

- **Integration** - Sadji shall be designed and implemented so that it can easily be inserted into a pipeline where an existing video source stream/video is present (such as Hudl, see section 2.3.9) and end-user using the produced metadata. This means that parameters such as video resolution and panorama image should be configurable and dynamic depending on what the user wants.

- **COTS components compatibility** - Sadji shall function efficiently using common of the shelf (COTS) components that are accessible for economically weaker teams. It should not be necessary to have to deploy the system using state of the art technology for it to function optimally.

- **Data ownership** - Produced metadata from Sadji shall be sent back to users without Sadji storing or using user provided stream or video any longer that needed.

3.3 System Model

This section will cover the system model, showcasing how Sadji fits into a larger, real-world system for translation and analysis of soccer players from a video source. Figure 3.1 illustrates the system model.

1. Stationary video source installment monitors soccer field. Video source is either streamed directly into Sadji or is recorded and saved, and at a later stage uploaded to Sadji.

2. Sadji, highlighted in red in figure 3.1, takes stream or uploaded video as input and translates soccer players coordinates in that footage to their designated coordinates on a user-specified image of a soccer field.

3. Coordinates produced by Sadji are passed as input to third-party analysis software (such as Sárgut and Guorrat).

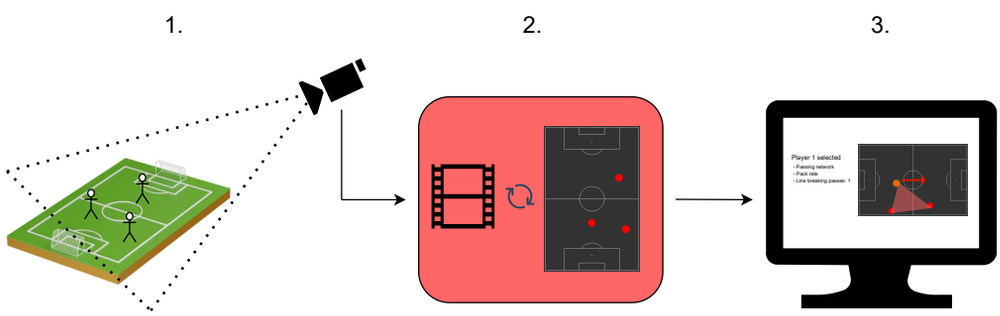


Figure 3.1: Sadji position in potential future pipeline.

4

Design and Implementation

In this chapter we will detail design choices made for the Sadji's architecture and how the different parts of the system was implemented. Many of the design choices were made after evaluating different iterations of the system, which will be detailed in chapter 5 on evaluation. This chapter will reference parts of evaluation chapter 5 to argue why certain choices were implemented. In section 4.1 we will highlight what prerequisites need to be in order for the system to operate as expected. In section 4.2 we will give a brief overview of how the system is pipelined before looking into each component of the system in section 4.3, 4.4, 4.5, and 4.6.

Sadji is structured as a pipeline consisting of 4 components. Each component has its own designated task to execute. Splitting a system into components was a conscious design choice because doing so offers several benefits for future work as well as continuous integration during the developing process. Splitting a system provides modularity, dividing the system into separate manageable parts which ensures easier development and understanding because of inherent less complexion. Components also promote re-usability, allowing them to be utilized into different projects in the future or act differently within the existing system, reducing redundancy and enhancing efficiency, improving testing and making debugging more efficient. The listed advantages collectively lead to a more flexible, robust, and adaptable systems which can be used in the future.

4.1 Prerequisites

Before initializing the pipeline, some prerequisites need to be made. For the system to function, the user will need to manually provide images to produce a panoramic image of the recorded area, a video, or stream source as input to the system, and select a detection model and tracker.

4.1.1 Video Source and Panoramic Image

A panoramic image of the recorded area is necessary to provide the system for the key-point homography generator component detailed in section 4.4 to function. It is important that the video source is mounted in a stationary position and can tilt and pan. A video source that moves side to side or backwards and forwards will not work with the system. Example of stationary video sources that would be suitable for the system:

- Hudl Focus cameras.
- TV broadcasting cameras that record field from the sides. See figure 4.1.
- Other video sources where the camera used is mounted in a stationary position.

While it is not required, a video source that can tilt and pan far enough in each direction to view the entire soccer field is beneficial for the inherent reason that players may be positioned at any point on the soccer field.

Figure 4.1: TV broadcasting camera angle from Romsaa Arena (previously known as Alfheim) in Tromsø.

The Hudl Focus camera works by stitching multiple cameras together and creating a virtual camera within the boundaries of the stitched camera sources. Inherently, this means that a finished panorama image does exist and this statement is also backed by the panoramic view angle available at Hudl's video viewing platform, see figure 4.2.

Figure 4.2: Hudl panoramic view angle from Romsaa Arena in Tromsø.

However, using for example a screenshot of this angle from the video viewing platform would not work due to the resolution of the image being to low relative to content contained (the soccer field). From the knowledge present during the writing of this thesis, an approach to download a full resolution panoramic image from the panoramic view angle does not exist. Same can be said for the TV broadcasting camera angle provided by TV2.

The vertical and horizontal distances from which the recording device is lo-

cated compared to the field plays an important role. In section 5.5 and 5.6 we demonstrate that the placement of the recording device affects both the performance and accuracy of the system due to two reasons. These are:

- Horizontal - placing the camera further away from the field will create a smaller panoramic image due to the principle of perspective. The farther away a camera is, the narrower the field of view it captures is. This condenses the scene, making objects appear smaller and reducing the overall area captured for each frame. This inherently results in a smaller panoramic image when stitching frames together. The size of the panoramic image affects performance because a large image has more pixels, resulting in more necessary computations.

- Vertical - placing a camera at a lower elevation when recording the field will result in more occlusions. This is because a lower perspective increases the risk of objects obstructing the view, obstructing the cameras line of sight to the action and thus missing out on visual information.

4.1.2 Creating Panoramic Image from Video Source

Because of these short-comings, the system provides functionality to create a panoramic image from any camera that follows the requirements mentioned earlier. The process of stitching images together to make a panorama image is shown in figure 4.3. A user needs to provide 5 images that roughly encapsulates the entire soccer field or area of interest. When 5 suitable images (suitable meaning keypoint detection method can match enough keypoints between neighbouring images) have been provided, the stitching process begins and works as following:

- Detect and match the features of the image. A feature can be explained as a unique property of some image such as textures, colors or shapes. Some examples of features that are found in the images encapsulating the soccer field are commercials, seating areas, stadium pillars and stadium equipment.

- Estimate homography matrix using the features that were found in previous step.

- Warp first image to align with second image.

- Blend the warped image together with second image.

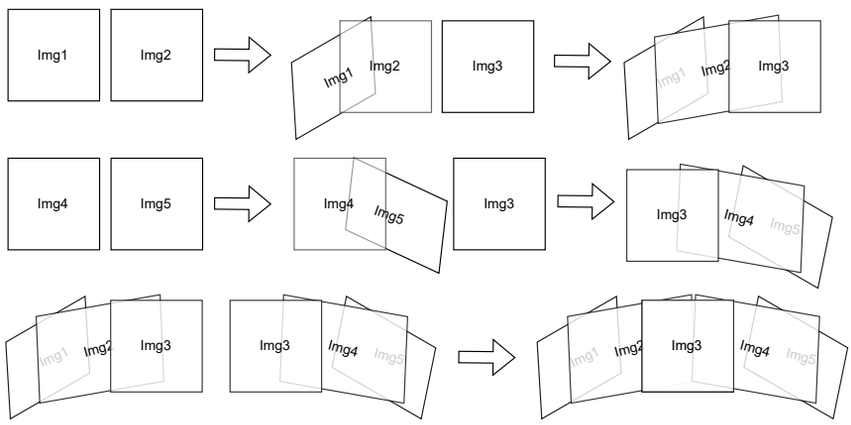


Figure 4.3: Panorama stitching process.

Following this process, a finished panoramic image will look like figure 4.4. Some post-processing of the panoramic image can also be made to improve system performance. Such changes are removing the bottom part of the image mostly made up of black empty area, down scaling the resolution, removing non-static features such as players, and animated commercials on digital screens. There are currently no way for the system to automatically blend players into the background and remove commercials. Different configurations of the panoramic image have been explored and tested, which will be shown in section 5.8.

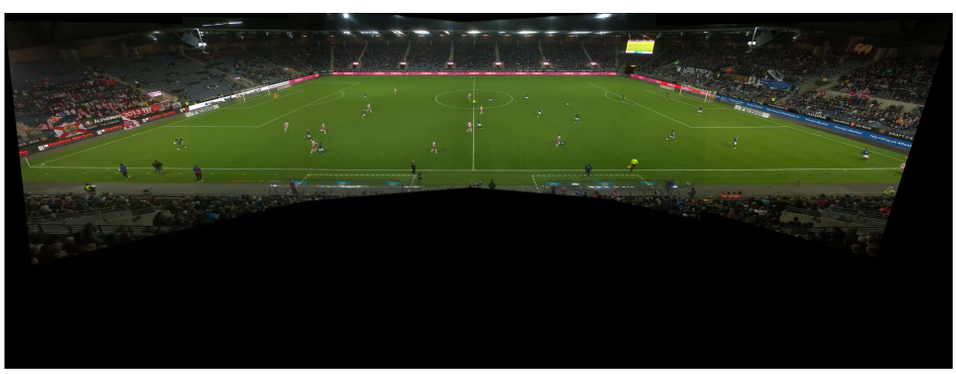


Figure 4.4: Panorama stitching process from SR-Bank Arena.

When a panoramic image has been created, the user will need to mark 12 keypoints in the panorama by clicking on the image. These 12 keypoints on the soccer field are chosen based on their strategic positioning at intersections along the field lines and corners, rendering them visually identifiable by a user. Furthermore, the spatial distribution of the keypoints ensure optimal coverage giving the homography matrix computation sufficient diversity of reference points. The location and order of the keypoints is shown in figure 4.5. These 12 keypoints are needed to compute the homography matrix between the panoramic image and the 2D image of a soccer field.

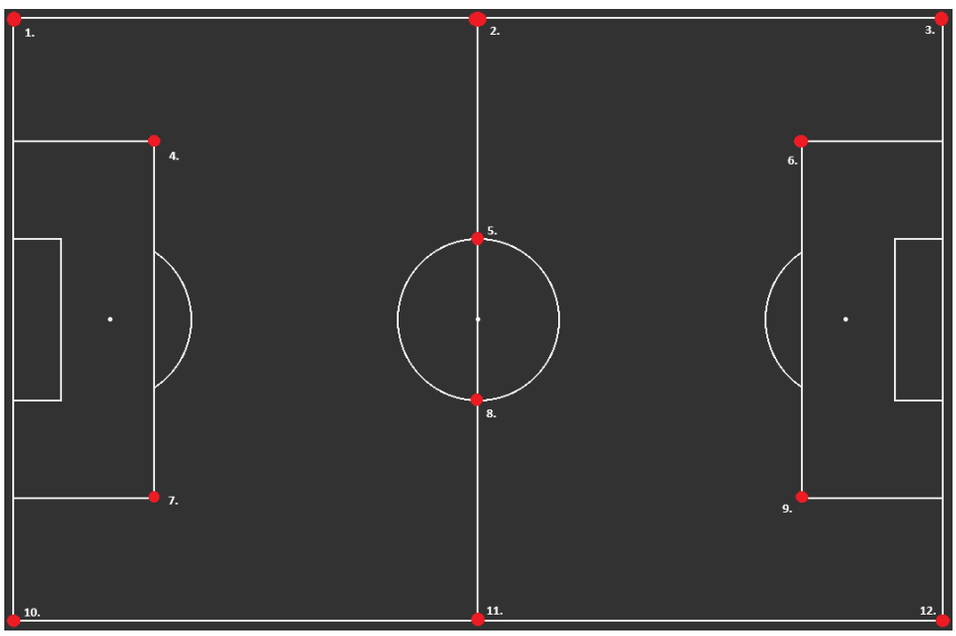


Figure 4.5: Soccer field keypoints between panorama and 2D image of soccer field.

4.1.3 Detection Model and Tracker

In order to generate inference data from the stream or video source, an object detection model is required which can identify players and/or classify what team/player type they are. For this purpose, we have trained a custom YOLOv8 model specifically to detect both (Tromsø Idrettslag (TIL) and Viking Fotballklubb (VIK)) players and goalkeepers from each team. While a general player detection model[1] has been utilized for a significant portion of the project's timeline, we opted to switch to a new custom-trained model to address classification and occlusion challenges. Additionally, the detection model needs to be complemented with a MOT model. Both ByteTrack[65] and BotSort[5]

1. URL to GitHub page: `https://github.com/noorkhokhar99/YOLOv8-football`

are recommended alternatives that seamlessly integrate with the Ultralytics YOLOv8 framework. The MOT component is necessary for calculating interpolation coordinates for players between frames, particularly in real-time scenario requirements.

Our player detection and classification model dataset is composed of 2,720 annotated frames separated into train and validation sets. The bounding box (BBOX) of all the players in each frame is retrieved by using an existing player detection model. Individual classification for every BBOX contained in the 2,720 frames is done manually by observing the produced BBOXs and assigning a classification for it. For our testing and evaluation of the system we have trained the model to recognize 4 different classes:

- TIL_P - a TIL player.
- TIL_K - TIL keeper.
- opponent_P - an opponent player.
- opponent_P - an opponent keeper.

The best weights for the model resulted in the following confusion matrix:

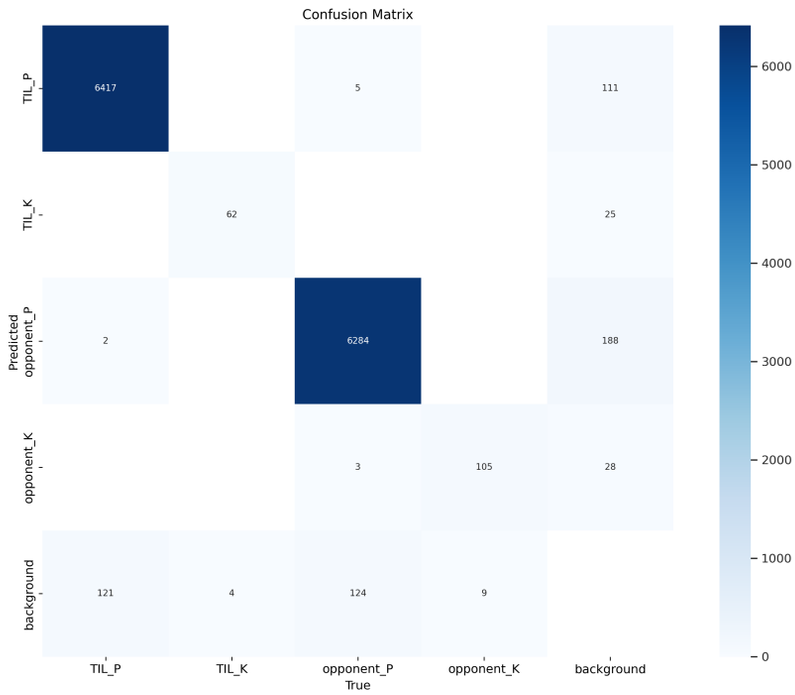


Figure 4.6: Confusion matrix for best weights.

In the first iteration of the systems design, the classification of players was not integrated into the detection model but operated as an isolated component using a custom trained YOLOv8 classification model. This however proved in-efficient and did not meet our requirement of real-time execution as player classification had to be carried out after player detection had been performed on a video. As it turned out, training a new detection model also proved beneficial in terms of occlusions happening with the trackers. A proposed system of using color concentration to find out what team a player was on also proved inaccurate and was discarded as a viable alternative early on in the design process.

The idea is that a model can be trained to recognise its own team players (in our case TIL) and all other variations of teams. We realize that such a process is tedious and faced with challenges due to teams often altering their soccer kits in between seasons. Work related to this issue has already been started at the CSG and will be discussed in section 7.3.1. For this thesis, training a model for the specific match (TIL-VIK) obtained via our collaboration with professional soccer team TIL worked well and proved that such an alternative is viable.

4.2 Pipeline Overview

Translating player coordinates from the video source to their respective positions on a 2D soccer field is a 4 step process. The pipeline consists of 4 components:

- The Detection and Tracking Component.
- The keypoint-based homography matrix generator (KBHMG) Component.
- The Coordinate Translation Component.
- The Metadata Aggregator Component.

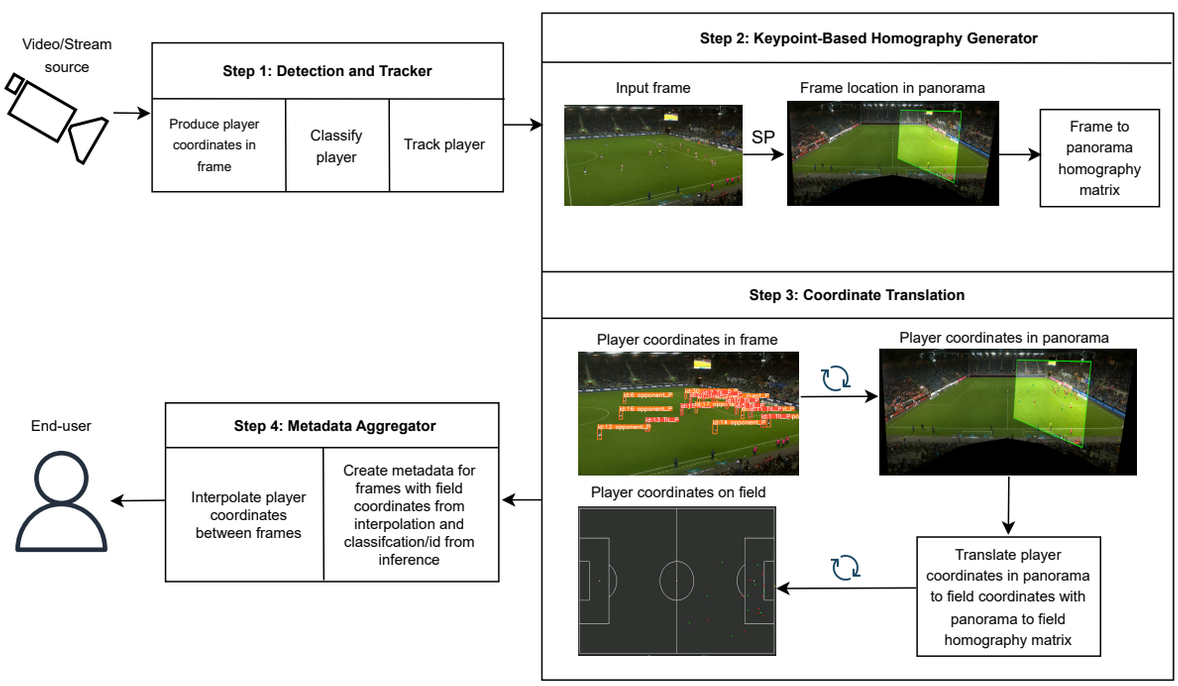


Figure 4.7: Pipeline overview.

Figure 4.7 illustrates the flow of the component pipeline. The detection and tracking component is run on its node, while the KBHMG, coordinate trans-

lation, and metadata aggregator components are run on another node. This is to isolate the components that are closely coupled and, in the future, be able to run the tracking component and the other components on their own machines. This is to increase performance which will be detailed in section 5 and 6. Communication between the nodes is facilitated through Javascript Object Notation (JSON). JSON is easy to read while maintaining a lightweight data size making it a good fit for the system. Performance is important in our system so JSON is a natural fit because of its fast data interchange [66]. It also provides great compatibility with different technologies promoting interoperability between our components by providing a common format. Section 4.3, 4.4, 4.5, and 4.6 will give a detailed description of design, implementation, and functionality of each of the components.

4.3 Detection and Tracker Component

Step 1 of the process begins in the detection and tracker component. A video source, either a stream or a default video is passed as an argument to the component. If video source is a stream, the manifest file of the stream is read to retrieve the Frames Per Second (fps) and stream segment size. This is to correctly segment produced metadata correctly in the metadata aggregator component (section 4.6). Coordinate translation interval and what resolution the panoramic image and frame in step 2 should be decreased to are also configured in the first component. This design choice was made so that the initial component of the pipeline is responsible for configuring the global variables used throughout the pipeline. Designing the first component this way also allows the system to be configured by an external user to fit the users requirements in terms of speed and precision from the system. In evaluation chapter 5, we showcase different efficiencies and accuracy when tweaking with said global variables.

Produced panorama from the camera source, for example, panorama showed in figure 4.4 is also configured at this component. The corresponding keypoints showed in figure 4.5 in the panorama also need to be passed to this component. Again, the design choice to have one component send all necessary prerequisites to other components is done because alternating the system so that an external user provides all necessary prerequisites is easier to integrate.

The detection and tracker component operates within a Flask[19] server, accessible for setup and inference through designated endpoints that the user can communicate with. The component also has an endpoint capable of receiving a YOLOv8 detection model. If no detection model is uploaded, it defaults to a player detection model (no player team classification). The detection and

tracker component also communicates with the next component through designated endpoints. Running the components within their own isolated Flask servers is done so that in the future, workload can be split over multiple computers/nodes to increase performance. This will be discussed further in chapter 6.

Figure 4.8 shows in detail how the detection and tracker component works.

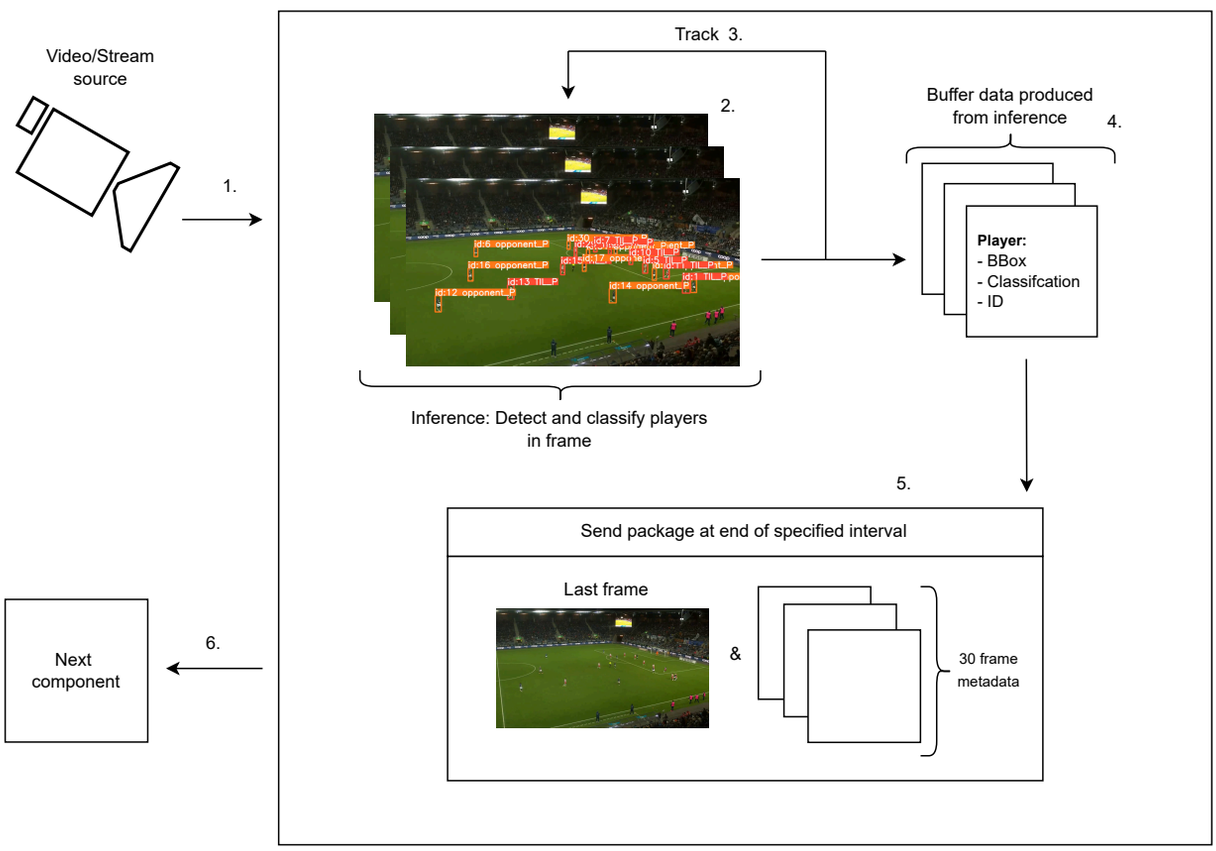


Figure 4.8: Detection and tracker component.

1. Video source is passed to *cv2.VideoCapture* object.

2. Frames from *cv2.VideoCapture* object are passed to object detection model for inference.

3. Integrated MOT (ByteTrack) keeps track of object ids between frames.

4. Data produced by inference (BBOXs, classification and id) is buffered for each frame (see listing 4.1 for example).

5. For every interval (30 frame interval as example), buffered frames metadata are packed into a message together with the last inference frame of the interval. Last inference frame is base64 encoded and then UTF-8 encoded before being appended to the buffer. This is so that the frame can be JSON serialized in next step.

6. Message (see listing 4.1) is then *JSON* serialized before being sent to KBHMG component. Buffer is cleared.

Listing 4.1: Message structure from component 1 to component 2.

```
[
    {
        "frame": 0,
        "data": list(zip(detection_boxes,
                         detection_ids,
                         detection_class))
    },
    {
    ...
    },
    {
        "frame": 29,
        "data": list(zip(detection_boxes,
                         detection_ids,
                         detection_class))
    },
    encoded_frame,
]
```

Inference and package sending are continuous until the video or stream ends. Upon inference completion, the detection and tracking component sends a notification to the KBHMG component.

4.4 Keypoint-Based Homography Matrix Generator Component

Step 2 of the pipeline happens at the KBHMG component. The homography between the panoramic image and the 2D field (*H1*), as well as the homography between the last frame of an interval and the panoramic image are computed.

The component initializes when incoming configuration is received from the first step of the pipeline as mentioned in section 4.3. Figure 4.9 illustrates how the component functions. The initializing works as follows:

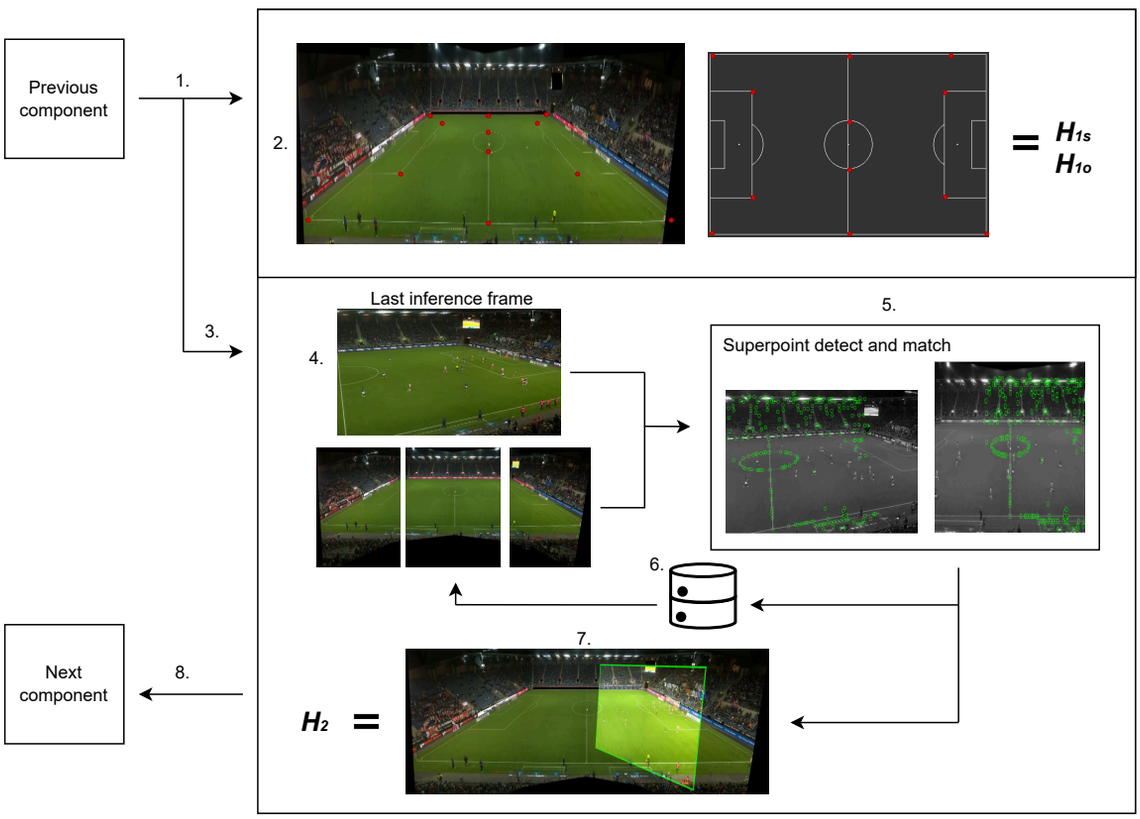


Figure 4.9: KBHMG component.

1. Receives configuration parameters from first component:

    - Panoramic image and corresponding keypoint coordinates (highlighted in red in figure 4.9).
    - Field keypoints from 2D field image (highlighted in red in figure 4.9).
    - Percentage decrease, which is defined as what percentage the frame and panorama should be decreased to.
    - Model names (what classifications the model is using).

- Segment variable, true or false depending on how the finished *JSON* files should be formatted (see section 4.6).

- Segment_duration, depending on how long each segment is if the video source is a stream (for example *m3u8*)

- Frames Per Second (fps) of the video source or stream source.

2. Uses panoramic image keypoints and field keypoints from 2D field image to compute homography between them. Two homography matrices are computed:

   - Homography matrix between decreased version of panorama image corresponding to received percentage decrease configuration from step 1 and keypoints in 2D field image (*H1s*).

   - Homography matrix between normal version of panorama image and keypoints in 2D field image (*H10*).

   Computing two homography matrices is done in case of the event that no matching keypoints between a frame and a panorama are found. If this does happen, the second homography matrix is instead utilised because not enough keypoints were detected in the decreased panorama and frame and original panorama and frame had to be utilized. This seldom happens but is implemented to ensure reliability. Computing the homographies is done by sending corresponding keypoints as arguments to the *cv2.findHomography* method. Both the decreased panoramic image and the original panoramic image are grayscaled and split into 3 equal parts (left, center and right). Splitting the image is done for two reasons; superpoint has a maximum image size and splitting the panoramic image enables caching. Converting panoramic image to grayscale is to reduce computational complexity and is a standard pre-procedure for keypoint detection algorithms.

After initialization, the component is ready to receive incoming frames from the detection and tracker component. Each frame is used to compute a homography between the frame and the panoramic image. The process is the following:

3. Last frame from an inference interval performed by the detection and tracker component is received and de-serialized from *JSON* and decoded from UTF-8 and base64.

4. Frame is decreased to configured percentage.

5. Decreased frame converted to grayscale and cached part of the split panoramic image are passed as arguments to superpoint module. The cached part is always the center when system initializes. Common keypoints between the frame and panoramic image. Listing 4.2 shows pseudo code for how matches are used to compute the homography matrix. Listing 4.2 also shows how the part of the panorama with most matches is cached for the next frame comparison. Figure 4.9 shows some of the matches highlighted in green between the frame and center part of the panorama.

6. Which part of the panorama had the most matches is then cached for next superpoint comparison.

7. Common points are then passed to *cv2.findHomography* method computing *H2*. Picture 7. in 4.9 shows how the homography can be used to warp the frame to fit into the panorama image. For our system however, the homography matrix is used to translate individual player coordinates and not warping images.

8. *H2* is passed to next component in the pipeline.

Listing 4.2: Pseudo code for keypoint detection from multiple parts of panorama.

```
while True:
    part = parts[cached_part]
    query_kpts_p, ref_kpts_p, matches = match(frame_gray, part)

    # Append matches with correct offset
    for match in matches:
        match.queryIdx = offset_query + match.queryIdx
        match.trainIdx = offset_train + match.trainIdx
        matches.append(match)
    for query_kpt in query_kpts_p:
        query_kpts.append(query_kpt)
    for ref_kpt in ref_kpts_p:
        ref_kpts.append(ref_kpt)
    offset_query += len(query_kpts)
    offset_train += len(ref_kpts)

    # Cache the number of matches for the current part
    cache[cached_part] = len(matches)

    rounds += 1

    # If enough matches exit loop
    if len(matches) > min_matches and rounds >= 1:
        break

    # Increase granularity if not enough matches, try again
    elif rounds >= 2 and len(matches) < min_matches:
        increase_granularity = True
        return increase_granularity

    # Check next part of panorama
    else:
        cached_part = increment_w_wraparound(cached_part, (len(parts)-1))

cached_part = most_matches(cache)

pts_frame = [query_kpts[m.queryIdx].pt for m in matches]
pts_panorama = [ref_kpts[m.trainIdx].pt for m in matches]
```

4.5 Coordinate Translation Component

The coordinate translation component is responsible for translating player coordinates in a frame to their position on a 2D field image. The homographies necessary to translate player coordinates are received from the KBHMG component (4.4). The player coordinates are from the same frame that was used to compute homography *H2* (the last frame of an inference interval). Figure 4.10 illustrates the components functionality. The steps are the following:

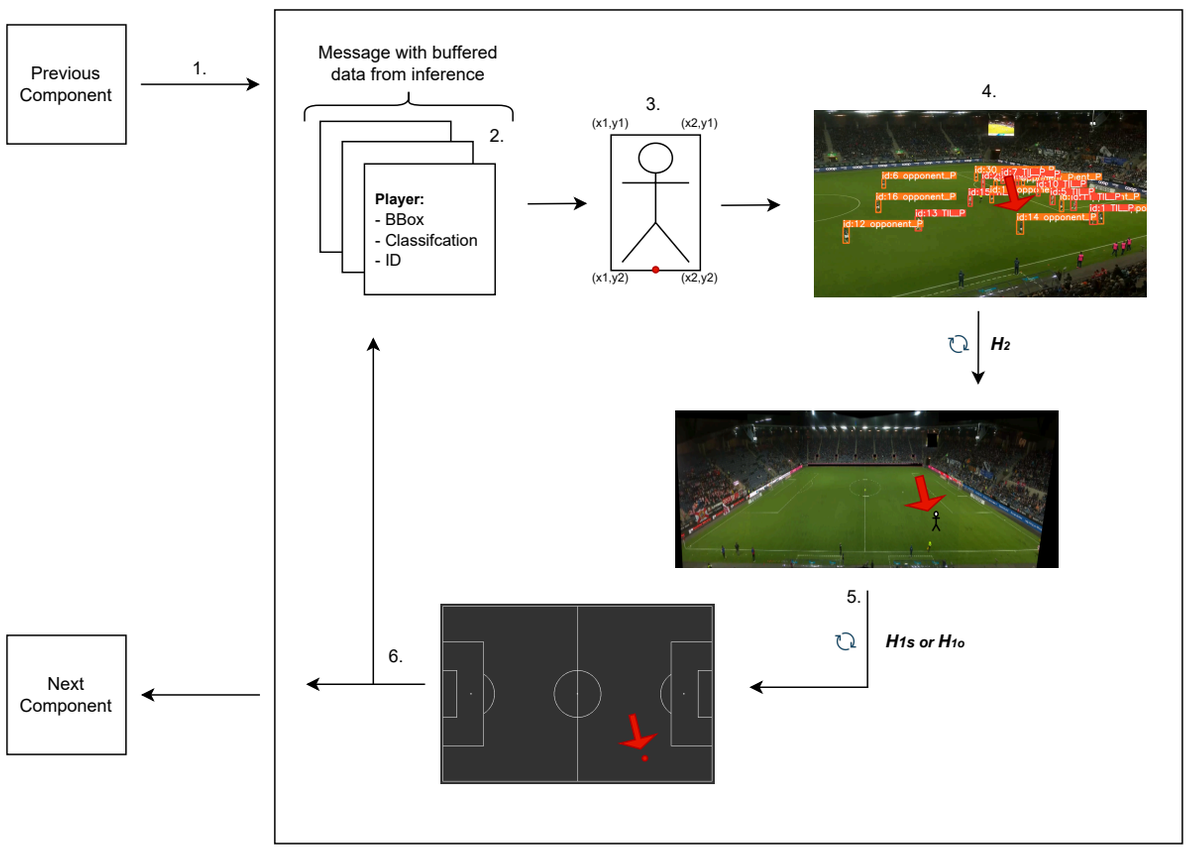


Figure 4.10: Coordinate translation component.

1. Component receives homography matrix *H2*. The coordinate translation component also receives homography matrix *H1s* and *H10*, however only once since these homographies remain unchanged.

2. The metadata from the last frame entry in the message sent from detection and tracker component (4.3) is extracted.

3. A single coordinate is computed from a players bounding box (BBOX) highlighted in red in figure 4.10. This coordinate was chosen as the most suitable because the center bottom coordinate corresponds to the point of contact between the player and the soccer field, best representing the position of a player.

4. The center bottom coordinate of a player is then scaled according to what percentage decrease is specified in detection and tracker component (4.3) before being translated to coordinate in panorama using homography

matrix *H2*. Highlighted player in figure 4.10 shows player coordinate translation.

5. Once a player coordinate is translated to the panorama, the player coordinate in the panorama is again translated to the 2D soccer field using homography matrix *H1s* (or *H10*).

6. This process continues for every player that is part of the metadata of the last frame entry in the message. Once all player coordinates have been translated, they are passed on to the metadata aggregator component together with previous translated player coordinates.

To compute interpolation coordinates for a player in the interval, two coordinates are needed. Once a player coordinate has been translated, that coordinate is saved for next player coordinate translation for that particular player. An edge case occurs for first interval as no previous translated player coordinates exist. To deal with this, the first interval consists of only one frame, which is the first frame of the inference. Every continuous frame after the first follows the pipeline as illustrated in figure 4.10.

4.6 Metadata Aggregator Component

The metadata aggregator is the last component of the pipeline and is responsible for, as the name suggests, aggregating produced metadata. The component has two configurations for aggregating metadata depending on user requirements:

- Stream segments - if the user wants to use the produced metadata for a stream, which is generally the case for when a user wants to run the system in real-time during for example a match or training session. Each segment (*JSON* file) will contain frame metadata for an interval of seconds (2 seconds is common segment size for streams).

- Packed segments - if the user does not need the produced metadata for a real-time use case, which could be post-game analysis or half-time analysis. The produced metadata will be aggregated to one large *JSON* file containing frame metadata for all frames.

This structure is illustrated in figure 4.11.

Figure 4.12 shows in detail how the metadata aggregator component functions:

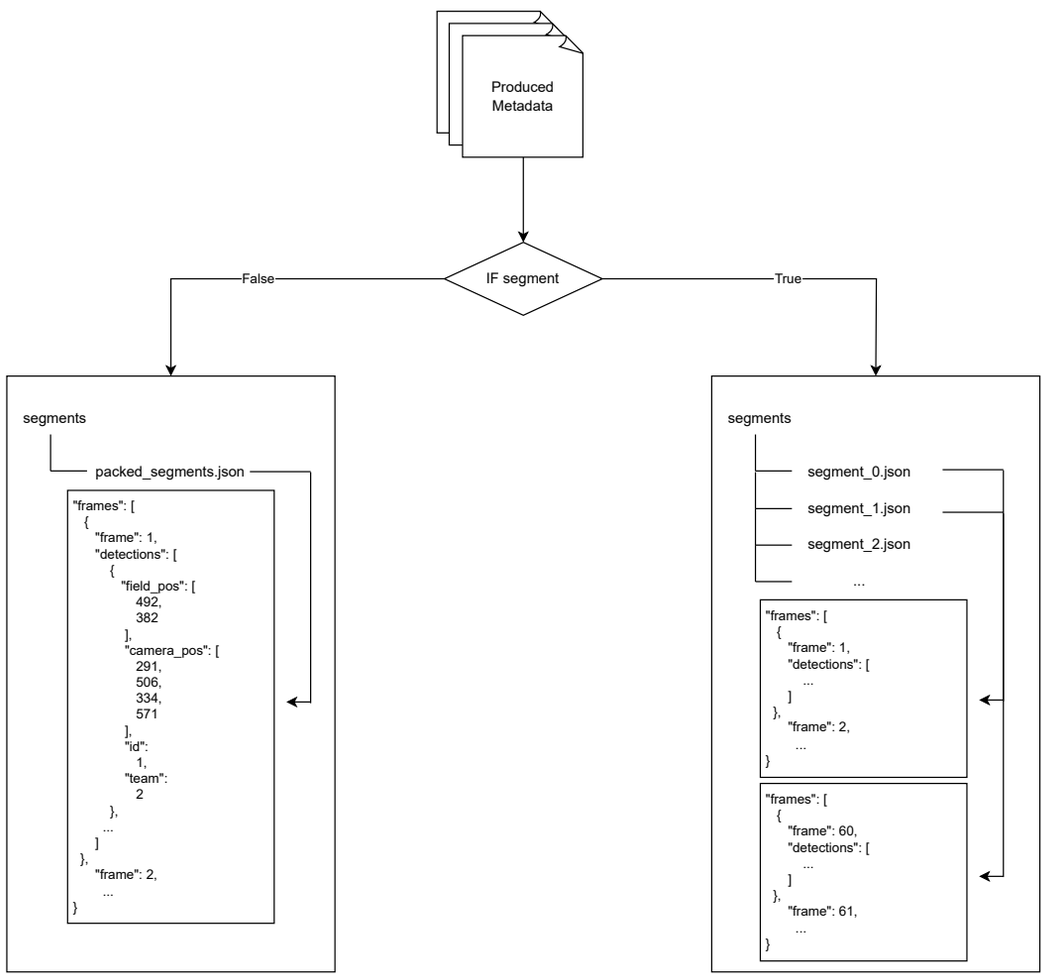


Figure 4.11: Aggregated metadata for segment an non-segment case.

1. Component receives player coordinates on field from previous round of coordinate translation and current round. If there are no previous translated player coordinates, meaning that the pipeline has only processed the one frame from the stream or video input, this component is ignored.

2. Component also receives the message containing the buffered data from the inference.

3. Interpolation coordinates for each player is computed in the interval between Point 1 and Point n + 1 highlighted in red and blue. If a player does not have both a previous translated coordinate (Point 1) and a current translated coordinate (Point n + 1) then interpolation computation does not occur for that player.

4. Each interpolation coordinate is then correctly put into the corresponding frame together with the original BBOX of the player in the video frame, its classification and ID.

5. Depending on if the original video source was a normal video or a stream, the produced metadata from previous step (4.) is either segmented (case for stream) or put into one large *JSON* package (case for normal video). Figure 4.11 illustrates the structure between segmented produced metadata and one large package. One large package or segmented packages are then sent using a POST request back to the user who initialized Sadji. The POST request is made to metadata Uniform Resource Locator (URL) handle at the analysis software Sárgut.

4.7 Cleaning Tool

The cleaning tool is a semi-automatic component that positioned outside of the pipeline showcased in figure 4.7. Designed as a proof of concept (POC), its purpose is to assist a person in cleaning the produced metadata from the metadata aggregator component. Cleaning produced metadata can be summarized into the following:

- Connect IDs that belong to the same player. MOTs assign new IDs to objects that it cannot recognize from previous frames, making the same object have multiple IDs throughout the duration of the video.

- Resolve occlusions between players. MOTs sometime assign an existing ID to another object, because it finds the objects similar. This can particularly occur at players far away from the camera where the amount of pixel

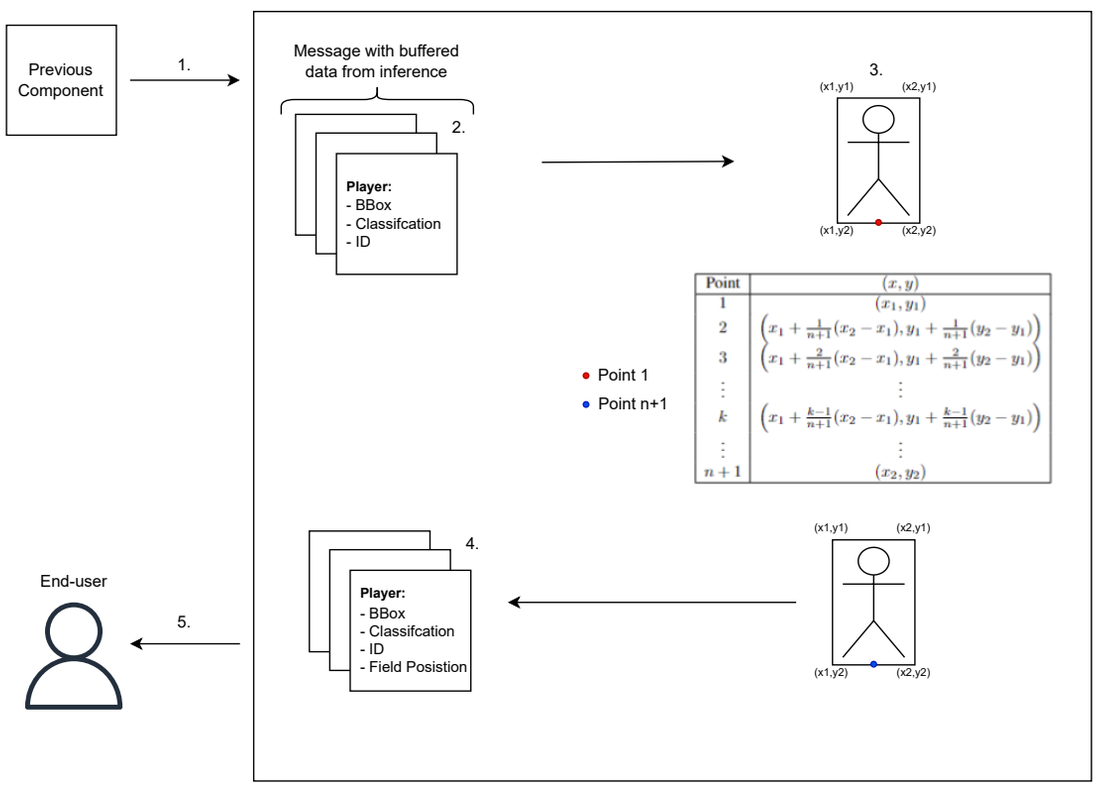


Figure 4.12: Metadata aggregator illustration.

detail is lacking. Occlusions also occur when players run in front of each other.

The cleaning tool works by using spatial locality, classification, and time since created for IDs to connect them.

4.7.1 Connecting Missing and New IDs

Figure 4.13 shows how the occlusion tool works and the following is a step by step:

1. Produced metadata is loaded into a dictionary that keeps track of all IDs that exist for each frame and begins iteration from first frame. For each frame, player detections is checked to see if any new IDs have appeared.

2. If an ID is not presented in a frame, but was present in the last, then a counter is incremented to count number of frames it has been missing. If the ID re-appears then the counter is reset. If an new ID is present, then that ID is marked as present.

3. If an ID is marked as missing and has been for more than 30 frames, then the cleaning tool initializes the connecting interface. The last coordinate the ID was located at is used to find nearby new IDs.

4. The connecting interface works by locating all nearby IDs to the ID that disappeared that have the same classification. In the example connecting interface in figure 4.13 ID 15 has disappeared while ID 54 has appeared, with the same classification. User is then showed all other potential matches in the area and selects which new ID belongs to the player that had the disappeared ID.

5. When a user connects the two IDs by (by pressing 1 in example case) every ID 54 is replaced with ID 15 in the dictionary containing the produced metadata in step 1. This process continues until the last frame. On completion, the dictionary is saved with the same *JSON* format as described in section 4.6.

4.7.2 Fixing Occluded IDs

Occluded IDs occur when players run in front of each other, obstructing the line of sight between camera and player. The MOTs is confused and moves bounding boxes between players. There are roughly three types of occlusions

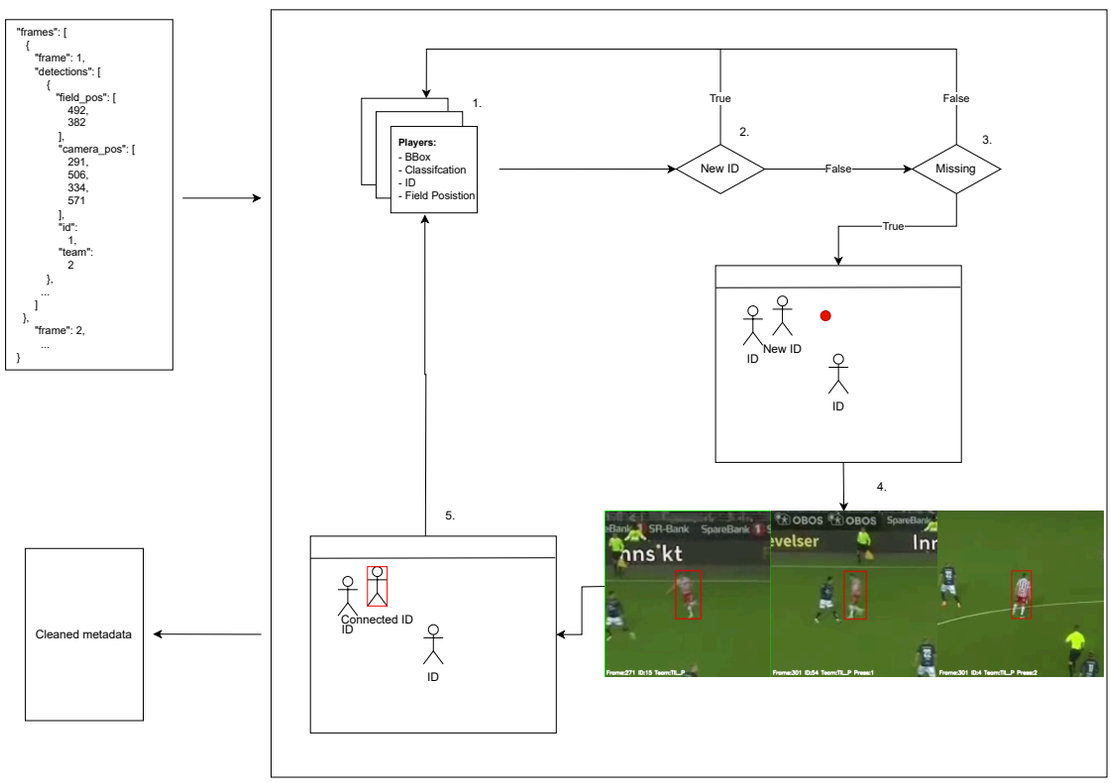


Figure 4.13: Cleaning component - connecting IDs.

that can happen between two players:

- Clean swap between IDs - two player bounding boxes swap clean between them, no new IDs are created.

- One new ID - 1 player receives the bounding box and ID from another, while the other player gets a new ID.

Detecting occlusions works by checking for classification change for a detection. If a bounding box moves from one player to another, the classification changes if the player is on the opposite team. Occlusion between players in different teams is most cases of occlusion events, however occlusion does occur within a team. This will be discussed in 6. The fixing of occluded IDs is illustrated in 4.14 and works as follows:

1. Produced metadata is loaded into dictionary that keeps track of all IDs that exist for each frame and begins iteration from first frame (same as in section 4.7.1).

2. If a detection does not have the same detection as in the previous frame, then a counter starts incrementing number of frames that the detection classification does not match the original classification. If it does match at a later frame, the counter is reset.

3. If the counter reaches 60 consecutive frames where classification is miss-matched, an occlusion event is triggered. 60 consecutive frames is chosen as it gives sufficient information of a players trajectory without compromising efficiency.

4. User is then presented with an interface showing the crops of all players that are nearby the ID that is occluded. The crops, referred to as the timeline is illustrated in figure 4.15. It display all crops of nearby players and the occluded ID forward and backward in time to help the user identify what IDs are involved in the occlusion events. In figure 4.15 we can see that ID 12 looses its bounding box at around frame 655. At frame 657, new ID 81 is created on the player that was originally ID 12. A user would then enter ID 12 as occluded ID and that the occlusion happened on frame 655. The involved ID, being 81, is then entered and the frame on which the new ID 81 was created. All metadata associated with ID 81 is moved to ID 12 from frame 655, and all metadata associated with ID 12 after 655 is moved to new ID 81 after frame 657.

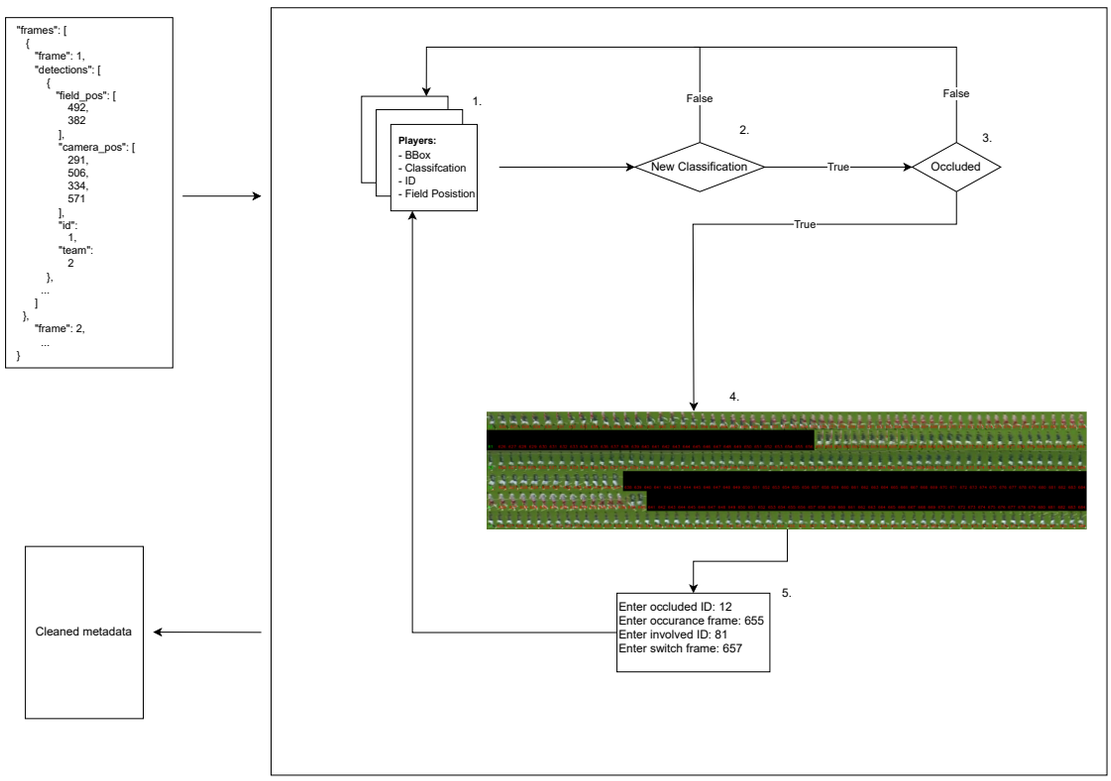


Figure 4.14: Cleaning component - fixing occluded IDs.

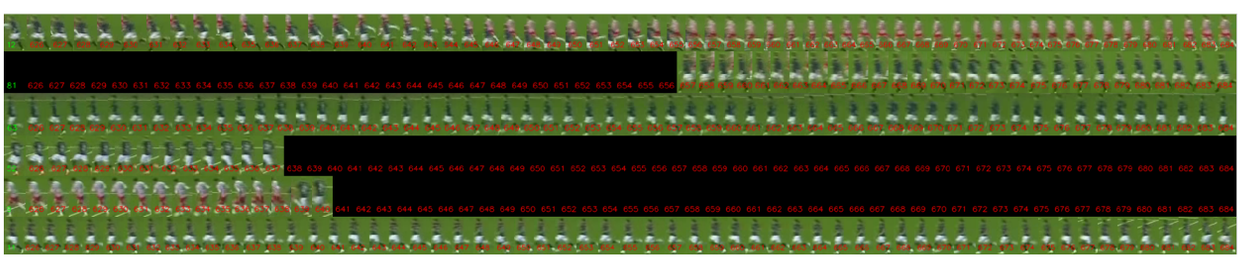


Figure 4.15: Timelines for players that could be involved in occlusion events.

4.8 Summary

In this section we have covered the design and implementation of Sadji. We have detailed how the coordinate translation pipeline operates and what prerequisites need to be handled on initialize. First, the detection and tracker components finds all players for each frame in the video source. In frame coordinates are then sent from the detection and tracker component and received at the KBHMG component. The KBHMG component then calculates the necessary homography matrices needed to translate player coordinates in each frame, to their respective coordinate on a user provided image of a soccer field. The homographies produced in the KBHMG component are then passed to the translation component, which in turn handles the player coordinates translation. Lastly, the metadata aggregator component receives translated coordinates from the translation component and interpolates player coordinates for the specified interval (user specified, 30 frames interval in example). Metadata aggregator component finishes by posting produced metadata back to the original requesting user. In section 4.7 we highlighted the cleaning tool and how it functions as a isolated component outside of the Sadji pipeline.

In the next section (5) we will evaluate Sadji and its components.

5

Evaluation

This chapter will give an in depth evaluation of the system and investigate the test-cases referenced in chapter 4 on design and implementation. The design paradigm 1.3.3 from section 1.3 states that it is expected that an engineer continuously re-iterates the steps presented in the paradigm. This fosters adaptability and improvements in the implementation process. By revisiting and refining the steps outlined in the paradigm, one can uncover and address potential unexpected flaws, optimize performance and ultimately enhance the quality of the work that has been done.

5.1 Experiment Hardware

All experiments have been performed with the following hardware:

- **Central Processing Unit (CPU):** 13th Gen Intel(R) Core(TM) i7-13700 2.10 GHz.
- **Graphics Processing Unit (GPU):** NVIDIA GeForce RTX 3070 8GB.
- **Random Access Memory (RAM)** 64 GB.
- **Storage:** NVMe 3400 NVMe SED Micron 1024GB SSD (Solid State Drive).

- **Operating System:** Windows 11.
- **Linux Subsystem:** Windows Subsystem for Linux (WSL) with Ubuntu.

The used GPU is consumer-graded with high performance capabilities making it a good fit for graphically demanding task such as machine learning applications.

5.2 Choosing Player Detection Model

Choosing a player detection model for the system is based on three factors as specified in requirements chapter; accuracy, efficiency and deployment on conventional hardware. The model should be able to at any given time detect all/as many as possible players within a frame from a video source and determine what team that player belongs too. At the time of writing this thesis YOLOv8 object detection models were some of the faster and more accurate alternatives. Maintaining a low level of total IDs over the duration of a video is also important for interpolation to function optimally as detailed in section 4.6.

5.2.1 Experiment

For this experiment we have run a general YOLOv8 soccer player detection model and compared it to our custom trained YOLOv8 soccer player detection model which can also classify team of detected player. 3 different video segments running in 1080p with 30 fps have been tested accumulating to $\approx$ 5 minutes worth of footage. The videos are all of active soccer play (meaning no corner, penalty, or similar) and are retrieved through our collaboration with professional Norwegian soccer team TIL. The experiment is conducted using the hardware specifications listed in section 5.1.

5.2.2 Results

Table 5.1 shows the difference in total produced IDs when comparing a model that is trained to detect players on soccer field versus our detection model which is trained to detect the player and their team affiliation. Video 3 has significantly more produced IDs because of its duration. Video 1 improved by $\approx$**44.9%**, slightly better than video 2 and 3 due to more false detection of sideline players (players not participating in the game).

Video	Model: no-team	Model: team	Improvement
1	118	**65**	**44.9%**
2	137	**83**	**39.4%**
3	1109	**683**	**38.4%**

Table 5.1: Model with and without team classification.

5.2.3 Discussion

Our custom trained YOLOv8 detection model is well within the threshold for real-time inference, achieving an average inference time per frame ≈**26.07ms**. Using a detection model trained to not only detect the player, but what team they are on reduces numbers of IDs produced by ≈**40.9%**. This decrease is a result of mainly two factors. Training the model to recognize exclusively players on the soccer field can result in false detection such as side-line players (players that are not participating in the game), referees or trainers. These false detection happen much less when the model is trained to recognize team affiliation. Secondly, occlusions between players (see section 4.7.2) is reduced because the MOT can more easily distinguish between bounding boxes.

5.3 Choosing MOT

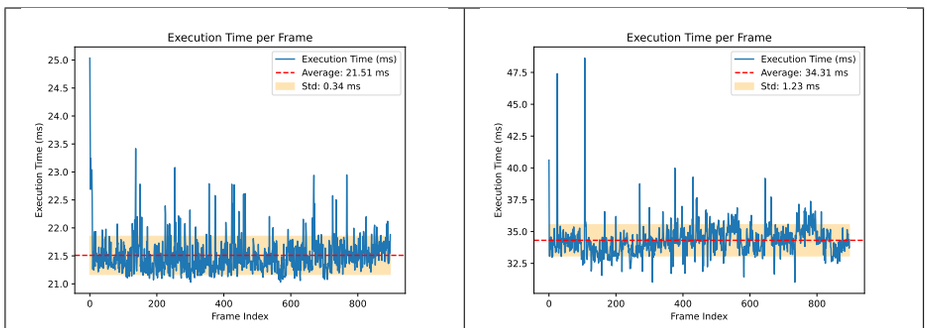
As mentioned in section 4.1.3, the system needs an MOT that can consistently track player over several frames. It's crucial that the MOT also functions in real-time and can be integrated with chosen player detection model. The Ultralytics framework offers support for two MOTs that can easily be integrated with YOLOv8 models. These are BotSort and ByteTrack (see section 2.1.3, 2.1.3).

5.3.1 Experiment

For this experiment case, we have run a YOLOv8 model with ByteTrack and BotSort using the specified hardware in section 5.1. We are testing for execution time per frame for both of the trackers. The video utilized is 1080p and 30 fps with a duration of 30 seconds of active soccer play.

5.3.2 Results

Table 5.2 compares ByteTrack and BotSort execution time per frame. Real-time requirements for a 30 fps video is ≈**33.33ms**. ByteTrack achieves average execution time per frame with **21.51ms** with BotSort achieving **34.31ms**.



(a) ByteTrack execution time per frame. **(b)** BotSort execution time per frame.

Table 5.2: ByteTrack and BotSort comparison.

5.3.3 Discussion

Both MOT's achieve similar results in terms of accuracy and persistence in regards to maintaining the same id for a player over several frames. This is observed by viewing total number of IDs over the course of the video which for ByteTrack is **64** and **57** for BotSort. However, inference speed differs with ByteTrack edging out with an inference time of **21.51ms** over BotSort's **34.31ms**. Both MOT's have been tested while running the entire coordinate production pipeline detailed in chapter 4. ByteTrack has been, based on these findings, chosen as our MOT of choice for this system.

5.4 Choosing Keypoint Detection and Feature Matching Algorithm

Several keypoint detection and feature matching algorithms have been tested and are listed in section 2.3.4. The SuperPoint algorithm showcased in section 5.4.10, 5.4.11, and 5.4.12 on accuracy and throughput is the first iteration of our implementation where no optimizations have been made. Optimizations made is further detailed in section 5.8. For our system we need an algorithm that works efficiently and accurately. The next section will go into detail on the

different keypoint detection and feature matching algorithms that have been tested, and their achieved accuracy and throughput.

Three methods have been used to determine the accuracy. These are:

- *Pixel spread* - If correct keypoints are found then resulting homography matrix will correctly translate coordinates to their respective areas in the panorama and then consequently to the 2D soccer field image. Over the course of a video sequence, each produced coordinate should be in relative close proximity to the previous translated coordinate. Our experiment parameter pixel spread measures the distance between coordinates in frames to measure accuracy. Pixel spread gives fast and accurate results from testing and is our main accuracy comparison measurement between different keypoint detection and feature matching algorithms.

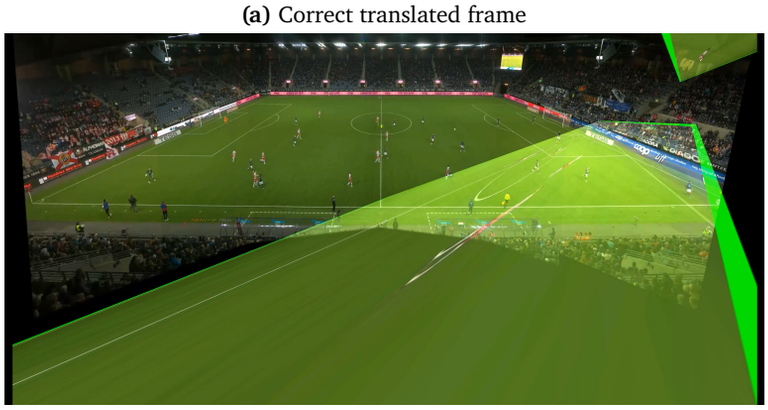
- Visually compare produced 2D soccer field coordinates with the GPS coordinates for player 1[1] at TIL.

- Visually examine translated frames over the course of the video. If keypoints are accurate then the resulting warped frame from the video sequence should perfectly align inside the panoramic image. This is shown in figure 5.1a. Figure 5.1b illustrates how a frame could inaccurately be translated because of incorrect keypoints.

It needs to be stated that pixel spread as a measure of accuracy does have some implications. Spread in between coordinates from one frame to the next can also naturally be higher due to players running at high speeds. However, a large deviation in pixel spread does indicate that a lack of accuracy is present which can be observed by plotting coordinates on the 2D soccer field over a given duration. From eliterserie team TIL we have obtained STATSports GPS coordinates for player 1 (number 4) which we have used to compare our coordinates from Sadji with. The coordinates pixel spread and visualization for GPS is shown in table 5.3.

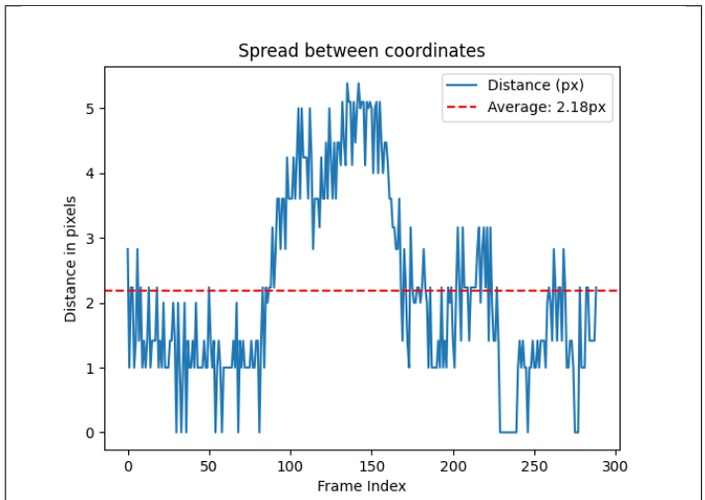
1. First player we are tracking at TIL

(a) Correct translated frame

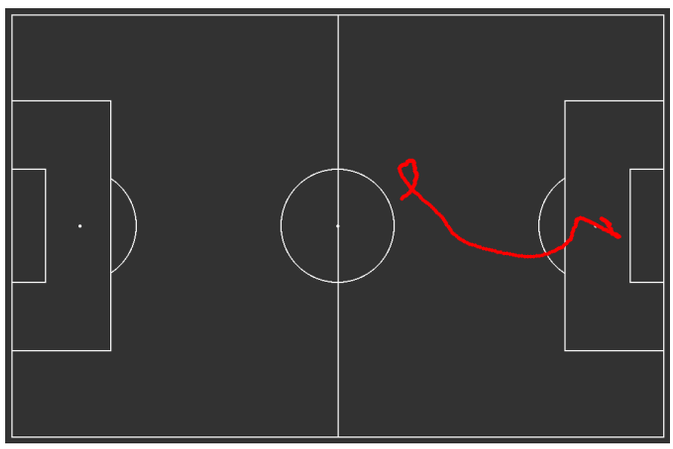


(b) Incorrect translated frame

Figure 5.1: Accurate vs inaccurate translation.



(a) GPS pixel spread



(b) GPS coordinate visualization

Table 5.3: GPS coordinate spread and visualization.

To measure throughput we have observed at execution time per frame for each of the different keypoint algorithms.

5.4.1 BRISK - Experiment

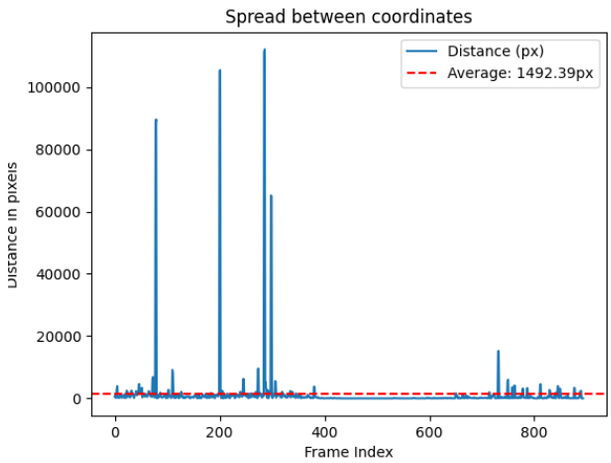
Experiment hardware is listed in section 5.1. Video segment 4[2] has been used and we are testing for pixel spread and throughput. Definition for pixel spread

2. Video clip from soccer match played between VIK-TIL the 22nd October 2023 at time interval 51:20-51:30 is referenced as video segment 4. Video is recorded using Hudl.

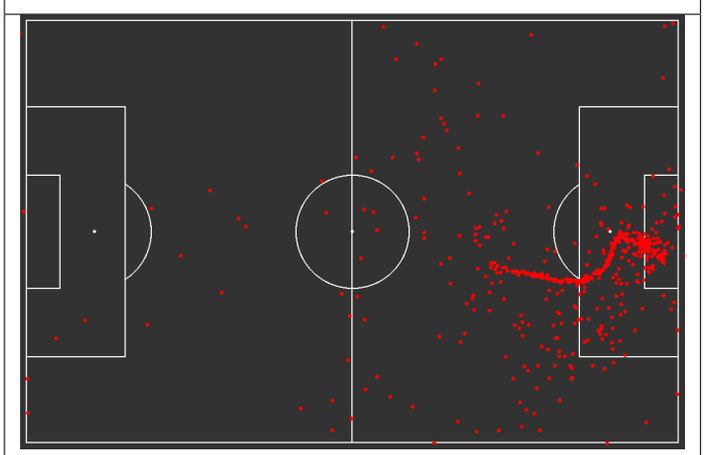
and throughput are stated in section 5.4. Player 1 is being tracked.

5.4.2 BRISK - Results

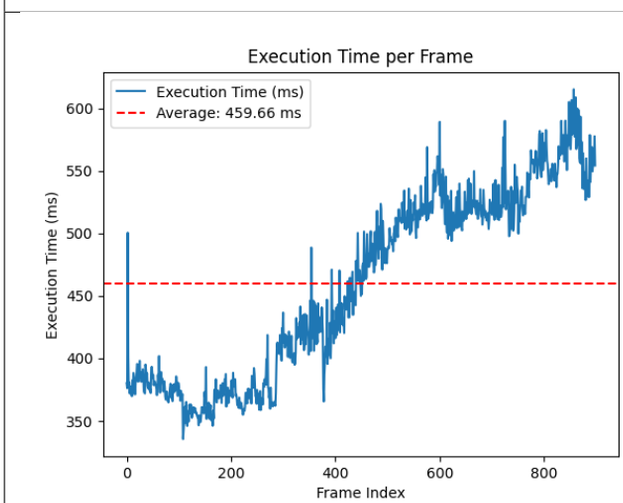
Entry a) in table 5.4 presents achieved pixel spread averaging **1,492.39 pixels**. BRISK is achieves relatively high precision between frame 300 and 700. This statement can be visualized in entry b) in table 5.4 where we can see that successive player coordinates are in close proximity to one another. BRISK throughput averages **459.66ms** per frame as shown in entry c) in table 5.4.



(a) BRISK pixel spread.



(b) BRISK coordinate visualization.



(c) BRISK throughput.

Table 5.4: Experiment results for BRISK.

5.4.3 BRISK - Discussion

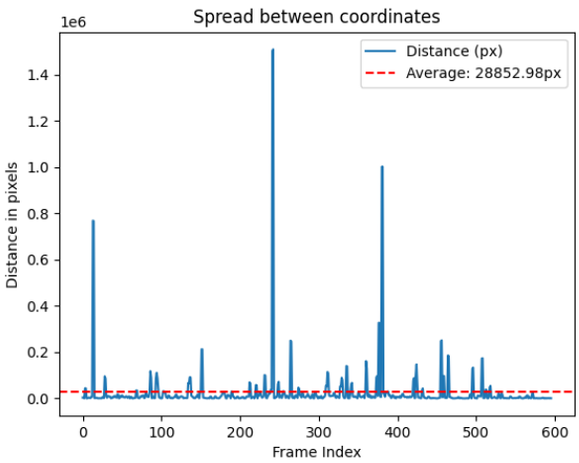
Pixel spread for BRISK, when compared with GPS, is substantially lower resulting in a less accurate representation of where a player is located. When camera moves to the right side of the field, BRISK pixel spread improves and averages a pixel spread more comparable to that of GPS. However, because of its lack of consistency especially when the player is located towards the middle of the field, BRISK is not a viable option. BRISK throughput averages to **459.66ms** per frame, meaning that for a 30 fps video, we could translate every 15th frame from the video in real-time.

5.4.4 ORB - Experiment

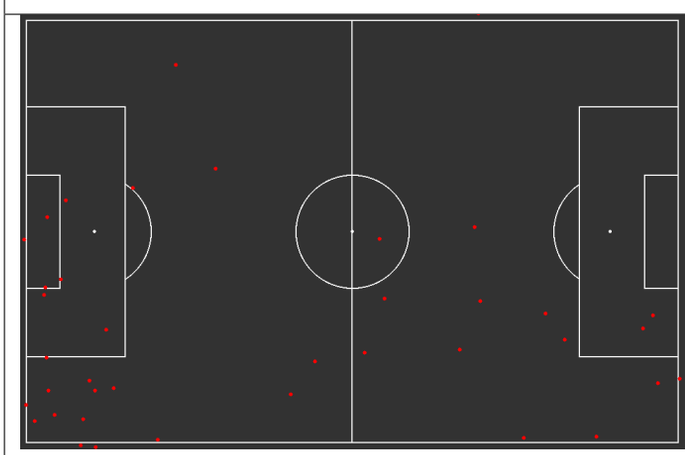
For this test case we are investigating ORB, measuring throughput and pixel spread (see section 5.4) using video segment 4 (see footnote 2). Player 1 is being tracked.

5.4.5 ORB - Results

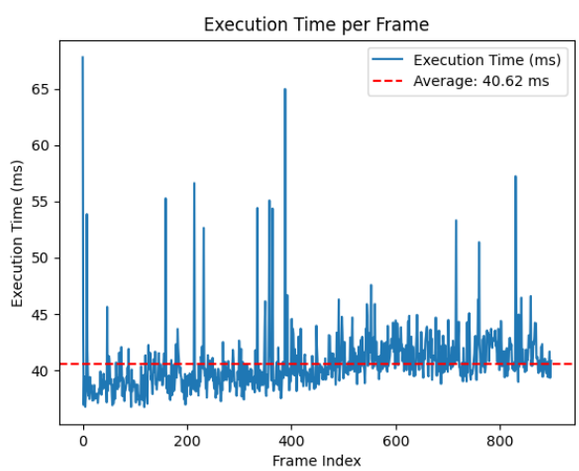
Table 5.5 shows experiment results. Entry a) displays ORB pixel spread averaging **28,852.98 pixels**. Pixel spread fluctuates throughout the video duration, achieving no consist result. Entry b) shows visualises the pixel spread on the soccer field image where no trajectory of player 1 is visible, only scattered coordinates. ORB throughput is fast, averaging **40.62ms** per frame as seen in entry c) in table 5.5.



(a) ORB pixel spread.



(b) ORB coordinate visualization.



(c) ORB throughput.

Table 5.5: Experiment results for ORB.

5.4.6 ORB - Discussion

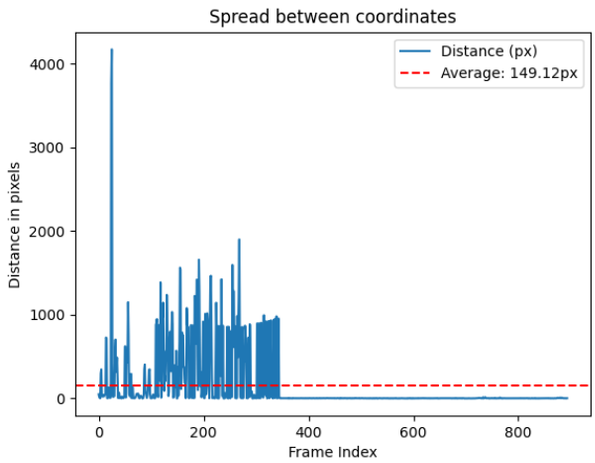
ORB achieves best throughput of the keypoint detection and matching algorithms tested, achieving throughput close to real-time (≈**33.33ms** per frame). However, ORB achieves an average pixel spread higher than the others with coordinates scattered across the soccer field image. Unlike BRISK, no improvement in pixel spread is obtained when the camera records the right part of the soccer field (where BRISK improved). Because of this pixel spread, ORB is not a viable alternative for our use-case.

5.4.7 SIFT - Experiment

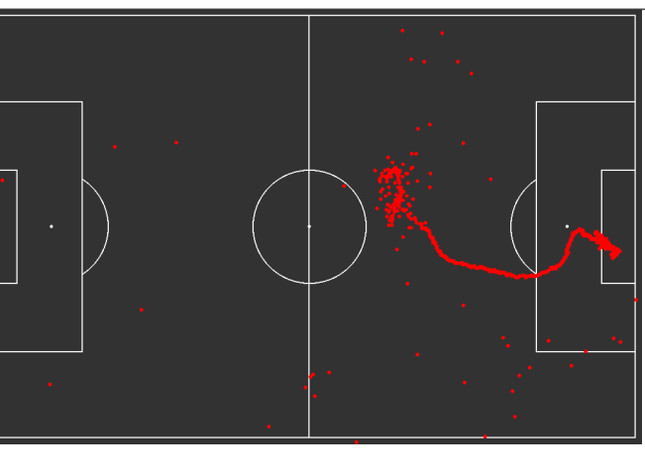
For this experiment we are investigating keypoint detection and feature matching algorithm SIFT. Video segment 4 (see footnote 2) is utilized, and we are tracking player 1. We are testing for pixel spread and throughput (see section 5.4 for definition). Experiment hardware is listed in section 5.1.

5.4.8 SIFT - Results

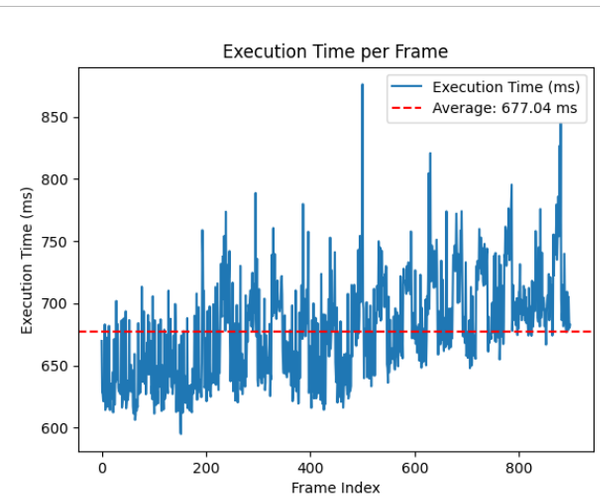
SIFT achieves average pixel spread of **149.12 pixels** as shown in entry a) in table 5.6. Pixel spread after approximately 350 frames remains low. This can also be observed in entry b) in table 5.6 where trajectory of player 1 remains consistent. Throughput averages at **677.04ms** as seen in entry c).



(a) SIFT pixel spread.



(b) SIFT coordinate visualization.



(c) SIFT throughput.

Table 5.6: Experiment results for SIFT.

5.4.9 SIFT - Discussion

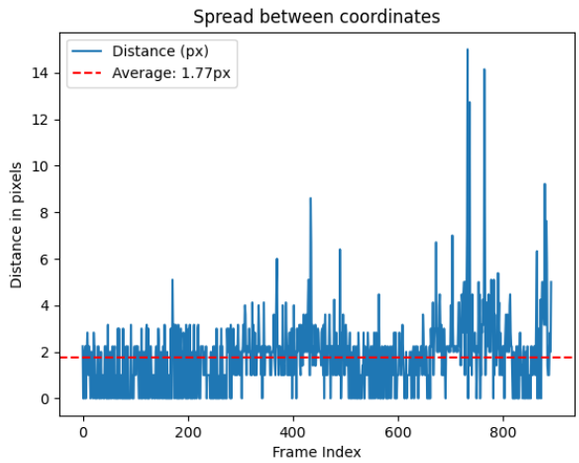
SIFT pixel spreads works well once the camera records the right part of the soccer field, similar to BRISK. This seems to be a common trait, possibly being a result of more unique features in each frame being detected once the camera moves to the right side. Pixel spread towards the center part of the soccer field remains high, resulting in an inconsistent trajectory for player 1. Throughput is relatively low, averaging at **677.04ms** which means approximately 1 frame can be translated every 20th frame. SIFT is not a viable option because of its inconsistencies in pixel spread towards the center area of the soccer field and because of its high execution time per frame.

5.4.10 SuperPoint - Experiment

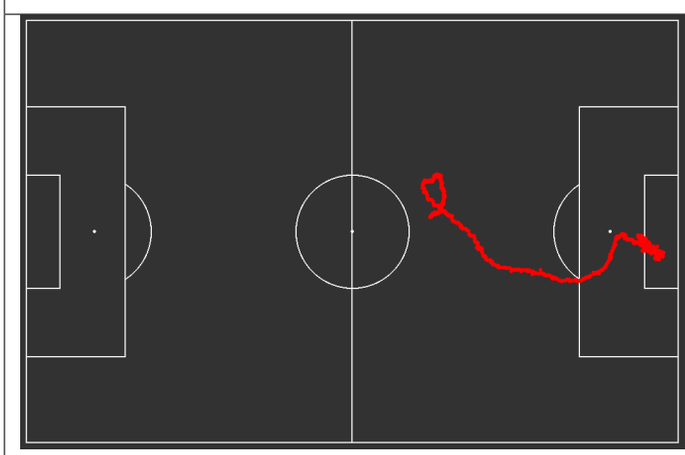
For this experiment, we are investigating SuperPoint (detailed in section 2.3.5), using the specified hardware in section 5.1. GPU computation has not been enabled, and all computation is performed using the CPU. Video segment 4 (see footnote 2) has been utilized and we are tracking player 1. We are investigating pixel spread and throughput (see section 5.4 for definition).

5.4.11 SuperPoint - Results

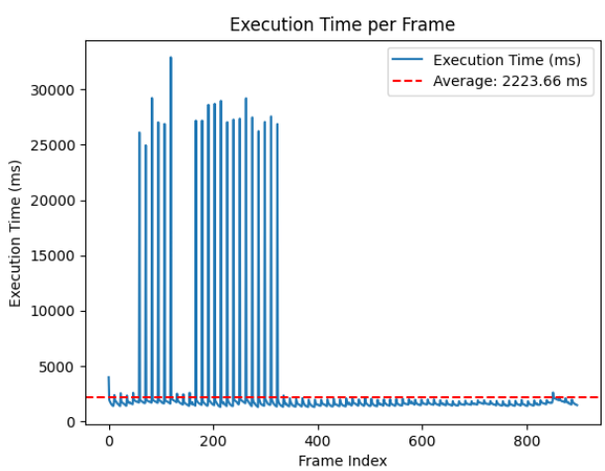
SuperPoint achieves average pixel spread of **1.77 pixels**. Pixel spread increases around frame 750 to a maximum of approximately 14 pixels as shown in entry a) in table 5.7. Entry b) displays the visualization of player 1 trajectory of the course of the video. Proximity of all drawn coordinates are high, resulting in a clear player trajectory. Throughput, as seen in entry c) in table 5.7, averages to **2,223.66ms** per frame, where execution time per frame between approximately frame 100 to 350 is higher reaching close to 35,000ms for some of the frames.



(a) SuperPoint pixel spread.



(b) SuperPoint coordinate visualization.



(c) SuperPoint throughput.

Table 5.7: Experiment results for SuperPoint.

5.4.12 SuperPoint - Discussion

SuperPoint achieves highest accuracy of investigated keypoint detection and matching algorithms, achieving a pixel spread comparable to that of the GPS pixel spread shown in table 5.3. Because of this result, SuperPoint is a viable alternative, providing an accurate representation of where a player is located on the soccer field. Throughput remains high averaging **2,223.66ms**. The first $\approx$ 350 frames have greater execution time. This is a result due to not enough keypoints being detected for each part of the panorama. As mentioned in section 4.4, because of a limitation of image size input for SuperPoint, the panorama is split into three parts. For our first iteration of the KBHMG (section 4.4), no information is cached and the comparison between frame and panorama part is in order center part, left part and right part. Optimization results of the SuperPoint implementation will be further explored in section 5.8.

5.4.13 Summary

In this section we have investigated different keypoint detection and feature matching algorithms using pixel spread and throughput as main determinants for viable options. The coordinate visualization and pixel spread produced by the GPS coordinates serve as our benchmark because of the low pixel spread and clear trajectory of the player. However, GPS also showed some inaccuracy which is highlighted with red arrows in figure 5.2 where the player trajectory deviates from what we observe in the footage. Table 5.8 summarize the findings from the choosing keypoint detection and feature matching algorithm section.

Method	Throughput (ms)	Spread (px)
BRISK	459.66	1492.39
ORB	40.62	28852.98
SIFT	677.94	149.12
GPS	N/A	2.18
SP	2223.66	1.77

Table 5.8: Throughput and pixel spread.

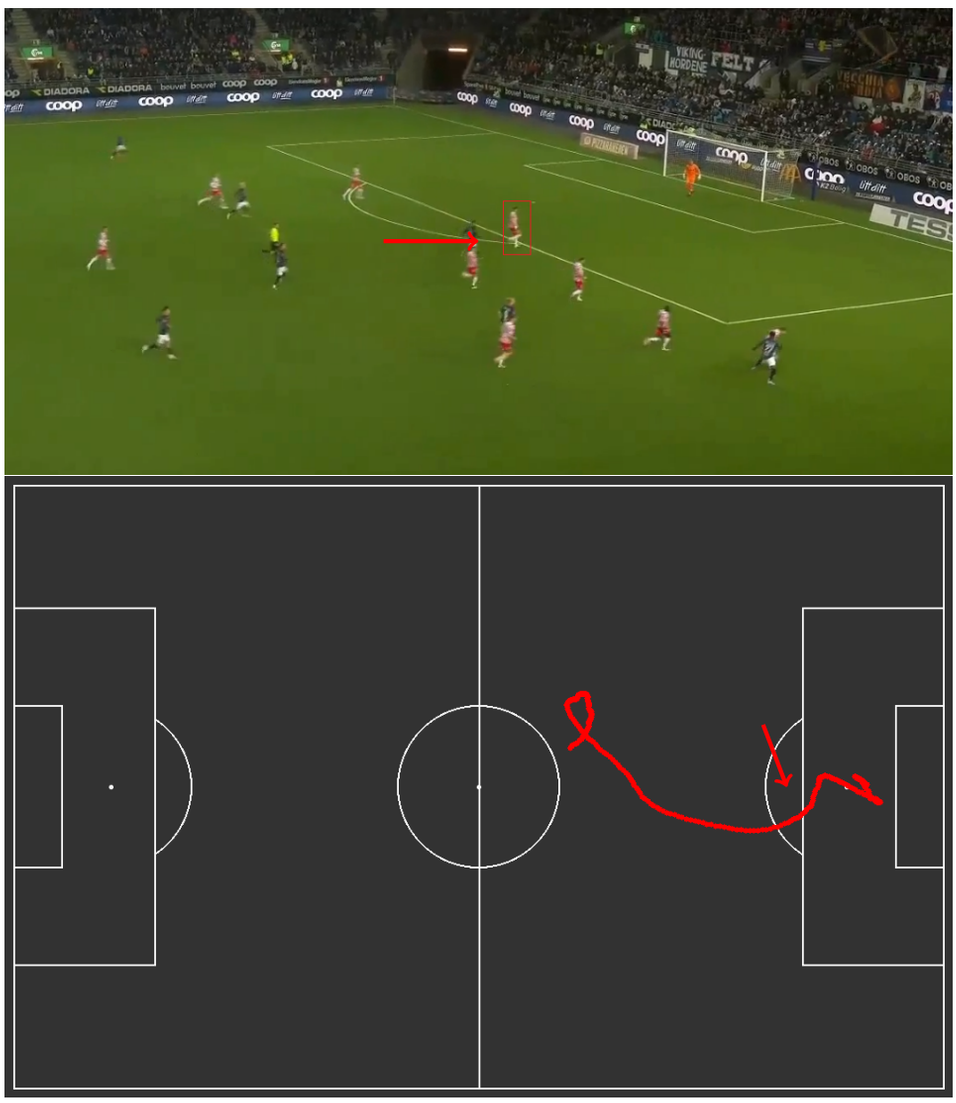


Figure 5.2: Coordinate trajectory deviation.

5.5 Video Source Placement on Tracking Quality

As detailed in section 4.1.1, video source placement is vital for producing fewer occlusions between players and minimizing the panoramic image size. This is because a low camera installment results in players more often running in front of each other, obscuring the line of sight between detected player and camera. Placing a camera to close to the soccer field inherently means that the camera needs to pan further left and right to capture the entirety of the

soccer field, resulting in a larger panoramic image. A larger panoramic image (larger resolution) will result in higher computation times for the KBHMG component, because there are more pixels that need to be checked when detecting keypoints between frame and panorama.

5.5.1 Experiment

For this experiment, we want to investigate how camera placement affects number of produces IDs over the duration of 5 minutes of soccer play. We will be investigating the Hudl camera placement at SR-Bank Arena (VIK home arena), Brann Stadion (Norwegian eliteserie team Brann home arena), and Romsaa Arena (TIL home stadium). A high number of produced IDs means that the tracker is having problems associating bounding boxes for players (see section 2.1.3 on ByteTrack). The experiment hardware is listed in section 5.1 and we are investigating 5 minutes of 1080p 30 fps worth of footage for each of the arenas. The same YOLOv8 player detection model is used for all arenas.

5.5.2 Result

Figure 5.3 shows experiment results. SR-Bank Arena and Bran Stadion achieve similar results, achieving respectively **1,089** and **1,176** unique IDs. Camera installment at Romsaa Arena results in **3,806** unique IDs. The sudden increase of IDs at approximately 120 seconds for Romsaa Arena is caused by an abrupt camera movement and consequently several newly produced IDs that are unable to be associated by the tracker. Figure 5.4 shows produced panorama and resolution from the different camera placements at the arenas.

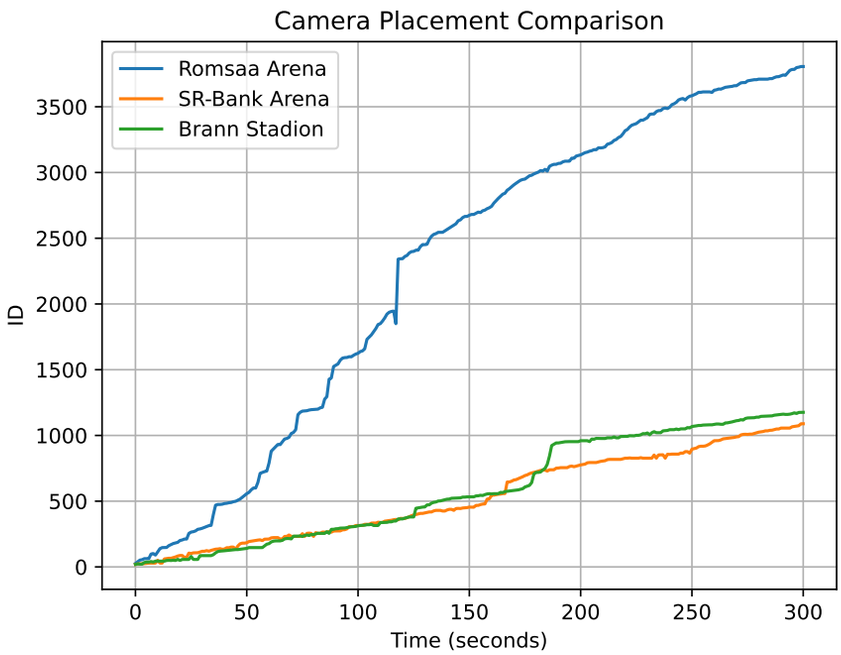


Figure 5.3: Total detected IDs with different camera placements.

(a) Romsaa Arena Panorama (9238x1372).

(b) SR-Bank Arena (4957x915).

(c) Bran Stadion (5254x1044).

Figure 5.4: Panorama image produced from camera angle at testing arenas.

5.5.3 Discussion

Both SR-Bank Arena and Brann Stadion achieve similar results, producing approximately 2,700 fewer IDs than Romsaa Arena. Camera placement is similar for both of these arenas, which is illustrated by showcasing panoramic images produced from the arenas. The soccer field appears larger, with the furthermost part of the field appearing closer to the camera. Both of these arenas have cameras placed further back and higher up. Panorama image produced by camera at Romsaa Arena is significantly larger at 9,238x1,372. Because of this lower and closer to the soccer field camera placement, the tracker is experiencing difficulties associating bounding boxes and keeping IDs persistent. Furthermore, the large panorama will increase overall Sadji performance because there are more pixels that need to be part of the computation done by the KBHMG component (see section 4.4). This deficiency in performance, due to larger produced panorama as a result of poor camera placement, is not covered in this experiment. However it serves as a visual illustration for total detected IDs with different camera placements.

5.6 Video Source Resolution for Inference Quality and Speed

It is important that the system can detect as many players on field as possible at any given time. Preferably, as long as all players are in the camera frame, a total of 22 players should at any time be detected. For the player detection model, resolution plays an important role for distinguishing all players. Resolution determines the level of detail that is captured, and a higher resolution enables finer details allowing the model to recognize important features.

5.6.1 Experiment

For this experiment, we will investigate how video source resolution affects the number of players the YOLOv8 model can detect for each second over the course of 5 minutes. We are experimenting with three different resolutions: 480p, 720p, and 1080p. We will also test for inference time per frame at the different resolutions. These resolutions are chosen because they were the once available through the Hudl video source. Experiment hardware is listed in section 5.1. Video is 30 fps for all resolutions and the same YOLOv8 player detection model is utilized.

5.6.2 Result

Figure 5.5 details the results from the experiment. 1080p and 720p maintain a similar number of detected IDs, with 1080p slightly detecting more. 480p detects consistently less, with a least number 6 IDs detected at approximately 100 seconds. 720p and 1080p maintained a significantly higher number of IDs during the same timestamp.

Figure 5.5: Detected IDs per second with different video source resolutions.

These are the inference times per frame for the different resolutions:

- **480p** ≈ 5.86 milliseconds per frame.
- **720p** ≈ 9.00 milliseconds per frame.
- **1080p** ≈ 16.82 milliseconds per frame.

5.6.3 Discussion

Figure 5.5 illustrates that video source resolution plays an important part for the YOLOv8 detection models ability to maintain a high number of detected players. Both the 720p and 1080p resolutions are capable of detecting close to 22 IDs, with 1080p slightly outperforming. This difference in performance can be attributed to the higher resolution of 1080p, which provides more detailed frames for the model to analyze, resulting in improved ID detection accuracy.

Opting for a lower resolution video source does not yield performance benefits to offset the disadvantage of reduced player detection due to the lower resolution. YOLOv8 maintains real-time inference performance for 1080p resolution and lower with the experiment hardware we are utilizing. Inference time is important to minimize for real-time system performance, and higher resolution does inherently affect this.

Threshold for real time execution for a video recorded at 30 fps is ≈ 33.33 milliseconds per frame. The inference time for 1080p is comfortably within the requirements for real-time performances, making it a suitable choice for our system because it also aids the model in detecting more players. 1440p might serve as a viable alternative to 1080p opting for even more detailed images. However, 1440p was not an available option for us at the time of writing this thesis.

5.7 Detection and Tracker Component

Decreasing the interval at which the detection and tracker component (see section 4.3 for component details) dispatches packages to the KBHMG component (section 4.4) means that a more accurate representation of a players coordinates on the soccer field is produced. This is because fewer coordinates have to be interpolated in metadata aggregator component (see section 4.6). However, decreasing the interval means that the detection and tracker component has to halt execution every time a package is dispatched.

5.7.1 Experiment

In this section we will investigate the overall throughput of the detection and tracker component. We will use a 1080p video with 30 fps, meaning that for the component to achieve real-time, it must have a throughput higher than 30 fps. The experiment hardware is listed in section 5.1. A general YOLOv8

player detection model is utilized. No other components are run during this experiment.

5.7.2 Result

Table 5.9. Only interval 2 and 1 achieve throughput lower than minimum threshold for real-time averaging **28.64** and **22.79** fps.

Interval	Throughput (fps)	Real-time
30	37.17	YES
25	37.14	YES
20	36.92	YES
15	36.65	YES
10	36.06	YES
5	33.36	YES
2	28.64	NO
1	22.79	NO

Table 5.9: Inference speed per frame at different intervals.

5.7.3 Discussion

Increasing the interval at which the detection and tracker component dispatches packages to the KBHMG component results in higher throughput as shown in table 5.9. By lowering the interval at which packages are dispatched, more frames are sent to the KBHMG component, resulting in a lower total throughput. This is because more frames have to be encoded and dispatched resulting in the player detection model having to halt inference for a short duration. For a video recorded at 1080p with 30 fps we achieved real-time throughput for intervals 30-5 highlighted in green in table 5.9. In section 5.10 we will evaluate how Sadji performs when all components are running.

5.8 KBHMG Component

The KBHMG component has undergone the most number of iterations in design to gradually increase the throughput while maintaining a high level of accuracy, measured in pixel spread (see section 5.4 for pixel spread definition). The next subsections will cover the experiment, result and discussion for each of the added features to increase KBHMG throughput. All KBHMG component experiments have been conducted while running the KBHMG component isolated,

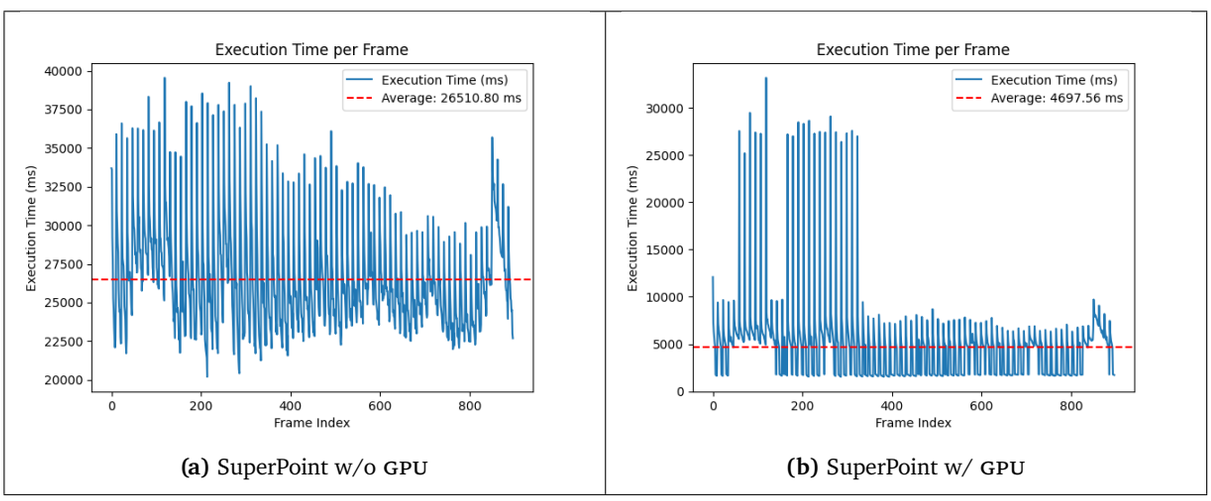
meaning no other components have been running at the same time.

5.8.1 GPU - Experiment

For this test we have moved the SuperPoint keypoint detection and feature matching algorithm from the CPU to the GPU. A 30 seconds video has been used with a resolution of 1080p and 30 fps (see footnote 2). The testing hardware is specified in section 5.1.

5.8.2 GPU - Result

Experiment results are listed in table 5.10. Moving execution from CPU to GPU improves execution time for all frames, from an average execution time per frame of **26,510.80ms** to **4,697.56ms** on the GPU. Execution time per frame decreases after approximately 350 frames. This is because the camera moves towards the right part of the soccer field, allowing SuperPoint to detect and match more keypoints thus reducing execution time.



(a) SuperPoint w/o GPU

(b) SuperPoint w/ GPU

Table 5.10: GPU vs CPU.

5.8.3 GPU - Discussion

Significant improvement is made when SuperPoint is run on the GPU. This is because the GPU enables enhanced parallel processing capabilities compared to the CPU, allowing for faster computation time of convolutions and matrix multiplications which are common operations for deep learning algorithms

Figure 5.6: Panorama with cropped bottom part.

such as SuperPoint.

5.8.4 Crop - Problem

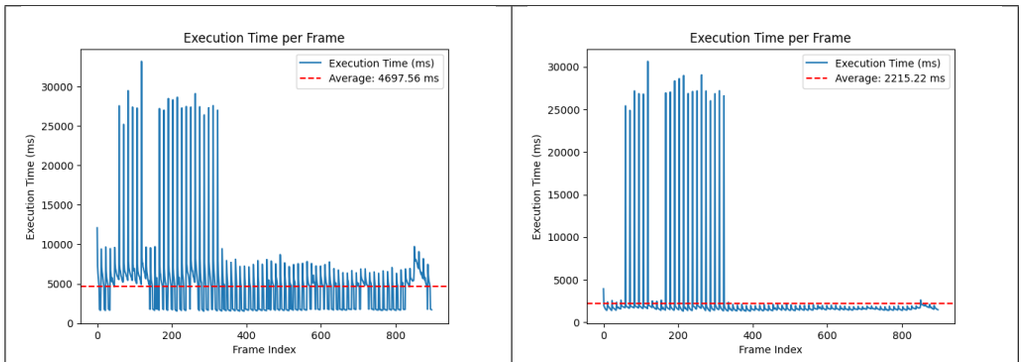
A significant portion of the created panorama (see section 4.1.2) is non-essential for SuperPoint to function correctly. When warping and stitching pictures to create the panorama image, the bottom part of the panorama is dark and stretched, containing no relevant information for the frame panorama comparison in the KBHMG component to function correctly. Figure 4.4 illustrates this. However, for other keypoint detection and feature matching algorithms (BRISK, ORB and SIFT) doing this did reduce number of keypoints found.

5.8.5 Crop - Experiment

For this experiment, we are cropping a large part of the bottom half of the panorama image (see figure 5.6). Sadji supports no automatic cropping of the produced panoramic image. However, a proposed method is detailed in section 6.1 in figure 6.1. Experiment hardware is specified in section 5.1, and we are utilizing a 30 seconds, 1080p 30 fps video (see footnote 2).

5.8.6 Crop - Result

Table 5.11 shows produced results after running the experiment. Using cropped panorama image reduced average execution time per frame to **2,2215.22ms** from **4,697.56ms** without cropped.



(a) SuperPoint w/o cropped panorama

(b) SuperPoint w/ cropped panorama

Table 5.11: With and without cropped panorama.

5.8.7 Crop - Discussion

Cropping the panoramic image improves average execution time per frame by approximately half. This is to be expected as number of pixels that need to be accounted for in the frame panorama comparison performed at the KBHMG component is significantly reduced. Using SuperPoint, cropping the panorama image had no effect for finding matches between the frame and the panorama (see section 4.4 for detailed description of this process).

5.8.8 Percentage - Problem

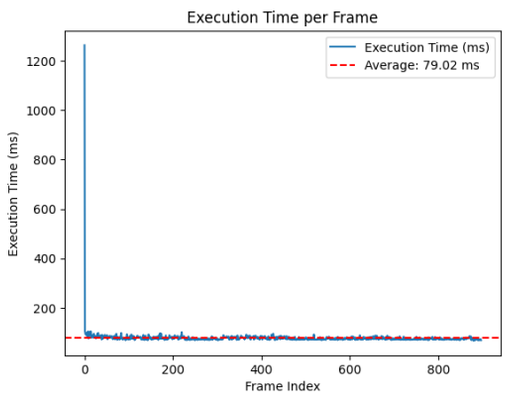
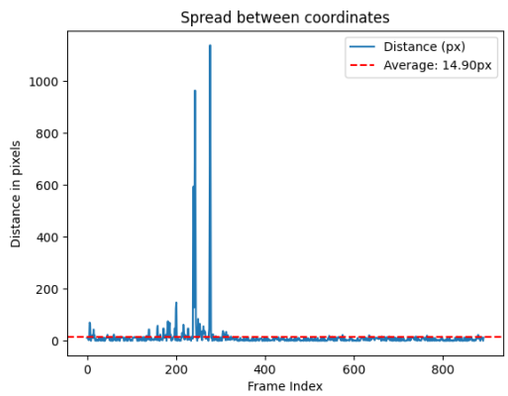
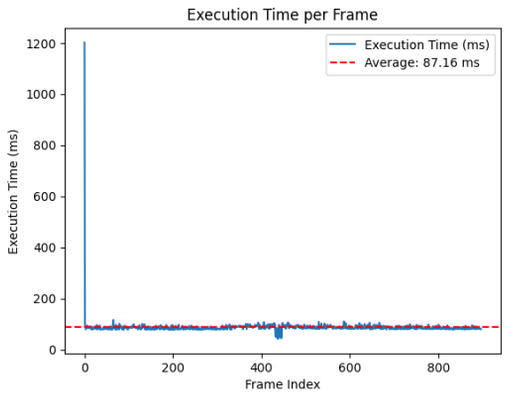
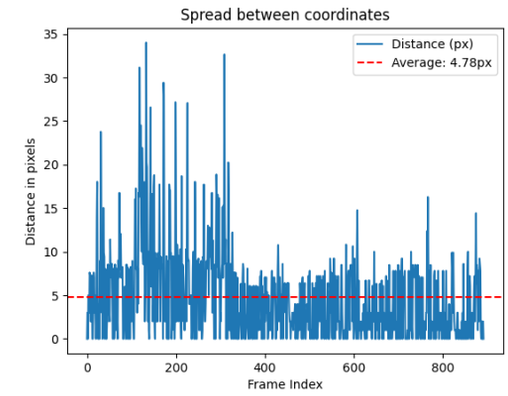
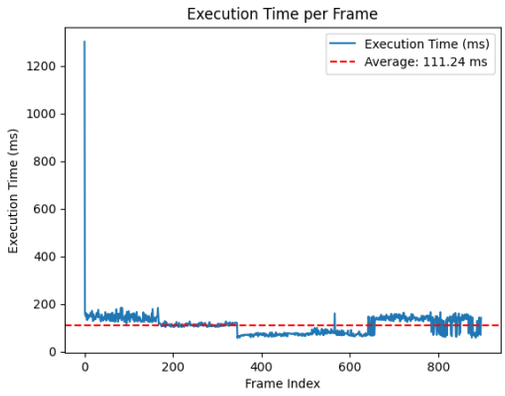
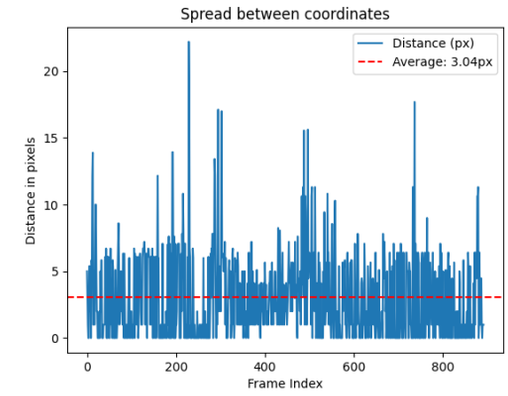
Decreasing the frame and panorama image by a percentage reduces number of pixels utilized by SuperPoint, which should improve the overall execution time per frame. However, decreasing frame and panorama by a percentage will result in images with less visual information, possibility interfering with the number of keypoints and features SuperPoint is able to detect and match. This essentially means that fewer pixels are used to represent the same scene, which can cause a loss of detail and granularity. This can lead to a less precise or accurate match between the frame and panorama, and the panorama and the user uploaded image of a soccer field (see figure 4.5 in section 4.1.2). The resulting homographies from these matches may be less granular in mapping coordinates meaning it may not accurately represent the geometric transformation between the two different images.

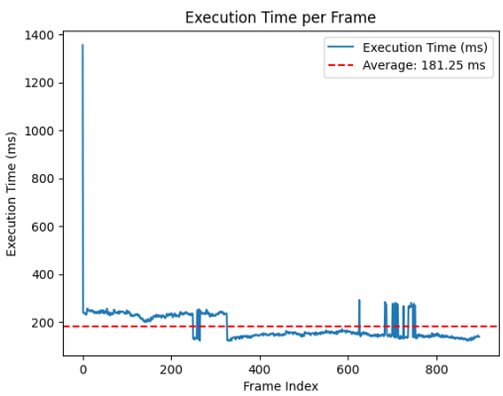
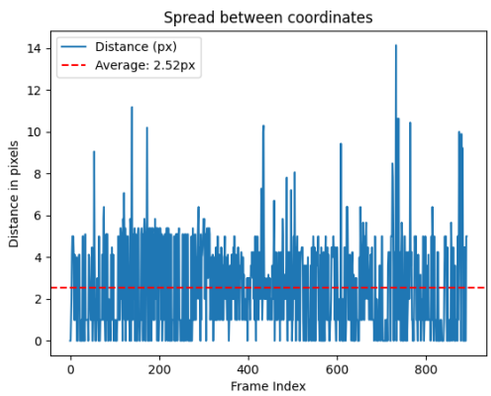
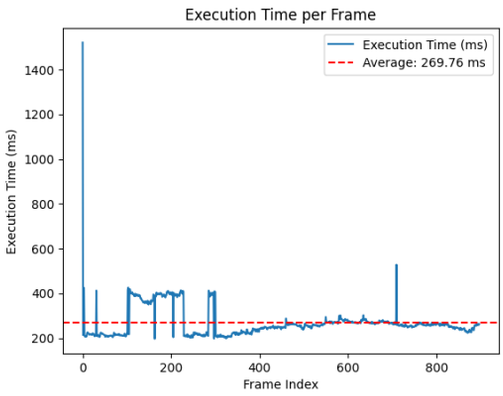
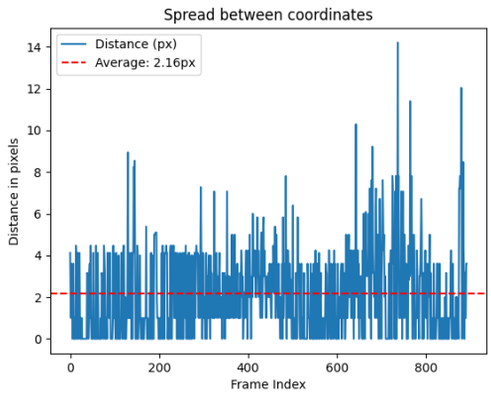
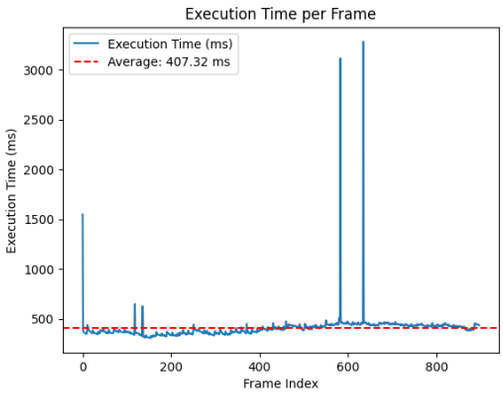
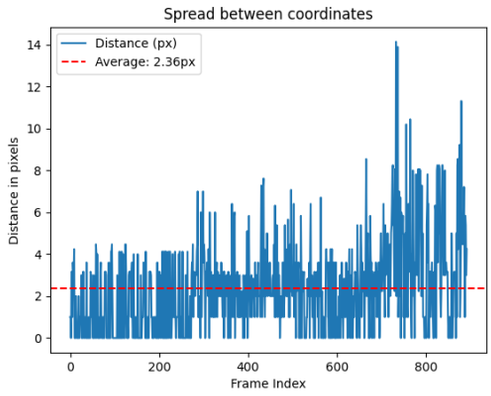
5.8.9 Percentage - Experiment

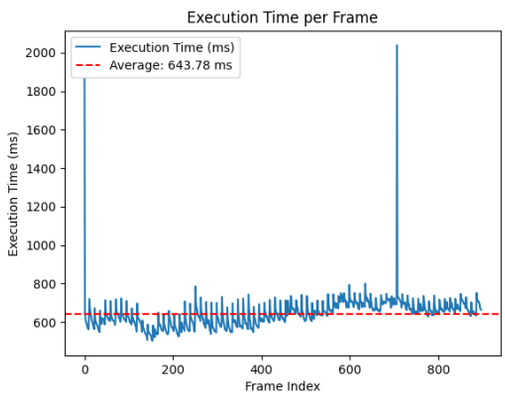
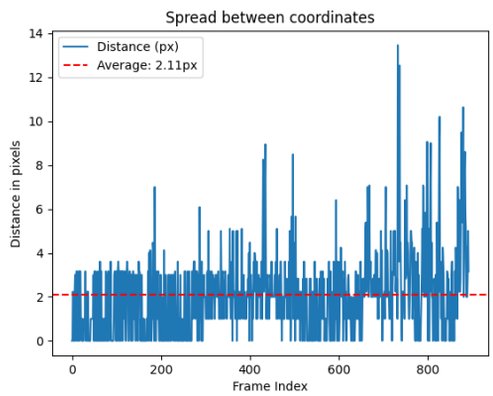
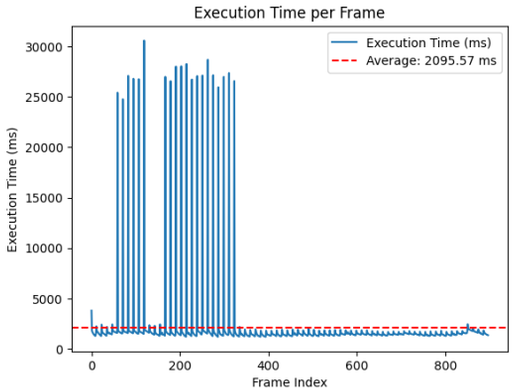
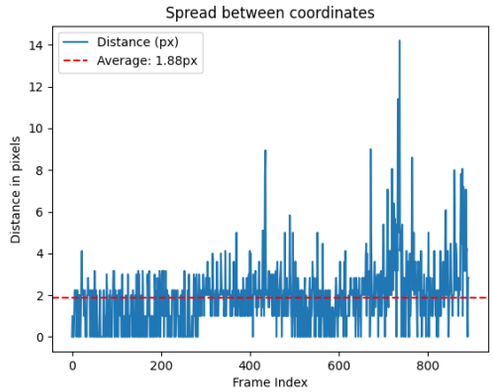
For this Experiment, we are decreasing the size of the frame and the panorama image used by SuperPoint (see section 4.4). We are testing with different percentages of original size, observing KBHMG component throughput and pixel spread (see section 5.4 for pixel spread definition). Experiment hardware is listed in section 5.1. The video utilized is 30 seconds long, with 1080p resolution and 30 fps (see footnote 2).

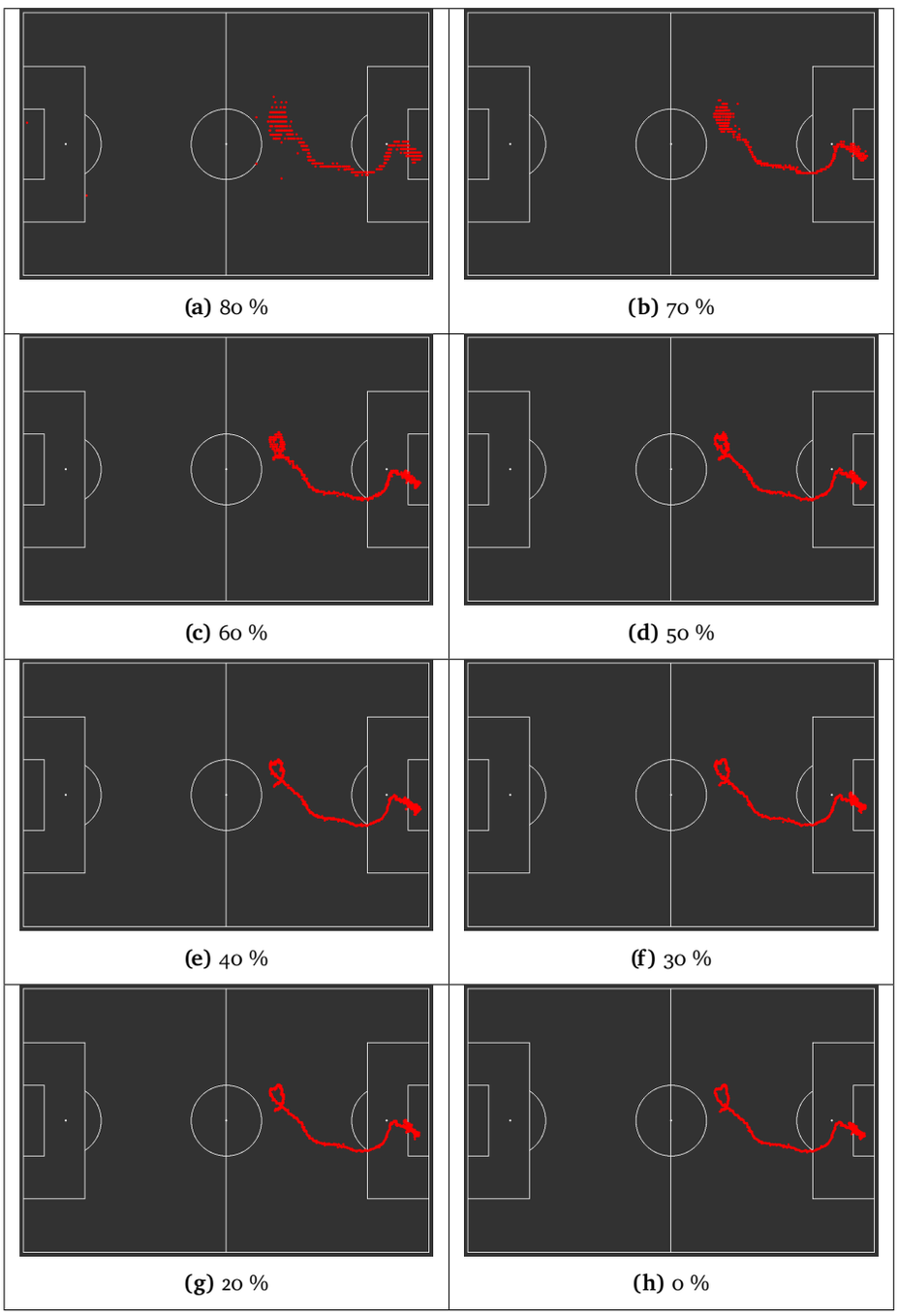
5.8.10 Percentage - Result

Table 5.8.10 shows the execution time per frame and pixel spread at different percentage reductions of original frame and panorama size. Both panorama and frame are decreased by same percentage. The first column shows how many percentage the frame and panorama are decreased by. Table 5.13 shows visualisation of how decreasing resolution of frame and panorama affects the accuracy and granularity of the produced on soccer field player coordinates. For percentage 80, average execution time per frame is **79.02ms** with a pixel spread of **14.90**. At 70 percent, pixel spread improves to an average of **4.79** with an average execution time per frame of **87.16ms**. Observation of entry a) and b) in table 5.13 shows a lack of accuracy in the resulting coordinates on the 2D soccer field image. At 60 percent, some improvement can be observed in entry c) in table 5.13. From percentage 50 to 0 we can observe a clear path of the players trajectory in entry d), e), f), g) and h). These percentages yield a clear enough frame and panorama giving SuperPoint a low enough granularity to produce accurate matches between keypoints. Pixel spread from percentage 50 to 0 remain relatively unchanged.

	Execution time per frame	Pixel spread
80 %		
70 %		
60 %		

	Execution time per frame	Pixel spread
50 %		
40 %		
30 %		

	Execution time per frame	Pixel spread
20 %		
0 %		



(a) 80 %

(b) 70 %

(c) 60 %

(d) 50 %

(e) 40 %

(f) 30 %

(g) 20 %

(h) 0 %

Table 5.13: Visualization of player coordinates at different frame and panorama percentages.

5.8.11 Percentage - Discussion

As mentioned in 5.8.8, opting for a high percentage decrease, increases the overall throughput because there are less pixels that need to be accounted for when SuperPoint detects and matches features between frame and panorama. However, as we can observe in table 5.13, a higher percentage decrease will result in a loss of positional information on the 2D soccer field image. When all components in Sadji are running, the interval at which frames are being compared with the panorama image increases, otherwise Sadji would not function in real-time. Because of this, decreasing the frame and panorama by 50 percent or 40 percent is suitable alternatives. What combinations of interval and percentage decrease result in the optimal combination of throughput and pixel spread (accuracy) will be further investigated in section 5.10.

5.8.12 Cache - Problem

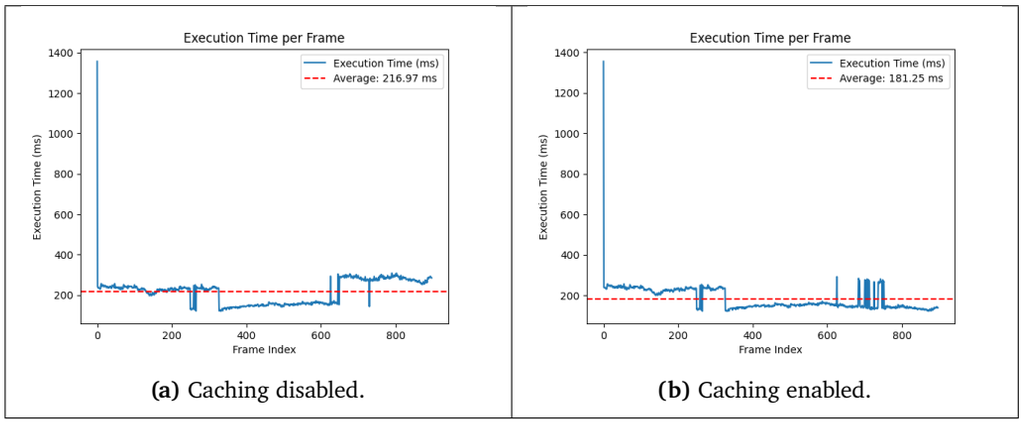
Because of a size limitation of input images to SuperPoint, as mentioned in section 4.4, the panorama needs to be split into smaller parts. For this, the panorama is split in three parts: left, center, and right. Each part is then sent into SuperPoint for comparison with the frame, and once enough matching keypoints are located, the homographies can be computed. Splitting the panorama allows for caching, as caching allows for storing results from previous rounds of keypoints computations in SuperPoint. For example, if the input frame captures the right most part of the soccer field and SuperPoint finds the most matches in the right part of the panorama, then we can cache that location for the next round. This is because the video camera recording will most likely capture the same area of the soccer field the next time around, because of the short interval duration for frame and panorama comparison.

5.8.13 Cache - Experiment

For this experiment we are investigating how caching can improve the throughput for the KBHMG component. We are using the experiment hardware specified in section 5.1. The video is 30 seconds long, recorded at 1080p and 30 fps (see footnote 2). The video captures soccer play moving from the center of the field to the right part. Both frame and panorama are reduced to 50 percent of original size (see section 5.8.8).

5.8.14 Cache - Result

Table 5.14 displays the result from the experiment. Entry a) in table 5.14 has caching disable. Between frame 0 and approximately frame 350, execution time per frame stays slightly above average for entry a). Execution time per frame remains under average in interval 350 to approximately 650 frames for entry a). After 650 frames, execution time increases to above average, maintaining approximately same execution time per frame for the remaining duration of the video. For entry b), the graph remains unchanged when compared to entry a). However, after approximately 650 frames, execution time remains at the same level as observed in interval 350 to 650. Some spikes in execution time are observed after 650 frames for entry b). Average execution time per frame with caching enabled is **181.25ms** compared to entry a) caching disabled at **216.97ms**.



(a) Caching disabled.

(b) Caching enabled.

Table 5.14: Without and with caching enabled.

5.8.15 Cache - Discussion

From approximately frame 650 the camera records the right part of the soccer field, and caches that the most amount of keypoint matches was located in that part of the panorama. This means that the KBHMG component does one less comparison, which in return improves performance. From approximately 0 to 350 frames the camera records play that is happening around the center area. The KBHMG component is not finding enough matches between the frame and the center part of the panorama and needs to check the left part of the panorama also. Execution time for both entry a) and b) between frame 0 and 350 is because of this higher. The efficiency of caching specific parts of the panorama is directly proportional to the duration of the video, as a longer video increases opportunities for iterative usage of the cached part.

5.8.16 Players/Commercials - Problem

We hypothesise that the live-commercials will decrease accuracy because they move and change on the digital display throughout the duration of a video. Most of the commercials also have the exact same graphics in multiple location, potentially confusing the SuperPoint algorithm. Player within the frames composing the panorama (see section 4.1.2) may also potentially influence the accuracy of SuperPoint, because they may introduce variables impacting the fidelity of feature extraction between frame and panorama.

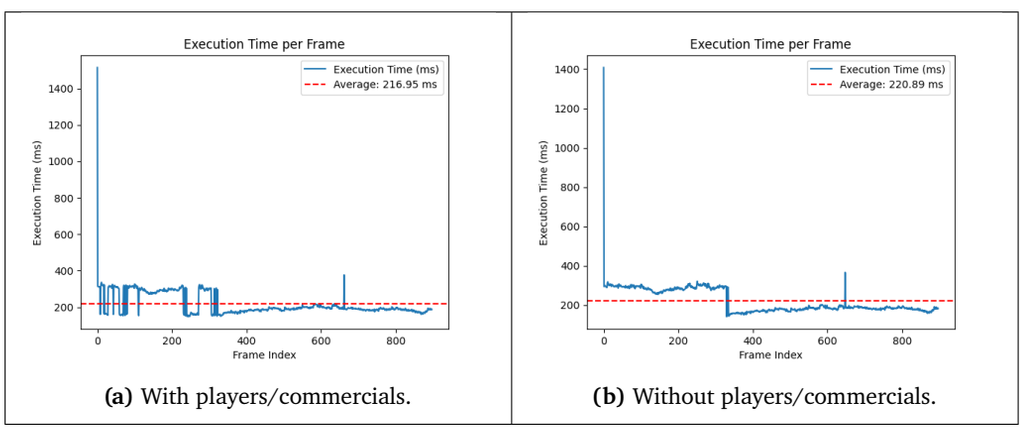
Figure 5.7: Panorama with players/commercials removed.

5.8.17 Players/Commercials - Experiment

For this experiment, we have removed the players and commercials/live screens from the panorama image (see figure 5.7 for example). Players/commercials are removed using a third party software. Hardware specifications are listed in section 5.1 and we are utilizing a 30 seconds video with 1080p and 30 fps (see footnote 2). Frame and panorama are decreased by 45 percent (see section 5.8.8 for details on percentage decrease).

5.8.18 Players/Commercials - Result

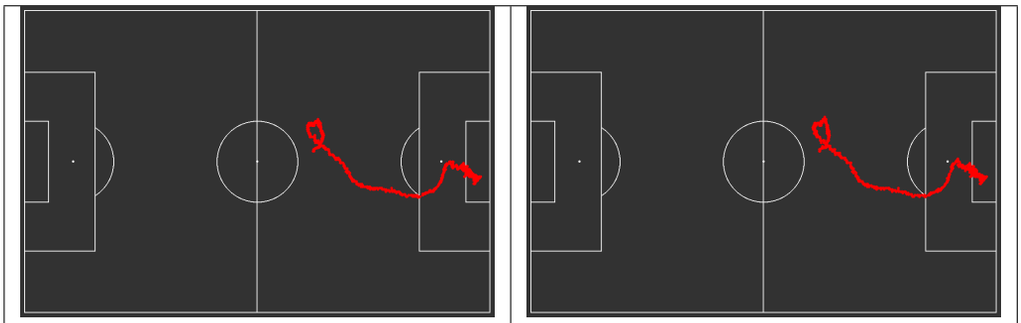
Table 5.15 shows experiment results. Without players/commercials averages **220.89ms** per frame. With players/commercials averages **216.95ms**. Table 5.16 shows produced trajectory of player 1. No noticeable visual deviation is observed between the two cases.



(a) With players/commercials.

(b) Without players/commercials.

Table 5.15: With and without players/commercials enabled.



(a) With players/commercials. **(b)** Without players/commercials.

Table 5.16: Visualization with and without players/commercials enabled.

5.8.19 Players/Commercials - Discussion

Removing players/commercials made no noticeable accuracy or performance gains as observed in table 5.16 and 5.15. Because of this, we did not make any design/implementation altercations such as incorporating automatic player and/or commercials removal from the panorama image. SuperPoint, regardless of players/commercials being present, functioned as expected.

5.9 KBHMG Component Experiments Summary

Table 5.17 summarises the findings from section 5.8. The sweet spots for combinations of metrics is highlighted in green in table 5.17. We found that at these metrics, throughput was relatively high without sacrificing the pixel spread. These metrics are configurable by a user of Sadji, meaning if a user wishes they can alternate metrics to for example lower throughput and decrease pixel spread (improve accuracy). Table 5.17 measures throughput in fps, detailing how many frames SuperPoint, in the KBHMG component, can process per second.

GPU	Crop	Percentage	Cache	Players / Commercials	Throughput (fps)	Spread (px)
X	✓	0	X	✓	0.037	N/A
✓	X	0	X	✓	0.21	N/A
✓	✓	0	X	✓	0.45	N/A
✓	✓	0	✓	X	0.47	1.88
✓	✓	0	✓	✓	0.45	1.77
✓	✓	20	✓	✓	1.55	2.11
✓	✓	30	✓	✓	2.45	2.36
✓	✓	40	✓	✓	3.71	2.16
✓	✓	45	✓	✓	4.62	2.14
✓	✓	45	✓	X	4.53	2.38
✓	✓	50	✓	✓	5.52	2.52
✓	✓	60	✓	✓	8.98	3.04
✓	✓	70	✓	✓	11.47	4.78
✓	✓	80	✓	✓	12.65	14.90

Table 5.17: Iterations of SuperPoint implementation.

5.10 System Performance

This section will cover the systems performance when running the entire pipeline on a single node. Running the components simultaneously reduces overall system performance in terms of throughput. This is because there is a resource contention and scheduling conflict between the components. This is particularly the case for the detection and tracker component (section 4.3) and the KBHMG component (section 4.4). Both of these use GPU intensive algorithms that can lead to context switching overhead because the GPU has a finite number of processing units and memory bandwith.

5.10.1 Experiment

For this experiment, we will run the entire Sadji pipeline on a single computer. Hardware specifications used is listed in section 5.1. We are using a 30 seconds long video, with a resolution of 1080p and 30 fps (see footnote 2). We are testing for the overall throughput of Sadji, using fps as measurement unit. We are also investigating execution time per frame for the KBHMG and detection and tracker component. We will experiment with different intervals (see section 4.3 on interval details) and different percentages of original frame and panorama resolution. Both frame and panorama are reduced to the same percentage (see

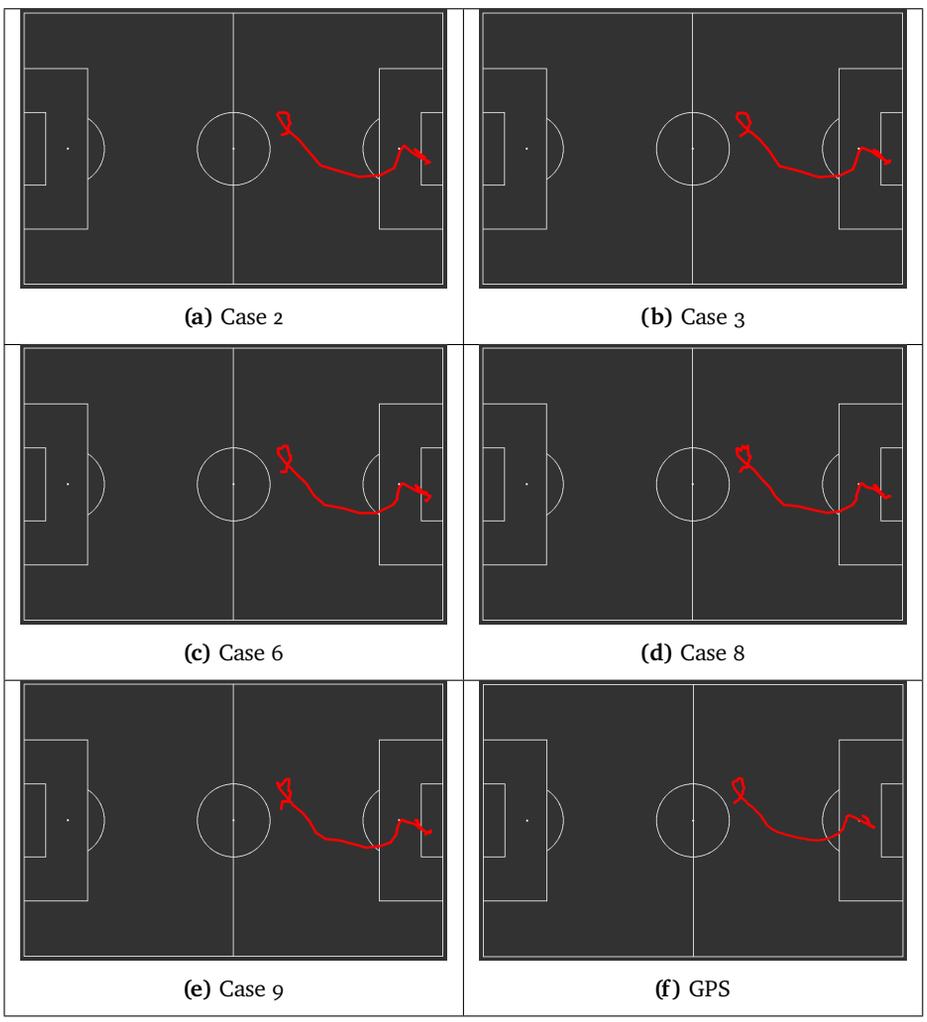
section 5.8.8 for details on percentage parameter).

5.10.2 Result

Table 5.18 shows system throughput for different values of interval and percentage size of original frame and panorama. Cases highlighted in green achieve real-time throughput where threshold for real-time with the video utilized for the experiment is over 30 fps. Some cases are above this threshold, however they are so close to the threshold that they are not reliable cases. Table 5.19 displays visual representation of trajectory for player 1 (number 4 in the video) for cases that are highlighted in green in table 5.18. Case 9 has the lowest interval at 20, meaning it has the fewest number of interpolated coordinates, producing a more frequent representation of player position. It however, has the highest percentage decrease with 65 percent reduction in frame and panorama resolution.

Case	Interval	Percentage	KBHMG (ms)	Detection and Tracker (ms)	Throughput (fps)
1	30	45	518.57	35.70	30.10
2	30	55	315.05	29.41	32.48
3	30	60	269.76	30.41	33.77
4	25	50	410.81	35.14	28.70
5	25	55	307.50	32.72	30.71
6	25	60	260.84	31.14	32.88
7	20	55	296.99	33.91	29.85
8	20	60	246.65	32.42	31.55
9	20	65	223.66	31.94	31.85
10	15	60	240.65	34.55	29.50
11	15	65	213.31	33.46	30.49
12	15	70	204.60	32.86	30.63

Table 5.18: Experiments with different interval and percentage on single node.



(a) Case 2 **(b)** Case 3 **(c)** Case 6 **(d)** Case 8 **(e)** Case 9 **(f)** GPS

Table 5.19: Visualization of player coordinates for different cases.

5.10.3 Discussion

As previously mentioned, some entries in the table may meet the required throughput for real-time applications, but are close to the threshold meaning they are not viable alternatives. This closeness to the threshold means that even small fluctuations in system load or data input could push them over the limit, causing them to miss real-time deadlines. Because of this, these cases are considered a risk for real-time applications due to their lack of reliability in maintaining the needed performance levels consistently. Coordinates in between the intervals are interpolated, as is covered in section 4.6.

5.11 Alternative Video Source

Panorama utilized as example in the Sadji pipeline is captured from a Hudl camera. Locating frames from the Hudl video works well because of this. However, Eliteserie broadcast footage records the soccer games from a similar position as the Hudl installments on the different arenas. Because of this we hypothesize that frames from an Eliteserie video source will also function with Sadji.

5.11.1 Experiment

For this experiment we are using an Eliteserie highlight[3] clip recorded at SR-Bank arena. The YOLOv8 model utilized can also classify which team the player belongs to (same model utilized as in section 5.2). Experiment hardware is listed in section 5.1 and we are using a 10 second long clip, recorded at 1080p with 30 fps. We are investigating the pixel spread and trajectory visualization (see section 5.4 for description). We are tracking player 2[4]. Frame and panorama are decreased by 45 percent.

5.11.2 Result

Figure 5.8 displays produced coordinates, with figure 5.9 details the achieved pixel spread. Pixel spread averages to **3.11**. Approximately around frame 170 we observe three spikes in pixel spread values. This can also be observed in figure 5.8, as there are visuals gaps in player 2 path towards the penalty area.

3. URL: `https://highlights.eliteserien.no/playlist/apqoh3i03rncz`
4. Second test player for TIL

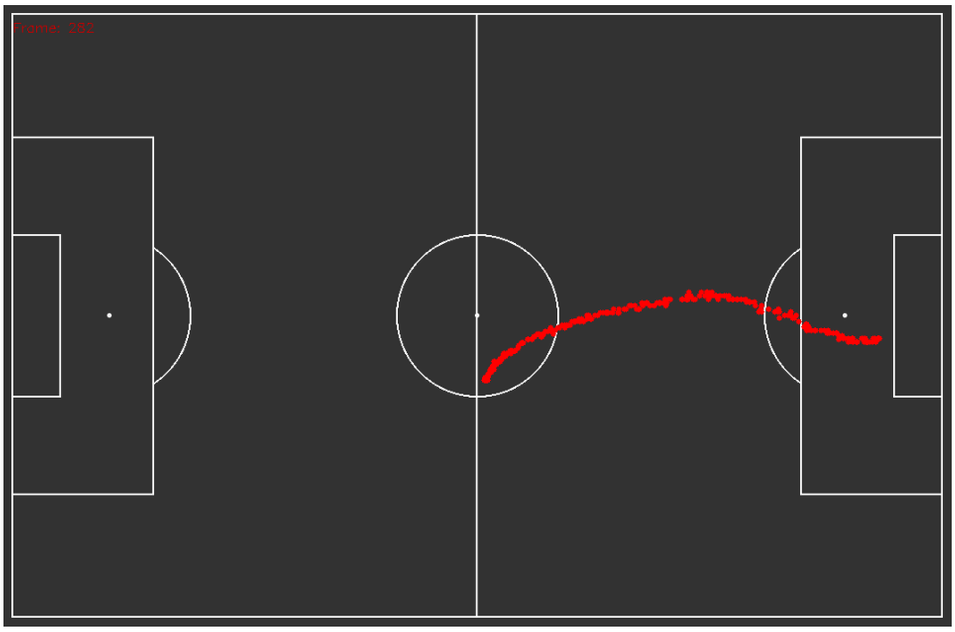


Figure 5.8: Coordinate trajectory for ID 7 in Eliteserie highlight clip.

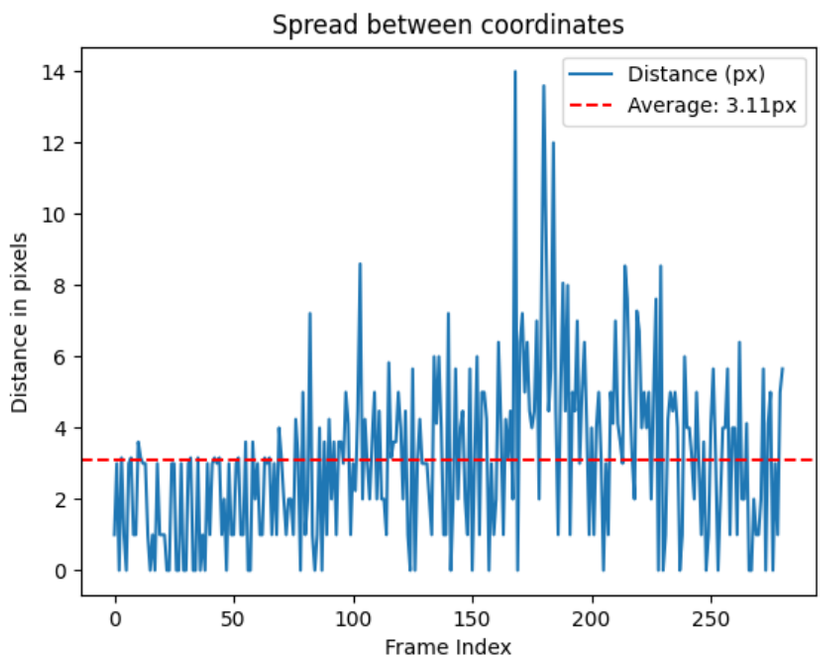


Figure 5.9: Pixel spread using Eliteserie highlight clip (decrease percentage is 45).

5.11.3 Discussion

The camera pans rapidly in interval 150 to 200 frames in the eliteserie highlight video, possibly being the result of the spikes we observe in figure 5.8. Before and after frame interval 150 and 200 we observe a steady pixel spread, which aligns with the cameras more slowly pan movement. The pixel spread using Eliteserie highlights clip is ≈ 3.11px, which is similar to achieved pixel spread using Hudl as video source as seen in table 5.17. The slightly higher deviation in pixel spread can be a result of player 2 running at high speeds from the midfield area to the penalty box and the eliteserie highlight camera moving rapidly. Figure 5.10 shows visually how one of the frames from the clip was placed into the panorama.

The results from the experiment performed in this section is an interesting discovery, as Eliteserie broadcasting records closer up to the soccer play, possibly enabling more accurate tracking at the detection and tracker component. Closer up footage also lays the foundation for the possibility of accurately being able to detect and track the soccer ball.

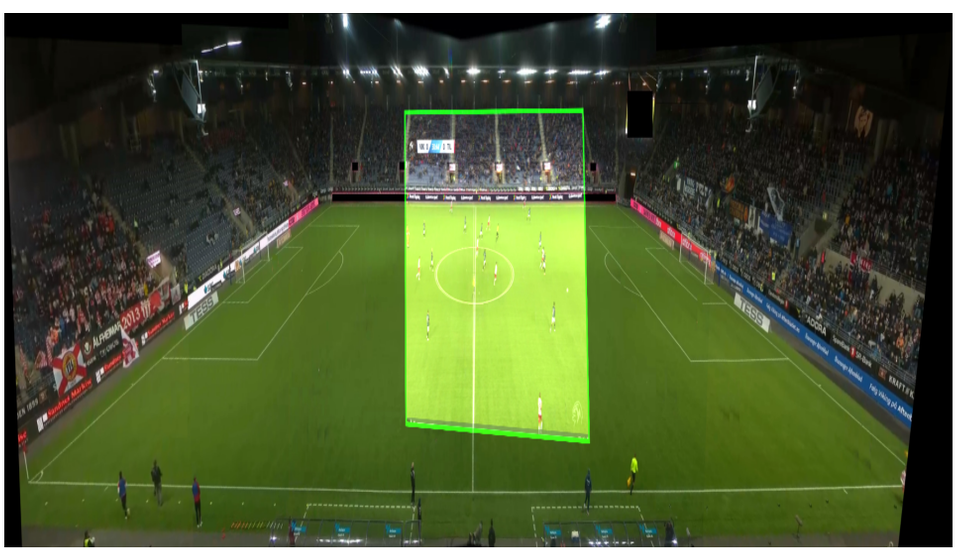


Figure 5.10: Eliteserie highlight frame placed in panorama.

5.12 Cleaning Tool - Connecting IDs

The connecting IDs feature of the cleaning tool (see section 4.7.1) aims to decrease number of total IDs in produced metadata from the metadata aggregator component (see section 4.6). For it function, it needs to be able to

detect what IDs are missing and which newly created IDs are good matches for connecting.

5.12.1 Experiment

To test the effectiveness of the connecting functionality of the cleaning tool, 3 clips have been run through the coordinate translation pipeline producing 3 metadata files. The clips are chosen based on interesting events taking place during the video (play leading to goal attempts for all 3 clips). A test-user has then been instructed on how to operate the cleaning tool (see section 4.13) before cleaning produced metadata from the 3 clips. Experiment hardware is listed in section 5.1 and the videos are all 1080p with 30 fps. Clip length is listed in table 5.20.

5.12.2 Result

Achieved results are listed in table 5.20. Optimal number of IDs column in table 5.20 indicates how many unique players are part of the video clip. For clip 1 and 2, this is 21 because home goalkeeper is out of the cameras view. For clip 3 all players and goalkeepers appear throughout the video. Improvement was made at all clips with clip 3 reducing number of unique IDs with 50 percent.

Clip	Length (*seconds*	IDs before	IDs after	IDs optimal
1	30	42	30	21
2	49	43	31	21
3	30	50	25	22

Table 5.20: User testing for cleaning tool.

5.12.3 Discussion

IDs that were not connected were a result of them disappearing and no newly created ID was assigned to the player within the 2 second window. This 2 second window is a configurable option. However, any longer duration than this will result in the spatial locality being inaccurate because the camera may have moved to much resulting in faulty closest IDs. Figure 5.11 shows one connecting interface that did not include the player whom the disappeared ID belonged too.

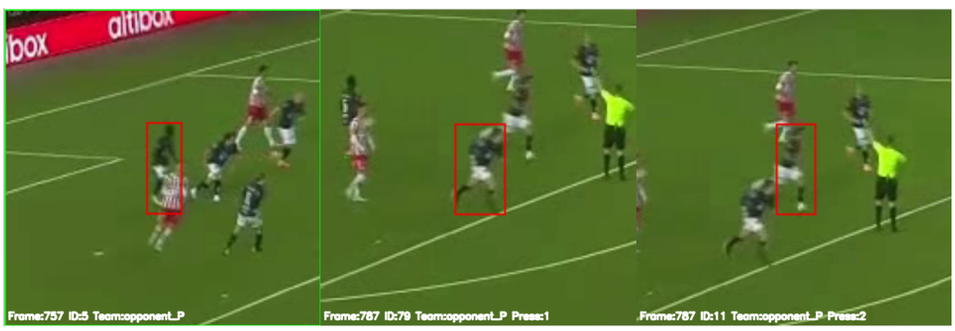


Figure 5.11: Connecting ID not found.

5.13 Summary

In this chapter we have investigated and evaluated different components and performance of Sadji. Evaluations shows that the pipeline presented in chapter 4 on design and implementation functions with a high level of precision and speed, as detailed in section 5.10. Section 5.8 and 5.7 shows evaluation performed during the implementation stage of Sadji, serving as a foundation for why certain choices were made. Each iteration of the KBHMG component has gradually aimed at increasing overall frame throughput measured in fps to approach real-time while maintaining a low pixel spread (see section 5.4 for pixel spread definition). Table 5.18 in section 5.10.2 highlights in green what combinations Sadji can be configured with to achieve real-time performance.

In section 5.11 we evaluated how Sadji performs utilizing an alternative video source, meaning a video source from which the panorama image in section 4.1.2 is not created from. This experiment gave confirmation that using an alternative video source, such as Eliteserie highlight clips, is viable. In section 7.3.2 we discuss a use-case of this discovery.

Finally, in section 5.12 we investigated the cleaning tool performance for reducing the total number of IDs from produced metadata from the Sadji pipeline.

6

Discussion

In this chapter we will discuss the findings from evaluation chapter 5 and design/implementation in chapter 4 on how the systems results match with the original system requirements specified in chapter 3. In section 6.1 and 6.2 we will discuss the design and implementation process and how the results from the evaluation of each incremental choice shaped the final version of the system. In section 6.3 we will present the cleaning-tool POC performance and how it can be integrated into a future system for cleaning, clipping, and analysing video segments.

6.1 Sadji: Functional Requirements

Sadji has the following listed as functional requirements:

- **Non-invasive tracking on soccer field.**
- **Detection model.**
- **Video source.**
- **Frame-panorama mapping.**
- **Coordinate translation to user provided soccer image.**

- **Formatted metadata for user specification.**
- **Panoramic image creation.**
- **Configurable.**

Sadji can accurately track players on a soccer field **without using any invasive technologies**, as showcased in table 5.17 in section 5.9. The most common invasive method for tracking soccer players is GPS. When comparing produced STATSports GPS coordinates for player 1 (see table 5.3) we can observe that produced coordinates from Sadji are near identical. Producing the coordinates shown in table 5.13 (h) in section 5.8.8 is however not achievable in real-time as every frame from the video segment is translated. Interpolation is therefor introduced to reduce the overall execution time as detailed in section 5.10. With interpolation Sadji achieves similar trajectory compared to GPS coordinates, as shown in table 5.19.

On initialize of the detection and tracker component, any YOLOv8 **detection model** will work, regardless of what type of classes the model is trained on. The detection and tracker component contains an API route for upload of models called */upload_model* which stores the model at the component. If no model has been uploaded to the detection and tracker component, it defaults to a detection model that only detects players (not their team affiliations).

Any **video source** that conforms with the OpenCV framework works with Sadji. Common video formats such as *.mp4* and *.avi* works as expected. Sadji can also handle URL for stream sources such a HTTP Live Streaming (HLS) (*.m3u8* format).

Sadji supports the building of large high resolution **panoramic images** by using the illustrated method in figure 4.3 from section 4.1.2. As of now, the functionality for building the panoramic image is not integrated into the pipeline detailed in section 4.2. This is because the panoramic image functionality is intended to be integrated with an interface that a potential user interacts with. The process of creating a panoramic image has many similarities to how the KBHMG component works (section 4.4) which is why Sadji also supports this. Figure 6.1 illustrates how a user interface for creating panorama could look like:

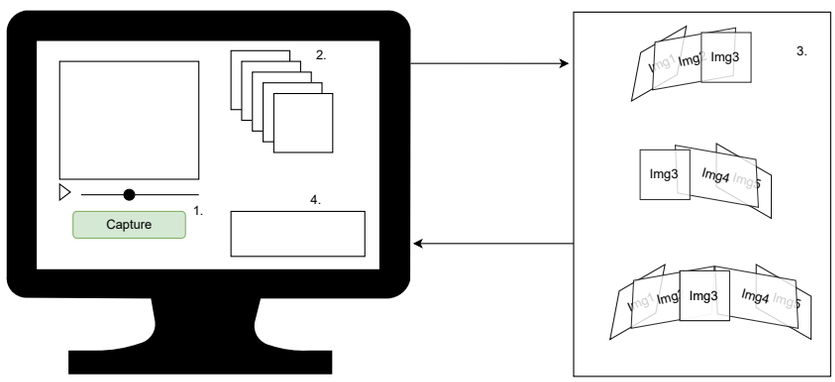


Figure 6.1: Example of how user interface for panorama creation could look like.

1. User interacts with stream or video, making screenshots by pressing capture.

2. Once 5 images that roughly capture the playing field have been captured, send to Sadji panoramic image creator for stitching.

3. Panoramic image is created, see section 4.1.2 for in-depth details.

4. User receives created panorama image and can use this to initialize the Sadji coordinate production pipeline.

Cropping the panorama to reduce the overall size is also something that could be incorporated into such a user interface.

Hudl is the main recording and analysis software used in the Norwegian soccer league Eliteserien[57]. During the writing of this thesis, no method for extracting a panoramic image from the Hudl framework is available, and is partially why we have made a tool for this process. A panoramic image created by Sadji will work in the coordinate translation pipeline for each stadium fitted with Hudl cameras as long as the stadiums appearance remains relatively unchanged. If a panoramic image does not work, then a user could again use for example the process illustrated in figure 6.1.

Frame-panorama mapping works for multiple video source that record the same soccer field. This is shown in section 5.11 where we use an Eliteserie highlight clip. This is an interesting discovery as the panorama image is composed of several images from the Hudl video source and the camera placement between Hudl and the device capturing the Eliteserie highlight clip from the match differs. Even with these conditions, Sadji is able to translate coordinates between video and user provided soccer image accurately.

Sadji supports **coordinate translation to user specified soccer image** as long as the keypoints conform to the keypoints displayed in figure 4.5. These keypoints can be taken as arguments to the detection and tracker component. Sadji has functionality for manually creating these keypoints in the user provided soccer image as well as the panorama image created from the video source. There is currently no user interface other than an OpenCV window for clicking and generating these keypoints. Such a user interface for clicking on the image to generate the needed keypoints can be integrated into for example the illustration in figure 6.1.

Sadji supports the production of coordinate metadata for both stream and video sources. Depending on what the user wants, the produced metadata can either be segmented into smaller packages or be one large package as illustrated in figure 4.11. If the source is a stream, the manifest file is read to retrieve the segment size and fps to correctly pack produced metadata into correctly sized segments. When the metadata aggregator component is finished aggregating data, either into individual segments or one large package, it sends the data back to the user.

The Sadji pipeline is highly **configurable** accommodating different user defined requirements. As mentioned, a user can provide different types of stream inputs, with different sized segments (normal is 2 seconds). Sadji can handle alternate pipeline parameters such as percentage decrease and coordinate translation interval (see section 4.3) to best suit a users system performance needs, for example real-time.

6.2 Sadji: Non-functional Requirements

In this section we will present how Sadji performs in terms of the non-functional properties listed in chapter 3:

- **Real-time.**
- **Accuracy.**

- **Automatic coordinate production.**
- **Integration.**
- **COTS components compatibility.**
- **Data ownership.**

Generating soccer field coordinates fast is crucial in modern soccer analysis. Increasing reliance on data-driven insight for precise spatial information is of the upmost importance for success. Real-time analysis is only achievable with an equally fast production of these field coordinates. Sadji, on **common of the shelf (COTS)** produces such coordinates at **real-time** performance. Table 5.18 demonstrates Sadji throughput at different configurations based on user requirements. As mentioned in section 5.10 all performance evaluation is performed on a single computer with the provided hardware listed in section 5.1.

Sadji components (see section 4.2) are split into designated Flask servers designed to communicate through APIs enabling seamless cross computer communication. We believe that introducing a second computer will increase the overall throughput of Sadji. A higher throughput will in turn lead to a reduction in interpolated coordinates, producing a more accurate representation of player coordinates on the soccer field.

Sadji, when compared to GPS produced coordinates, produce an equally accurate representation of player positions. This is illustrated in table 5.19. As previously stated, introducing a second computer as host for one of the Flask servers can increase the overall performance, increasing the rate of which player coordinates can be translated further improving the accuracy.

Sadji is meticulously designed with **integration** in mind. Output adheres to a standardized format that has been collectively agreed upon by the creators of Sárgut and Guorrat from CSG working on analysis tools utilizing produced data from Sadji. Inputs to Sadji listed in section 4.1 on prerequisites has also been considered during the design phase of the system. Figure 6.1 illustrates a user interface that could be utilized to produce necessary input for Sadji. Sadji facilitates integration with such a proposed system using well-defined APIs, streamlining the assimilation process.

Input to Sadji is only temporarily stored in memory solely for the purpose of processing and producing soccer field coordinates. Once the requested data is generated, Sadji discards all input, maintaining the requirement that the **ownership of the data** is solely the user.

6.3 Cleaning Tool Discussion

The cleaning tool is designed as a proof of concept, investigating whether or not it is possible to create a system for cleaning the produced metadata from the Sadji pipeline. As of now, the cleaning tool user interface is created using OpenCV windows and the terminal as input form for registering user input. Section 5.12 shows that the cleaning tool works well for a lot of cases, reducing the number of redundant IDs. It is good at detecting when a new ID appears and what other IDs nearby are good candidates for connecting. The small crops showcasing the player the ID belonged to will for someone who is acquainted with the team be particularly useful, as they have more experience and knowledge about player jersey numbers and other unique features associated with the player. This is a common approach used in a lot of industry products such as Forzify[2] by Forzasys[1].

However it is not perfect, the cleaning tool uses exclusively position data that comes from the YOLOv8 detection models inference, and does not use the soccer field coordinates produced from the Sadji pipeline. This is a shortcoming, as the soccer field coordinates for players have the potential to accommodate the ID connecting by using a second coordinate system for finding nearby IDs.

6.4 Summary

For this chapter, we have re-stated the requirements specifications listed in chapter 3, evaluating Sadji's adherence to the functional and non-functional requirements outlined. Functional properties such as non-invasive tracking on the soccer field, diversity of video sources, configurability, and coordinate translation to user-provided soccer field image have been examined and evaluated. Additionally, the systems compliance to the non-functional properties of real-time execution speeds, accurate tracking and detection, and architecturally engineered to fit proposed system model (see section 3.3) has been thoroughly investigated. By integrating these evaluations, this chapter provides a comprehensive understanding of Sadji's capabilities, limitations and position in proposed system model.

7

Conclusion

This chapter presents concluding remarks, restating the original problem definition from section 1.2 and a summary of the thesis. We will also present some future work and how Sadji into a potential digital twin coach (see section 1.5).

7.1 Concluding Remarks

In section 1.2, the following thesis problem definition was made:

> ***This thesis aims to develop a system capable of automatically and precisely detect and track soccer players in video footage. Utilizing computer vision, the system will be capable of determining the team affiliation of each player. The goal is to promptly and accurately map players to their respective on-field coordinates, with the goal of real-time speeds. The research within this thesis will explore the YOLOv8 deep learning models for the precise detection of players and their teams in video content using MOTs for tracking. Additionally, the investigation will look into methods for accurate and efficient spatial coordinate mapping of soccer players from video to a top-down view image of a soccer field.***

Adhering to the design paradigm presented in section 1.3.3, this thesis has been meticulously written by first specifying what system requirements are present. Chapter 3 details what non-functional and functional properties are required with section 3.3 highlighting Sadji's adherence or placement in a practical real-world system.

Chapter 4 presents the design and implementation of Sadji, detailing the construction of each component Sadji is composed of. Each section details functionality, overall architecture of each component, inter-component connections and technologies utilized. Furthermore, each section advocates for certain design decisions by referencing experiments performed in chapter 5, fostering adaptability and improvements throughout the implementation. This process of refinement adheres to the testing stage of the design paradigm, providing an approach of continuous optimization, revelation of unexpected flaws, and ultimately providing a higher quality of work.

Key findings are presented in chapter 5, with section 5.10 providing a comprehensive evaluation of how the Sadji system performs utilizing COTS on a single computer or node. Section 5.8.8 investigates how size reduction of frame and panorama resolution affects throughput and accuracy when detecting and matching keypoints between them using SuperPoint. This investigation reveals that a *high level of accuracy is maintainable with a decrease in resolution of over 50 percent while improving overall execution time per frame from **2095.57ms** to **216.97ms*** (without caching enabled). The introduction of caching part of panorama with most matches, presented in section 4.4 and evaluated in section 5.8.14, reveals *improvement in execution time per frame from **216.97ms** to **181.25ms***. *Camera placement is identified as vital to the inference accuracy and MOT tracking* (see section 5.5 and 5.6) with *elevation and distance from the soccer-field* highlighted as key properties for accurate results. Alternative video sources, from which the panorama image was not created out (see section 5.11), is *revealed as a viable and functional option*, laying the foundation for future work where such footage is utilized to complement ball detection, tracking, and analysis.

Interpolation of player coordinates was introduced because of insufficient execution time per frame (approximately 33.33ms is needed for real-time speed with 30 fps video) when detecting and matching keypoints between frame and panorama. However, as displayed in table 5.19, interpolation of player coordinates at low intervals produces a similar trajectory to that of GPS (GPS position is updated every 100 ms).

IDs disappearing randomly and occlusion as a result of faulty MOT tracking remains a challenging problem. Because of the nature of soccer, players run close to one another, obscuring the line of sight between player and camera.

Elevated, and further back camera placement proved beneficial, however a tool for cleaning produced metadata from Sadji could help with this problem. The cleaning tool introduced in section 4.7 is implemented as a POC with evaluation in section 5.12 showcasing the effectiveness of the connection mechanism for reducing total number of IDs. Occlusions (player swapping IDs) remains a challenge but, as we will detail in section 7.3.3, solutions such as multiple cameras may be possible.

7.2 Thesis Summary

This master thesis introduces Sadji, a system designed for automatic translation of player coordinates from video frames to corresponding locations on a 2D soccer field image. Leveraging state-of-the-art technologies, Sadji utilizes YOLOv8 object detection models for precise player detection and team affiliation supplemented with accurate tracking using MOTs through the Ultralytics framework. Incorporating SuperPoint for keypoint detection and feature matching, Sadji seamlessly locates video frames in a panoramic image composed from the video source to produce homography matrices that can translate player coordinates from video frames to their respective location on the 2D soccer field image. This happens in real-time utilizing COTS hardware and existing camera systems installed on soccer arenas, like Hudl or Eliteserie broadcasting footage, enabling widespread adoption and accessibility. By delivering accurate player positions on a 2D soccer field, Sadji supplies third party software systems (such as Guorrat and Sárgut) the data needed to perform advanced analysis of player tactics, strategic insight, and formations. Additionally, Sadji offers a unique advantage over invasive tracking technologies such as GPS by providing the team utilizing it valuable insight both their own players positions and those of their opponents. This insight can potentially allow for invaluable insight into opposing team tactics, formations, and strategy in real-time during a soccer game.

7.3 Future Work

As highlighted through the design chapter and described in requirements chapter, Sadji is created as a component of a potential larger future pipeline. There are several interesting new work that can be applied to Sadji.

7.3.1 Image Clustering for Team Classification

As of now, the detection model defaults to a normal player detection model if no other model is provided by the initiator of Sadji. Soccer jerseys between opposing teams need to be distinguishable according to law 4.3[3] presented by the International Football Association Board. This is a handy law when we consider the possibility for utilizing image clustering techniques such as Deep Adaptive Clustering (DAC)[12], deep density-based image clustering (DDC)[49] or DeepCluster[11] for the determination of player team affiliation. Such technologies have the potential to analyze crops of team jersey design produced by a general player detection model like the one Sadji defaults to. In return, team classification can be automated and dynamically applied depending on what teams are playing against one another eliminating the need to train a new model for opposing team combinations.

7.3.2 Ball Detection

As stated in section 5.11, video sources recorded from other cameras than the panoramic image was created from works very well with Sadji. This is an interesting discovery because now more close up footage such as the one produced by broadcasting cameras can be used to gain more detail from the soccer play. One drawback of using the Hudl camera systems as input is that the ball becomes tiny because the priority for this system is to frame as many players as possible at any given time. Because of this, detecting the ball poses a challenge. However, the broadcasting cameras follow the soccer play more closely, giving a more zoomed in video source of the game. The SoccerSum dataset[53] introduces a large dataset consisting of 750 annotated frames with 10 different classes including the ball. The dataset is comprised of frames from the Norwegian Eliteserie, making it a great match. Ability to track the ball, lays the foundation for analysis of player movement in relation to the ball which can provide interesting insight into strategy and other gameplay dynamics.

7.3.3 Multiple Cameras

Introducing multiple cameras around the target soccer field can potentially help with occlusion issues in player detection. Comprehensive coverage and overlapping viewpoints minimize blind spots and the possibility of players obstructing line of sight between one another, ensuring players are captured from various angles. Redundancy allows the system to rely on other cameras if one's view is obstructed. Different perspectives help distinguish occlusion from genuine player interactions, aiding accurate tracking and more persistence in the produced IDs by the trackers. Dynamic switching between feeds can

ensure continuous tracking even amidst occlusion. Figure 7.1 illustrates how using multiple cameras for tracking could look like. If more than 3 cameras are utilized than a consensus algorithm could be utilized such as Raft[40] or Paxos[30] to achieve quorum once an occlusion event happens. Such an event could be automatically detected in that the different cameras no longer agree on a position of an ID, because it has been occluded.

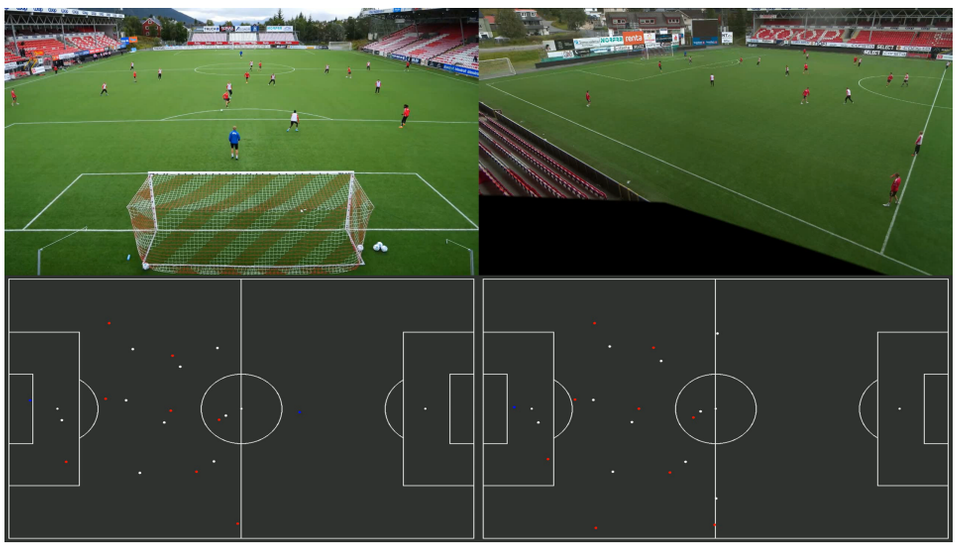


Figure 7.1: Multiple cameras.

References

[1] Forzasys. https://forzasys.com/. [Accessed 03-05-2024].

[2] Forzify. https://www.forzasys.com/Forzify.html. [Accessed 03-05-2024].

[3] Law 4 - The Players' Equipment | IFAB — theifab.com. https://www.theifab.com/laws/latest/the-players-equipment/#offences-and-sanctions. [Accessed 04-05-2024].

[4] General Data Protection Regulation (GDPR). https://gdpr-info.eu/, 2022. [Accessed: 12-12-2023].

[5] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[6] Sara Akan and Songül Varlı. Use of deep learning in soccer videos analysis: survey. *Multimedia Systems*, 29(3):897–915, 2023.

[7] Mohammad Al-Sa'd, Serkan Kiranyaz, Iftikhar Ahmad, Christian Sundell, Matti Vakkuri, and Moncef Gabbouj. A social distance estimation and crowd monitoring system for surveillance cameras. *Sensors*, 22(2):418, 2022.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

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

[10] Børge Bårdsen. Sárgut: Architecting of a real-time soccer performance analytics software toolkit. Master's thesis, UiT The Arctic University of Norway, 2024. Will be submitted for review, May 15, 2024.

[11] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[12] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887, 2017.

[13] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, Paul R. Young, and Peter J. Denning. Computing as a Discipline. *Communications of the ACM*, 32(1):9–23, 1989.

[14] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[15] Rublee Ethan. Orb: An efficient alternative to sift or surf. *ICCV, 2011*, 2011.

[16] Dirk Farin, Susanne Krabbe, Wolfgang Effelsberg, et al. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, volume 5307, pages 80–91. SPIE, 2003.

[17] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[18] Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis: Nonparametric discrimination: Small sample performance. 1952.

[19] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[20] Pål Halvorsen, Simen Sægrov, Asgeir Mortensen, David KC Kristensen, Alexander Eichhorn, Magnus Stenhaug, Stian Dahl, Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Carsten Griwodz, et al. Bagadus: an integrated system for arena sports analytics: a soccer case study. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 48–59, 2013.

[21] Gunnar Hartvigsen and Dag Johansen. Stormcast—a distributed artificial intelligence application for severe storm forecasting. *IFAC Proceedings Volumes*, 21(12):99–102, 1988.

[22] Gunnar Hartvigsen and Dag Johansen. Co-operation in a distributed artificial intelligence environment—the stormcast application. *Engineering Applications of Artificial Intelligence*, 3(3):229–237, 1990.

[23] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. Soccer field localization from a single image. *arXiv preprint arXiv:1604.02715*, 2016.

[24] Inc. Agile Sports Technologies. Hudl. `https://www.hudl.com/`, 2007–2022. [Accessed on Dec. 12, 2023].

[25] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, January 2023.

[26] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[27] Dag Johansen, Magnus Stenhaug, Roger BA Hansen, Agnar Christensen, and Per-Mathias Høgmo. Muithu: Smaller footprint, potentially larger imprint. In *Seventh International Conference on Digital Information Management (ICDIM 2012)*, pages 205–214. IEEE, 2012.

[28] Sebastian Lyng Johansen. Dárkon. Master's thesis, UiT The Arctic University of Norway, 2023.

[29] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[30] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58, 2001.

[31] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011.

[32] Michael Lewis. *Moneyball: The art of winning an unfair game*. W.W. Norton, 2003.

[33] Cuiyin Liu, Jishang Xu, and Feng Wang. A review of keypoints' detection and feature description in image registration. *Scientific programming*, 2021:1–25, 2021.

[34] Tao Liu, Peiliang Li, Haoyang Liu, Xiwen Deng, Hui Liu, and Fangguo Zhai. Multi-class fish stock statistics technology based on object classification

and tracking algorithm. *Ecological Informatics*, 63:101240, 2021.

[35] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[36] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[37] Krista Merry and Pete Bettinger. Smartphone gps accuracy study in an urban environment. *PloS one*, 14(7):e0219890, 2019.

[38] Noorkhokhar. Yolov8-football. `https://github.com/noorkhokhar99/YOLOv8-football`, 2023.

[39] Fredrik Stenvoll Nylund. Guorrat. Master's thesis, UiT The Arctic University of Norway, 2024. Will be submitted for review, May 22, 2024.

[40] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*, pages 305–319, 2014.

[41] Yash Pandya, Kaustav Nandy, and Shivam Agarwal. Homography based player identification in live sports. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5208–5217, 2023.

[42] Ceyhun Necati Pehlivan. The eu artificial intelligence (ai) act: An introduction. *Global Privacy Law Review*, 5(1), 2024.

[43] Liang Peng. Pixel2field: Single image transformation to physical field of sports videos. In *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I 14*, pages 661–669. Springer, 2019.

[44] Svein A Pettersen, Håvard D Johansen, Ivan AM Baptista, Pål Halvorsen, and Dag Johansen. Quantified soccer using positional data: A case study. *Frontiers in physiology*, 9:314255, 2018.

[45] Svein Arne Pettersen, Dag Johansen, Håvard Johansen, Vegard Berg-Johansen, Vamsidhar Reddy Gaddam, Asgeir Mortensen, Ragnar Langseth, Carsten Griwodz, Håkon Kvale Stensland, and Pål Halvorsen. Soccer video and player position dataset. In *Proceedings of the 5th ACM multimedia systems conference*, pages 18–23, 2014.

[46] Ashwin A Phatak, Franz-Georg Wieland, Kartik Vempala, Frederik Volk-

mar, and Daniel Memmert. Artificial intelligence based body sensor network framework—narrative review: proposing an end-to-end framework using wearable sensors, real-time location systems and artificial intelligence/machine learning algorithms for data collection, data mining and knowledge discovery in sports and healthcare. *Sports Medicine-Open*, 7(1):79, 2021.

[47] Akshay Rangesh and Mohan Manubhai Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars. *IEEE Transactions on Intelligent Vehicles*, 4(4):588–599, 2019.

[48] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[49] Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841, 2020.

[50] Roboflow. Roboflow [Software]. `https://roboflow.com`, 2024.

[51] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. Orb: An efficient alternative to sift or surf. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool, editors, *ICCV*, pages 2564–2571. IEEE Computer Society, 2011.

[52] Mehdi Houshmand Sarkhoosh, Sayed Mohammad Majidi Dorcheh, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, Dag Johansen, Michael A Riegler, and Pål Halvorsen. Ai-based cropping of soccer videos for different social media representations. In *International Conference on Multimedia Modeling*, pages 279–287. Springer, 2024.

[53] Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Thomas Torjusen, and Pål Halvorsen. The soccersum dataset for automated detection, segmentation, and tracking of objects on the soccer pitch. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 353–359, 2024.

[54] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[55] Bruno Silva, Zhibo Pang, Johan Åkerberg, Jonas Neander, and Gerhard

Hancke. Experimental study of uwb-based high precision localization for industrial applications. In *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*, pages 280–285, 2014.

[56] Rory Smith. *Expected Goals: The story of how data conquered football and changed the game forever*. HarperCollins Publishers, 2022.

[57] Tony Sprangers. The effect of hudl's league-wide deal with norwegian football. Available at: `https://www.hudl.com/blog/the-effect-of-hudls-league-wide-deal-with-norwegian-football`.

[58] STATSports Group Limited. APEX Athlete Series - GPS Performance Tracker. Available at: `https://statsports.com/`, 2023. [Accessed: 11 December 2023].

[59] Alexander Torkelsen. Mearka-architecting and evaluation of a sports video tagging software toolkit. Master's thesis, UiT The Arctic University of Norway, 2023.

[60] Ba Tran. superpoint-superglue-deployment 0.0.3. `https://pypi.org/project/superpoint-superglue-deployment/`, 2023.

[61] VICON Motion Systems Ltd. VICON Motion Capture Systems. `https://www.vicon.com`, [Accessed 2024].

[62] Fei Wang, Lifeng Sun, Bo Yang, and Shiqiang Yang. Fast arc detection algorithm for play field registration in soccer video mining. In *2006 IEEE International conference on systems, man and cybernetics*, volume 6, pages 4932–4936. IEEE, 2006.

[63] Tim Woinoski and Ivan V Bajic. Towards automated key-point detection in images with partial pool view. In *Proceedings of the 5th International ACM Workshop on Multimedia Content Analysis in Sports*, pages 9–17, 2022.

[64] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[65] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer, 2022.

[66] Saurabh Zunke and Veronica D'Souza. Json vs xml: A comparative

performance analysis of data exchange formats. *IJCSN International Journal of Computer Science and Network*, 3(4):257–261, 2014.

Appendix A

Paper to be submitted: **Towards Real-time Soccer Player Localization and Analytics**

Contributors: William Alexander Stimpson-Karlsson, Tor-Arne Schmidt Nordmo, Martin Rypdal, Dag Johansen

Towards Real-time Soccer Player Localization and Analytics

To be submitted. *

William Alexander Stimpson-Karlsson
UiT The Arctic University of Norway
Tromsø

Tor-Arne Schmidt Nordmo
UiT The Arctic University of Norway
Tromsø

Martin Rypdal
UiT The Arctic University of Norway
Tromsø

Dag Johansen
UiT The Arctic University of Norway
Tromsø

Abstract

By harnessing real-time soccer analytics through advanced computer vision, coaches gain immediate access to strategic insights and player dynamics, dramatically enhancing decision-making capabilities beyond what traditional analytical methods offer. This paper presents a comprehensive exploration of the development and implementation of a computer vision system, designed for real-time player tracking and analytics in soccer. Throughout the iterative design process, requirements evolved, notably simplifying the problem to focus solely on detecting players and their teams after consultations with elite coaches. We utilize existing state-of-the-art detection and tracking models, along with keypoint detection and matching models to localize soccer players on the field and translate their positions to a 2D representation for analytical purposes. We compare different keypoint detection and feature matching algorithms, GPS suits vs. video-based detection, and we discuss the several trade-offs that allow us to achieve real-time localization of the players for assisting coaches on consumer-grade hardware, such as utilization of CPU vs GPU, cropping the frames, and caching previous results. The work in this paper lays the foundation for a digital twin coach, enabling coaches to make data-driven decisions with unprecedented precision and timeliness.

1. Introduction

More than a decade ago, we built a series of novel multi-camera array systems to cover an entire soccer field. Coupled with coach tagging apps and on-player wearable IoT devices connected with radio-positional technologies, we were able to gain training and game insights in real-time [3, BLINDED]. Select video snippets tagged in real-time by coaches on their phones were seconds later automatically fetched from within the video streams and ready for analysis and feedback purposes. Replay could happen, e.g., in the locker rooms during half-time match breaks or even during practice with videos displayed on the digital billboard screen next to the field. We were among the very first movers, if not the first, into what has become a multi-million dollar athlete player tracking and analytics industry. Our multimedia systems were in use by both a Norwegian premier elite soccer club and the national Norwegian A-team during practice and play.

Much has happened since. Numerous tracking and analytics companies are now providing similar or related solutions that are in daily operational use throughout the world of soccer. In-house analytics departments relying on these external data-driven providers are common in premier division clubs, and even lower level division clubs have their own analytics coaches deriving insights from training and match data captured during a week (micro cycle in coaching terms). This is not without problems. First, current tracking devices and videos captured amount to massive data volumes. Hence, finding relevant insights requires tedious and manual operations to identify actionable, soccer-relevant insights. Second, data to be analyzed is normally collected, stored, and managed by external companies that are less transparent. Their algorithms used are not public, so determining, e.g., accuracy in their analysis is difficult. Some of the tagging might be semi-automated with humans involved, which adds latency. A fourth problem is that an external enterprise creates a data lock-in situation where the clubs do not control and fully manage their own data, and the business model for the enterprises might potentially include sharing with other parties. Yet a potential problem is compliance and regulatory issues, like the EU General Data Protection Regulation (GDPR) and EU's recent AI Act. The

*Identify applicable funding agency here. If none, delete this.

list is longer.

Our early foray into the soccer domain was motivated by the need for insights based on quantitative data from training and match play. Our current focus is way more ambitious than obtaining such general objective insights like, e,g, a corner, offside, goal, and free kick. These are events that most general soccer veterans can identify, but probably not the most relevant for a head coach trying to manage and control his or her team while the match is unfolding. We are interested in context-sensitive, higher level insights that resemble specific actions and events relative to the team's game plan and opponent's style-of-play. Hence, we are attempting to derive coach-specific actionable insights in real-time for interventions by coaches while a game (or training practice) is unfolding. Our approach is to create a digital twin coach using artificial intelligence (AI). This is a challenging problem, and at the core of such a solution is obtaining accurate quantification data in real-time. Related to this is what specific type of data is necessary, not just neat to obtain.

The main contributions of this paper are as follows:

- Development of a real-time computer vision system for soccer player tracking and registration, emphasizing low-latency data processing on consumer-grade hardware. This is one component of the digital twin coach described below.
- Analysis of algorithmic trade-offs in real-time execution, including computational optimizations and data management strategies to enhance system performance.

2. Digital Twin Coach

The main functionality of an AI-based digital twin coach is to identify high-level patterns and context-specific trends that emerge during a match. This applies for both teams, and the idea is to identify their strengths and weaknesses similar to how an experienced coach works. One example of a context-relevant insight can be that there are open spaces between players in an opponent's defensive line, another is that an individual opponent is wrongly positioned relative to the other players of the team. One can consider this as anomalies relative to an expected behavior of a single individual or a group of players.

Our system architecture consists of a (1) data federation layer, which collects one or several video streams from remote cameras. Next, (2) we have a three-stage coordinate production pipeline. The net result of this pipeline is a (3) longitudinal on-field player location dataset and visualization user interface (soccer field).

Initially, we set out to identify each individual player on a soccer pitch, which had its challenges. This process is compute intensive and is not accurate enough during, e.g. occlusions. Our digital twin coach system is being developed with elite coaches closely involved, and when exposing such constraints to them, we got surprising feedback. During an intense match play, they primarily wanted positional data of players on both teams. As such, open spaces, available rooms for the ball to be passed into, players not accurately positioned with regard to the game plan, and similar were the actionable insight needed. If they wanted to identify individuals, video footage available would be more proper to use.

The sweet spot of design considerations, accuracy, and real-time requirements for this vital task will be further detailed in the following.

3. Coordinate production pipeline

Converting player frame coordinates to on-field coordinates is a four-stage process, starting with the source of the frame. Each frame is read from either a continuous stream source or a video source. Player detection, tracking, and classification are done using a custom YOLOv8 [9] model trained to recognise players on our team, [BLINDED], and their opponents. The tracking is handled by ByteTrack [19] which is a multi-object tracking model. The metadata produced from the YOLOv8 inference is piped to the frame translation component. Each frame received is located within a panorama built up of frames from the video or stream source. For our system we have relied on Hudl [8] for our videos. Once a frame has been correctly placed in the panorama using state-of-the-art key point detection, we compute the homography matrix to translate all player coordinates to where they are situated on the panorama image. Panorama-translated coordinates are then again translated to a 2D image of a soccer field using a pre-calculated homography matrix between keypoints in the panorama and the 2D image. For real-time coordinate computation, tweaking the interval of frames that are localized in the panorama is necessary. The sweet spot for both interval and image quality is detailed in section 4. The on-field player coordinates in between these intervals are computed using interpolation between previous and current coordinates. The on-field coordinates are then packed into segments from which a user can retrieve via requests.

From the system we want as accurate and rich (classification, bounding box, on-field coordinate and identification) metadata as possible while keeping real-time execution. Accurate metadata is needed so that the end-users of the system can feed their systems with correct metadata to compute playing space, possession, passing networks, packing rate and other soccer-relevant metrics. Real-time execution is critical for fast feedback for sideline analysis and coaches if the system is to be deployed in a real-world setting.

3.1. System prerequisites

The system will need some pre-requisites to function including a pre-made panoramic image of the recorded area and a detection model that can find the players and track them.

3.1.1 Panoramic image

A panoramic image composed of several images making up the total view of the camera source is required to be made or retrieved before the system can function. Hudl video sources used in our example are made up of 4 cameras that each record their own sections of the soccer field as shown in Fig. 1. At the time of writing, the panoramic camera mode was unavailable to us for usage in a high enough resolution to work with the system.

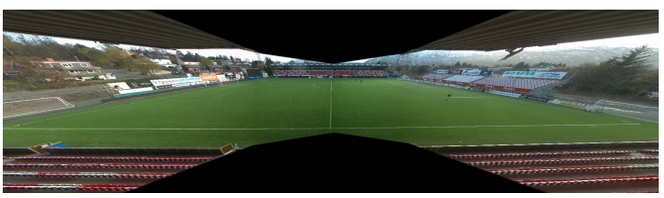


Figure 1. Panorama image [BLINDED] Arena (Hudl).

Panoramic image creation is supported by the system if the user can provide 3 or 5 images that roughly encapsulate the entire soccer field. This process is visualised in Fig. 2.

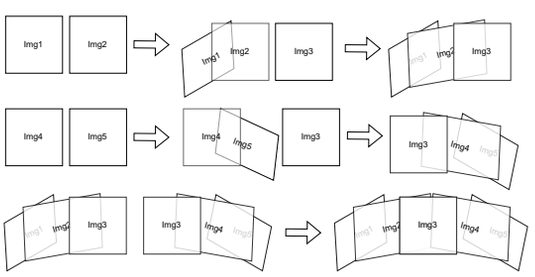


Figure 2. Panoramic image creation process.

The stitching process consists of 4 steps:

- Detect and match the features of the image. A feature can be explained as a unique property of some image such as textures, colors or shapes. Some examples of features that are found in the images encapsulating the soccer field are commercials, seating areas, stadium pillars, and stadium equipment.
- Estimate homography matrix using the features that were found in previous step.
- Warp first image to align with second image.
- Blend the warped image together with second image.

Figure 3 shows a panoramic image created from 5 images of SR-Bank Arena (Norwegian soccer team Viking's home arena) using the systems functionality. The frames are all taken from a video recorded by Hudl.

Figure 3. Result from panoramic stitching process.

Computing the homography matrix between the panoramic image and the 2D soccer field is achieved by mapping areas of the soccer field in the panorama with the corresponding areas in the 2D soccer field. The areas used are highlighted in red in both Figs. 3 and 4.

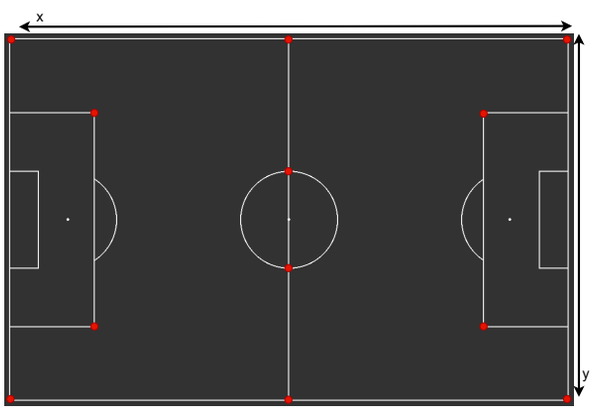


Figure 4. 2D soccer field image used for coordinate illustrations.

3.1.2 Detection model and tracker

An object detection model to detect players and, optionally, the ball is needed to produce the inference data from the video or stream source. For this system, we have trained a custom YOLOv8 model which can detect the players on each of the teams. A general player detection model would also be suitable and has been used for a large part of the project duration. The decision to switch to a new custom trained model was made to deal with a classification and occlusion problem which is discussed in Sec. 5. The detection model should also be coupled with a MOT (multiple object

tracker) model. ByteTrack and BotSort [1] are good alternatives and are easy to integrate with the Ultralytics YOLOv8 framework [9]. A MOT is necessary to calculate interpolation coordinates for players in between frames when the system runs in real-time.

3.2. Step 1: Run inference

The initial step of the system is to load in the trained model coupled with a MOT and pass in the video or stream source to the model. A client socket is created to pass produced data from the video/stream source and frames from the video to the translation component using the Python multiprocessing module. Player bounding boxes, classifications, and tracking ID produced by the model is sent to the transformation component every 30th frame. The information sent is a package consisting of information for all of the 30 frames in that interval. The last frame in the interval is also sent as a `NumPy` array. A decrease in frame intervals leads to lower throughput, but an increase in accuracy. Sending a message is an expensive operation in terms of time because the inference comes to a stop during this process. An increase in frame intervals would result in higher speeds, but would lead to lower interpolation accuracy between translated coordinates in step 2.

3.3. Step 2: Generate homography matrix for coordinate translation

The keypoint-based homography generator initialises with a metadata message received from the detection component containing setting parameters. The setting parameters contain what frame interval should be used for packing produced metadata into segments in step 3 and what classes are detected by the model. Finally, thesuperglue/superpoint [4, 16] module is set up and loaded, ready for receiving images for comparison from the detection and tracker component. The panorama image is decreased to specified resolution depending on configuration and is split into three equal, smaller parts (left, center, and right). This split is performed because of a constraint regarding image size at the superglue/superpoint module; this however proved beneficial in terms of performance which will be shown in Sec. 4. The transformation component receives metadata from the inference component once at an interval of 30 frames or whichever frame interval is specified and receives an image to calculate a homography matrix between it and the panoramic image. The process for computing the homography matrix between a frame and its placement in the panorama works as follows:

- Frame is resized to a configured percentage of original image (same percentage as the panorama is decreased).
- Frame and panorama part are fed to the superglue/superpoint module.
- Frame and cached panorama part used in previous homography calculation are compared.
- Continue if enough common keypoints are found with cached part, otherwise check next part. When enough keypoints are found between the frame and a part, that part (either left, center or right) is cached.
- Common keypoints are used to calculate homography matrix.

3.3.1 Step 3: Translate coordinates

The coordinate translation component uses the produced homography matrix from the keypoint-based homography matrix generator component to translate player coordinates from the video source to their respective 2D soccer field positions. The translation works as follows:

- Player coordinate is translated from frame coordinate in video to panoramic image coordinate.
- Player coordinate is translated from panoramic image coordinate to coordinate in the 2D soccer field.

Producing interpolation coordinates for the frames in between the intervals is done in the metadata aggregator component.

3.4. Step 4: Produce metadata segments

The metadata aggregator is the last component of the system and is responsible for packing all the necessary metadata. Because coordinates are only translated once every 30 frames, we use interpolation to estimate coordinates in between the intervals. To compute interpolation coordinates we need a previous and a current player coordinate. We therefore need a method to identify which coordinate belongs to which player, and this is where the tracker producing player IDs comes into play. As long as an ID is present in the current and previous translated frame, interpolation coordinates can be computed for that interval. If an ID was present in the previous frame but is absent in the current, interpolation for that ID does not occur. Interpolation between given coordinates (x_1, y_1) and (x_2, y_2), and the desired number of interpolated points n, is expressed as:

Point	(x, y)
1	(x_1, y_1)
2	$\left(x_1 + \frac{1}{n+1}(x_2 - x_1), y_1 + \frac{1}{n+1}(y_2 - y_1)\right)$
3	$\left(x_1 + \frac{2}{n+1}(x_2 - x_1), y_1 + \frac{2}{n+1}(y_2 - y_1)\right)$
$\vdots$	$\vdots$
k	$\left(x_1 + \frac{k-1}{n+1}(x_2 - x_1), y_1 + \frac{k-1}{n+1}(y_2 - y_1)\right)$
$\vdots$	$\vdots$
$n+1$	(x_2, y_2)

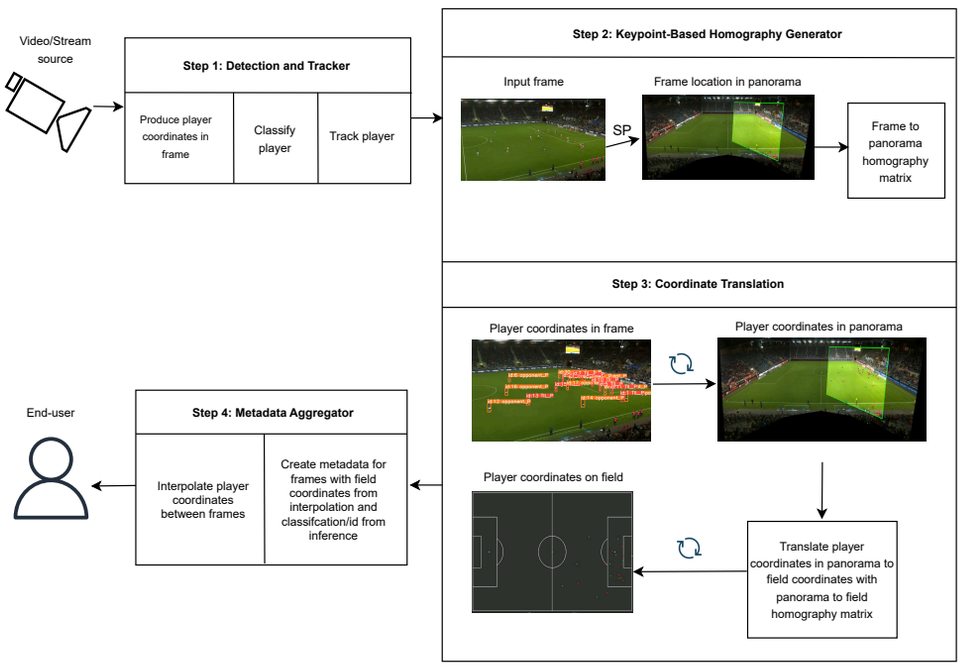


Figure 5. Metadata production pipeline.

Player on-field coordinates, in-frame bounding box, ID, and classification are packed into their corresponding frames using the given format shown in listing 1. The classification or "team" option is configurable and depends on the classifications present in the model. The system is designed for both videos and stream as input. Depending on the use case of the system, metadata for multiple frames can be packed into segments so that they can be synced together with a stream source. For example, if we have a `HLS` (HTTP Live Streaming) stream that manifests a 2 second duration for each segment then the metadata can be packed so that there are frame metadata for 60 frames. Each `json` file would then contain frame metadata for a given interval such as frame 61-120 or 121-180. Such a use-case of the system could be real-time analysis of what is happening on the field during a soccer match. Post-game analysis using a finished, recorded match could be a second use case of the system. Instead of using a stream source as input, a video could be used. For that case, instead of the metadata being segmented, it would then be one `json` file containing the metadata.

```
{
  "frames": [
    {
      "frame": 32,
      "detections": [
        {
          "field_coordinate": {
            "x": 360,
            "y": 168
          },
          "camera_coordinate": {
            "x1": 522,
            "y1": 201,
            "x2": 541,
            "y2": 260
          },
          "id": 6,
          "team": "0"
        },
        ...
      ]
    },
    ...
  ]
}
```

Figure 6. Metadata structure.

4. Experiments and Results

A series of design choices have been made with the requirement of real-time execution without sacrificing precision when translating between player coordinates on camera and their actual field coordinates.

4.1. Dataset

The dataset is composed of 2,720 annotated frames separated into train and test sets. The bounding box of all the players in each frame is retrieved by using an existing player detection model [12]. Individual classification for every bounding box contained in the 2,720 frames is done manu-

ally. For our testing and evaluation of the system we have trained the model to recognise 4 different classes which are home team keeper and player, and away team keeper and player.

4.2. Hardware

The following hardware components are used for conducting the experiments:

- **Graphics Processing Unit (GPU):** NVIDIA GeForce RTX 3070 8GB. This GPU is a high-performance consumer-grade GPU suitable for demanding graphics-intensive tasks like machine learning applications.
- **Random Access Memory (RAM):** 128 GB RAM.
- **Processor (CPU):** 13th Gen Intel(R) Core(TM) i7-13700 2.10 GHz.

4.3. Model and tracker

At the time of implementing the system, YOLOv8 [9] object detection models were among the fastest and most accurate while being easy to implement, customise and run on conventional hardware. Quick integration with MOTs using the Ultralytics framework [9] made it a solid choice, meeting our requirements. Choosing what tracker to use came down to what tracker would be able to function in real-time while keeping tracking consistent over the course of the video. ByteTrack and BotSort are both available through the Ultralytics framework. For our 1080p 30fps video source, ByteTrack achieved an average inference time of 27.14ms per frame while BotSort achieved an average time of 39.81ms. Thus we chose the first as our tracker.

4.4. Keypoint detection and feature matching

Deciding on the algorithm to use for finding keypoints between the video frames and the panoramic image was based on the following requirements:

- Accuracy - Matching keypoints between frame and panorama should result in a homography matrix that translates coordinates with high accuracy. A homography matrix of low accuracy would result, if used to warp an image, in something similar as Fig. 7.
- Speed - Detecting, matching and computing homography matrix based on matching keypoints should be as fast as possible to meet real-time goals.

Pixel spread between computed coordinates in sequential frames gives a good indication on the level of accuracy the algorithms possess for our usage. A player coordinate in frame 2 being far away from that same players coordinate in frame 1 indicates that computed homography matrix based on the matching keypoints found was inaccurate. Fig. 8 demonstrates this statement. Table 6 shows our results from testing different algorithms. SuperPoint (SP) had the lowest spread compared to the other algorithms.

Figure 7. Warped image using homography with low accuracy.

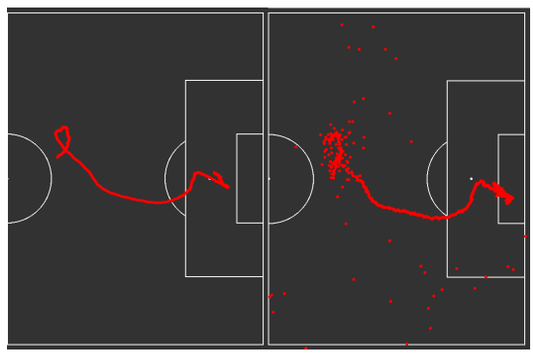


Figure 8. Visualisation of GPS (left) and SIFT (right) player coordinates.

Moving the SuperPoint execution from CPU to GPU improved performance close to a factor of 5. Further, the SuperPoint implementation was through multiple iterations improved by adjusting the original panoramic image by decreasing resolution of both it and the frame from the video, and cropping out around 50 percent of the bottom half of the original image. As detailed in Sec. 3.3 due to an image size constraint for SuperPoint, the panoramic image was split into three parts. This however proved beneficial as we could now take advantage of caching principles, further improving the speed. A hypothesis that players and live-commercials from the panoramic image could interfere with the accuracy of the computed homography was investigated. However, at 55 percentage of original resolution and at 100 percent with both players and commercials removed

Method	GPU	Crop	Cache	Through-put (fps)	Spread (px)
BRISK [10]	✓	✓	✓	2.18	1492.39
ORB [5]	✓	✓	✓	24.62	28852.98
SIFT [11]	✓	✓	✓	1.48	149.12
GPS	N/A	N/A	N/A	N/A	2.18
SP	✓	✓	✓	0.45	1.77

Table 1. Keypoint detection and feature matching algorithms.

(see Fig. 9), no benefit was observed in both fps and spread (see Tab. 3).

Figure 9. Panorama with players and commercials removed.

Testing showed spread or accuracy after a decrease to 55-50 percent of original resolution started to get inaccurate. Any lower than this percentage resulted in missing out on some relevant positional information on a player. Fig. 10 shows the plotting of produced coordinates at 20 and 40 percent of original resolution.

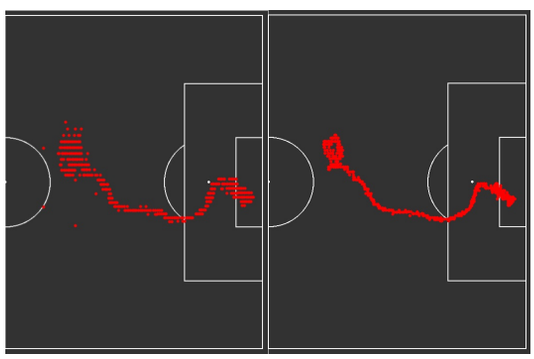


Figure 10. Visualisation of player coordinates at 20 and 40 percent of original resolution.

Interval	Throughput (fps)	Real-time
30	37.17	YES
25	37.14	YES
20	36.92	YES
15	36.65	YES
10	36.06	YES
5	33.36	YES
2	28.64	NO
1	22.79	NO

Table 2. Inference speed per frame at different intervals.

4.5. Coordinate translation interval

Decreasing the interval for which the inference component sends packages to the coordinate translation component, increases the overall time of inference. By increasing the interval, more frames can be located in the panorama, decreasing the interval for which the system has to compute interpolation coordinates. This increases the accuracy of a players position, however it increases the time it takes. Tab. 2 shows the different processing times per frame at different intervals.

4.6. Real-time performance on single node

Running both the Yolov8 detection model and SuperPoint on the same computer results in a lower throughput in both components of the pipeline because of higher workload. Using the hardware specified in Sec. 4.2 and a video with 30 fps we achieved real-time performance for case 2, 4, and 6 in Tab. 3. Fig. 11 shows the coordinates produced for these cases.

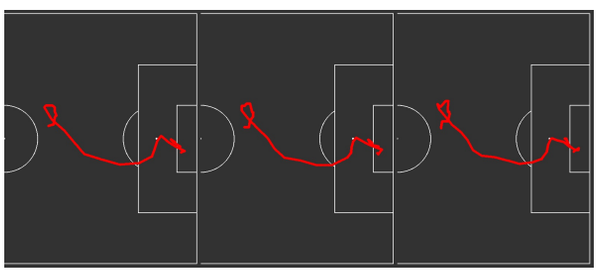


Figure 11. From the left: Case 2, 4 and 6.

5. Related Work

Computing homography matrices using keypoints detection on specific keypoints or features known to exist in the frame has been done for several sports [6, 7, 17, 18]. Similar to these articles, we compute a homography matrix from keypoints that match between a frame and a corresponding reference area (panorama for us). We however use SuperPoint for image comparison between two images, a frame and a panorama made up of several frames to detect features and common keypoints.

Akan and Varlı [2] highlight some of the challenges related to occlusions using deep learning models and trackers to find players. Due to to the nature of the game and the angle of the camera, players often run parallel to one another making the MOT lose players for several frames before re-tracking them with a new ID. When a player has multiple IDs over the course of a video or stream, calculating interpolation coordinates for that player in frames between such ID switches is challenging. Currently, we are working on a semi-automatic tool that can assist a user with connecting such IDs to a player by comparing crops of the player before an ID was lost and compare those crops with new IDs in the area. As is also pointed out by Akan and Varlı [2], ball detection is difficult because of the size of the ball in a frame. Hudl cameras, which has been the main source of footage for our experiments are located far away from the field with a limited zoom because its main purpose is to capture all of the players. SmartCrop [15] has had great results with ball

GPU	Crop	Cache	Percentage	Players / Commercials	Throughput (fps)	Spread (px)
X	✓	X	100	✓	0.037	N/A
✓	X	X	100	✓	0.21	N/A
✓	✓	X	100	✓	0.45	N/A
✓	✓	✓	100	X	0.47	1.88
✓	✓	✓	100	✓	0.45	1.77
✓	✓	✓	80	✓	1.55	2.11
✓	✓	✓	70	✓	2.45	2.36
✓	✓	✓	60	✓	3.71	2.16
✓	✓	✓	55	✓	4.62	2.14
✓	✓	✓	55	X	4.53	2.38
✓	✓	✓	50	✓	5.52	2.52
✓	✓	✓	40	✓	8.98	3.04
✓	✓	✓	30	✓	11.47	4.78
✓	✓	✓	20	✓	12.65	14.90

Table 3. Iterations of SuperPoint implementation.

Case	Interval	Percentage	Throughput (fps)	Real Time
1	30	55	27.85	NO
2	30	40	31.97	YES
3	25	50	28.34	NO
4	25	40	31.58	YES
5	20	45	28.83	NO
6	20	35	31.46	YES
7	15	40	28.64	NO
8	10	35	26.48	NO

Table 4. Fps at different interval and percentages on single node.

detection using a combination of object detection and interpolation in broadcasting footage. A combination of Hudl footage to detect the players and broadcasting footage to detect the ball combined with the technology introduced by SmartCrop [15] could be an interesting system to explore further.

GPS vests are commonly used for gathering both health related data and player position during a match using real-time location systems (RLTS) [14]. In a complementary approach, Pandya et al. [13] leverage RFID sensor data for player identification, enhancing field registration accuracy and reducing latency. These positional technologies have the advantage of not having to deal with occlusions because positional data is collected by receivers that are not affected by players running in front of one another. However, the positions received are only for own team players and not any from the opposing teams. Positional data of these opponent players would be very advantageous in an analytics context. With a camera based system such as ours this problem is overcome as video stream alternatives such as broadcasting, Hudl, or a personal camera is often available.

6. Conclusion

In conclusion, this paper introduces a computer vision system that marks a significant step forward in soccer analytics, and a foundation for a digital twin coach. By integrating advanced detection and tracking models with keypoint detection and matching, it enables real-time player tracking and analysis. The system's ability to localize players and translate their movements into actionable insights offers coaches new strategies based on immediate data. Throughout the development process, requirements evolved, notably simplifying the problem to focus solely on detecting players and their teams after consultations with elite coaches. The exploration of trade-offs such as CPU versus GPU utilization, frame cropping, result caching, and resolution scaling has been crucial in achieving real-time performance.

Future work will focus on extending the system to include soccer ball detection, which presents distinct challenges due to the ball's small size and fast movement. Additionally, applying this technology to television broadcast videos, which vary in camera angles and movements, introduces complexities in maintaining consistent tracking accuracy. Addressing these areas will enhance the system's utility and provide a more detailed analysis of game dynamics.

References

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Botsort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022. 4

[2] Sara Akan and Songül Varlı. Use of deep learning in soccer videos analysis: survey. *Multimedia Systems*, 29(3):897–915, 2023. 7

[3] Blinded. Blinded. 1

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 4

[5] Rublee Ethan. Orb: An efficient alternative to sift or surf. *ICCV, 2011*, 2011. 6

[6] Dirk Farin, Susanne Krabbe, Wolfgang Effelsberg, et al. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, pages 80–91. SPIE, 2003. 7

[7] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. Soccer field localization from a single image. *arXiv preprint arXiv:1604.02715*, 2016. 7

[8] Agile Sports Technologies Inc. Hudl. https://www.hudl.com. Accessed: 2024-03-15. 2

[9] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, 2023. 2, 4, 6

[10] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011. 6

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 6

[12] noorkhokhar99. Yolov8-football. https://github.com/noorkhokhar99/YOLOv8-football, 2023. 5

[13] Yash Pandya, Kaustav Nandy, and Shivam Agarwal. Homography based player identification in live sports. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5208–5217, 2023. 8

[14] A. Wieland F. G. Vempala K. Volkmar F. Memmert D. Phatak, A. Artificial intelligence based body sensor network framework—narrative review: Proposing an end-to-end framework using wearable sensors, real-time location systems and artificial intelligence/machine learning algorithms for data collection, data mining and knowledge discovery in sports and healthcare. *???*, 2021. 8

[15] H. Dorcheh S. M. M. Midoglu C. Sabet S. S. Kupka T. Johansen D. Riegler M. A. Halvorsen P. Sarkhoosh, M. Ai-based cropping of soccer videos for different social media representations. *???*, 2024. 7, 8

[16] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 4

[17] Fei Wang, Lifeng Sun, Bo Yang, and Shiqiang Yang. Fast arc detection algorithm for play field registration in soccer video mining. In *2006 IEEE International conference on systems, man and cybernetics*, pages 4932–4936. IEEE, 2006. 7

[18] Bajic I. V. Woinoski, T. Towards automated key-point detection in images with partial pool view. *???*, 2022. 7

[19] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. 2022. 2